Collective Machine Learning:
Team Learning and Classification in Multi-Agent Systems

by
Christopher M. Gifford
University of Kansas, 2009

Submitted to the graduate degree program in Computer Science and the
Graduate Faculty of the University of Kansas
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy.

_____
Dr. Arvin Agah (Chairperson)

_____
Dr. Swapan Chakrabarti

_____
Dr. Brian Potetz

_____
Dr. Sarah Seguin

_____
Dr. Leigh Stearns

Date Defended: _____

The Dissertation Committee for Christopher M. Gifford certifies
that this is the approved version of the following dissertation:

Collective Machine Learning:
Team Learning and Classification in Multi-Agent Systems

Committee:

Dr. Arvin Agah (Chairperson)

Dr. Swapan Chakrabarti

Dr. Brian Potetz

Dr. Sarah Seguin

Dr. Leigh Stearns

Date Approved: _____

Abstract

This dissertation focuses on the collaboration of multiple heterogeneous, intelligent agents (hardware or software) which collaborate to learn a task and are capable of sharing knowledge. The concept of collaborative learning in multi-agent and multi-robot systems is largely under studied, and represents an area where further research is needed to gain a deeper understanding of team learning. This work presents experimental results which illustrate the importance of heterogeneous teams of collaborative learning agents, as well as outlines heuristics which govern successful construction of teams of classifiers.

A number of application domains are studied in this dissertation. One approach is focused on the effects of sharing knowledge and collaboration of multiple heterogeneous, intelligent agents (hardware or software) which work together to learn a task. As each agent employs a different machine learning technique, the system consists of multiple knowledge sources and their respective heterogeneous knowledge representations. Collaboration between agents involves sharing knowledge to both speed up team learning, as well as to refine the team's overall performance and group behavior. Experiments have been performed that vary the team composition in terms of machine learning algorithms, learning strategies employed by the agents, and sharing frequency for a predator-prey cooperative pursuit task. For lifelong learning, heterogeneous learning teams were more successful compared to homogeneous learning counterparts. Interestingly, sharing increased the learning rate, but sharing with higher frequency showed diminishing results. Lastly, knowledge conflicts are reduced over time, as more sharing takes place. These results support further investigation of the merits of heterogeneous learning.

This dissertation also focuses on discovering heuristics for constructing successful teams of heterogeneous classifiers, including many aspects of team learning and collaboration. In one application, multi-agent machine learning and classifier combination are utilized to learn rock facies sequences from wireline well log data. Gas and oil reservoirs have been the focus of modeling efforts for many years as an attempt to locate zones with high volumes. Certain subsurface layers and layer sequences, such as those containing shale, are known to be impermeable to gas and/or liquid. Oil and natural gas then become trapped by these layers, making it possible to drill wells to reach the supply, and extract for use. The drilling of these wells, however, is costly. Here, the focus is on how to construct a successful set of

classifiers, which periodically collaborate, to increase the classification accuracy. Utilizing multiple, heterogeneous collaborative learning agents is shown to be successful for this classification problem. We were able to obtain 84.5% absolute accuracy using the Multi-Agent Collaborative Learning Architecture, an improvement of about 6.5% over the best results achieved by Kansas Geological Survey with the same data set. Several heuristics are presented for constructing teams of multiple collaborative classifiers for predicting rock facies.

Another application utilizes multi-agent machine learning and classifier combination to learn water presence using airborne polar radar data acquired from Greenland in 1999 and 2007. Ground and airborne depth-soundings of the Greenland and Antarctic ice sheets have been used for many years to determine characteristics such as ice thickness, subglacial topography, and mass balance of large bodies of ice. Ice coring efforts have supported these radar data to provide ground truth for validation of the state (wet or frozen) of the interface between the bottom of the ice sheet and the underlying bedrock. Subglacial state governs the friction, flow speed, transport of material, and overall change of the ice sheet. In this dissertation, we focus on how to construct a successful set of classifiers which periodically collaborate to increase classification accuracy. The underlying method results in radar independence, allowing model transfer from 1999 to 2007 to produce water presence maps of the Greenland ice sheet with differing radars. We were able to obtain 86% accuracy using the Multi-Agent Collaborative Learning Architecture with this data set. Utilizing multiple, heterogeneous collaborative learning agents is shown to be successful for this classification problem as well. Several heuristics, some of which agree with those found in the other applications, are presented for constructing teams of multiple collaborative classifiers for predicting subglacial water presence.

General findings from these different experiments suggest that constructing a team of classifiers using a heterogeneous mixture of homogeneous teams is preferred. Larger teams generally perform better, as decisions from multiple learners can be combined to arrive at a consensus decision. Employing heterogeneous learning algorithms integrates different error models to arrive at higher accuracy classification from complementary knowledge bases. Collaboration, although not found to be universally useful, offers certain team configurations an advantage. Collaboration with low to medium frequency was found to be beneficial, while high frequency collaboration was found to be detrimental to team classification accuracy. Full mode learning, where each learner receives the entire training set for the learning phase, consistently outperforms independent mode learning, where the training set is distributed to all learners in a team in a non-overlapping fashion. Results presented in this dissertation support the application of multi-agent machine learning and collaboration to current challenging, real-world classification problems.

Acknowledgements

I would like to specially thank Professor Arvin Agah for being my academic advisor and committee chair. His support and guidance throughout this process made this work possible. I would also like to thank Professors Swapan Chakrabarti, Brian Potetz, Sarah Seguin, and Leigh Stearns for serving on my doctoral committee.

Most of all, I would like to thank my wife, Jaclyn, for her undying support and patience throughout my college career. She kept me focused, listened even when nothing made sense, and encouraged me to reach for my goals. Without her, this dissertation would not have been possible. Special thanks also goes out to my parents, Mike and Dale, the entire Gifford family, my sister Sarah, the Herrick family, the entire Tillman family, and my friends for all their support. I am greatly appreciative for all of the help, advice, and support provided by the faculty, staff, and students at the Center for Remote Sensing of Ice Sheets and the School of Engineering at the University of Kansas.

A portion of this material is based upon work supported by the National Science Foundation under Grant No. ANT-0424589. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"Teamwork is the ability to work together toward a common vision. The ability to direct individual accomplishments toward organizational objectives. It is the fuel that allows common people to attain uncommon results." –Andrew Carnegie

1.1 Learning in Humans

In education, sequences of collaborative knowledge transfer, where students take on both teacher and learner roles in the classroom, have been reported to improve student learning, communication, and teamwork skills [56]. Different levels of expertise are also reported to affect the form and progress of interactions between individuals. These asymmetries in knowledge can be very beneficial in that they can spark more successful cooperative learning, which involves the gaining and sharing of expertise among various learning individuals. Collaborative learning can therefore be an effective approach for teams consisting of heterogeneous learners, where both teacher and student roles can be assumed by learners (agents) during the life-long learning process.

The intuition our research follows is that people and therefore learners are different from one another. Different learning styles can be utilized such that learners can adequately learn. However, different learners benefit differently from the same material. Visual learners may learn very well from a picture of a process flow, whereas other learning styles may not benefit as greatly. Learning styles depend on the individuals, not necessarily their field of expertise. Some are more flexible and versatile in terms of learning than others. Also, human learners refine how they learn (i.e., learn how to learn) over time. Thus, material can be better learned by combining the views of several learning methods for increased benefit, compared to using a single learning method.

A *Learning Style* can be defined as a strategy or mental behavior which an individual applies to learning based on his or her underlying likes, dislikes, and potential. How individuals problem-solve, as well as their prior knowledge, may also affect their learning.

Individuals also vary considerably in their general ability at learning. This variance of stable learning ability between individuals is typically termed the *aptitude* of the individual. Thus, different learners can learn differently from not only different material, but the same material as well. This concept is the catalyst for our research, which is geared toward a heterogeneous, collaborative team of learning individuals (agents).

Studies in education have shown that having heterogeneous abilities in a cooperative learning team should be the primary concern [48]. Having teams entirely composed of poor-performing students will result in low instructional benefit, whereas composing a team entirely of strong students will likely distribute the work rather than spark participation in group discussions, which can be instructionally beneficial. This creates a fine line for proper construction of a team in both human and agent environments.

This work is also related to Gardner's theory of multiple intelligences, stating that various types of intelligence exist in humans [52]. This theory also suggests that each individual has a unique *cognitive profile*, which comes from a definition of intelligence that is more broad than the traditional definition (which Gardner argues is too focused). Gardner gives an example in which one child easily masters a multiplication table, but another struggles to do so. This theory explains the example by suggesting that the child that struggled may be stronger in another *kind* of intelligence, and that the child who mastered the task is not necessarily more intelligent *overall*.

Such a theory may also explain why some individuals are very good at some things while not good at others. Activities likely involve several different kinds of intelligence to (efficiently or sufficiently) be completed. It may also be that different kinds of intelligence are more important than others, or that all are equally important. This theory can also relate to our research involving learning agents, where certain learning methods are combined and knowledge shared so that each individual, and the team as a whole, are more intelligent (based on knowledge from several different kinds of learners).

Human studies have shown that collaborative learning, especially learning together, is highly beneficial; outperforming individual learning and competitive learning in nearly all cases [65]. Collaborative learning can thus be thought of as shared thinking or shared understanding, where individual skills are transformed into collective capabilities. Cooperative and collaborative learning, however, cannot happen efficiently without forming a successful team.

1.2 Team Building and Success

Cooperation can be defined as working together to accomplish a common goal. Extending this definition, cooperative learning can be defined as the instructional use of teams of individuals that act to benefit both their own utility and that of the group. Cooperative learning groups can introduce the interesting dynamic of the desire to have all members

of the team maximize their learning to optimize the performance of the entire group. Cooperative learning groups also rely on the use of positive interdependence, which strives to agree on answers/solutions to strategies and fulfilling individual and team responsibilities [117]. Higher achievement and greater productivity typically result as well.

Would constructing a team of strong individuals be the best way to form a team? Having strong individuals in a team is not enough, as teams must be organized in such a way to foster productive cooperation with one another [109]. Teams must also focus on dividing tasks for individual attention, or work on tasks as a collaborative unit. Some tasks can be shared, whereas others can be done individually. For example, in a predator-prey scenario, each predator can perform the task of getting close to the prey, but the team must share the task of capturing/surrounding the prey (which an individual cannot do alone). Teams can then develop a team identity and strategy that belongs to the whole, rather than one or more individuals.

It is important to investigate how teams might fit within a specific strategy or application, and how they may offer certain benefits. It must be clear how the team is expected to help or improve performance for the task. Some therefore state that "it's not about teams; it's about performance" [61]. It is also stated that in professional teams, performance can be improved by not changing the members, but by changing the organizational design. Only some of performance is due to members.

Team-based organizations typically involve balanced roles. Some overly focus on teams that they forget the entire point of having a team is for collaboration between independent parties [61]. Bringing individuals together to collaborate in many different ways and for many different purposes is highly beneficial. Teams also contribute individual differences, bringing unique talents, skills, and interests. By harnessing each of these differences, the team receives the best from each member rather than the best or worst of a single individual. Together, they can combine their potential to achieve greater potential.

There are some differences between teams and their focus. Self-directed teams work together full-time and are typically of high-performance. Problem-solving teams on the other hand will likely meet less often until their problem is solved, then disband. Finally, project teams typically have different levels of involvement for members which typically work on entirely different tasks. This makes sharing leadership, progress, and knowledge more challenging. As such, it is recommended that people work together and do not focus on restricted or specific forms of collaboration [61]. Collaboration should be tailored around the team members and support the purpose of their task.

There are also other variations of teams, such as task forces, which are mostly composed of several highly-specialized individuals to solve complex problems. These teams are typically short-lived. Long-term teams participate in long-term learning, allowing knowledge of the team to be refined, corrected, and advanced. The result of long-term teams is usually a valuable tool that is difficult to achieve in a short time span.

Rather than trying to select a team for specific purposes, as is the case with strategic teaming, in this dissertation we are concerned with finding high-performance teams, comprised of members utilizing learning algorithms, that work well together for a problem or task. Our objective is for the learning team to achieve synergy, where simultaneous actions of individuals in the team have greater total effect, compared to sums of their individual effects. This can yield a better group identity in which all of the knowledge "kinks" have been smoothed out over time. An optimal team organization likely depends on the desired result, previous experience of members, and culture of the organization [109]. The theory is that some optimal team configuration exists for specific problems, and we attempt here to study the methodology behind it.

In a team, members usually work with the understanding that both individual and team goals can be best accomplished with mutual support [99]. Personal development and learning is of importance because it directly helps the team. They also offer contributions to success by applying unique talents and knowledge. This is highly the case with heterogeneous or diverse teams. Good team members also make a strong effort to understand other members' points of view. Additionally, team members expect conflicts will arise (as it is normal in human communication), but view these situations as opportunities to learn new things, update their knowledge, and resolve the conflicts constructively. Teams also allow the power to recognize connections between what has been learned and discovered by integrating their knowledge.

Individual training is of great importance as the team is only as good as its weakest link in a cooperative or collaborative task. For instance, a machine learning algorithm may perform very well and not be able to finalize a team task in which all other team members have their responsibilities completed. Completion of a task relies on individual mastery or understanding of individual roles and the task itself. This promotes the use of knowledge transfer events that can attempt to level the skill of all members to function more efficiently together. Teams rarely begin in a state where their performance is at its best; it takes time to develop individual specialization, niches, and knowledge refinement. Over time, the group must resolve conflicts or fill gaps in their individual and collective knowledge. Learning teams should learn how to learn together [97].

A team of diverse/heterogeneous members not only introduces several viewpoints, but also forces individuals to learn outside their expertise and gain additional knowledge while contributing to the group's overall intelligence. This occurs in businesses and human teams that are constructed to solve complex problems or are assigned to tasks which they are not familiar with. This helps advance their experience, robustness, and utility. Potential for diverse teams composed of several different learners can be substantial, which is why they can be seen in many modern organizations [97].

1.3 Problem Statement

In this dissertation, we develop a methodology for utilizing a heterogeneous mixture of learning techniques in a team of intelligent agents which share knowledge (i.e., collaborate). We analyze the performance based on team configuration and collaboration frequency. We use the term "intelligent agent" in this research to represent both robots and software agents. We recognize a team as *heterogeneous* if it consists of members that use different learning techniques. For example, a team consisting of one reinforcement learner and three neural network learners is considered heterogeneous. We recognize a team as *homogeneous* if the learning techniques are the same for all members, for example, a team composed entirely of decision tree learners.

We also distinguish between the strategies which the learners use, namely, centralized, distributed, or a hybrid (combination) of the two. A *centralized* strategy is one in which the agent is guided or directed by a centralized entity. A *distributed* strategy is one in which the agent performs without any outside assistance or direction from a centralized entity. A team of intelligent agents that utilizes a combination of these approaches is considered to be using a *hybrid* strategy.

Research in the area of multi-agent learning teams has been extensive, but has focused little on heterogeneity in terms of learning methods and strategies for teams of multiple, simultaneously learning entities. Most researchers have focused on homogeneous teams (utilizing the same learning technique) with varying learning rates [23]. Furthermore, sharing knowledge between agents/robots has received limited attention, especially efforts involving multi-agent collaborative learning. This is due to the difficulty in terms of communication and the frequency in which it takes place. Even more complex is sharing knowledge between entities that employ different learning methods, as the underlying knowledge representations typically differ.

This work investigates several research questions and underlying issues associated with multi-agent team learning. The primary research question we attempt to address is:

Primary Research Question:

> *How does incorporating multiple learning techniques, knowledge-sharing, and collaboration affect overall task performance of a team of learning agents/robots?*

This research question involves investigating multiple underlying issues. The study involves the employed machine learning techniques and strategies, the size and composition of the teams, and the knowledge-sharing language and protocol. The supporting research questions are as follows:

Supporting Research Question Series #1: Learning Techniques and Strategies

> *How can multiple, different learning techniques be used in concert with one another? Which learning techniques and strategies work together more effectively and/or efficiently?*

This dissertation investigates machine learning techniques and their application to tasks involving multiple intelligent agents. As each agent is simultaneously learning, multiple learning algorithms are considered and used to assess the proposed approach. Some learning techniques may work better than others for certain situations, domains, and team sizes. Some may also better complement other machine learning techniques being used in the same team. Some learning algorithm and strategy combinations may result in faster or more robust team learning.

Supporting Research Question Series #2: Team Size and Configuration

> *How well does the methodology scale from small teams to large teams composed of many simultaneously-learning agents? How does team composition change as the size of the team grows in terms of performance?*

A team configuration that works well with a small team may not work as well on a larger scale. This same concept can also be applied to the strategies (centralized or distributed) used by the team's members. Experiments involving various team sizes and configurations are performed to address these issues.

Supporting Research Question Series #3: Knowledge-Sharing Protocol and Decision Combination Method

> *How can multiple agents/robots, each using a different learning technique and underlying representation, share their knowledge? How does the frequency of sharing/collaboration affect overall team performance? How can knowledge conflicts be effectively resolved? What decision combination methods are generally successful, and how are they affected by team configuration and size?*

As learning algorithms may employ different underlying knowledge representations, communicating from one learner to another requires a medium that each can understand and convert into their specific representation. This represents a difficult aspect of the collaboration process. As communication affects time and resources, it results in a trade-off. Knowledge conflicts may arise, and the method with which these are resolved requires study. Additionally, how decisions are combined from multiple learners can have an impact on team classification accuracy. Several combination methods are investigated to study their effect on performance, as well as how the performance changes based on the team's size and composition.

1.4 Research Challenges

Distributed, cooperative learning presents challenges of scalability and communication overhead for sharing knowledge. As a team increases in size, combining knowledge not only becomes complex, but also time-consuming. Problems also arise in terms of conflicting knowledge, where one agent may have been rewarded for an action whereas one or more other agents were penalized for the same action and situation. This would cause action evaluations to be different during knowledge fusion, where interpolation or conflict resolution would be required. Combining knowledge in a distributed fashion requires explicit communication or interaction with a globally-available knowledge base (e.g., blackboard) or acting manager.

Communication is time-consuming and represents a bottleneck of the knowledge-sharing process. Extensive communication also uses energy and can reduce the lifetime of a mobile robot, for example. Real-time fusion of knowledge from many agents is infeasible for teams of more than two due to the time it takes to do so. Near real-time knowledge sharing is possible, but overall may slow down a task. Communication data can also be lost if interactions are not organized. Therefore, communication is a primary reason for the scalability of such systems being hindered.

Different learning techniques typically represent their knowledge in different formats. For example, reinforcement learning associates a reward/punishment weight with actions for certain situations. Artificial neural networks represent their knowledge in inter-neuron weights and their overall connection structure and hierarchy. Decision trees represent their knowledge as a tree-like structure, with branches to subtrees based on how they model the underlying data. Rule-based systems store knowledge in terms of explicit rules or facts. As knowledge will be fused from each of these representations, a unified knowledge structure (or middle language) needs to be established so that knowledge from agents using different learning techniques can be directly converted and translated for other agents to be utilized. This structure must be designed in such a manner that it can be efficiently communicated to all agents in the team.

Such a learning formulation could greatly speed up the learning process and allow a team of agents to gain a deeper understanding. Therefore, aggressive learning could emerge from the collective sharing of learned information using common communication and knowledge organization. Team learning then becomes a collaborative effort, synonymous with a research team sharing information from different fields to design a new device. This could prove very useful for challenging environments, where learning may need to take place on-the-fly, but where frequent communication and knowledge fusion are not desired due to power restrictions and energy conservation requirements. Multiple, different learning techniques may also be able to more efficiently learn about a complex environment than a single agent, or a team employing the same learning method.

This research can be generalized to several applications and environments, as well as scaled to team sizes ranging from a single agent, to a large team, or potentially to multiple cooperating teams of intelligent agents. As an example, a team of agents employing several learning techniques could be deployed in polar regions to collect data and learn about the ice characteristics. The same methodology could also be applied to the Internet, where several Web robots learning different aspects or topics, could periodically share their knowledge.

1.5 Research Approach

The concepts observed in human workgroups and teams can be directly realized in groups or teams of intelligent agents, such as mobile robots and multi-agent systems. Robots can learn individually using their own style (or learning algorithm) and periodically share what they have learned with their teammates. This can then help the team, which can range from a completely homogeneous to a completely heterogeneous composition of learning styles, to achieve its goal. The team can learn a task much faster and increase its overall performance. This simultaneous learning of individual viewpoints being shared between several agents can help the team learn more complex tasks and cooperative behaviors while working toward a common goal.

The use of multiple heterogeneous learning algorithms has been recognized as offering advantages over the use of homogeneous teams and single learning algorithms. It is well-known that different learning algorithms go about learning, or modeling the data, in different ways based on the underlying mathematical background and theory. This provides a much higher probability that the learners will be complementary in their expertise and weaknesses. An important aspect that supports this notion is how the learners are correlated with one another: the less they are correlated, the more likely they misclassify different instances. By combining the resulting theories (learned knowledge or model) of these heterogeneous learners, increased accuracy may be attainable over individual learning algorithms for complex learning tasks. Additionally, by distributing the learning problem over multiple learners, improved accuracy may be possible much faster.

The primary focus of this research is on applied artificial intelligence in the form of collaborative multi-agent and multi-robot learning. This introduces the notion of asking other learners for help during the learning process at specific times, termed *Collaboration Events*. During these scheduled events, each learning agent evaluates which of its training instances it is misclassifying and accumulates a list. It then requests that all learners (including itself) attempt to learn these training instances as well. This process is repeated until all training instances have been exhausted, following a collaboration schedule which governs the learning team's collaboration frequency. Thus, a collaboration event consists of all learners trying to learn particularly challenging instances with which other learners are having difficulty. Alternatively, the process could continue until the error rate increases,

with special attention to avoid over-fitting.

We also aim to have the learning agents collaboratively and cooperatively learn the problem by breaking the training data up into L stratified (i.e., evenly distributed) training portions, one for each of the L learning agents. If the team's collaboration schedule consists of C collaboration events, each of the L training portions will be further separated into C stratified training portions. The process starts with each learner being trained on the first of C sub-portions of its portion of the learning problem. Each learner then evaluates itself on all training instances it has seen thus far, compiling a list of those instances which it misclassifies (i.e., the most challenging instances to learn/model). A collaboration event then takes place, where all learners provide all other learners with their misclassification list. All learners then attempt to help one another by training on those instances as well. The process is repeated until all learners have exhausted their learning portions.

When the team is confronted with test instances after the process has completed (which it may have never seen before), the learners will hopefully be able to classify difficult instances with increased accuracy. This consists of asking each of the L learners their decision for each test instance, combining those decisions in a weighted manner based on the confidence of their decision. Confidence will be measured by the probability that the learner associates with that decision compared to other possible decisions. Another option for combination weighting could be based on the learner's final error rate. The decision receiving the highest weighted sum will be chosen as the team's final decision. We compare several popular combination methods and study their general performance across team configurations.

This process is similar to "boosting" in the Machine Learning field, but with the added components of collaboration between learning agents and a collaboration schedule. This method can be considered analogous to announcing, "I am having trouble with the following instances. Could you please help me learn them?" Multiple learners are then exposed to the problematic instances. If some instances are continually problematic, all learners will be trained on those instances multiple times, increasing the probability that those instances will be properly classified. Such a method is hypothesized to be better suited for problems with many training examples, so that ample collaboration can take place to increase team classification accuracy.

This dissertation focuses on this approach for a team of mobile robots or agents, where each robot or agent can employ a different learning technique (e.g., reinforcement learning, neural networks, decision tree, etc.). The idea is that each agent or robot will likely learn different information, at different rates, as it progresses with its task. Combining the differently-learned information within the team can allow distributive learning at a faster rate than relying only on individually-learned knowledge, or knowledge using a single learning technique/algorithm.

Three applications are utilized in this work to study several aspects of multi-agent

collaborative machine learning. The first application deals with real-time learning in the pursuit domain, involving multiple agents collaboratively learning to surround and capture an evasive prey (i.e., the predator-prey problem). The other two applications deal with unique and challenging subsurface classification data, namely, rock facies classification from wireline well log data, and subglacial water presence classification from radar data acquired in Greenland. The goal is a set of general heuristics governing team composition and strategies for solving complex learning/modeling problems using a heterogeneous team of learners. A Multi-Agent Collaborative Learning Architecture has been designed and developed to study these aspects and several facets surrounding team construction, knowledge fusion, and team learning dynamics.

1.6 Dissertation Organization

This dissertation is organized into seven chapters. Background topics including machine learning, multi-agent learning, learning techniques utilized in this research, and knowledge sharing/collaboration are discussed in Chapter 2, along with the primary related work. The research methodology is presented in Chapter 3, covering the design and implementation of the Multi-Agent Collaborative Learning Architecture, as well as the experimental setup and evaluation details.

Chapter 4 presents a collaborative multi-agent learning study for the application of cooperative pursuit and capture. Results are presented using two mixed learning techniques and strategies with a team of four intelligent agents in a simulated environment. Chapters 5 and 6 involve geologic rock facies classification using wireline well log data and ice sheet basal water classification using ice-penetrating radar data, respectively. These two studies utilize the Multi-Agent Collaborative Learning Architecture developed as part of this research. Team studies involving a variety of machine learning techniques, combination methods, data distribution techniques, and collaboration frequencies have been performed for each technique.

Each application chapter (4, 5, and 6) contains its own background and related work discussions, as well as detailed team-based learning results. A summary of generalized results across applications are included in Chapter 7, focusing on heuristics that were discovered from this work that can be applied to team learning for other applications. Finally, conclusions, contributions, and future work are outlined in Chapter 7.

Chapter 2

Background and Related Work

Machine learning is a well-established field, and algorithmic learning tools such as reinforcement learning, neural networks, support vector machines, and rule-based systems have become popular in the learning literature on robotics and multi-agent systems, with significant focus being on cooperation and learning.

In a recent study [95] on the current state of multi-robot learning and major unsolved problems, several aspects were raised proposing that heterogeneous collaborative learning can help further understanding to:

- Develop effective on-line multi-robot learning algorithms
- Achieve creative adaptation to dynamic conditions and environments
- Learn teamwork through experience
- Understand impact of heterogeneous capabilities toward team performance

Multi-agent and multi-robot cooperative learning have seen some recent research attention [24][69][92], mostly involving homogeneous teams in terms of learning methods [2][23][125]. The majority of research in cooperative learning involves reinforcement learning, with some focusing on long-term learning applications [40]. Knowledge sharing and integration have also been studied, but mostly outside the realm of intelligent robots [11][26][103]. The consensus from current literature is that sharing knowledge from multiple heterogeneous sources is challenging, where it is necessary to establish a common middle-language for communication. This dissertation studies these aspects, with an added aspect of a sharing schedule and the development of an inter-knowledge representation based on an instance/situation-action model.

2.1 Machine Learning

Machine Learning (ML) is concerned with algorithms and mechanisms to allow computers to learn, or model behavior, data, or an environment. Learning is induced by automatically

extracting information from data, sometimes in a statistical manner, and adjusting the resulting action/output, or adapting behavior or its structure over time. As a subfield of Artificial Intelligence (AI), ML is highly related to other subfields of data mining, pattern recognition, and classification.

Enabling computers to learn and automatically update control parameters and/or behaviors can eliminate the need for extensive human intervention or need for *a priori* assignment. It also provides a way to quickly and intelligently search through data to learn underlying patterns which are not evident. This can also give rise to multiple entities which learn to cooperate with one another, perform a task more efficiently, and intelligently coordinate. In short, systems can be created for addressing more complex problems.

The manner in which learning takes place and the output of the learning process depends on the underlying algorithms. These algorithms typically fall into one of several categories, depending on the task, knowledge structure, and the output desired as a result. The three main approaches to learning are:

- *Reward-Based Learning*: rewarded based on quality assessment of output
- *Unsupervised Learning*: feedback not provided
- *Supervised Learning*: correct output provided

Others [69] have listed three main categories for teaching/training systems:

- Learning by being told
- Learning from complete examples
- Learning from rewards

Many techniques exist within each of these learning categories. Those related to this research are discussed in this section, which focuses more on the former list of categories.

The more a situation or instance is encountered, the more confident the experiencing agent can feel about the resulting action. This is because experiencing the same situation over time will help agents refine their knowledge for that situation, correcting previous mistakes or confirming that previous actions were beneficial. Some learning algorithms are *unstable*, meaning they experience major changes with small changes to the input. Examples of unstable algorithms include neural networks, decision trees, and rule-learning algorithms.

2.2 Learning in Multi-Agent Systems

Distributed Artificial Intelligence (DAI) is a field within AI, receiving extensive research attention in the last decade. DAI is concerned with multiple, independent entities working toward a goal or interacting in a domain [122]. A subfield of DAI is Multi-Agent Systems (MAS), which is concerned with collections of robotic or software agents. This is an

extension of single agent approaches, in which the agent is only concerned with its own knowledge and actions toward one or more goals. Multi-agent systems, on the other hand, can be less centralized (more distributed), involving many agents working toward a common goal and concerned with the group's overall knowledge and performance. They are inherently complex, which makes the use of machine learning algorithms advantageous to help tackle the complexity and improve performance.

Multi-agent systems are systems composed of several software agents working independently or in cooperation with one another to solve complex problems. Multi-agent systems have the ability to solve problems which single agents cannot, or solve them in a more efficient manner. In this respect, multi-agent systems are agent-based models which can focus on several different applications, including team learning, cooperation, social behavior, and business and technology problems. Learning systems involving multiple agents can also help with better understanding the learning processes of humans, animals, and other natural multi-agent systems.

As such a system contains several intelligent agents, each agent can encompass different strategies, knowledge about others and its environment, and differing capabilities. Agents can be diverse and/or specialized to further aid in solving complex problems. At any given time in a multi-agent system, agents' knowledge about the world may differ from one another (including their internal states). If all agents knew everything about the world and one another, they would essentially be omniscient (acting as appendages of a single body). Using a variety of methods, agents can communicate with one another for cooperation, coordination, and to advance their knowledge. This communication can help increase the speed in which the system performs its task. However, unrestricted communication can reduce a multi-agent system to a single-agent system [122].

Intelligent behavior of agents within a multi-agent system can result from several sources, including how the agents learn, how they cooperate, and how they model others and their environment. Communication between agents can also range from being very simple to highly sophisticated. For example, when cooperating on a task and sharing knowledge, agents can enter into debates or negotiation sessions about how to solve problems or resolve conflicting knowledge. These properties of multi-agent systems make them a powerful research tool.

These aspects are central to this research, involving the sharing of knowledge between a heterogeneous mixture of intelligent, learning agents. This section discusses background and related work in terms of our definitions of heterogeneous teams and distributed strategies, team learning, multi-agent machine learning, and concurrent learning. The subsequent section discusses knowledge sharing and integration with respect to agent systems. A review of multi-agent systems, machine learning, and the combination of the two can be found in [146]. Additional discussions on heterogeneity and communication can also be found in [122], while includes a detailed overview of multi-agent learning.

2.2.1 Team Learning

Team learning occurs when members of a team cooperate and coordinate to learn a task. Collaboration can take place by sharing knowledge situations, episodes, or entire policies. By working together, the hope is that teams of individuals can learn how to execute a task with increased efficiency. Adaptation and emergent behavior can then be a byproduct of intelligent team interaction. Human teams can be thought of as cooperating systems of multiple intelligent agents that are learning and interacting. Thus, knowledge is distributed amongst team members. This distributed knowledge can be combined to harness the expertise of an entire system, as each entity encapsulates different pieces of the collective knowledge.

The majority of team learning efforts have been focused on utilizing a single learning method for an entire team, or a team that uses the same learning method for all members. Some learning algorithms exhibit better individual performance than others for specific tasks. Combining them can improve overall robustness and increase team capabilities.

Techniques are needed, including learning, to allow a team to adapt to a changing environment over time, as well as change team composition, size, and member capabilities [93]. Adaptation can be achieved by monitoring other robots in the team and adjusting control parameters accordingly [93]. This adaptation can take place over a long period of time, representing life-long or long-term learning. Life-long tasks are characterized by the need for agents or robots to be able to respond to changes in both the environment and the capabilities of other team member capabilities. These capabilities could be learned behaviors, knowledge representation, or the learning algorithms themselves.

Part of the inspiration for this research is from Tan's work [126], which discusses cooperation in multi-agent reinforcement learning. That work investigates whether having the same number of reinforcement learners that cooperate outperforms completely independent learners who do not communicate during learning. It also investigates the cost of communication. In our terminology, however, this represents a homogeneous learning team. Sharing was also investigated in three main forms: sharing sensations, sharing episodes, and sharing learned policies. Sharing episodes occur by sharing the entire solution space with another agent only when a capture was made. The other agent then used this solution space to update its knowledge (and therefore learn from it). This work supports the theory that novices can learn quickly from experts. Cooperation during the early learning phase showed statistical significance, but no statistical significance was observed on which sharing mechanism to use.

It was found that additional sensation information from other agents is beneficial to the learning process, if used efficiently. Sharing episodes or entire policies provides a learning speedup, but at the cost of increased communication. It is noted that sharing can be utilized by heterogeneous agents, as long as they can interpret the episodes and use them accordingly. It is mentioned that a heterogeneous team of reinforcement learners could

be employed, where heterogeneity is based on the visual field depth of each agent (not the underlying learning algorithm). For these experiments involving averaging of entire policies at a certain frequency, the best performance was seen when averaging policies every 10 time steps with a limited visual field depth. Performance, however, was dependent on the visual field depth.

2.2.2 Diversity, Specialization, and Team Heterogeneity

A recent article discussing the state-of-the-art in cooperative multi-agent learning [92] stated that team heterogeneity is a direction of research that has not been adequately explored. Agents frequently embody different capabilities; however, most literature presumes that all agents are identical in both behavior and (learning) ability. Our research attempts to directly address this focus via the use of heterogeneous learning and sharing methodologies.

We recognize a team as *heterogeneous* if it consists of members that use different learning techniques. For example, a team consisting of one reinforcement learner and three neural network learners is considered heterogeneous. On the other hand, we recognize a team as *homogeneous* if the learning techniques are the same for all members of the team, for example, a team composed entirely of reinforcement learners.

The literature contains different definitions of "heterogeneous" and "homogeneous", stemming mainly from specific applications. For example, [115] defines heterogeneity in terms of focusing on different problems or tasks. Others define agents in a team to be heterogeneous if the members utilize the same learning method (e.g., reinforcement learning) with different learning rates [67]. Similarly, [87] has investigated the use of heterogeneous reward functions. Further, [16] defines a team to be heterogeneous based on its members' policies after training. Heterogeneity can also arise from sensors and sensor capabilities of each entity. Uniformity of agents in a team is analogous to its heterogeneity in some situations as well. Most work involving heterogeneous teams focus on the robot platform rather than the underlying learning mechanism [94].

Use of heterogeneous agents and teams has some support in the literature as well. In [16] heterogeneous teams and team diversity were studied, noting that heterogeneity of a team can be beneficial for some types of tasks, but unsuitable for others. Depending on the task, a homogeneous approach may not be able to achieve the optimal result [83]. Results from [67] state that observed evidence supports a heterogeneous system providing complementarity in learning. They also saw evidence of learners with higher learning rates helping those with lower learning rates, and vice versa. Results from [141] also support that complex reinforcement learning systems can learn faster with help from simpler reinforcement learning systems (i.e., heterogeneous in the form of complexity).

With team learning, topics of diversity and specialization are important. For instance, a team of learners may begin homogeneous in terms of knowledge, but can be considered

highly heterogeneous as the task progresses due to learning differences and specialization that can occur through unique experiences. A system can be driven to become specialized as long as agent diversity offers an advantage to a team's performance [83]. Assuming agents are clustered based on similarity, each cluster can be believed to be a group of specialists. The more specialized clusters that exist translate to more diversity in the system. In general, diversity means differences (whether good or bad) among individuals in a group with respect to team performance. In [83] *specialization* is defined as the part of diversity that is demanded for better performance. Alternatively, if diversity enhances performance, it is more specialized. As an example, a worker that takes on many different jobs becomes less specialized compared to workers who focus on a single job. Thus, if performance generally increases with more diversity, the degree of specialization should increase as well [83]. Their results showed that, for the stick pulling problem, further diversity does not necessarily improve performance in all cases.

Behavior-based multi-agent robot teams and reinforcement learning have been studied [15][16], mainly focusing on behavior differences in teams of mechanically similar learning robots learning the game of soccer. The work analyzed different reinforcement functions based on their affect on diversity, ranging from local to global with varying reinforcement structure/focus. The mechanically similar robots were only different in their behavior. Thus, as agents continue to learn, their behaviors change and they become increasingly diverse. The behavior of a single robot affects the entire team, in that the behavior propagates through the team via sharing and cooperation. Even if the agents or robots are homogeneous, their *behavior* will eventually become heterogeneous. In addition, use of behaviorally heterogeneous agents, utilizing different learning algorithms, may make it difficult for other agents to model or predict agent action.

The results showed that, in many cases, behavioral diversity automatically occurs during the learning process for a team learning task. However, the level of diversity and specialization depended on the reward structure. For the soccer task, it was reported that global reinforcement caused the formation of heterogeneous policies, and that local reinforcement generated identical policies. Local reinforcement is said to translate to learning greedy policies, which will degrade team performance. However, our results show that local reinforcement can result in quite different policies for agents, even those that are using the same learning algorithm and parameters. This local specialization can lend itself well to team success, where the reward structure's effects on performance may be highly related to the problem domain or discretization of the environment.

Differences within a team, sometimes called castes [83], can also be identified and studied. For instance, agents in a caste can be networked and coordinate their adaptation process such that the caste homogeneity remains intact. In relation to our research, identical learning methods within a group represent a caste that could learn together (separately from the other different learning methods, which could learn as castes themselves).

The works by Kelly *et al.* [69], as well as our research, parallel that of a single learning entity (the team) that actually consists of many simultaneously learning agents. Balch does raise the question of whether the best policy depends on the size of the team [16]. The size of the team affects the role of specialization, as more agents will reduce the importance of specialization. It is also stated that future work would be to investigate and evaluate the impact of learning types on robotic systems. Our research strives to use diversity to increase team specialization for a given team task. However, sharing knowledge in certain manners can reduce both diversity and specialization.

2.2.3 Multi-Agent Machine Learning

Our research focuses on the topic of Multi-Agent Machine Learning (MA-ML) [146], where a multi-agent system is participating in some form of learning event. The learning is distributed among intelligent agents in a team, where agents collaborate by periodically sharing knowledge to achieve a common goal. Learning with physical robots introduces a more difficult challenge, as the environment is much more continuous and complex in terms of situations that are experienced and actions that are required [22]. Learning takes far more trials than is typically feasible in a physical environment, which is why agents are used to simulate our mixed learning methodology. A detailed discussion of issues, challenges, and current attempts with each of the four combinations of heterogeneity and communication in multi-agent systems are included in [122] and [92].

There are several positive aspects of using multi-agent systems for cooperative learning [116]. Utilizing multiple agents allows a system of individual agents with limited resources to scale up with the use of cooperation. Use of multiple learners may also increase speed and efficiency for large and complex tasks. Distributed systems tend to degrade in performance more gracefully in failure situations, and checking results between multiple learners can help the system be more reliable and fault tolerant. Finally, multiple learners allow the system to encapsulate specialized knowledge or expertise in particular agents. All of these aspects motivate the use of multiple learning agents.

Multi-agent learning systems are composed of three main components:

1. The agents
2. Their learning algorithms
3. Their task(s)

Agents typically vary in their properties, such as learning method, learning strategy, expertise, skill, level of awareness, etc. Levels of awareness can be determined by the number of sensors used for sensing other agents or the environment [136]. This can also affect the search space size, how detailed the agents can represent their knowledge, and additional sharing complexity if different levels are used in the same team. Our research incorporates a form of awareness, as the sharing frequency can be the measure of awareness level of team

members with respect to their teammates' knowledge. Using forms of awareness has been found to be beneficial [136].

Agents acting in an environment typically only know a fraction of the actual state of the environment. Some agents may have different skills or know different aspects of the task or environment. It may also be the case that there are aspects of the environment that no agent currently knows. However, by inter-agent communication and interactions with the environment, agents can merge their skill(s) and experiences (current knowledge). Interactions between agents and their environment can be classified as follows [146]:

- Frequency of interaction (low to high)
- Persistence of interaction (short-term to long-term)
- Pattern of interaction (unstructured to structured)
- Variability in interaction (fixed to changeable)
- Type of interaction (random to goal-oriented)

When extending interaction and learning to multiple agents, current learning methodologies can be affected. This is discussed in [116], which also introduces structured dialogue between agents as a means for cooperative learning. Multi-agent and/or multi-robot learning is challenging for many reasons (adapted from [96]):

- Large state spaces
- Uncertain credit assignments
- Limited training time
- Uncertainty in sensed and shared information
- Non-deterministic actions
- Difficulty in defining appropriate abstractions for learned information
- Difficulty of merging information learned from different experiences

To accompany these challenges, common factors that can affect the results of learning in a multi-robot system are [128]:

- Reward scope (local, global, number of moves)
- Global information delay (short-term and long-term)
- Diversity of robots (heterogeneous, homogeneous, behavior, role)
- Number of robots (team size and scale)

It is stated that experiments showed that reward scope and global information delay were the only factors that truly affected the final learning results. We additionally show that the diversity of robots in terms of learning methods greatly affects the results of learning for a collaborative team.

Global information can be defined as that which robots exchange about the world, where a "delay" translates to information not being shared for some amount of time [128]. In times

between sharing events, robots learn on their own and may not know where other robots are or what they are doing. Global team rewards typically do not scale well to difficult problems due to the lack of individual feedback for local actions. Global reward also can cause agents to co-adapt to others, which may hurt performance if some of the agents being adapted to are poor performers. Although local rewards may develop greedy behaviors, they can help improve individual performance in cases where group tasks can benefit from such focus. By sharing and integrating knowledge from several different learners, each agent eventually receives the local rewards of other agents through periodic sharing and knowledge synchronization events.

Tangamchit *et al.* [127] demonstrated superiority of average-reward-based learning for task-level multi-robot systems. They argue that at the task level, cumulative discount reward methods (such as Q-learning) cannot provide optimal solutions, whereas average-reward-based methods (such as Monte Carlo Learning) can potentially provide that. However, Monte Carlo Learning is rarely used as it is slow compared to other methods. In addition to local learning methods, such as those used in our research, some work has been done using a single robot learning method for an entire team [86]. It is recognized in literature that using a single learning method for multi-robot systems is insufficient [86][16][127].

Highly related to our research is work of Haynes and Sen [60][58][59], promoting that the use of k different behavioral strategies for controlling a group of k agents and their actions can be combined to form an efficient cooperation strategy for global goals. They use Genetic Programming in a predator-prey game, where each chromosome in the population represents k programs, each corresponding to an agent. Cooperation without implicit communication was examined, and thus, cooperation was defined as the agents being able to see one another. Teams are constructed by randomly selecting members from the population of chromosomes and assigning rewards based on a percentage of the total fitness. The percentage of capture was used as the measure of fitness [58]. Crossover techniques were used to evolve teams comprised of heterogeneous agents, and uniform crossover was said to be advantageous in this application and potentially in others as well. In addition to the agents being able to evolve, experiments were performed allowing the prey population to evolve. The idea was that potentially better or more efficient solutions would be discovered for both the agents and the prey.

Multi-agent learning has been referred to by many different names in the literature, such as mutual learning, cooperative learning, collaborative learning, co-learning, shared learning, team learning, social learning, pluralistic learning, and organizational learning [146]. Multi-strategy and multi-perspective learning in multi-agent systems are challenging research issues [146] that our research attempts to address. Current multi-strategy methods typically involve more of a multi-stage approach [54] or utilize different learning rates for Q-learning teams [12][13]. Multi-strategy approaches may also involve combinations of human

advice and agents learning or modifying the advice. Recent work has been done under the name of multi-method learning [123]. In this work, a new strategy for behavior acquisition for domestic robots is proposed. Behaviors are acquired using multiple differing learning methods that are subsequently incorporated into a common behavior selection system, enabling them to be performed in appropriate situations. The authors also state that in domestic robotics, applying a single learning algorithm is "clearly insufficient", and that a single learning method is inadequate for acquiring the wide repertoire of behaviors required for domestic robots. No single learning method is fully capable of learning all behaviors at the highest efficiency, at least not yet [149]. Rather than trying to find the optimum learning algorithm, a system can be developed that can incorporate various behaviors from different learning methods [123]. Results prompt further investigation on whether adopting different learning methods can be an effective approach for many applications. This is precisely the motivation and goal for our research.

2.2.4 Mutual and Concurrent Learning

Concurrent learning is defined as using multiple simultaneous learners, often one per agent [92]. Use of concurrent learning as defined here, we may be able to reduce the problem space by projecting it onto N separate spaces (one per learning agent). Heterogeneity can also emerge as a result. However, using multiple concurrent learners causes the environment to become non-stationary, violating assumptions made by some learning techniques. The non-stationary environment may modify agent behavior in a negative manner, possibly corrupting the learned behaviors of others. As research advances in this area, significantly modified versions of existing learning techniques - or new learning techniques - may be required.

Further inspiration for this dissertation are the works by Kelly *et al.* involving mutual learning of robots sharing experiences [62][68][70][71]. They call this homogeneous approach "dual learning", "joint learning", and "group learning". That work focuses on two and four mobile robots, each using reinforcement learning, passing experiences from one robot to the others. The goal was to increase learning rates by sharing every learning cycle, transmitting the input state, chosen action, and action reward (alpha factor). Results showed that using four mutual learning robots learned the fastest, followed by using mutual learning with two robots, and lastly with individual learning. Experimental results also showed a nearly two-fold increase in learning rate as the number of robots doubled, but not quite linearly. Nonetheless, as their method shares experiences every learning cycle, they state that the approach is not scalable to very large groups in terms of communication costs. They estimate larger groups would show diminishing results even if communication was perfect.

Allowing several learning agents to acquire knowledge from one another can accelerate the entire learning system, making learning more efficient. This was observed in [66], where several agents served as learning processes for a navigation problem, and the learning

rate was adapted over time. The longer the agents learned, the less the learning rate was adapted/changed. One of the results from their work showed that adding a non-collaborating Q-learner agent to the team did not improve the system significantly. Their work also appeared to share at every step, taking the maximal value of all learners when choosing the action.

Ideally, the overall learning time should be inversely proportional to the number of robots simultaneously learning together [71]. Furthermore, if communication is perfect between a pair of learning robots, the learning rate should effectively double. This would cause the system to act as if a single agent was learning two situations at the same time. These statements assume that each situation that is learned is unique or new, which is not always the case. Of course, when multiple agents are learning concurrently on the same task, the probability of conflicting knowledge or points of view increases. In these cases, knowledge conflicts should be recognized and resolved. In all dual learning cases, a lower standard deviation was seen compared to single learning cases [71]. This shows that dual shared experience learning produces more repeatable results compared to single robot learning.

Work similar to ours in terms of learning and cooperation components is described in [116][114]. In that work, agents learn from their experiences. They utilize an interaction board (based loosely on a blackboard) for agent negotiation, with agents "talking" about hypotheses with an interaction language (consisting of nine operators). They use an integration function for combining agent opinions. For resolving conflicts, agents vote on whether they confirm/agree, agree with modification, or disagree. Cooperative learning is defined by the authors to consist of agents with partial hypotheses interacting to improve them. Our methodology allows the number of learning and/or participating agents to be arbitrary. This however affects the time required to synchronize team knowledge.

The concept of simultaneously utilizing multiple, mixed learning methods in a team can be directly extended to the case where each agent is capable of switching learning methods on-the-fly, based on how the knowledge from one learning method can be converted to the knowledge representation for another learning method using the common middle-language. Thus, if one learning method is performing badly, it can dynamically switch to some other learning technique to attempt to improve its performance. An extension of this approach, which is discussed as future work, would be to have multiple different learning methods concurrently learning per agent. This essentially embodies multiple learning paradigms in a single agent. For example, N threads could be started, each running a different learning method, to perform simultaneous learning within one robot [141]. When an action needs to be chosen for a given situation, each learning method can be consulted on what it would recommend and a (weighted) result can be computed. Additional future work in these efforts involves investigation into how learning algorithms should be improved to be more flexible in a dynamic environment [62].

2.2.5 Agent-Based Knowledge Sharing and Integration

Sharing knowledge and experience between intelligent agents allows them to learn from one another, potentially providing the advantage of increasing overall team learning rate, efficiency, and success for a task. Sharing can also be considered another form of learning. How to share and combine knowledge with others is not obvious, and this area of focus represents a challenge.

The goal of combining knowledge from learners is to utilize knowledge and experiences that complement the other agents in an advantageous manner. The gain from combining is also typically related to how correlated or dependent the learners are with one another. Some definitions of representations and combination strategies are the following [11]:

- *Uni-Representation Combination*: All learners using the same input representation
- *Multi-Representation Combination*: Different learners employ different representations of the same input
- *Multi-Expert Strategy*: Learners work in parallel, where they all give their decisions and a separate combiner computes the final decision using the chosen combination method
- *Multi-Stage Strategy*: Next classifier is trained on what the previous classifier(s) rejected
- *Cascading*: Using a sequence of learners that use the next learner if the current learner's confidence level is low

Knowledge integration is a concept used in several fields, such as information retrieval, expert systems, and learning systems which combine information in any way. A knowledge integration method can be thought of as a filter that helps resolve noisy knowledge (e.g., knowledge conflicts) between multiple knowledge sources. Knowledge integration and retention has the advantage that knowledge can be compiled over time. This combined knowledge from several sources therefore represents an integrated policy, a more compact representation of group knowledge. It also represents a good starting place for future learning and adaptation.

There is no single knowledge representation that is best for every problem, and such a representation will likely never exist [49]. If the knowledge sources being combined are different in any way (ontology, overall structure, scale, etc), then combining or sharing becomes difficult. A common knowledge format, or inter-knowledge representation, then becomes required so proper conversion can take place from one representation to another. However, for teams using heterogeneous learning methods and knowledge representations, translation to a common language becomes a necessity. Our work attempts to establish an inter-knowledge representation system as a mechanism for converting between knowledge representations from heterogeneous learners. Other works have used a different language for sharing than their internal knowledge representation [81].

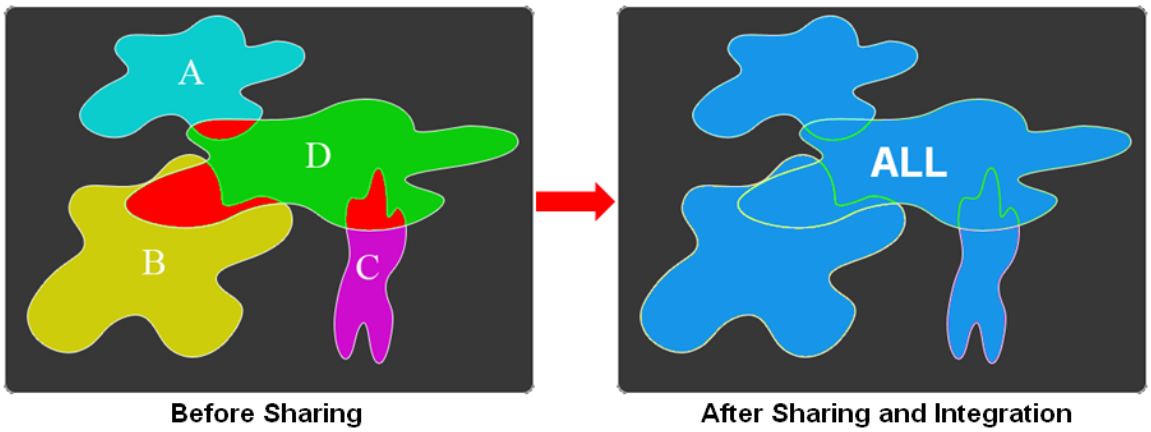

Figure 2.1: Illustration of the process of knowledge sharing and integration, where individual agents share what they have learned about a problem and knowledge conflicts are resolved where differing opinions overlap.

Sharing learned knowledge can also capitalize on different perspectives in addition to allowing for significant reduction in search effort [100]. This cooperation and sharing of information during learning can therefore increase the quality of learned knowledge as well as redirect and reduce the search. Allowing different learners to search the problem space in different orders may also increase the effect of sharing knowledge [100]. If the sensory or sharing information is insufficient, sharing events and incorporating this information into agent knowledge can actually hinder training and interfere with team learning [126].

Figure 2.1 illustrates the process of knowledge sharing and integration. Before sharing, four agents (A, B, C, D) have learned different portions of the universe of knowledge for the problem, the box. Some agents may have learned information which overlaps with others. It is in these areas where agents can agree or where knowledge conflicts can arise. The sharing process essentially gets the team together and they each share their knowledge with all other agents. During this process, knowledge conflicts are resolved so that the final result is a knowledge base that is synchronized (in terms of agreement) among all agents. The agents then adjust their knowledge using the result, a process we define as integration. Thus, after sharing and integration, each agent has learned experiences and expertise from other agents without having to directly experience or build it up themselves, essentially speeding up the learning process and increasing the skill of the team as a unit. The sharing and conflict resolution methods can dictate how agents retain their specialization.

Conflicting knowledge can be due to incorrect, incomplete, or local knowledge; different goals, priorities, or evaluation criteria; and resources [81]. During the sharing process, knowledge agreements and conflicts can be further used to adapt an agent's confidence for certain situation-action experiences. These knowledge conflicts can provide valuable information about individual viewpoints, behaviors, and views of both the world and the goal(s) of the respective knowledge sources [51]. A set of distributed heterogeneous

knowledge sources creates a social layer above the knowledge layer, containing interesting relationships between knowledge facets (e.g., contradiction and approval).

Reaching consensus on resolving conflicting knowledge should be the decision that has the most support [115]. However, we argue that this depends on the task and how the task and environment are discretized and described for the learners. For example, in our capture task, a situation may describe all agents in a near-capture position. Thus, each agent will likely choose a completely different action than the others. Even in the case where there are action choices that overlap in the team, having all agents perform the same action for that situation may be counter-productive (e.g., allowing the prey to escape in a sure-fire capture situation). Furthermore, reaching consensus requires confidence levels or experience to tag the knowledge with, and may also incorporate negotiation. This could become a time-consuming and difficult process, which may not provide substantial overall gain as a result.

In order to try to combat these collisions, [115] introduced an index of expertise to measure the quality of advice from one or more other robots, defined as the ratio between the total quality of all advice and the number of times advice was given. Advice quality was separated into three categories of good, regular, or bad, based on the reward from the advice. The index of expertise is then used to select the most preferable advice from multiple advising sources, where a higher index translates to higher probability of choosing the advised state-action. It is claimed that this measure helps increase the probability of finding a solution within a search space. As knowledge conflicts are resolved, redundant and overlapping knowledge are automatically removed and distributed to the other agents during the sharing event. Results from [103] showed that sharing knowledge allowed escape from local minima, and that the total number of knowledge collisions and length of the learning period were decreased by sharing learned knowledge.

Brazdil and Torgo have attempted to integrate knowledge acquired by individual agents into a single integrated theory [25]. This integrated theory is then used by a single agent to solve problems. Results showed that using an integrated theory provided significantly better performance compared to individual theories. Our research involves periodically forming combined theories (both integrated and partially integrated) via sharing events for all agents to subsequently use, rather than just a single agent. The challenge that must be addressed is the difficulty in properly constructing an integrated theory. Utilizing integrated knowledge from several different learning methods will encounter differing knowledge representations that are likely not directly compatible [24].

Work by Sian is also related to the notion of an integrated theory, as the experiments involved establishing consensus rules by soliciting opinions of other agents when certain situations arise (e.g., an agent obtaining a confident hypothesis) [116][114]. These consensus rules then form the integrated theory, where the interaction takes place via a blackboard architecture. Similar to our work, agents are also allowed to interact during the learning

phase, which makes sense as using better hypotheses earlier leads to the possibility of better performance.

Constructing the integrated theory can also be considered a form of learning, as agents learn from being told rather than from direct experience. Knowledge integration can complement other existing machine learning methods [26]. However, some information can be lost in the process. For the purposes of our research, resolving conflicting knowledge will cause viewpoints/opinions to change for at least one agent involved in the sharing event. Even simple averaging causes exact knowledge weights to be adjusted. Knowledge integration also helps save learning time as theories can be shared so that other agents do not have to personally experience the situations to learn from them. The integrated theory can be constructed qualitatively or quantitatively, where theories can be ordered based on estimates of quality by selecting rules with the highest quality.

Integrated theories have also proven to be quite stable compared to individual theories [26], meaning that they fluctuate less than individual theories in terms of performance (small standard deviations). This also means that the integrated theory can be more reliable. Performance gains have been reported to be between 15-20% for the integrated theory compared to the use of individual theories [25]. The knowledge integration process provides a good way to help eliminate random variations of noise and avoid overfitting situations. However, there are several difficulties of knowledge integration [51][124]:

- Knowledge sources operating with incompatible representation syntax
- Naming problems: similar names for different knowledge facets, different names for the same knowledge facet
- Semantic inconsistencies: divergent opinions, or incompatible knowledge bases
- Incompatible background knowledge
- Differing contexts of distributed local knowledge
- Heterogeneous knowledge domains or domain access
- Communication, coordination, and contribution
- Asynchronous dynamics of individual knowledge contributions
- Unreliable or untrustable knowledge sources
- Non-normal knowledge source contribution
- Differing scopes and viewpoints (potentially overlapping) of knowledge sources

There are also three main approaches for integrating data from multiple ontologies [51][124]:

1. *Merging*: Creation of a new single, coherent ontology that includes information from all merged ontologies
2. *Mapping*: Original ontologies remain separate, but are made consistent and coherent with one another
3. *Matching*: Finding the semantic meaning between two or more ontologies

Our case of distributed heterogeneous learners sharing and combining their knowledge periodically is a mapping method of integrating data. Methods used in this research for combining data from separate knowledge sources will be discussed in a later section.

In [2] and [3] a cooperative learning method is introduced called Weighted Strategy Sharing (WSS), which involves detecting expert agents. Each agent assigns a weight to its knowledge, based on the amount of its teammates' "expertness." A homogeneous group of reinforcement learning agents is discussed that share knowledge to learn faster and gain more knowledge. They demonstrated this concept in a predator-prey simulation environment for an object-pushing task. A difficulty with such an approach is estimating expertise. Both experts and non-experts can have important knowledge which can increase team success or efficiency.

Both experts and non-experts can have knowledge that is useful: maybe a "non-expert" learning a critical move for an experience that the "expert(s)" have not yet encountered [3]. They define a model agent as one which other agents use as reliable knowledge sources for learning. Differences in agent expertise can occur via initial knowledge, different experiences, different training sets, etc. Agents were allowed to share their sensory data in terms of (situation, action, reward) triples, termed "episode sharing." Sharing this knowledge could be done via explicit communication, storage on shared memory, or by cooperatively updating a unique memory store.

The expertise of an agent can change over time [3], based on its experiences and success of particular actions in certain situations. Other efforts have either assumed expertness is equal among all agents or that it is static. Expertness has been measured in the past using numbers of successes or failures in an N-move window. This approach has been said to be non-optimal, where they measure expertness based on the relative level of expertness of other agents and an impressibility factor (i.e., how much an agent relies on other agents' knowledge). Six simple methods for measuring agent expertness were tested [3]:

1. Algebraic sum of reinforcement signals
2. Sum of absolute values of reinforcement signals
3. Sum of absolute values of negative reinforcement signals
4. Sum of positive reinforcement signals
5. Inverse of the number of moves each agent performs to reach the goal
6. Change in received reinforcement signals since the last cooperation time

Results showed that nearly all these methods produce approximately the same result as independent learning [2]. A method we investigate in this dissertation is the effect of avoiding knowledge conflicts, and only learning experiences from other agents which have not yet been encountered. This method would help the team share unique experiences to speed up learning, but would also allow the individual members of the team to remain specialized and behaviorally diverse.

Similarly, [125] discusses the Strategy Sharing Method, where agents learn from all of the agents in the team. Each agent individually learns using one-step Q-learning, forming a homogeneous team according to our terminology. At "special times", agents gather the Q-tables (all state/action values for the experiences they have encountered) of all other agents and average them to get the new strategy. A strategy is equivalent to a knowledge policy. Averaging may suffer from a reduced convergence rate for policies, and may slow adaptation in dynamic environments. Simple averaging can however be beneficial for overall team performance, as newly learned situations can immediately be incorporated.

Other variations of transferring knowledge have also been attempted. An example is transferring knowledge by saving learned knowledge from one problem in matrices and then loading this knowledge before performing another task [111][110]. From this, it was determined that learning from an original problem helped increase performance on the new problem by approximately 10% for the number of trials to converge. This is a promising result, as it supports the concept that learning one problem can help speed up learning or increase performance on another problem.

Some works have even experienced success in learning to coordinate without sharing information [111][110]. At the time of this work, nearly all efforts had assumed explicit or implicit communication of information. They believe that the less an agent depends on shared information, and the more flexible it is to online arrival of knowledge, the better it can adapt to changing environments. This is where our approach involving periodic sharing of knowledge fits well, as there will be some local learning before synchronizing with the group. The concept for that work was that agents would share no problem-solving knowledge, and that while each agent focuses on its own local reward, global coordination could emerge without the use of explicit or implicit communication. Reinforcement learning (Q-learning) was used in a block-pushing task to show that complementary policies could be learned without any knowledge from other agents.

The desired goal for this focus in the field is to have intelligent agents that can learn when to cooperate and which cooperation methods to use to achieve maximum gain in terms of performance and success [126]. Scaling up distributed machine learning may be possible using parallel processors and message passing [100]. When an agent finds a good rule, or conversely a challenging instance, it can share it with the group. Future work in terms of knowledge integration is focused on overcoming differences between agent languages [25]. These aspects are primary foci of our research.

2.3 Classifier Combination

2.3.1 Combining Decisions from Multiple Learners

Classifier combination is an area in machine learning that has offered advances in classification accuracy for complex data sets. It has been termed differently in the

literature, namely, classifier fusion, mixture of experts, committees, ensembles, teams, pools, collective recognition, composite systems, etc. When predictions from multiple classifiers are combined, they are said to form an ensemble that is then used to classify new examples. Several methods have been developed to combine classifiers, the most popular ones being voting, boosting, bagging, and stacking. The majority of efforts in combining classifiers have incorporated homogeneous learning algorithms. Fewer works have focused on combining heterogeneous learners, which is the focus of this dissertation. The term "heterogeneous" is used in different contexts in the literature. Popular classifier combination approaches such as boosting and bagging involve homogeneous learning algorithms and manipulation of the training data set. The produced models are therefore homogeneous in representation; although the learners may output slightly different predictions. Use of different learning algorithms, and therefore heterogeneous model representations, is considered purely heterogeneous in the contexts of both algorithm and model. Integrating multiple learning paradigms within the same learner is viewed as a hybrid scheme [73].

When developing a multi-classifier system, its members can be a mixture of weak (i.e., high error rate) and strong (i.e., low error rate) classifiers. Weak classifiers are typically simple to create, at the expense of their accuracy on complex data sets. Strong classifiers are typically time-consuming and expensive to create, as their parameters are fine-tuned and tweaked for maximum performance. Furthermore, some classifiers perform better than others on the same data set due to their algorithmic nature. Combining weak/strong or homogeneous/heterogeneous classifiers offers the benefit of encompassing different levels of expertise and knowledge bases. Exploiting and studying these properties as advantages for classification is a driving force for multi-agent machine learning, as proposed in this dissertation.

An important aspect related to difference in learners is their level of error correlation relative to one another, and as a whole. The more correlated (i.e., less disjoint) individual learners are, the less complementary they may be. If they are uncorrelated, they likely will misclassify different instances and combining them better enables the system to correctly classify more instances. A significant improvement over a single classifier can only happen if the individual classifier theories are substantially different. It is desired to obtain a balance between high performance and complementarity in a team where decisions are combined. If one learner does not predict correctly, the other learners should be able to do so. Diverse models are therefore more likely to make errors in different ways. Methods to accomplish this typically involve introducing diversity in terms of learning paradigms, feature subsets, or training sets [140].

The data set size also plays a role in the success of classifier combination methods. With smaller data sets, methods that rely on dividing or shuffling training sets become less effective. This is due to a lack of sufficient samples for learners to compensate for each

other's decrease in accuracy. Such methods are more applicable to large data sets, where learners can be provided enough training examples to adopt more comprehensive views of the feature space.

One of the main questions in this area of study is whether combining classifiers is better than selecting the best classifier. Several works support that classifier combination provides an improvement in most cases, assuming that the classifiers exhibit reasonable individual accuracy. One such study utilized stacking with model trees to combine multiple heterogeneous learners [46]. Each learner utilized the full data set to produce a base-level model, the output of which is then combined with other base-level models using a meta-level classifier. Their results also indicated that the number of base-level classifiers did not significantly affect the results. Similarly, [74] found that when using voting and entropy methods as the heterogeneous classifier combination mechanism for word-sense disambiguation, the addition of increasing number of less accurate classifiers adversely affected those with higher accuracy. Use of more classifiers, even if heterogeneous, does not always translate to better results. Depending on the combination method, the relative impact of adding classifiers diminishes as team size increases. Our work attempts to address this by implementing a framework where team configurations can be analyzed to determine what factors make a team of classifiers successful.

Although selection of the best individual classifier is easier and occasionally effective, combination techniques scale better to larger and more complex learning problems. Even combining all classifiers in an ensemble can be improved upon by selecting for combination only those that perform significantly better than others, termed Selective Fusion [139]. Using this technique, together with simple voting methods, enables fine-tuning diverse ensembles for specific data sets. It also offers performance comparable to other heterogeneous classifier combination methods like stacking, without the additional computation and meta-learning costs. Researchers have studied the effectiveness of switching between selection (occurring in regions of the feature space where single learners are dominant) and fusion (occurring in every region not dominated by a single learner) [79]. The results offer motivation to investigate other methods that may offer comparable performance, such as collaborative learning.

Just as humans reason about trust and evaluation of opinions from others, similar concepts can be utilized to weigh predictions from specific learners. Three heterogeneous classifiers were employed in [14] to study the effect of five combination methods, including weights proportional to each expert's accuracy on hold-out data incorporated during voting. Other variations on employing ranking and accuracy estimates for classifier selection have used accuracy in local regions of the feature space surrounding testing samples [151], and representing decisions from multiple classifiers as intersecting dimensions in an N-dimensional Behavior-Knowledge Space [63]. Uncertainty and reliability among experts have been studied by using entropy calculations to arrive at an aggregate decision score

[106], as well as the Dempster-Shafer theory for evidential reasoning [7]. Similarly, Effective Voting [138] represents a method standing between the use of all models (stacking, voting, etc.) or a single model (Evaluation and Selection), which selects models to combine based on pair-wise significance tests. How these differing opinions are managed plays a primary role in team success.

Stock selection and forecasting has become an application with increased machine learning efforts in recent years. Majority voting of heterogeneous linear and non-linear classifiers has been shown to marginally improve accuracy and profitability, with greater gains resulting from enforcing unanimity among all classifiers before a decision takes action [9]. Applying machine learning algorithms to human sciences is another currently active research field, with works advancing the use of support vector machines and kernel machines in general. Other applications of combining multiple learners include land cover type mapping with spatial classifiers [121], cursive word [73] and general handwriting recognition [152].

Techniques such as boosting and stacking can also be chained together to achieve high accuracy. In [47], a base-level classifier is trained, and examples that it incorrectly classifies are serially added to the next classifier's training data set. Each base-level classifier likely then has a different error rate, but when combined still represent homogeneous models even though they are largely viewed as diverse. This process continues for all base-level classifiers, their outputs are averaged, and finally combined using a cross-validated tree classifier. Results showed that this ensemble method outperformed conventional boosting and stacking. A homogeneous ensemble of decision stump learners combining the popular methods of boosting, bagging, and dagging via the sum voting methodology has also proven successful with some data sets [77]. This is an example of how heterogeneity can lead to more robust and high-performance classifiers. Our work is similar, but with the addition of collaboration during the learning process and use of purely heterogeneous team compositions.

As modern data volumes grow to be extremely large, attempts at distributed processing and parallel machine learning are gaining popularity. Further advantages are also possible, such as large-scale cooperation and scaling up the number of machine learning algorithms simultaneously running. One method to do so is to partition the data set among different network computers, each running the same rule-learning algorithm [101]. As each learner discovers an acceptable rule, it is broadcast to all other learners for evaluation or further specialization, and a list of globally consistent rules is compiled. Results using this approach demonstrated that distributed learners generated differing individual rule sets, but the integrated rule set was essentially the same as a single learner on the entire data set. Distributing the learning load also completed the process faster.

Other efforts have studied the use of training multiple classifiers on different feature subsets prior to their combination [35]. Focusing on differing and potentially overlapping

feature subsets creates additional diversity, which could lead to an improved combined model. Some researchers also found that performance degradation occurs as the percentage of training batches (sets of training examples) overlap [135]. Multi-agent learning systems provide a mechanism for additional study on how knowledge from learners with different (potentially overlapping) feature and data subsets can be combined. Interaction between the learners during the learning process may prove to increase learning efficiency, robustness, and accuracy. Our work aims to provide insight into the effect of collaboration on team learning. An overview of classifier combination and soft computing is included in [78].

2.3.2 Overview of Combination Techniques

Various approaches exist for combining decisions from multiple classifiers. The simplest method is pure voting, where votes from each classifier are combined or manipulated to select a single resulting class. Some voting techniques perform better than others in specific situations. For example, [130] concluded that averaging is preferred when posterior probabilities are not well estimated; whereas product combination is preferred for problems involving multiple classes and good estimates of posterior class probabilities. Other popular methods incorporate data manipulation techniques to expose learners to different portions of the training data, or provide them with potentially complementary information. More advanced methods are accuracy-driven, or involve additional learning algorithms which learn the mapping between the outputs of multiple classifiers (the decision sequence) and the correct class. This is typically termed meta-learning. Popular combination methods for each of these categories are listed in this section.

Pure Voting Methods

Pure voting methods are among the simplest ways to combine decisions from multiple classifiers. Each entity classifies (votes) that an instance it is being provided belongs to one of multiple classes. These votes are tallied in one of a number of ways to arrive at the final decision of the classifiers being combined in the form of a single class. Some of the popular pure voting methods are:

Maximum of predictions: Selects the class with the absolute maximum probability value out of all learners.

Average of predictions: Averages all predictions for each class from all learners, and selects the class corresponding to the highest average value.

Sum of predictions: Sum of class probability values from each classifier, where the class receiving the highest cumulative probability is selected for classifying the example.

Product of predictions: Multiplies all prediction values for each class from all learners and selects the class corresponding to the highest resulting value.

Majority vote: Occasionally termed Plurality Voting, each individual classifier votes, and the class receiving the most votes is used for classification (majority voting means more than 50% of votes).

Unanimity: In some situations (e.g., stock forecasting), all classifiers must be in agreement on the predicted class or the instance is rejected.

Oracle: Typically used as a comparison classifier, predicting correct class if any of the classifiers predict the correct class.

Accuracy-Driven Voting Methods

Accuracy-driven voting methods are those that take into account the accuracy, confidence, or probability of each learner offering its vote. This moves a step beyond simple voting, where votes from classifiers that have exhibited strong classification accuracies are weighted more, and weak classifier votes are either discarded or weakly weighted. This category also includes meta-learning, where a separate classifier is trained on the predictions of a group of classifiers. Its goal is to model the classifier group's prediction pattern for a training set, so that it can be used for future unseen data. Some of the popular accuracy-driven voting methods are:

Weighted voting: Each classifier is assigned a voting weight (usually proportional to its accuracy), and the class with the highest weighted vote is used for classification.

Accuracy on training/testing data: Only those classifiers which equal or outperform the team's average accuracy on the training or testing data are used for voting.

Select best: The classifier with the highest accuracy on the training data is used for all final predictions.

Rank-based: Rank or weight individual classifier outputs by how well they perform on a hold-out test data set (higher rank equals better performance, which is the chosen class). Similarly, a classifier is selected that correctly classifies the most consecutive neighboring training samples (considered to have the highest rank).

Selective fusion: Select a subset of classifiers using statistical measure of error/performance, and use a simple combination method to combine the decisions.

Local feature space accuracy: Combine classifiers based on their local accuracy in feature space around a test instance. The classifier with highest local accuracy is used for instance, where local can be defined as nearest neighbor.

Maximum entropy: The reliability of experts is related to the success rate of each expert (i.e., the probability of taking the correct decision). The correlation of experts is related to the probability that experts agree. The maximum entropy approach tries to find the probability density that has maximum entropy.

Evidence: Combine classifiers using Dempster-Shafer theory of evidence, which is focused on individual classifier beliefs and combining these beliefs for all classifiers.

Data Manipulation Methods

Data manipulation methods, which manipulate the training and testing data to attempt to achieve optimal classification accuracy, are by far the most popular means to train and combine multiple classifiers. They have seen widespread use due to their simplicity and mathematical background, and have produced some of the best accuracies on challenging machine learning data sets. Some of the popular data manipulation methods include:

Input decimation: Classifier correlation can be reduced by purposefully withholding some parts of each pattern (i.e., only using a subset of the features for certain classifiers). Feature inputs can be "pruned" by measuring how each affects the classifier output. Those features that have the least effect on the output can be selected for removal without compromising overall classifier performance. One example is to have one classifier per class, where each classifier only uses the features with high correlation with that particular class.

Boosting: Based on results of previous classifiers, diverse training samples are setup such that instances that were wrongly predicted by previous classifiers play a more important role in training (i.e., further learning focuses on difficult examples). This method is based on multiple learning iterations. At each iteration, instances incorrectly classified are given greater weight in the next iteration. Thus, the classifier in each iteration is forced to concentrate on instances it was unable to correctly classify in previous iterations. All classifiers are then combined after all iterations have been processed, or a threshold is met. Combination occurs by weighing the models according to their estimated error rate. This method adaptively changes the training set distribution based on performance of previous classifiers, attempting to reduce both bias and variance.

Bagging: A family of classifiers is created by training on different portions of the training set. N training "bags" are initially created, each obtained by taking a training set of size S and sampling the training set S times with replacement. Some instances could occur multiple times, while others may not appear at all. Each bag is then used to train a classifier, which are then combined using an equal weight for each. This method changes training set distribution stochastically. It has been concluded that learners must be responsive to changes in training data for this technique to be effective.

Dagging: Creates a number of disjoint, stratified folds out of the data and feeds each chunk of data to a copy of the supplied base learner, while predictions are made via majority vote. Techniques like bagging and dagging reduce variance and are more robust in noisy settings.

Ensemble Learning, Meta-Learning, and Other Abstract Methods

One of the most active areas in supervised learning is constructing successful ensembles of classifiers. Ensemble methods are learning algorithms that construct a set of classifiers and then classify new data points by taking a (weighted) vote of their predictions [41]. Ensembles are well-established as a method for obtaining highly accurate classifiers by combining less accurate ones. They can often outperform any single classifier, which represents a motivating factor for our research involving multiple concurrent learners utilizing different learning methods. However, to be more accurate than a single classifier, the ensemble must be composed of both accurate and diverse classifiers [41].

Similar to ensemble methods are those that involve the strategic sequencing of machine learning algorithms together, which can be a learning event in and of itself. For example, [38] utilized a library of machine learning algorithms to autonomously select and sequence appropriate algorithms. Their method involved three steps: (1) determine cause of performance failure, (2) post learning goals and change background knowledge, and (3) use planning techniques to assemble a sequence of algorithms to achieve learning goals. However, no study was conducted on which learning algorithms consistently appeared in successful sequences, or which learning algorithms never appeared in sequences. Thus, they were concerned with sequences of learning algorithms rather than concurrent heterogeneous learning. They concluded that sequencing of learning goals was necessary to avoid negative interactions between learners, which can lead to less effective learning. We are similarly concerned with forming a team of learners that work well together and maximize performance of a classification system for a task. Our methodology can be used to determine which learning algorithms work poorly together and those that predominantly perform well together.

Use of a single learning paradigm does not always produce the most accurate result. One approach is to try many different learners separately and use the single learner which performs the best for the task. Some learning paradigms may produce more accurate results than others. Since different learning methods typically converge to different solutions, learning based on different paradigms can be combined for improved accuracy [11]. Further, better accuracy can be achieved if different learners use different input representations, such as the use of different types of sensors. Learning multiple models of data has also been shown to reduce generalization error compared to learning a single model of the data [10]. Accuracy can be affected by the manner in which the results are combined, typically via voting schemes or a separate learner (called a meta-learner) that is tasked with learning how to best combine the knowledge. For instance, one of the approaches is to make separate predictions using different sources and combining those predictions.

Meta-learning is additionally presented as a general technique to integrate multiple classifiers on a single learner [34]. Meta-learning can be defined as learning from information generated by one or more learners. Learning can be performed in parallel (using a single

thread per learning method, for example) to facilitate near real-time learning. As an example, [34] separates data into subsets and uses the same learner on each subset in parallel. Our research aims to employ multiple different learners that concurrently learn subsets of the data and share to increase their performance and discover patterns which a single learner cannot.

Some popular ensemble, meta-learning, and other abstract techniques are:

Stacking: Sometimes termed stacked generalization. Uses a meta-level classifier that learns a mapping from predictions by individual classifiers for a given true class of an example. Other popular variations of stacking involve regression and decision trees.

Multi-stage and multi-level hierarchical classifiers: Serial sequencing of classifiers which build upon learned model from previous classifiers. This also includes classifier systems that are hierarchical, or tree-like in structure, containing one or more classifiers at each level of the hierarchy.

Trust-based meta-level classifier: A meta-level classifier predicts whether a base-level classifier is to be trusted. Meta-level attributes are probabilities of each of the class values returned by the base-level classifiers. They are said to thus include confidence.

Behavior-knowledge space approach: A behavior-knowledge space is an N-dimensional space, where each dimension corresponds to a classifier's decisions. Each unit corresponds to the intersection of the individual classifiers, and accumulates the number of training samples from each class. The unknown sample is assigned to the class with the most training samples in that unit.

Genetic algorithms: Genetic algorithms (GA) have even been used to determine how to best distribute feature subsets to specific classifiers for ensemble learning. For example, the GA chromosome can encode the number and types of classifiers to use as well as the features each should use. The GA can then evolve the classifier and feature subset population to consist of highly accurate individual classifiers, with the fitness function focusing on both the individual and group classification levels.

(K-1)-of-K combination: This method is similar to K-fold cross-validation. K different classifiers are constructed by training one using each fold's training set, thus training the classifiers on sets that are slightly different. This aims at producing classifiers that are less correlated, compared to the situation where each classifier was trained on the full training set.

2.4 Utilized Machine Learning Algorithms

2.4.1 Reinforcement Learning

Reinforcement Learning is a machine learning algorithm aimed at forming a *policy* for an agent based on attempting to maximize a reward. This style of learning is also used for

children and pets, for example, to teach them right from wrong. This algorithm is central to the notion of positive and negative reinforcement, where positive reinforcement is given when actions are performed correctly (or improved on a task) and negative reinforcement is provided in cases where an action was incorrect (or not improved on a task).

The algorithm is based on associating specific actions to states or situations of its world or environment. Accordingly, knowledge is represented and adjusted as situation-action pairs. The environment is typically discretized in terms of time and action space, modeled as a finite-state Markov Decision Process (MDP). Each state and action at specific times are described by a state transition probability (state-action weights) for a Markov chain involving maximizing or minimizing a reward function.

In general, reinforcement learning is a primitive form of learning. Because of its structure and reinforcement nature, reinforcement learning is more suitable for long-term rather than short-term learning applications. This is due to rewards being potentially delayed and the time it takes to converge to a sufficient policy. This policy can be thought of as the current state of the machine's knowledge, which is updated and adapted over time by adjusting state-action probabilities based on the utility measure. Policy (knowledge) convergence is typically measured by how frequently an agent's policy changes. Nash equilibrium occurs when a collection of strategies for each agent represents a best-response to the other agents' strategies [153].

Reinforcement learning has various implementation strategies and flavors. The most popular variant is Q-learning [144]. Q-learning involves learning an action-value function for encountered situations/states. The value for each action represents a measure of utility of executing that action for the accompanying world state. The value of each state-action pair is called its Q-value. This allows possible actions to be compared to previously encountered situations without modeling the environment. In this manner, the policy can be used at each step to choose the action that currently maximizes (or minimizes) the utility function. Reinforcement then repeats for all actions until the policy becomes stable (reliable), or when learning is halted. The reinforcement schedule can vary in frequency, from rewarding after each action or in a delayed manner [144].

The algorithm can be more formally expressed via the following equation [144][145]:

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \alpha_t(s_t, a_t)[r_t + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

where s = state, S = set of states, a = action, A = set of actions, r_t = reward at time t, $\alpha_t(s, a)$ = learning rate [0..1], and γ = discount factor [0..1).

In contrast to supplying local (individual) rewards, global reinforcement for a group of agents can also be performed. As in [137], a reinforcement function can be utilized to measure the performance of a team of robots as a unit. The reinforcement function value is distributed to the entire team, which represents the only group information available to the members. This can then be used to help individual members alter their behaviors toward

the direction of better team performance. Others have rewarded locally or globally upon the completion of a goal [127].

Our research utilizes local reinforcement functions that contribute to choosing an action in certain environmental situations, rather than a global reinforcement function. We believe that this better allows the agents to specialize their behavior based on their experience. This specific experience can then be shared with other agents to adjust their behaviors and therefore potentially learn from an agent utilizing knowledge which was a result of local reinforcement. This in turn may help the team achieve better performance compared to those which use strict global reinforcement. The reinforcement learning technique employed in our research also utilizes a form of delayed reward [144], as a history of actions is kept and reinforced in a delayed manner back in time from the current action. This can be thought of an additional sub-reward on top of the original reward provided by the update equation.

Varying the learning rate of reinforcement learners also represents a viable option, where the rate can be adjusted based on the agent's current performance. Such an approach is the WoLF (Win or Learn Fast) principle [23][22], which entails learning quickly when losing and learning cautiously when winning. The idea behind this is that the agent should adapt quickly when doing poorly, but should be cautious when winning as other agents are likely to change their policies. This is the case if focusing on agent behaviors with respect to other agent behaviors. They trained all players using the same learning method, and randomly executed players' actions to introduce a non-deterministic element.

A reinforcement learning system can be expected to create more effective rules than could be predetermined by simple heuristics. A reinforcement learning system creates rules based on carefully selected situation conditions. Therefore, the knowledge of the system designers is embedded in the system through the definition of the situation. This gives the system the ability to create rules based on a richer set of preconditions than considered by simple heuristics, and thus the ability to create rules that can differentiate effective action in a wider variety of situations. A reinforcement learning system also modifies rule outcomes based on experienced failure and success. This provides the system the ability to correct rules that have been designated an ineffective outcome by discovering more effective actions. Therefore, rules are given the flexibility to change if experience implies they should.

2.4.2 Artificial Neural Networks

Artificial Neural Networks (ANN) are computational models consisting of a network of interconnected neurons. The original concept for this approach is inspired by attempts to analyze and model the human brain, focusing on its nodes/neurons, dendrites, and axons. They are typically used to model, discover, or learn patterns in complex relationships between inputs and outputs. They operate on a connectionist approach, where neurons are fired, and their output is adjusted by weighted connections in route through the network

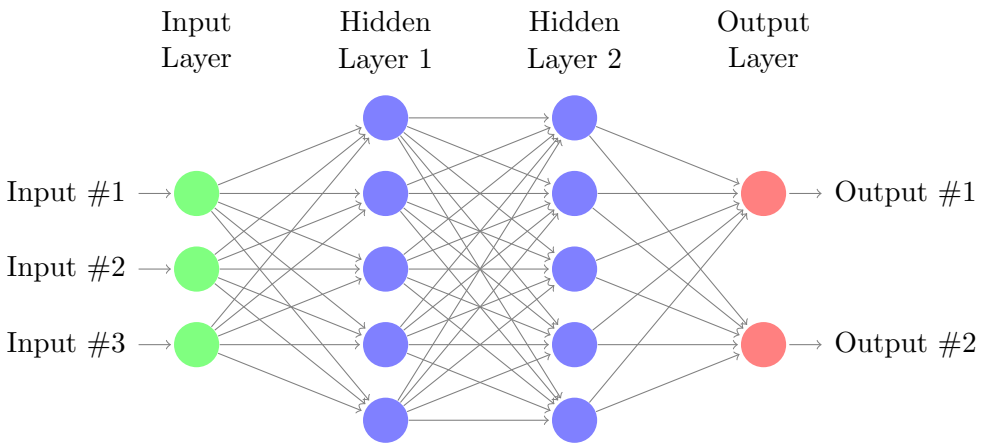

Figure 2.2: Fully-connected neural network configuration with three input neurons, two hidden layers with five neurons each, and two output neurons.

via other neuron interfaces and layers.

Neurons represent simple processing entities which, when connected in concert with other neurons, can exhibit complex global behavior. Each neuron utilizes a transfer function, which it uses to adjust input for forwarding through connections to other neurons in subsequent neuron layers. A neural network is typically broken down into layers of neurons which behave similarly in terms of transfer function or connection interfaces. There is typically an input layer representing the nodes for inputs into the neural network, some hidden intermediate layer(s) containing one or more neurons each, and an output layer which produces the outputs from passing the input through the neural network. Figure 2.2 demonstrates an example fully-connected neural network configuration, consisting of three input neurons, two hidden layers with five neurons each, and two output neurons.

Neural networks are trained by repeatedly providing the network with inputs and iterating until the error goal for network output has been achieved, or a certain iteration threshold has been met. This occurs by providing the network with outputs corresponding to the inputs that it should attempt to learn. The network learns the association between inputs and outputs by passing each input through the network and calculating the error the network produces with its current connection weights. This error is then typically backpropagated to adjust connection weights in reverse, from output nodes to input nodes. By adjusting connection weights, the network can then adjust its output for given input. This process is repeated until the convergence goal is achieved. The connection weights of the network therefore represent the neural network's knowledge, given the situations it has seen and the output it has been provided to train toward.

One advantage and overall property of neural networks is that they can have static or varying structure, be composed of any number of interconnected neurons and layers with various transfer functions, and trained for specified periods of time. This allows them to be tailored to specific applications, adjust and learn extremely complex and nonlinear

problems, and also allow dynamic restructuring of the network for adaptive purposes. The neural network topology used in our research, however, remains fixed as we want to investigate only its use with other learning algorithms.

Another advantage of neural networks and similar classifier systems are that they can be used to predict the output for unseen inputs. This is a powerful property, as the neural network can be further used to find inputs that are similar to others as well as use what it has currently learned to classify new problems. For example, the drug Taxol is known to be an anti-cancer drug that has seen some success. By training the neural network the properties of Taxol as a drug, other drugs can be passed through the network to see how close they relate to Taxol in hopes that another anti-cancer drug can be found or designed. The models can also be adjusted to account for fuzzy and/or noisy input. Overall, neural networks do suffer occasionally from problems of getting stuck in local minima. However, their ability to learn complex information outweighs these disadvantages.

As neural networks represent a learning technique, notions of learning rate, momentum rate, and decay rate can be introduced. As with other learning methods, the learning rate represents the speed or ratio at which knowledge values (or weights) are adjusted. The higher the learning rate, the more aggressive the learning becomes. However, higher learning rates can lead to local minima. The momentum rate can help avoid local minima by pushing over/through it. Momentum utilizes a history of change direction and rate for additional adjustment of knowledge values or weights. However, momentum must be used carefully as it can cause the neural network to be trained very poorly if momentum is too high, or can cause the learning process to skip over optimum portions of the search space. The decay rate represents the rate at which weights adjust (or decay over time) toward 0. Higher decay rates cause weights to decay faster, whereas lower decay rates cause weights to decay slower. A decay rate of 0 causes no weight decay to take place.

Neural networks are a popular learning method in classification research. For example, [55] presents an approach in which a simulated mobile robot learns a navigation task by imitating a teacher, being further reinforced throughout the learning process. A constructive high-order neural network was used to represent behaviors as situation-action rules. The neural network learns incrementally and constructively, where new units are added when a connection weight should differ in certain circumstances. Two overall networks are employed, one for long-term memory (learns from only positive experiences) and one for short-term memory (keeps track of unsuccessful actions). Results show that temporal transition hierarchies can be used to integrate several robot learning techniques. In this case, however, this only supports using one learning technique at certain points in time; learning methods are not concurrently learning. They also found that transition hierarchy networks are very sensitive to noise while learning, where more units were created than necessary. Further discussion and comparison of results between Q-learning and classifier systems are included in [110].

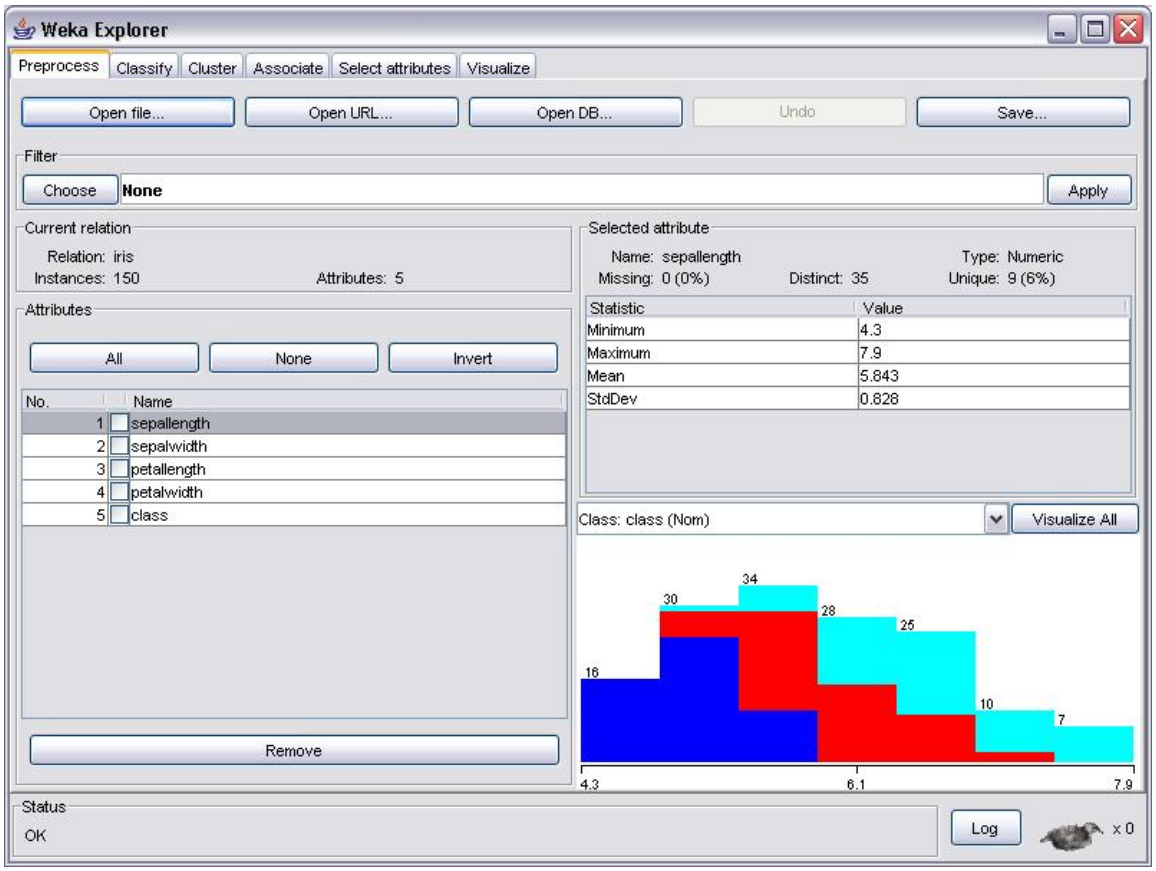

Figure 2.3: An example user interface of the WEKA Explorer.

2.4.3 WEKA Machine Learning Suite

WEKA, developed and managed by the University of Waikato, is a comprehensive open-source set of object-oriented machine learning and data mining algorithms written in Java [147]. In addition to classification, regression, clustering, and association rule algorithms, it also provides methods for data filtering, pre-processing, and visualization. The set of packages provides interfaces to preprocessing routines including feature selection, categorical and numeric learning tasks, performance enhancement of classifiers, evaluation to different criteria such as root mean squared error, and experimental support for verifying the robustness of models. Functionalities such as training, testing, and using classifiers are provided for all algorithms. Additionally, all algorithms are implemented using the same software architecture, so comparing, combining, and cross-validating algorithms is straightforward. WEKA has been extensively used in machine learning research to develop new techniques and compare them to the state-of-the-art. Figure 2.3 shows an example of the WEKA graphical user interface. The library also offers methods for command-line usage, examples of which are illustrated in Figure 2.4.

```
//Training a C4.5 decision tree learner using a weather data set
> java weka.classifiers.trees.J48 -t weather.arff -i

//Training and testing a Naive Bayes classifier on a soybean data set
> java weka.classifiers.bayes.NaiveBayes -t train.arff -T test.arff -p 0
```

Figure 2.4: Examples of WEKA's command-line execution.

Naive Bayes

Statistical and probabilistic approaches to machine learning have been used for many years, achieving greatest success in the medical domain. Several such approaches have become popular, such as Bayesian networks and the Naive Bayesian Classifier, due to their simplicity and clear probabilistic semantics [64].

For example, the Naive Bayes classifier operates on the assumption that numeric attributes result from a single Gaussian (normal) distribution. For real-world problems, this approximation is unlikely to be accurate, but offers reasonable success for real-world problems that is comparable to decision trees and other induction methods. It also assumes the predictive attributes are conditionally independent given the class, working on the assumption that no concealed attributes influence prediction.

Naive Bayes prediction typically takes place using the following process, where C represents a random variable indicating an instance's class, A represents a vector of random variables indicating the attribute values that are observed, c represents the label of a class, and a represents a vector of observed attribute values:

- Use Bayes' rule to compute the probability of each class given a:

$$p(C = c|A = a) = \frac{p(C = c)\prod p(A_i = a_i|C = c)}{Z} \tag{2.1}$$

- Predict the most probable class

The product in the numerator is due to the conditional independence assumption, and the denominator is a normalization factor so the sum of $p(C = c|A = a)$ overall classes is 1. Normally, discrete attributes are modeled using a single probability value, while numeric/continuous attributes are modeled using a continuous probability distribution. A popular density estimation method is a single Gaussian. Kernel density estimation has been proposed so that continuous variables can be better estimated [64], especially in the case where the distribution is multi-modal rather than normal. In our research, we make use of the Naive Bayes classifier with kernel estimation for modeling numeric attributes.

Instance-Based Learners (K-Nearest Neighbor and K*)

Some algorithms make use of specific instances rather than precompiled abstractions (trees, networks, etc) during prediction. These methods are categorically termed instance-based learning or example-based learning. They have a strong underpinning as a nearest neighbor method, and generally use some form of similarity measure to determine "distance" or "match" between instances [1]. One of the more popular instance-based learning algorithms is the K-nearest neighbor classifier, which utilizes the K most similar (nearest) training examples in various ways for prediction. In general, these algorithms work on the assumption that similar instances have similar classes.

There are several aspects of nearest neighbor algorithms that make them problematic for some domains and data sets, including:

- Computationally and memory/storage expensive (store or measure all instances)
- Sensitive to noise and choice of similarity/distance function
- Sensitive to choice of classification approach (decision process using measures from many neighbors, such as voting)
- Offer little information regarding structure of the data set
- Create challenges associated with missing and nominal attribute values

An advantage of such methods is their simplicity, which make them applicable to many domains and problems. In our research, we employ a k-nearest neighbor classifier that uses K=5 of the nearest neighbors for prediction, weighting the neighbors by the inverse of their distance when voting.

K* is another flavor of instance-based learner that utilizes entropy (motivated by information theory) as a distance measure [36]. This allows for a more consistent approach to handling missing values and symbolic and real attribute values. The K* distance method's general approach is similar to work done in comparing DNA sequences, where the distance between two instances is computed as a measure of the sum of all possible transforming paths (represented by transformation probabilities). The selected number of instances is termed the "sphere of influence", which dictates how the instances are weighed. During classification, the category corresponding to the highest probability is selected. We make use of this K* algorithm in our research with default behavior.

Logistic Regression

Logistic regression is a statistical model that predicts the probability of an event occurring by fitting a logistic curve to the data. Several predictor variables are typically used to describe the relationship between risk factors and outcomes. Logistic regression has seen use in medical and social fields, including customer trending.

Logistic regression is based on the logistic function:

$$f(Z) = \frac{1}{1 + e^{-Z}} \tag{2.2}$$

where Z represents the input (set of risk factors) and the returned value represents the output (probability of a particular outcome). Z normally takes on the following form:

$$Z = a + b_1 x_1 + b_2 x_2 + ... + b_n x_n \tag{2.3}$$

where a is the intercept (where all risk factors are zero) and b_i represent the regression coefficients (risk factor contributions). Positive coefficients translate to that specific risk factor increasing the probability of an outcome; while negative coefficients translate to a decrease in probability of an outcome. Similarly, larger values correspond to higher influence. Our work utilizes a multinomial logistic regression model with a ridge estimator as described in [82], which iterates until convergence.

Decision Table

Another rule learning method is the decision table classifier with a default rule mapping to the majority class [75]. A decision table has two main components: a schema (set of features) and a body (set of instances with feature values and class labels – the training set). An optimal feature set is determined by transforming the problem into a state space (feature subset) search, using best-first search with the heuristic being k-fold cross-validation to estimate the future prediction accuracy. Incremental cross-validation can be used to provide a speedup for algorithms which support incremental addition and deletion of rules [75].

Prediction for a test instance takes place by finding all labeled training examples that exactly match the features of the test instance. If this set is empty, the majority class of the entire training set is used for labeling the test instance; otherwise, the majority class of the set of matching training examples is used for labeling. Decision tables are more ideal for discrete attributes, as they attempt to find exact matches of the feature set for an instance for class assignment. A perfect match would be highly unlikely for continuous attributes in practice, but may be more likely if the number of values is low.

Also, as the accuracy prediction method is heuristic-based, the best feature set may not be found. For some domains, the decision table was found to perform comparably with the C4.5 decision tree algorithm. We utilize the decision table in our research using five fully expanded non-improving subsets before terminating best-first search and leave-one-out cross-validation.

Decision Tree

Decision trees are one of the most well-known machine learning methods, and have seen successful use in a variety of real-world problem domains (e.g., diagnosis). Their utility as a simple and fast learning approach is widely accepted. Many decision tree algorithms and variations exist, the most popular of which are Quinlan's ID3 and successor C4.5 [102]. Decision trees produced by these algorithms are typically small/shallow and accurate, making for fast and reliable classification.

The process of building a decision tree begins with a set of example cases or instances, each containing a series of numerical or symbolic attributes and a membership class. Each internal node of the tree represents a test which determines the branch to travel down. For example, if an internal node's test is "$x > 42$" and x is 30, the test returns False and proceed down the right branch of the tree at that node (tests returning True proceed down the left branch). The tree's leaf nodes represent the possible classes which instances can belong to, which is the prediction produced by the tree for test instances.

C4.5 utilizes formulas based on information theory (information gain and gain ratio) to measure the "goodness" of tests for nodes. Thus, tests are chosen that extract the maximum information given a single attribute test and the set of cases. Overfitting is reduced by estimating the error rate of each subtree and replacing it with a leaf node if its estimated error is smaller, a process termed "pruning" [102]. Finally, the decision tree can be transformed into a set of rules by traversing the tree paths from the leaf nodes. The resulting set of rules can then be simplified in a variety of ways (and some rules completely removed) to arrive at a final set of rules for classification. In our research, we make use of the C4.5 Decision Tree classifier, and found the unpruned tree to offer higher testing accuracy for the studied data sets.

Random Forest

Rather than using a single decision tree for classification, many of them can be used in conjunction. Such a method is termed a random forest [27]. Random forests have their roots in work involving feature subset selection, random split selection, and ensemble classifiers. A random forest is constructed by selecting a random feature vector that is independent of the previous chosen feature vectors (but identically distributed), and growing a tree-structured classifier using this vector and the training set. This is done T times to create a random forest of T trees, each which independently votes on a class. The most popular class from all trees is used to label the test instance.

Two primary measures are used to determine the accuracy and inter-dependence of the trees both individually and as a whole. The goal is to minimize tree correlation while maintaining individual tree strength. The accuracy of random forests, which has been found to be comparable to and sometimes better than AdaBoost, depends on the strength of the

individual tree classifiers and the correlation/dependence between them [27]. Typically, choosing one or two random features to grow each tree can provide near optimum results. The following characteristics are favorable [27]:

- Expected to always converge, so overfitting is less of an issue
- Relatively robust to outliers and noise
- Provides estimates of correlation, feature importance, and strength
- Faster than boosting and bagging
- Is simple, fast, and easily parallelized

Random forests allow for any number of trees, which in some cases can be in excess of 100. We utilize a random forest consisting of 10 tree classifiers and consider $logM + 1$ features (where M is the number of inputs).

Rule Learners (RIPPER and PART)

Rule learning systems offer several desirable properties, including their human understandable format and general efficiency. However, they are known to not scale well in relation to data size. One dominant rule learning variant is the Repeated Incremental Pruning to Produce Error Reduction (RIPPER) propositional rule learner [37]. RIPPER is based on iterative reduced error pruning (IREP), which for rules involves the training data being split into a growing set and a pruning set. A rule set is constructed, one rule at a time, in a greedy manner. When a rule is constructed, all examples that are covered by the rule are deleted via a pruning method.

The uncovered examples are randomly partitioned into two subsets: a growing set and a pruning set. A rule is grown by adding conditions that maximize an information gain criterion until the rule covers no negative examples. The rule is then pruned by deleting condition(s) that maximize a pruning function until the function's value cannot be improved by any deletion. This process is repeated to generate, prune, and optimize the rule set until there are no positive examples remaining, or a rule is found with a large error rate.

RIPPER is competitive with C4.5 rules in terms of accuracy, and is able to efficiently operate on very large (hundreds of thousands of examples) and noisy data sets. In [37] more details are provided on RIPPER's performance and comparison to C4.5. We utilize RIPPER in our research as implemented in the WEKA machine learning suite [147].

Another successful rule learning algorithm is PART [50]. PART is a decision list algorithm based on partial decision trees, combining the advantages of both C4.5 and RIPPER while eliminating some of their disadvantages. It is considered a separate-and-conquer algorithm, meaning that it generates one rule at a time, remove covered instances, and repeats. PART differs in the general construction of rules by creating a pruned decision tree for all current instances, builds a rule corresponding to the leaf node with the largest coverage, then discards the tree and continues. Partial decision trees are constructed in

a manner similar to C4.5. This is said to help avoid the problem of generalizing without knowing its implications [50].

PART avoids global optimization that is present in C4.5 and RIPPER, but still produces accurate rule sets. In comparison, PART compares favorably to C4.5 in accuracy and outperforms RIPPER at the cost of a larger rule set. We utilize the PART algorithm with reduced error pruning (three folds), a confidence threshold of 0.25 for pruning, and a minimum of two instances per leaf.

Chapter 3

Research Methodology

In this chapter, the research methodology of this work is described, along with the experimental setup and evaluation criteria that were utilized. The following three chapters each present a unique application partaining to the research methodology, and each will be used to study specific aspects of team learning and collaboration. Experimental results are presented in each chapter.

3.1 The Multi-Agent Collaborative Learning Architecture

The presented collaboration methodology has its roots in other machine learning techniques such as boosting [47], bagging [77], and dagging [77]. The following aspects are inherently embedded in our approach, which are integrated from motivation in the literature that has shown the individual utility of each:

- Randomized training sets are utilized in certain operation modes
- Misclassified instances are repeated as collaboration and learning continues
- Instances are dispersed/distributed to different learners
- Independent learning of the data set, where learners focus on developing local models of disjoint portions of the data
- Training on multiple instances of challenging examples to better learn them
- Classifier ensembles
- Decision combination and weighted methods based on classifier accuracy estimates

Boosting is a technique based on results of previous classifiers, creating diverse training samples, such that the instances that were wrongly predicted by previous classifiers play a more important role in training (i.e., further learning focuses on difficult examples). Boosting adaptively changes the training set distribution based on performance of previous classifiers. Such an approach attempts to reduce both bias and variance.

Bagging creates a family of classifiers by training on different portions of the training set. N training bags are initially created, where each is obtained by taking a training set

of size S and sampling the training set S times with replacement (some instances could occur multiple times, while others may not appear at all). Each bag is then used to train a classifier. Thus, this technique changes the training set stochastically. An extension of this would be bootstrapping, which creates a series of patterns using random sampling with replacement. Dagging, on the other hand, creates a number of disjoint, stratified folds out of the data and feeds each fold to a copy of the supplied base learner, while predictions are made via majority vote. Therefore, techniques like bagging and dagging reduce variance and are more robust in noisy settings.

The WEKA machine learning suite [147] is utilized as a base for the implementation of our proposed Multi-Agent Collaborative Learning Architecture. WEKA is implemented as a series of Java classes, allowing for manipulation and modeling of data via a variety of machine learning algorithms. The main uses of WEKA within the Multi-Agent Collaborative Learning Architecture are to:

- Convert data from CSV format to WEKA's ARFF format
- Normalize and remove attributes from data sets
- Split and randomize data sets, including creation of stratified folds
- Perform cross-validation
- Provide implementations of various machine learning algorithms

Additional Java and script/batch file implementations act as a wrapper for machine learning experiments involving single or multiple learners (i.e., for teams of any size), homogeneous or heterogeneous team composition, independent or collaborative learning (with the ability to vary the number of times learners collaborate during learning), learning with independent portions of or the full training set, and combining the decisions of the learners using a variety of vote-based combination techniques. This offers a robust and flexible architecture for machine learning experiments and studies involving multiple, collaborative learning agents.

The user is abstracted from all of the low-level details through a single command-line call, specifying:

- Single training file
- Single testing file
- Number of collaboration events
- Path to folder for storing results
- Attribute to consider the class for training and testing
- Specify use of independent or full portions of the data set for the learners
- List of WEKA learning algorithms and the options for each

A series of experiments can be specified in a batch file, and the Multi-Agent Collaborative Learning Architecture runs all specified experiments, stores all of the results in the specified folder, and produces a results summary file to easily compare training and testing results for learning teams of varying size and composition. All intermediate/incremental results, models, and data files are stored in the specified folder. The architecture is flexible in terms of the number of learning agents and the learning algorithms that they represent. If only a single learner is specified, a dedicated run is performed with just that learner on the entire (full) training data set. Once trained, the full training and testing data sets are separately passed through the learner's model, and statistics regarding its performance are collected. The Multi-Agent Collaborative Learning Architecture is described in more detail in this chapter.

3.1.1 Full versus Independent Learning Experiments

If a ***full*** experiment is specified, each learner is provided a randomized version (different order of instances for each learner) of the full training data file as its training data set. If an ***independent*** experiment is specified, each learner is provided an independent stratified portion of the training data file as its training data set. In this mode, the union of all learner training sets equals the entire training data file. In an "independent" experiment, the agents will benefit by collaborating with the other agents, as it represents a purely distributed learning problem. In "full" experiments, each learner is provided with all of the instances, focusing more on fusing low-level details learned by the participating learning algorithms.

The merits of the Multi-Agent Collaborative Learning Architecture are revealed when multiple learners are utilized with multiple collaboration events. In these situations, the training data are split into stratified files based on the number of learners and number of collaboration events. Stratified means that each separate file has the same number of instances for a class, with each file also containing instances from each class (where possible). Thus, each stratified file is constructed to be as close to the same size as possible. This is analogous to how folds are created for K-fold cross-validation.

First, a stratified training file is created for each learner, based on the selection of "full" or "independent" and numbered by learner ID. For example, if there are four learners and six requested collaboration events, there will be one training file per learner created from the provided training file and numbered from 1 to 4. Each learner's training file is then split into stratified portions, one portion per requested collaboration event and numbered by collaboration event number. Thus, in our example, each learner's training file (e.g., learner 1) is split into six stratified portions and numbered from 1 to 6. The single stratified training file that these portions was created from is then no longer needed. In this example, each learner would have a total of six training files, one for each collaboration event. If no collaboration events are requested, each learner is trained on its single stratified training

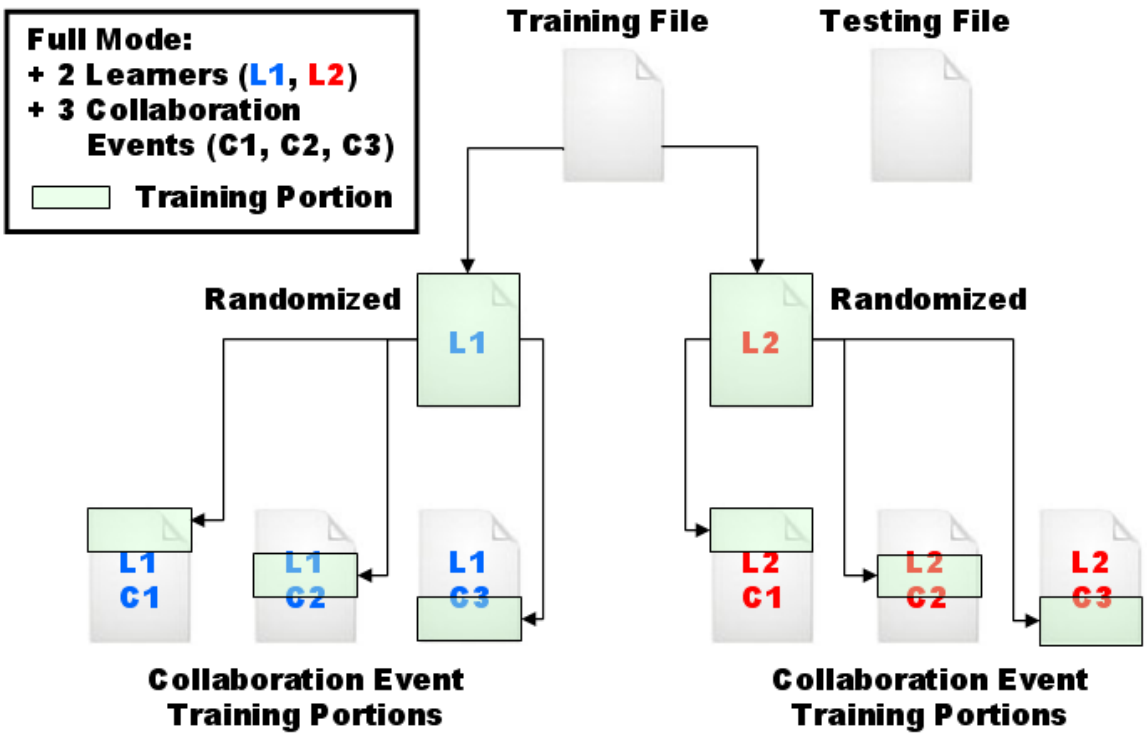

Figure 3.1: Training setup for full learning experiments.

file to build a local model of the data.

Training takes place by all learners, one training file at a time, corresponding to the collaboration event training file number. In our example, each learner would train on its first collaboration event-numbered training file (e.g., portion 1). At this time, each learner tests its model on the data on which it was just trained. The instances that it misclassifies are collected, and pooled together with all misclassified instances from all other learners. This is the collaboration file, and represents those instances that were not easily learned by the individual learning algorithms. By pooling all misclassified instances together and providing it to all learners, each learner is informing all other learners which instances it found to be difficult to learn. This collaboration file could contain duplicate instances if full mode was requested, or if independent mode and multiple collaboration events were requested. Figures 3.1 and 3.2 illustrate the training setups for full and independent learning experiments, respectively.

The next training set is constructed for each learner separately by appending the team's misclassified instances to the previous training set, and then appending the learner's next collaboration event-numbered training portion to the result. This places the misclassified instances between the two collaboration event training portions for each learner, and creates the training file that is to be used for the next learning session and collaboration event. This simulates continuous/progressive learning that is possible in updateable versions on normal classifiers. The drawback is the extra time required to retrain on overlapping portions of

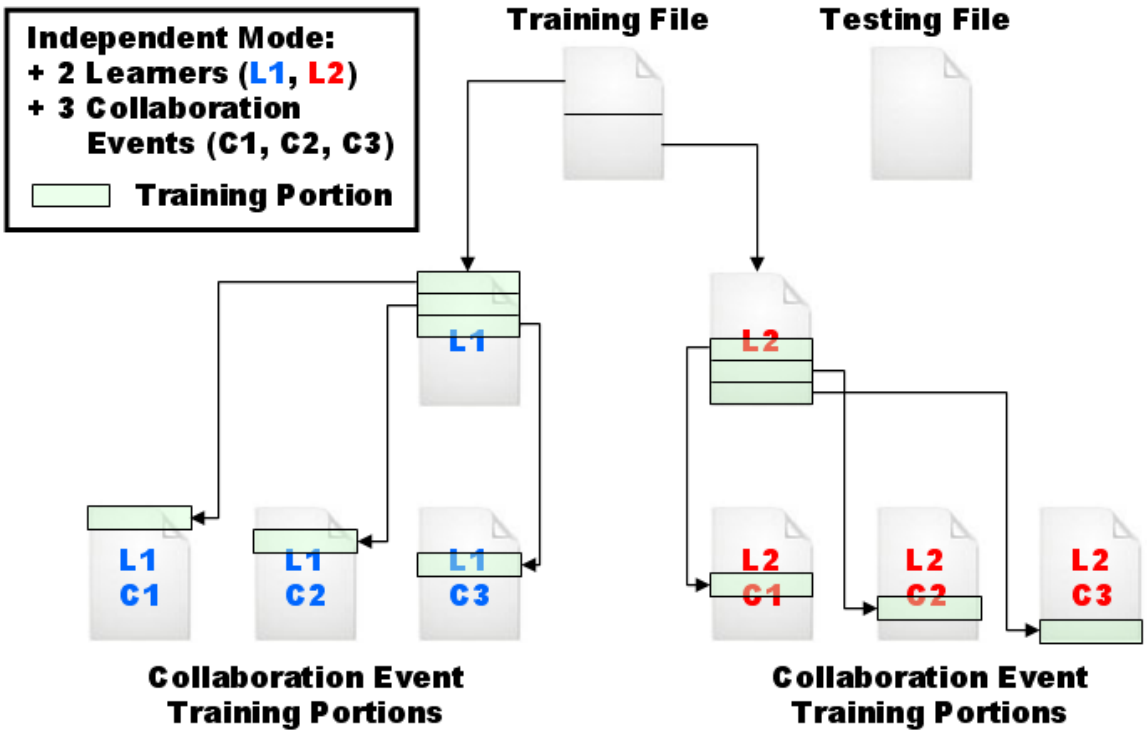

Figure 3.2: Training setup for independent learning experiments.

the training data as collaboration events are processed.

This process repeats for each collaboration event and continues until all collaboration events have been processed. Misclassified instances are only collected for the current collaboration event, as the current training set is built upon past misclassified instances and the collaboration event-numbered training portions. As collaboration events are processed and misclassified instances are collected, each learner will essentially be asking the other learners for help with instances it is having trouble with by having them attempt to learn them as well. The primary idea here is that when it is time to combine decisions from multiple learners, the consensus classification will be more accurate for the difficult instances compared to the situation where the agents do not collaborate.

Once learning is completed and incremental statistics are gathered throughout the learning process, the entire provided training and testing files are separately passed through each learner's most updated model and the corresponding class probabilities (predictions) are recorded. The final individual testing predictions are then used for combining decisions from multiple learners. This acts as the final collaboration step, which fuses the knowledge from multiple learners to a single team classifier via vote-based mechanisms. If only one learner was requested, decision combination does not take place. Decision combination is performed regardless of the number of collaboration events that are specified.

3.1.2 Multi-Classifier Decision Combination

Decision combination utilizes each learner's classification for each testing instance to arrive at a single team classification per combination method. A learner's prediction consists of a probability that the testing instance belongs to each of the possible class values. The highest probability represents the predicted class. The Multi-Agent Collaborative Learning Architecture calculates team classification accuracy for each of the 12 implemented vote-based combination methods. The combination method resulting in the best classification accuracy is selected, reflecting the overall performance of the team. The following vote-based combination methods are used for evaluation. Ties are broken by using the class selected by the learner with the highest accuracy on the testing data, unless otherwise stated.

Max: Selects the class with the absolute maximum probability value out of all learners.

Selective Max: Same as Max, but specialized by only taking into account learners that have a testing accuracy of greater than or equal to the mean of all testing accuracies from the team.

Average: Averages all predictions for each class from all learners and selects the class corresponding to the highest average value.

Selective Average: Same as Average, but specialized by only taking into account learners that have a testing accuracy of greater than or equal to the mean of all testing accuracies from the team.

Multiply: Multiplies all prediction values for each class from all learners and selects the class corresponding to the highest resulting value.

Selective Multiply: Same as Multiply, but specialized by only taking into account learners that have a testing accuracy of greater than or equal to the mean of all testing accuracies from the team.

Majority Vote: Counts the number of votes (highest class probability value) for each class from all learners. The class with the most votes wins. Ties are broken by using the first occurrence of the highest sum of testing accuracies from voters of the class receiving the most votes.

Selective Majority Vote: Same as Majority Vote, but specialized by only taking into account learners that have a testing accuracy of greater than or equal to the mean of all testing accuracies from the team.

Weighted Vote: The class vote from each learner is multiplied by its accuracy on the training data (as a percentage). All votes are then summed, and the class with the highest vote tally wins. This essentially represents the learner with the highest accuracy on the training data having the most influence on which class is selected. Ties are broken by using the first occurrence of the maximum vote value.

Selective Weighted Vote: Same as Weighted Vote, but specialized by only taking into account learners that have a testing accuracy of greater than or equal to the mean of all testing accuracies from the team.

Select Best: The learner with the highest accuracy on the training data is used for all final predictions.

Oracle: All learners are examined to determine if any of them classifies the instance correctly. If any classifier is correct, the team's final prediction is considered as correct. This is used occasionally in the literature to study how decisions can be combined to maximize accuracy when one or more participants "knows" the correct class. The Oracle measure's accuracy is always greater than or equal to the highest accuracy of the other above methods. We denote the value of 1 to represent "correct", and the value of -1 for "incorrect".

3.1.3 Experimental Result Outputs

There are several outputs which provide details (time, model information, individual testing and training accuracies, combined testing accuracy, etc) about each experiment. These results are analyzed to learn heuristics about team learning. For example, our study focuses on several aspects, including which learning methods form a good team, how large a team should be, and how often to perform collaboration events, to name a few. The following discusses these outputs in detail.

Individual and Team Models

During a team learning experiment, several models are created. The models range from each individual machine learning algorithm's model, to incremental models produced during the learning experiment, to prediction files from the team's combined model. These models represent snapshots of the individual and team knowledge at different times during the experiment. Each is saved and could be used for further classification if desired, offering the potential application of models with different ages. The following lists the primary models produced by the architecture during an experiment:

- Model before each collaboration event, for each learner **(incremental models)**
- Model for individual classification and decision combination, for each learner **(final models)**
- Combined model representing a fusion of individual team members, for each combination method **(combined models)**
- Predictions from each model, for both training and testing data files as the learning experiment proceeds **(incremental predictions)** (e.g., Figure 3.3)
- Predictions from each combination method, for both training and testing data files in full **(combined predictions)** (e.g., Figure 3.4)

Ex.	Predicted	Prob.	Actual	Probability for Each Class
1	one	0.7708	one	0.7708 0.2292 0.0 0.0 0.0 0.0 0.0 0.0
2	one	0.7143	one	0.7143 0.2857 0.0 0.0 0.0 0.0 0.0 0.0
3	one	0.7708	one	0.7708 0.2292 0.0 0.0 0.0 0.0 0.0 0.0
4	one	0.7708	one	0.7708 0.2292 0.0 0.0 0.0 0.0 0.0 0.0
5	one	0.7708	one	0.7708 0.2292 0.0 0.0 0.0 0.0 0.0 0.0
6	one	0.7708	one	0.7708 0.2292 0.0 0.0 0.0 0.0 0.0 0.0
7	one	0.7708	one	0.7708 0.2292 0.0 0.0 0.0 0.0 0.0 0.0
8	two	0.6047	one	0.3953 0.6047 0.0 0.0 0.0 0.0 0.0 0.0
9	two	0.6047	one	0.3953 0.6047 0.0 0.0 0.0 0.0 0.0 0.0
10	one	0.7708	one	0.7708 0.2292 0.0 0.0 0.0 0.0 0.0 0.0

Figure 3.3: Example of a learner predictions file, containing the class predicted by the learner, the probability assigned to this class, the actual class of the examples, and the probability associated with each class for each example. Examples 8 and 9 were incorrectly classified by this learner for this portion of this testing file.

Combination Method	Classification for Each Example
Actual:	1 1 1 1 1 1 1 1 1 1
Max:	1 2 1 2 1 1 1 2 1 1
Selective Max:	1 2 1 2 1 1 1 2 1 1
Average:	1 2 1 2 1 1 1 2 1 1
Selective Average:	1 2 1 2 1 1 1 2 1 1
Multiply:	1 2 1 2 1 1 1 2 1 1 ...
Selective Multiply:	1 2 1 2 1 1 1 2 1 1 ...
Majority Vote:	1 2 1 2 1 1 1 2 1 1
Selective Majority Vote:	1 2 1 2 1 1 1 2 1 1
Weighted Vote:	1 2 1 2 1 1 1 2 1 1
Selective Weighted Vote:	1 2 1 2 1 1 1 2 1 1
Select Best:	1 2 1 2 1 1 1 2 1 1
Oracle:	1 1 1 1 1 1 1 -1 1 1

Figure 3.4: Example of a team's combined predictions file, containing a classification for each example for each combination method. Examples 2, 4, and 8 were missclassified by all true combination methods. The Oracle method shows that for examples 2 and 4, at least one of the learners in the team correctly classified the example, but the team's overall decision using each method was incorrect. No team member was able to correctly classify example 8.

Individual and Team Statistics

In addition to the models themselves, several statistics are generated and recorded during the learning experiment for individual team members and the team as a combined unit. These statistics facilitate the comparison and analysis of individual and team performance. Many of these measures are computed by WEKA, and compiled and organized by the Multi-Agent Collaborative Learning Architecture during each learning experiment. The following lists the primary statistics recorded for each experiment, accompanied by examples:

- Misclassified instances for each collaboration event, for each learner **(the challenging instances)**
- Pre-collaboration event statistics, for each learner (e.g., Figure 3.5):
  - Training and testing statistics
    - Correctly classified instances (classifier accuracy)
    - Incorrectly classified instances (classifier error)
    - Kappa statistic: Measures agreement of prediction with the true class (1.0 being complete agreement)
    - Mean absolute error
    - Root mean squared error
    - Relative absolute error
    - Root relative squared error
    - Total number of instances
  - Detailed accuracy for each class
    - True Positive (TP) rate: Proportion of examples classified as class C among all examples which belong to class C
    - False Positive (FP) rate: Proportion of examples classified as class C (but belong to a different class) among all examples not of class C
    - Precision: Proportion of examples which belong to class C among all those which were classified as class C
    - Recall: Equivalent to TP rate
    - F-Measure: A combined measure for precision and recall, calculated using 2*Precision*Recall/(Precision+Recall)
    - ROC area: Area under Receiver Operating Characteristics (ROC) curve, which is a plot of the true positive rate (or sensitivity) versus the false positive rate (or 1-specificity)
  - Confusion matrix: Table of actual versus classified/predicted classes over all instances

- Incremental training accuracies: Accuracy for each learner on training data at each collaboration event
- Incremental testing accuracies: Accuracy for each learner on testing data at each collaboration event (e.g., Figure 3.6)
- Combination accuracies on the training file for each combination method
- Combination accuracies on the testing file for each combination method (e.g., Figure 3.7)

3.2 Experimental Setup

A tournament-style process with rounds based on team size is employed to perform a comprehensive study on all aspects of multi-agent collaborative learning. Team configurations exhibiting the highest testing accuracy move on to the next round, where larger teams are constructed from the best teams from the previous round.

The first step for a specific data set is to coarsely test a variety of individual machine learning algorithms, each with a few different settings and initialization seeds, to determine which algorithms perform well in general for the data set. The best settings and corresponding 10-fold cross-validation testing accuracy for each algorithm are recorded, and the top five algorithms are selected to advance to the next round, consisting of teams of size two.

All pairs of these top five algorithms with their best individual settings are then run, varying the number of collaboration events per experiment from None (0), Low (2), Medium (5), to High (8). This inherently includes homogeneous and heterogeneous team compositions. Similarly, the top teams of size two are selected based on testing accuracy for any of the collaboration event variations. Thus, even if one pair of learners had the best four testing accuracies, that team of size two advances to the next round, and four other pairs of learners advance as well. The same process takes place for teams of size four, where the top teams of size four are selected to advance to the final round of size eight teams. Once the round of size eight teams is complete, all results are compiled and a full study can take place.

This entire process is performed twice, once by providing all learners with the full training set and once by providing all learners independent, stratified portions of the full training set (the union of portions within a team equals the full training set). This allows a study to be performed on full versus independent individual training sets. The use of independent/portioned training sets reduces the overall time required to learn, but limits the individual exposure to training data (requiring the team members to collaborate to achieve higher levels of classification accuracy).

```
Testing Results
========================
Correctly Classified Instances         429               84.7826%
Incorrectly Classified Instances        77               15.2174%
Kappa statistic                          0.8154
Mean absolute error                      0.0548
Root mean squared error                  0.1655
Relative absolute error                 26.5384%
Root relative squared error             51.5392%
Total Number of Instances              506

Detailed Accuracy By Class
========================
TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.787     0.047      0.842     0.787     0.814      0.961    one
  0.85      0.067      0.797     0.85      0.823      0.96     two
  0.974     0.006      0.925     0.974     0.949      0.998    three
  0.647     0.008      0.846     0.647     0.733      0.98     four
  0.903     0.027      0.884     0.903     0.894      0.988    five
  0.8       0.008      0.8       0.8       0.8        0.995    six
  0.948     0.02       0.859     0.948     0.902      0.996    seven
  0.81      0.004      0.895     0.81      0.85       0.998    eight

Confusion Matrix
========================
   a   b   c   d   e   f   g   h   <-- classified as
 102  18   0   0   0   0   0   0 | a = two
  26  96   0   0   0   0   0   0 | b = one
   0   0  55   2   0   0   1   0 | c = seven
   0   0   3  84   2   1   0   3 | d = five
   0   0   0   1  37   0   0   0 | e = three
   0   0   2   1   0  17   0   1 | f = eight
   0   0   1   1   1   1  16   0 | g = six
   0   0   3   6   0   0   3  22 | h = four
```

Figure 3.5: Example of a pre-collaboration event statistics file, containing training or testing results, detailed accuracy (multiple measures) by class, and confusion matrix. The classes are non-numeric (nominal). This example represents an individual learner's performance on a portion of the testing data.

Collaboration Event	Learner_1	Learner_2
1	0.655673	0.601583
2	0.658311	0.598945
3	0.701847	0.518470

Figure 3.6: Example of an incremental accuracies file, containing training or testing accuracies for each learner incrementally throughout the learning experiment. This example shows that, for this experiment involving a team of size two and three collaboration events, Learner #1's testing accuracy progressively increased throughout the experiment, whereas Learner #2's testing accuracy decreased through the experiment.

Combination Method	Accuracy for Each Method
Max:	0.7018469656992085
Selective Max:	0.7018469656992085
Average:	0.7058047493403694
Selective Average:	0.7018469656992085
Multiply:	0.7071240105540897
Selective Multiply:	0.7018469656992085
Majority Vote:	0.7018469656992085
Selective Majority Vote:	0.7018469656992085
Weighted Vote:	0.7018469656992085
Selective Weighted Vote:	0.7018469656992085
Select Best:	0.7018469656992085
Oracle:	0.8060686015831134

Figure 3.7: Example of a combination method accuracies file, containing accuracies for each of the combination method for the corresponding learning team. The Multiply combination method worked best for this data set.

3.3 Evaluation

In order to study and address the stated primary and supporting research questions in this dissertation, a series of specific studies have been performed on three different applications. The following team learning and collaboration aspects are the focus of this research:

- Single versus multiple learners (team size)
- Collaboration/sharing versus self-learning
- Inter-knowledge representation
- Knowledge (opinion) conflict resolution
- Collaboration's affect on opinion change
- Collaboration's affect on learning rate/curve
- Collaboration/sharing frequency
- Multi-learner decision combination method
- Heterogeneous versus homogeneous team composition
- Learning team composition and task performance
- Full (overlapping) versus independent (non-overlapping) training distribution
- Individual learner contribution to team success

The goal of evaluating these aspects through detailed team studies is to support the fact that teams are better than individuals; combining decisions from multiple learners is useful; and, most importantly, that collaboration between heterogeneous learning algoithms leads to higher task performance/accuracy compared to teams which do not collaborate during the learning process.

The three distinct experiments allow for direct comparisons for each data set, as well as for generating novel heuristics and general conclusions about these aspects across applications. Plots are generated based on testing accuracies for each team and individual study. Conclusions are then made and general aspects of team learning and collaboation are discussed. The scalability and generalizability of the proposed approach are illustrated throughout the dissertation, based on the experimental results and the data analysis.

Chapter 4

Multi-Agent Predator-Prey Pursuit and Capture

4.1 Introduction

There are typically many ways in which a problem can be solved. Some of these options use different algorithms, approaches, metrics, levels of adaptation, combinations of methods, and other fundamental differences in route to a solution. These decisions typically stem from how an individual or team has learned a task, or the combination of knowledge, experience, and expertise. In terms of a team, success therefore can depend on the team composition, variance in knowledge, communication and transfer of that knowledge, and different levels of experience for the task.

The more complex problems become, more resources (team members, time) are typically required to successfully tackle them. This also speaks to the composition of the team in terms of expertise, specialization, and diversity of its members. Team-building is a major aspect of human workgroups in research and commercial organizations. Assembling a very diverse team is desired so that a wide variety of knowledge can not only be utilized, but shared between individuals in the team in order to increase the team's overall efficiency and individual skill. Knowledge transfer is therefore a major aspect of learning, efficiency, and success.

The education field readily notes that individuals learn how to do things differently. Using different learning styles, some learn more effectively from audio, where others better learn from visual aspects. Some learn better from performing or experiencing an activity. These aspects seen in human workgroups and teams can be directly realized in teams of intelligent agents. Agents can learn individually using their own style (or learning algorithm) and periodically share what they have learned with their teammates. The team can thus learn a task much faster and increase its overall performance and success. This simultaneous learning of individual viewpoints being shared between several agents

can help the team learn complex tasks and cooperative behaviors while working toward a common goal.

Cooperative, mutual learning can be defined as the process by which several entities work together to learn a task, process, or aspects of an environment. This involves the individual entities of the group learning independently, then sharing what they have learned (i.e., their knowledge, or experiences) with a portion of or the entire team. In this manner, the team can then learn as a whole more quickly and build a more detailed knowledge base. An example of this is a group of students learning the same material for a course, where each student potentially learns differently and at a different rate. When the students team up to work on a task, they can effectively share what they learned to achieve a shared understanding and possibly a very efficient solution as a problem-solving unit.

In this work, we focus on studying the effects of periodically sharing knowledge in teams of intelligent agents utilizing a heterogeneous mixture of machine learning techniques. We use the term "intelligent agent" to represent both robots and software agents. The idea is that each agent will likely learn different information by using different learning algorithms, at different rates, as it progresses with the task. Our agents utilize a sharing schedule, where all agents can learn on their own but must participate in periodic sharing events. If there is conflicting knowledge between agents, these conflicts are immediately resolved before further learning takes place.

Performance is analyzed based on team configuration and sharing frequency. We recognize a team as *heterogeneous* if it consists of members that use different machine learning techniques. For example, a team consisting of one reinforcement learner and three neural network learners is considered heterogeneous. On the other hand, we recognize a team as *homogeneous* if the machine learning techniques are the same for all members, for example, a team composed entirely of reinforcement learners.

We also distinguish between the learning "strategies" which are used. A *centralized* strategy is where the agent is guided or directed by a centralized entity. A *distributed* strategy, on the other hand, is where the agent performs without any outside assistance or direction from a centralized entity. A team of intelligent agents that utilizes a combination of these approaches is considered to be using a *hybrid* strategy.

The following sections describe the implementation details, experimental setup, collaborative learning simulation results, and conclusions from this study.

4.2 Implementation

This section discusses implementation details. The predator-prey pursuit problem is introduced, as it is the application of the heterogeneous collaborative learning simulations. Implementation mainly focuses on that of the domain, inter-knowledge representation, and knowledge conflict resolution.

4.2.1 Predator-Prey Pursuit Problem

The predator-prey domain, occasionally referred to as the fox & hound problem, represents a popular Distributed Artificial Intelligence (DAI) testbed for multi-agent simulations involving coordination schemes and other multi-agent tasks. It involves a pursuit game between one or more predator agents and one or more prey agents. The objective is to capture the prey by touching it with a predator agent or simultaneously surrounding it with several agents.

This game can incorporate various strategies used by one or more agents to capture the prey. These approaches can be greedy or involve more intelligent behaviors stemming from learning and/or team cooperation. Greedy predator approaches tend to lead to deadlock, where agents get "stuck" behind other agents attempting to minimize their distance to the prey. As an example, integrating periodic random movements helped our predators avoid deadlock in the task by choosing non-greedy actions with a low and random frequency. Other work [111] also supports this, where it was found that performing a fraction of the actions randomly helps fix some problems inherent in the learning process.

The prey can also utilize several different approaches: run in a straight line, randomly choose actions, run until an obstacle is hit and choose a random open direction (the "linear prey"), or act more intelligently by maximizing its distance from all predators. The linear prey has proven to be able to consistently evade capture from certain predator algorithms [59]. This style of prey is effective because it avoids locality of movement, where greedy strategies rely on the prey staying in a small, local neighborhood in the environment. Our work strictly utilizes the linear prey, as it creates a very difficult capture task for teams of agents. Other predatory-prey applications for multi-robot learning are discussed in [58] and [76].

4.2.2 Simulation Environment

The simulation environment used for this work was TeamBots [17]. The simulated environment is a 30x30 grid with each cell available for occupation by either the prey or a predator. The environment is toroidal (wraparound) in that if an entity approaches a border and moves across that border, it emerges on the field opposite where it exited. For example, if the prey moves off the field on the eastern border, it will move to the western edge, while retaining its latitude.

The environment is discretized as a simplification of the continuous real world in the following ways:

- Five actions $\{N, S, E, W, \emptyset\}$
- Timestep-based
- Full visibility
- Ability to accurately estimate distance

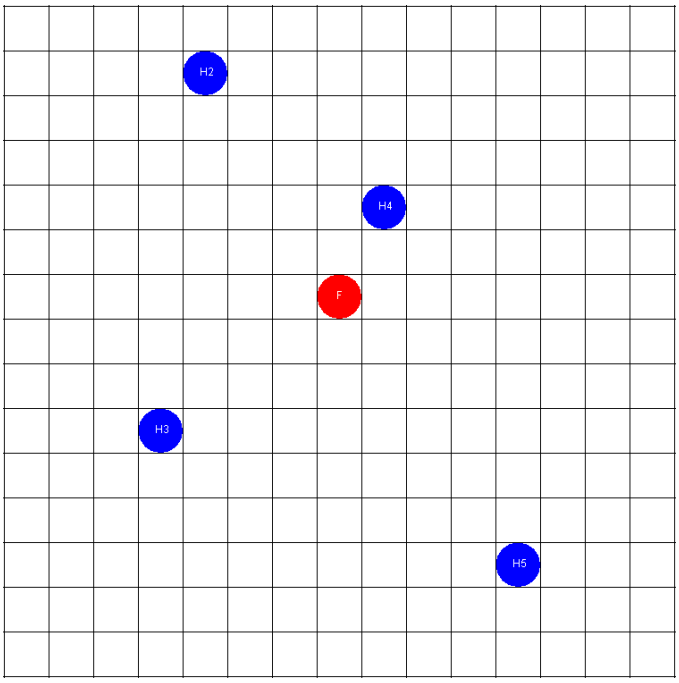

Figure 4.1: Example of a grid world generated in TeamBots, where four learning predator agents (darker shade) will attempt to minimize their distance to and collectively capture the prey (lighter shade).

Using this environment, we can calculate the number of possible states, given four predators and a single prey. As described shortly, there are five possible headings for the agents. There are C(30x30 grid, 5 agents) = C(900, 5) $\approx 4.866 \times 10^{12}$ possible configurations. With five possible headings for the agents, this results in approximately $5 \times (4.866 \times 10^{12}) \approx 2.433 \times 10^{13}$ possible states. Figure 4.1 shows an example grid-based simulation environment in TeamBots.

Movement and Turns

The simulator progresses through a series of steps within which the prey and each predator are allowed a turn. The movement of each entity is governed by its learning algorithm and its current knowledge. The prey picks a horizontal or vertical direction (N, S, E, W) and begins moving in that direction. It continues doing so until impeded by a predator directly in its way, when it will randomly pick an open direction to move. Once every 10 timesteps, the prey opts out of moving in order to represent the prey being 90% as fast as the predators. Each predator has an opportunity to move at each step of the simulation. The predators may move one space in any horizontal or vertical direction (N, S, E, W), or may choose to not move (Ø). If a predator chooses to move into another entity (i.e., another predator or the prey), it will forfeit its move for that step. Figure 4.2 shows the defined predator movement possibilities.

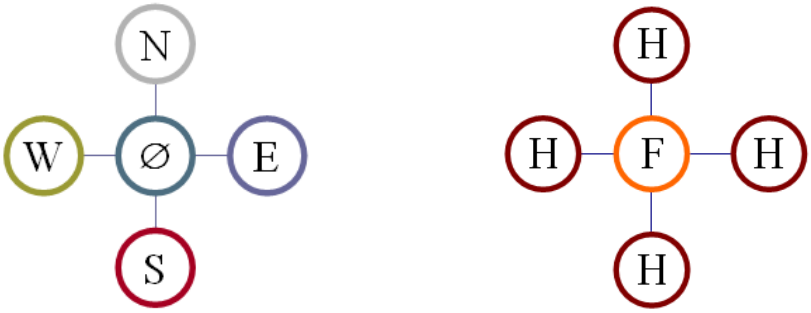

Figure 4.2: (Left) The five movement choices for predators: North, South, East, West, and Stay. (Right) Finished state of the game (cooperative capture of the prey).

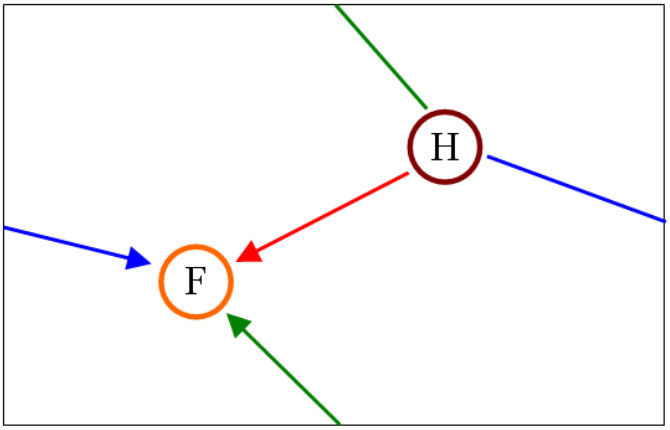

Figure 4.3: Multiple direct paths from predator to prey in a wraparound world to select a direction of movement.

Winning the Game

Team size is kept static at four learning predators, each employing Q-Learning or a Neural Network to learn. This is described as the orthogonal pursuit game given our movement restrictions [76]. The simulation ends when the four predators surround the prey so it can no longer move in any horizontal or vertical direction. Figure 4.2 shows the capture configuration for winning the orthogonal game. For this and subsequent figures, F represents the prey (Fox) and H represents a predator agent (Hounds).

Toroidal Consequences

Due to the nature of a toroidal environment, there are at any instance three possible direct paths from a predator to the prey: one crossing the open field, one crossing the North-South boundaries, and one crossing the East-West boundaries. These distance measurements are used to determine which of these three paths is shortest, suggesting a movement action that approximates this path. Figure 4.3 demonstrates these three paths.

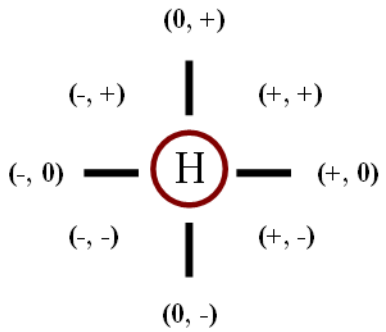

Figure 4.4: Orientation vector values to a prey depending on its region compared to the predator.

4.2.3 Situation Feature Vector Definition

Perhaps the most valuable step in developing a reinforcement learning algorithm, as well as input for other learning algorithms, is situation definition. Through situation definition, knowledge of the problem domain can be imparted to the learning system by its designers. Based on thorough consideration of the domain and strategies required, we have determined the following four components to be critical situational information to succeed most efficiently:

- *Prey Heading (F_H)*: Current direction of the prey's movement, used to factor in where the prey is going rather than where it is.
- *Orientation Vectors (X_{OV}, Y_{OV})*: Relation between the prey's position and the predator's position relative to its X and Y axes, respectively. Each of the vectors has one of the values from the set {-,0,+}, where '-' represents the prey being to the left of the predator, '0' represents the prey being in-line with the predator in that direction, and '+' represents the prey being to the right of the predator. Figure 4.4 shows the orientation vector representation.
- *Prey Surrounded (F_N, F_S, F_E, F_W)*: Status of each strategic point surrounding the prey. Is there another predator already there? If so, the deciding predator may learn to use this information to find a better position to approach even if an occupied position would minimize its distance to the prey. Figure 4.5 demonstrates the prey surrounded flag positions.
- *Predator Surrounded (M_N, M_S, M_E, M_W)*: Status of each adjacent position surrounding the deciding predator. Is there a prey or predator surrounding me? Figure 4.5 demonstrates the predator surrounded flag positions.

Given the situation definition, the following represents the situation feature vector (S):

$$\{F_H, X_{OV}, Y_{OV}, F_N, F_S, F_E, F_W, M_N, M_S, M_E, M_W\}$$

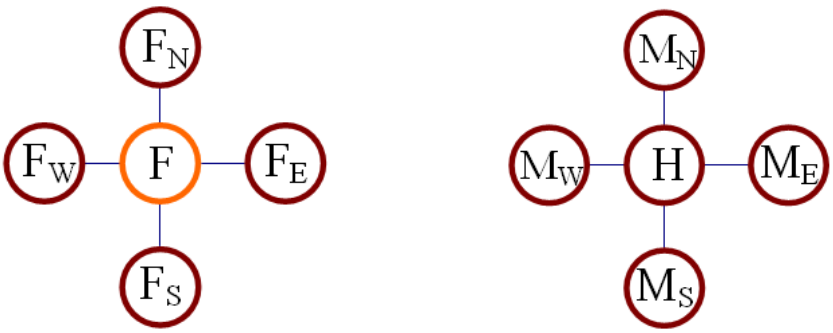

Figure 4.5: (Left) Prey surrounded flag positions. (Right) Predator surrounded flag positions.

For example, the situation feature vector $S = N \mid +- \mid 0010 \mid 0001$ represents the prey moving North, in the '+−' quadrant with respect to this predator, a predator already occupying the cell East of the prey, and an entity occupying the cell West of this predator. Other agents' orientation vectors were not used, as we wanted to keep the knowledge as independent as possible for the sharing events.

This decomposition results in 9216 possible situations, although some are eliminated due to rules of the problem. We imagine that the final eight conditions of the situation feature vector will be false, allowing the predators to easily adjust their long-range approach to the prey based on the associated heading and relative orientation. Once the predator is close to the prey, the eight status flags will allow it to discern effective action in different group situations.

4.2.4 Reinforcement Learners

Reinforcement learning is a machine learning algorithm aimed at forming a policy for an agent based on attempting to maximize or minimize a reward. This style of learning is also used for children and pets, for example, to teach them right from wrong. This algorithm is central to the notion of positive and negative reinforcement, where positive reinforcement is given when actions are performed correctly (or it has provided improvement) and negative reinforcement is provided in cases where an action was incorrect (or has not provided improvement).

The flavor of reinforcement learning utilized here is Q-Learning [144]. At each state s, an action is chosen which maximizes the utility function $Q(s, a)$. Q tells us how good an action is in a given state. The general algorithm involves choosing an action and executing it, receiving immediate reward, observing the resulting state after taking that action, and updating knowledge accordingly using an estimate of future reward. This process is repeated for the lifetime of an agent for life-long learning.

Other details of the implementation involve using a decaying reward function based on a reinforcement window of size 4. This provides weighted reinforcement for recent actions

given the amount of improvement in approaching or capturing the prey. Actions are chosen in a weighted random fashion, based on the action weight, to allow random movements at a low frequency for avoiding deadlock.

Knowledge (situation-action weights) are stored in a Q-table, which can be thought of as a hash table. As briefly discussed earlier, Q-Learning knowledge weights can become quite large over time, both positive and negative, as reinforcement is repeated for certain situations. The larger the action weight, the better that action is for that certain situation.

4.2.5 Neural Network Learners

Neural Networks are computational models consisting of a network of interconnected neurons similar to the vast network of neurons, dendrites, and axons in the human brain. They are typically used to model, find, or learn patterns in complex relationships between inputs and outputs. They operate on a connectionist approach, where neurons are fired, and their output is adjusted by weighted connections in route through the network via other neuron interfaces and layers.

Neurons represent simple processing entities which, when connected in concert with other neurons, can exhibit complex global behavior. Each neuron utilizes a transfer function, which it uses to adjust input for forwarding through connections to other neurons in subsequent layers. A neural network is typically broken down into layers of neurons which behave similarly in terms of transfer function or connections. There is typically an input layer representing the nodes for inputs into the neural network, some intermediate "hidden" layer(s), and an output layer which produces the outputs from passing the input through the neural network.

Neural networks are trained by repeatedly providing the network with inputs and iterating until the error goal for network output has been achieved, or the maximum iterations have taken place. The network then learns the association between inputs and outputs by passing each input through the network and calculating the error the network produces. This error is then typically used to adjust connection weights in reverse (backpropagation), from output nodes to input nodes. By adjusting connection weights, the network can adjust its output given input. The connection weights of the network therefore represent the neural network's knowledge. Compared to Q-Learning, network connection weights are at a much smaller scale, also positive and negative.

The fully-connected backpropagation neural network architecture used for the Neural Network learners consists of 11 inputs (situation feature vector), 22 hidden nodes in a single layer, and five outputs (action weights for the five possible actions in each situation). The 11 inputs correspond to the situation description defined in Section 4.2.3 (the description of the grid at a specific timestep), converted into a numerical representation. A single hidden layer composed of 22 nodes was empirically found to work well for this task, and therefore used here. The five outputs correspond to the five possible movement actions and their

weights. Thus, the network's goal is to learn what actions to take in certain situations. After each move has been made, the network is trained on what action was best to choose in that situation (i.e., the action resulting in getting closer to the prey or moving into a capture position). The network is trained for a maximum of 100 epochs or a training error of 0.001, and after some brief experimentation, the learning rate was set to 0.1.

4.2.6 Inter-Knowledge Representation

As we use heterogeneous learners with different knowledge representations, a common middle language is necessary to provide a way for agents to share their knowledge with one another, and so they can properly learn from the communicated representation. As we assume each learner utilizes the same environment discretization and possible actions, our inter-knowledge representation directly involves the common factors of the situation definition (feature vector) and action weights.

For each knowledge entry from each learner, the following format is used for converting to the sharing representation. This assumes that action weights are normalized for each situation, forcing them to the same [0,1] scale:

$$SituationFeatureVector = ActWt_N, ActWt_S, ActWt_E, ActWt_W, ActWt_\emptyset$$

As seen in the below example, the North action for this situation is clearly favored over the others, which are nearly equal in weight.

$$E \mid +o \mid 0100 \mid 1010 = \underline{0.543445}, 0.114105, 0.114235, 0.114106, 0.114109$$

This is the format used by all agents as the inter-knowledge representation. Therefore, each must convert its knowledge from its own representation into this so that all knowledge is in the same format and scale. Reinforcement learners accomplish this by simply walking their hash table, normalizing the action weights for each seen situation, and formatting the knowledge entries accordingly. On the other hand, neural network learners must keep track of the situations they have seen. When it is time to share, each of these situations is passed through the network, the output is normalized, and the knowledge is properly formatted. Sharing and integration can then take place.

However, normalization destroys the scale of the knowledge being converted. This is detrimental in that the scale represents experience that has built up over time for a learner. To counter this, the process needs to be reversible so that after sharing knowledge can be restored to its original scale (to retain experience). This is accomplished by saving two pieces of information during situation normalization: (1) the smallest action weight, and (2) the sum of weights after all have been made positive. The normalization process can take place in reverse to restore the knowledge back to its original scale.

Table 4.1 provides an example of this procedure for a situation being shared by a learning agent. The process starts with actual weights for each action. These weights are

Table 4.1: Example normalization and restoration process for a situation's action weights during a sharing event.

Action	Original	Normalized	Post-Sharing	Restored
North	5000	0.240481	0.25	5257.25
South	7500	0.332988	0.35	7959.75
East	2345	0.142239	0.15	2554.75
West	6183	0.284255	0.3	6608.5
Stay	-1498	0.000037	0.00005	-1497.65

then normalized (sum to 1, [0,1] scale) and shared with all other agents. After sharing, the weights come back slightly adjusted (based on conflict resolution). Using the two saved pieces of information from normalization for this situation, the weights can be restored back into their original scale. Note here that the restored weights reflect the small changes in weights from the sharing event, but with the experience retained. Note that agents only communicate during sharing events.

4.2.7 Knowledge Conflict Resolution

As we briefly saw earlier, there will be portions of the problem (the same situations) which have been experienced by more than one agent. These are areas where different levels of agreement and conflict can occur. If differences in opinion (preferred action in a situation) exist, the knowledge is said to conflict and needs to be resolved.

Conflict resolution takes place here using the method of weighted averaging [126]. This averages individual action weights for each unique situation. Thus, if the same situation has been seen by multiple agents, the resulting integrated knowledge for that situation will represent the average for each action over all identical situations. However, some agents may have higher weights (i.e., higher preference), which raise the average.

We define knowledge to be *new* during a sharing event if only one agent has experienced it. This means that that agent has seen that situation since the previous sharing event, and thus needs to be integrated into all of the other agents' knowledge. In this case, the averaging method outputs the situation untouched. Knowledge is considered to be in *agreement* if multiple agents have experienced the situation and have the same opinion (highest weighted action) for it. On the other hand, knowledge is considered to be *conflicting* if multiple agents have experienced the situation and have differing opinions. For both of these cases, each action is averaged over identical situations.

The following example of averaging with conflicting knowledge further demonstrates this process. Notice that the first agent prefers a different action (East) than the other three (North). The result of averaging shows that the overall "vote" of the group is retained, and the single preference of East bumped the resulting weight for that action to second place.

Normalized action weights for a common situation ($E \mid +o \mid 0100 \mid 1010$):

$$A_1 = 0.17528, 0.17528, \underline{0.29875}, 0.17528, 0.17540$$
$$A_2 = \underline{0.33313}, 0.16672, 0.16671, 0.16671, 0.16672$$
$$A_3 = \underline{0.54344}, 0.11411, 0.11424, 0.11411, 0.11411$$
$$A_4 = \underline{0.54344}, 0.11411, 0.11424, 0.11411, 0.11411$$

Resolved knowledge using the averaging method:

$$All = \underline{0.39883}, 0.14255, 0.17348, 0.14255, 0.14258$$

This process is performed for all shared knowledge. This new knowledge is then distributed back to the agents. Any differences between this knowledge and their own, including newly shared experiences, will be integrated into their knowledge base and used by the learners to adjust their knowledge. For example, Neural Network learners retrain themselves using the result of conflict resolution for situations where knowledge conflicts occurred. Knowledge agreements and conflicts are indicative of the behavior of both the individual agents and the team. If agents have agreeing opinions, this further adds to that knowledge being more reliable and trustworthy. Conflicting knowledge can contribute to the specialization of the agents, or mean that they actually experienced (rewarded) the same situation differently.

The following sections present the experimental setup and results from software simulations involving heterogeneous collaborative learning for cooperative pursuit and capture.

4.3 Experimental Setup

Using TeamBots for software simulations, 33 total learning team configurations were run. All teams have four agents, and are composed of combinations of two learning algorithms (Q-Learning and Neural Network, one per agent), two learning strategies (centralized and distributed), and three sharing frequencies (sharing two times per trial, four times per trial, and not sharing). This allows us to create many different teams to study individual aspects such as the effect of heterogeneity, sharing frequency, and learning strategy composition.

The said **centralized** strategy incorporates a pre-defined position for an agent to occupy around the prey (e.g., cell North of the prey). The **distributed** strategy allows an agent to freely, without external or predefined guidance, determine the position to occupy during a capture. Combinations of these in a team were investigated to see whether it positively affects performance or success.

Each of the 33 team configurations is simulated 10 times, and results averaged. This results in 330 total simulations, each of which consists of 100 learning trials and 100,000 timesteps per trial. Thus, sharing two times per trial takes place every 50,000 timesteps. Similarly, sharing four times per trial takes place every 25,000 timesteps.

Table 4.2: Team configurations, abbreviations, and labels used throughout.

Label	Team Configuration	Label	Team Configuration
A	2CNN+2CRL	R	4DRL (50000)
B	2CNN+2CRL (25000)	S	1CNN+1CRL+1DNN+1DRL
C	2CNN+2CRL (50000)	T	1CNN+1CRL+1DNN+1DRL (25000)
D	4CNN	U	1CNN+1CRL+1DNN+1DRL (50000)
E	4CNN (25000)	V	2CNN+2DNN
F	4CNN (50000)	W	2CNN+2DNN (25000)
G	4CRL	X	2CNN+2DNN (50000)
H	4CRL (25000)	Y	2CNN+2DRL
I	4CRL (50000)	Z	2CNN+2DRL (25000)
J	2DNN+2DRL	AA	2CNN+2DRL (50000)
K	2DNN+2DRL (25000)	BB	2CRL+2DNN
L	2DNN+2DRL (50000)	CC	2CRL+2DNN (25000)
M	4DNN	DD	2CRL+2DNN (50000)
N	4DNN (25000)	EE	2CRL+2DRL
O	4DNN (50000)	FF	2CRL+2DRL (25000)
P	4DRL	GG	2CRL+2DRL (50000)
Q	4DRL (25000)		

Each of the 33 team configurations is listed in Table 4.2. In this table and subsequent figures, the configurations are abbreviated. The following represents an example of a learning team configuration:

Example: 2CRL+2DNN (25000)

This example translates to, out of a team of four learning agents, two agents using reinforcement learning (RL) with centralized strategies (C), and two agents using neural networks (NN) with distributed strategies (D). This team also shares their knowledge every 25,000 timesteps (four times per trial). Of special note here is that this configuration represents a heterogeneous learning team (the focus of this work), as well as a hybrid strategy team.

As this is targeted at life-long learning, all trials and timesteps are used for learning to increase task success and efficiency. Initially, and after each successful capture, all four predator agents and prey are randomly positioned in the environment. Over time, several aspects about the learning and sharing process are recorded for analysis. The main metrics used to compare teams in terms of learning rate and overall success for the cooperative pursuit task are as follows:

- *Number of cumulative captures*: The most important metric in that it is a measure of overall success for the cooperative pursuit task.
- *Number of experienced/learned situations*: Allows the study of learning rate and the learning curve (new experiences) over time.

- *Number of opinion conflicts*: Allows a look into how agents and team configurations learn differently.
- *Number of opinion changes*: Allows the study of how sharing and conflict resolution change agent opinions over time.

4.4 Experimental Results

This section presents the results of the learning simulations. Namely, overall team comparisons will be discussed, along with learning curves, and how knowledge conflicts change over time due to sharing events.

4.4.1 Overall Team Comparisons

Figure 4.6 shows a plot of the cumulative number of captures during the entire learning session for all team configurations. The embedded table shows the corresponding best and worst five team configurations for captures during learning. Note the legend, which differentiates between heterogeneous non-sharing teams, homogeneous non-sharing teams, and their sharing counterparts. To clarify, Team B is the same configuration as Team A but shares four times per trial. Similarly, Team C is the same configuration as Team A but shares twice per trial.

Success is the most important metric. One can see that all heterogeneous non-sharing teams are much more successful compared to the conventional homogeneous teams. The table shows this, as each of the top five teams are heterogeneous. We envisioned this result, where one learning algorithm (neural network) performed poorly on its own, the other learning algorithm (reinforcement learning) performed better on its own, but the combination of the two as part of a heterogeneous learning team performed best (most success) overall. Additionally, hybrid strategy non-sharing teams also performed consistently well, further demonstrating that integrating heterogeneous aspects into the team-based learning tasks is beneficial.

Interestingly, the sharing teams performed consistently poor in comparison. It appears that sharing more often performs worse in terms of capture success. This was unexpected, but could be related to the averaging conflict resolution method. As this method forces each agent to agree after a sharing event, it could be corrupting specialization. For this task especially, agents may be required to take different actions in the same situation to successfully capture the prey. Averaging during sharing events for a cooperative pursuit task may not be sufficient or sophisticated enough. Averaging may provide the best solution for other applications, however.

Figure 4.7 presents the average captures per testing session for all team configurations. As before, the best and worst five teams in terms of average captures per experiment are shown. Reinforcement learning teams clearly outperform neural network teams on

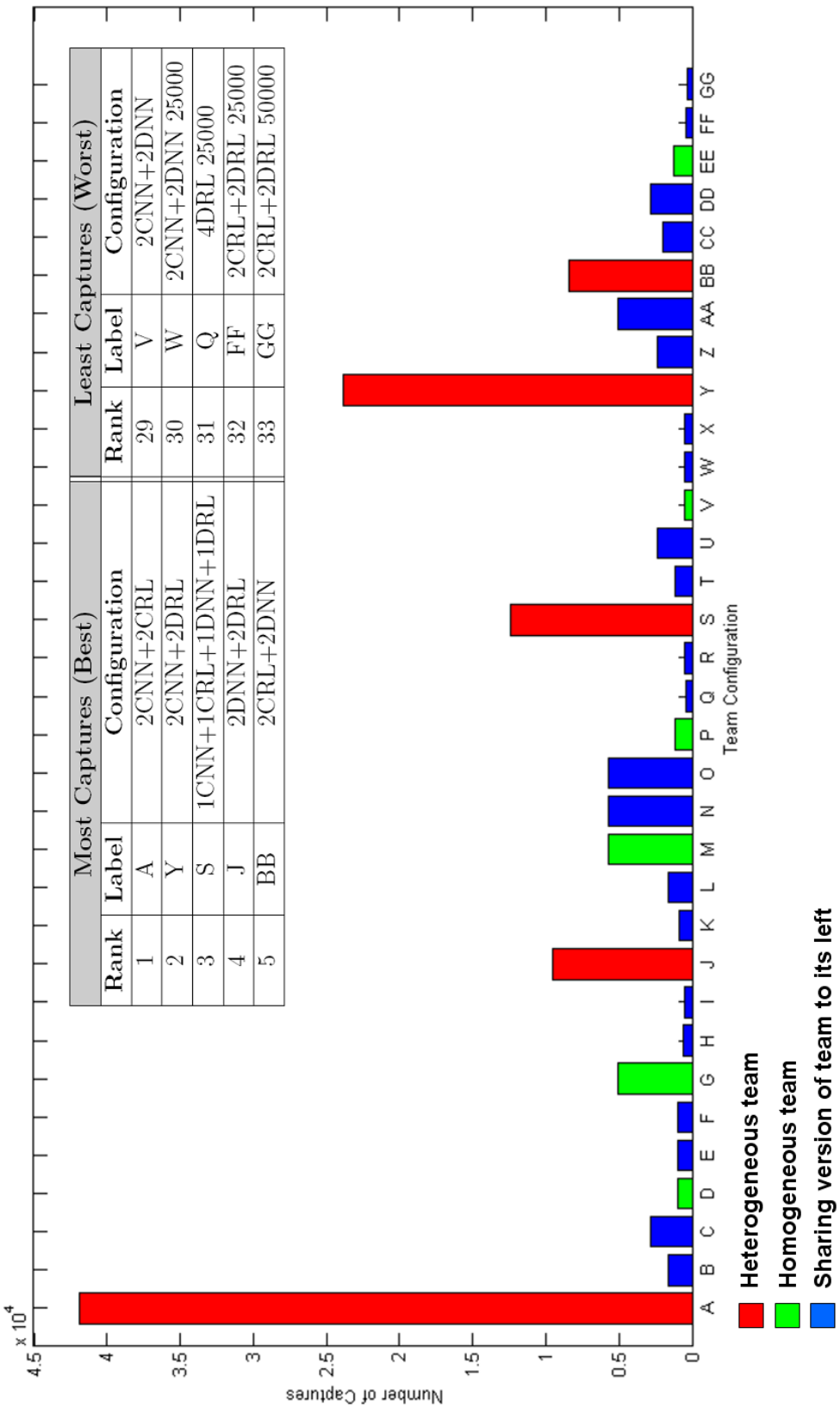

Figure 4.6: Number of cumulative captures per learning session for all team configurations, including the best and worst five teams (most and least cumulative captures, respectively).

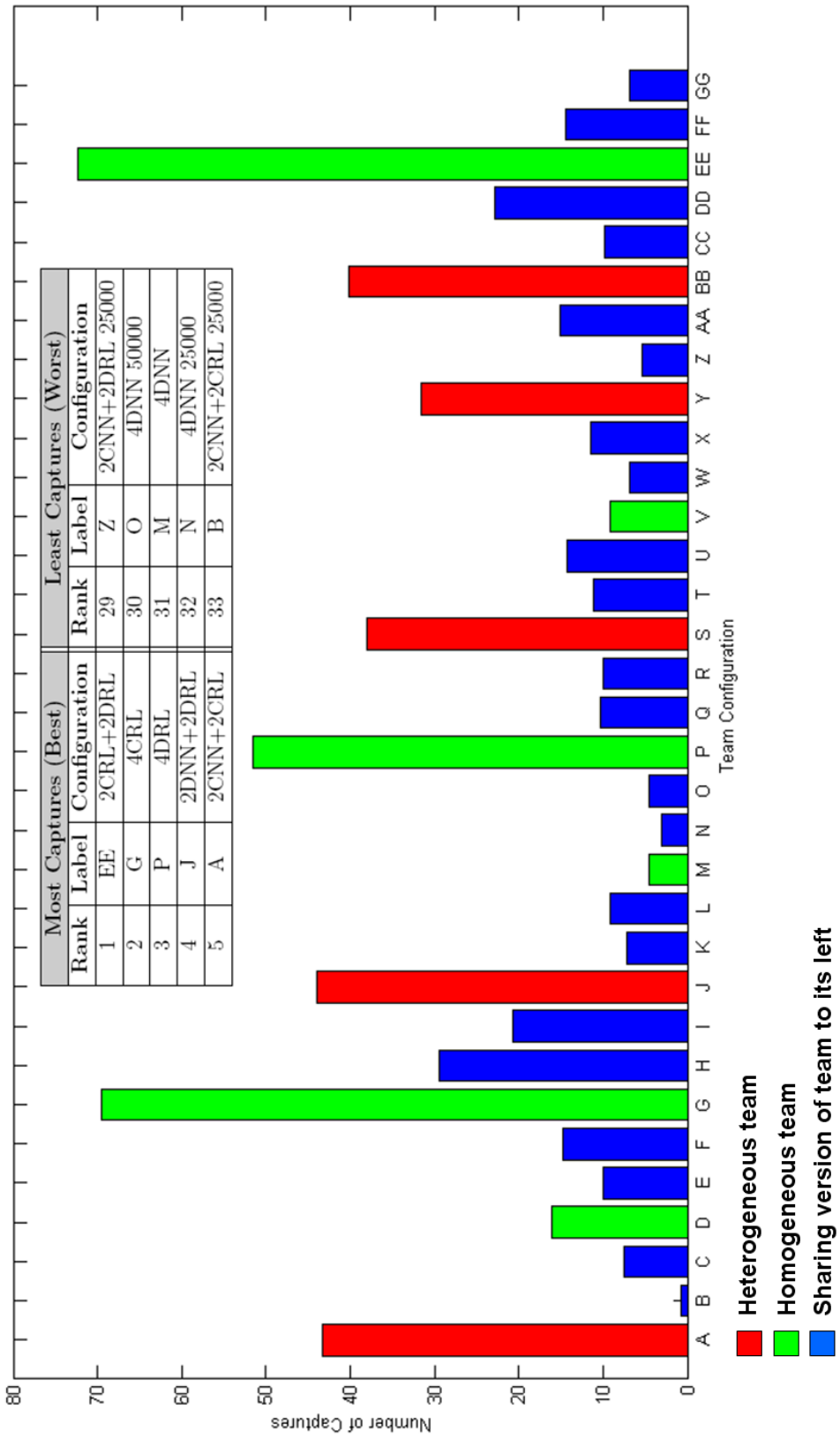

Figure 4.7: Average captures per testing session (out of 100) for all team configurations, including the best and worst five teams (most and least cumulative captures, respectively).

average. Hybrid reinforcement learning teams perform best overall, compared to purely centralized and purely distributed teams of the same composition. Another trend is that pure reinforcement learning teams tend to have their performance increased as a function of increasing sharing frequency, as opposed to heterogeneous teams which consistently degrade in performance from sharing more frequently.

All heterogeneous teams again perform very well, only being outperformed by pure reinforcement learning teams, supporting other results that heterogeneity offers a distinct advantage in multi-agent applications. Non-sharing teams performed consistently better in comparison. However, for the hybrid neural network teams, sharing (but sharing with low frequency) produced better results (more captures) than if sharing did not take place. As the neural network learners perform relatively poor individually, cooperation via sharing experiences allowed them to increase their cooperative capture success. This may mean that sharing produces its largest benefit on teams composed of weak individual components. Interestingly, here we see that sharing too frequently can cause a degrade in performance for certain team configurations. We also see that by combining good-performing learning algorithms (RL) with poor-performing algorithms (NN), increased team performance can be observed.

Figure 4.8 presents the average steps per capture during testing for all team configurations, the primary efficiency measure. All teams which experience a significant increase in performance due to sharing are fully distributed or hybrid homogeneous teams. Sharing with higher frequency increases efficiency for six out of 11 team configurations, all but one of which is homogeneous. Thus, sharing in homogeneous teams provides a more noticeable advantage in terms of capture efficiency. Surprisingly, the homogeneous team of four neural network agents that are directed what position to occupy for capture (centralized) performed very poorly in terms of capture efficiency. However, sharing greatly improved team capture efficiency. For this application, we can conclude that heterogeneous team composition, hybrid strategies, and sharing (but not too frequently) contribute to increased success.

4.4.2 Learning Curves

A visualization of learning progression over time is the learning curve, which can be represented in two different ways for our application. First, the learning curve can be measured in terms of the number of captures per trial and a view of how that changes (positively or negatively) over time. Second, the learning curve can be the actual number of unique situations the team learns/experiences over time, a non-decreasing curve. The following figures show these variations for learning and sharing configurations.

Figure 4.9 shows the learning curve for the top five sharing and non-sharing teams in terms of the number of captures during learning. Again, the top five teams are composed of heterogeneous learners. The top two performed considerably better than all other teams.

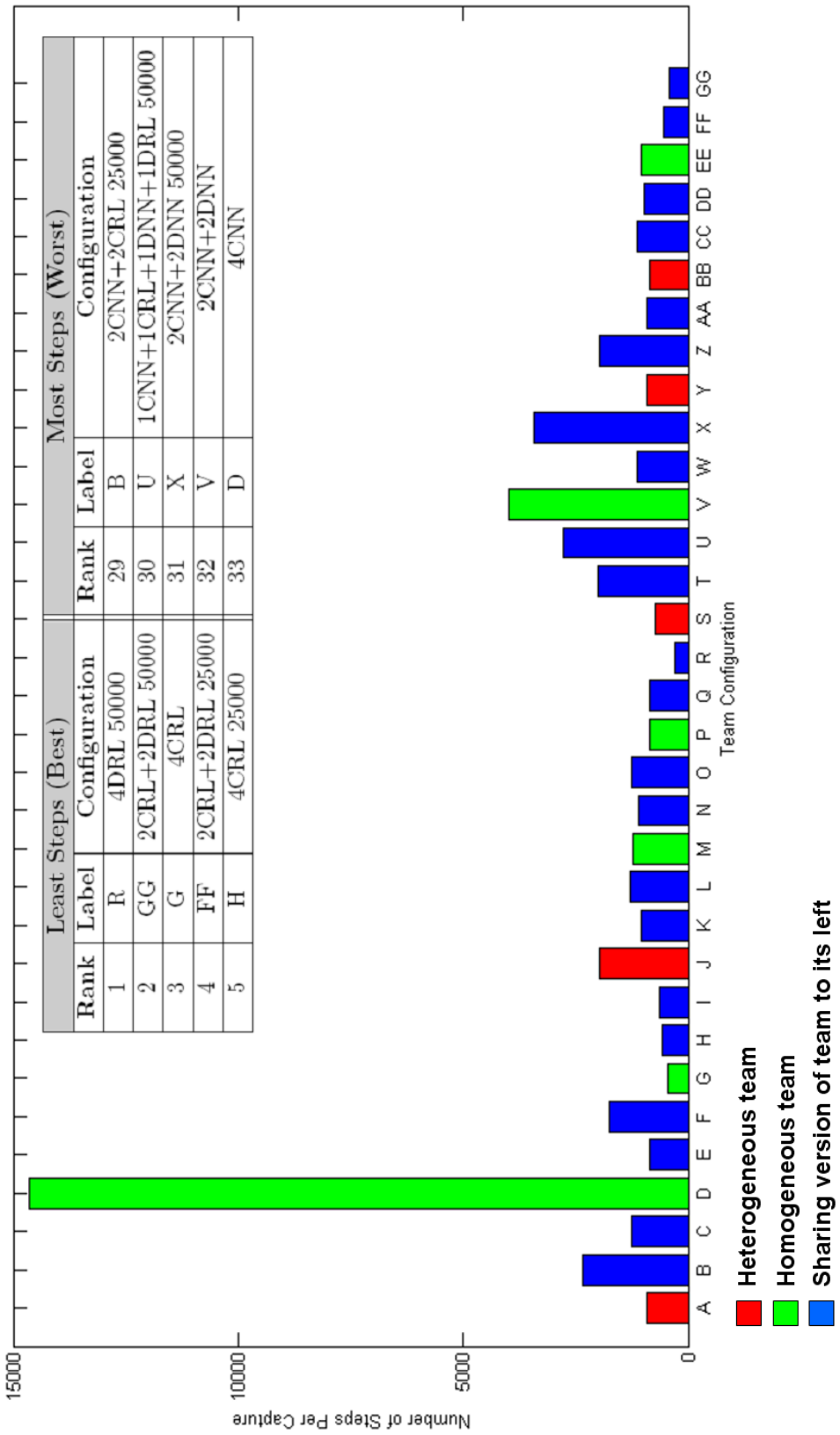

Figure 4.8: Average moves (timesteps) per capture per testing session for all team configurations, including the best and worst five teams (least and most steps per capture, respectively).

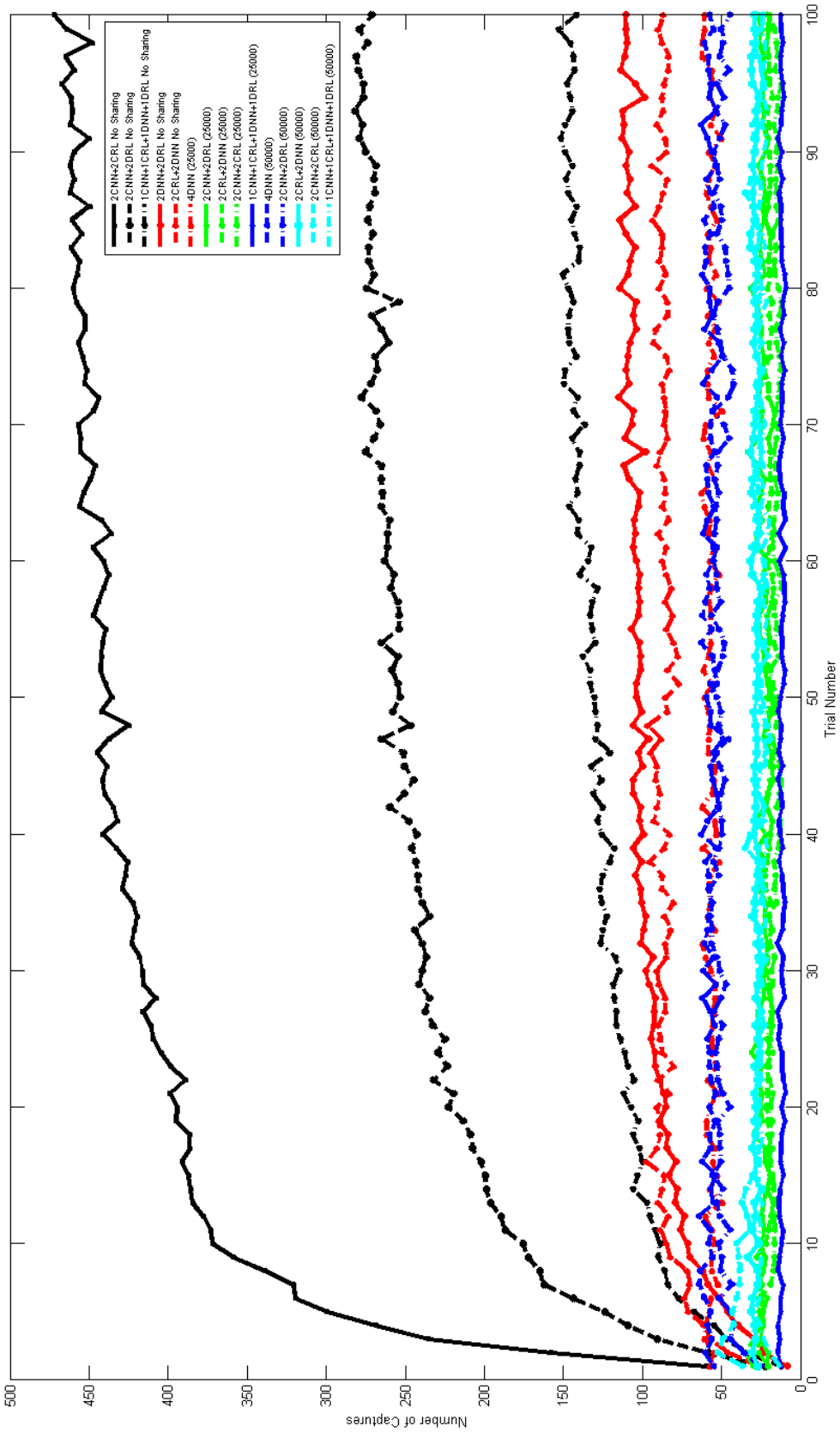

Figure 4.9: Learning curve comparison (number of captures per trial) for the top five sharing and non-sharing team configurations.

Three out of these top five teams employ heterogeneous strategies as well. Heterogeneous learning teams learned more much faster, in the form of capturing more often with less learning time. Learning new situations continually was observed, so that the number of captures kept increasing (an ascending curve).

One can see that the best sharing teams (all clustered at the bottom) exhibit a near-flat learning curve. This result again tells us that the averaging conflict resolution method may be hindering learning progression by corrupting specialization between sharing events. This plot also allows us to see the difference in magnitude between the best performing team configurations from each sharing setup. The magnitudes are clearly lower for sharing teams, which we again attribute to the averaging sharing method performing poorly for this application, and potentially the coarse discretization of the environment. This figure provides further support for using teams of heterogeneous learners.

Figures 4.10, 4.11, and 4.12 present the capture-based learning curves for non-sharing, sharing twice per trial, and sharing four times per trial, respectively. The top five for each sharing frequency for Figure 4.9 were selected from these results. Notice that there is only one homogeneous team that appears in these figures (4DNN), which captured more than any other sharing team configuration. We attribute the success here to the coupled sharing and pure distributed strategy of this team configuration, producing cooperative capture improvement. As noted before, three out of the top five employ a hybrid strategy. Generally, after 10 learning trials, sharing teams exhibit a leveling of capture success, with the 2CNN+2CRL configuration experiencing a descending learning curve when sharing.

Let's now switch to the learning curve representation in which the number of learned/experienced situations is the measure of learning progress. Learning induces new situations being seen as the learners explore different paths to the prey within the environment. Sharing allows situations to be learned without needing to directly experience them. This is important as it allows investigation into how sharing is increasing the team's overall experience/knowledge of the problem over time. As the top five teams for all sharing experiments were identical, Figure 4.13 shows these top five teams along with the different sharing configurations. This offers a comparison of how both sharing and sharing frequency affect learning new situations. Homogeneous RL teams performed best in terms of number of learned situations. Closely behind, the fourth through seventh best teams are heterogeneous learning teams. In the first 30 trials, the 2DNN+2DRL heterogeneous learning team was performing best, but was overtaken by other teams as time progressed. There is no clear advantage of using hybrid strategies from a vantage point of learned/experienced situations.

All solid lines in the figure are non-sharing teams, and these all perform comparably worse (least number of learned situations). Clearly, sharing allows teams to learn more situations faster. Interestingly, sharing less often allowed learners to learn slightly more situations than teams which shared at a higher frequency. This tells us that periodic

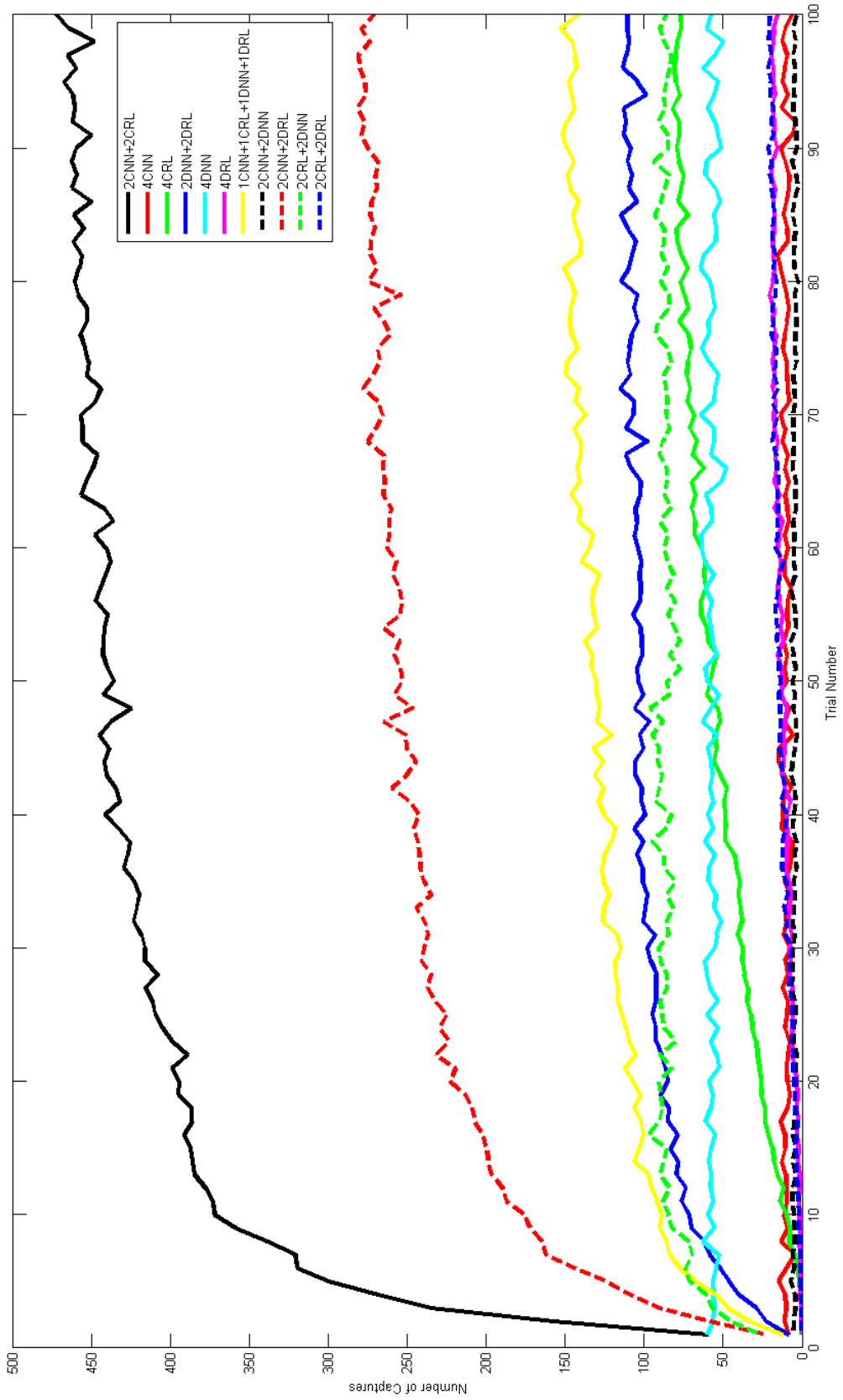

Figure 4.10: Learning curve comparison (number of captures per trial) for non-sharing team configurations.

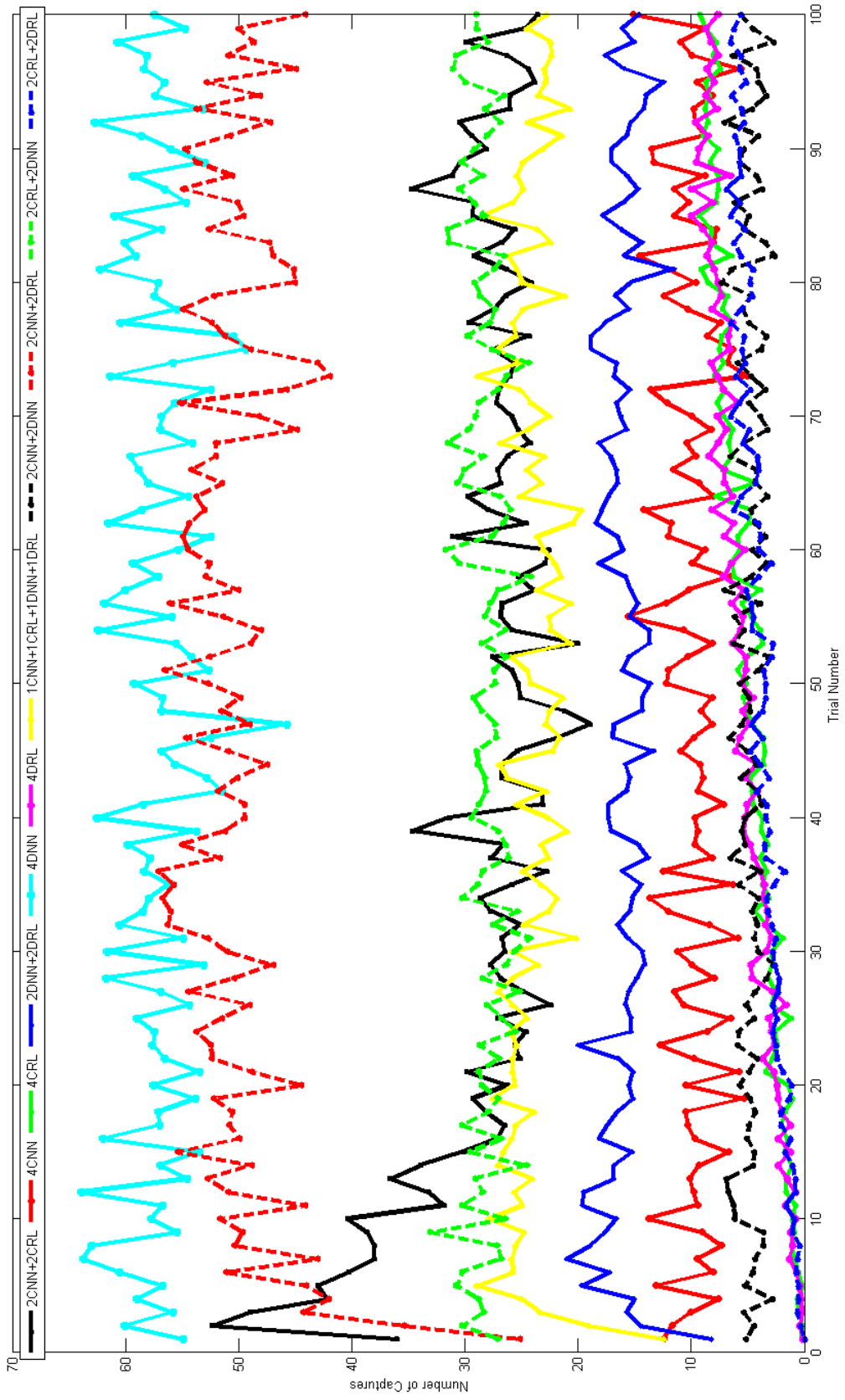

Figure 4.11: Learning curve comparison (number of captures per trial) for team configurations which share every 50000 steps (twice per trial).

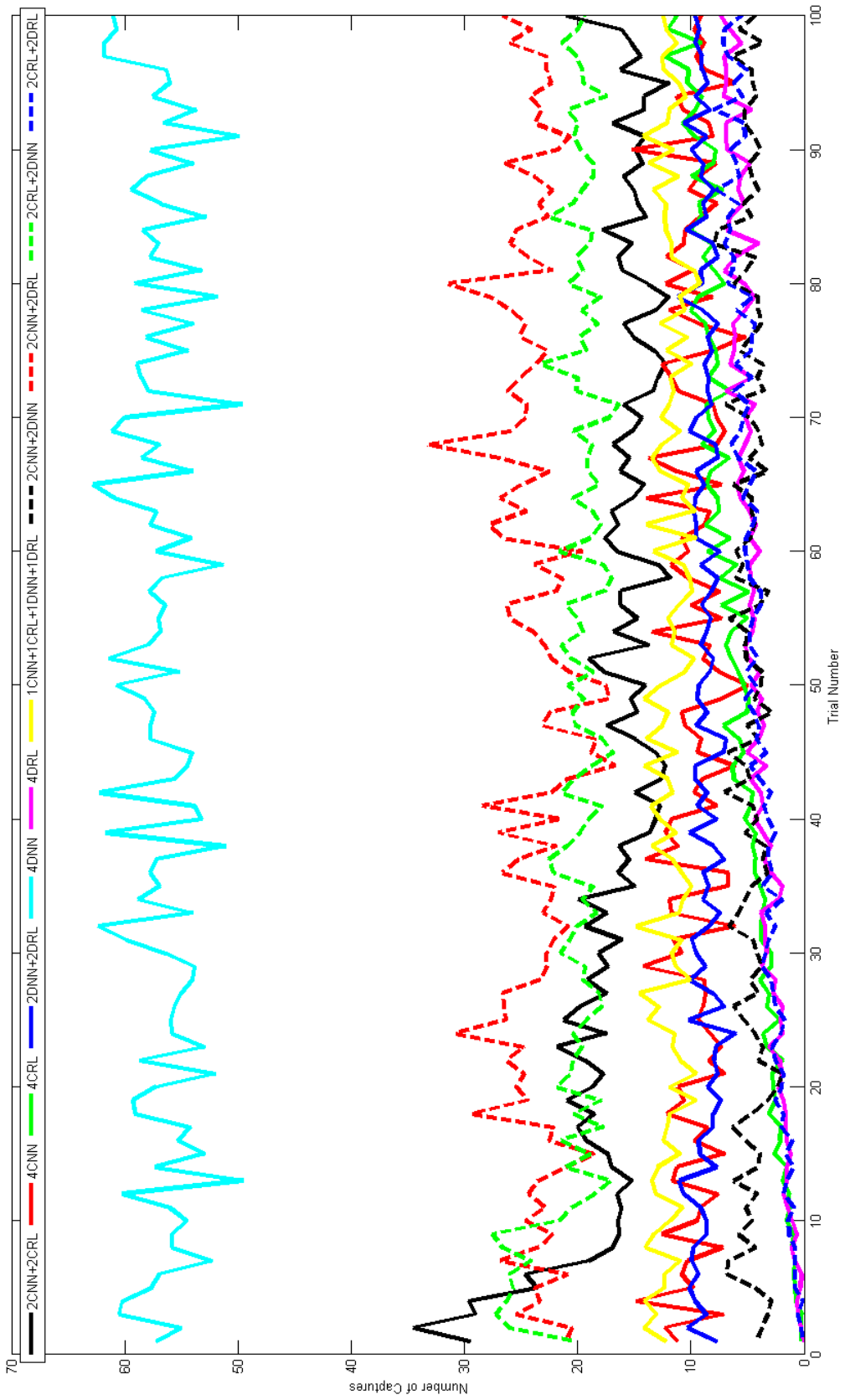

Figure 4.12: Learning curve comparison (number of captures per trial) for team configurations which share every 25000 steps (four times per trial).

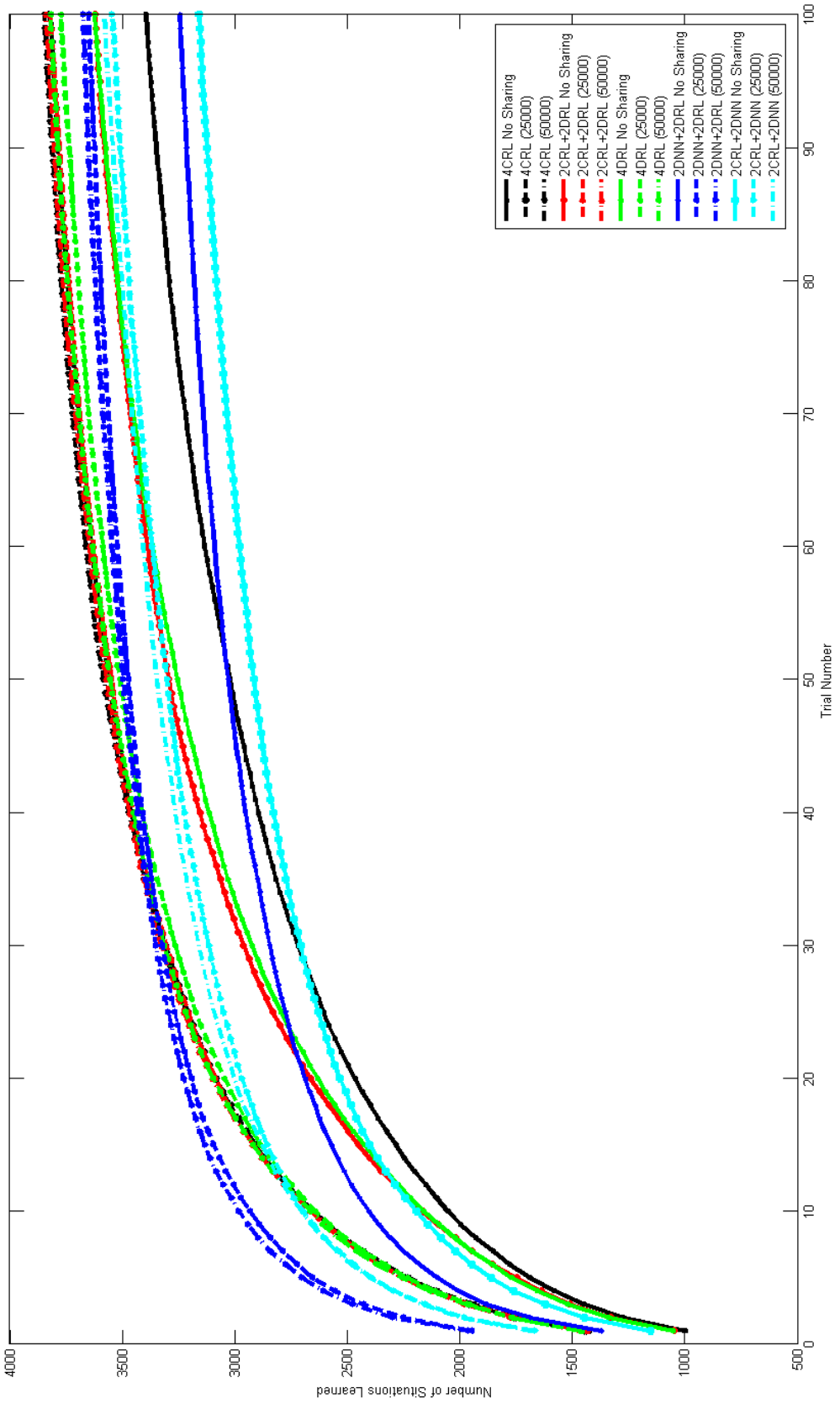

Figure 4.13: Learning curve comparison (number of learned situations) for the top five sharing and non-sharing team configurations.

sharing is useful for learning rate, and that some communication cost can be saved by sharing with lower frequency.

Figures 4.14, 4.15, and 4.16 present the situation-based learning curves for non-sharing, sharing twice per trial, and sharing four times per trial, respectively. The top five for each sharing frequency for Figure 4.13 were selected from these results, which happened to be identical for all sharing frequencies. These figures further support the results discussed for Figure 4.13.

4.4.3 Opinion Conflict and Change Curves

When agent opinions differ, we say there exists an *opinion conflict*. During a sharing event, all opinion conflicts are resolved by averaging the weight contributions from all agents for each action in the overlapping situations (those experienced by more than one team member). The result is a single situation-action weight distribution which all agents adopt. If this result differs from an agent's knowledge prior to the sharing event, we say that an *opinion change* occurred as a result. These two measures can help gauge how sharing is affecting the result of learning and overall team performance. Note that opinion changes were not measured for non-sharing team configurations.

Figure 4.17 shows the opinion conflict curve over all trials during learning for the top five sharing and non-sharing team configurations. Those curves with a smoothly increasing trend correspond to non-sharing teams. Curves exhibiting a smoothly decreasing trend correspond to homogeneous reinforcement learning teams which share. The more jagged curves belong to homogeneous neural network teams which share. As expected, it was observed in general that for non-sharing teams the more diverse and heterogeneous the team is in terms of learning and strategy, the more opinion conflicts there are during sharing. Sharing more often causes the number of conflicts to decrease over time (a good byproduct), due to the team exhibiting more agreement over time. Reinforcement learning teams also appeared to agree more consistently compared to the erratic neural network teams. This could explain why homogeneous neural network teams performed consistently poor. Another possibility could be that overfitting is occurring due to frequent exposure to the same (or highly similar) situations.

Figures 4.18, 4.19, and 4.20 present the opinion conflict curves for non-sharing, sharing twice per trial, and sharing four times per trial, respectively. The top five for each sharing frequency for Figure 4.17 were selected from these results. From these figures, one can see that homogeneous reinforcement learning teams have more conflicting opinions, homogeneous neural network teams have fewer conflicting opinions, both which bound the number of conflicts for heterogeneous team compositions. This could be linked to the consistently good performance of reinforcement learning teams, as the goal of the task is to occupy different locations around the prey for a capture. This likely inherently requires different agents to perform different actions in the same situation, to avoid deadlock

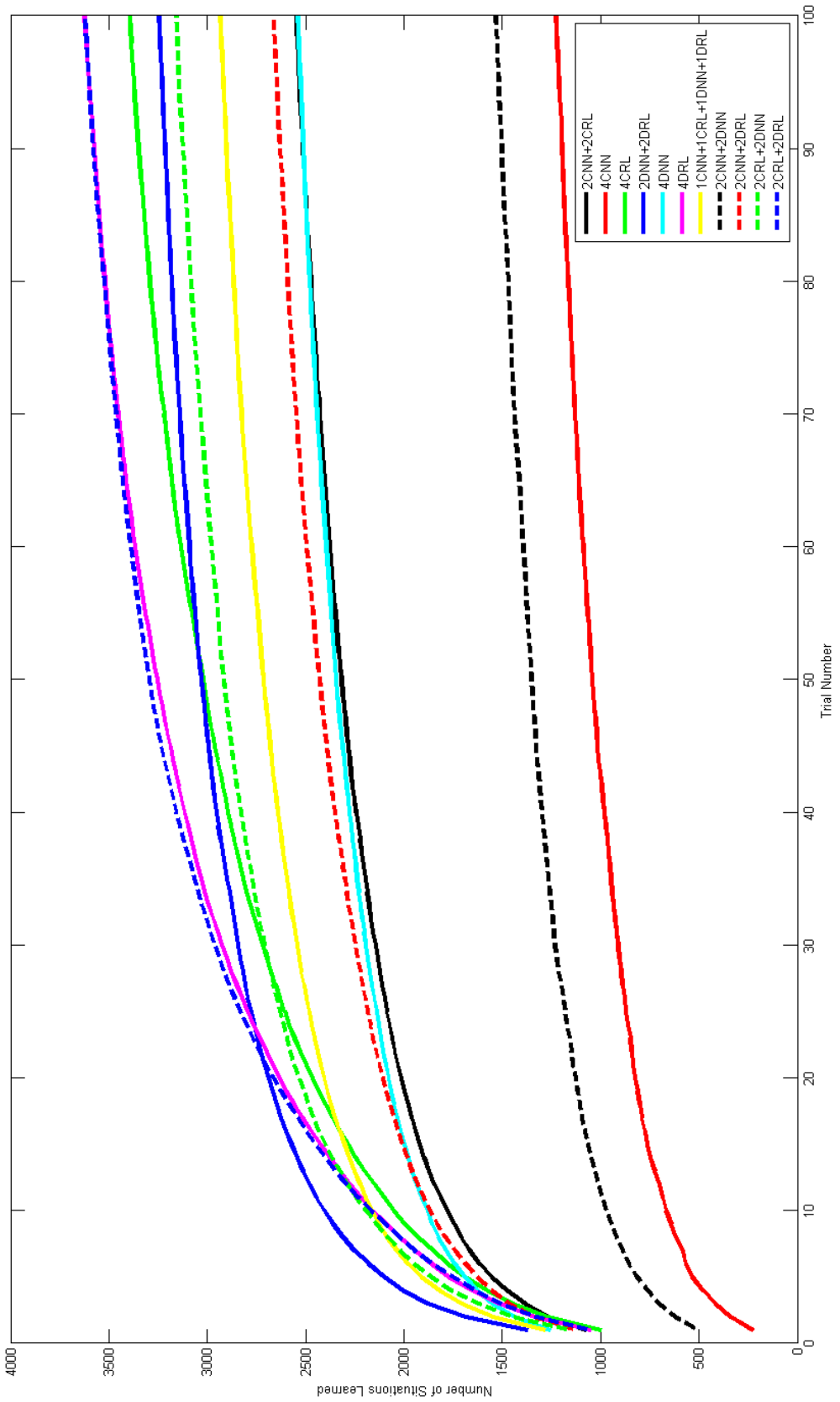

Figure 4.14: Learning curve comparison (number of learned unique situations) for non-sharing team configurations.

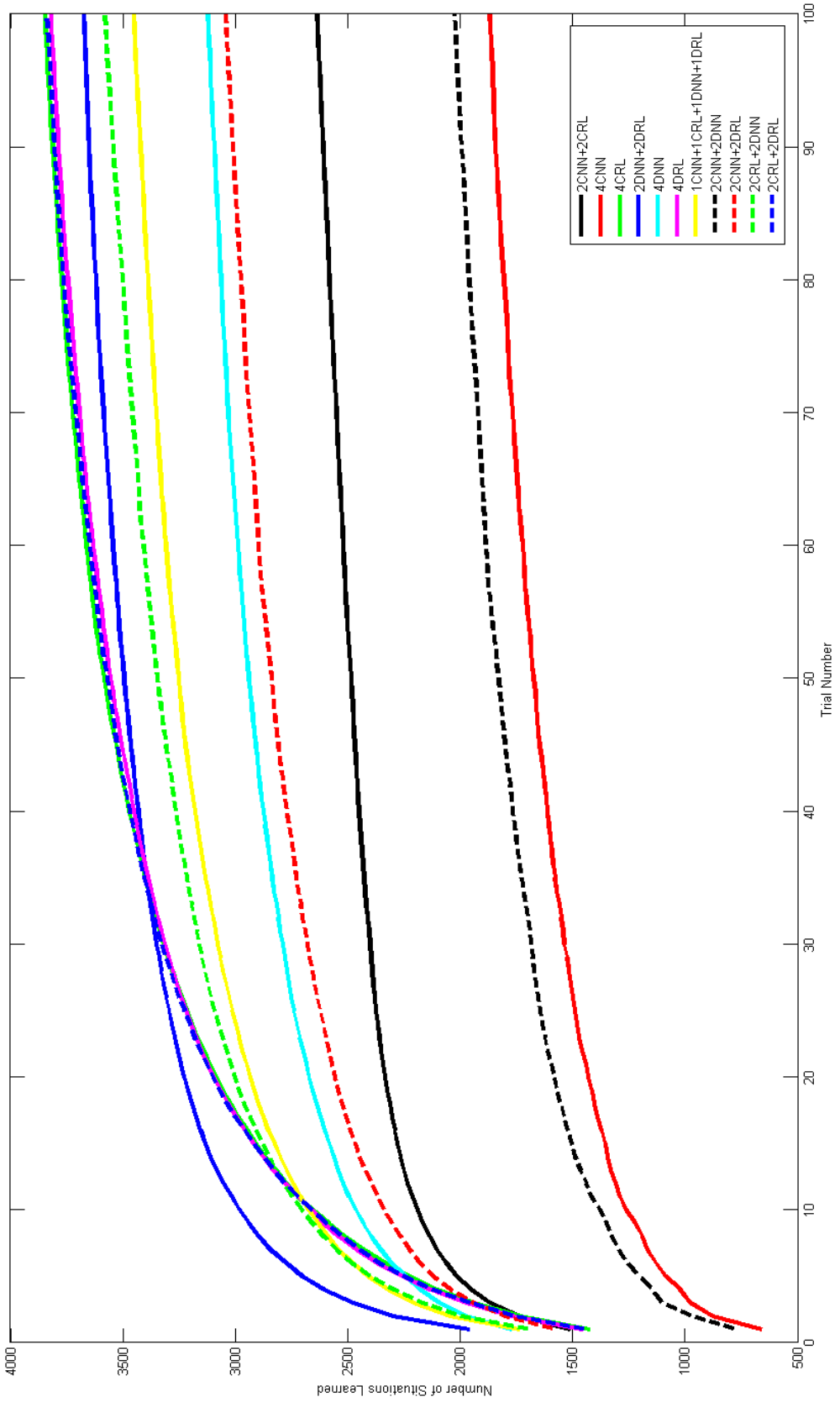

Figure 4.15: Learning curve comparison (number of learned unique situations) for team configurations which share every 50000 steps (twice per trial).

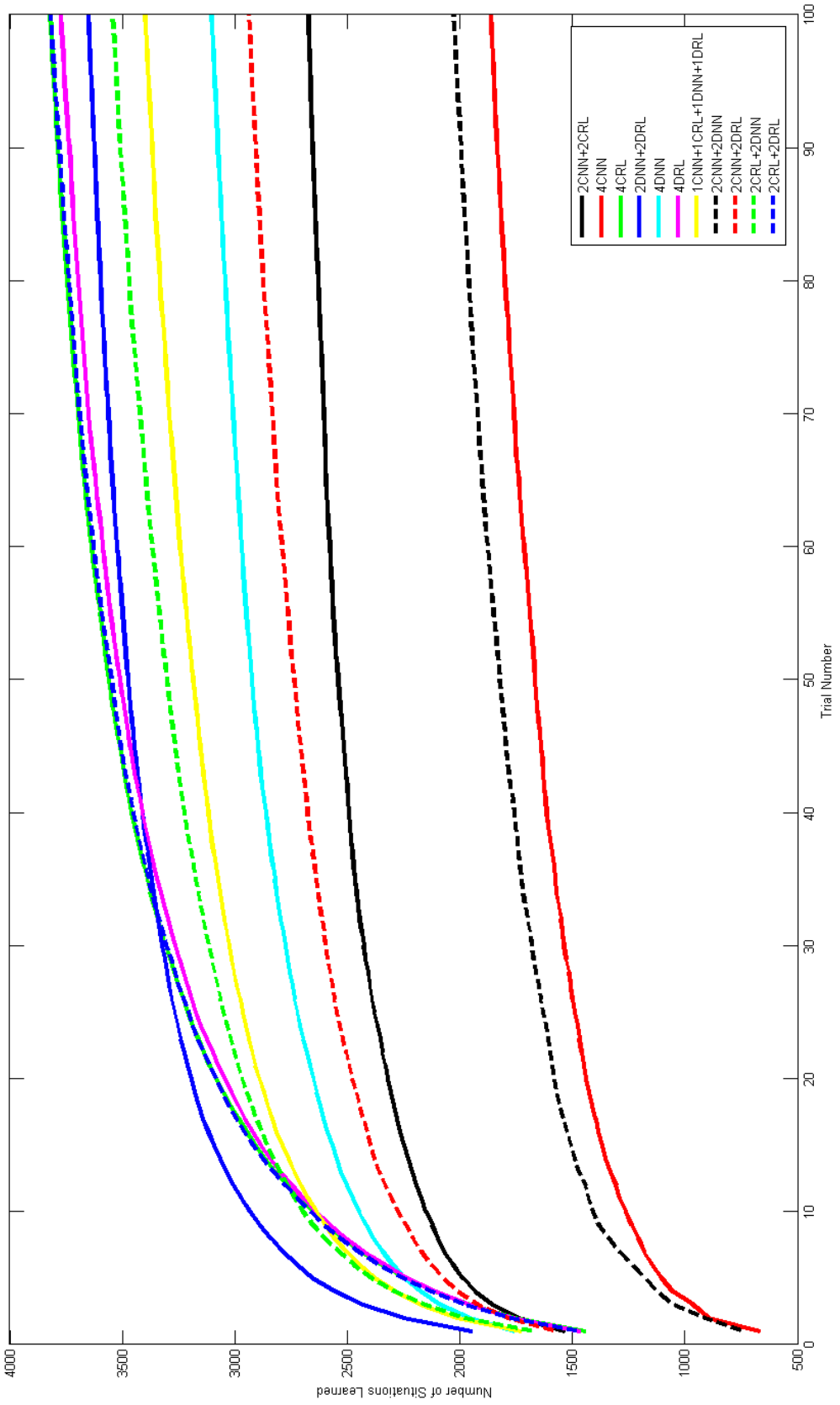

Figure 4.16: Learning curve comparison (number of learned unique situations) for team configurations which share every 25000 steps (four times per trial).

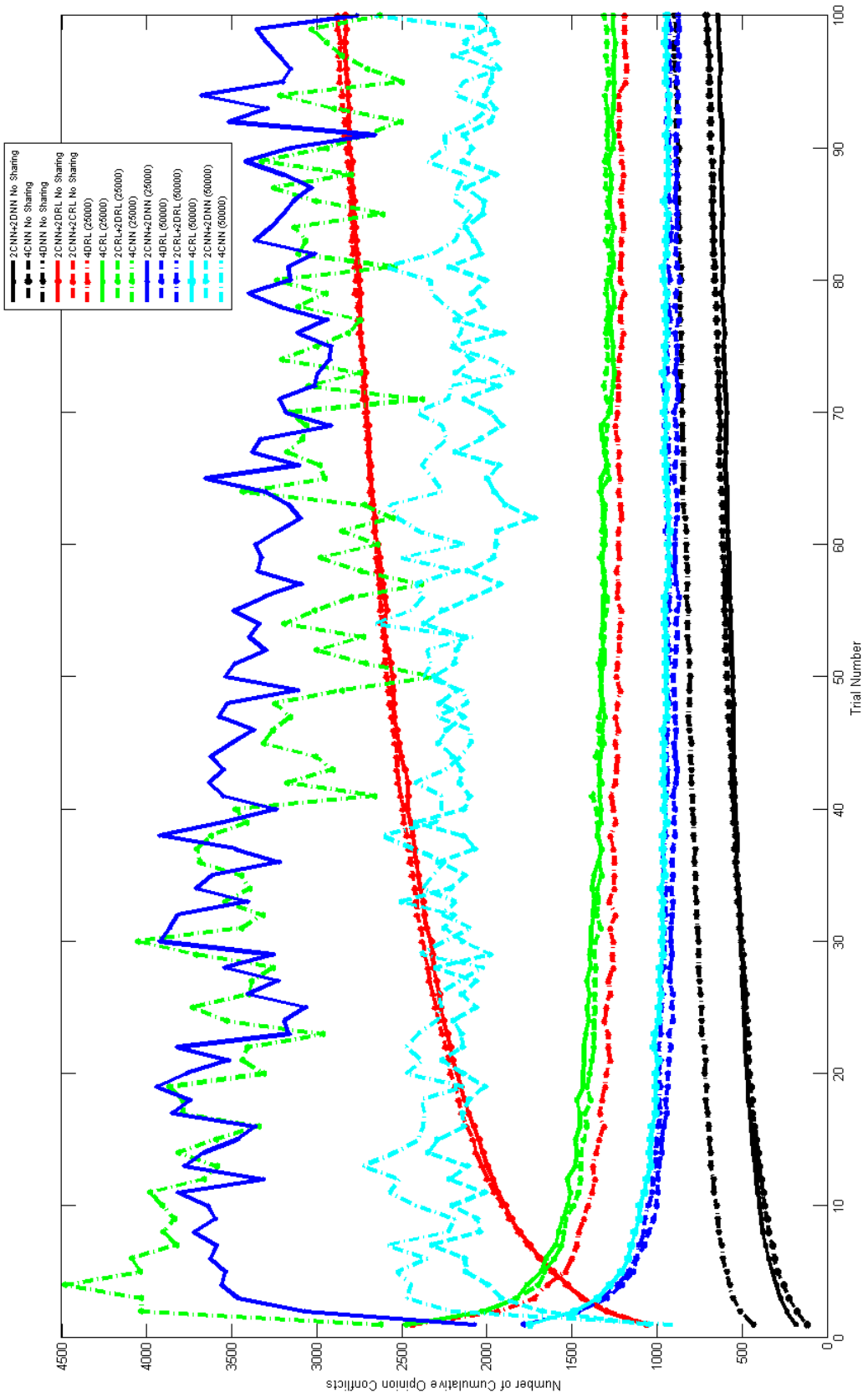

Figure 4.17: Opinion curve comparison (number of opinion conflicts) during learning for the top five sharing and non-sharing team configurations.

situations and fill an unoccupied position next to the prey to finalize a capture. Conversely, the neural network teams show fewer opinion conflicts and captures, mostly due to one or two agents fighting for a specific position around the prey. The figures showing conflict curves for sharing teams further support results that sharing reduces opinion conflicts, but sharing with lower frequency is more beneficial.

As sharing events can change the opinion of the participants, another aspect to analyze is the number of opinion changes due to sharing. The number of changes were accumulated over all sharing events per trial and averaged over all agents in the team. Non-sharing teams did not have opinion changes accumulated, as they cannot be measured properly if sharing does not take place. This merely represents another way to look at sharing. In some cases, no opinions will change (agents are in *agreement*), whereas in others several opinions will change (agents are in *disagreement*). Here, fewer opinion changes is viewed as better.

Figure 4.21 shows the opinion change curve, due to sharing and knowledge integration, for the best five sharing team configurations. Sharing less often again greatly reduced the number of opinion changes. This is seen by the difference in magnitude between the configurations which share every 25000 steps and those that share every 50000 steps. The top five teams are all homogeneous learning teams, where strategy is the only difference in team configuration besides sharing frequency. Distributed strategies performed the best, followed closely by a hybrid strategy composition and centralized teams. Similar to the opinion conflict curves, homogeneous neural network teams exhibit an erratic trend.

Figures 4.22 and 4.23 present the opinion change curves for teams which share twice per trial and four times per trial, respectively. The top five for each sharing frequency for Figure 4.21 were selected from these results. Clearly, the less sharing that takes place, fewer opinion conflicts and changes occur. Again, sharing is shown to reduce the amount of opinion changes over time, with heterogeneous teams experiencing more substantial decrease. It is also apparent that heterogeneous teams produce more opinion change, as each learning algorithm models the data/environment in different manners, the combination of which can produce opinion conflicts and resulting changes via opinion averaging.

As the conflict resolution is a major aspect of sharing, we conclude that the conflict resolution method to use depends on the team's task. For this application involving a cooperative capture, a more sophisticated method may be more beneficial to help retain specialization while still learning new experiences from other agents. An application where agents do not depend on one another for success would be a much better fit for the averaging conflict resolution method.

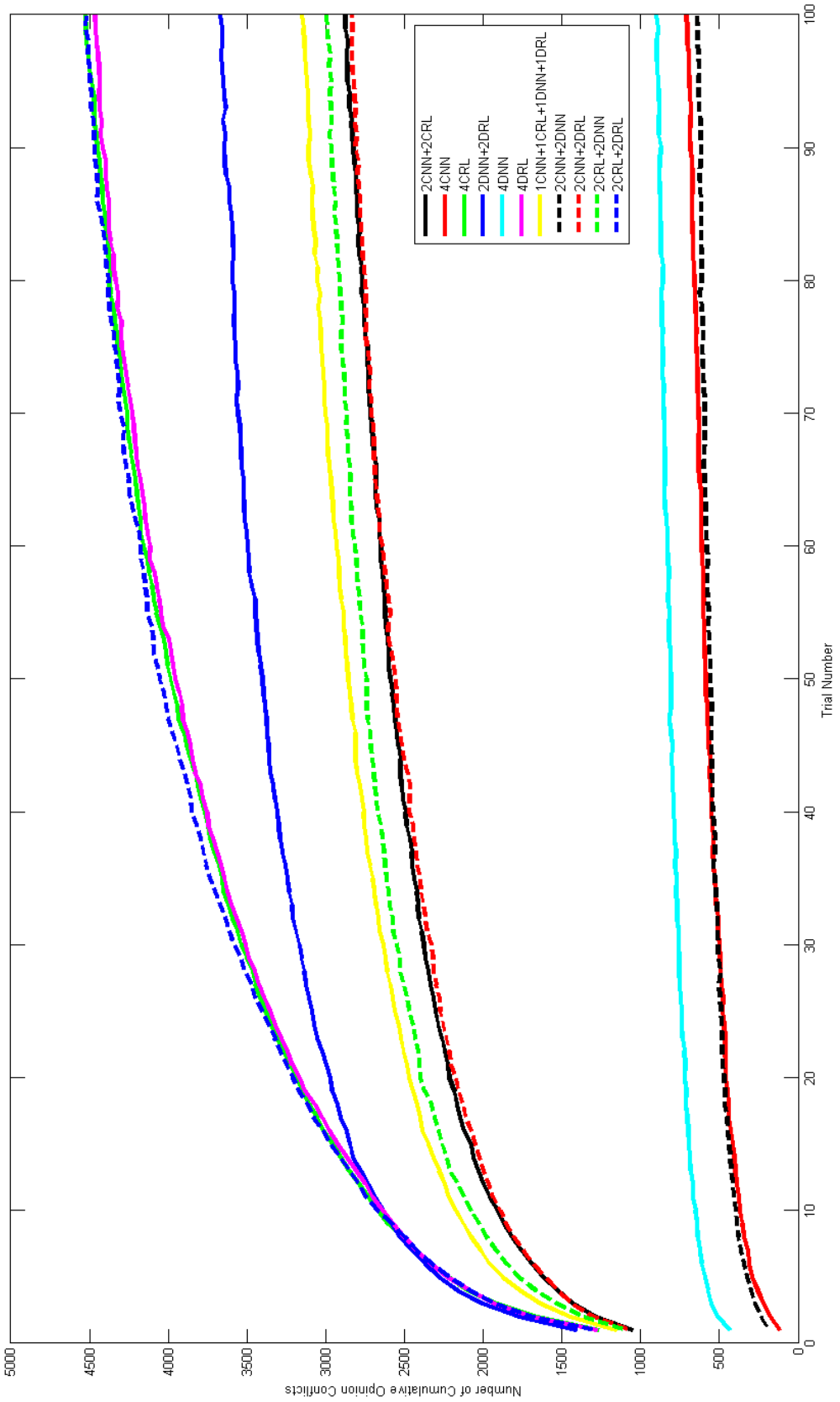

Figure 4.18: Opinion curve comparison (cumulative number of opinion conflicts) for non-sharing team configurations.

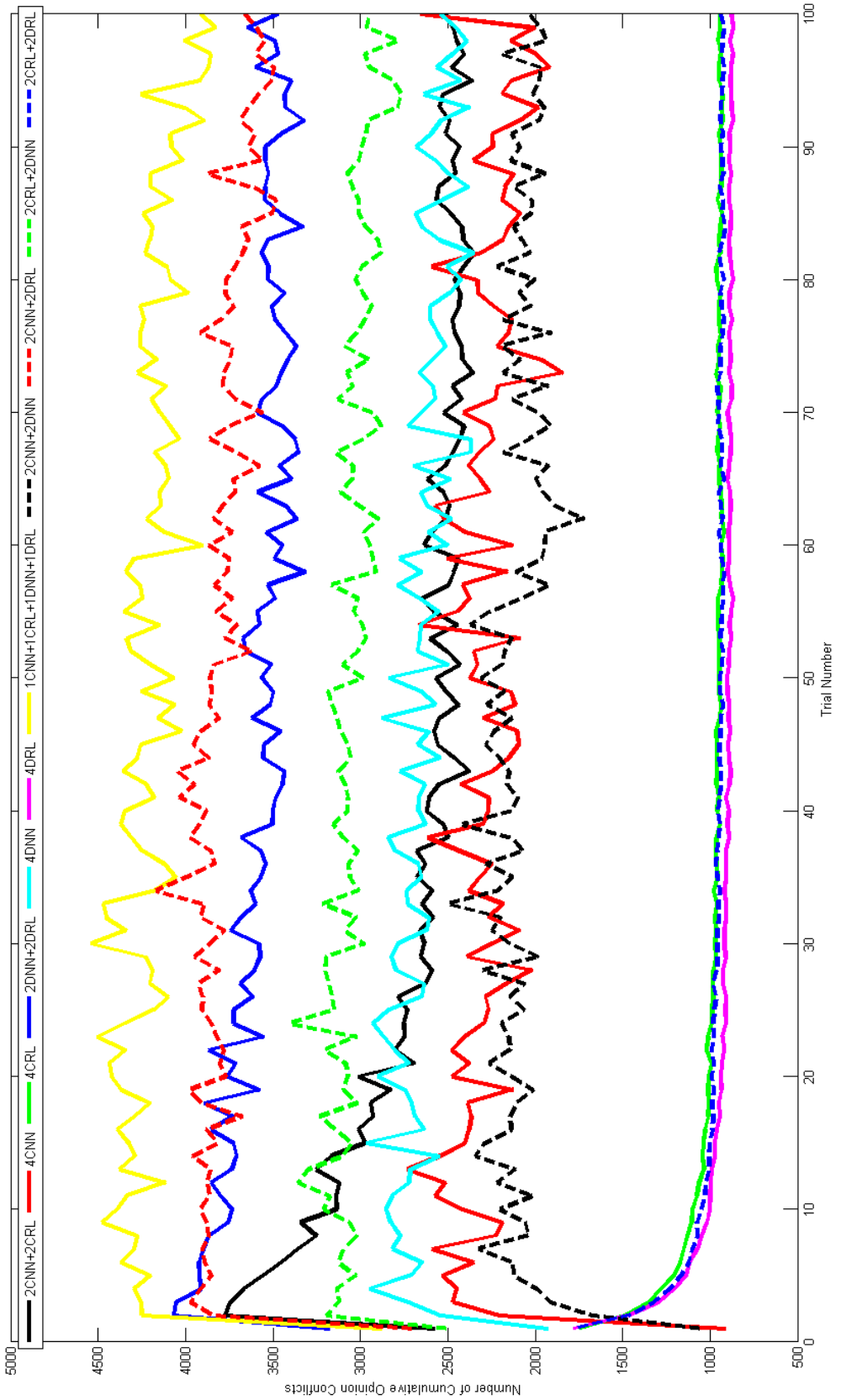

Figure 4.19: Opinion curve comparison (cumulative number of opinion conflicts) for team configurations which share every 50000 steps (twice per trial).

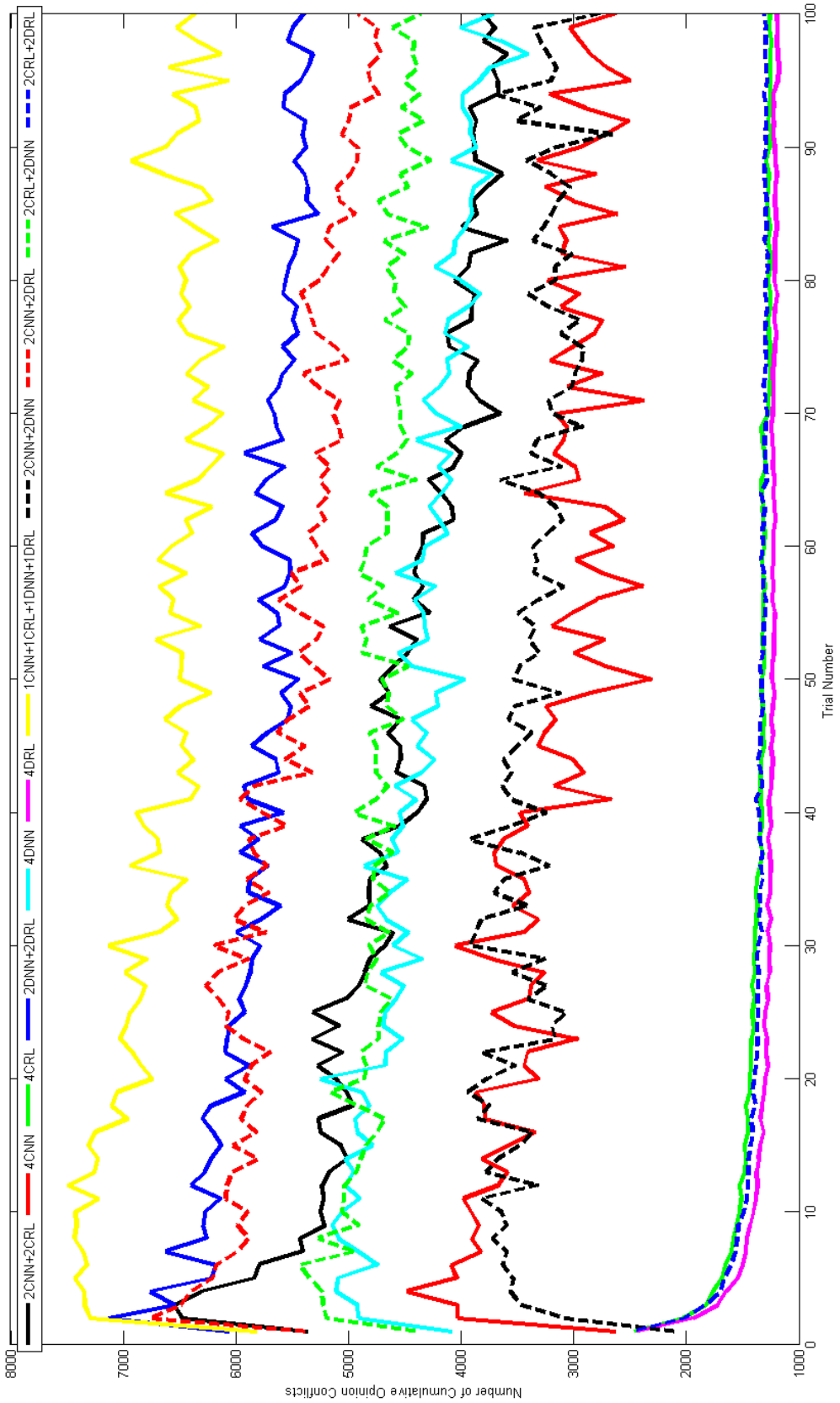

Figure 4.20: Opinion curve comparison (cumulative number of opinion conflicts) for team configurations which share every 25000 steps (four times per trial).

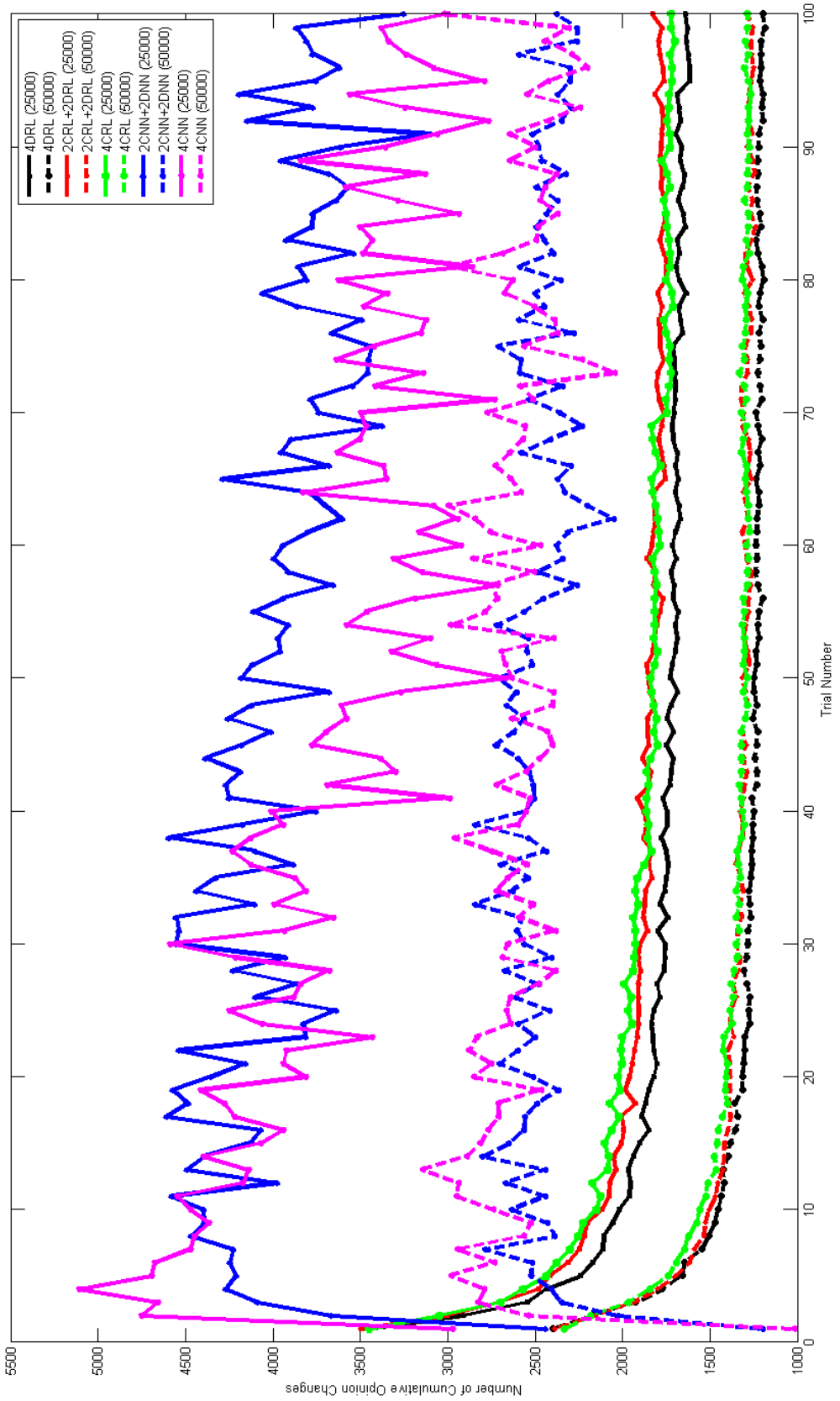

Figure 4.21: Opinion curve comparison (number of opinion changes) due to sharing and knowledge integration for the top five sharing team configurations.

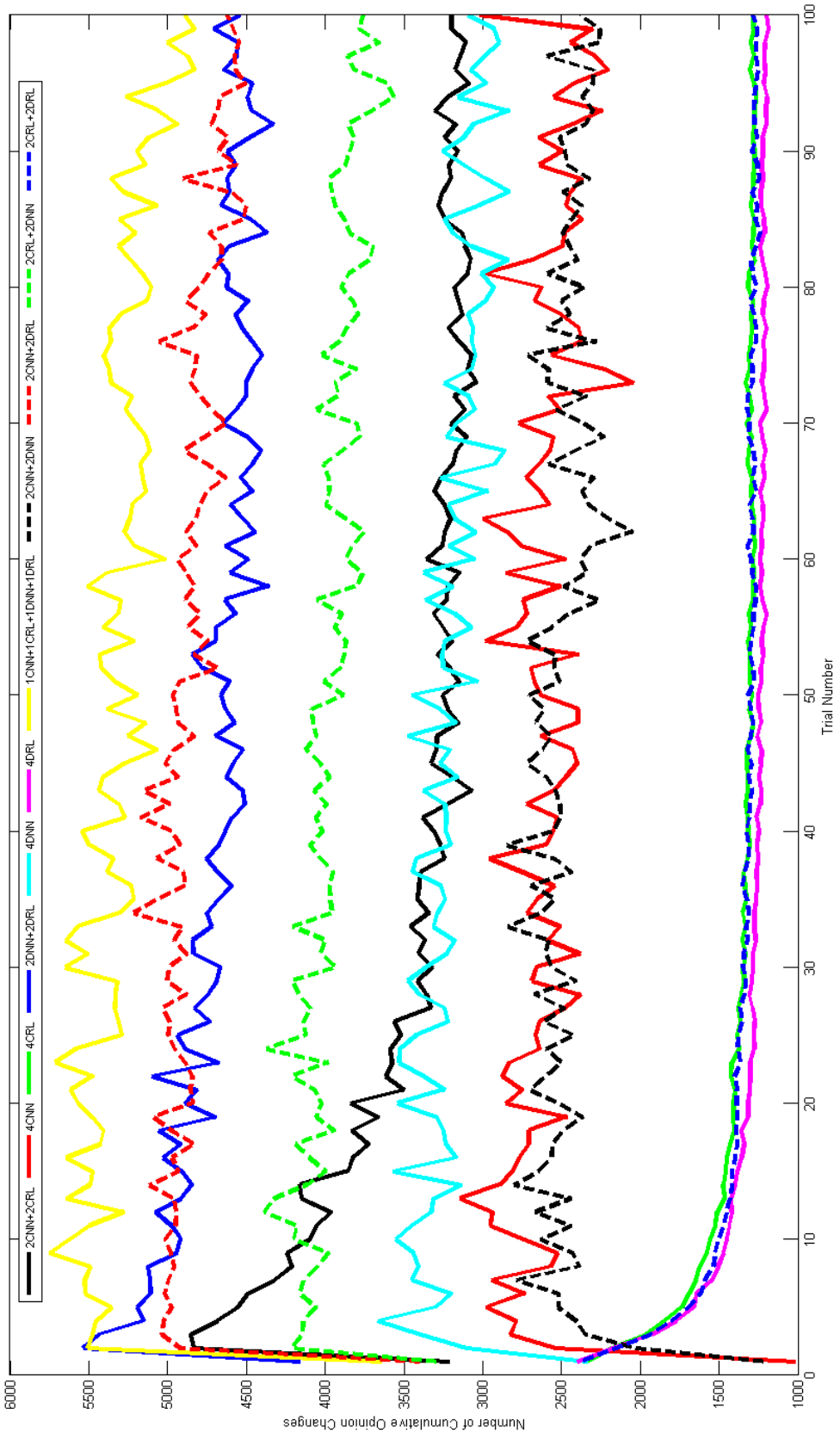

Figure 4.22: Opinion change comparison (cumulative number of opinion changes) for team configurations which share every 50000 steps (twice per trial).

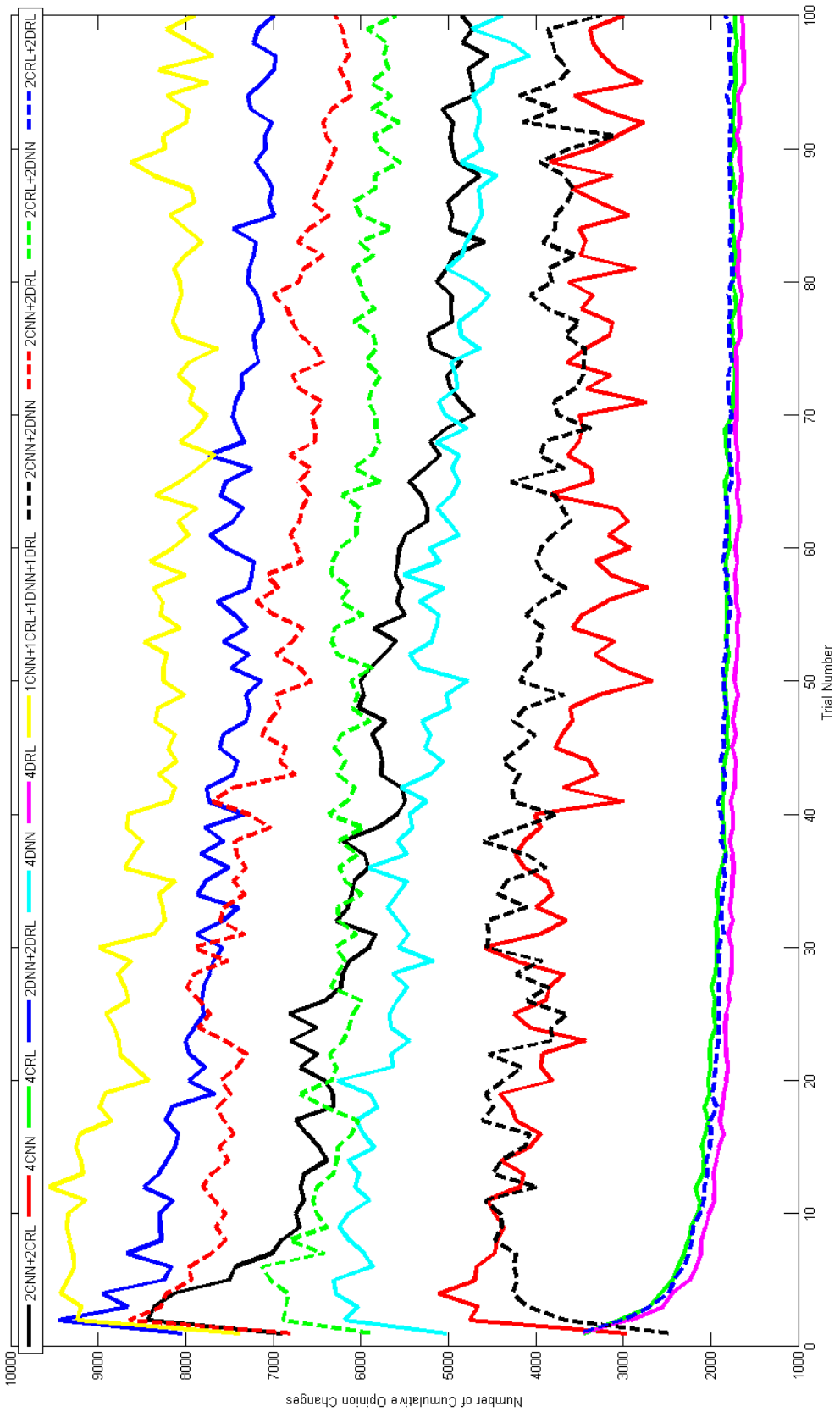

Figure 4.23: Opinion change comparison (cumulative number of opinion changes) for team configurations which share every 25000 steps (four times per trial).

4.5 Conclusions

Results have shown that using a heterogeneous mixture of learning algorithms for a concurrent cooperative task performed better than the typical homogeneous learning teams. This represents a more advanced approach to team learning. Periodic knowledge sharing also showed benefits in terms of increasing the learning rate, effectively learning more in less time. However, results suggest that sharing with higher frequency may actually reduce the learning rate, depending on the task. Hybrid combinations of learning strategies showed no clear advantage when compared to fully centralized and distributed teams.

We have also demonstrated that knowledge from heterogeneous sources can be successfully communicated and integrated by using an inter-knowledge representation common to all learners. Knowledge conflicts are extremely important to properly resolve, as they represent areas of agreement or disagreement between learning agents. The conflict resolution method depends on the application's focus. Here, we suspect that simple averaging was insufficient for combining differing viewpoints, suggesting that it may not be suitable for complex tasks. A more sophisticated approach, which retains agent specialization, may produce better results.

Some aspects of this approach are problem- and learner-specific. For example, the inter-knowledge representation may drastically change between applications. As the inter-knowledge representation changes, the conversion methods for individual learning algorithms will require changing as well. These characteristics are important, as poor choices in conflict resolution method and inter-knowledge representation can potentially lead to poor results.

Scaling team size and observing the effects of heterogeneity, sharing frequency, and conflict resolution method may provide conclusions on how to properly construct successful learning teams at varying team sizes. It may also allow for fine-tuning the sharing frequency to optimize learning rate while reducing communication bandwidth. Experimenting with several conflict resolution methods is another area we would like to further investigate. Additional research is necessary to theoretically characterize where and why heterogeneous learning teams are preferable over homogeneous learning teams. As the problem domain influences this characterization, further study into performance and generalization across problems and domains is warranted.

The broader impacts of this work will be further use of heterogeneous, collaborative multi-agent learning for complex problems. Demonstrating our approach for other applications and domains – as is done in the following chapters – is important to show that it can be used more generally, and that its utility is beyond the Pursuit Domain. We would also like to extend our techniques and approaches to physical robots, demonstrating that such concepts can be applied to mobile robots as well. Real-time team learning and sharing is difficult, but shows potential for robotics and applied artificial intelligence.

Chapter 5

Multi-Agent Geologic Facies Classification from Wireline Well Logs

5.1 Introduction

The study of rocks both on and below the surface is of geologic interest. Rocks visible on the surface are exposed to weathering and migration over time. Those beneath the surface are deformed and layered, providing a history of weathering and geologic events. In some cases, the surface rocks are the result of processes which push the subsurface material back up to the surface (e.g., volcanoes, earthquakes/faults). It is the study of these rocks and processes which make scientific contributions to both the history of the Earth and the exploitation of that knowledge to aid human civilization. This study requires geologists and geophysicists, which have the necessary knowledge to relate rock types and layering to specific events and resources. The focus of this paper is on how machine learning can be used to automate such tasks.

Gas and oil reservoirs have been the focus of modeling efforts for many years as an attempt to locate zones with high volumes. As the Earth changes over time, different rock type layers (or lithofacies) are created. These subsurface layers form time-based cyclic sequences, where layers that are superimposed on top of others are geologically younger. Certain layers and layer sequences are known to be impermeable to gas and/or liquid, such as shale. Oil and natural gas then become trapped by these layers, making it possible to drill wells (also called cores) to reach the supply and extract for use. The ability to predict where these layers and known sequences are, based on properties of core samples, provides the potential for more efficient "payzone" drilling. This process of well log analysis is an important step in geophysical exploration, as it can aid in drilling path optimization. Figure 5.1 shows an example of layered and folded rock strata.

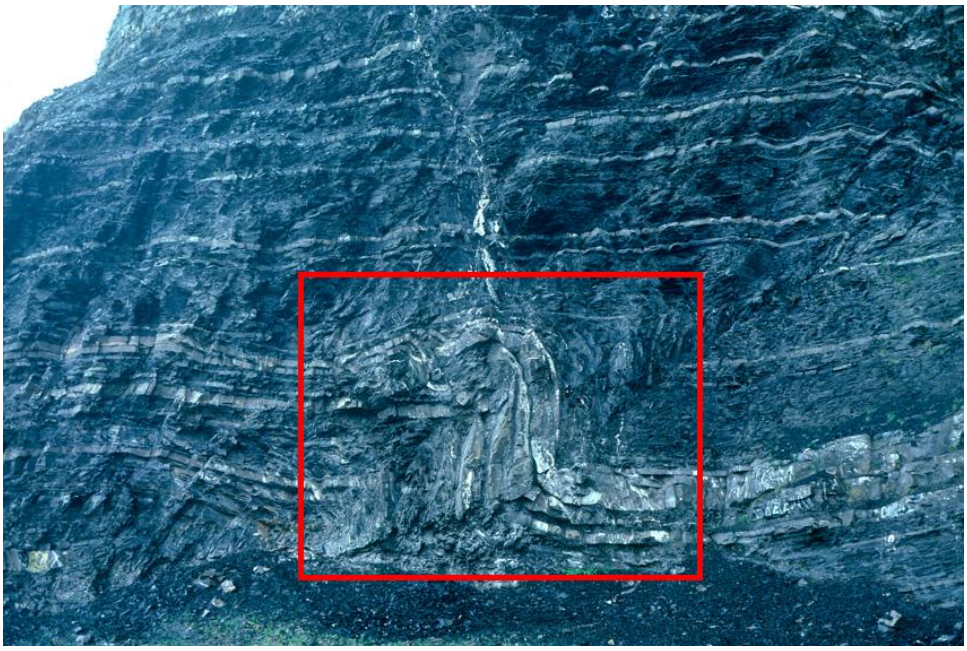

Figure 5.1: Rock strata, illustrating the differences in layers in rock type, color, age, and disruption due to geologic processes. In the outlined area, a small fold has occurred over time.

To date, rock facie (subsurface layer) classification has largely used wireline log measurements together with multivariate statistical methods, fuzzy mathematical approaches, and artificial neural networks. Wireline well logs are physical and chemical rock measurements recorded by lowering logging tools with specialized sensors into wells after they have been drilled. Properties such as permeability, porosity, and liquid content can be extracted using specific measurement mediums. Electrical measurements offer information on saturation (e.g., oil or water); while acoustic measurements can provide information about grain size. Thus, certain rocks and layers will exhibit characteristic measurement signatures based on what they contain. Experts analyze these data to determine the rock type and sequence. This not only represents a tedious and time-consuming task, but the process of drilling wells and analyzing them is costly. These costs greatly increase as the number of wells and log data attributes increase.

As we explore the solar system, the need for embodying expert knowledge and models in software greatly increases. When robots are sent to other planets, such as the Mars Exploration Rovers, they represent remote semi-autonomous geologists that need to be capable of performing science experiments and identifying surface and near-surface materials. They utilize advanced onboard science instruments to analyze rocks in-situ. Thus, being able to automatically classify subsurface and other geologic features, especially those of greatest scientific interest, becomes very attractive both on Earth and other planets.

The field of Computer Vision coupled with Machine Learning has been used in some

cases to visually inspect the surroundings and identify types of rocks that are currently in view. This is where supervised and unsupervised machine learning methods can be applied for autonomous model creation and classification. Examples of such work includes [133] and [134], which focus on automatic detection and classification of geologic features of interest using stereo vision on an autonomous mobile robot. Rocks in view are segmented, categorized, and classified into groups based on their structure and inferred geologic property signature. A belief network and other statistical methods are applied to identify and cluster rocks by specific locales. Experimental evaluation was performed with a robot in the Atacama Desert in Chile to demonstrate its potential use on planetary rovers.

Other applications of machine learning to geology include material classification while drilling for coal mine roof stability [80], mapping rock mechanical properties using seismic data [84], identification of water-flooded oil layers [112], tephra layer correlation for predicting volcanic eruptions [104], and soil classification [20]. These works demonstrate the effectiveness of the use of artificial intelligence in the geoscience fields.

In this study, we utilize wireline well log data collected in the Midwest as part of a model-creation effort for a rock facies classification problem. Specifically, a detailed study of team learning, collaboration, and decision combination has been conducted. Experimental results are presented, including successful team compositions, individual learning algorithm contribution, team size, collaboration frequency, and team diversity. Finally, the primary team learning and collaboration aspects discovered as part of this application are summarized.

5.2 Background and Related Work

There are traditional log curves that are used in practice to differentiate between rock facies, each measurement of which offers an important piece to be able to classify rock content. The following are a few of these curves [113]:

- *Resistivity*: Electrical measurement that detects vertical changes in conductivity; Resistivity is the reciprocal of Conductivity. Different liquids respond differently to this measurement (e.g., salt water conducts electricity, whereas oil does not).
- *Gamma Ray*: Device measuring natural radiation. Formations that contain higher amounts of organic material (such as shale) emit more radiation, resulting in higher gamma ray counts.
- *Neutron*: Device measuring hydrogen atom presence. Given the assumption that pores are filled with either water or hydrocarbons, this provides information about pore space.
- *Acoustic*: Sonic tool measuring porosity and potential gas content of formations.

Figure 5.2 provides an example illustration of well log parameters collected from a well by the Kansas Geological Survey [43].

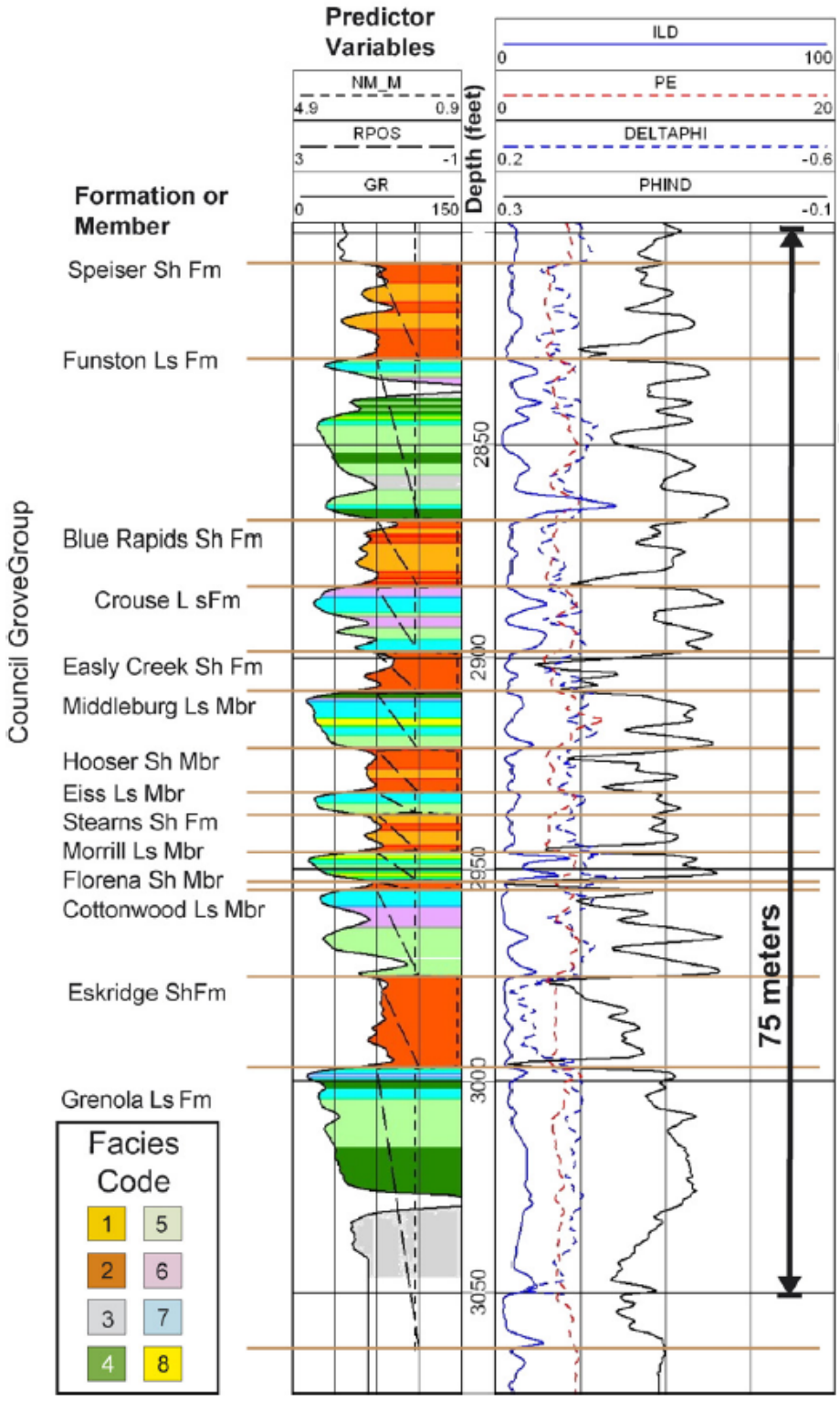

Figure 5.2: Example wireline well log data, including digital parameters, collected from a well by the Kansas Geological Survey [43]. Rock facies type varies with depth, and the digital log data varies based on rock type.

Several research efforts have applied machine learning methods to the problems of reservoir and rock formation identification. Machine learning fits well into this problem as it has the potential to make the process more efficient. Using existing wireline well log data, algorithms can be used to create (or learn) a model that relates the log measurements to rock facies. Once these models are created, they can be validated on other known well sequences to evaluate their accuracy. Furthermore, information from these learned models can be used to intelligently interpolate subsurface behavior between well locations, allowing for general 3D subsurface models of an area. However, due to spatial distribution and heterogeneity of subsurface properties, this yields additional complexities.

Researchers have used a combination of data mining and an expert system to perform intelligent well log analysis with data from the Xinjiang Province, China [113]. That study included boreholes containing subsurface layers of water, mixture of water and oil, oil, and shale. The RIPPER inductive learning algorithm was used for its efficiency and human-readable production rules. Decision trees have also been used for facies identification. In [8], the C4.5 decision tree learning algorithm was used to classify a total of five facies with a variety of natural and synthetic variables. The primary attributes used in the study were gamma ray, bulk density, neutron porosity, and depth information. Several other variables were computed from these attributes, and incrementally provided to the decision tree to increase overall testing accuracy.

In naturally occurring strata, multiple geologic facies exist and are non-linearly coupled. Over time, processes deform these layers in complex ways. Boundaries between them are difficult to properly model, and represent areas of classification difficulty. This translates to a multi-class classification problem with non-linear decision boundaries. Support vector machines (SVM), due to their relatively recent surge in popularity and rigorous mathematical basis, have also been applied to this problem of facies delineation. Synthetic hydraulic conductivity data were utilized in [129] to study SVM and statistical method performance for the idealized problem of identifying the stratified boundary between two layers (a two-class, linearly separable problem). The SVM was found to outperform the geostatistical approach for boundary estimation. This work was extended [148] for the non-linear boundary case using similar synthetic data. It was found that error decreased logarithmically with increasing sampling density.

Neural networks have seen more widespread use in geophysical applications (e.g., [143] includes a review of neural network use in these applications, and [150] presents a porosity prediction example). The work presented in [107] used unsupervised analysis to segregate wells into classes based on well logs, and then applied a supervised learner to create a model for estimating lithology in horizontal wells in Saudi Arabia. Their method also allowed for confidence measures given facies data uncertainty. Unsupervised analysis took place using a single-layer neural network, where the neuron most resembling the input is rewarded by progressively moving closer to the center of the corresponding input data cluster (termed

"competitive learning"). Intuitively, confidence was found to significantly decrease at facie transition zones.

Some researchers have concluded that more advanced machine learning methods are required to increase accuracy beyond that of single classifiers. Multi-classifier systems, such as neural network ensembles and committee machines, have been evaluated for lithology recognition and reservoir characterization. Further, the use of multiple classifiers brings about methods for combining their decisions.

Ensembles of neural networks, sometimes called committee machines, were used in [108] with data from a Brazilian offshore well. Different committees of neural networks with identical structures (single hidden layer) were developed via approaches of pattern replication (one neural network per class), bootstrapping (series of patterns created using random sampling with replacement), and iterative pattern selection (sample probabilities for misclassified patterns iteratively increased). Classifier combination methods of averaging, majority vote, rank counting, and fuzzy integrals were assessed and compared against the performance of a single neural network. Gamma ray, acoustic, bulk density, resistivity, caliper, and depth information were used as attributes for three rock classes. For each committee formation method, several experiments were performed with up to 100 networks per committee. The fuzzy combination method was found to provide the highest accuracy. It was also found that having more neural networks generally offered increased classification accuracy. The best committee machine outperformed the reference neural network by over 7%. This study was an extension of [42], which removed the caliper attribute and increased the classes from three to eight. The authors also separated the wells into two categories: entire well and reservoir (depth greater than 2500 meters). Results were similar, showing that pattern replication worked well for the case of non-stratified data sets.

The work in [19] and [18] also utilizes several neural networks for wireline and measurement-while-drilling analysis to predict porosity, permeability, and fluid saturation. These properties are key for determining flow patterns. The measurement-while-drilling approach offers a more real-time look at reservoir properties, allowing the drillers to steer their efforts based on subsurface properties (called "geosteering"). Each property-based neural network in this study is identical with individual randomized starting weights; each being trained on the same pattern or a subset, but focusing on a single property (e.g., permeability, porosity). The outputs of each network are combined using optimal linear combination, which aims at reducing variance. The authors noted that such an approach is useful as each neural network typically found differing local minima. For lithofacies prediction, each neural network was trained to be an expert on a single facie, rejecting others. A gating network was also utilized to introduce prior knowledge of geology (i.e., local stratigraphy) into the network models. Facies prediction for the Ness Formation achieved an average hit-rate of above 90%.

From these works, it has been shown that using machine learning to create geophysical models for well log analysis is not only feasible, but can provide high levels of accuracy while offering a significant increase in efficiency. In the following sections, we discuss our approach, using multiple heterogeneous collaborative learning agents to further increase accuracy for the problem of well log analysis and facie classification. Experiments focused on agents' team size, composition, collaboration frequency, and data division results are presented. This approach could also be applied to measurement-while-drilling, but is not explicitly discussed here.

5.3 Kansas Geological Survey Well Log Data Set

This section describes the data set statistics, its collection details, and recent machine learning and model-creation efforts by the Kansas Geological Survey. We have utilized this data set for the multi-agent collaborative learning study.

5.3.1 Data Set Properties

The Kansas Geological Survey (KGS) [72] has been studying the Hugoton and Panoma Fields in Southwest Kansas and Northeast Oklahoma to create subsurface models [43][21][45] [44]. These fields comprise the largest gas-producing area in North America as of 2007, with 963 billion mm^3 of gas from over 12,000 wells [43]. With such a large field and possibility for many core sites, manual study of wireline logs becomes very time consuming and impractical. Automatic methods, such as the use of machine learning algorithms or statistical approaches, are therefore desirable.

KGS has focused primarily on comparing four individual model-creation methods: Bayesian, K-Nearest Neighbors (KNN), Fuzzy Logic, and Back-propagation Neural Networks. KGS created a data set for the Council Grove Group in the Panama Field, which was divided into eight rock facies, each representing a class (type) of rock:

1. Continental origin, coarse siltstone
2. Continental origin, shaley fine siltstone
3. Marine origin, marine siltstone
4. Marine origin, carbonate mudstone
5. Marine origin, wackestone
6. Marine origin, fine-crystalline dolomite
7. Marine origin, packstone
8. Marine origin, grainstone

The order of these facies generally translates to increased permeability for a given porosity, and adjacent classes mostly exhibit similar properties. Facies 6, 7, and 8 represent the primary gas payzone facies.

Table 5.1: Testing results obtained by KGS using four independent learning methods.

Learning Method	Approach	Accuracy
Bayesian	Traditional quadratic Bayesian method	62%
Fuzzy Logic	Used a fuzzifier to turn the 7-element input into a 56-element feature vector by computing a degree of membership for each element to all eight classes.	62%
KNN CDF	K=20 nearest neighbors	67%
Neural Network	Network comprised of 7 input nodes (attributes as inputs), a single layer with 50 hidden nodes, and 8 output nodes (probability for each class/facie). Damping parameter of 0.1 and iteration 100 times.	78%

Seven numeric measurement features/attributes are used in the data set. These attributes are utilized to model the data, using rock type as the target for training.

1. RPOS: Relative position of sample from marine/continental half-cycle boundary
2. NM-M: Non-marine/marine indicator (sample origin)
3. GR: Natural gamma radiation
4. RTA: Apparent true resistivity
5. N-D: Average neutron and density porosity
6. PHIND: Neutron and density porosity difference
7. PE: Photoelectric effect

The addition of the marine/non-marine indicator and relative position of a sample from the boundary of its indicator were included to incorporate geologic knowledge. These features were recorded at half-foot intervals within the wells. Some wells do not contain the PE feature, so the data set was separated into one set with and one set without this feature for independent study. Samples not containing PE were removed from the data set that includes this feature. Our research using this data set only focuses on the data set containing PE measurements (a total of five geographically distributed wells).

KGS performed a detailed study to compare the four classification methods on both data sets [43]. As facies close to one another are generally similar, the authors computed absolute accuracy, within-one accuracy, and accuracy on a specific set of payzone facies known for reservoir storage and flow. Within-one accuracy was important in their study, as they wanted to maximize accuracy on a specific set of facies (classes), where being within one facie of being absolutely correct was acceptable. As our work focuses on the PE data set and absolute accuracy for all facies, Table 5.1 summarizes absolute accuracy results they achieved on the testing portion of this data set [43]. The data was randomly split into two-thirds training and one-third testing portions.

The authors note that the fuzzy logic classifier was not effective in classifying some facies, two of which were payzone facies. The KNN classifier showed a lower absolute correctness for the payzone facies, but improved the overall absolute accuracy. Finally, the

Table 5.2: Attribute statistics for the KGS data set.

Attribute	RPOS	NM-M	GR	RTA	PHIND	N-D	PE
Maximum	1	2	361.15	32.107	30	28.2	6.321
Minimum	0.0098	1	12.036	1.24	0.55	-20.74	0.096
Average	0.5233	1.5211	63.021	5.3459	12.2258	4.4465	3.7544

neural network did a fairly good job of predicting facies, but also had some trouble with the payzone facies. The final neural network that was used in their work was the culmination of many cross-validated tests to determine the best network architecture for this data set. Therefore, the 78% achieved by the neural network represents the current best performance that has been achieved using any learning approach with this data set.

5.3.2 Data Set Statistics

KGS results implied that different learning algorithms had difficulty with different areas and classes of the data set. This prompted us to look to classifier combination and collaborative learning to improve overall absolute facies classification accuracy. This is a difficult classification problem, as rock properties controlling gas production overlap and wireline logging tools tend to average and blur the facies boundaries [44]. Although the KGS neural network was able to achieve a high payzone accuracy, our work is more concerned with improving the overall absolute accuracy for all facies.

The same KGS data set was obtained to perform our collaborative multi-agent learning study. Statistics of the entire (PE and NoPE combined) data set are presented in [43]. Our study is focused on use of all seven attributes (the portion of the data set containing the PE measure). All instances with missing PE values were removed, resulting in a total of 2274 training and testing instances. These instances were randomly split and stratified into two-thirds training and one-third testing portions, the same training and testing setup utilized by KGS to create their models and compare their classifiers. Each attribute is normalized prior to learning, with the testing data being normalized using the training data's mean and standard deviation.

Tables 5.2 and 5.3 present the statistics of this data set, including the seven attributes and distributions for the eight classes (facies) which form the classification problem. These facies distinguish between rock type and texture, and were selected to maximize distinguishable wireline log curves, minimize the number required to represent the reservoir accurately, and distinguish between porosity/permeability/pressure relationships [43]. The boundaries between facies represent the most difficult classification area, as the boundaries can be blurred by the logging sensors.

The structure of the features reveals the difficulty of classifying facies based upon it. Considering two features with the lowest variance (GR and RTA), one would expect a significant amount of overlap. As shown in Figure 5.3, the data exhibits substantial overlap.

Table 5.3: Class distribution for the KGS data set, including training and testing portions.

Class	1	2	3	4	5	6	7	8	Total
Entire Data Set	549	539	170	152	422	87	261	94	2274
Training Set (Two-Thirds)	366	359	113	102	281	58	174	63	1516
Testing Set (One-Third)	183	180	57	50	141	29	87	31	758

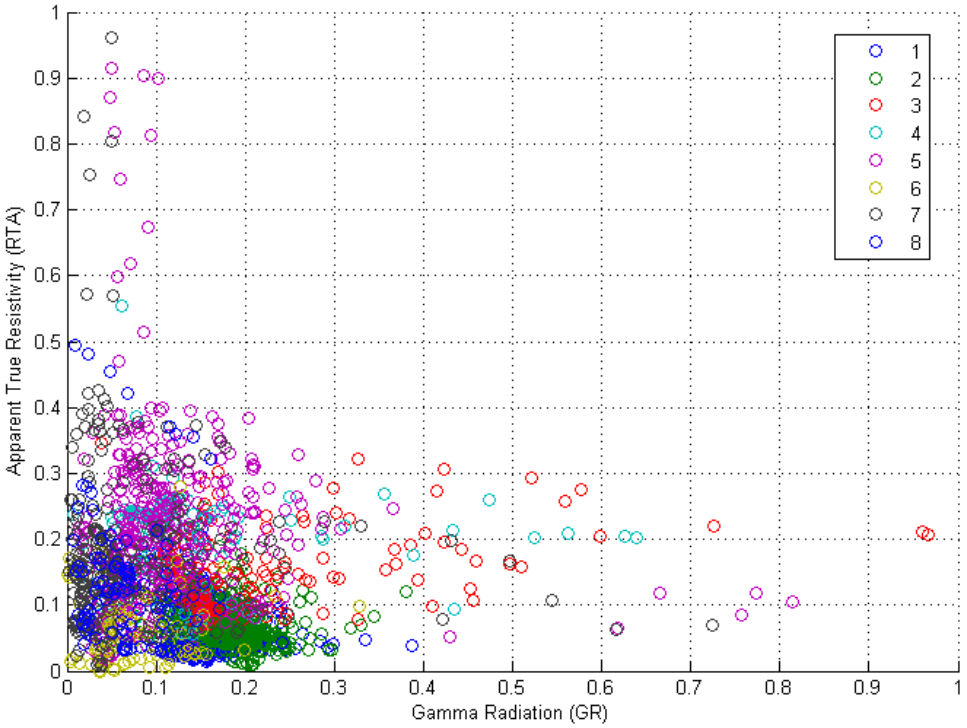

Figure 5.3: Overlap for two lowest variance features by facie/class: Gamma Radiation and Apparent True Resistivity.

Each individual attribute in the data set has been normalized from 0 to 1. Most facies appear to be somewhat evenly distributed from 0 to 0.4 for gamma radiation (GR), and from 0 to 0.4 for apparent true resistivity (RTA). The most pronounced differences between two facies are between facies 3 and 8, which also substantially overlap.

As for the two features with the highest variance (NM-M and RPOS), Figure 5.4 shows even these exhibit substantial overlap. In theory, this separates the facies into two clearly differentiable groups. However, even this distinction is not absolute. For example, while most of facies 8 is denoted marine, some of its samples are also non-marine. Additionally, the next highest variance feature, relative position (RPOS), does not allow for much differentiation. It is difficult to distinguish on this dimension where one facies begins and another ends. These aspects illustrate the difficulty of this classification problem.

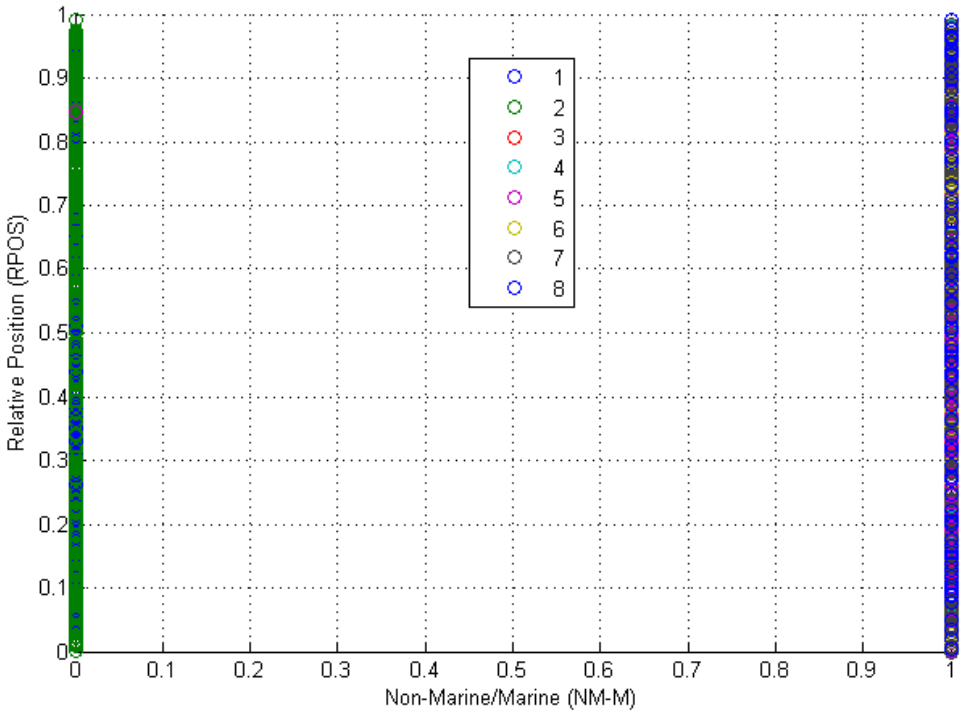

Figure 5.4: Overlap for two highest variance features by facies/class: Non-marine/Marine and Relative Position. The Non-marine/Marine indicator is binary (0 or 1).

5.4 Experimental Setup

The Multi-Agent Collaborative Learning Architecture, together with the WEKA machine learning suite, was utilized to study aspects of team learning and collaboration. As discussed in Section 3.2, an accuracy-based tournament-style series of experiments were performed. A summary of the learning algorithms that are used as part of this study is presented in Section 2.4. Note that we have also included the artificial neural network and K-nearest neighbors classifiers (along with various others) to directly compare results with those achieved with this data set by KGS. The goal of this study was not only to study various aspects of team-based learning and collaboration, but also to exceed the best accuracy achieved by the single classifier systems utilized by KGS. The following section presents experimental results collected as part of this study.

5.5 Experimental Results

This section presents the experimental results on various aspects of team learning and collaboration from the KGS study. These aspects are comprised of independent team-based studies that allow us to make conclusions and formulate heuristics for team learning with this data set. These results will be combined with those from Chapters 4 and 6 to generate

a set of novel, general heuristics for applying team-based learning to other domains.

The notation used in the subsequent figures and analyses are abbreviations of learning algorithms, collaboration frequencies, and data distribution modes. The abbreviations for the 11 learning algorithms are: Naive Bayes (NB), Decision Tree (DT), Instance-Based KNN (IBK), Neural Network (NN), Logistic Regression (LGR), Radial Basis Function Network (RBF), K* (KST), Decision Table (DTB), RIPPER Rule Learner (JRP), PART Rule Learner (PRT), and Random Forest (RFT). When listing team compositions, the number of learners for each learning algorithm of the team is listed, separated by "+", followed by the collaboration frequency for that team. For example, "3ibk+2nn+2dt+1lgr (med)" represents a team of size eight composed of three IBK, two NN, two DT, and one LGR classifiers which collaborate five times (Medium) during learning. Collaboration frequencies are broken down into None (0 collaborations per experiment), Low (2), Medium (5), and High (8). Finally, Full and Independent (Indp) data learning distribution modes are also abbreviated as shown.

A total of 918 learning experiments were performed, including the variation of team size, composition, collaboration frequency, and learning mode. Of these, 459 were Full mode team learning experiments, and 459 were Indp mode team learning experiments. For each learning mode, experiments are broken down as follows:

- 11 size one, 264 size two, 100 size four, and 84 size eight teams
- 112 teams for each collaboration frequency (None, Low, Medium, and High)
- 95 homogeneous (including size one teams) and 364 heterogeneous teams

5.5.1 Decision Combination Method

Decision combination methods are the means for combining classifications from multiple learning algorithms to formulate the team's collective decision for an instance requiring classification. In the following figures and tables, each count for a combination method represents an experiment which resulted in that combination method providing the highest testing accuracy. Ties (in terms of combined decision testing accuracy from all learners) count as a win for each method. The reported win percentages have been normalized, as there are varying numbers of teams of different sizes and there are more heterogeneous teams than homogeneous teams, due to heterogeneous teams generally offering higher classification accuracy.

Table 5.4 presents the most successful five decision combination methods over all Full and Indp mode team experiments for this data set. In general, team size dictates the combination method to choose. Regardless of learning mode, Average, Selective Average, Multiply, and Selective Max are successful methods for the experiments. Selective Multiply works very well for Full mode teams, whereas Weighted Vote performs fairly well for Indp mode teams. This table suggests that the Multiply method is superior for Full teams, while Average is most successful for Indp teams.

Table 5.4: Most successful five decision combination methods, based on combined testing accuracy, over all experiments for Full and Indp learning distribution modes.

Full Mode		Independent Mode	
Combination Method	**Win Count**	**Combination Method**	**Win Count**
Multiply	189	Average	198
Selective Multiply	176	Selective Average	107
Average	160	Multiply	98
Selective Average	160	Selective Max	98
Selective Max	137	Weighted Vote	89

Figure 5.5 presents results for Full versus Indp mode teams as a function of team size. For Full mode teams, any of Average, Multiply, or Selective Multiply are good combination methods. Selecting the combination method to optimizing performance, however, depends on the team size: Multiply for size two, Selective Multiply for size four, and Average for size eight teams. The Average combination method is the consensus method to select for all Indp team sizes. Max, Majority Vote, Weighted Vote, and Select Best consistently perform poorly for all team sizes and learning modes. For Indp teams of size four, all methods other than Average perform poorly.

Figure 5.6, focusing on homogeneous versus heterogeneous teams, also shows that Multiply and Selective multiply perform best for Full mode teams, while Average performs best for Indp mode teams. Average also performs best for homogeneous team compositions. All other combination methods perform considerably worse for homogeneous Indp teams, whereas all combination methods perform well for homogeneous Full teams. Average, Multiply, and Selective Multiply are good combination methods to choose for heterogeneous teams. For Full mode heterogeneous teams, Multiply or Selective multiply should be selected. For Indp mode heterogeneous teams, Average should be selected for combining decisions.

Combination method results for Full and Indp mode teams as a function of collaboration frequency are presented in Figure 5.7. For teams which do not collaborate or utilize Medium collaboration, the Average (for Indp mode) and Selective Multiply (for Full mode) methods should be selected. Low and High collaboration teams should utilize Average (for Indp mode) or Multiply (for Full mode). It appears that collaborating with Medium frequency decreases the variance between combination methods for both Full and Indp learning modes. We again see that the more sophisticated decision combination methods (such as Majority Vote, Weighted Vote, and their Selective variations) perform similarly and with comparatively lower success than the more simple combination methods. Specific combination methods appear to be geared more toward Indp mode team classification (non-overlapping, distributed data set modeling); whereas Full mode teams experience a more distributed success from all combination methods that were studied. Overall, averaging class probabilities performs consistently well for teams of collaborating learners for this application.

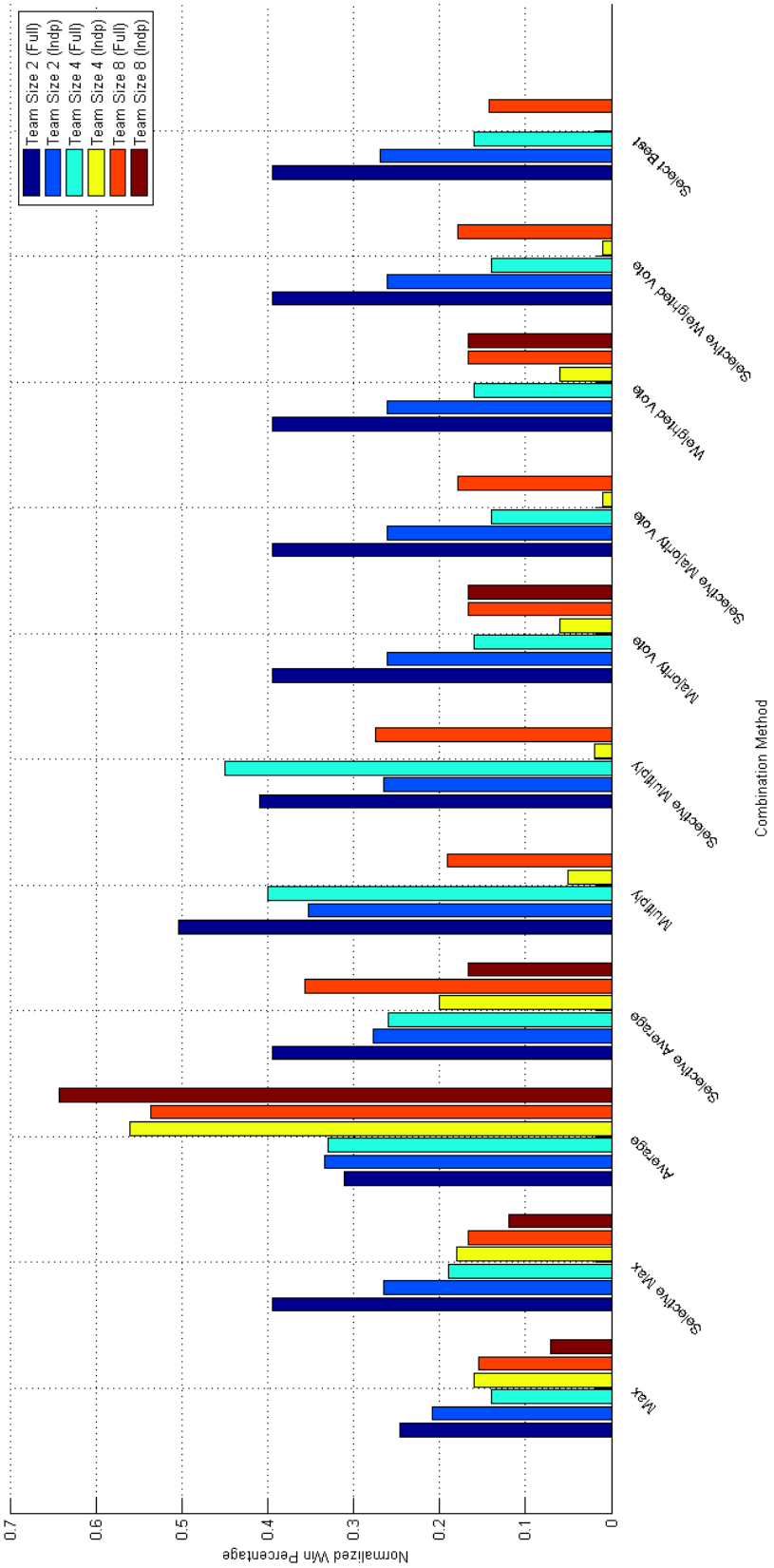

Figure 5.5: Analysis of the effect of team size on combination method win percentage for Full versus Indp learning modes.

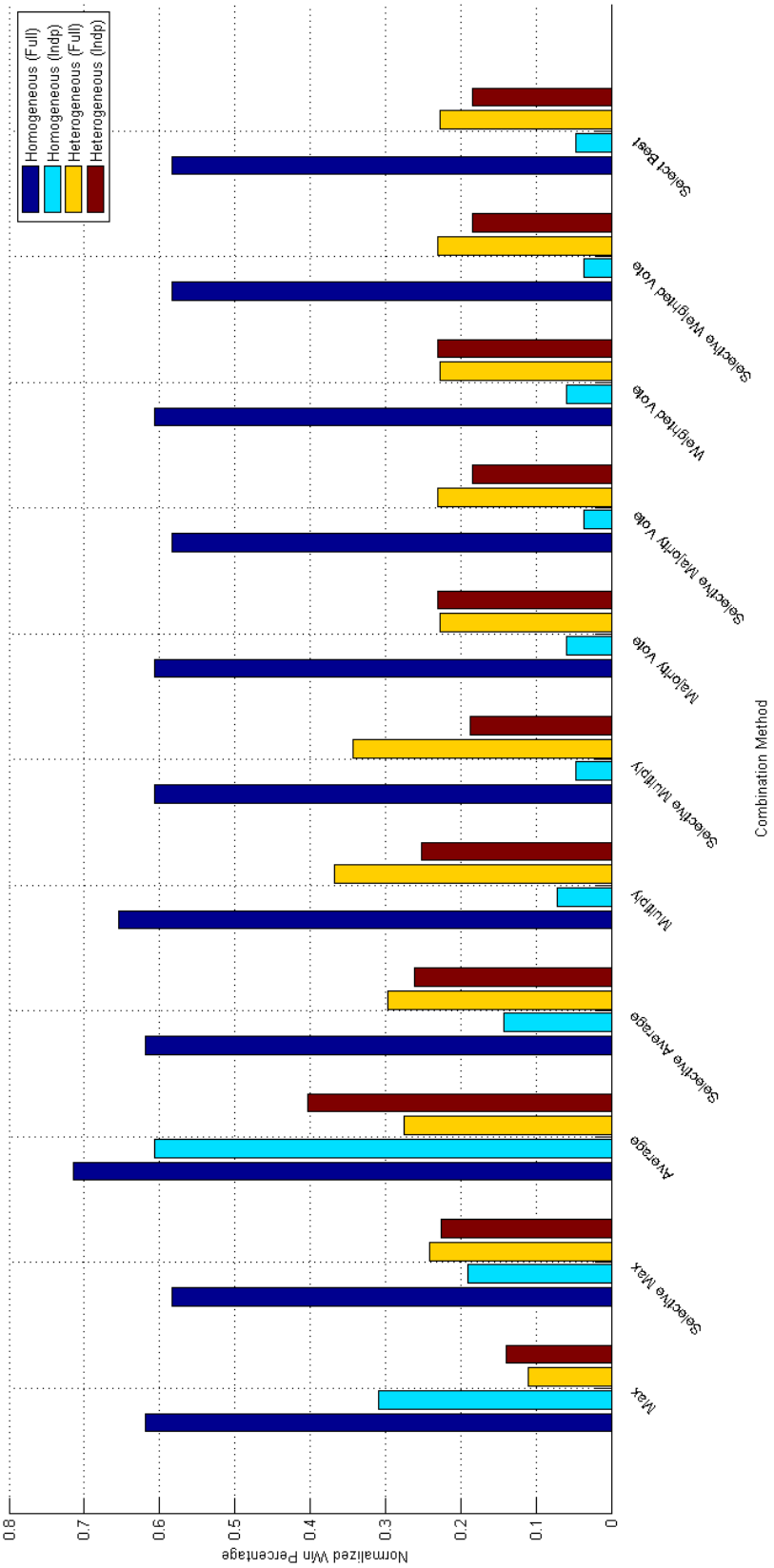

Figure 5.6: Analysis of the effect of team heterogeneity on combination method win percentage for Full versus Indp learning modes.

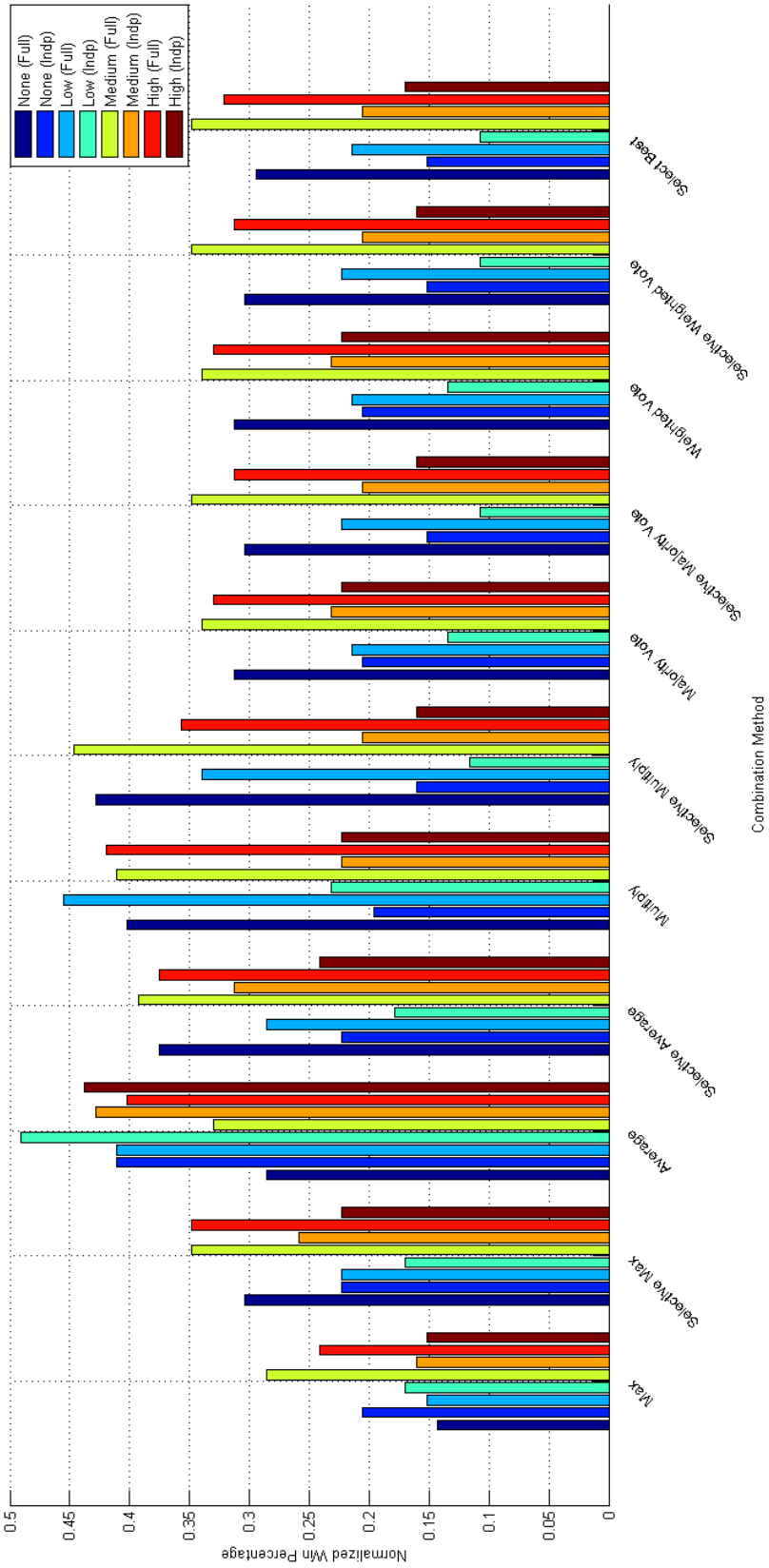

Figure 5.7: Analysis of the effect of collaboration frequency on combination method win percentage for Full versus Indp learning modes.

5.5.2 Full versus Independent Data Learning Distribution

The study of learning distribution modes was aimed at determining if each learner required the full training data set for training, or if it could be distributed among the learners in a non-overlapping fashion. More importantly, the goal was to investigate collaboration's impact on testing accuracy, given these two modes. It was envisioned that Full mode teams would perform better, and that Indp mode teams would require higher levels of collaboration to achieve high classification accuracy. It is also understood that distributing the learning load for training offers a speedup in terms of the time required for teams to model the data, as each learner will be responsible for fewer training instances. The following tables and figures show our findings.

Figure 5.8 presents the minimum, average, and maximum testing accuracy for Full and Indp mode teams by team size. The worst performing Full and Indp teams have similar accuracy over all team sizes. In both cases, teams of size four produce the highest accuracy for their lowest performing teams, as the minimum decreases when scaling up to teams of size eight. Certain pairings of learning algorithms perform very poorly in comparison to the majority of the teams. However, as team size increases, the teams perform better and appear to level off between sizes four and eight. Thus, for this application, some learning algorithms perform very poorly when paired together, but increasing the number of learners in the team can help increase team accuracy. Team composition does make a difference in team accuracy, as each learning algorithm models the data in different manners. Combining these different models in different ways can increase accuracy.

Full mode teams perform better than Indp mode teams on average, with the difference in performance growing with team size. A team of size two produced the highest team classification accuracy over all Indp mode teams, whereas teams of size four and eight produced the highest team classification accuracy for Full mode teams. This introduces a tradeoff between team size and classification accuracy between size four and eight teams. Team size inherently involves a difference in training and testing time. Some of the learning algorithms require more resources and time to perform their modeling effort. For example, decision trees do not require the memory space and time that neural networks do. The K* algorithm is the most memory-intensive algorithm utilized in our experiments. Thus, if similar accuracy can be achieved with a smaller team size and the application permits, it may not be efficient for a minimal amount of accuracy gain.

Figure 5.9 shows the training plus testing timing differences as a function of team size for homogeneous, non-collaborating Decision Tree and Neural Network teams. The reported times are the sum of the total time required for each learning algorithm to train and test, a collective model to be created, and the teams predictions to be recorded. Note that these experiments were performed on a shared Linux computer with several other processes running in an ad-hoc manner. Thus, these timing results are general and could be skewed by multiple factors, including waiting for processor time and memory. Also, as

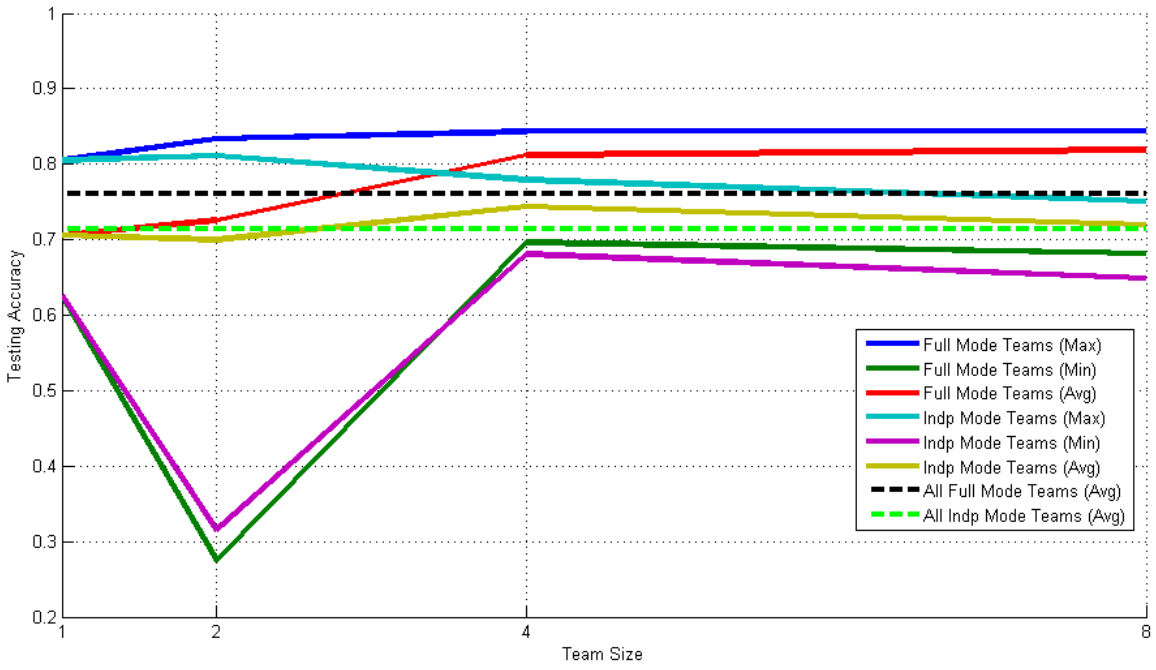

Figure 5.8: Analysis of the effect of team size on testing accuracy for Full versus Indp mode teams. The average testing accuracies over all Full and Indp teams are shown for reference.

none of these timings involve teams which collaborate, their training and testing accuracy cannot be directly correlated to the time variations.

The timing differences between Full and Indp mode teams can be observed, demonstrating the increase in efficiency (at the sacrifice for accuracy) that the distributed learning mode offers. As a larger team size results in smaller independent portions of the training set being distributed for model creation, the training plus testing time decreases as team size increases. On the other hand, for Full mode teams, training plus testing time increases with team size, as each learner in the team receives the entire training set for training. The difference between neural networks and decision trees is clear, where in the worst case decision tree teams take seconds as opposed to hours to perform the learning and classification task. Interestingly, Indp mode decision tree teams are faster than Full mode counterparts, but the time continues to increase with team size. It is not clear exactly why this is the case.

We can further investigate team compositions that are generally successful and not. Table 5.5 shows the best 15 teams for Full and Indp modes, accompanied by the associated collaboration frequency and team size. For both Full and Indp mode teams, multiple teams can produce the same testing accuracy. For example, the sixth through eighth Full mode teams only differ in the last learner out of all eight. In these cases, the last learner may not be necessary, as the other seven may have covered all of its correctly classified instances. Several Indp mode teams achieve identical accuracy, even when the collaboration frequency

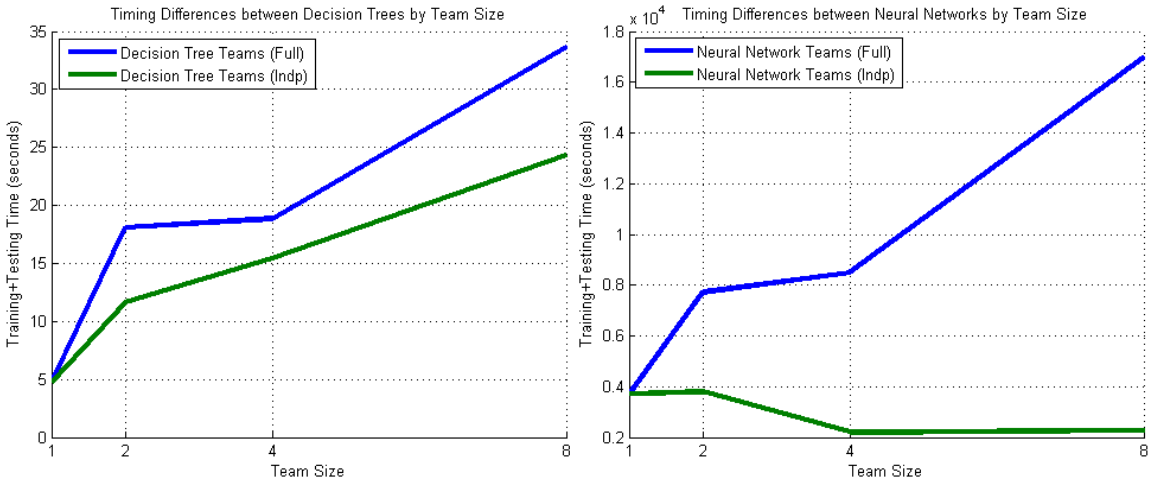

Figure 5.9: Training plus testing timing differences between homogeneous, non-collaborating decision tree and neural network teams as a function of team size.

is varied. For such team compositions, collaboration may not be desired.

Two primary trends are apparent when analyzing the best performing Full mode teams. First, highly diverse/heterogeneous teams consistently produce high team classification accuracy. Second, larger teams (dominantly size eight teams in the top 15) consistently perform better. Conversely, Indp mode teams exhibit a coupling between small team size and higher accuracy. Homogeneous teams (including two individual learners) also appear in the ranking of best 15 Indp teams, demonstrating that the use of Indp learning distribution is only beneficial up to teams of size two in this case. As teams get larger, the data set becomes more distributed and the learners are given smaller portions of the data for training. Lastly, the best Full mode team performed ∼3% better than the best Indp mode team. Due to the tournament-style experimentation process, it is also observed that the same small set of learning algorithms are highly used for both Full and Indp mode teams.

Table 5.6 presents the worst performing 15 teams out of all experiments. Small team size and more homogeneous team compositions are indicated as prominent trends for the worst performing Full and Indp mode teams. Many of the worst performing teams collaborate with medium and high frequency. This does not necessarily mean that collaboration is bad, but rather that these teams do not collaborate well (learn examples from one another during the learning process). It is shown later in this chapter that collaboration does have its advantages for specific teams and team sizes. The majority of poor-performing teams here are combinations of poorly performing individual classifiers, or homogeneous teams of the same poor individual classifier on this data set. There is a clear separation between those learning algorithms that offer high accuracy for this data set, and those that do not. Similarly, we can conclude that teams of strong learners are outperforming collective decisions of multiple weak learners. The worst Indp mode teams outperform the worst Full mode teams by 4-5%. This table also indicates that identical team compositions can

Table 5.5: Overall best performing 15 Full and Indp mode teams, based on combined testing accuracy.

Full Mode Team	Size	Accuracy	Indp Mode Team	Size	Accuracy
1ibk+1kst+1rft+1dt (high)	4	0.84301	1kst+1rft (none)	2	0.811346
2ibk+2kst+2nn+2rft (low)	8	0.84301	1kst+1rft (low)	2	0.808707
2kst+2rft+2ibk+1prt+1nn (low)	8	0.84301	1ibk (none)	1	0.804749
4ibk+2kst+2rft (none)	8	0.84301	1ibk+1kst (high)	2	0.803430
2ibk+1kst+1rft (med)	4	0.84169	2kst (none)	2	0.802111
3rft+2kst+2ibk+1dt (none)	8	0.84169	2kst (low)	2	0.802111
3rft+2kst+2ibk+1nn (none)	8	0.84169	2kst (med)	2	0.802111
3rft+2kst+2ibk+1prt (none)	8	0.84169	1ibk+1kst (none)	2	0.802111
1ibk+1kst+1nn+1rft (none)	4	0.84037	1ibk+1kst (low)	2	0.802111
1ibk+1kst+1nn+1rft (low)	4	0.84037	1ibk+1kst (med)	2	0.802111
2rft+1kst+1ibk (med)	4	0.84037	2kst (high)	2	0.800792
2ibk+2kst+2nn+2rft (none)	8	0.84037	1kst+1rft (high)	2	0.800792
3ibk+2kst+2rft+1nn (low)	8	0.84037	1kst+1rft (med)	2	0.799472
3ibk+2kst+3rft (low)	8	0.83905	1kst (none)	1	0.795515
3rft+2kst+2ibk+1prt (med)	8	0.83905	1nb+1kst (low)	2	0.791557

produce better accuracy using distributed (Indp) learning (e.g., 2dtb and 2lgr).

Figure 5.10 presents a measure of improvement of using Full mode learning over Indp mode learning as a function of team size. This measure is calculated by dividing the average testing accuracy of Full mode teams by the average testing accuracy of Indp mode teams. Thus, a value above 1.0 represents Full mode teams performing better on average. Full mode teams perform approximately 7% better than Indp mode teams on average over all the experiments. The increase in performance is coupled with team size, which grows approximately 5% as team size doubles. There are cases where Indp mode teams perform slightly better than Full mode teams, most prominently being teams of size two. For size four and eight teams, the minimum improvement of Full over Indp is nearly identical. Further, the maximum improvement also appears to grow linearly as a function of team size.

From these experiments, it can be concluded that, on average, providing all learners the entire data set during training (Full mode) outperforms distributing the data set in a non-overlapping fashion over all learners (Indp mode). As previously noted, Indp mode teams do have the advantage of completing the training process much faster than Full mode teams, as the learners train on smaller and smaller portions of the data set as team size increases. This negatively affects peak performance for Indp mode teams.

5.5.3 Team Diversity: Homogeneous versus Heterogeneous Teams

Team diversity, or the composition of a team in terms of multiple heterogeneous learning algorithms, is central to a study of team learning dynamics. As different learning algorithms model the data and error differently, combining them can be beneficial for increasing

Table 5.6: Overall worst performing 15 Full and Indp mode teams, based on combined testing accuracy.

Full Mode Team	Size	Accuracy	Indp Mode Team	Size	Accuracy
2dtb (high)	2	0.27441	2dtb (high)	2	0.315303
2dtb (med)	2	0.29420	2dtb (med)	2	0.368074
2lgr (med)	2	0.36544	2lgr (med)	2	0.374670
1lgr+1dtb (high)	2	0.39314	2lgr (high)	2	0.437995
2lgr (high)	2	0.41161	1lgr+1rbf (high)	2	0.474934
2nb (med)	2	0.42876	1nb+1dtb (high)	2	0.481530
2lgr (low)	2	0.43931	1nb+1lgr (high)	2	0.482850
2dtb (low)	2	0.46306	1lgr+1dtb (high)	2	0.489446
1nb+1lgr (med)	2	0.46570	2nb (med)	2	0.494723
2nb (high)	2	0.46834	2nb (high)	2	0.507916
1lgr+1rbf (med)	2	0.47362	1lgr+1dtb (med)	2	0.510554
1nb+1dtb (high)	2	0.48285	2rbf (high)	2	0.510554
1lgr+1dtb (med)	2	0.48681	1nb+1dtb (med)	2	0.521108
1nb+1dtb (med)	2	0.50000	1nb+1lgr (med)	2	0.531662
1lgr+1rbf (high)	2	0.51715	2rbf (med)	2	0.536939

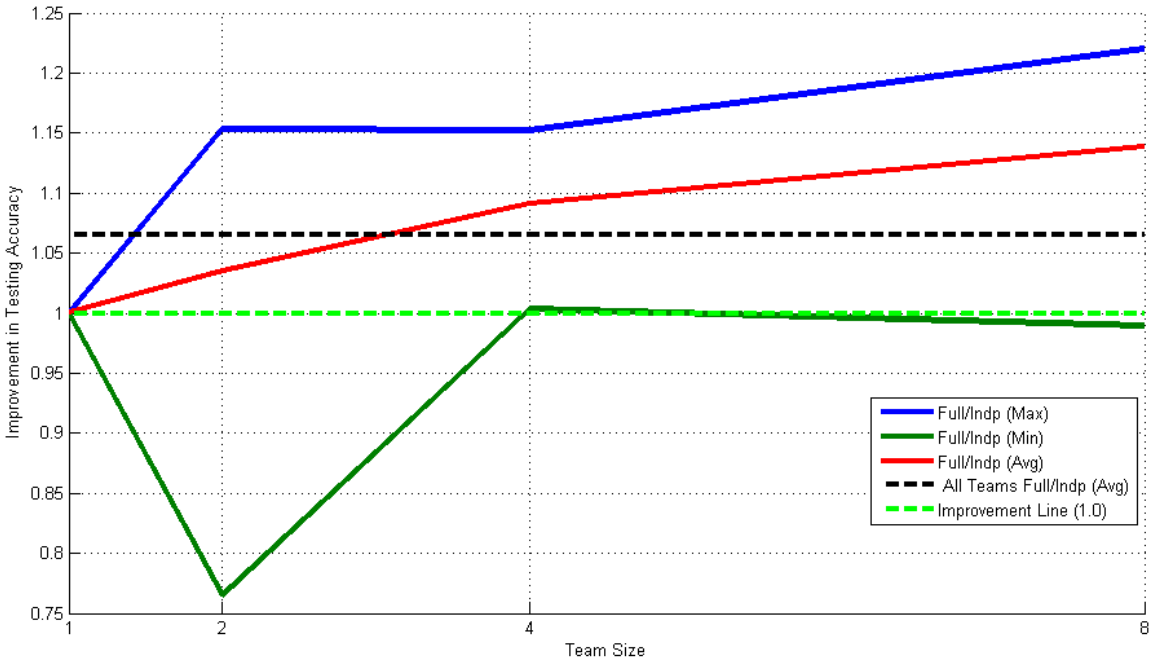

Figure 5.10: Investigation of the improvement/advantage of testing accuracy for Full mode teams versus Indp mode teams. Values above 1.0 represent Full mode teams performing better than Indp mode teams. The average testing accuracies over all Full and Indp teams are shown for reference.

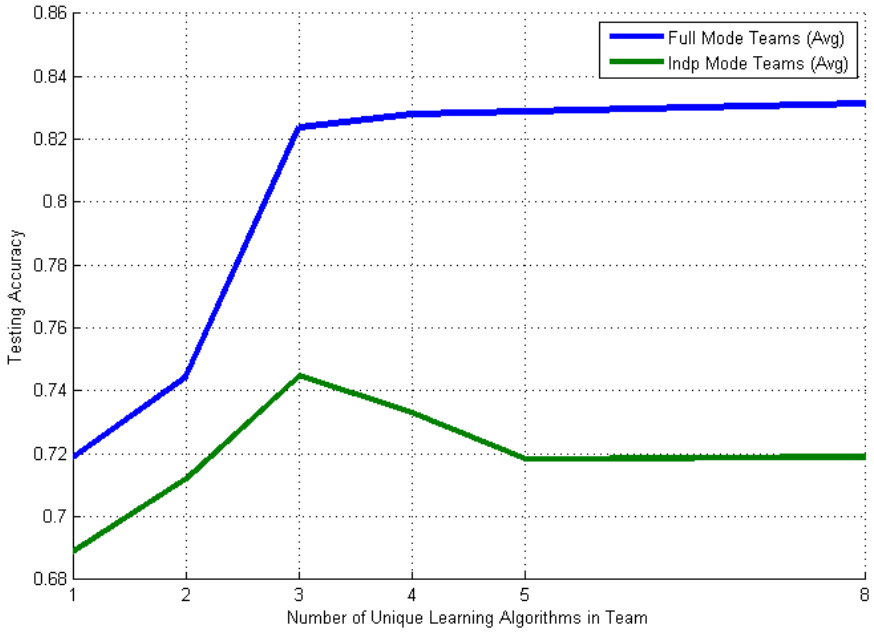

Figure 5.11: Average testing accuracy for Full and Indp mode teams as a function of team diversity.

classification accuracy. Teams composed of the same learning algorithm may not experience these advantages, especially when collaboration is involved. Here, we specifically focus on comparing results from homogeneous and heterogeneous teams. When discussing a team's diversity level, it represents the number of unique learning algorithms in that team.

One method for analyzing team diversity is to observe its effect on testing accuracy as a function of the diversity level, as shown in Figure 5.11. No team existed which contained six or seven unique learning algorithms, and the testing accuracies are assumed linear between diversity levels five and eight. Full mode teams perform better for all diversity levels. Full mode teams also see a steady increase in testing accuracy coupled with an increase in diversity. This is a promising result, as it supports that the more heterogeneous a team is, the better its accuracy. Indp mode teams experience a similar increase in testing accuracy up to diversity level of three, but experience a decrease and leveling off of testing accuracy as diversity increases. This could be related to how a team should be composed of some supporting classifiers and some complementary classifiers to achieve its greatest potential. These results suggest that a diversity level of three (i.e., a team consisting of three heterogeneous learning algorithms) is desirable in this application, as it maximizes testing accuracy for both Full and Indp teams.

Table 5.7 offers a different view of team diversity by showing the number of teams of each diversity level. Diversity level counts are shown for all teams, the best 50 teams, and the worst 50 teams (in terms of testing accuracy). The tournament-style experimentation process produced many teams of diversity levels 1, 2, and 4. By viewing the best 50 teams

Table 5.7: Number of teams of each diversity level, which is a count of the number of unique learning algorithms in a team, over all, best 50, and worst 50 teams. No team contained six or seven unique learning algorithms.

	All Teams		Best 50 Teams		Worst 50 Teams	
Diversity	**Full**	**Indp**	**Full**	**Indp**	**Full**	**Indp**
1	95	95	0	16	21	18
2	240	240	7	31	29	32
3	40	40	14	3	0	0
4	68	68	24	0	0	0
5	12	12	4	0	0	0
6	NA	NA	NA	NA	NA	NA
7	NA	NA	NA	NA	NA	NA
8	4	4	1	0	0	0

and their diversity levels, we can draw a conclusion about which diversity level achieves highest performance overall. Here, diversity levels of 3 and 4 are best for Full mode teams, and a diversity level of 2 for Indp teams. This supports earlier results, where the majority of the best performing Full mode teams contained four or eight total learners, and the best performing Indp mode teams contained mostly two learners and were mostly heterogeneous in composition. Diversity levels of 1 and 2 proved to be of lowest performance for Full mode teams. However, we again see diversity levels of 1 and 2 are involved in the worst performing Indp mode teams.

Figures 5.12 and 5.13 show results comparing homogeneous and heterogeneous team performance as a function of team size for Full and Indp mode teams, respectively. Heterogeneous teams perform approximately 5% (Full mode) and 3% (Indp mode) better on average. Homogeneous teams produce their highest average testing accuracy with size four, and their lowest testing accuracy with size two. Homogeneous teams also experience a decrease in accuracy when scaling from size four to eight. These results again support that some teams of size two perform very poorly while others perform quite well. Homogeneous and heterogeneous Indp mode teams both produce a similar maximum accuracy over all team sizes; however, Indp mode teams steadily decrease in accuracy as team size increases from four to eight. The effect of heterogeneous team composition is therefore more pronounced for Full mode teams, but beneficial for both learning modes.

Tables 5.8 and 5.9 list the best and worst performing homogeneous and heterogeneous 15 teams for both Full and Indp learning modes, respectively. Out of the best 15 Indp mode homogeneous and heterogeneous teams, only two perform better than the best individual learning algorithm on this data set (IBK), both of which are heterogeneous. The majority of the Indp mode homogeneous teams are of small size (including four individual learners), while all of the Indp mode heterogeneous teams are of size two. Full mode teams are consistently of large size for both homogeneous and heterogeneous compositions. They also are all diversity levels 3 and 4, with one team exhibiting a diversity level of 5. There are several examples where an increase in collaboration frequency produced higher accuracy,

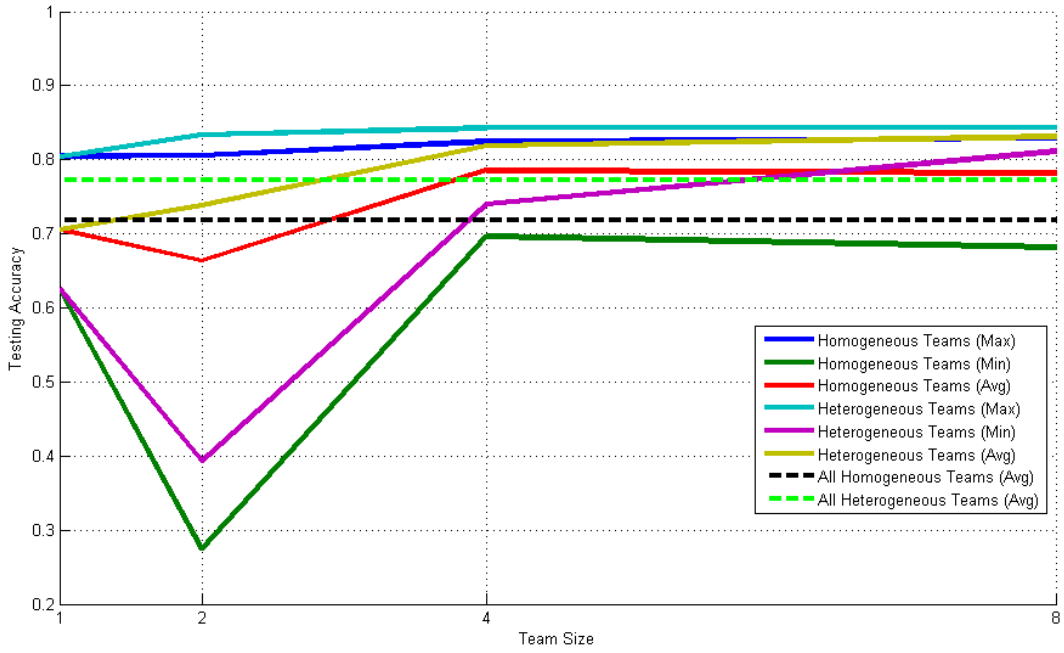

Figure 5.12: Homogeneous versus heterogeneous team testing performance comparison by team size for Full mode teams. The average testing accuracies over all homogeneous and heterogeneous Full mode teams are shown for reference.

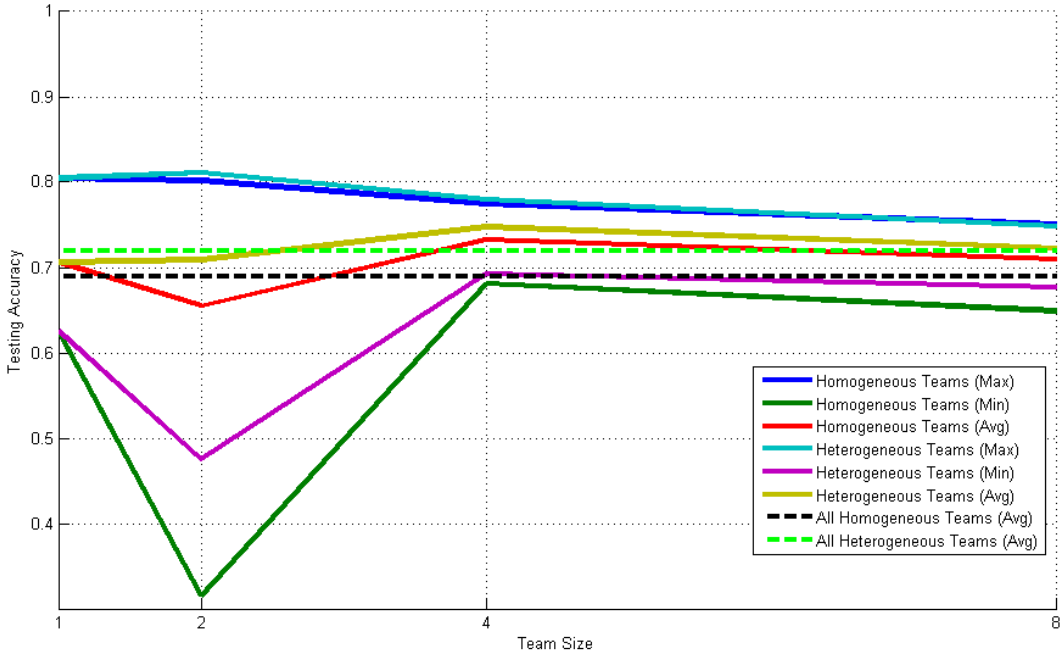

Figure 5.13: Homogeneous versus heterogeneous team testing performance comparison by team size for Indp mode teams. The average testing accuracies over all homogeneous and heterogeneous Indp mode teams are shown for reference.

but a decrease in accuracy when collaborating with high frequency. This again supports that collaborating can be beneficial to an extent, after which it degrades performance.

An advantage is shown in favor of heterogeneous teams, which perform between 12-16% better in the pool of lowest accuracy teams. Size two teams again dominate the Indp mode homogeneous rankings, demonstrating that some combinations of learning algorithms coupled with higher collaboration frequencies can actually degrade the performance. Low diversity levels and team sizes are foremost out of those homogeneous and heterogeneous teams that perform the worst. These results parallel those which were discussed earlier for team diversity. Specific learning algorithms consistently offer low accuracy, even when paired with other learners, such as Naive Bayes and Logistic Regression. Team heterogeneity again prevails over homogeneous team compositions.

5.5.4 Team Size: Single versus Multiple Learners

A study of team learning inherently involves the question, "How large should the team be?" Specifically, it is desirable to study how scaling the team size affects different team dynamics (successful team composition, testing accuracy, collaboration frequency, etc.). The objective is to study what can be gained by using multiple collaborating learning algorithms, as opposed to a single learning algorithm for the entire data set. The following tables shed more light on these aspects, and are intended to supplement the results presented as part of other studies in this chapter.

Tables 5.10 and 5.11 list the five most successful and least successful teams as a function of team size. These tables show the direct result of using a tournament-style experimentation setup, as the best individual learners are highly involved in all successful teams of larger sizes. There exists a coupling between diversity level and team size for the most successful teams. Specifically, the Full mode teams that perform best at each team size are nearly equally diverse (team size divided by diversity level). This suggests that the best way to design a team is to compose it of the best performing individual algorithms and construct it so that the team is equally diverse (team size divided by diversity level is between 1 and 2). As we have observed certain advantages from homogeneous and heterogeneous teams, constructing a team in this manner combines aspects of both. For large teams, this would translate to having a heterogeneous mixture of homogeneous teams. Additionally, benefits of collaboration are observed, but largely at Low and Medium frequencies. Teams of size four and eight, which happen to contain the best overall teams, generally exhibit all of these aspects: heterogeneous mixture of homogeneous learning teams, with Low to Medium collaboration. This is not the case for Indp mode teams, in that they are all homogeneous in nature.

Conversely, the least successful teams over all team sizes generally do not follow this equation for success. They exhibit low diversity equality (high value for team size divided by diversity level) and are almost entirely composed of homogeneous learning algorithms.

Table 5.8: Overall best performing 15 homogeneous and 15 heterogeneous teams, based on combined testing accuracy and comparing Full and Indp modes.

Full Mode Teams				Independent Mode Teams			
Homogeneous	**Accuracy**	**Heterogeneous**	**Accuracy**	**Homogeneous**	**Accuracy**	**Heterogeneous**	**Accuracy**
8nn (high)	0.828496	1ibk+1kst+1rft+1dt (high)	0.843008	1ibk (none)	0.804749	1kst+1rft (none)	0.811346
4nn (low)	0.824538	2ibk+2kst+2nn+2rft (low)	0.843008	2kst (none)	0.802111	1kst+1rft (low)	0.808707
8nn (low)	0.819261	2kst+2rft+2ibk+1prt+1nn (low)	0.843008	2kst (low)	0.802111	1ibk+1kst (high)	0.80343
8nn (med)	0.813984	4ibk+2kst+2rft (none)	0.843008	2kst (med)	0.802111	1ibk+1kst (none)	0.802111
8rft (high)	0.813984	2ibk+1kst+1rft (med)	0.841689	2kst (high)	0.800792	1ibk+1kst (low)	0.802111
4nn (none)	0.811346	3rft+2kst+2ibk+1dt (none)	0.841689	1kst (none)	0.795515	1ibk+1kst (med)	0.802111
4rft (med)	0.810026	3rft+2kst+2ibk+1nn (none)	0.841689	1nn (none)	0.790237	1kst+1rft (high)	0.800792
4rft (high)	0.810026	3rft+2kst+2ibk+1prt (none)	0.841689	1rft (none)	0.782322	1kst+1rft (med)	0.799472
4rft (low)	0.808707	1ibk+1kst+1nn+1rft (none)	0.840369	2ibk (high)	0.781003	1nb+1kst (low)	0.791557
8nn (none)	0.808707	1ibk+1kst+1nn+1rft (low)	0.840369	2ibk (none)	0.779683	1kst+1dtb (med)	0.788918
8rft (med)	0.808707	2rft+1kst+1ibk (med)	0.840369	2ibk (low)	0.779683	1kst+1dtb (high)	0.787599
8rft (low)	0.807388	2ibk+2kst+2nn+2rft (none)	0.840369	2ibk (med)	0.779683	1rbf+1kst (low)	0.78628
2ibk (low)	0.806069	3ibk+2kst+2rft+1nn (low)	0.840369	4kst (none)	0.774406	1ibk+1rft (none)	0.783641
4ibk (low)	0.806069	3ibk+2kst+3rft (low)	0.83905	4kst (low)	0.774406	1rbf+1kst (med)	0.783641
2ibk (none)	0.804749	3rft+2kst+2ibk+1prt (med)	0.83905	4kst (med)	0.774406	1lgr+1kst (none)	0.782322

Table 5.9: Overall worst performing 15 homogeneous and 15 heterogeneous teams, based on combined testing accuracy and comparing Full and Indp modes.

Full Mode Teams				Independent Mode Teams			
Homogeneous	**Accuracy**	**Heterogeneous**	**Accuracy**	**Homogeneous**	**Accuracy**	**Heterogeneous**	**Accuracy**
2dtb (high)	0.274406	1lgr+1dtb (high)	0.39314	2dtb (high)	0.315303	1lgr+1rbf (high)	0.474934
2dtb (med)	0.294195	1nb+1lgr (med)	0.465699	2dtb (med)	0.368074	1nb+1dtb (high)	0.48153
2lgr (med)	0.365435	1lgr+1rbf (med)	0.473615	2lgr (med)	0.37467	1nb+1lgr (high)	0.48285
2lgr (high)	0.411609	1nb+1dtb (high)	0.48285	2lgr (high)	0.437995	1lgr+1dtb (high)	0.489446
2nb (med)	0.42876	1lgr+1dtb (med)	0.486807	2nb (med)	0.494723	1lgr+1dtb (med)	0.510554
2lgr (low)	0.439314	1nb+1dtb (med)	0.5	2nb (high)	0.507916	1nb+1dtb (med)	0.521108
2dtb (low)	0.463061	1lgr+1rbf (high)	0.51715	2rbf (high)	0.510554	1nb+1lgr (med)	0.531662
2nb (high)	0.468338	1nb+1lgr (high)	0.522427	2rbf (med)	0.536939	1nb+1rbf (high)	0.546174
2rbf (med)	0.558047	1rbf+1dtb (high)	0.523747	2lgr (low)	0.558047	1lgr+1rbf (med)	0.560686
2rbf (high)	0.573879	1nb+1rbf (med)	0.532982	2dtb (none)	0.58971	1dtb+1prt (low)	0.585752
2nb (low)	0.58971	1rbf+1dtb (med)	0.587071	2dtb (low)	0.605541	1rbf+1dtb (high)	0.591029
1dtb (none)	0.62533	1nb+1rbf (high)	0.594987	2nb (low)	0.612137	1nb+1rbf (med)	0.598945
2dtb (none)	0.631926	1nb+1lgr (low)	0.630607	1dtb (none)	0.62533	1rbf+1dtb (med)	0.612137
1nb (none)	0.637203	1nb+1dtb (low)	0.633245	2prt (none)	0.629288	1lgr+1dtb (low)	0.613456
2nb (none)	0.637203	1lgr+1rbf (low)	0.635884	2nb (none)	0.630607	1dtb+1jrp (none)	0.616095

Table 5.10: Most successful five teams, based on combined testing accuracy, for Full and Indp learning distribution modes over each team size.

Full Mode Teams		Independent Mode Teams	
Size 1 Team	**Accuracy**	**Size 1 Team**	**Accuracy**
1ibk (none)	0.804749	1ibk (none)	0.804749
1kst (none)	0.795515	1kst (none)	0.795515
1nn (none)	0.790237	1nn (none)	0.790237
1rft (none)	0.782322	1rft (none)	0.782322
1dt (none)	0.69657	1dt (none)	0.69657
Size 2 Team	**Accuracy**	**Size 2 Team**	**Accuracy**
1ibk+1kst (none)	0.8337731	1kst+1rft (none)	0.8113456
1ibk+1kst (low)	0.8337731	1kst+1rft (low)	0.8087071
1ibk+1kst (med)	0.8337731	1ibk+1kst (high)	0.8034301
1ibk+1kst (high)	0.8337731	2kst (none)	0.8021108
1kst+1rft (med)	0.8311346	2kst (low)	0.8021108
Size 4 Team	**Accuracy**	**Size 4 Team**	**Accuracy**
1ibk+1kst+1rft+1dt (high)	0.8430079	2kst+1ibk+1rft (high)	0.7796834
2ibk+1kst+1rft (med)	0.8416887	4kst (none)	0.7744063
1ibk+1kst+1nn+1rft (none)	0.8403694	4kst (low)	0.7744063
1ibk+1kst+1nn+1rft (low)	0.8403694	4kst (med)	0.7744063
2rft+1kst+1ibk (med)	0.8403694	4kst (high)	0.7744063
Size 8 Team	**Accuracy**	**Size 8 Team**	**Accuracy**
2ibk+2kst+2nn+2rft (low)	0.8430079	8kst (none)	0.7506596
2kst+2rft+2ibk+1prt+1nn (low)	0.8430079	8kst (low)	0.7506596
4ibk+2kst+2rft (none)	0.8430079	8kst (med)	0.7506596
3rft+2kst+2ibk+1dt (none)	0.8416887	8kst (high)	0.7506596
3rft+2kst+2ibk+1nn (none)	0.8416887	3ibk+2kst+2rft+1dt (low)	0.7480211

These tables further support the notion that adding individual learners to a team (increasing the team's diversity level) can help fill niches of the data and positively contribute to team success. For example, combining IBK and K* offers a significant increase in team testing accuracy. Focusing on Indp mode teams, adding two K* learners to a team of two Logistic Regression learners nearly doubles the testing accuracy.

Full mode teams increase in accuracy and diversity with team size, while Indp mode teams remain predominantly homogeneous and low diversity across team sizes. Indp mode teams experience their peak in accuracy with teams of size two. Collaboration appears to offer an advantage for teams of size two and four when using Full learning mode, and for teams of size four and eight when using Indp learning mode. Teams of size four appear to be the best choice, in terms of testing accuracy and time requirements for this application. Teams of size 5, 6, and 7 were not tested as part of this study.

5.5.5 Self-Learning versus Collaboration

Collaboration allows a learner in a team to distribute difficult training instances to all other learners, increasing the chance that one or more of them will properly learn it and be able

Table 5.11: Least successful five teams, based on combined testing accuracy, for Full and Indp learning distribution modes over each team size.

Full Mode Teams		Independent Mode Teams	
Size 1 Team	**Accuracy**	**Size 1 Team**	**Accuracy**
1dtb (none)	0.62533	1dtb (none)	0.62533
1nb (none)	0.637203	1nb (none)	0.637203
1jrp (none)	0.64248	1jrp (none)	0.64248
1lgr (none)	0.64248	1lgr (none)	0.64248
1prt (none)	0.666227	1prt (none)	0.666227
Size 2 Team	**Accuracy**	**Size 2 Team**	**Accuracy**
2dtb (high)	0.2744063	2dtb (high)	0.3153034
2dtb (med)	0.2941953	2dtb (med)	0.3680739
2lgr (med)	0.3654354	2lgr (med)	0.3746702
1lgr+1dtb (high)	0.3931398	2lgr (high)	0.4379947
2lgr (high)	0.4116095	1lgr+1rbf (high)	0.474934
Size 4 Team	**Accuracy**	**Size 4 Team**	**Accuracy**
4dt (none)	0.6965699	4dt (none)	0.6807388
4dt (high)	0.7044855	4dt (low)	0.6860158
4dt (med)	0.7189974	2lgr+2kst (high)	0.6926121
4dt (low)	0.7229551	4dt (high)	0.6926121
2lgr+2kst (med)	0.7401055	2kst+1ibk+1lgr (high)	0.7031662
Size 8 Team	**Accuracy**	**Size 8 Team**	**Accuracy**
8dt (low)	0.682058	8dt (low)	0.6490765
8dt (none)	0.6965699	2ibk+2kst+2rft+1dt+1nn (med)	0.676781
8dt (high)	0.7044855	8nn (med)	0.6807388
8dt (med)	0.707124	8dt (high)	0.6807388
8kst (high)	0.7427441	2ibk+2kst+2rft+1dt+1nn (high)	0.682058

to correctly classify that instance as a team once learning has finished. In this respect, it is expected that highly heterogeneous teams will benefit more from collaboration, as each learning algorithm models the data in different ways. Thus, if one learning algorithm cannot incorporate an instance into its classification model, one or more others would. Homogeneous teams would likely not be able to take advantage of this, contributing to the estimate that collaboration could be detrimental to homogeneous teams. Additionally, collaboration can potentially contribute to overfitting, especially for large teams which collaborate with very high frequency on a small data set. Collaboration was intended to be used on large data sets with moderate team sizes (those studied in this dissertation). It was expected that Indp mode teams would require additional collaboration, to gain exposure to more of the training set (albeit the most difficult instances found by other learners). Based on results from Chapter 4, it was expected that some collaboration would be beneficial, but high frequency collaboration would become detrimental to team success.

Figures 5.14 and 5.15 present collaboration results, comparing homogeneous and heterogeneous collaborating and non-collaborating teams by team size, for Full and Indp mode teams, respectively. On average, both collaborating and non-collaborating Full mode teams increase in accuracy with team size. Heterogeneous non-collaborating Full mode teams perform the best over all team sizes, and heterogeneous collaborative teams perform better than all homogeneous teams. Homogeneous collaborative teams outperform non-collaborative homogeneous teams only for teams of size four. Also, the average testing accuracy over all collaborative Full mode teams becomes slightly better than the average over all non-collaborative Full mode teams for teams of size eight. The advantage of Full mode collaboration over self-learning becomes minimal with teams larger than size four, as only 1% is gained in accuracy when scaling to eight learners.

On the other hand, for Indp mode learning, collaborating heterogeneous teams perform best for size four teams. Non-collaborating heterogeneous teams are best for size two and eight teams. This was expected for Indp mode teams, given their non-overlapping training portions. Surprisingly, both collaborating and non-collaborating teams experience a decrease in accuracy when team size is scaled from four to eight. We therefore conclude that teams of size four with some collaboration is best for Indp mode teams. The frequency of collaboration is investigated next.

Figure 5.16 presents further collaboration versus self-learning results as a function of collaboration frequency and team size. This allows the study on how collaboration frequency affects different team sizes. As seen before, the only increase in accuracy as a result of collaboration is size four teams employing a Medium collaboration frequency. All other team sizes experience a decrease in accuracy with an increase in collaboration frequency. Some team sizes are not affected as much as others. For example, size four and eight teams (both Full and Indp modes) are not affected as much as size two teams. Indp mode teams appear to be least affected in a negative manner by collaboration. Size

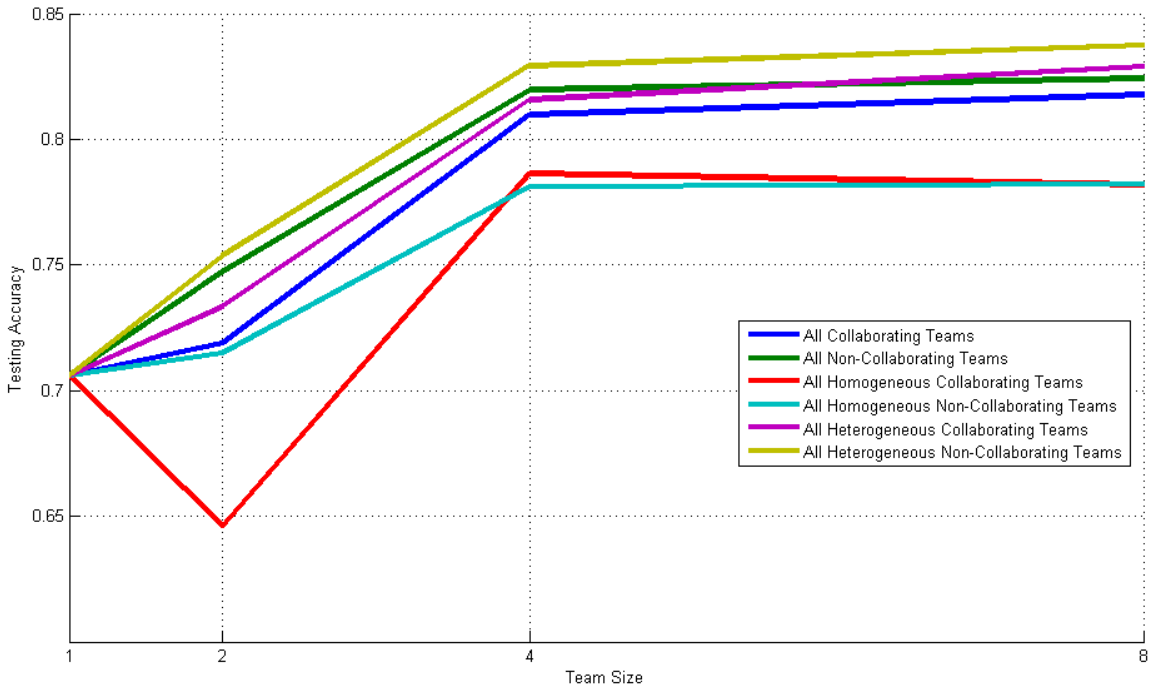

Figure 5.14: Collaboration versus self-learning average testing accuracy by team size for Full mode teams.

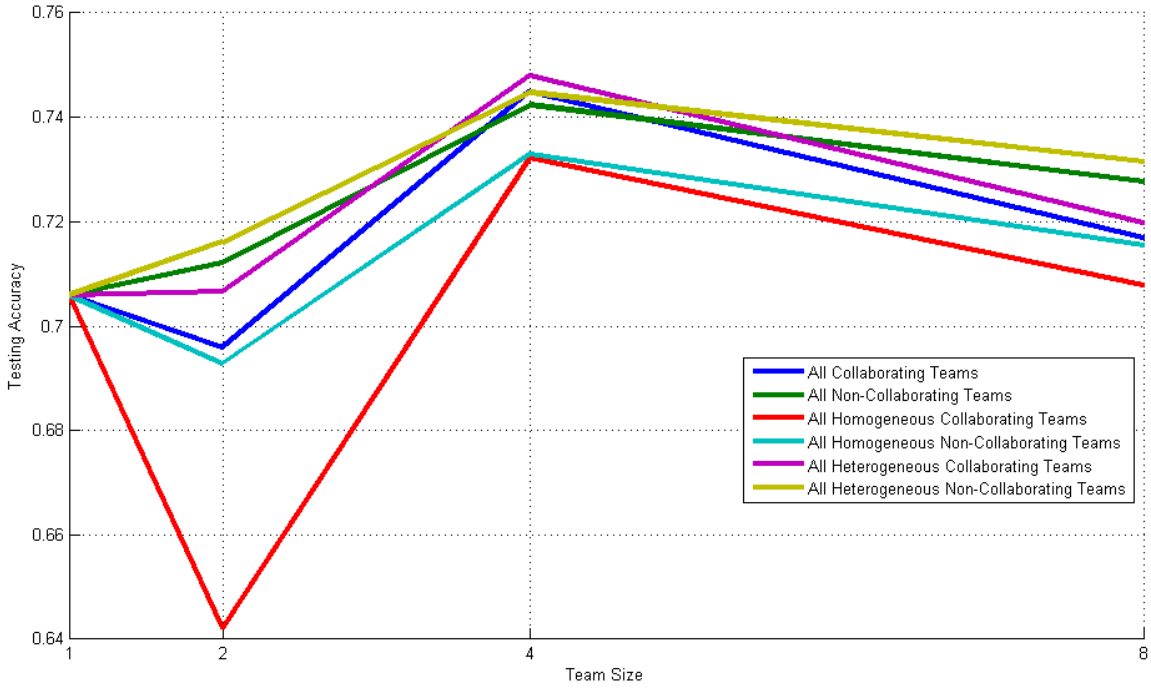

Figure 5.15: Collaboration versus self-learning average testing accuracy by team size for Indp mode teams.

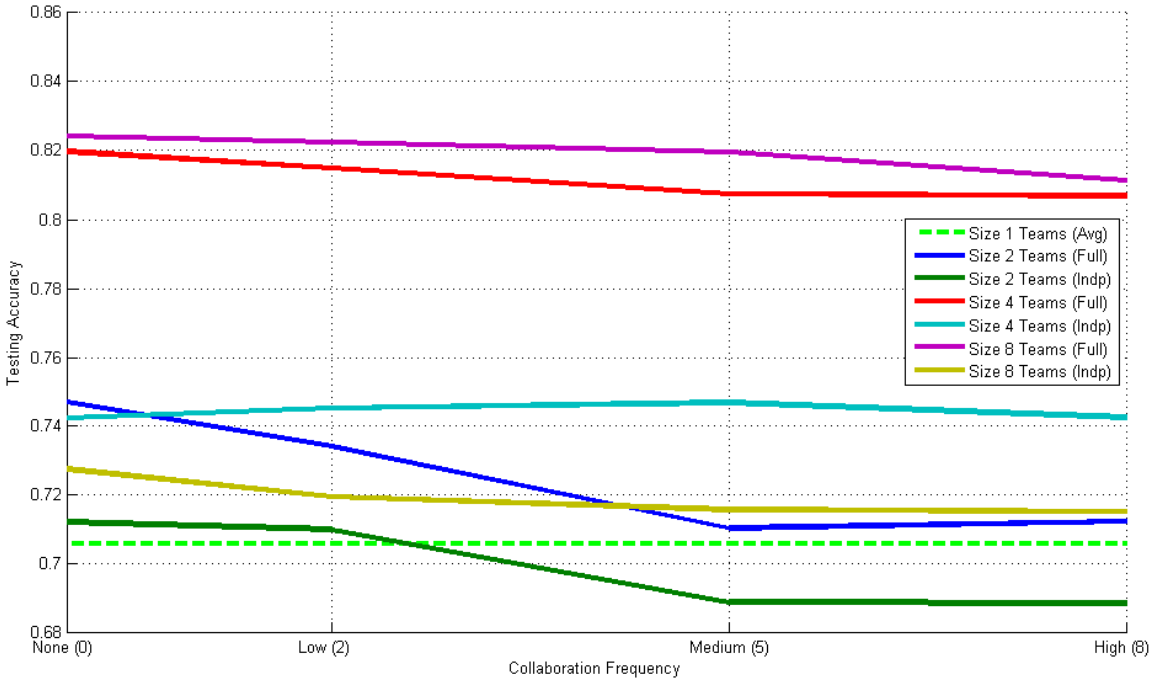

Figure 5.16: Collaboration versus self-learning average testing accuracy by collaboration frequency and team size.

eight Full mode teams are also not as greatly affected from collaborating None to Medium frequency. Based on these results, it is observed that collaboration has little benefit, only positively manifesting itself for heterogeneous Indp mode teams of size four.

Figures 5.17 and 5.18 illustrate how team heterogeneity affects average testing accuracy as functions of team size and collaboration frequency for Full and Indp mode teams, respectively. There are only two cases where collaboration causes an increase in testing accuracy for Full mode experiments: size eight homogeneous teams which collaborate with Medium frequency, and size four homogeneous teams which collaborate with Low frequency. This suggests that, for Full mode teams, collaboration is more effective for larger teams which are homogeneous in composition. For other Full mode team sizes, regardless of team composition, collaboration causes a reduction in testing accuracy.

For Indp mode experiments, there also exist two cases where collaboration causes an increase in testing accuracy: size four homogeneous teams which collaborate with Medium frequency, and size four heterogeneous teams which collaborate with Low or Medium frequency. This suggests that, for Indp mode teams, collaboration is more effective for size four teams, regardless of composition. All other team sizes and compositions result in a decrease in accuracy with increased collaboration frequency. These results again support that collaborating can be beneficial, but with Low to Medium frequency.

Another important aspect of collaboration to investigate is the improvement, if any, that it provides over not collaborating. Figure 5.19 presents a measure of improvement of

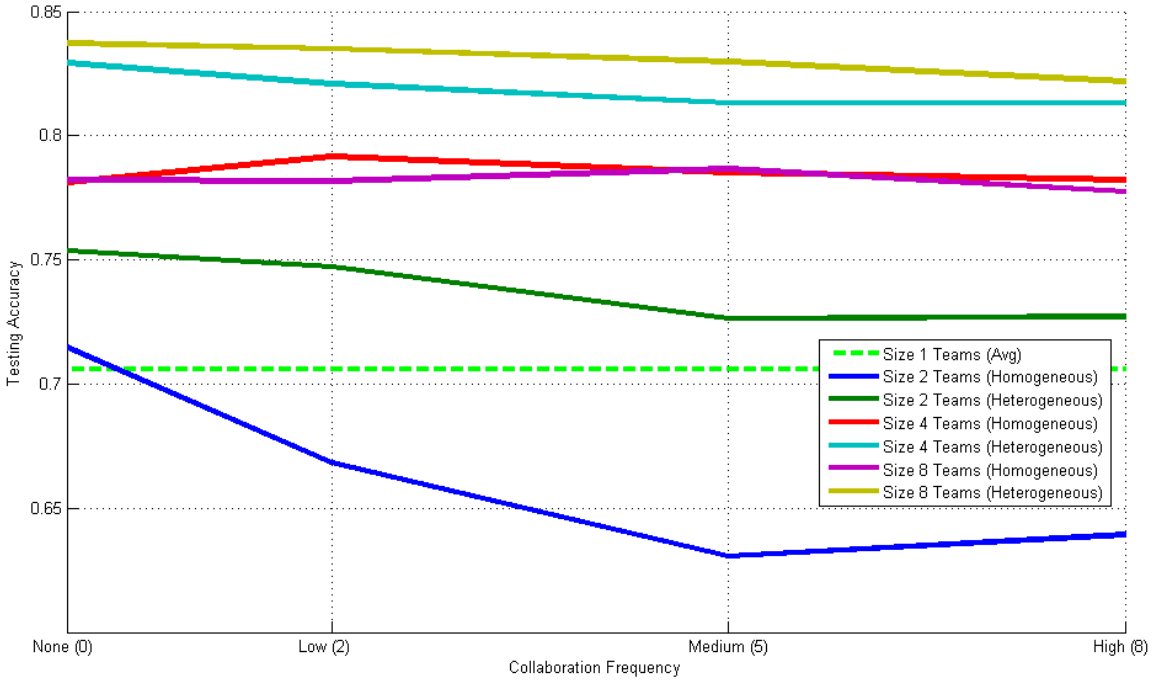

Figure 5.17: Collaboration versus self-learning average testing accuracy by collaboration frequency for Full mode homogeneous and heterogeneous teams.

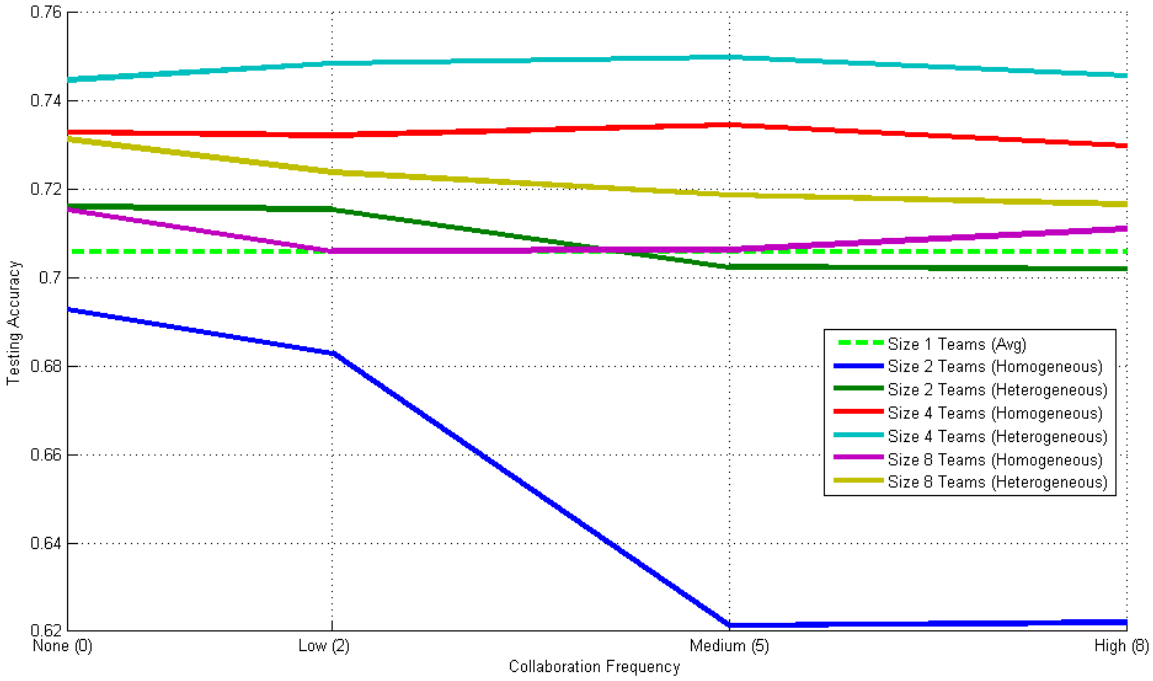

Figure 5.18: Collaboration versus self-learning average testing accuracy by collaboration frequency for Indp mode homogeneous and heterogeneous teams.

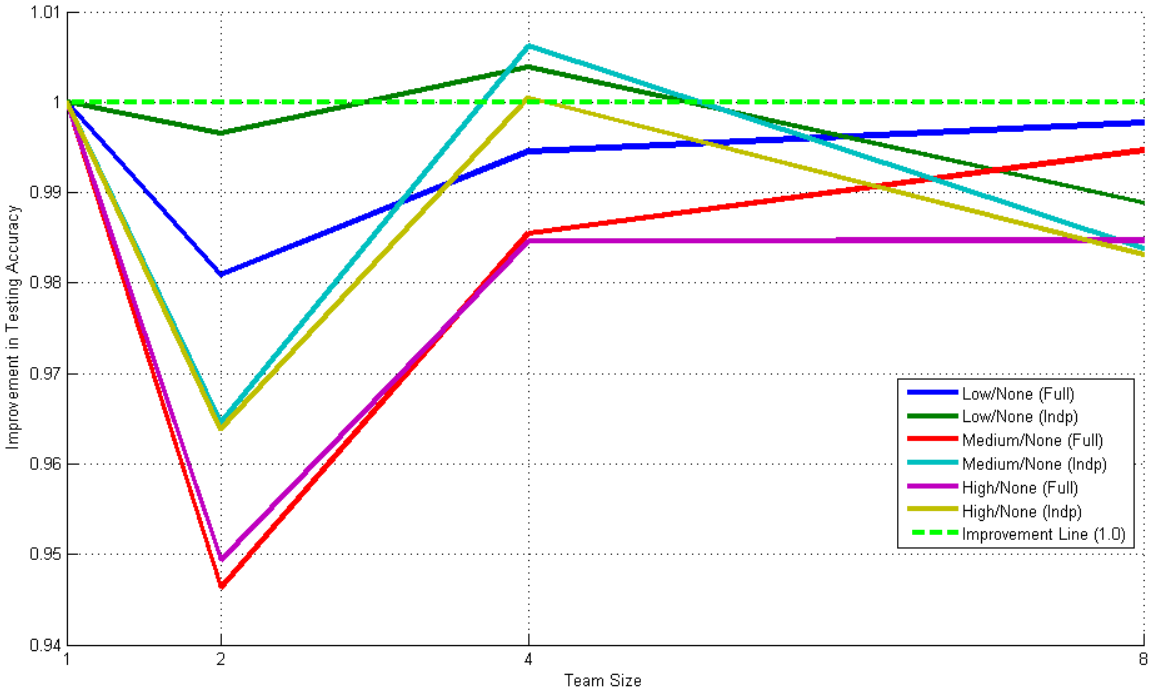

Figure 5.19: Effect/improvement of collaboration on average testing accuracy versus non-collaborating teams by team size.

collaboration over not collaborating as a function of collaboration frequency over varying team sizes for Full and Indp mode teams. This represents another way to look at how collaboration is affecting team accuracy, and how collaboration frequency is tied to team size. Improvement is calculated by dividing the average testing accuracy of a collaborative team over the average testing accuracy of a non-collaborative team of the same size. A value above 1.0 signifies that collaborating with that frequency improves accuracy for that team size. Therefore, each collaboration frequency's effect on team accuracy can be analyzed for various team sizes.

Very small improvements are observed for size four Indp mode teams, in which Medium/None provides the largest improvement, followed by Low/None, and High/None. A similar trend is observed for Full mode homogeneous teams. Heterogeneous Indp mode teams also exhibit a similar trend, but with Low/None providing the largest improvement, followed by Medium/None, and High/None. Collaborating does not appear to offer an improvement for other team configurations.

Tables 5.12 and 5.13 list the best and worst 10 collaborative and non-collaborative teams, respectively. These tables illustrate what team compositions perform best for collaborative teams and non-collaborative teams for Full and Indp mode teams. Of the most successful teams, both collaborative and non-collaborative Full mode teams are of larger team sizes and more equally diverse. Conversely, both collaborative and non-collaborative Indp mode teams are of smaller team sizes. For Full mode teams, overall performance is

Table 5.12: 10 best performing collaborating and non-collaborating teams, based on average testing accuracy, for Full and Indp modes.

Full Mode Teams		Independent Mode Teams	
No Collaboration	**Accuracy**	**No Collaboration**	**Accuracy**
4ibk+2kst+2rft (none)	0.843008	1kst+1rft (none)	0.811346
3rft+2kst+2ibk+1dt (none)	0.841689	1ibk (none)	0.804749
3rft+2kst+2ibk+1nn (none)	0.841689	2kst (none)	0.802111
3rft+2kst+2ibk+1prt (none)	0.841689	1ibk+1kst (none)	0.802111
1ibk+1kst+1nn+1rft (none)	0.840369	1kst (none)	0.795515
2ibk+2kst+2nn+2rft (none)	0.840369	1nn (none)	0.790237
1ibk+1rft+1lgr+1kst (none)	0.837731	1ibk+1rft (none)	0.783641
1kst+1rft+1ibk+1prt (none)	0.837731	1rft (none)	0.782322
2kst+2rft+2ibk+1prt+1dt (none)	0.837731	1lgr+1kst (none)	0.782322
2kst+2rft+2ibk+1prt+1nn (none)	0.837731	2ibk (none)	0.779683
Collaboration	**Accuracy**	**Collaboration**	**Accuracy**
1ibk+1kst+1rft+1dt (high)	0.843008	1kst+1rft (low)	0.808707
2ibk+2kst+2nn+2rft (low)	0.843008	1ibk+1kst (high)	0.80343
2kst+2rft+2ibk+1prt+1nn (low)	0.843008	2kst (low)	0.802111
2ibk+1kst+1rft (med)	0.841689	2kst (med)	0.802111
1ibk+1kst+1nn+1rft (low)	0.840369	1ibk+1kst (low)	0.802111
2rft+1kst+1ibk (med)	0.840369	1ibk+1kst (med)	0.802111
3ibk+2kst+2rft+1nn (low)	0.840369	2kst (high)	0.800792
3ibk+2kst+3rft (low)	0.83905	1kst+1rft (high)	0.800792
3rft+2kst+2ibk+1prt (med)	0.83905	1kst+1rft (med)	0.799472
4rft+2kst+2ibk (low)	0.83905	1nb+1kst (low)	0.791557

nearly identical for collaborative and non-collaborative teams; however, more best-accuracy teams utilize collaboration as opposed to not collaborating. For Indp mode teams, a non-collaborative team achieves the highest testing accuracy; however, there are more high-accuracy collaborative teams.

Examining the lowest accuracy teams, non-collaborative teams for both Full and Indp modes perform substantially better than those which collaborate. Further, those Indp mode teams which perform the worst appear to collaborate with high frequency. This suggests that certain combinations of learning algorithms, when collaborating often, experience a significant decrease in team classification accuracy. This result was unexpected, as we believed higher frequency would directly produce the best accuracy for Indp mode teams. Results indicate that collaboration frequency is coupled with both team size and learning mode. We therefore conclude that collaboration with Low or Medium frequency will lead to the best results, especially when performing distributed learning.

5.5.6 Individual Learner Contribution

The contribution and success of individual machine learning algorithms were investigated. By analyzing a learning algorithm's contribution to success, we can extract which learning algorithms are generally successful for this application, and those which also perform well

Table 5.13: 10 worst performing collaborating and non-collaborating teams, based on average testing accuracy, for Full and Indp modes.

Full Mode Teams		Independent Mode Teams	
No Collaboration	**Accuracy**	**No Collaboration**	**Accuracy**
1dtb (none)	0.62533	2dtb (none)	0.58971
2dtb (none)	0.631926	1dtb+1jrp (none)	0.616095
1nb (none)	0.637203	1dtb (none)	0.62533
2nb (none)	0.637203	2prt (none)	0.629288
1nb+1lgr (none)	0.639842	2nb (none)	0.630607
1jrp (none)	0.64248	1nb+1lgr (none)	0.631926
1lgr (none)	0.64248	1dtb+1prt (none)	0.635884
2lgr (none)	0.64248	1nb (none)	0.637203
1lgr+1dtb (none)	0.656992	1lgr+1dtb (none)	0.638522
1dtb+1jrp (none)	0.656992	1jrp (none)	0.64248
Collaboration	**Accuracy**	**Collaboration**	**Accuracy**
2dtb (high)	0.274406	2dtb (high)	0.315303
2dtb (med)	0.294195	2dtb (med)	0.368074
2lgr (med)	0.365435	2lgr (med)	0.37467
1lgr+1dtb (high)	0.39314	2lgr (high)	0.437995
2lgr (high)	0.411609	1lgr+1rbf (high)	0.474934
2nb (med)	0.42876	1nb+1dtb (high)	0.48153
2lgr (low)	0.439314	1nb+1lgr (high)	0.48285
2dtb (low)	0.463061	1lgr+1dtb (high)	0.489446
1nb+1lgr (med)	0.465699	2nb (med)	0.494723
2nb (high)	0.468338	2nb (high)	0.507916

together. If teams that contain a certain learning algorithm consistently perform better than others not containing that learning algorithm, then that algorithm can be considered valuable to (or a major contributor) a team's combined testing classification accuracy.

This is measured in two primary ways. First, as shown in Figure 5.20, average testing accuracy is compared for each learning algorithm over all teams in which that algorithm participated. On average, NB, LGR, RBF, DTB, and JRP perform the worst, whereas IBK, KST, NN, RFT, and DT perform the best. Teams containing DTB, LGR, and NB learners are less stable, as they can produce very poor classification accuracy when included in a team. On the other hand, learning algorithms such as IBK, KST, and NN are stable learning algorithm choices, as their minimum team accuracies are still comparably high.

Each algorithm has a comparable maximum accuracy, meaning that it was involved in one or more teams which contained heterogeneous learning algorithms that collectively produced high testing accuracy for both Full and Indp mode teams. These results indicate that for this application learning algorithms such as IBK, KST, NN, RFT, and DT are reliable when constructing a team of multiple classifiers. Learning algorithms such as NB, LGR, RBF, DTB, and JRP are not as reliable. Combining these algorithms is encouraged, to produce a team that has a heterogeneous mixture of strong homogeneous classifiers. Results are similar across learning modes.

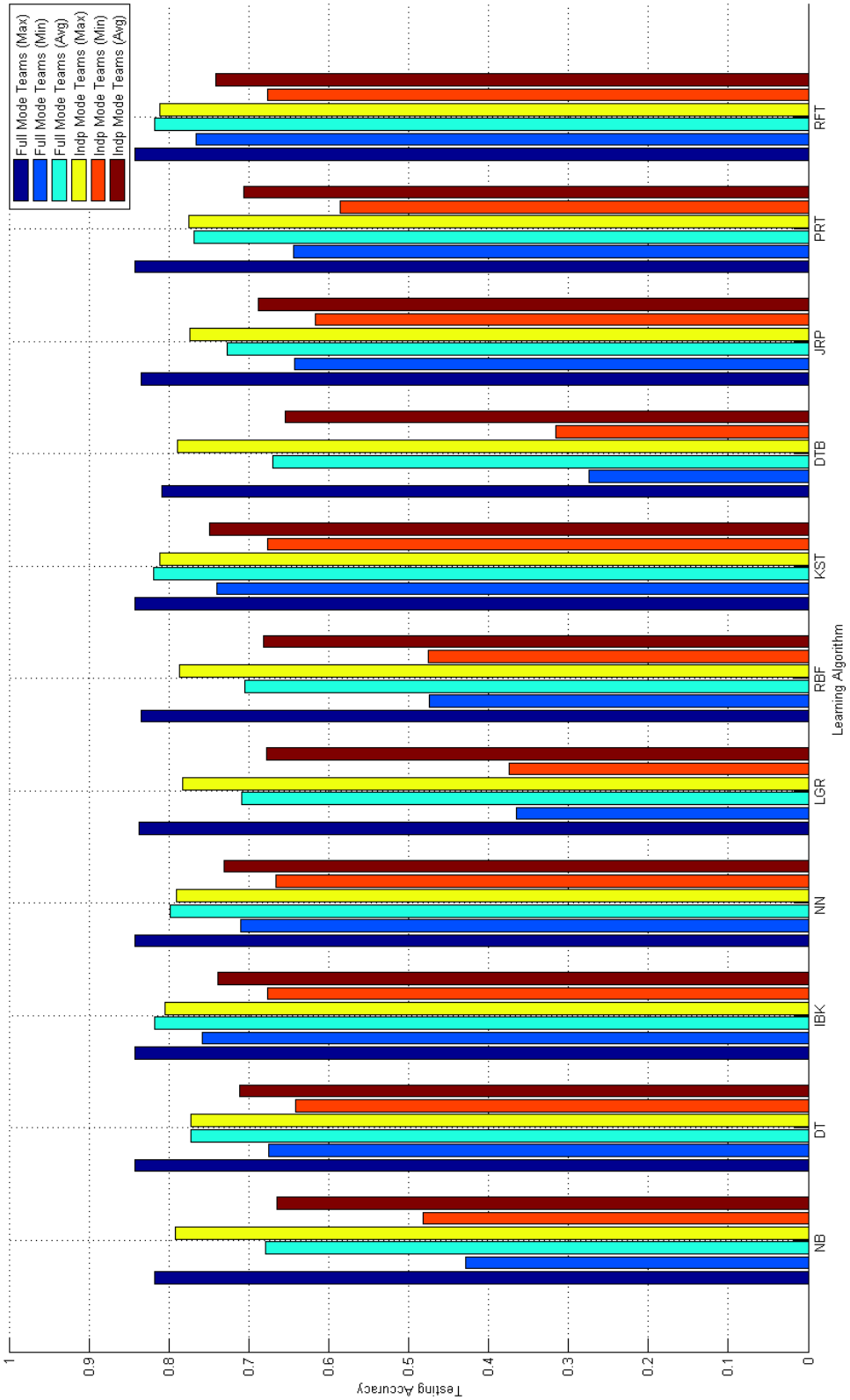

Figure 5.20: Minimum, maximum, and average testing accuracies over all teams containing one or more of each learning algorithm for Full and Indp mode teams. This offers a view into how teams performed based on their learning algorithm composition, specifically focused on membership of certain learning algorithms.

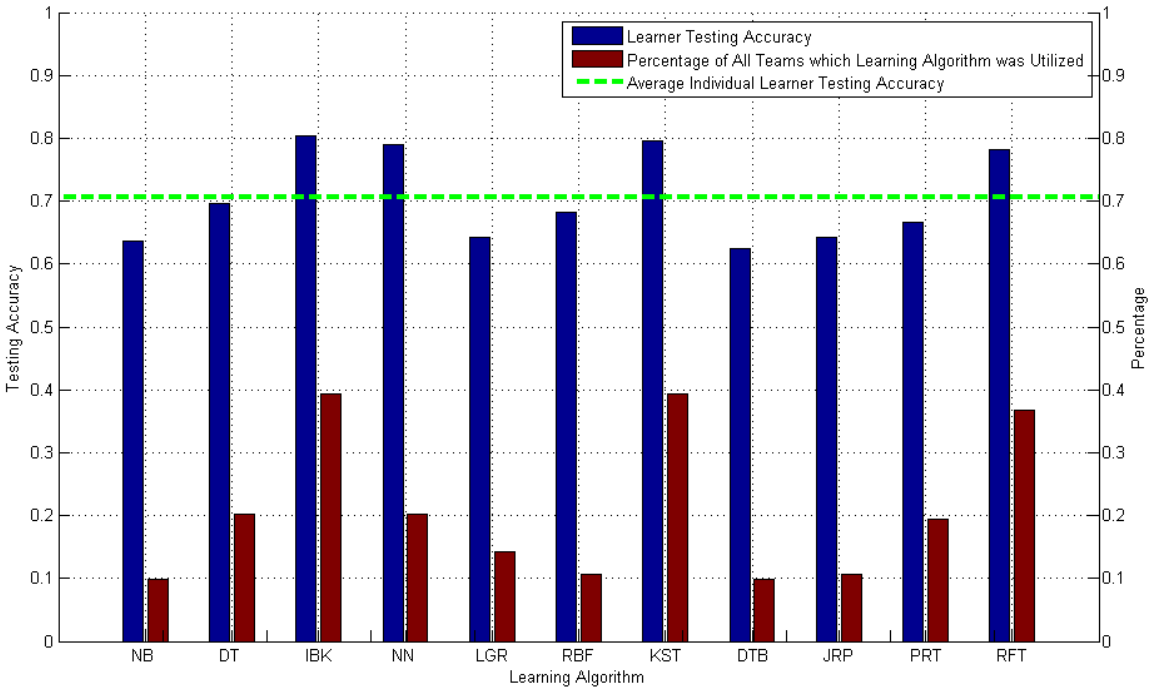

Figure 5.21: Individual learning algorithm testing accuracies for entire data set (team size of 1), accompanied by each learning algorithm's overall participation by the percentage of teams in which it was a member. This offers a view of general success and ranking of the individual learning algorithms for this data set.

As shown in Figure 5.21, the number of total teams which each learning algorithm participated in can be compared. This figure shows individual learning algorithm accuracies on the entire data set (i.e., size one teams), and learning algorithm participation over all team learning experiments as a percentage. The byproduct of the tournament-style experimental setup is the use of the best performing algorithms in more teams. It is also evident that the IBK, KST, and RFT classifiers are used in 20% more teams than the next best set of classifiers. This is a result of combining teams of size two to form teams of size four, and combining teams of size four to form teams of size eight. These three classifiers comprised more size two and four teams in comparison. The NB, DTB, JRP, and RBF classifiers were utilized the least over all teams.

5.6 Conclusions

Using the proposed Multi-Agent Collaborative Learning Architecture, we were able to obtain approximately 84.5% absolute accuracy, an improvement of about 6.5% over the best results that KGS was able to achieve on this rock facies classification data set. Several heuristics were discussed for constructing teams of multiple collaborative classifiers. A tradeoff between team size and team composition also became apparent, including the time required to complete the training, testing, and combination processes.

Highly diverse and heterogeneous team compositions are preferred, due to their robustness, complementary nature, and general good performance. A guideline for team construction was also discovered. A heterogeneous mixture of strong homogeneous teams, especially those compositions that are equally diverse (team size divided by diversity level being 1 to 2), allows a team of multiple agents to classify with high combined accuracy. Furthermore, collaboration was found to be most beneficial for homogeneous teams of size four, and generally produced an increase in testing accuracy when utilized with Low to Medium frequency.

Size four and eight teams proved to be well-suited for this data set. Combination methods of Average, Selective Average, Multiply, and Selective Multiply were observed to offer the highest combined team testing accuracy. Lastly, the IBK, KST, NN, RFT, and DT algorithms performed consistently better than the others. For this difficult classification problem, utilizing a team of multiple heterogeneous collaborative machine learning algorithms was necessary to maximize accuracy. These results, together with those in Chapter 6, are summarized and compared in Chapter 7.

Chapter 6

Multi-Agent Subglacial Water Classification from Polar Radar Data

6.1 Introduction

Remote sensing methods, such as radar and seismic/acoustic surveys, attempt to acquire data in order to infer properties of the subsurface from a remote location such as the surface, air, or space. A prominent example is the use of radar sensors to gather data from the polar ice sheets, specifically internal layers and the ice-bedrock interface, from the surface of an ice sheet. Other examples are satellite-based imagery and identification of landcover and events, and the observation of light and its characteristics to infer properties of distant galaxies and stars.

Ground and airborne depth-soundings of the Greenland and Antarctic ice sheets have been used for many years to determine characteristics such as ice thickness, subglacial topography, and mass balance of large bodies of ice. Radar sounding of ice sheets is challenging due to the rough surface interface, various stages of melting both on top of and within the ice sheet, and spatial variation of ice thickness and bedrock topography. Processing the data, including the incorporation of knowledge about the sensing medium, is important for proper interpretation and dissemination of accurate data to the scientific community. For example, in Greenland, water is present over continuous finite distances which are smooth, but do not qualify as large lakes that are evident in Antarctica.

At the University of Kansas, the Center for Remote Sensing of Ice Sheets (CReSIS) [39] performs polar research to gather data and model ice sheets to better understand global climate changes and the possible effects, including sea level impacts. We have designed, built, and utilized mobile robots to autonomously traverse polar terrain in Greenland and Antarctica and support radar remote sensing data acquisition [53][120][5][57][4][6].

Airborne systems have been developed to offer large-scale studies of polar regions. One main technology for this research is the development of sophisticated radars. CReSIS researchers have provided first views of the ice sheet bed in fast-flowing areas and certain internal ice sheet layers, and continue to provide detailed subsurface images and models describing their behavior and dynamics. CReSIS has a wide variety of radar data from various radar designs over many years. All of these data are available to the public, providing information such as latitude, longitude, radar travel times, bed echo intensity, and ice thickness for extended flight segments. This data repository can then be used for advanced machine learning, data mining, and modeling efforts for polar environments.

In this dissertation, we utilize ice-penetrating radar data collected in Greenland in May 1999 and September 2007 as part of a model-creation effort for subglacial water presence classification. Using radar data from ice sheets, the goal is to learn a model of water presence or absence for each radar measurement. This enables the production of classified water presence maps for the scientific community without needing to drill resource-intensive ice cores. Specifically, a detailed study of team learning, collaboration, and decision combination has been conducted. Experimental results are presented, including successful team compositions, individual learning algorithm contribution, team size, collaboration frequency, and team diversity. Finally, the primary team learning and collaboration aspects discovered as part of this investigation are summarized.

6.2 Background and Related Work

Machine learning techniques have seen limited application to ice sheet and polar subsurface data. Most efforts involve identification or tracing of specific layers that hold historical importance. Internal layers of ice sheets have been investigated to predict the depth and thickness of certain layers. For example, initial efforts to predict the depth and thickness of the Eemian Layer in the Greenland ice sheet utilized a Monte Carlo Inversion of the flow model to estimate unknown parameters constrained by the internal layers [29][28]. One of these parameters is the basal melt rate, an important parameter for ice sheet models. Similar work has been done to classify the presence of bottom crevasses, and to estimate their height using radar data of the Ross Ice Shelf in Antarctica [98]. A physical model was developed by studying the radar data and long echo tails that were found to be characteristic of bottom crevasses. That work largely involved studying the data to determine power reflection coefficients, and did not incorporate any regression or machine learning methods.

Efforts to classify subsurface layers, occasionally called facies in the literature, using ground penetrating radar (GPR) are more abundant. Together with seismic surveying, GPR allows the remote sensing of subsurface properties to guide decisions of where to drill for oil or natural gas, as well as help explain observed surface changes. Radar reflection patterns can be used to distinguish echo returns in different regions, which lends well

to machine learning methods to associate such patterns with distinct classes. In [88], a neural network was utilized to learn and predict patterns of radar reflections, termed "radar textures", in GPR data collected from a river delta in Canada. They found that higher classification accuracies were obtained when using attributes that incorporated the spatial arrangement of reflections, such as spatial covariance, as opposed to amplitude. It was also found that reflection dip angle was a more important radar facie discriminate instead of dip direction.

GPR signal interpretation is one of the more studied applications of subsurface classification, as interpretation of subsurface materials or layers requires coring or drilling to determine ground truth. Experts are required to manually study the materials to segment them into distinct rock or object classes. These data can then be used by classifiers to learn a model between GPR return signals and the excavated ground truth. As GPR has been used for many applications, machine learning can aid in automating these tasks by learning a model, interpolating between measurement sites, and characterize the subsurface at other unknown locations. Machine learning algorithms such as support vector machines (SVMs) have been used in many signal interpretation applications, including roadbed degradation recognition [154].

Learning classifiers have been recently employed for autonomous polar event detection via satellites. As part of NASA's New Millennium Program, the Autonomous Sciencecraft Experiment (ASE) has been used for detecting dynamic events - such as volcanic eruptions, floods, and cryospheric events – using onboard science algorithms. One such focus was on the detection of ice-based surface events (lakes freezing/thawing and sea ice breakup) [31][32]. As part of that work, four pixel-based classifiers (manually constructed classifier, exhaustive threshold band ratio search classifier, decision tree, and SVM) were employed to generate models to recognize events using hyperspectral images. The SVM classifier, which has seen regular use onboard the EO-1 spacecraft, successfully classified sea ice breakup near Antarctica which autonomously triggered a spacecraft reaction. Such a classifier can be used to identify high priority data and reduce the bandwidth for data communication back to Earth. In [119], preliminary results are presented for unsupervised discovery of geophysical processes (including snow, ice, and clouds) from a spaceborne instrument over Greenland. Their results showed that regional classifiers provide higher classification accuracy for polar regions, but still had difficulty differentiating between certain clouds and snow/ice. Others have taken a knowledge-engineering approach to build an intelligent satellite sea ice classifier by creating rules from sea ice experts [118].

From these works, it has been shown that using machine learning to create models utilizing polar radar data analysis is not only feasible, but also can provide high levels of accuracy while offering a significant increase in efficiency. In the following sections, we discuss our approach, using multiple heterogeneous collaborative learning agents, to further increase accuracy and efficiency for the application of subglacial water presence classification

from radar data acquired on the Greenland ice sheet. Experimental results focus on team size, composition, collaboration frequency, and data distribution. This approach could also be applied to the Antarctic and other large bodies of ice, but is not explicitly discussed here.

6.3 Radar, Coring, and Subglacial Water

As part of NASA's Program for Arctic Regional Climate Assessment (PARCA) and others, the University of Kansas and CReSIS have collected ground and airborne radar data in Greenland which cover the majority of the continent's ice sheet. CReSIS now hosts these data for public use. Other major efforts, such as ice coring, have produced data that complement these radar data sets to study subglacial activity of the Greenland ice sheet.

Studying the extent/presence of subglacial water is important, as the state of the bed of the ice sheet (wet or frozen) provides information about flow, friction, and roughness of the underlying bedrock interface. The presence of water (wet state) means that the interface is likely slippery, or that heat has caused a downflow of water from upper or nearby portions of the ice sheet. Portions of the ice with larger amounts of water exhibit faster flow properties and dynamic changes of the ice in those regions due to lubrication at the bed. The lack of water means that the interface is frozen, likely exhibiting more friction and therefore little movement. A smooth bedrock interface lends more to sliding of the ice sheet along its surface, whereas a rough interface introduces more friction between the ice sheet and bedrock on which it rests. If water is present at the bed, then the interface between the ice and water will be smoother, compared to a mixed water and rock interface (which will exhibit the shape of the underlying rock). Transport of material, however, may cause this lower rock interface to change over time.

Ice coring offers an additional method to study, among many other things, the state of the ice-bedrock interface by manually examining the bottom of the cored ice column. Thus, ice cores represent the only ground truth available for large-scale ice sheet modeling and remote sensing validation. In Greenland, there are two primary drill sites which we focus on in this work, namely, GRIP and N-GRIP. The Greenland Ice Core Project (GRIP) drilled a core from 1989-1992 to a depth of 3029 meters, located at 72.58 N, -37.63 W [89]. The subglacial state of this core was found to be frozen. The North GRIP (N-GRIP) drilled a core from 1999-2003 to a depth of 3085 meters, located at 75.1 N, -42.32 W [142]. The subglacial state of this core was found to be wet (high melt).

As a radar transmits energy down to and into the ice – typically from one or more transmitting antennas mounted on the wings of an aircraft – changes in dielectric properties cause the signal energy to reflect and refract. Larger changes in dielectric properties (such as air to ice, and ice to rock) cause strong reflection intensities to be received by the radar's receiving antenna(s). Figure 6.1 shows an example radar echogram over an ice sheet,

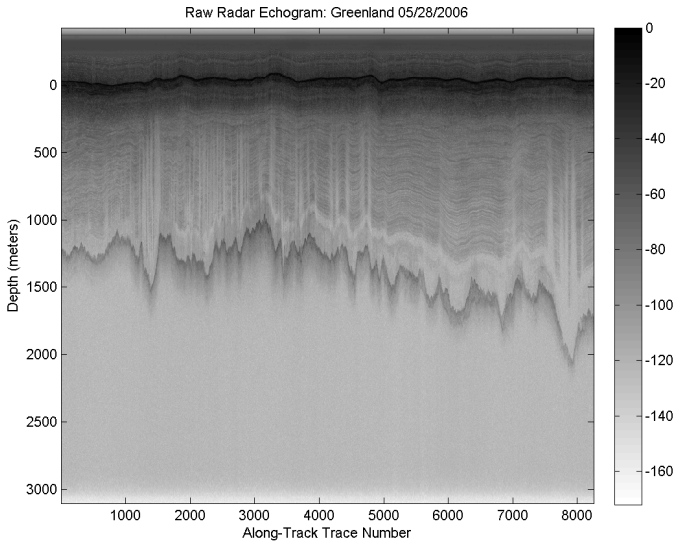

Figure 6.1: Radar data echogram acquired over Greenland in May 2006, showing reflection intensities ranging from strong (dark) to weak (light). The surface (top) and bedrock (bottom) interfaces are the strongest standalone reflectors, and internal layering (middle) can be visualized and spatially tracked.

illustrating the reflection of internal layers and the bedrock interface beneath the ice sheet. Each column of the echogram is a trace (right), representing sensor measurements with time (and therefore depth). Water is a comparatively strong reflector when surrounded by ice/rock and affects the sensed signal return intensity, whether internal to or at the bottom of the ice sheet. Frozen ice against bedrock also represents a strong reflecting interface. Thus, other properties of the signal and the return envelope can be analyzed to determine whether water is present or not. These factors highly depend on the radar and its settings.

6.4 Implementation and Processing

The work presented here is developed around a method that recovers radar reflection characteristics at the bed of the ice sheet that discriminate between frozen and wet subglacial properties for each trace of a radar survey [91]. The method is said to be most appropriate for Greenland, but a similar method could be applied to the Antarctic ice sheet. It is independent of radar characteristics, as the two primary signal return attributes are radar independent if computed properly and normalized. Numerical and statistical analyses of radio-echo profiles and interface geometries, as well as ice and water physics, represent the basis of the method. Results were validated using known basal water presence at the GRIP and N-GRIP core sites. The reader is referred to [91] for more details of the method, and related work on radio-echoing and depth-sounding of ice sheets.

13 strategic radar flights covering various portions of the Greenland ice sheet were flown throughout May 1999. These flight lines are shown in Figure 6.2, with one line/color per

Table 6.1: ICARDS Radar System Specifications [91].

Description	Characteristic
Radar Type	Pulse compression
Antennas	4-element $\lambda/2$ dipole arrays
Carrier Frequency	150 MHz
Up-chirp Bandwidth	17.00 MHz
Transmitted Pulse Width	1.6 ms
Compressed Pulse Width	60 ns
Peak Transmit Power	200 W
Pulse Repetition Frequency (PRF)	Up to 18.4 kHz
Coherent Integrations	32 to 4096
Incoherent Integrations	0 to 64,000
Baseband Inphase & Quadrature Bandwidth	8.5 MHz
Analog/Digital Dynamic Range	12-bit 72 dB
Sampling Period	53.3 ns (18.75 MHz)
Sampling Delay	0.080-300 ms
Pixel Window in Ice (Range)	4.494 m
Trace Spacing	$\sim$135 m

flight. The radar data set used to develop the original method was collected from aerial surveys over Greenland on May 14 and 25, 1999. These surveys included the GRIP and N-GRIP core sites, as shown in Figure 6.3.

Specifications of the radar used to acquire these data, named Improved Coherent Antarctic and Arctic Radar Depth Sounder (ICARDS), is presented in Table 6.1. Uncertainty of the radar ice thickness measurements have been estimated to be $\pm$10 meters from comparisons to ice coring sites. We reproduced the results, with minor differences, of the published method, with minor modifications, on the same data set.

GPS technology allows the same flight lines to be flown for data acquisition over several years to study regional ice sheet changes due to climate and dynamics. In this work, we use the method's radar independence to generate a subglacial water presence model for radar data acquired in 2007 from a new radar system, called the Multi-Channel Radar Depth Sounder (MCRDS), over an identical extended flight segment covering the GRIP and N-GRIP core sites. These flight lines (May 14, 1999 and September 17, 2007), which comprise the primary data set for our study, are shown in Figure 6.4. Table 6.2 presents the specifications of the MCRDS radar system.

Using the extracted basal water presence information by the published method for the 1999 ICARDS radar data, training and testing sets for classification were constructed for our machine learning study. This assumes regional similarity in water content and that the water presence has not drastically changed (prominently wet or frozen) at the GRIP and N-GRIP sites in the years from 1999-2007. Given that these sites are located near the Greenland ice divide, this assumption can be made without loss of generality.

This section presents the implementation details associated with processing the radar data for classification. The following sections discuss the implementation, classification,

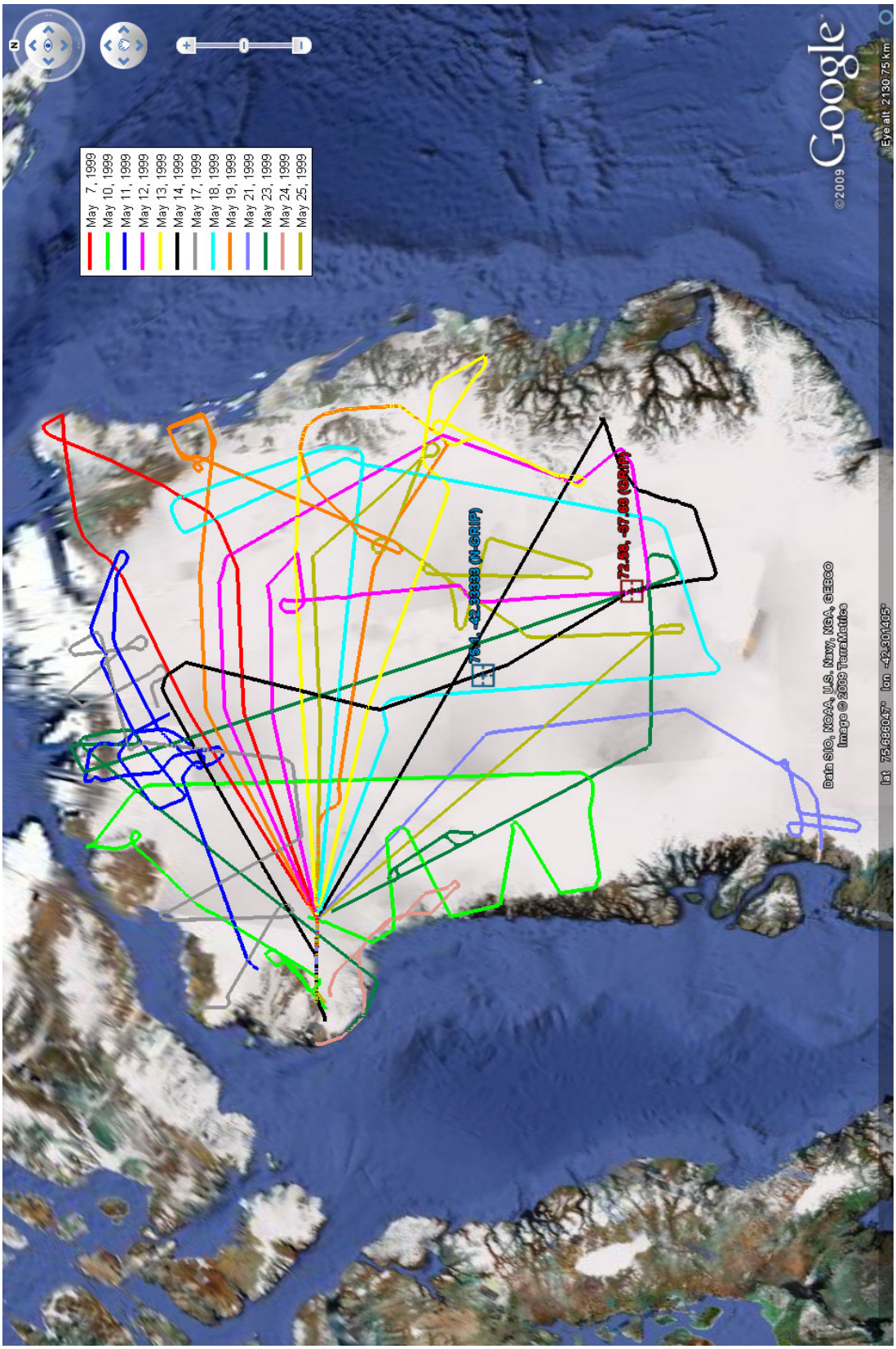

Figure 6.2: Flight lines for all airborne radar surveys flown throughout May 1999 over various regions of Greenland, with one line per flight.

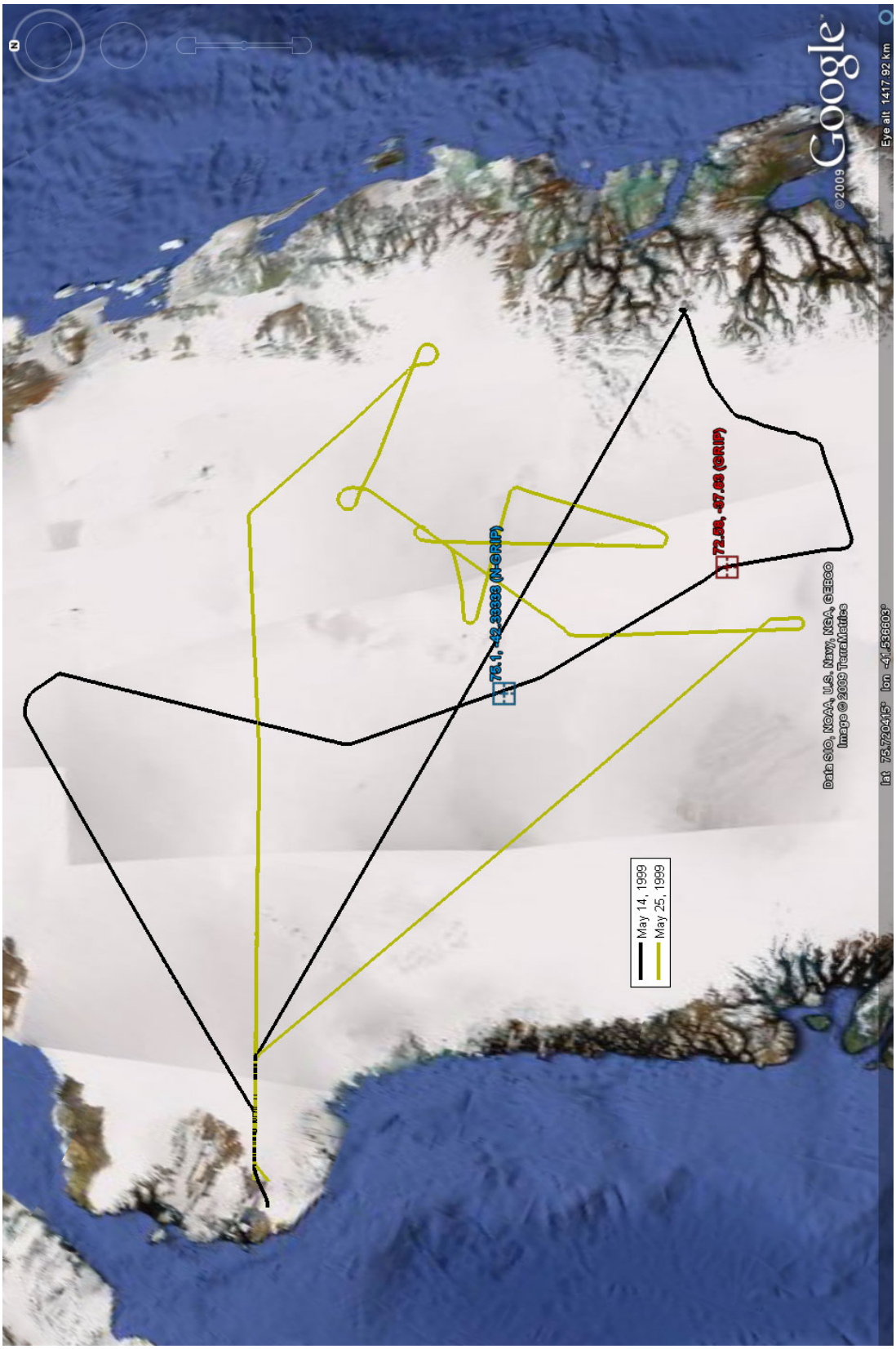

Figure 6.3: May 14, 1999 and May 25, 1999 flight lines, including the GRIP and N-GRIP core sites.

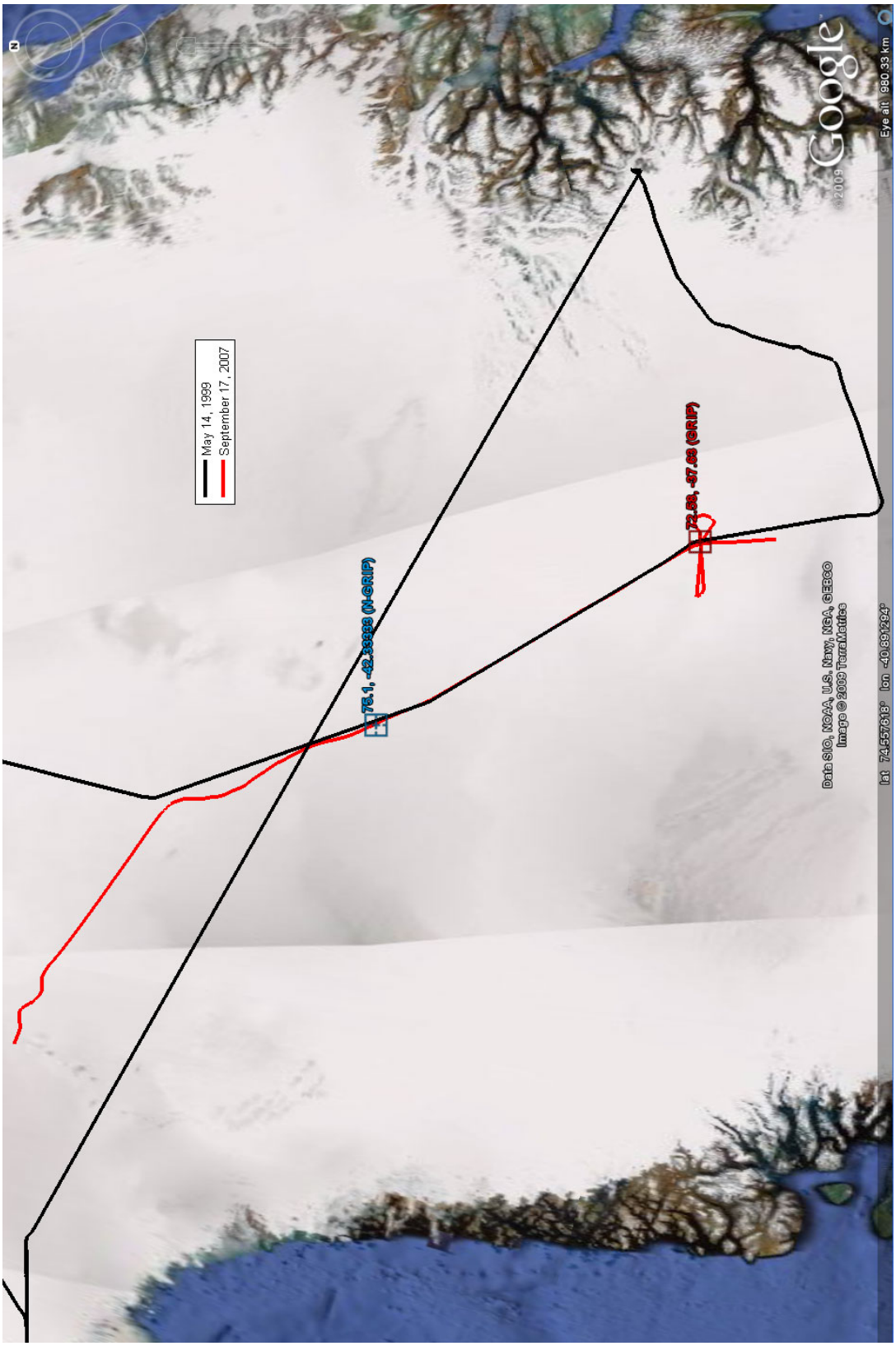

Figure 6.4: September 17, 2007 MCRDS flight line compared to the May 14, 1999 ICARDS flight line. The flight segments which are closest represent the primary area of study.

Table 6.2: MCRDS Radar System Specifications [85].

Description	Characteristic
Receive Channels	4
Transmit Location	(0,0)
Receive Locations	(-1.5,0),(-0.5,0),(0.5,0),(1.5,0)
Waveforms	1
Waveform Duration	10 us
Sampling Frequency	120 MHz
Start Frequency	140 MHz
Stop Frequency	160 MHz
Transmit Pulse Window	None (Uniform)
Trajectory	1 km (Y direction)
Trajectory Sampling	0.5 m
Target Location (in Ice)	1500 m
Receive Samples	5000
Platform Height	500 m
Pixel Window in Ice (Range)	1.4087 m
Trace Spacing	∼13.5 m

and results of a detailed multi-agent machine learning study for the application of radar-independent model generation for polar subglacial research. The implementation incorporates additional information to that of the original method for more robust water presence classification and transfer of the concept to radar data acquired with new technology in the midst of global warming and climate change. Matlab [132] was utilized for processing and manipulating the radar data into attributes for classification using the described method with slight modifications.

6.4.1 Smooth and Filter Data

Pre-processing of raw data consists of presumming (averaging) in hardware during acquisition, as well as offline coherent integration, range compression, frequency filtering, initial incoherent integration, azimuth compression, and waveform combination. This results in pre-processed data products, which are then passed through several cleaning stages before attributes can be computed using the received radar returns.

First, the data are incoherently integrated (using both magnitude and phase information for averaging) over an along-track distance of ∼200 meters for smoothing purposes. Next, individual traces are removed which exhibit the following "bad data" aspects:

- Latitude or longitude is erroneous (well away from traversed flight line, possibly due to poor GPS visibility during aircraft turns)
- Visually smeared surface or bed portions of preliminary-processed echogram
- Non-characteristic reflection magnitudes at bed, likely due to aircraft banking, nearby mountains or exposed rock, or large amounts of surface water
- Ice thickness is <300 meters (surface and bed reflections difficult to distinguish)

- Bed reflection is not visible
- Ice thickness or bottom depth could not be computed due to lack of visible bed reflection

Portions of the airborne survey that contain banks/turns of the aircraft may produce results that are not reliable or indicative of the true subglacial state. These portions are therefore removed prior to attribute calculation and processing.

6.4.2 Data Calibration and Boundary Alignment

Most airborne ice sheet surveys begin on a known smooth, frozen path (runway) or portion of the ice sheet. This represents another known aspect of the ice sheet, to accompany ice cores, with which can be used for calibrating and bounding the intensity distributions received by the radar. These portions of the data are assumed to be indicative of typical frozen subglacial interfaces. The GRIP (frozen) and N-GRIP (wet) core sites provide reflection intensities that are known to range from frozen to wet. By focusing on these sources of ground truth, encompassing the entire spectrum of subglacial water extent (fully wet to fully frozen), the method can be applied to the entire ice sheet.

The observed lower intensity bound from these areas can be used as a reference. The upper intensity bound in the received intensities over flights represents wet interfaces with varying fractions of water content. Using the lower intensity bound, all extended flight segments have their intensity distributions shifted to exhibit matching lower bounds. This, if done correctly, removes the data's dependence on radar settings and sensitivities, yielding radar independence.

During processing, received bed intensities are transformed from varying over 40 dB down to $\sim$20 dB for more reliable basal reflection coefficient determination. Intense returns accompanied by high abruptness are indicative of the water presence. Large internal layer disturbances in the ice (ice stress and deformation) are also indicative of locations which likely contain water at the base. The original method verified this using the coherence index and echograms exhibiting such distortion [91]. The method underestimates the extent of water presence, producing a decision of frozen, where intensity is high but abruptness falls below threshold.

6.4.3 Radar Signal Attributes

Three primary radar data attributes are utilized by the method at each trace: relative signal intensity, signal abruptness index, and signal coherence index [91]. The first two are used to determine the binary result of wet or frozen, where relative signal intensity is the primary determinant. They employ coherence to further analyze and explain the results. These attributes are now discussed in more detail.

Relative Intensity

This attribute represents the radar signal return intensity at the basal interface, including the first return and echo tail. The strength of the reflection intensity degrades with depth, as the signals attenuate and experience loss as they travel through solid ice and other media. Thus, the intensity variations can be modeled by an equation similar to the following to depth-adjust the received reflection due to rate of absorption in ice:

$$B = \frac{2.3 \times 3000}{H + 2000} \tag{6.1}$$

where H represents surface altitude, which governs intensity variation as a function of depth. The intensity after adjustment still requires recalibration for variations of the system, but is otherwise representative of the reflection coefficient at the bed of the ice. Pre-processed radar data go through several stages of processing and adjustment to arrive at the final relative intensity value of the bed reflection.

Abruptness Index

This attribute provides information on whether the echo tail is beamwidth-limited. Abruptness constrains the angular spectrum of the received signal. The abruptness index can be computed as follows:

$$Abruptness = \frac{P_{peak}(X)}{P_{agg}(X)} = \frac{\max(P_{window}(X))}{\sum(P_{window}(X))} \tag{6.2}$$

The numerator in the calculation is the peak (max) basal reflection intensity (P_{window}) for a specific trace/position. The denominator is the aggregate (sum) basal reflection intensity, or the sum of return intensities over depth (X) bins in the basal return window (P_{window}). High values of abruptness are correlated with high aggregate intensity, but is not a clear discriminator of wet versus frozen on its own. Thus, high abruptness that is accompanied by intense returns is indicative of the presence of water. For the pulse length and sample rate used by the radar, the reflection abruptness is typically between 0.05 and 0.5. The published method [91] uses a value of 0.25 to identify locations that are significantly smooth.

Coherence Index

This attribute provides information on whether the first Fresnel zone is determined by roughness or wavelength. Thus, coherence is another measure of smoothness at a smaller scale. Coherence can be calculated using the following equation:

$$Coherence = \frac{\sum(abs(\sum(R_{complex}(D,X))^2))}{\sum(\sum(abs(R_{complex}(D,X))^2))} \tag{6.3}$$

The numerator sums the complex echo signal ($R_{complex}(D, X)$) over the distance along-track (D) for coherent and incoherent integrations, which is evaluated for a slope that maximizes the coherent integral. The denominator is the signal power summed over the distance (D) and depth (X) bins. Coherence is calculated from the complex received signal and abruptness, independent of the received signal intensity. Coherent integrations are performed over a distance of ∼200 meters. A coherence value of 0.3 or larger is said to indicate flat surface segments.

Coherent integration is in the along-track direction, using only the magnitudes (no phase information) for averaging. Incoherent integration is also in the along-track direction, but uses both magnitude and phase information for averaging. Received intensities are incoherently averaged over a distance of ∼200 meters to achieve smoothing and reduce signal intensity variation. For smooth interfaces, the response is more impulse-like; whereas slightly rough interfaces exhibit more intensity variation, and rough interfaces produce intensity variations more evenly distributed. The intensity distribution is largely caused by the scatter of energy as a function of interface roughness. Variance of the received signal is further reduced by aggregating the basal reflection envelope over all beam pattern directions [91].

6.4.4 Computing Data Attributes

Each trace has the bed reflection (peak intensity depth from expected bed region of A-scope, where visible) selected by a radar expert, assisted by surface- and bed-picking software. Figure 6.5 demonstrates this process using the CReSIS picking software, showing the surface returns fully picked and the bed reflections partially picked for a segment of polar region radar data. The surface can be easily selected, as it typically represents the first and largest reflection intensity spike in the trace's return profile. Using the corresponding radar echo time, peak depth and ice thickness can be calculated.

A five-sample window is created around the peak bed reflection for each trace, containing two samples above and below the determined peak reflection intensity depth. Abruptness can be computed using the peak reflection and the sum of the bed reflection window intensities (i.e., the depth bins). The received reflections are depth-adjusted, as previously described, using the ice thickness at each trace. Peak reflection intensity is converted to a relative intensity value in dB by dividing the peak at each trace by the maximum peak intensity in the data's basal reflection window.

An important aspect of this method is the alignment of the lower boundary of the intensity distributions. The upper boundary corresponds to the wet interface, while the lower boundary corresponds to the rock/frozen interface. By studying the ground-truth GRIP (frozen) and N-GRIP (wet) coring sites, this lower/frozen reflection intensity boundary can be found. For radar data taken over these core sites in 1999, this lower bound was found to be ∼31.37 dB. Each flight's lower boundary is aligned using this reference

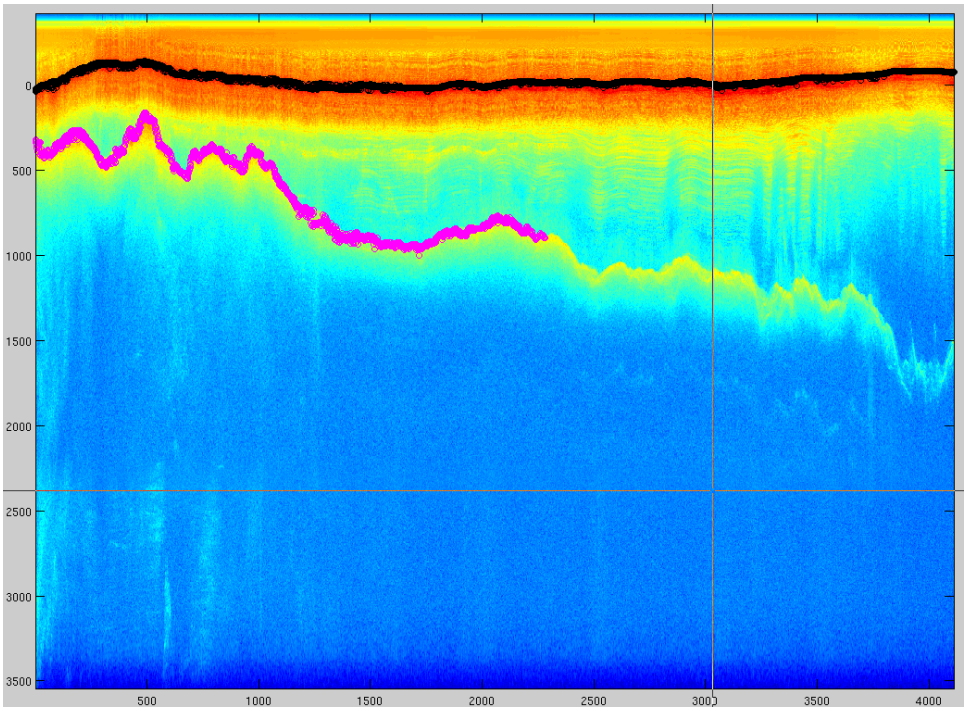

Figure 6.5: The process of picking the surface and bedrock reflections (required to compute ice thickness), where the surface has been fully picked (top) and the bedrock interface partially picked (middle).

value to perform the final step of removing the effects of radar settings.

Figure 6.6 plots the relative intensity distribution over the May 14, 1999 flight segment containing wet and frozen bed characteristics (survey data over the N-GRIP and GRIP core sites, respectively), showing a bimodal distribution (one hypothetical Gaussian distribution for the wet intensities and another for the frozen intensities). Some portions of the 20-30 dB variation in intensity overlap at lower probabilities. The overlapping area is where it can be distinguished between wet (higher relative reflection intensities) and frozen (lower relative reflection intensities) areas by computing a threshold which separates the bimodal distribution into wet and frozen portions. Observed intensities showed a separation of $\sim$13 dB between the mean of each Gaussian mode with high confidence [91]. Once the thresholds and lower bound of the intensity distribution are found, the method can be applied to radar data acquired over the entire Greenland ice sheet to recover the extent of basal water presence. Finally, coherence is computed for each trace using the previously described formula.

Figure 6.7 presents the attributes, thresholds, and binary basal water presence classification for the extended flight segment 42-70 from the May 14, 1999 flight. Along with the echogram for this segment, the bedrock reflection window is shown, on which relative intensity calculations are based. The empirically-found thresholds of -15 dB (relative intensity) and 0.27 (abruptness) dictate water presence for each radar trace, as previously

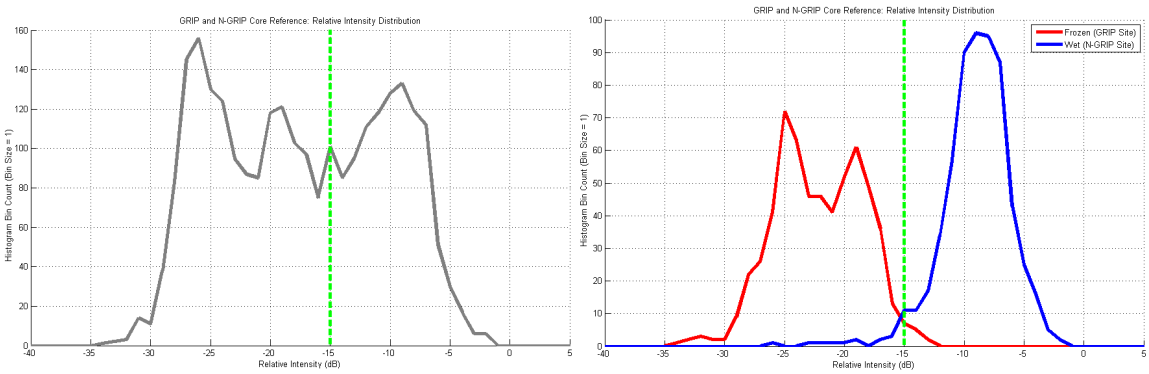

Figure 6.6: Relative basal reflection intensity distributions for the entire May 14, 1999 GRIP and N-GRIP extended flight segment (left), and for those traces near GRIP and N-GRIP sites (right).

described. Coherence values above 0.3 represent smoother surfaces, which are utilized as additional signal information for classification. The method classifies, out of a total of 7995 traces for this segment, that 1565 are wet and 6430 are frozen. The GRIP and N-GRIP sites are classified as frozen and wet, respectively, and water tends to be present in continuous regions.

Figure 6.8 presents these attribute details and binary basal water presence classification for the extended flight segment 39-46 from the May 25, 1999 flight. Along with rougher bedrock topography, it can be seen that water is detected on relatively smooth interfaces in this region and in the valleys between peaks, representing water collection. The method classifies, out of a total of 2000 traces for this segment, that 307 are wet and 1693 are frozen.

6.4.5 Confidence-Based Water Presence

Rather than the binary wet or frozen decision for each radar trace, confidence levels can be used, based on the combination of each attribute's value with respect to its binary threshold. Here, we specify for each attribute fuzzy confidence values of *Low*, *Medium*, and *High*. For classifying individual traces, we discretize the possible classes to *Low*, *Medium*, and *High* confidence of water presence, as well as *Low*, *Medium*, and *High* confidence of water absence. Additionally, we include *Uncertain* for values very close to the thresholds.

Confidence-based water presence classification is an extension of the binary water extent method. It takes into account aspects that the original binary method of [91] notes that it underestimates. For example, the original method underestimates the extent of water presence (marks trace as frozen) where intensity is *High* and abruptness falls below threshold. We can now represent this as *High* relative intensity and *Low* abruptness to properly classify these instances as containing water with *Medium* confidence.

In order to arrive at a confidence-based data set, the data are processed similar to the

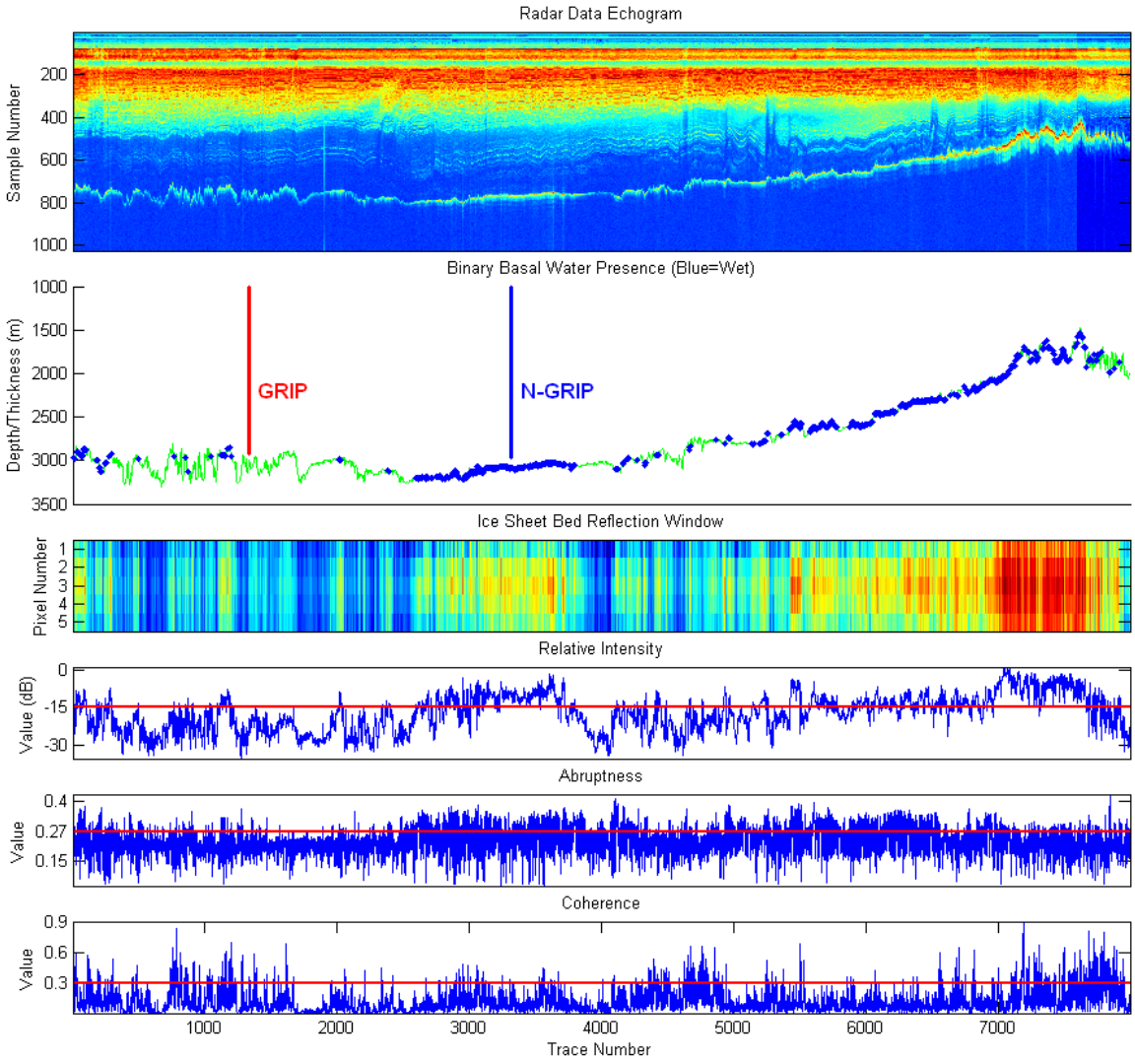

Figure 6.7: Attributes, associated thresholds, and binary water presence for each radar trace from the 42-70 extended flight segment from May 14, 1999. GRIP and N-GRIP core locations are shown for reference.

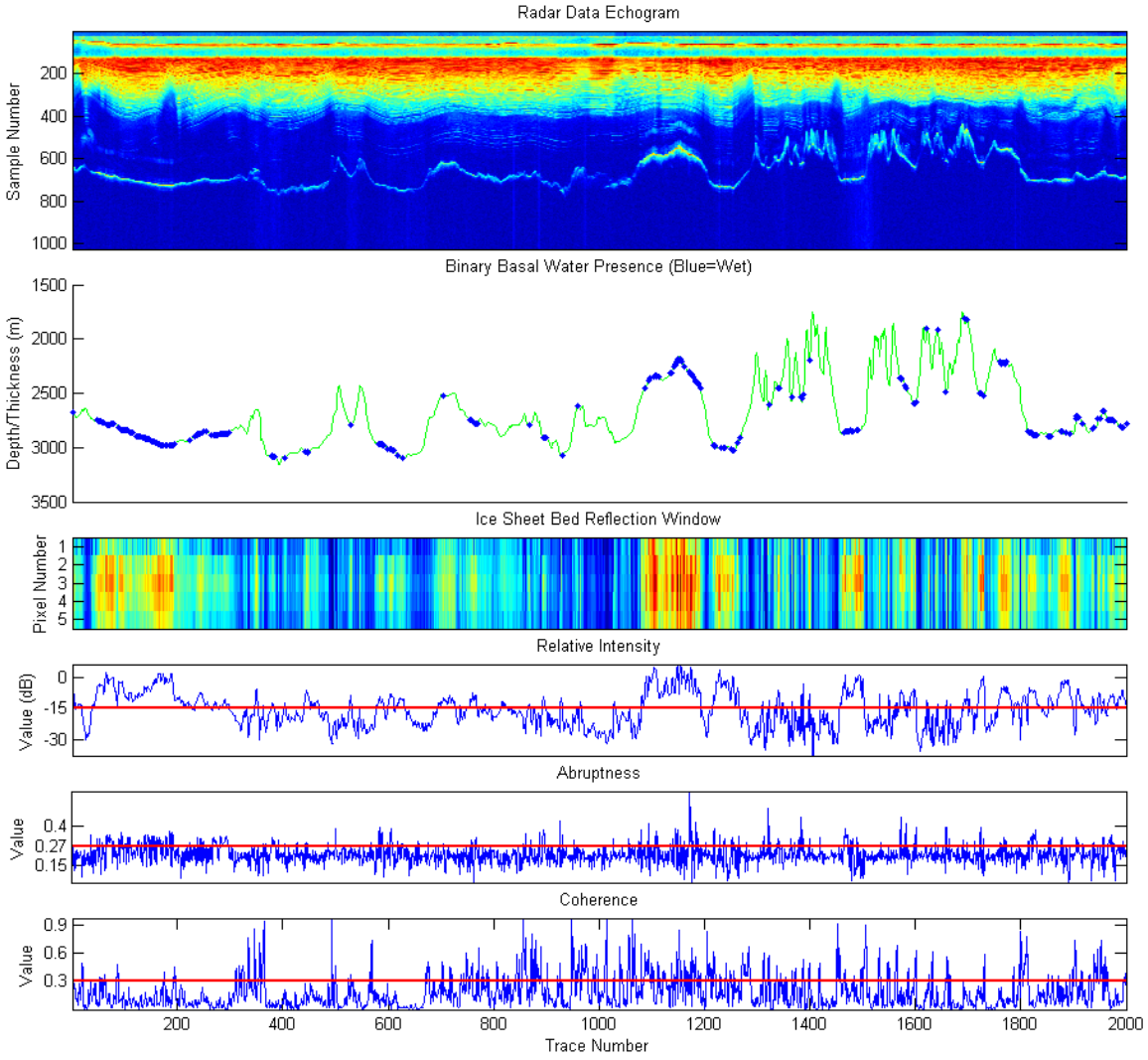

Figure 6.8: Attributes, associated thresholds, and binary water presence for each radar trace from the 39-46 extended flight segment from May 25, 1999.

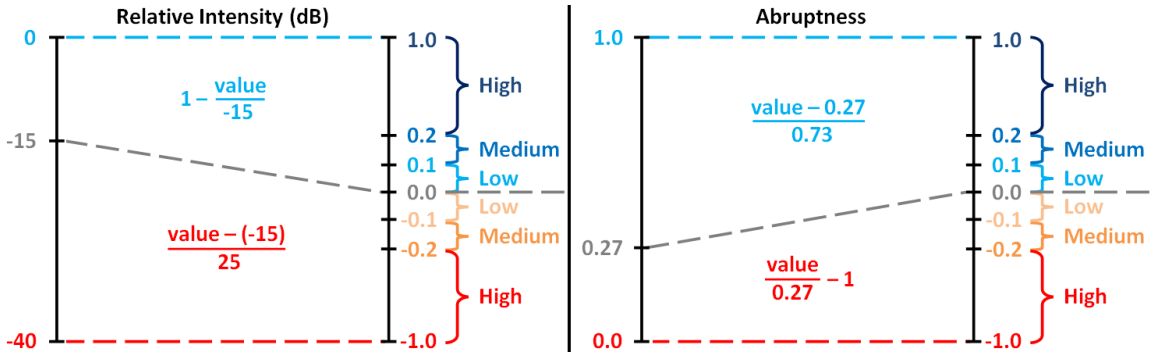

Figure 6.9: Method used for confidence classification/coloring to transform relative intensity and abruptness attribute value bounds to the range of [-1,1]. 0.0 represents the threshold value for each attribute after the transformation.

binary method, but with two additional final steps. First, using the associated thresholds, each radar trace has its relative intensity and abruptness attribute values passed through a boundary transformation step to convert them into the range of [-1,1]. The threshold value for each attribute represents 0.0 in this new range of *High* above (1.0) to *High* below (-1.0) threshold. Figure 6.9 illustrates this boundary transformation step for relative intensity and abruptness. Confidence level bounds for each attribute range from:

Above Threshold: High (0.20,1.0], Medium (0.10,0.20], and Low [0.0,0.10]
Below Threshold: Low (0.0,-0.10], Medium (-0.10,-0.20], and High (-0.20,-1.0]

Now that each attribute has been discretized into fuzzy intervals, rules can be formulated which govern the assignment of each trace into confidence levels of water presence and absence. The following are the primary rules which were utilized to accomplish this. These are accompanied by other supporting rules to handle those combinations of values not shown here.

- **Water Present:**
  - High Confidence: High (above) relative intensity + High (above) abruptness
  - Medium Confidence:
    - Medium (above) relative intensity + Medium (above) abruptness
    - High (above) relative intensity + Low (below) abruptness
  - Low Confidence: Low (above) relative intensity + Low (above) abruptness
- **Water Absent:**
  - High Confidence:
    - High (below) relative intensity + High (below) abruptness
    - High (below) relative intensity + High (above) abruptness

  - Medium Confidence: Medium (below) relative intensity + Medium (below) abruptness
  - Low Confidence: Low (below) relative intensity + Low (below) abruptness
- **Uncertain:** relative intensity + abruptness between Low (above) and Low (below)

The closer the values are to the thresholds, the lower are the confidence of presence or absence of water. In general, water is typically present when relative intensity and abruptness values are *High* (strong return from a smooth surface), or where relative intensity is *High* and abruptness is slightly below threshold. The *Uncertain* class is included for those traces exhibiting attribute values very close (within 0.02) to their thresholds. A byproduct of this approach is a more continuous classification of regions exhibiting characteristics of a predominantly wet or frozen bedrock interface.

Figure 6.10 compares the binary and confidence-based basal water presence classification models for the extended flight segment 42-70 from the May 14, 1999 flight. As the confidence-based classification is an extension of the binary classification model, GRIP and N-GRIP are still dominantly classified as frozen and wet, respectively. All shades of blue indicate water presence, with darker shades representing higher confidence (values further away from the thresholds). All shades of red indicate lack of water presence, with darker shades representing higher confidence. Traces with high intensity and abruptness just below threshold are now classified as low-confidence water presence.

Using the described modifications, it can be seen that a smoother water presence transition can be realized with high-confidence gaps filled in with lower confidence water presence. From a total of 7995 traces, 3823 were classified as various confidence levels of water presence (424 *High*, 1072 *Medium*, 2327 *Low*); 4134 were classified as various confidence levels of water absence (801 *High*, 1962 *Medium*, 1371 *Low*); and 38 were considered uncertain due to how close both relative intensity and abruptness were to the thresholds. Use of confidence levels more than doubled the amount of traces considered as containing water, largely due to the high number of *Low* confidence water presence decisions.

Figure 6.11 presents the classification comparison for the extended flight segment 39-46 from the May 25, 1999 flight. Confidence-based colors are as before, with blue representing wet and red representing frozen. From a total of 2000 traces, 1033 were classified as various confidence levels of water presence (204 *High*, 300 *Medium*, 529 *Low*); 959 were classified as various confidence levels of water absence (159 *High*, 1552 *Medium*, 248 *Low*); and 8 were considered uncertain due to how close both relative intensity and abruptness were to the thresholds. Confidence levels triple the number of wet traces, primarily due to an abundance of *Low* confidence traces as before. This segment illustrates the more continuous water presence classification, primarily on the peaks and valleys.

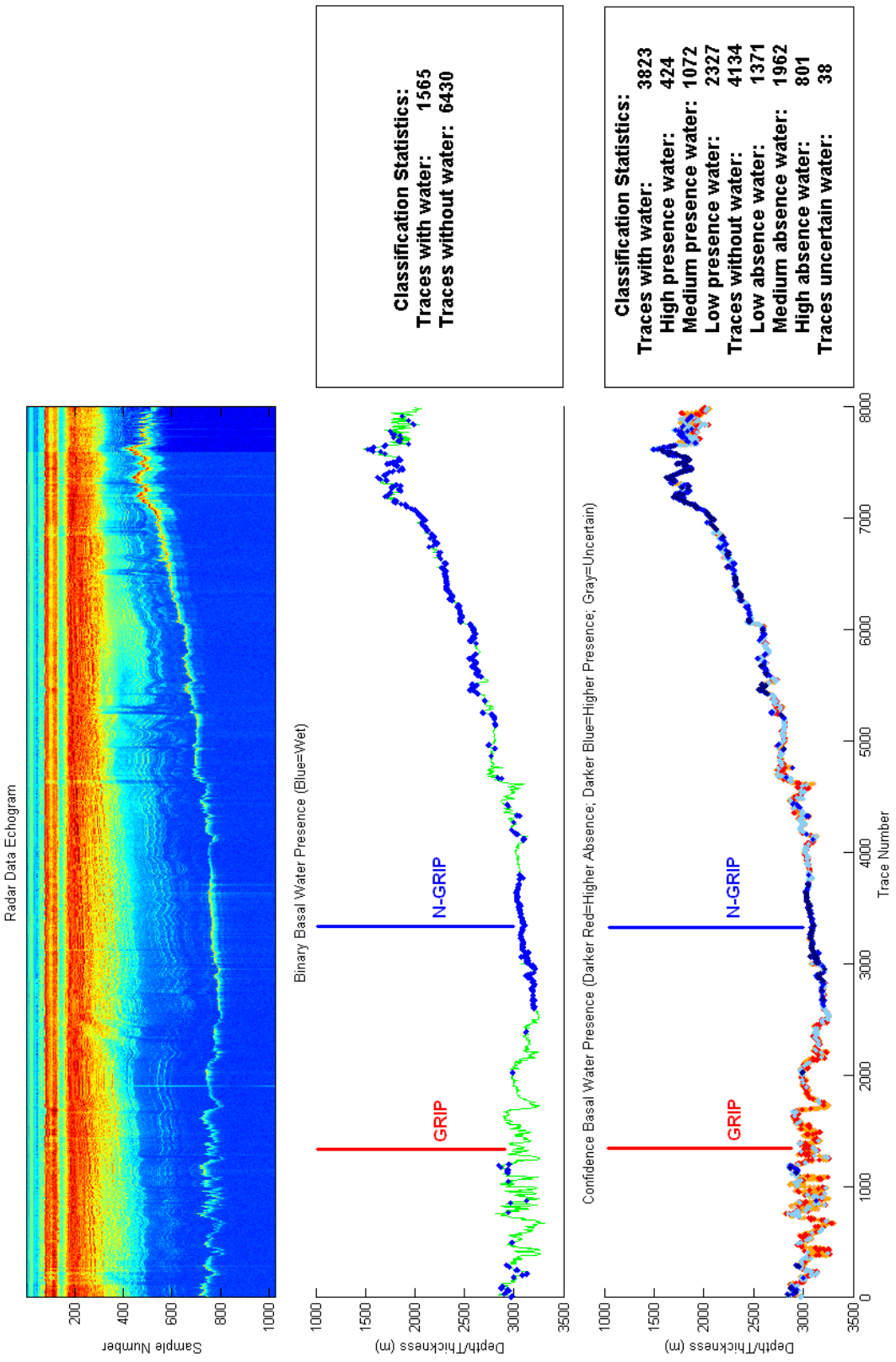

Figure 6.10: Comparison between binary and confidence-based water presence classifications for each radar trace of the 42-70 extended flight segment from May 14, 1999. GRIP and N-GRIP core locations are shown for reference.

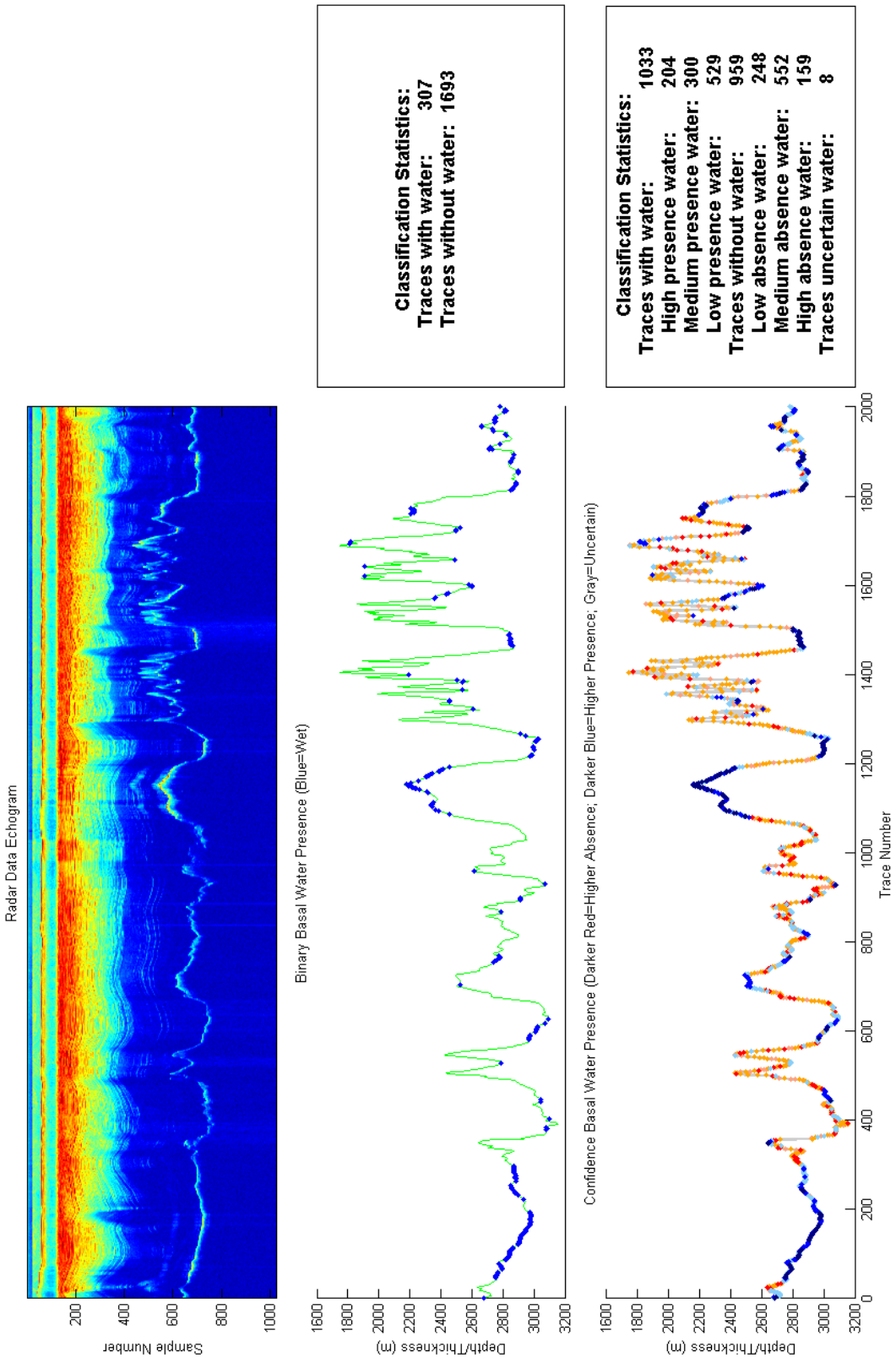

Figure 6.11: Comparison between binary and confidence-based water presence classifications for each radar trace of the 39-46 extended flight segment from May 25, 1999.

Table 6.3: 1999 ICARDS binary data set statistics.

Data Set Traces (Instances)				
Total	3901			
Wet	680			
Frozen	3221			
Measure	**Relative Intensity**	**Abruptness**	**Coherence**	**Ice Thickness**
Minimum	-34.345503	0.045497	0.000733	2812.797059
Maximum	-0.1621700	0.411527	0.833055	3259.922497
Average	-18.078544	0.233150	0.107303	3067.671329

6.5 1999 ICARDS Radar Data Set

Each trace is marked as being wet or frozen using the calculated abruptness and relative intensity for each trace. A trace, t, is determined to contain water at the bedrock if:

$$(RelativeIntensity(t) \geq -15.0) \;\; and \;\; (Abruptness(t) \geq 0.27)$$

The binary (wet/frozen) 1999 ICARDS data set are then generated by saving the following information for each trace:

- Latitude/Longitude
- Relative intensity (negative dB)
- Abruptness ([0,1])
- Coherence ([0,1])
- Ice thickness (m)
- Subglacial state: wet (1), frozen (0)

Ice thickness is included to allow the learning algorithms to take into account depth when fitting a model to the data for classification. All three primary attributes are used, plus others, for learning a model connecting the radar data and subglacial water presence. The original method's paper reports results using only relative intensity and abruptness attributes, with some analysis using coherence [91].

Each trace of the radar survey represents an instance which can be used for training or testing a classifier. The above information represents the attributes for each instance. Finally, the method's output represents the instance's class, used as the target for training a classifier or for computing classification accuracy during testing. This represents a binary classification problem. This data set's statistics are presented in Table 6.3.

Using the binary water estimation method, water presence has been classified for the entire May 1999 Greenland radar survey, as shown in Figure 6.12. The icons marking water presence over the flight lines are sized for visualization purposes, and should not be interpreted as containing water in the entire true area beneath the icon. Areas where flight lines overlap exhibit similar water presence results. Portions of flight segments where the aircraft banked for turns are less reliable in terms of wet versus frozen, as the returns the

radar receives are off-nadir. Figures 6.13 and 6.14 offer views of the binary water presence classification for May 14 and May 25, 1999, as well as the May 14 segment surrounding the GRIP and N-GRIP core sites. These are the corresponding spatial views for the binary classification results shown in previous figures.

6.6 2007 MCRDS Radar Data Set

As the data and method were studied and developed in detail, the May 1999 ICARDS radar data set represents ground truth for further studies. This portion of the 1999 data set encompasses some of the limited pure ground truth available in Greenland. This data set is used in our work to train teams of classifiers to learn a model between radar surveys acquired with different radars over the Greenland ice sheet.

The corresponding flight segment from the 2007 MCRDS Greenland radar survey was flown on September 17, 2007. In particular, the D-K data segments form a flight path that overlaps the same flight path flown over GRIP and N-GRIP in 1999. We can thus use the method's output from 1999 data at those locations to provide the target class (wet or frozen) for training using the 2007 data. Figure 6.15 shows the September 17, 2007 flight line with respect to the GRIP and N-GRIP core sites. Figure 6.16 shows the geographically closest portion of the 2007 flight line to that from 1999.

The 2007 MCRDS radar data are processed using the same method as the 1999 data, with the following minor differences:

- Use of seven traces on each side of the current trace (15 total traces) for incoherent integration and computing coherence (as opposed to 1 on each side, 3 total traces, for the 1999 data). This is due to the higher resolution, more dense radar data in the 2007 survey from advanced radar and computing technologies.
- Differences between the oscillators used in the 2007 radar, which affects the coherence attribute. This is due to improved technology.
- Alignment of the lower boundary of the reflection distributions of the 2007 data is performed using reflection intensity bounds from the MCRDS data.

Figure 6.17 plots the relative intensity distribution over the September 17 flight segment from the 2007 Greenland MCRDS radar survey containing wet and frozen bed characteristics (survey data over the N-GRIP and GRIP core sites). This is the flight segment which is nearly identical to that flown over GRIP and N-GRIP in 1999. Again, a bimodal distribution (one hypothetical Gaussian distribution for the wet intensities and another for the frozen intensities) is observed. However, compared to the 1999 ICARDS relative intensity distribution, it is more difficult to distinguish between wet and frozen basal states, as the distribution exhibits more overlap.

The 2007 MCRDS data sets are created using attributes computed from the MCRDS radar data and the geographically closest location's class (wet or frozen) from the 1999

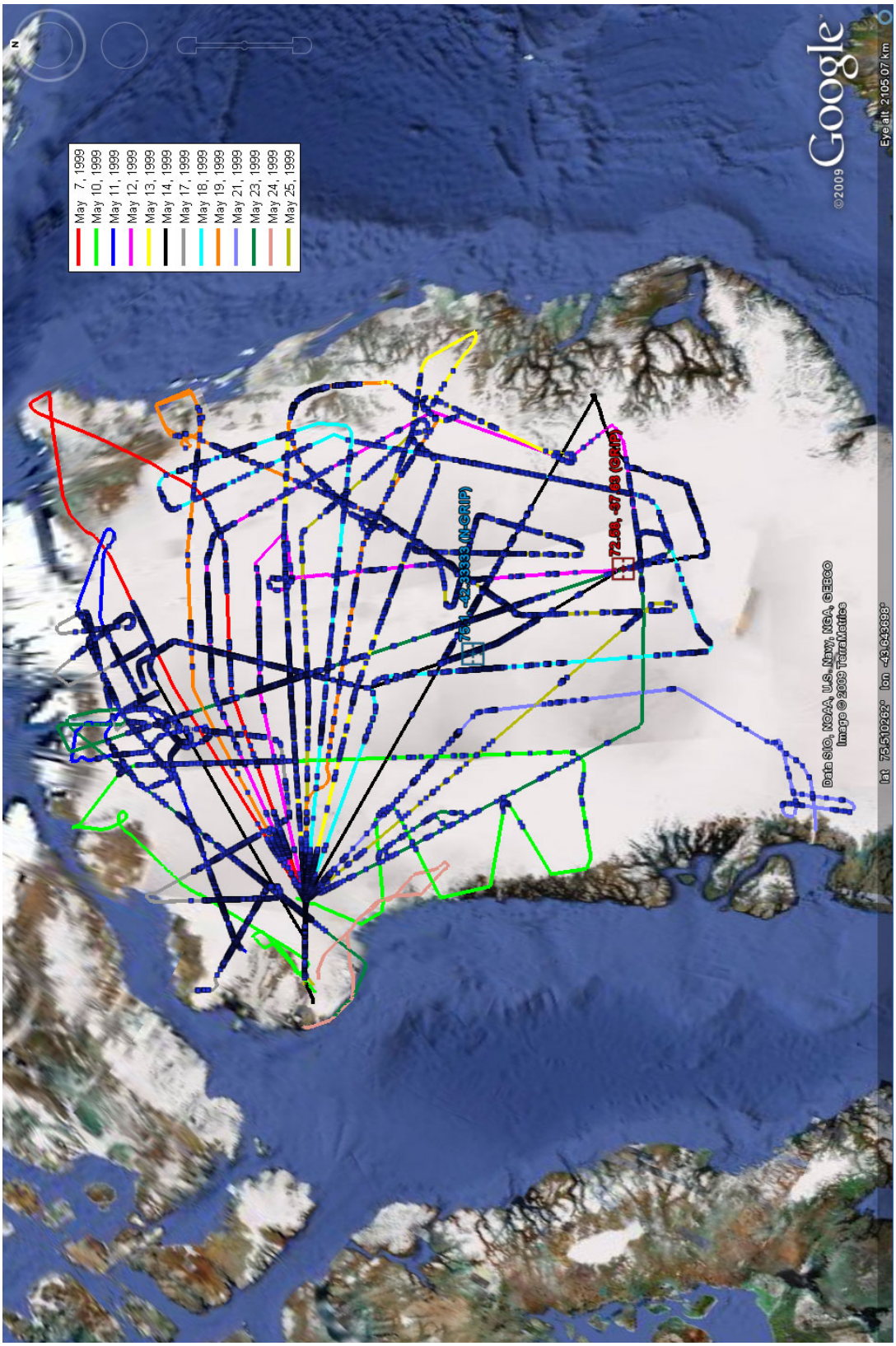

Figure 6.12: Binary water presence for all May 1999 Greenland flight lines. Water presence is marked with icons along the flight path. Absence of an icon represents a frozen bedrock interface.

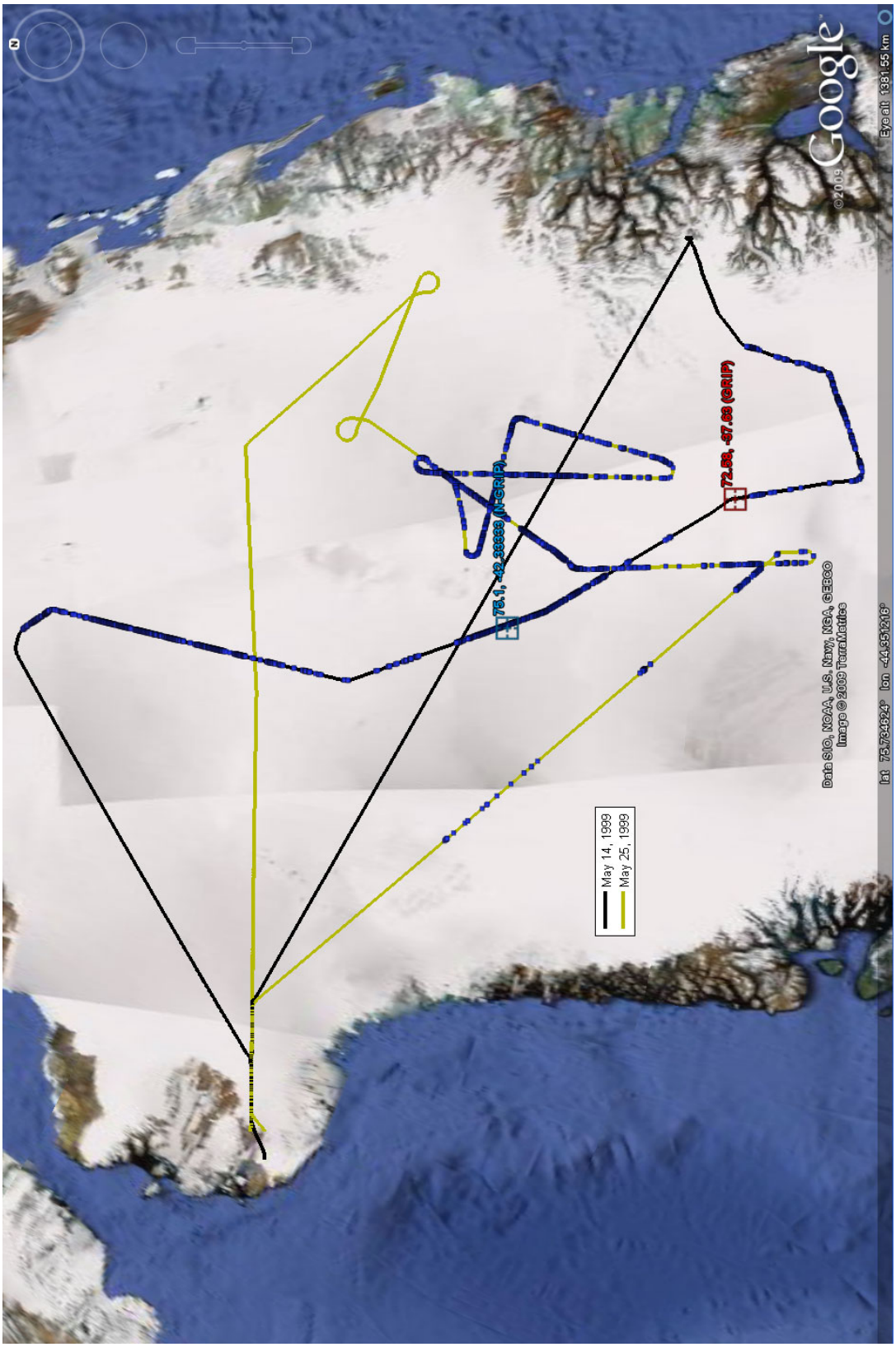

Figure 6.13: Binary water presence for the May 14, 1999 and May 25, 1999 Greenland flights. Water presence is marked with icons along the flight path. Absence of an icon represents a frozen bedrock interface.

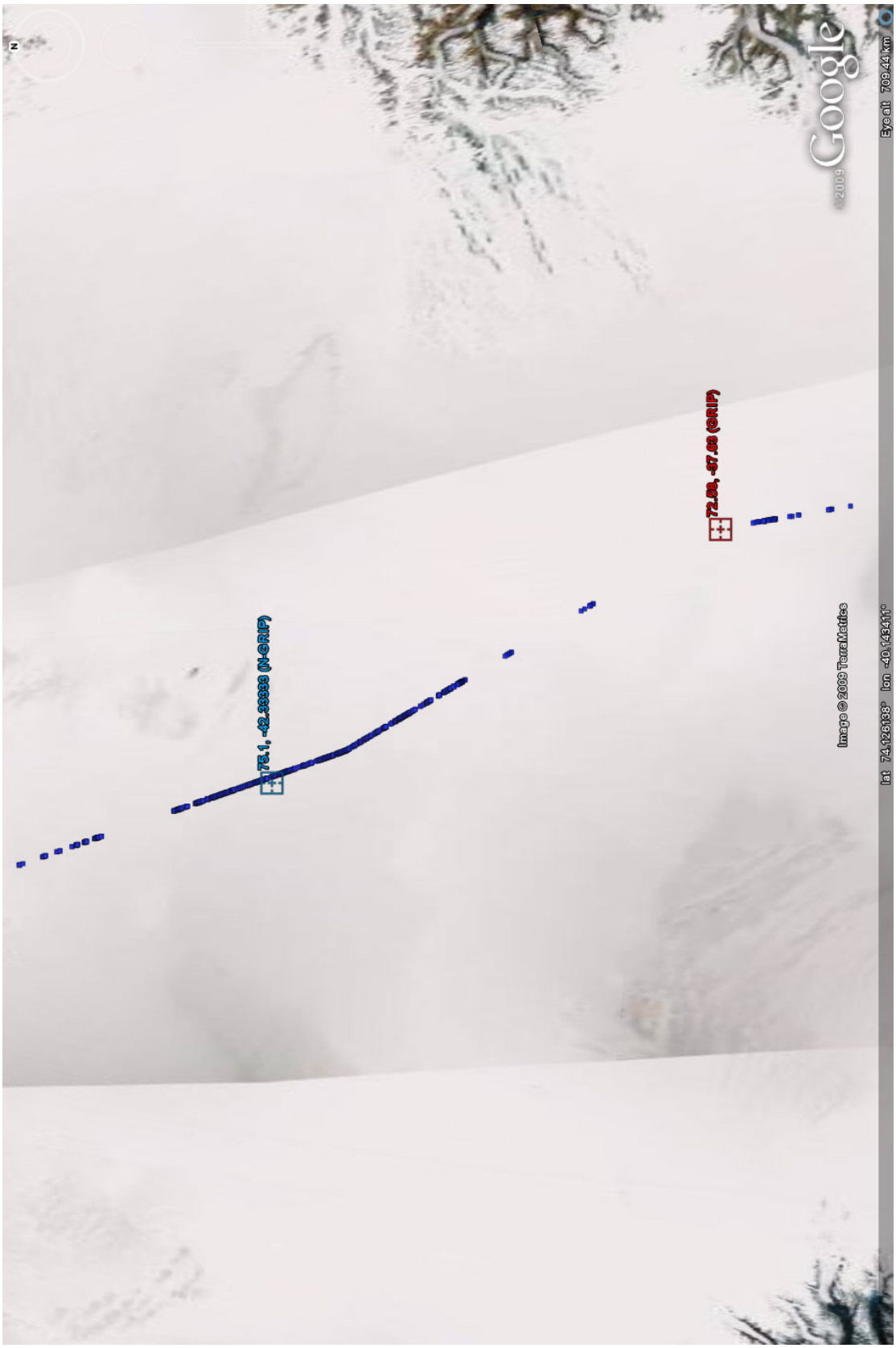

Figure 6.14: Binary water presence for the extended segment of the May 14, 1999 flight surrounding the GRIP and N-GRIP core sites. Icons mark the presence of water.

Figure 6.15: September 17, 2007 Greenland MCRDS radar survey flight line, with the GRIP and N-GRIP core sites marked for reference.

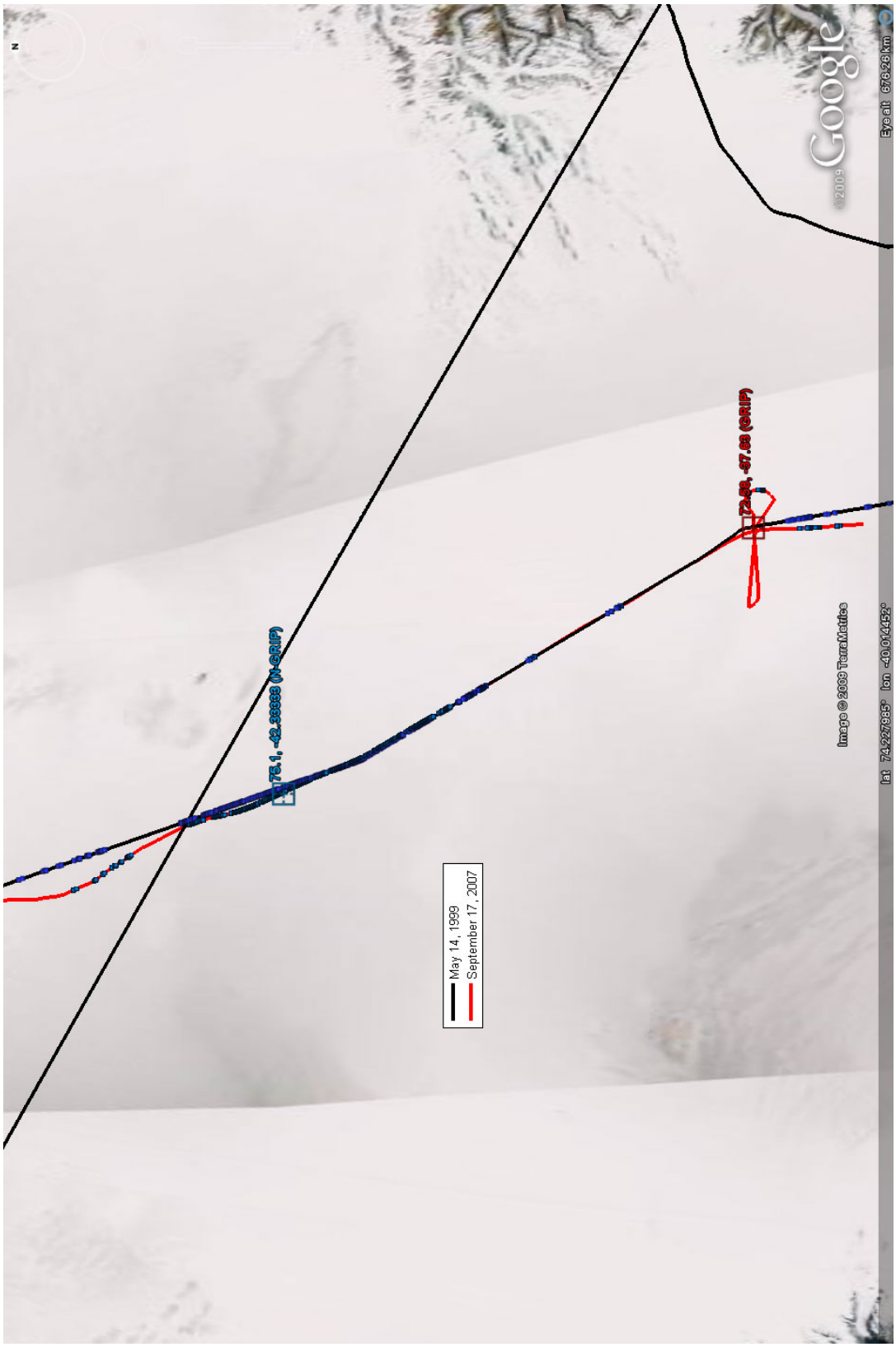

Figure 6.16: The extended flight segments from 1999 and 2007 which are closest to one another. Water presence from the 1999 survey line is used for each radar trace of the 2007 survey.

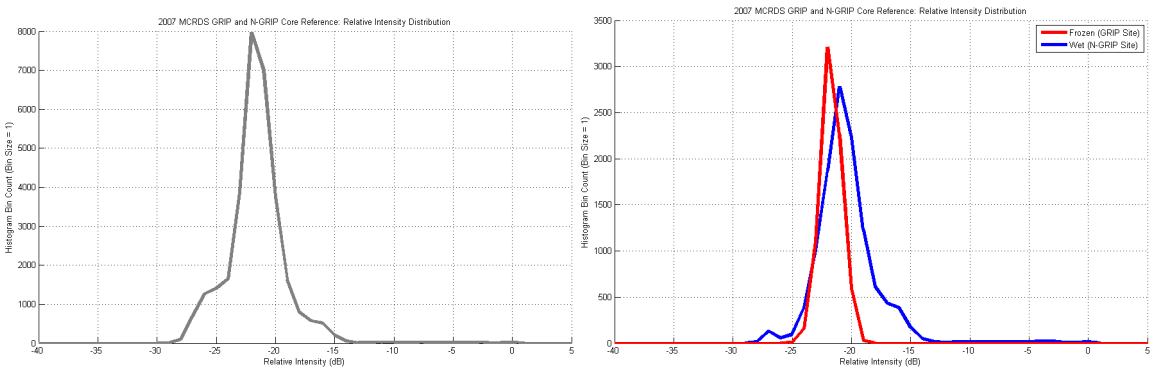

Figure 6.17: Relative basal reflection intensity distributions for the entire September 17, 2007 GRIP and N-GRIP extended flight segment (left), and for those traces near GRIP and N-GRIP sites (right).

ICARDS model. This assumes regional similarity in subglacial state, and the general stability of this state over the time difference between data acquisition between these systems. Thus, the ground truth water presence output from the published method is used to label the 2007 MCRDS data for each trace's water presence. The reason that this is possible, even with heterogeneous radars, is that attributes of the 2007 processed data are normalized independent of the others, and independent of the 1999 data. Even though the reflection intensity, abruptness, and coherence magnitudes are different from radar to radar and with different settings, a common classification target and normalization facilitate model transfer from one year's survey to another.

The 2007 binary and confidence-based data sets are comprised of the same attributes as the 1999 ICARDS data set. Figure 6.18 presents the attributes and binary basal water presence classification for the extended flight segment D-K from the September 17, 2007 flight. Along with the echogram for this segment, the bedrock reflection window is shown, on which relative intensity calculations are based. Coherence values above 0.3 represent smoother surfaces, which are utilized as additional signal information for classification. The method classifies, out of a total of 45477 traces for this segment, that 7819 are wet and 37658 are frozen. The GRIP and N-GRIP sites are classified as frozen and wet, respectively, just as in the 1999 data. Again, water tends to be present in continuous regions. Figure 6.19 shows the water presence (marked with icons) in relation to the location on the ice sheet.

Figure 6.20 compares the binary and confidence-based basal water presence classification models for the extended flight segment D-K from the September 17, 2007 flight. As the confidence-based classification is an extension of the binary classification model, GRIP and N-GRIP are still dominantly classified as frozen and wet, respectively. All shades of blue indicate water presence, with darker shades representing higher confidence (values further away from the threshold). All shades of red indicate lack of water presence, with darker shades representing higher confidence. From a total of 45477 traces, 17370 were classified

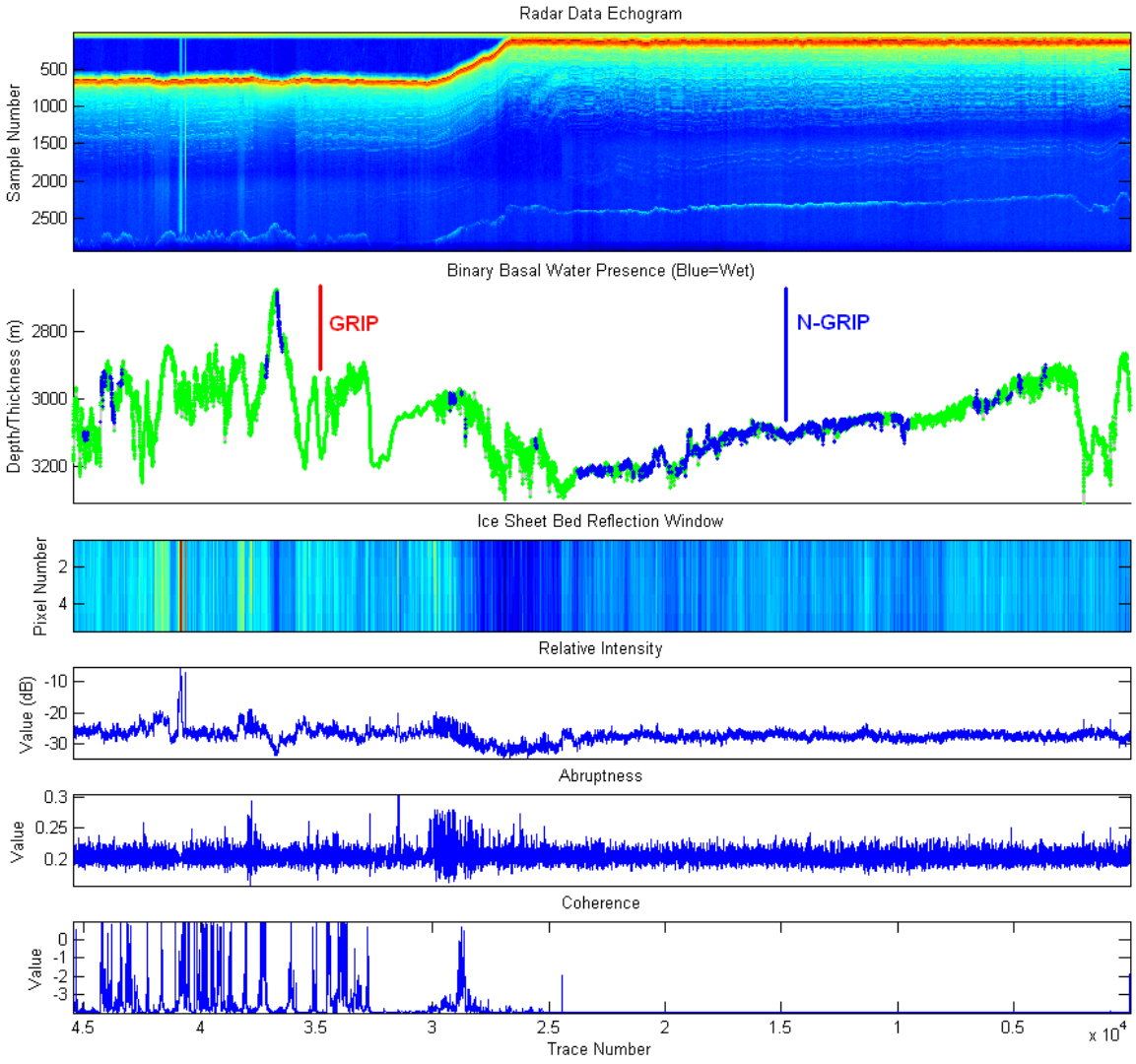

Figure 6.18: Attributes and binary water presence classifications for each radar trace of the D-K extended flight segment from September 17, 2007. GRIP and N-GRIP core locations are shown for reference.

Figure 6.19: Binary water presence classifications for each radar trace of the D-K extended flight segment from September 17, 2007. Water presence is marked with icons along the flight path.

Table 6.4: 2007 MCRDS binary data set statistics.

Training Set				
Total	30318			
Wet	5213			
Frozen	25105			
Testing Set				
Total	15159			
Wet	2606			
Frozen	12553			
Measure	**Relative Intensity**	**Abruptness**	**Coherence**	**Ice Thickness**
Minimum	-31.399587	0.158905	-3.997464	2675.122382
Maximum	-2.4782150	0.304487	0.9984250	3306.220237
Average	-24.078880	0.205357	-3.730076	3062.460763

as various confidence levels of water presence (2444 *High*, 5562 *Medium*, 9364 *Low*); 27969 were classified as various confidence levels of water absence (5086 *High*, 14348 *Medium*, 8535 *Low*); and 138 were considered *Uncertain* due to how close both relative intensity and abruptness were to the thresholds. In the areas where binary classification reports no water presence, the modifications report that some samples have *Low* confidence of water presence. Due to these localized cases, the confidence-based method more than doubles the number of traces marked as containing water.

Now that binary and confidence-based data sets have been created for the 1999 and 2007 flights, we can compare what they each determine to be wet and frozen. Figure 6.21 presents the comparison of binary water presence classification for the geographically closest 1999 and 2007 flight segments. The ice sheet's bottom topography is very similar between the two, except where the 2007 line gets further away. The 2007 survey has much higher data resolution. This figure demonstrates how the water presence from the 1999 binary model was used to mark the 2007 data traces for our classification study. Similarly, Figure 6.22 presents the comparison of confidence-based water presence classification for the geographically closest 1999 and 2007 flight segments.

Once the binary data set has been produced, including wet/frozen output for each trace via referencing the 1999 ICARDS method output, each attribute is individually normalized to be in the range [0,1]. Training and testing sets are created by randomly splitting the data into two-thirds training and one-third testing portions. The statistics of this data set and its training and testing portions are presented in Table 6.4.

The 2007 data set is provided to teams of classification algorithms to create a combined model of the radar data against the determined subglacial state for all traces. Classification results offer probability estimates for each trace's inferred state (wet, frozen). Probability estimates from multiple learning algorithms, using different model fitting techniques, provide further information with which to make a determination on the extent of the ice sheet's subglacial water presence.

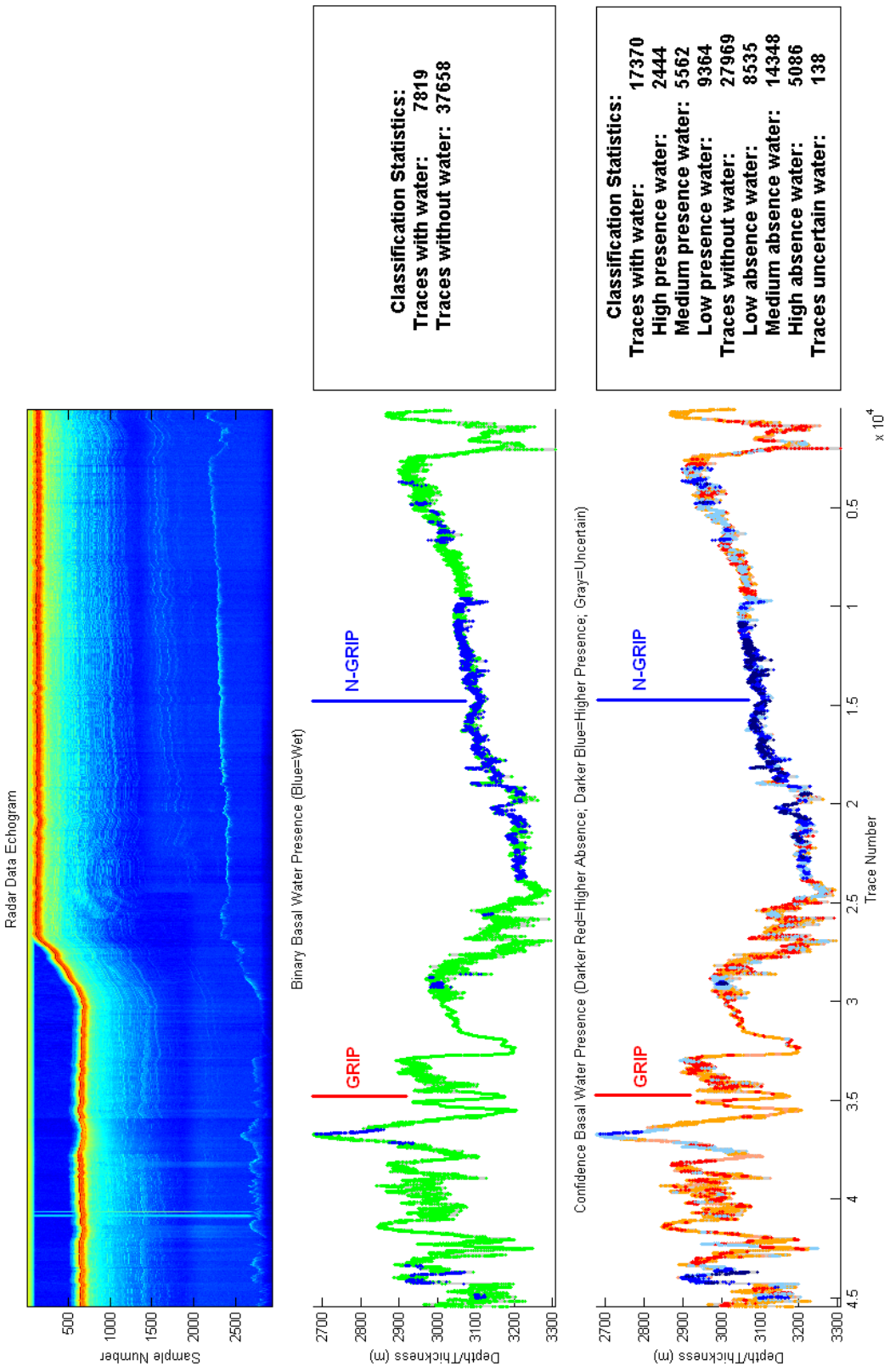

Figure 6.20: Comparison between binary and confidence-based water presence classifications for each radar trace of the D-K extended flight segment from September 17, 2007. GRIP and N-GRIP core locations are shown for reference.

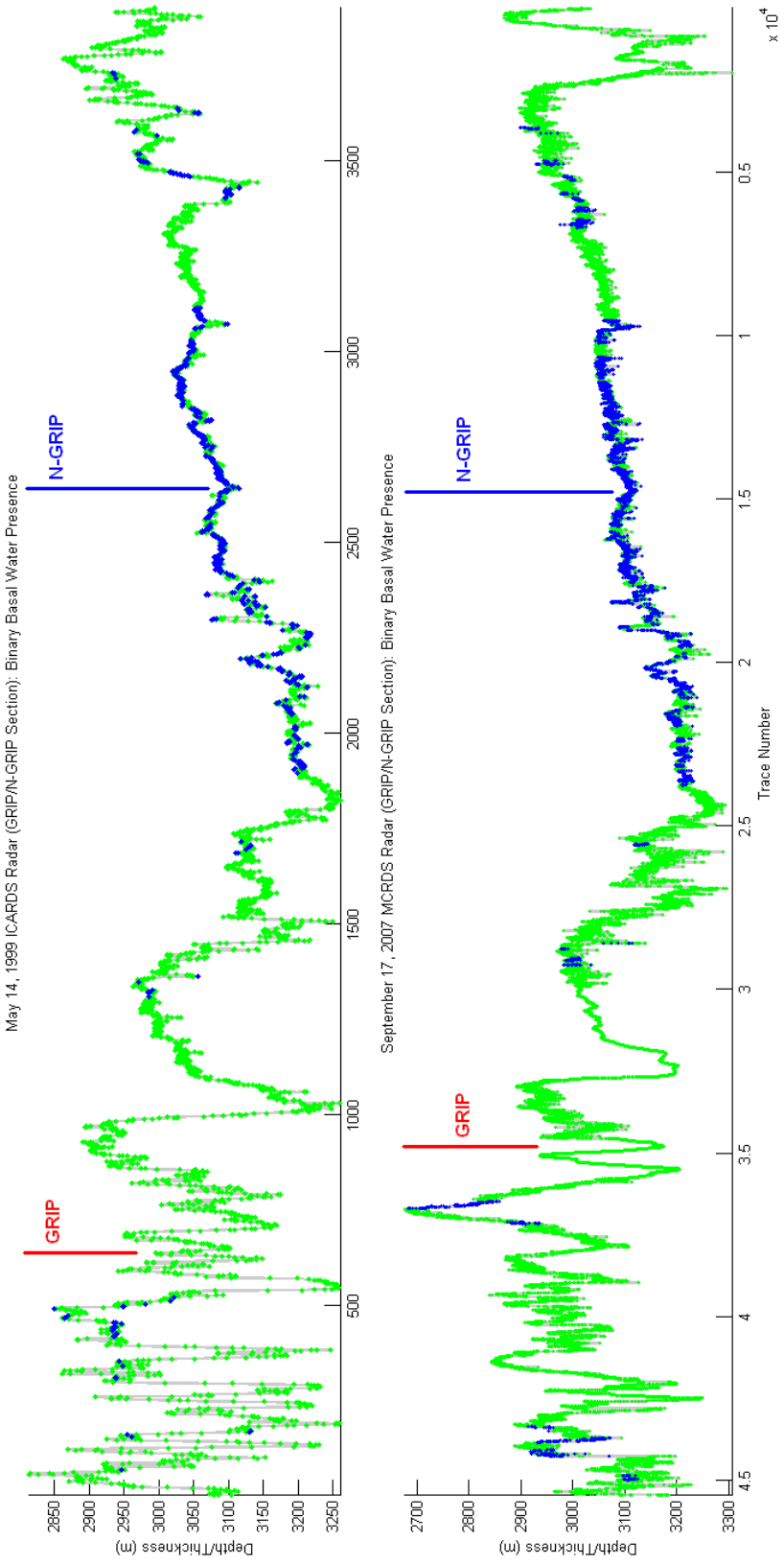

Figure 6.21: Comparison between binary water presence classifications for each radar trace of the 1999 and 2007 flight segments surrounding the GRIP and N-GRIP core sites.

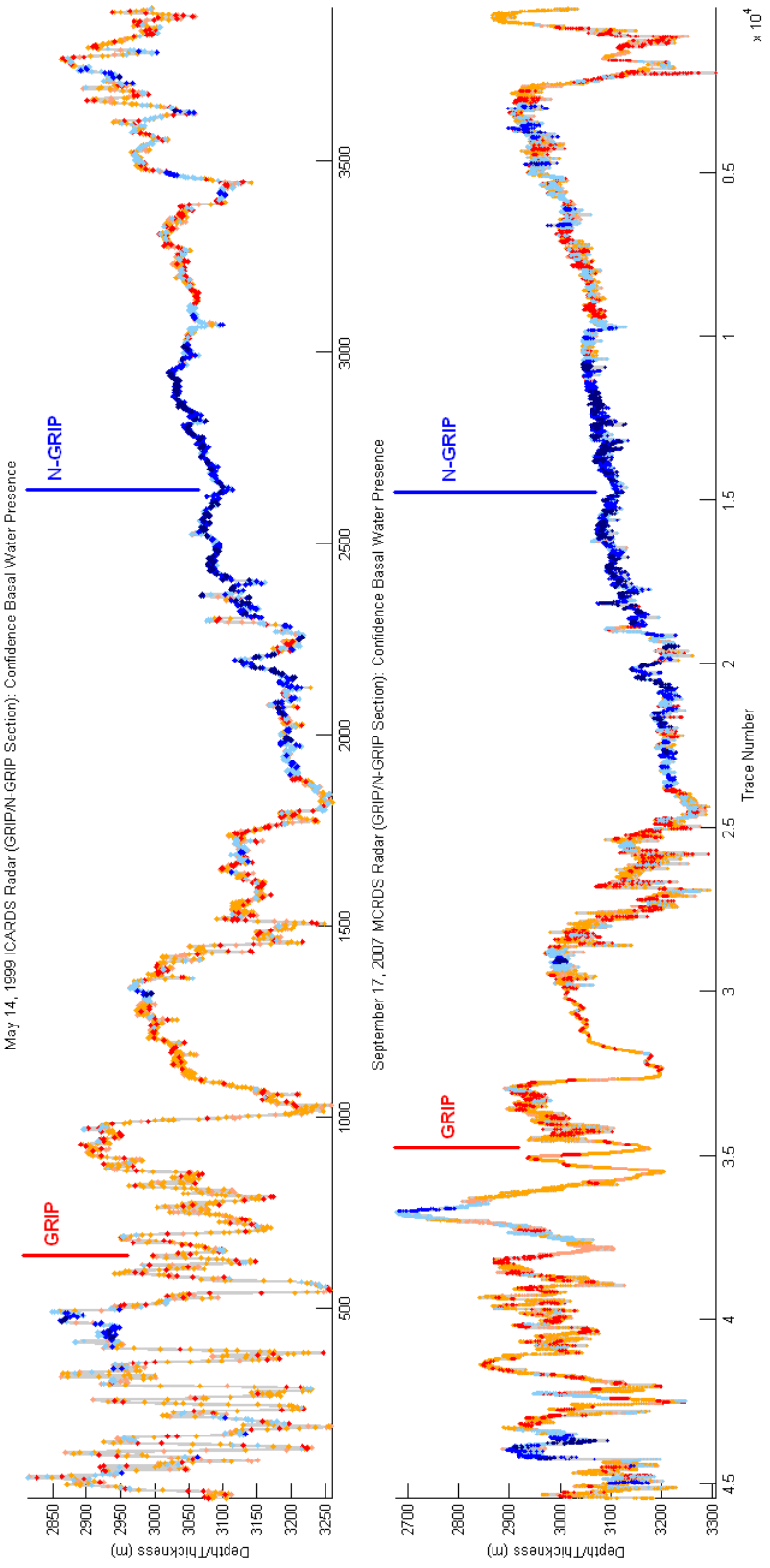

Figure 6.22: Comparison between confidence-based water presence classifications for each radar trace of the 1999 and 2007 flight segments surrounding the GRIP and N-GRIP core sites.

The Multi-Agent Collaborative Learning Architecture, together with the WEKA machine learning suite, were utilized to study aspects of team learning and collaboration. As discussed in Section 3.2, an accuracy-based tournament-style series of experiments were performed.

6.7 Experimental Results

This section presents the experimental results on various aspects of team learning and collaboration from the subglacial water presence classification study. These aspects are comprised of independent team-based studies that allow us to make conclusions and formulate heuristics for team learning with this data set. These results are combined with those from Chapters 4 and 5 to generate a set of novel, general heuristics for applying team-based learning to other domains.

The notation used in the below figures and analyses are abbreviations of learning algorithms, collaboration frequencies, and data distribution modes. The abbreviations for the 11 learning algorithms are: Naive Bayes (NB), Decision Tree (DT), Instance-Based KNN (IBK), Neural Network (NN), Logistic Regression (LGR), Radial Basis Function Network (RBF), K* (KST), Decision Table (DTB), RIPPER Rule Learner (JRP), PART Rule Learner (PRT), and Random Forest (RFT). When listing team compositions, the number of learners for each learning algorithm of the team is listed, separated by "+", followed by the collaboration frequency for that team. For example, "3ibk+2nn+2dt+1lgr (med)" represents a team of size eight composed of three IBK, two NN, two DT, and one LGR classifiers which collaborate five times (Medium) during learning. Collaboration frequencies are broken down into None (0 collaborations per experiment), Low (2), Medium (5), and High (8). Finally, Full and Independent (Indp) data learning distribution modes are also abbreviated as shown.

A total of 518 learning experiments were performed, including the variation of team size, composition, collaboration frequency, and learning mode. Of these, 259 were Full mode team learning experiments, and 259 were Indp mode team learning experiments. For each learning mode, experiments are broken down as follows:

- 11 size one, 76 size two, 96 size four, and 76 size eight teams
- 62 teams for each collaboration frequency (None, Low, Medium, and High)
- 71 homogeneous (including size one teams) and 188 heterogeneous teams

6.7.1 Decision Combination Method

Decision combination methods are used for combining classifications from multiple learning algorithms to formulate the team's collective decision for an instance requiring classification. In the following figures and tables, each count for a combination method represents

Table 6.5: Most successful five decision combination methods, based on combined testing accuracy, over all experiments for Full and Indp learning distribution modes.

Full Mode		Independent Mode	
Combination Method	**Win Count**	**Combination Method**	**Win Count**
Multiply	87	Multiply	69
Average	83	Average	68
Selective Multiply	81	Selective Weighted Vote	59
Selective Average	80	Selective Majority Vote	59
Weighted Vote	75	Weighted Vote	55

an experiment which resulted in that combination method providing the highest testing accuracy. Ties (in terms of combined decision testing accuracy from all learners) count as a win for each learner that is tied. The reported win percentages have been normalized, as there are varying numbers of teams of different sizes and there are more heterogeneous teams than homogeneous teams, due to heterogeneous teams generally offering higher classification accuracy.

Table 6.5 presents the most successful five decision combination methods over all Full and Indp mode team experiments for this data set. In general, team size dictates the combination method to choose. Regardless of learning mode, Multiply, Average, Weighted Vote, and their Selective variations are successful methods for the experiments. Selective Multiply and Selective Average work well for Full mode teams, whereas Selective Weighted Vote and Selective Majority Vote perform well for Indp mode teams. This table suggests that the Multiply method is generally superior for Full and Indp teams.

Figure 6.23 presents results for Full versus Indp mode teams as a function of team size. Selecting the combination method to optimizing performance depends on the team size: Average or Multiply for size two, Selective Average or Selective Multiply for size four, and Majority Vote or Weighted Vote for size eight teams. Max, Selective Max, and Select Best consistently perform poorly for all team sizes and learning modes. Average and Multiply do not perform consistently well for teams larger than two.

Figure 6.24, focusing on homogeneous versus heterogeneous teams, shows that Max, Average, and Multiply combination methods perform generally well for homogeneous team compositions. Heterogeneous, Full mode teams should utilize Selective Average or Selective Multiply, whereas heterogeneous, Indp mode teams should utilize Selective Majority Vote or Selective Weighted Vote for combining decisions.

Combination method results for Full and Indp mode teams as a function of collaboration frequency are presented in Figure 6.25. For teams which do not collaborate, the Max, Multiply, or Selective Multiply methods should be selected. Collaborating with Low frequency yields best performance using Average and Multiply. Medium frequency collaborative teams should select Majority Vote or Weighted Vote; however, the subset of Indp mode teams benefits most from the Multiply combination method. High frequency collaboration achieves best performance from Majority Vote, Weighted Vote, and their

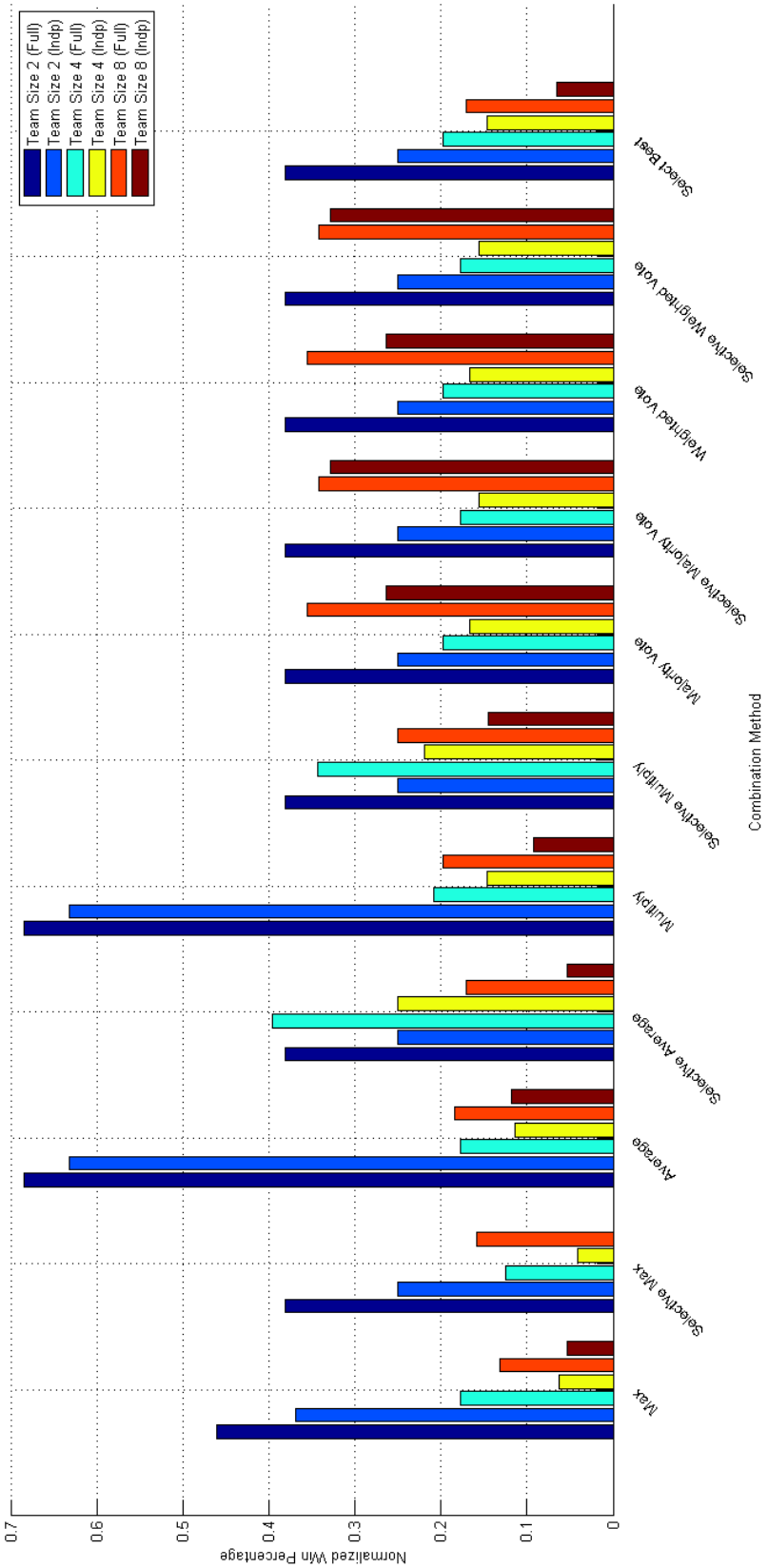

Figure 6.23: Analysis of the effect of team size on combination method win percentage for Full versus Indp learning modes.

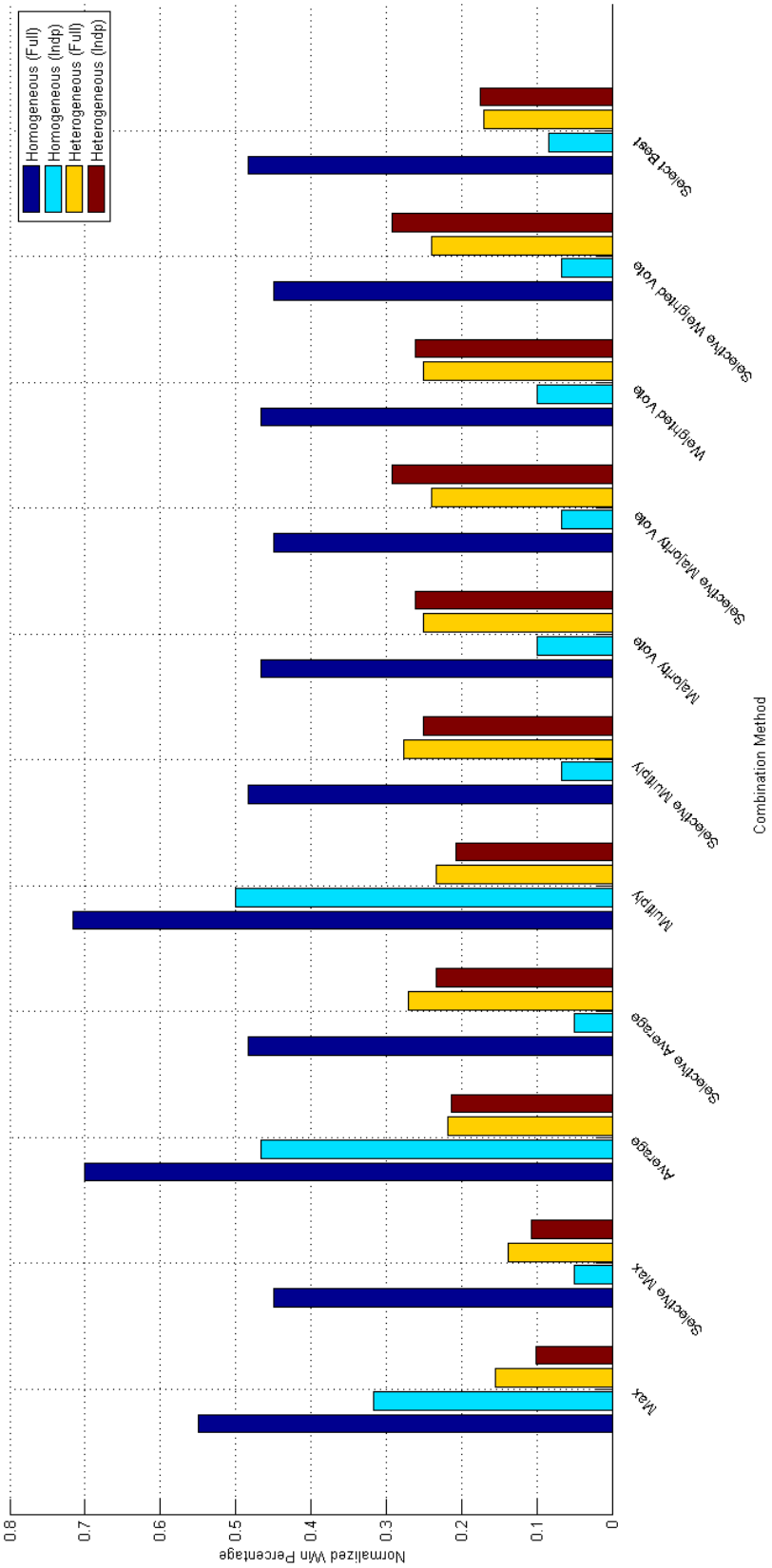

Figure 6.24: Analysis of the effect of team heterogeneity on combination method win percentage for Full versus Indp learning modes.

Selective variations. Non-collaborative teams experience little benefit from these particular combination methods. Specific combination methods appear to be geared more toward Indp mode team classification (non-overlapping, distributed data set modeling); whereas Full mode teams experience a more distributed success from all combination methods that were studied. Overall, averaging and multiplying class probabilities perform consistently well for teams of collaborating learners for this application.

6.7.2 Full versus Independent Data Learning Distribution

The study of learning distribution modes was aimed at determining if each learner required the full training data set for training, or if it could be distributed among the learners in a non-overlapping fashion. More importantly, the goal was to investigate collaboration's impact on testing accuracy, given these two modes. It was envisioned that Full mode teams would perform better, and that Indp mode teams would require higher levels of collaboration to achieve high classification accuracy. It is also understood that distributing the learning load for training offers a speedup in terms of the time required for teams to model the data, as each learner will be responsible for fewer training instances. The following tables and figures show our findings.

Figure 6.26 presents the minimum, average, and maximum testing accuracy for Full and Indp mode teams by team size. The worst performing Full and Indp teams suffer a substantial decrease in testing accuracy. The poor Full mode teams recover when scaling up from size four to eight; however, Indp mode teams can continue to decrease with an increase in team size. These results illustrate that certain pairings of learning algorithms perform very poor in comparison to the majority of the teams. On average, Full mode teams outperform Indp mode teams by approximately 3%. In regards to the best teams, performance levels off after teams of size two. Thus, for this application, some learning algorithms perform very poorly when paired together, but increasing the number of learners in the team can help stabilize team accuracy. Team composition does make a difference in team accuracy, as each learning algorithm models the data in different manners. Combining these different models in different ways can increase accuracy.

Full mode teams perform better than Indp mode teams on average, with the difference in performance growing slightly with team size. A team of size eight produced the highest team classification accuracy over all Full mode teams, whereas teams of size two and four produced the highest team classification accuracies for Indp mode teams. This introduces a tradeoff between team size and classification accuracy between size four and eight teams. Team size inherently involves a difference in training and testing time. Some of the learning algorithms require more resources and time to perform their modeling effort. For example, decision trees do not require the memory space and time that neural networks do. The K* algorithm is the most memory-intensive algorithm utilized in our experiments. Thus, if similar accuracy can be achieved with a smaller team size and the application permits, it

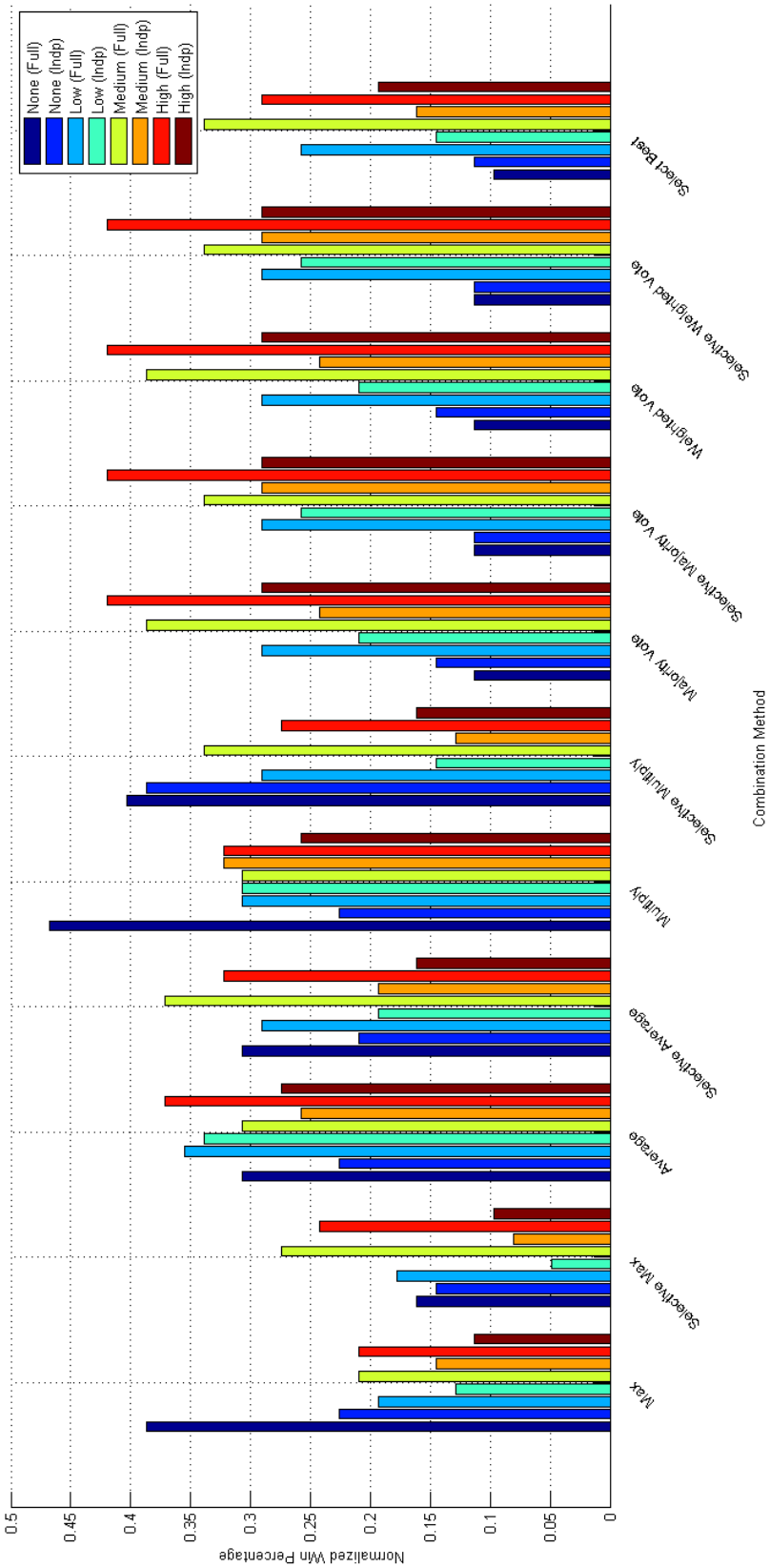

Figure 6.25: Analysis of the effect of collaboration frequency on combination method win percentage for Full versus Indp learning modes.

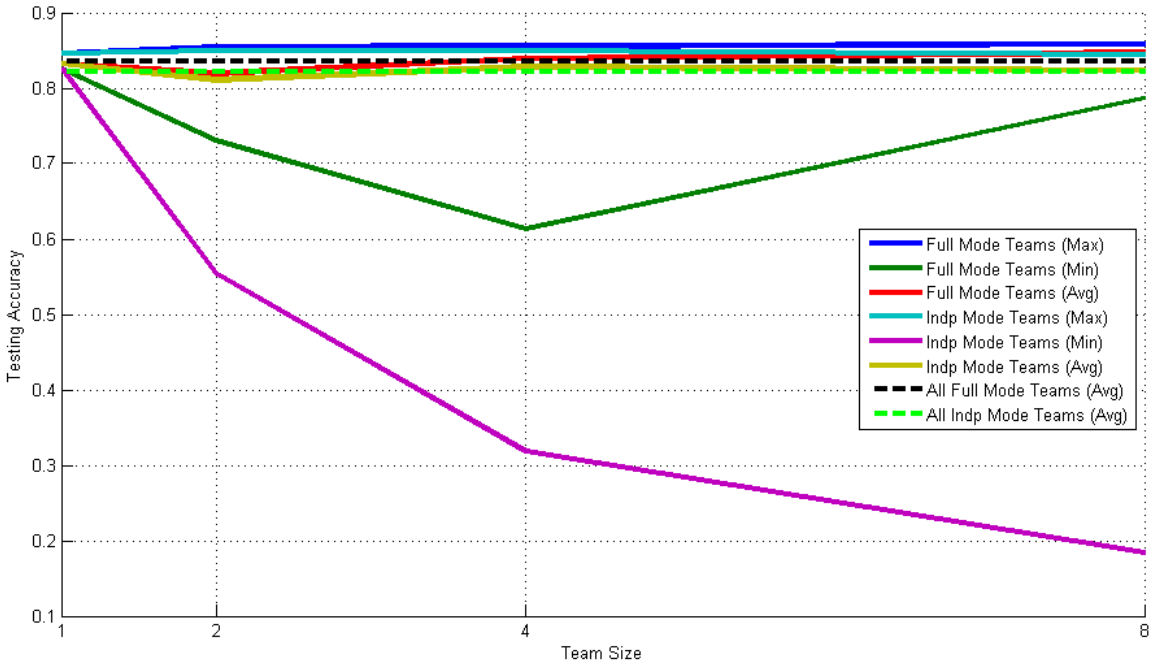

Figure 6.26: Analysis of the effect of team size on testing accuracy for Full versus Indp mode teams. The average testing accuracies over all Full and Indp teams are shown for reference.

may not be efficient for a minimal amount of accuracy gain.

We can further investigate team compositions that are generally successful and not. Table 6.6 shows the best 15 teams for Full and Indp modes, accompanied by the associated collaboration frequency and team size. For both Full and Indp mode teams, multiple team compositions can produce the same testing accuracy. For example, the Random Forest classifier is the base for each of the best Full mode teams. These teams only differ in the last one or two learners in the team. In these cases, the last learner may not be necessary, as the other seven may have covered all of its correctly classified instances. Several Indp mode teams achieve high accuracy, specifically through the use of collaboration. The best Full mode teams largely do not utilize collaboration. These results match our hypothesis of collaboration having a greater impact on Indp mode teams.

Two primary trends are apparent when analyzing the best performing Full mode teams: (1) heterogeneous teams consistently produce high team classification accuracy, and (2) larger teams (dominantly size eight teams in the top 15) consistently perform better. Conversely, Indp mode teams exhibit a coupling between smaller team size and higher accuracy. Homogeneous teams also appear more in the ranking of best 15 Indp teams, demonstrating that the use of Indp learning distribution is only beneficial up to teams of size four in this case. As teams get larger, the data set becomes more distributed and the learners are given smaller portions of the data for training. Lastly, the best Full mode team performed ∼1% better than the best Indp mode team. Due to the tournament-style

Table 6.6: Overall best performing 15 Full and Indp mode teams, based on combined testing accuracy.

Full Mode Team	Size	Accuracy	Indp Mode Team	Size	Accuracy
8rft (low)	8	0.857774	4rft (high)	4	0.849594
7rft+1dt (none)	8	0.857181	4rft (low)	4	0.849396
7rft+1dtb (none)	8	0.856719	2rft (med)	2	0.849330
7rft+1kst (none)	8	0.856719	2rft (high)	2	0.848671
8rft (none)	8	0.856521	3rft+1kst (med)	4	0.848143
8rft (high)	8	0.856521	2rft (low)	2	0.848011
6rft+2nn (none)	8	0.856323	4rft (none)	4	0.847615
5rft+2nn+1dtb (none)	8	0.856191	2nn+2rft (med)	4	0.847154
7rft+1kst (med)	8	0.856191	3rft+1kst (low)	4	0.847154
7rft+1dt (low)	8	0.856059	3rft+1dt (none)	4	0.847022
5rft+2nn+1dt (none)	8	0.855993	3rft+1dtb (none)	4	0.847022
6rft+1dt+1kst (none)	8	0.855861	3rft+1kst (none)	4	0.847022
6rft+1dtb+1dt (none)	8	0.855861	3rft+1nn (none)	4	0.847022
6rft+1dtb+1kst (none)	8	0.855861	2rft (none)	2	0.846758
6rft+2dt (none)	8	0.855861	3rft+1kst (high)	4	0.846758

experimentation process, it is also observed that the same small set of learning algorithms are highly used for both Full and Indp mode teams.

Table 6.7 presents the worst performing 15 teams out of all experiments. Smaller team size is indicated as a prominent trend for the worst performing Full and Indp mode teams. Many of the worst performing teams collaborate with Medium and High frequency. This does not necessarily mean that collaboration is bad, but rather that these teams and classifiers do not collaborate well (learn examples from one another during the learning process). It is shown later in this chapter that collaboration does have its advantages for specific teams and team sizes. The majority of poor-performing teams here are combinations of poorly performing individual classifiers, or homogeneous teams of the same poor individual classifier on this data set. There is a clear separation between those learning algorithms that offer high accuracy for this data set, and those that do not. Similarly, it can be concluded that teams of strong learners are outperforming collective decisions of multiple weak learners. The worst Full mode teams outperform the worst Indp mode teams by up to 43% in the worst case. For this application, combinations of specific algorithms perform worse than individual classifiers, demonstrating that team composition governs testing accuracy.

Figure 6.27 presents a measure of improvement of using Full mode learning over Indp mode learning as a function of team size. This measure is calculated by dividing the average testing accuracy of Full mode teams by the average testing accuracy of Indp mode teams. Thus, a value above 1.0 represents Full mode teams performing better on average. Full mode teams perform similar to Indp mode teams on average over all the experiments. The increase in potential Full/Indp performance is coupled with team size, which grows linearly as team size doubles. There are cases where Indp mode teams perform slightly better than

Table 6.7: Overall worst performing 15 Full and Indp mode teams, based on combined testing accuracy.

Full Mode Team	Size	Accuracy	Indp Mode Team	Size	Accuracy
4nn (high)	4	0.613629	8dtb (high)	8	0.185236
4nn (med)	4	0.613629	4dtb (high)	4	0.318425
1nn+1kst (high)	2	0.730787	2dtb (med)	2	0.553994
1nn+1kst (med)	2	0.731183	8dtb (low)	8	0.700508
4nn (low)	4	0.735735	1nn+1dtb (high)	2	0.711722
1kst+1dtb (med)	2	0.744970	2dtb (low)	2	0.746685
1dtb+1jrp (low)	2	0.753414	1kst+1dtb (med)	2	0.748664
2dtb (low)	2	0.756118	1kst+1dtb (high)	2	0.751897
2jrp (low)	2	0.762187	4nn (med)	4	0.753216
2nn (low)	2	0.764035	2nn (high)	2	0.761264
4dtb (low)	4	0.765618	4dtb (low)	4	0.761528
2nn (high)	2	0.770236	1dt+1nn (low)	2	0.762122
1kst+1dtb (high)	2	0.770895	2nn (low)	2	0.768586
2nn (med)	2	0.785804	1nn+1jrp (low)	2	0.777426
8kst (high)	8	0.786727	1dtb+1jrp (low)	2	0.778284

Full mode teams, most prominently being teams of size four. For size two and eight teams, the minimum improvement of Full over Indp is nearly identical.

From these experiments, it can be concluded that, on average, providing all learners the entire data set during training (Full mode) outperforms distributing the data set in a non-overlapping fashion over all learners (Indp mode). As previously noted, Indp mode teams do have the advantage of completing the training process much faster than Full mode teams, as the learners train on smaller portions of the data set as team size increases. This negatively affects peak performance for Indp mode teams; however, the magnitude of its affect is small.

6.7.3 Team Diversity: Homogeneous versus Heterogeneous Teams

Team diversity, or the composition of a team in terms of multiple heterogeneous learning algorithms, is central to a study of team learning dynamics. As different learning algorithms model the data and error differently, combining them can be beneficial for increasing classification accuracy. Teams composed of the same learning algorithm may not experience these advantages, especially when collaboration is involved. Here, we specifically focus on comparing results from homogeneous and heterogeneous teams. When discussing a team's diversity level, it represents the count of the number of unique learning algorithms in that team.

One method for analyzing team diversity is to observe its effect on testing accuracy as a function of the diversity level. Figure 6.28 presents this study for Full and Indp mode teams. No team existed which contained five, six, or seven unique learning algorithms, and the testing accuracies are assumed linear between diversity level four and eight. Full mode teams perform better (1-2%) for all diversity levels. Full mode teams also see a steady

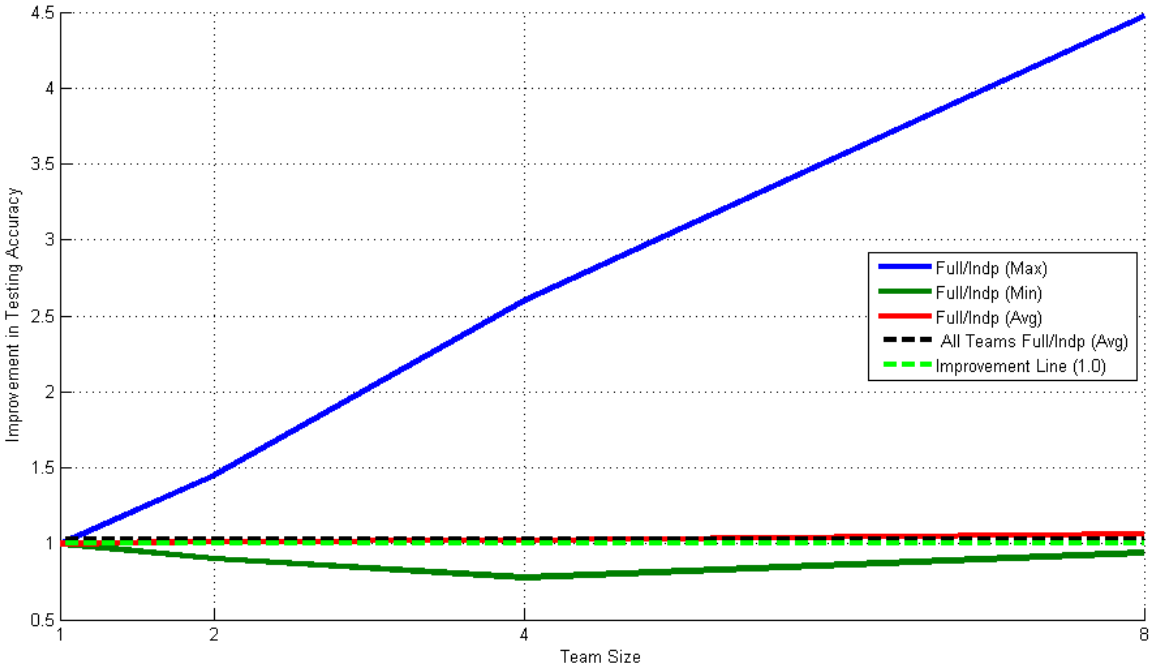

Figure 6.27: Investigation of the improvement/advantage of testing accuracy for Full mode teams over Indp mode teams. Values above 1.0 represent Full mode teams performing better than Indp mode teams in that case. The average testing accuracies over all Full and Indp teams are shown for reference.

increase in testing accuracy coupled with an increase in diversity up to a diversity level of three. This is a promising result, as it supports that the more heterogeneous a team is, the better its accuracy will or can be. Indp mode teams experience a similar increase in testing accuracy up to diversity level of three, but experience a decrease and leveling off of testing accuracy as diversity increases. This could be related to how a team should be composed of some supporting classifiers and some complementary classifiers to achieve its greatest potential. These results suggest that a diversity level of three (i.e., a team consisting of three heterogeneous learning algorithms, regardless of size) is desirable in this application, as it maximizes testing accuracy for both Full and Indp teams.

Table 6.8 offers a different view of team diversity by showing the number of teams of each diversity level. Diversity level counts in terms of testing accuracy are shown for all teams, the best 50 teams, and the worst 50 teams. The tournament-style experimentation process produced many teams of diversity levels 1, 2, and 3. By viewing the best 50 teams and their diversity levels, we can draw a conclusion about which diversity level achieves highest overall performance. Here, a diversity level of 2 is best for Full and Indp mode teams, with some support from a diversity level of 3. This supports results seen earlier, where the majority of the best performing Full mode teams contained eight total learners, and the best performing Indp mode teams contained four learners and were heterogeneous in composition. Diversity levels of 1 and 2 produced the lowest performance Full and Indp

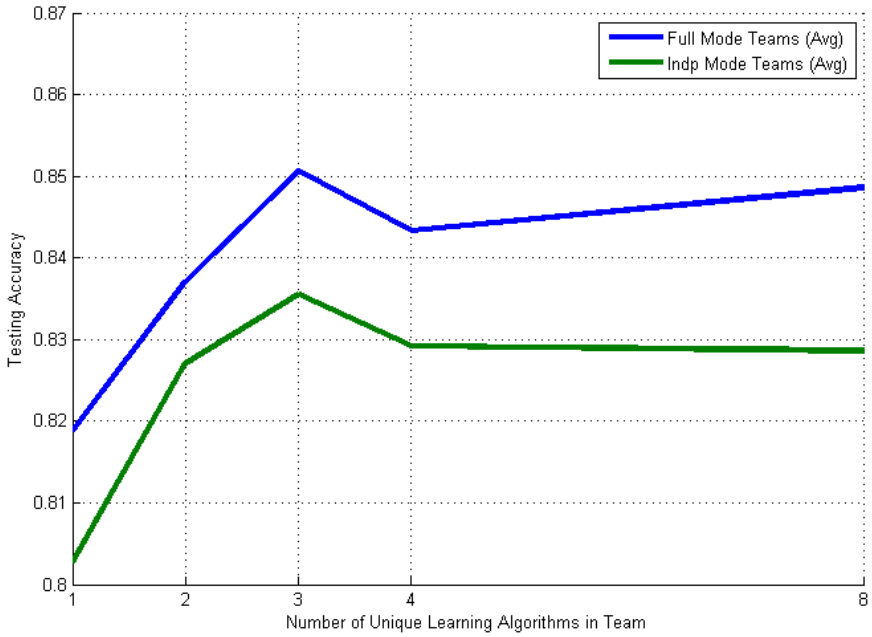

Figure 6.28: Average testing accuracy for Full and Indp mode teams as a function of team diversity.

mode teams. These results again stress the importance of team composition as a driving factor, as diversity level is not a clear discriminator of high accuracy.

Figures 6.29 and 6.30 show results comparing homogeneous and heterogeneous team performance as a function of team size for Full and Indp mode teams, respectively. Heterogeneous teams perform approximately 3% better on average for both Full and Indp modes. Homogeneous Full mode teams produce their highest average testing accuracy with size eight, and their lowest testing accuracy with size four. Homogeneous Indp mode teams experience a gradual decrease in accuracy when scaling up to size eight. The worst heterogeneous teams begin to outperform the homogeneous team average at sizes four and higher. These results again support that some teams of sizes two and four perform

Table 6.8: Number of teams of each diversity level, which is a count of the number of unique learning algorithms in a team, over all, best 50, and worst 50 teams. No team contained five, six, or seven unique learning algorithms.

	All Teams		Best 50 Teams		Worst 50 Teams	
Diversity	**Full**	**Indp**	**Full**	**Indp**	**Full**	**Indp**
1	71	71	11	10	25	19
2	112	112	27	30	24	26
3	48	48	12	9	0	0
4	20	20	0	1	1	4
5	NA	NA	NA	NA	NA	NA
6	NA	NA	NA	NA	NA	NA
7	NA	NA	NA	NA	NA	NA
8	4	4	0	0	0	1

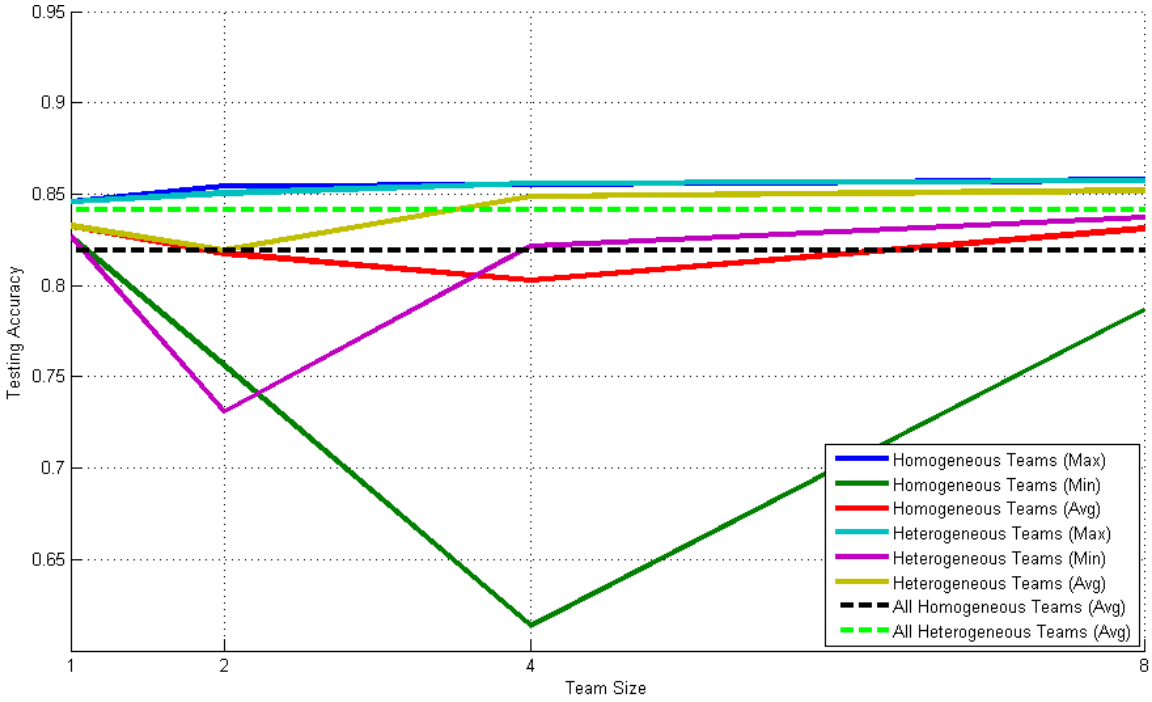

Figure 6.29: Homogeneous versus heterogeneous team testing performance comparison by team size for Full mode teams. The average testing accuracies over all homogeneous and heterogeneous Full mode teams are shown for reference.

badly while others perform quite well. The utility of heterogeneous teams is illustrated in Figure 6.30, in that heterogeneous teams are more stable with an increase in team size. Heterogeneous team composition is therefore beneficial for both learning modes.

Tables 6.9 and 6.10 list the best and worst performing homogeneous and heterogeneous 15 teams for both Full and Indp learning modes, respectively. Successful Full mode teams are generally of size eight, compared to the typical size four Indp mode team. Random Forest (RFT) is the best individual learning algorithm for this data set, and represents a base for strong heterogeneous teams. The best homogeneous teams benefit from Low frequency collaboration, achieving higher accuracy than that of the same team configuration which did not collaborate. Medium and High frequencies achieve best performance for Indp mode teams. There are several examples where an increase in collaboration frequency produced higher accuracy, but a decrease in accuracy when collaborating with High frequency. This again supports that collaborating can be beneficial to an extent, after which it degrades performance.

An advantage is shown in favor of heterogeneous teams, which perform between 12-52% better in the pool of lowest accuracy teams. The worst heterogeneous teams are of size two for both Full and Indp learning modes, demonstrating that some combinations of learning algorithms coupled with higher collaboration frequencies can actually degrade performance. Small team sizes are foremost out of those homogeneous and heterogeneous

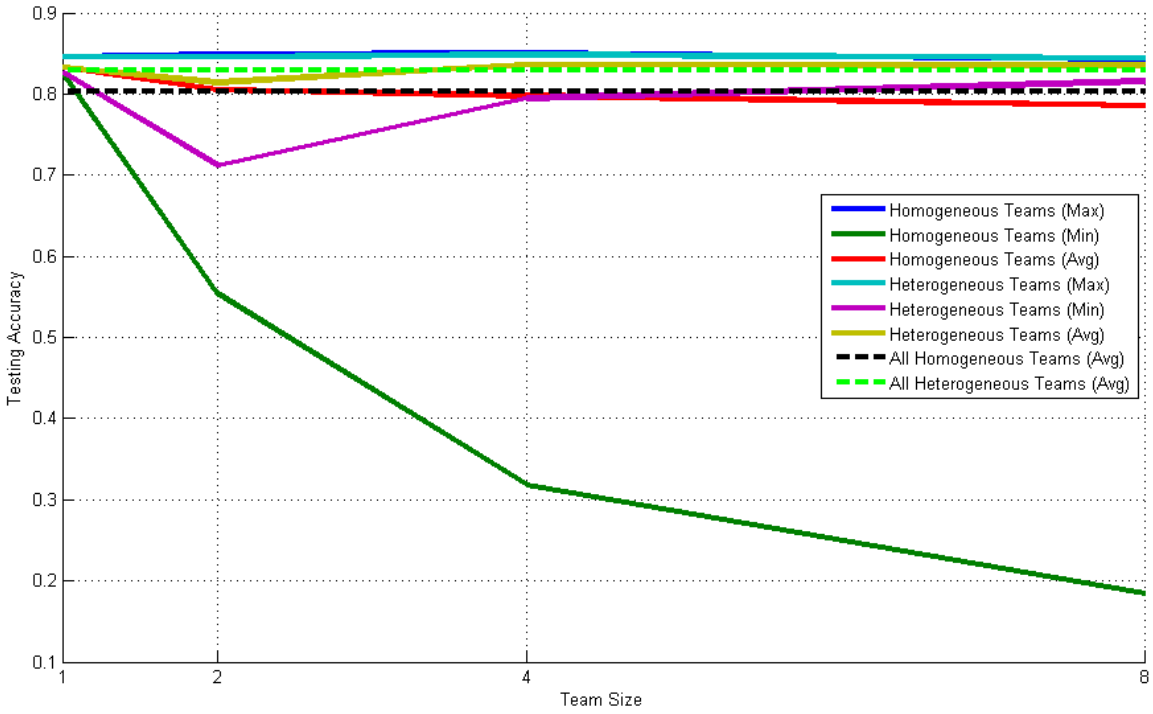

Figure 6.30: Homogeneous versus heterogeneous team testing performance comparison by team size for Indp mode teams. The average testing accuracies over all homogeneous and heterogeneous Indp mode teams are shown for reference.

teams that perform the worst. These results parallel those which were discussed earlier for team diversity. Specific learning algorithms consistently offer low accuracy, even when paired with other learners, such as Decision Table. Team heterogeneity again prevails over homogeneous team compositions in terms of learning team stability.

6.7.4 Team Size: Single versus Multiple Learners

A study of team learning inherently involves the question, "How large should the team be?" Specifically, it is desirable to study how scaling the team size affects different team dynamics (successful team composition, testing accuracy, collaboration frequency, etc.). The objective is to study what can be gained by using multiple collaborating learning algorithms, as opposed to a single learning algorithm for the entire data set. The following tables attempt to shed more light on these aspects, and are intended to supplement the results presented as part of other studies in this chapter.

Tables 6.11 and 6.12 list the five most successful and least successful teams as a function of team size. These tables show the direct result of using a tournament-style experimentation setup, as the best individual learners are highly involved in all successful teams of larger sizes. There exists a coupling between diversity level and team size for the most successful distributed learning (Indp mode) teams. The larger the team size, the more diverse must the teams be to achieve high accuracy. Size four teams prevail for Indp

Table 6.9: Overall best performing 15 homogeneous and 15 heterogeneous teams, based on combined testing accuracy and comparing Full and Indp modes.

Full Mode Teams				Independent Mode Teams			
Homogeneous	**Accuracy**	**Heterogeneous**	**Accuracy**	**Homogeneous**	**Accuracy**	**Heterogeneous**	**Accuracy**
8rft (low)	0.857774	7rft+1dt (none)	0.857181	4rft (high)	0.849594	3rft+1kst (med)	0.848143
8rft (none)	0.856521	7rft+1dtb (none)	0.856719	4rft (low)	0.849396	2nn+2rft (med)	0.847154
8rft (high)	0.856521	7rft+1kst (none)	0.856719	2rft (med)	0.849330	3rft+1kst (low)	0.847154
8rft (med)	0.855795	6rft+2nn (none)	0.856323	2rft (high)	0.848671	3rft+1dt (none)	0.847022
4rft (med)	0.855202	5rft+2nn+1dtb (none)	0.856191	2rft (low)	0.848011	3rft+1dtb (none)	0.847022
4rft (low)	0.855004	7rft+1kst (med)	0.856191	4rft (none)	0.847615	3rft+1kst (none)	0.847022
4rft (none)	0.854740	7rft+1dt (low)	0.856059	2rft (none)	0.846758	3rft+1nn (none)	0.847022
2rft (med)	0.854476	5rft+2nn+1dt (none)	0.855993	1rft (none)	0.845834	3rft+1kst (high)	0.846758
2rft (low)	0.854212	6rft+1dt+1kst (none)	0.855861	4rft (med)	0.845702	3rft+1nn (med)	0.846428
4rft (high)	0.853288	6rft+1dtb+1dt (none)	0.855861	8rft (none)	0.842602	2rft+1nn+1kst (none)	0.845966
2rft (high)	0.853157	6rft+1dtb+1kst (none)	0.855861	8rft (low)	0.841480	1kst+1rft (none)	0.845834
2rft (none)	0.852827	6rft+2dt (none)	0.855861	2nn (none)	0.839699	1dtb+1rft (none)	0.845438
1rft (none)	0.845834	6rft+2dtb (none)	0.855861	8rft (med)	0.839171	2rft+1kst+1dt (none)	0.845174
4nn (none)	0.842338	6rft+2kst (none)	0.855861	2dt (none)	0.838776	2rft+1kst+1dtb (none)	0.845174
2nn (none)	0.840557	6rft+2kst (med)	0.855861	4dt (none)	0.837588	1kst+1rft (low)	0.844977

Table 6.10: Overall worst performing 15 homogeneous and 15 heterogeneous teams, based on combined testing accuracy and comparing Full and Indp modes.

Full Mode Teams				Independent Mode Teams			
Homogeneous	**Accuracy**	**Heterogeneous**	**Accuracy**	**Homogeneous**	**Accuracy**	**Heterogeneous**	**Accuracy**
4nn (high)	0.613629	1nn+1kst (high)	0.730787	8dtb (high)	0.185236	1nn+1dtb (high)	0.711722
4nn (med)	0.613629	1nn+1kst (med)	0.731183	4dtb (high)	0.318425	1kst+1dtb (med)	0.748664
4nn (low)	0.735735	1kst+1dtb (med)	0.744970	2dtb (med)	0.553994	1kst+1dtb (high)	0.751897
2dtb (low)	0.756118	1dtb+1jrp (low)	0.753414	8dtb (low)	0.700508	1dt+1nn (low)	0.762122
2jrp (low)	0.762187	1kst+1dtb (high)	0.770895	2dtb (low)	0.746685	1nn+1jrp (low)	0.777426
2nn (low)	0.764035	1dt+1jrp (low)	0.790356	4nn (med)	0.753216	1dtb+1jrp (low)	0.778284
4dtb (low)	0.765618	1dt+1nn (low)	0.795699	2nn (high)	0.761264	1nn+1dtb (med)	0.780460
2nn (high)	0.770236	1nn+1jrp (high)	0.796491	4dtb (low)	0.761528	1dt+1dtb (low)	0.781582
2nn (med)	0.785804	1dt+1nn (high)	0.796491	2nn (low)	0.768586	1nn+1dtb (low)	0.787651
8kst (high)	0.786727	1dt+1jrp (med)	0.797612	2jrp (low)	0.781846	1dtb+1jrp (high)	0.790685
2jrp (high)	0.804407	1dt+1dtb (low)	0.802626	2jrp (med)	0.785144	1dt+1dtb (high)	0.790685
2dt (high)	0.804407	1nn+1jrp (med)	0.803945	4nn (high)	0.788311	1dt+1jrp (high)	0.791609
2dt (low)	0.804473	1dt+1nn (med)	0.803945	2jrp (high)	0.790883	1dt+1nn+1kst+1dtb (med)	0.794709
2dtb (med)	0.805264	1dtb+1jrp (high)	0.804341	2dt (low)	0.795105	1dt+1jrp (low)	0.795171
4dt (high)	0.808497	1dt+1dtb (high)	0.804341	2nn (med)	0.804077	1nn+1jrp (high)	0.799195

Table 6.11: Most successful five teams, based on combined testing accuracy, for Full and Indp learning distribution modes over each team size.

Full Mode Teams		Independent Mode Teams	
Size 1 Team	**Accuracy**	**Size 1 Team**	**Accuracy**
1rft (none)	0.845834	1rft (none)	0.845834
1dt (none)	0.837324	1dt (none)	0.837324
1nn (none)	0.835191	1nn (none)	0.835191
1kst (none)	0.834488	1kst (none)	0.834488
1dtb (none)	0.834158	1dtb (none)	0.834158
Size 2 Team	**Accuracy**	**Size 2 Team**	**Accuracy**
2rft (med)	0.854476	2rft (med)	0.849330
2rft (low)	0.854212	2rft (high)	0.848671
2rft (high)	0.853157	2rft (low)	0.848011
2rft (none)	0.852827	2rft (none)	0.846758
1nn+1rft (none)	0.850650	1kst+1rft (none)	0.845834
Size 4 Team	**Accuracy**	**Size 4 Team**	**Accuracy**
3rft+1dtb (low)	0.855795	4rft (high)	0.849594
2nn+2rft (none)	0.855202	4rft (low)	0.849396
4rft (med)	0.855202	3rft+1kst (med)	0.848143
3rft+1dt (none)	0.855070	4rft (none)	0.847615
4rft (low)	0.855004	2nn+2rft (med)	0.847154
Size 8 Team	**Accuracy**	**Size 8 Team**	**Accuracy**
8rft (low)	0.857774	7rft+1kst (low)	0.844053
7rft+1dt (none)	0.857181	6rft+2kst (low)	0.843525
7rft+1dtb (none)	0.856719	6rft+1dtb+1dt (none)	0.843459
7rft+1kst (none)	0.856719	6rft+2dt (none)	0.843459
8rft (none)	0.856521	6rft+2dtb (none)	0.843459

mode learning, while size eight teams are best for Full mode teams. Additionally, benefits of collaboration are observed, but largely at Low and Medium frequencies. Teams of size four and eight, which happen to contain the best overall teams, generally exhibit all of these aspects: mixture of learning algorithms with Low to Medium collaboration.

Conversely, the least successful teams are of size two, and are largely heterogeneous mixtures of poor individual classifiers. Collaboration between these poorly-matched classifiers may be the cause for low accuracy, as higher collaboration frequencies are observed for the worst teams. These results suggest that heterogeneous teams should not collaborate for this application. These tables further support the notion that adding individual learners to a team (increasing the team's diversity level) can help fill niches of the data and positively contribute to team success. However, a requirement appears to be a strong base of successful classifiers with which to add heterogeneous components to achieve a little more accuracy. The differences observed between the best and worst teams are generally only 10%, illustrating that many successful teams of classifiers can be constructed. Teams of size four appear to be the best choice, in terms of testing accuracy and time requirements for this application.

Table 6.12: Least successful five teams, based on combined testing accuracy, for Full and Indp learning distribution modes over each team size.

Full Mode Teams		Independent Mode Teams	
Size 1 Team	**Accuracy**	**Size 1 Team**	**Accuracy**
1ibk (none)	0.826440	1ibk (none)	0.826440
1nb (none)	0.826528	1nb (none)	0.826528
1rbf (none)	0.826967	1rbf (none)	0.826967
1lgr (none)	0.828067	1lgr (none)	0.828067
1prt (none)	0.831497	1prt (none)	0.831497
Size 2 Team	**Accuracy**	**Size 2 Team**	**Accuracy**
1nn+1kst (high)	0.730787	2dtb (med)	0.553994
1nn+1kst (med)	0.731183	1nn+1dtb (high)	0.711722
1kst+1dtb (med)	0.744970	2dtb (low)	0.746685
1dtb+1jrp (low)	0.753414	1kst+1dtb (med)	0.748664
2dtb (low)	0.756119	1kst+1dtb (high)	0.751897
Size 4 Team	**Accuracy**	**Size 4 Team**	**Accuracy**
4nn (none)	0.842338	1rft+1nn+1kst+1dtb (low)	0.829474
1rft+1dt+1nn+1kst (low)	0.842734	1rft+1dt+1nn+1dtb (med)	0.829606
1rft+1nn+1kst+1dtb (high)	0.842866	2rft+2dtb (low)	0.829738
2rft+2dtb (med)	0.843525	2rft+2dt (low)	0.829870
1dt+1nn+1kst+1dtb (none)	0.843657	2rft+1kst+1dtb (low)	0.829936
Size 8 Team	**Accuracy**	**Size 8 Team**	**Accuracy**
8kst (high)	0.786727	8dtb (high)	0.185237
8dt (med)	0.812587	8dtb (low)	0.700508
8dt (high)	0.812851	8 (1 of each) (low)	0.815951
8dt (low)	0.816875	8dt (low)	0.822680
8dtb (low)	0.821096	6rft+2nn (low)	0.824197

6.7.5 Self-Learning versus Collaboration

Collaboration allows a learner in a team to distribute difficult training instances to all other learners, increasing the chance that one or more of them will properly learn it and be able to correctly classify that instance as a team once learning has finished. In this respect, it is expected that highly heterogeneous teams will benefit more from collaboration, as each learning algorithm models the data in different ways. Thus, if one learning algorithm cannot incorporate an instance into its classification model, one or more others would. Homogeneous teams would likely not be able to take advantage of this, contributing to the estimate that collaboration could be detrimental to homogeneous teams. Additionally, collaboration can potentially contribute to overfitting, especially for large teams which collaborate with very high frequency on a small data set. Collaboration was intended to be used on large data sets with moderate team sizes (those studied in this dissertation). It was expected that Indp mode teams would require additional collaboration, to gain exposure to more of the training set (albeit the most difficult instances found by other learners). Based on results from Chapter 4, it was expected that some collaboration would be beneficial, but High frequency collaboration would become detrimental to team success.

Figures 6.31 and 6.32 present collaboration results, comparing homogeneous and heterogeneous collaborating and non-collaborating teams by team size, for Full and Indp mode teams, respectively. On average, non-collaborative Full mode teams experience an increase in accuracy with an increase in team size. Heterogeneous non-collaborative teams perform best overall, while collaborative heterogeneous teams experience an increase coupled with team size except for teams of size two. Collaboration appears to negatively affect homogeneous teams of size two and four, and specifically should not be utilized for pairs of classifiers.

On the other hand, for Indp mode learning, collaborating homogeneous teams experience a substantial decrease in accuracy as a function of team size. Out of those teams which collaborate, heterogeneous teams of size four perform best. Again, heterogeneous non-collaborative teams achieve best overall accuracy for distributed learning. Heterogeneous teams do offer more stability in terms of retaining accuracy during collaboration. Collaborative teams do not appear to be beneficial for distributed learning for this application. This result was not expected, and was likely a product of the binary classification problem. However, collaboration was observed to offer advantages for certain team configurations. The frequency of collaboration is investigated next.

Figure 6.33 presents further collaboration versus self-learning results as a function of collaboration frequency and team size. This allows the study on how collaboration frequency affects different team sizes. All team sizes and learning modes experience a decrease in accuracy with an increase in collaboration frequency. Full mode collaborative teams of size four and eight outperform individual classifiers, but do not offer an increase in accuracy over identical teams which do not collaborate. Therefore, size four and eight teams

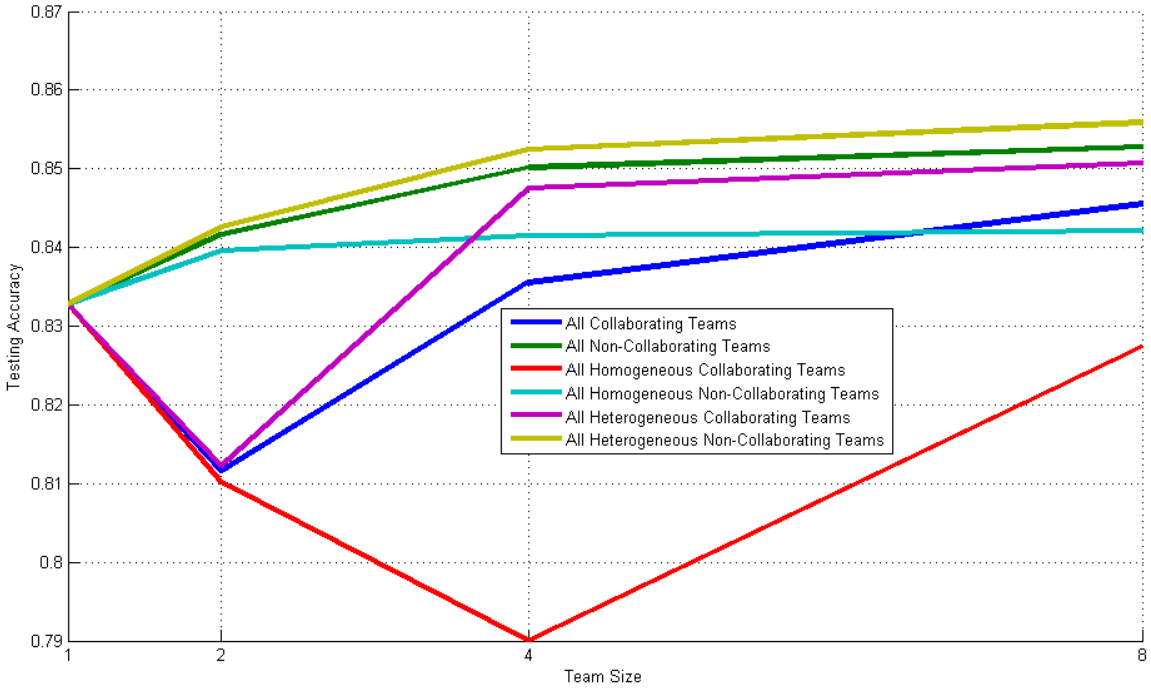

Figure 6.31: Analysis of collaboration versus self-learning average testing accuracy by team size for Full mode teams.

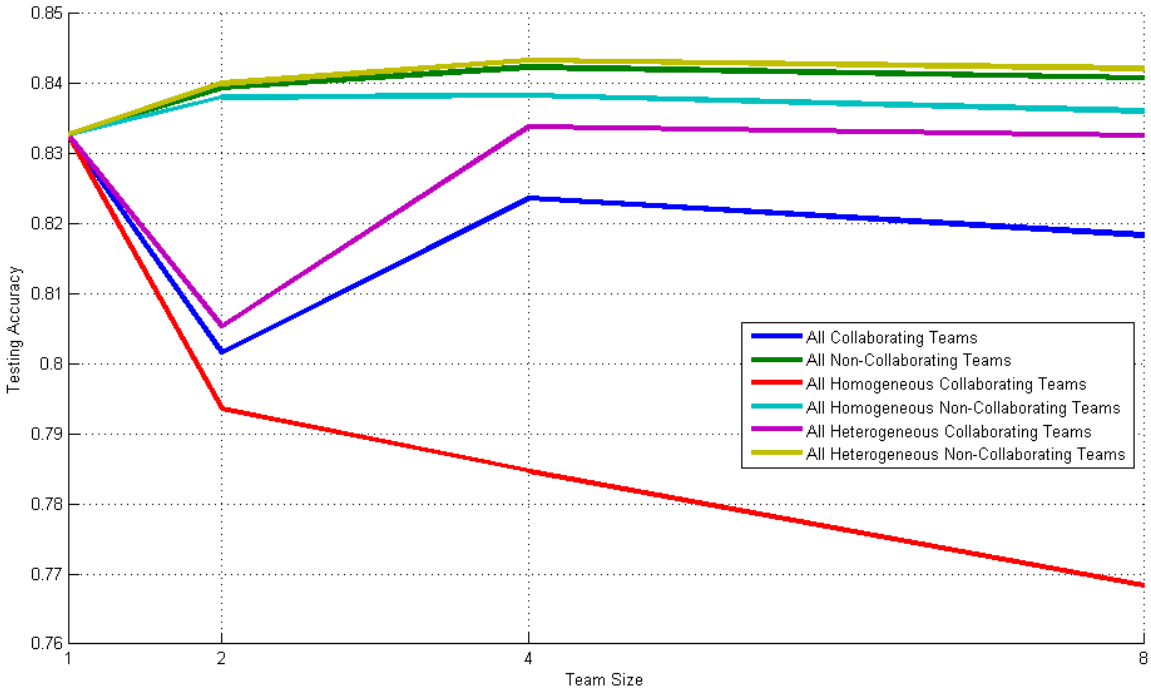

Figure 6.32: Analysis of collaboration versus self-learning average testing accuracy by team size for Indp mode teams.

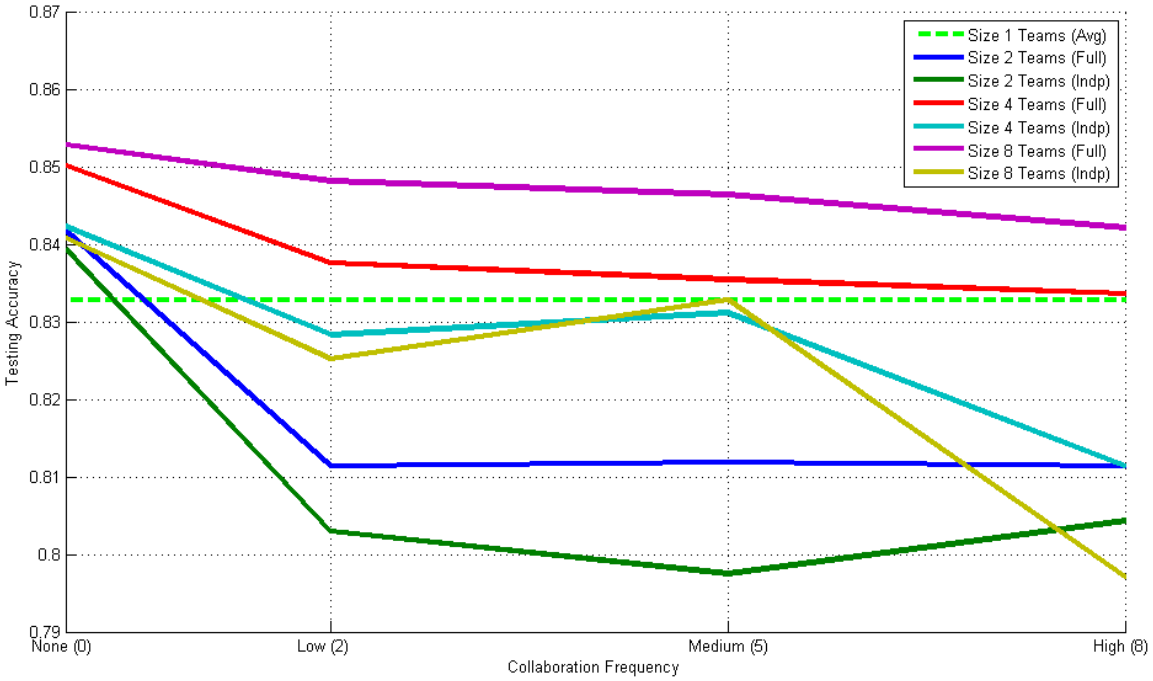

Figure 6.33: Analysis of collaboration versus self-learning average testing accuracy by collaboration frequency and team size.

are not affected as much as other team sizes by collaboration. Indp mode teams which collaborate achieve their highest accuracy when collaborating with Medium frequency. Based on these results, it is observed that collaboration has little benefit, only positively manifesting itself for certain team configurations of size four and eight.

Figures 6.34 and 6.35 illustrate how team heterogeneity affects average testing accuracy as functions of team size and collaboration frequency for Full and Indp mode teams, respectively. For this data set, collaboration causes a reduction in average testing accuracy for any team size, regardless of team composition. For Indp mode experiments, collaboration with Medium frequency performs as well on average as not collaborating for size eight homogeneous teams. This suggests that, for Indp mode teams, collaboration can be effective, but does not offer a substantial increase in accuracy to warrant the additional resources to do so.

Another important aspect of collaboration to investigate is the improvement, if any, that it provides over not collaborating. Figure 6.36 presents a measure of improvement of collaboration over not collaborating as a function of collaboration frequency over varying team sizes for Full and Indp mode teams. This represents another way to look at how collaboration is affecting team accuracy, and how collaboration frequency is tied to team size. Improvement is calculated by dividing the average testing accuracy of a collaborative team over the average testing accuracy of a non-collaborative team of the same size. A value above 1.0 signifies that collaborating with that frequency improves accuracy for that team

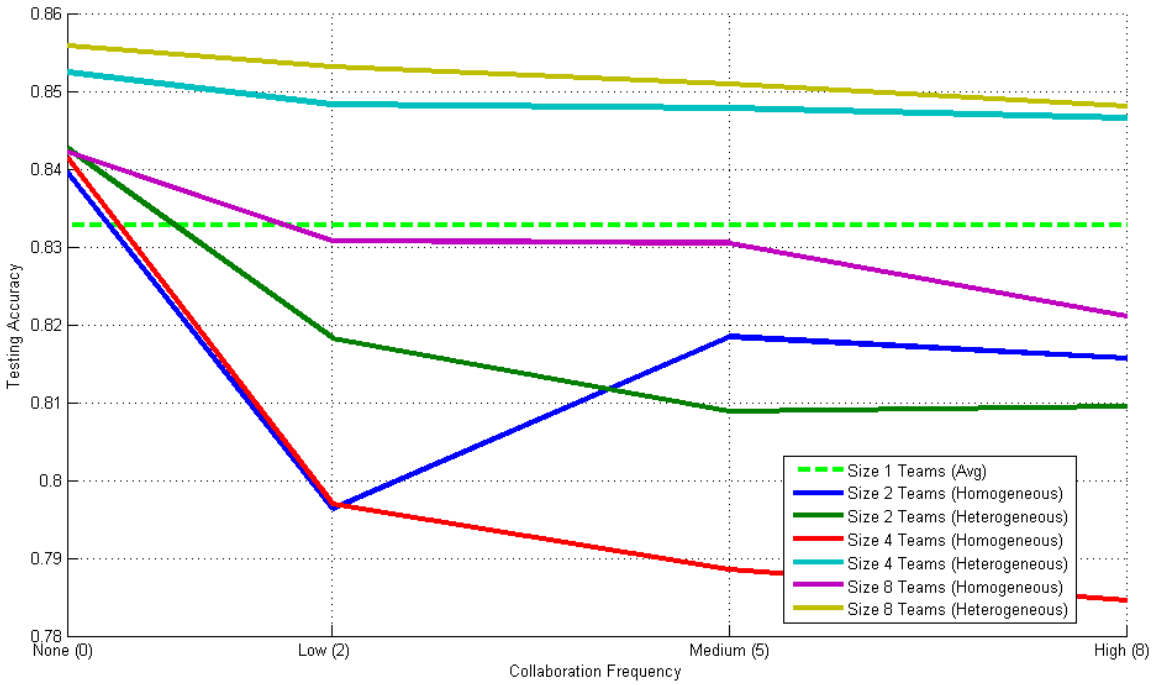

Figure 6.34: Analysis of collaboration versus self-learning average testing accuracy by collaboration frequency for Full mode homogeneous and heterogeneous teams.

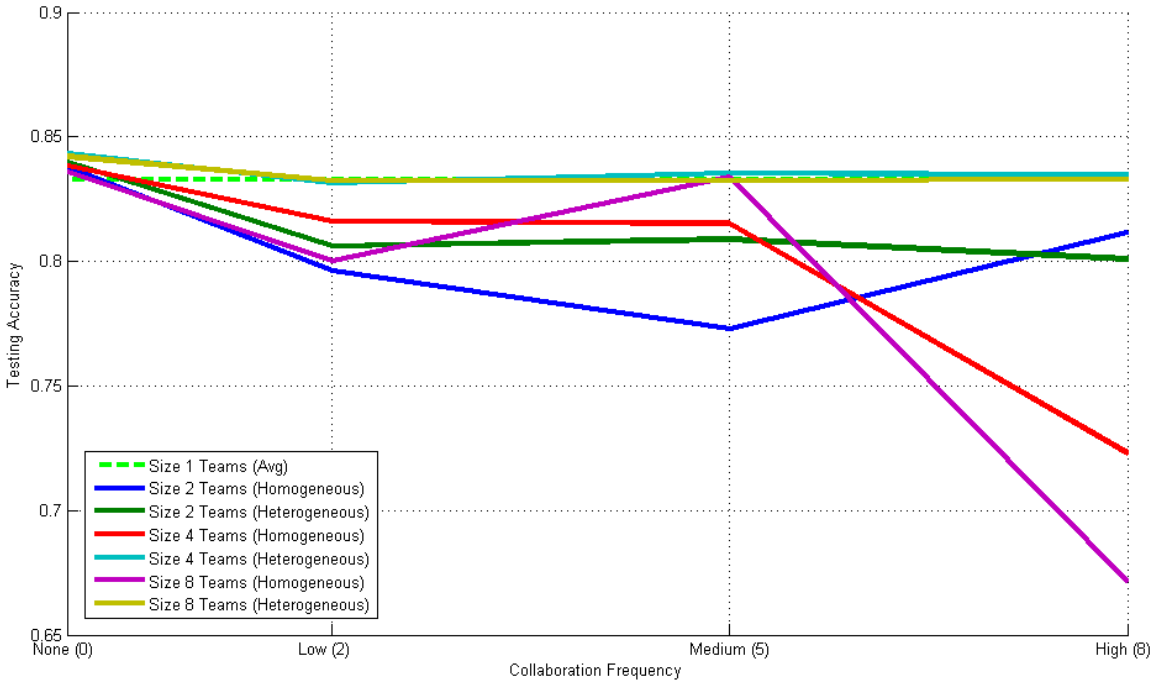

Figure 6.35: Analysis of collaboration versus self-learning average testing accuracy by collaboration frequency for Indp mode homogeneous and heterogeneous teams.

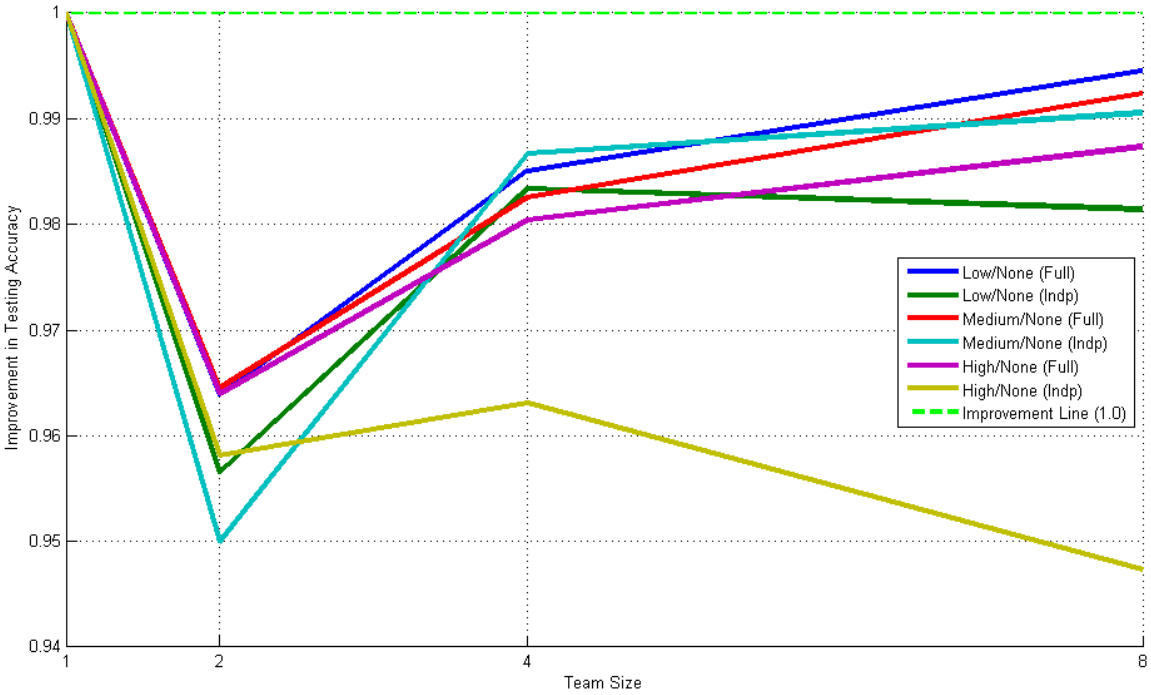

Figure 6.36: Investigation of the effect/improvement of collaboration (Low,Med,High) on average testing accuracy compared to non-collaborating teams by team size.

size. Therefore, each collaboration frequency's effect on team accuracy can be analyzed for various team sizes.

No average accuracy improvement is observed for any team size and collaboration frequency for this data set. However, a trend is observed suggesting that, as team size increases, collaboration makes progress toward improvement. Low collaboration frequency performs best for Full mode teams, while Medium collaboration frequency performs best for Indp mode teams. For both learning modes, High frequency collaboration performs worst, suggesting that collaboration should be performed with Low or Medium frequency. Contrary to our hypothesis, Indp mode teams appear to benefit less from collaboration with larger team sizes. This again is likely a product of the classification problem being binary. With very large team sizes, collaboration may offer the hypothesized accuracy improvement.

Tables 6.13 and 6.14 list the best and worst 10 collaborative and non-collaborative teams, respectively. These tables illustrate what team compositions perform best for collaborative teams and non-collaborative teams for Full and Indp mode teams. Of the most successful teams, both collaborative and non-collaborative Full mode teams are of larger team sizes. Conversely, both collaborative and non-collaborative Indp mode teams are of smaller team sizes. For Full mode teams, overall performance is nearly identical for collaborative and non-collaborative teams. For Indp mode teams, a collaborative team achieves the highest testing accuracy, and there are more high-accuracy collaborative teams.

Table 6.13: 10 best performing collaborating and non-collaborating teams, based on average testing accuracy, for Full and Indp modes.

Full Mode Teams		Independent Mode Teams	
No Collaboration	**Accuracy**	**No Collaboration**	**Accuracy**
7rft+1dt (none)	0.857181	4rft (none)	0.847615
7rft+1dtb (none)	0.856719	3rft+1dt (none)	0.847022
7rft+1kst (none)	0.856719	3rft+1dtb (none)	0.847022
8rft (none)	0.856521	3rft+1kst (none)	0.847022
6rft+2nn (none)	0.856323	3rft+1nn (none)	0.847022
5rft+2nn+1dtb (none)	0.856191	2rft (none)	0.846758
5rft+2nn+1dt (none)	0.855993	2rft+1nn+1kst (none)	0.845966
6rft+1dt+1kst (none)	0.855861	1kst+1rft (none)	0.845834
6rft+1dtb+1dt (none)	0.855861	1rft (none)	0.845834
6rft+1dtb+1kst (none)	0.855861	1dtb+1rft (none)	0.845438
Collaboration	**Accuracy**	**Collaboration**	**Accuracy**
8rft (low)	0.857774	4rft (high)	0.849594
8rft (high)	0.856521	4rft (low)	0.849396
7rft+1kst (med)	0.856191	2rft (med)	0.84933
7rft+1dt (low)	0.856059	2rft (high)	0.848671
6rft+2kst (med)	0.855861	3rft+1kst (med)	0.848143
3rft+1dtb (low)	0.855795	2rft (low)	0.848011
7rft+1kst (low)	0.855795	2nn+2rft (med)	0.847154
8rft (med)	0.855795	3rft+1kst (low)	0.847154
7rft+1dtb (low)	0.855597	3rft+1kst (high)	0.846758
4rft (med)	0.855202	3rft+1nn (med)	0.846428

Examining the lowest accuracy teams, non-collaborative teams for both Full and Indp modes perform substantially better than those which collaborate. This suggests that certain combinations of learning algorithms, when collaborating often, experience a significant decrease in team classification accuracy. Results indicate that collaboration frequency is coupled with both team size and learning mode. We therefore conclude that collaboration with Low or Medium frequency will lead to the best general results, especially when performing distributed learning.

6.7.6 Individual Learner Contribution

The contribution and success of individual machine learning algorithms were investigated. By analyzing a learning algorithm's contribution to success, we can extract which learning algorithms are generally successful for this application, and those which also perform well together. If teams that contain a certain learning algorithm consistently perform better than others not containing that learning algorithm, then that algorithm can be considered valuable (or a major contributor) to a team's combined testing classification accuracy.

This is measured in two primary ways. First, as shown in Figure 6.37, average testing accuracy is compared for each learning algorithm over all teams in which that algorithm participated. On average, JRP, RBF, NB, and IBK perform the worst, whereas DT, LGR,

Table 6.14: 10 worst performing collaborating and non-collaborating teams, based on average testing accuracy, for Full and Indp modes.

Full Mode Teams		Independent Mode Teams	
No Collaboration	**Accuracy**	**No Collaboration**	**Accuracy**
1ibk (none)	0.826440	1ibk (none)	0.826440
1nb (none)	0.826528	1nb (none)	0.826528
1rbf (none)	0.826967	1rbf (none)	0.826967
1lgr (none)	0.828067	1lgr (none)	0.828067
1prt (none)	0.831497	8dtb (none)	0.829342
2dtb (none)	0.833828	1prt (none)	0.831497
4dtb (none)	0.833894	2jrp (none)	0.832707
1jrp (none)	0.834026	1dtb+1jrp (none)	0.833828
2jrp (none)	0.834092	8all (none)	0.833960
1dtb (none)	0.834158	1jrp (none)	0.834026
Collaboration	**Accuracy**	**Collaboration**	**Accuracy**
4nn (high)	0.613629	8dtb (high)	0.185236
4nn (med)	0.613629	4dtb (high)	0.318425
1nn+1kst (high)	0.730787	2dtb (med)	0.553994
1nn+1kst (med)	0.731183	8dtb (low)	0.700508
4nn (low)	0.735735	1nn+1dtb (high)	0.711722
1kst+1dtb (med)	0.744970	2dtb (low)	0.746685
1dtb+1jrp (low)	0.753414	1kst+1dtb (med)	0.748664
2dtb (low)	0.756118	1kst+1dtb (high)	0.751897
2jrp (low)	0.762187	4nn (med)	0.753216
2nn (low)	0.764035	2nn (high)	0.761264

KST, PRT, and RFT perform the best. Teams containing NN, DTB, and KST learners are less stable, as they can produce very low classification accuracy when included in a team. On the other hand, learning algorithms such as LGR, PRT, and RFT are stable learning algorithm choices, as their minimum team accuracies are still comparably high.

Each algorithm has a comparable maximum accuracy, meaning that it was involved in one or more teams which contained heterogeneous learning algorithms that collectively produced high testing accuracy for both Full and Indp mode teams. These results indicate that learning algorithms such as DT, LGR, KST, PRT, and RFT are reliable when constructing a team of multiple classifiers for this and other similar data sets. Learning algorithms such as JRP, RBF, NB, and IBK are not reliable when constructing a team for this and other similar data sets. Combining these algorithms is encouraged, to produce a team that has a mixture of strong classifiers. Results are similar across learning modes.

As shown in Figure 6.38, the number of total teams which each learning algorithm participated in can be compared. This figure shows individual learning algorithm accuracies on the entire data set (i.e., size one teams), and learning algorithm participation over all team learning experiments as a percentage. The byproduct of the tournament-style experimental setup is the use of the best performing algorithms in more teams. It is also evident that the RFT classifier is used in 25% more teams than the next best set of

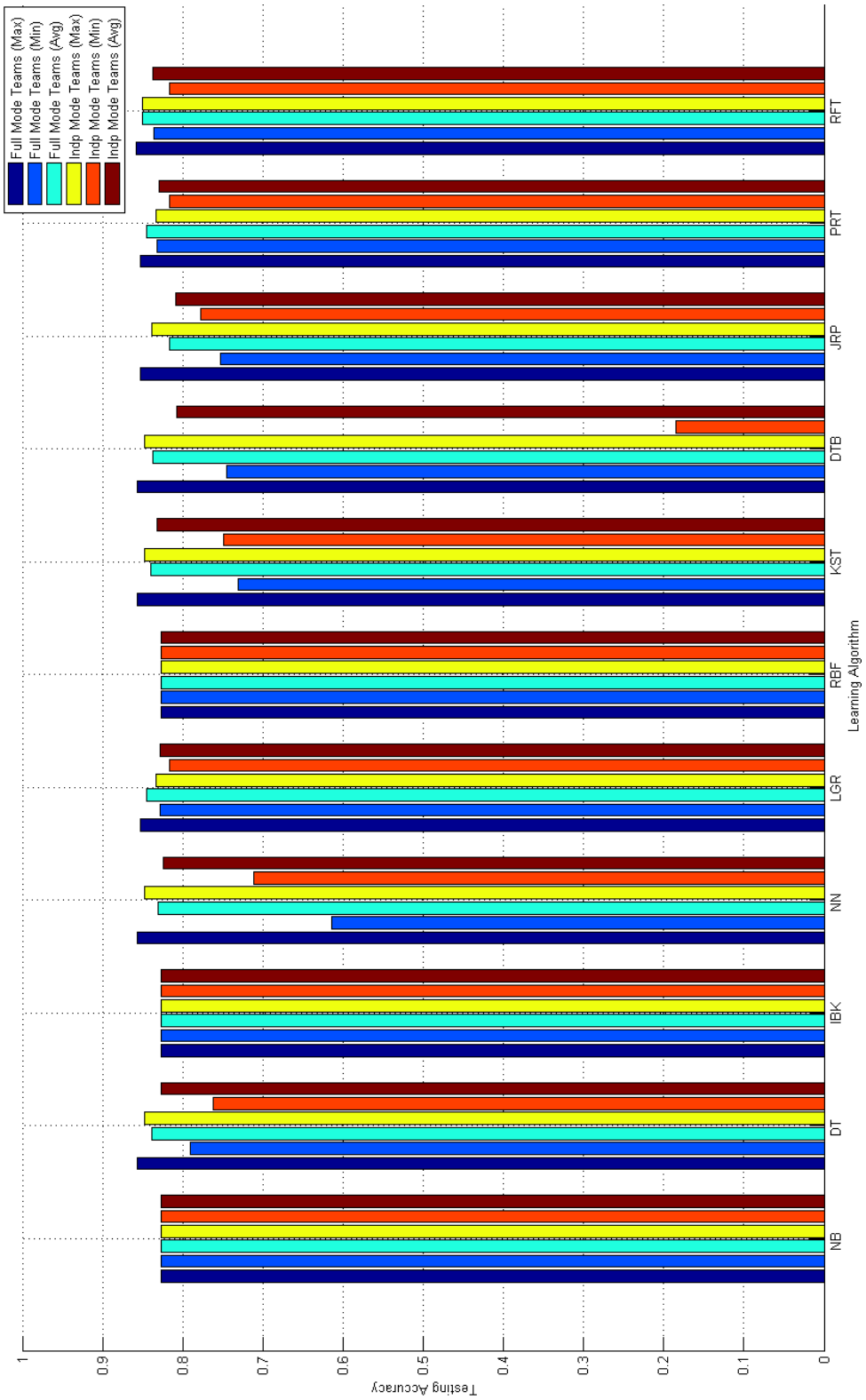

Figure 6.37: Minimum, maximum, and average testing accuracies over all teams containing one or more of each learning algorithm for Full and Indp mode teams. This offers a view into how teams performed based on their learning algorithm composition, specifically focused on membership of certain learning algorithms.

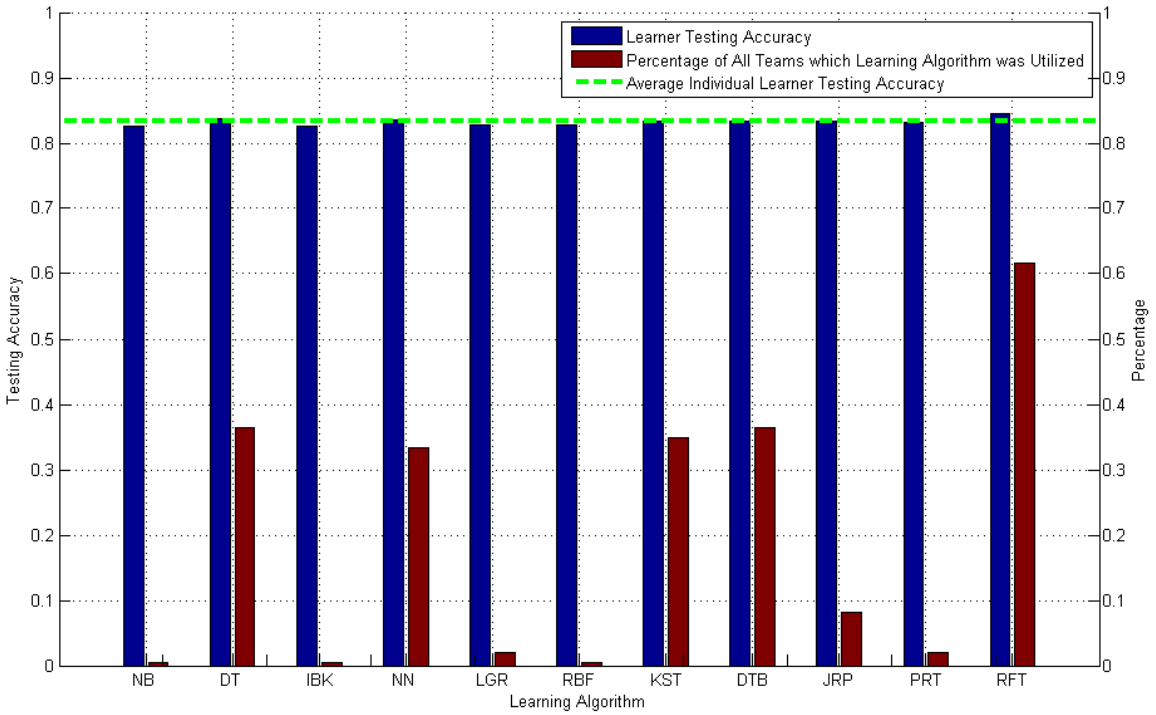

Figure 6.38: Individual learning algorithm testing accuracies for entire data set (team size of 1), accompanied by each learning algorithm's overall participation by the percentage of teams in which it was a member. This offers a view of general success and ranking of the individual learning algorithms for this data set.

classifiers. This is a result of combining teams of size two to form teams of size four, and combining teams of size four to form teams of size eight. DT, DTB, KST, and NN are also used in many of the teams as part of this study. These classifiers comprised more size two and four teams in comparison. The NB, IBK, RBF, LGR, and PRT classifiers were utilized the least over all teams.

6.8 Conclusion

Using the proposed Multi-Agent Collaborative Learning Architecture, we were able to obtain ∼86% classification accuracy. Several heuristics were discussed for constructing teams of multiple collaborative classifiers. A tradeoff between team size and team composition, including the time required to complete the learning process, also became apparent.

It was found that Full mode learning consistently achieves higher classification accuracy compared to distributed learning of the data. Heterogeneous team compositions perform best on average; however, the best team was of homogeneous composition and collaborated with Low frequency. Specifically, the Random Forest classifier represented the best individual classifier, as well as a necessary base for increased classification accuracy in

heterogeneous teams. Collaboration was found to be beneficial in specific cases, generally produced positive results in testing accuracy when utilized with Low and Medium frequency. In general, collaboration was not found to be beneficial for this application. Equally diverse teams (team size divided by diversity level of 1 to 2) were not among the best.

Assumptions made as part of this classification problem and general study could, in part, contribute to the similar accuracy of most classifier teams. Specifically, the radar data were acquired over the ice sheet eight years apart. With global warming and climate change being of increased focus and research in recent years, this time difference could have allowed for small changes on top of, within, and at the base of the ice. These changes could, in turn, affect the radar data and therefore the method utilized to achieve radar independence in a manner which affects classification. For instance, temporal temperature, snow accumulation, and/or melting alter the ice sheet. Radar settings and hardware differences may also contribute to small-scale changes in the radar data and measured intensity at the subglacial interface. However, at the locations where these data were acquired, subglacial water presence is assumed to be relatively static over the time difference between data sets.

Additionally, this is a binary (two class) classification problem. As such, more heterogeneous, diverse teams may not be required to adequately classify the data set with high accuracy. This is a unique classification problem, attempting to associate remotely sensed radar data properties to subglacial water presence. To our knowledge, this is the first attempt to classify water presence from radar data acquired over the polar regions using multiple machine learning algorithms. Knowledge of ice dynamics, ground truth core data, and attribute extraction from radar data have allowed us to learn a model of subglacial water presence to create water presence maps over the entire continent of Greenland. We plan to extend this concept to Antarctic radar data as well, such that water presence maps can be generated and monitored from new radar data as they become available.

For this classification problem, it was found that utilizing a team of multiple collaborative machine learning algorithms was necessary to maximize classification accuracy. Size four and eight teams proved to be well suited for this data set. These results, together with previous results, are summarized and compared in Chapter 7.

Chapter 7

Conclusions

7.1 Summary

As part of this dissertation, three primary applications have been the focus of collaborative multi-agent learning and classification studies: (1) multi-robot cooperative pursuit and capture, (2) subsurface rock facies classification from wireline well log data, and (3) subglacial water presence classification from radar data acquired over the Greenland ice sheet. Through simulation and experimentation, a deeper understanding of team learning has been accomplished. Specifically, heuristics which govern successful learning team construction, combination of multiple classifiers, and sharing/collaboration have been discovered. These heuristics offer general guidelines for approaching multi-agent machine learning problems, including learning tasks for multi-robot teams.

The multi-robot cooperative pursuit and capture experiments compared teams of size four in terms of task success (captures), team composition (homogeneous and heterogeneous mixtures of Reinforcement Learning and Neural Network learners), learning strategy (centralized, distributed, and a hybrid mixture), opinion conflicts and resulting opinion changes from conflict resolution, and learning speed. Results have shown that using a heterogeneous mixture of learning algorithms for a concurrent cooperative task performs better than the typical homogeneous learning teams. This represents a more advanced approach to team learning. Periodic knowledge sharing also shows benefits in terms of increasing the learning rate, effectively learning more in less time. However, results suggest that sharing with higher frequency may actually reduce the learning rate, depending on the task. Hybrid combinations of learning strategies show no clear advantage when compared to fully centralized and distributed teams.

Through this application, we also demonstrated that knowledge from heterogeneous sources can be successfully communicated and integrated by using an inter-knowledge representation common to all learners. Knowledge conflicts are extremely important to properly resolve, as they represent areas of agreement or disagreement between learning

agents. The conflict resolution method depends on the application's focus. Here, we suspect that simple averaging was insufficient for combining differing viewpoints, suggesting that it may not be suitable for complex tasks. A more sophisticated approach, which retains agent specialization, may produce better results. Some aspects of this approach are problem- and learner-specific. For example, the inter-knowledge representation may drastically change between applications. As the inter-knowledge representation changes, the conversion methods for individual learning algorithms will require changing as well. These characteristics are important, as poor choices in conflict resolution method and inter-knowledge representation can potentially lead to unacceptable results.

A Multi-Agent Collaborative Learning Architecture has been developed to perform large-scale team classification studies. This architecture, built using the WEKA Machine Learning Suite, was utilized to study various aspects of team classification for two subsurface classification problems.

The Kansas Geological Survey (KGS) rock facies data set consists of seven numerical attributes and eight rock facies classes from various wells in Kansas. Using the Multi-Agent Collaborative Learning Architecture, we were able to obtain approximately 84.5% absolute testing accuracy, an improvement of about 6.5% over the best results that KGS was able to achieve on this rock facies classification data set. Several heuristics are proposed for constructing teams of multiple collaborative classifiers. A tradeoff between team size and team composition also became apparent, including the time required to complete the training, testing, and combination processes.

Highly diverse and heterogeneous team compositions are preferred, due to their robustness, complementary nature, and general good performance. A guideline for team construction was also discovered. A heterogeneous mixture of strong homogeneous teams, especially those compositions that are equally diverse (team size divided by diversity level being 1 to 2), allows a team of multiple agents to classify with high combined accuracy. Furthermore, collaboration was found to be most beneficial for homogeneous teams of size four, and generally produced an increase in testing accuracy when utilized with Low to Medium frequency. Size four and eight teams proved to be well-suited for this data set. Combination methods of Average, Selective Average, Multiply, and Selective Multiply were observed to offer the highest combined team testing accuracy. Lastly, the IBK, KST, NN, RFT, and DT algorithms performed consistently better than the others. For this difficult classification problem, utilizing a team of multiple heterogeneous collaborative machine learning algorithms was necessary to maximize accuracy.

The third study represents a unique classification problem, attempting to associate remotely sensed radar data properties to subglacial water presence. To our knowledge, this is the first study to attempt to classify water presence from radar data acquired over the polar regions using multiple machine learning algorithms. Knowledge of ice dynamics, ground truth core data, and attribute extraction from radar data has allowed us to learn a

model of subglacial water presence to create water presence maps over the entire continent of Greenland. Using methods to achieve radar independence, attributes were extracted to create a binary (wet or frozen) data set from radar data acquired in 1999 and 2007. Using the Multi-Agent Collaborative Learning Architecture, we were able to obtain ∼86% classification testing accuracy.

As with the KGS study, it was found that Full mode learning consistently achieves higher classification accuracy compared to distributed learning of the data. Heterogeneous team compositions perform best on average; however, the best team was of homogeneous composition and collaborated with Low frequency. Specifically, the Random Forest classifier represents the best individual classifier, as well as a necessary base for increased classification accuracy in heterogeneous teams. Collaboration was found to be beneficial in specific cases, generally producing positive results in testing accuracy when utilized with Low and Medium frequency. In general, collaboration was not found to be beneficial for this application. Equally diverse teams (team size divided by diversity level of 1 to 2) were not among the best for this data set.

Assumptions made as part of this classification problem and general study could, in part, contribute to the similar testing accuracy of most classifier teams. Specifically, the radar data was acquired over the ice sheet eight years apart. With global warming and climate change being of increased focus and research in recent years, this time difference could have allowed for small changes on top of, within, and at the base of the ice. These changes may, in turn, affect the radar data and therefore the method utilized to achieve radar independence in a manner which affects classification. For instance, temporal temperature, snow accumulation, and/or melting alter the ice sheet. Radar settings and hardware differences may also contribute to small-scale changes in the radar data and measured intensity at the subglacial interface. However, at the locations where these data were acquired, subglacial water presence is assumed to be relatively static over the time difference between data sets. We plan to extend this concept to Antarctic radar data as well, such that water presence maps can be created and monitored from new radar data, as they become available.

7.2 Contributions

The primary contributions of this dissertation are the detailed studies of team learning and collaboration across three different learning domains. Experiments with these unique applications allow for extraction of general heuristics that can be utilized for other domains and classification problems. Lessons learned as part of the overall study are discussed below, for both theoretical and application contexts, to better tie the results from all applications together.

7.2.1 Theoretical

General findings from these experiments suggest that constructing a team of classifiers using a heterogeneous mixture of homogeneous teams is preferred. Larger teams generally perform better, as decisions from multiple learners can be combined to arrive at a consensus decision. Employing heterogeneous learning algorithms integrates different error models to arrive at higher accuracy classification from complementary knowledge bases. Collaboration, although not found to be universally beneficial, offers certain team configurations an advantage. Collaboration with Low to Medium frequency was found to be beneficial, while High frequency collaboration was found to be detrimental to team classification accuracy. Full mode learning consistently outperforms independent mode learning, as more data is provided to each learner.

Unless the decision space is easily separable, multi-agent systems should be used in place of a uni-agent solution. Multi-agent learning systems offer the robustness and flexibility of integrating multiple viewpoints, differing error models, and varying levels of expertise. Heterogeneous teams are better on average for these reasons, and have shown their utility for the data sets discussed in this dissertation. As the number of attributes and classes for the classification problem increases, larger and more diverse team compositions appear to be necessary to adequately associate inputs to target categories. For example, the KGS study involved seven numerical sensor attributes and eight rock classes. The best teams were highly diverse and of size eight. The subglacial water classification study involved only four numerical attributes and two subglacial state classes (binary classification problem). The best teams were still large in size, but were composed of several Random Forest classifiers accompanied by one or two other heterogeneous classifiers. In this case, a strong base classifier was more important than team diversity. In both applications, larger teams performed best.

Sharing/collaboration also proved to be beneficial in certain situations, but only at Low to Medium frequency. Collaboration is more useful for distributed learning problems, and more challenging data sets. Collaboration increases the learning rate (i.e., knowledge is gained more rapidly) and distributes the responsibility of learning challenging examples to each member of the team. A general rule for collaboration would be to collaborate at Low to Medium frequency with larger teams composed of multiple different learning algorithms. This way, the different algorithms can experience multiple iterations in their collective attempt to learn more challenging classification instances. The repetition and multiple exposures of instances that collaboration offers, allows the teams of classifiers the opportunity to spend less learning effort on simple instances and a more dedicated effort on those instances that one or more learning algorithms are misclassifying during the learning process. Multiple collaboration events integrates an interactive aspect to the learning process, which appears to be required to achieve the best testing accuracy for the data sets discussed in this dissertation.

A tradeoff between time, memory, and accuracy was observed for all experiments. Larger teams require more training time, especially if each learner is provided the entire training set (Full mode). Distributed learning reduces the time required to generate a collective classification model, at the sacrifice of accuracy. Certain algorithms have time and memory requirements. For instance, K* requires a significant amount of memory, and Artificial Neural Networks typically require longer training times. Therefore, some learning algorithms are better suited for certain tasks. Reinforcement learning is more applicable to reward-based tasks that allow for substantial training time. Thus, if memory, time, or accuracy requirements need to be upheld, specific algorithms will be more applicable than others.

This dissertation will hopefully provide the machine learning and pattern classification communities with theoretical guidelines for constructing high-accuracy teams of classifiers. This work also supports recent advances in large-scale classification efforts, such as the Netflix Prize [90] that was recently won using a large heterogeneous ensemble of classifiers. In this case, two of the top teams combined their algorithms and efforts to construct the best classification model. The work presented in this dissertation parallels the process used by these human teams, which used a team of heterogeneous classifiers and collaborated to combine their approaches into the best collective classifier for the Netflix user recommendation data set.

7.2.2 Applications

Results presented in this dissertation support the application of multi-agent machine learning and collaboration to current challenging, real-world classification problems. The Multi-Agent Collaborative Learning Architecture is currently being applied to other problems, such as stand-off device classification and novelty detection in space imagery. Multi-agent collaborative learning and classification can be applied to many different problems and domains, namely, decision support, bioinformatics, medicine, gene sequencing, etc. Military and medical institutions are increasingly discovering needs which only advanced machine learning and collaborative techniques can address. Use of heterogeneous teams of classifiers has been proven commercially as well, with the recent completion of the Netflix Prize and other challenges involving large and highly complex data sets.

Machine learning and pattern recognition will continue to be applied to other disciplines to advance the state-of-the-art, and enhance our understanding of how humans and animals learn and operate in their environments. As discussed in the following section, robotics is an example of a field that could see a significant advance, with the integration of learning, to increase autonomy and overall ability of platforms. Further, with the growing importance of resources and prediction of events, sophisticated classification techniques for determining location of subsurface reservoirs, disease outbreaks, and the change in weather and climate

will experience increased attention. Many applications exist that can benefit from use of such technology, especially with the amount of data available from today's information systems. Finding new ways in interdisciplinary fields to leverage these approaches should be a focus for the future.

7.3 Limitations and Future Work

The current Multi-Agent Collaborative Learning Architecture is a serial process, i.e., the learning algorithms are not learning and collaborating in a truly parallel fashion. A logical extension to this architecture is to incorporate parallel processing for large-scale, massively-parallel classification problems. Grid WEKA [155] and WEKA-Parallel [33] are currently available for parallel machine learning applications. Efficient collective communication for sharing knowledge and collaborating could be achieved through the use of a parallel cluster, employing the Message Passing Interface (MPI) [131]. As communication is a primary source of overhead and requires significant computing and energy resources for hardware systems, introducing parallel aspects to multi-agent machine learning can aid in its application to larger problems (large training/testing sets, as well as memory and temporal restrictions).

An interesting study in regards to collaborative learning is how incremental training and testing accuracies are affected during the learning/collaboration process. By analyzing incremental performance of each algorithm in a team of classifiers, overfitting can potentially be avoided for certain learning algorithms. For example, it was observed that, for some team configurations, certain learning algorithms achieved their best individual performance after a couple of collaboration events, whereas others achieved their best individual performance after four collaboration events. By forcing the mandatory continuation of the collaboration process for all learners, some classifiers experienced a decrease in individual performance once all collaboration events were completed. Theoretically, it would be beneficial to observe these incremental performances and use the corresponding classifier models when making team decisions. Incremental accuracies and models for all learning algorithms were saved as part of the studies discussed in Chapters 5 and 6. These data can be directly used for such a study.

There are many facets of team learning that require further investigation. Team learning introduces social interaction, team-building, and knowledge transfer that are inherent in human teams. In order to achieve true team-based learning and collaboration, models are needed for how humans cognitively learn, interact, and teach one another. New and novel machine learning methods also need to be developed for information sharing and collaboration. Incorporating agent trust, confidence, and negotiation, along with using such concepts to intelligently reason to arrive at a collective decision, may allow for higher accuracy classification or task performance. Investigating other next-generation

learning architectures, such as deep learning (incorporate both substance and meaning) and hierarchical systems, is also important to advance the state-of-the-art in multi-agent machine learning.

Some real-world applications are restricted by limitations of computing power, working memory, and decision time. Additional future work could be performed with focus on groups of classifiers that have various limitations on memory, time, and processing. General heuristics could be produced for specific system constraints for real-world problems. For example, missile defense or target classification requires real-time classification. Space applications also have restrictions such as computation power and memory, due to the radiation-hardening constraint on spaceflight processors and systems. Such problems cannot always leverage traditional team learning and conventional machine learning algorithms. Similarly, research on dynamic switching of algorithm based on performance may provide further insight into situation-based learning and classification.

Embodying human-level learning ability and intelligent reasoning into mobile robots is a significant goal of robotics. Robotic learning is currently in its infancy compared to the field of machine learning, primarily due to the fact that robots must sense, perceive, and act in a continuous dynamic world. Extending learning to mobile robots can be achieved by incorporating one or more machine learning algorithms into the decision architecture of the robot. As the robot moves about in and interacts with its environment, machine learning algorithms can be internally trained and tested [123]. Similar systems exist on autonomous platforms like the Mars Exploration Rovers, where algorithms have been implemented to automatically detect dust devils in surface imagery [30] and autonomously infer science onboard a rover [105]. By incorporating machine learning onto these platforms, it may be possible to extend rover lifetime and utility while increasing autonomy for rovers performing deep space missions where communication is lagged.

Finally, abstract concepts such as negotiation and modeling intent are interesting areas of further study. For example, an approach we call "intention sharing" could be investigated, where robots could communicate their intentions or expected behavior pattern for the near future based on their current state. Robots could then model teammate behavior, and be able to better perform a task or react accordingly. A natural extension of pure software machine learning team studies would be to perform similar studies with teams of cooperative, learning mobile robots.

Bibliography

[1] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-Based Learning Algorithms," *Machine Learning*, vol. 6, pp. 37–66, 1991.

[2] M. N. Ahmadabadi, M. Asadpour, and E. Nakano, "Cooperative Q-learning: The Knowledge Sharing Issue," *Advanced Robotics*, vol. 15, no. 8, pp. 815–832, 2001.

[3] M. N. Ahmadabadi, M. Asadpur, S. H. Khodaabakhsh, and E. Nakano, "Expertness Measuring in Cooperative Learning," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3. Takamatsu, Japan: IEEE/RSJ, 2000, pp. 2261–2267.

[4] E. L. Akers, H. P. Harmon, R. S. Stansbury, and A. Agah, "Design, Fabrication, and Evaluation of a Mobile Robot for Polar Environments," in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS).* Anchorage, Alaska: Proceedings of the IEEE International Geoscience and Remote Sensing Symposium, September 2004, pp. 109–112.

[5] E. L. Akers, R. S. Stansbury, and A. Agah, "Long-term Survival of Polar Mobile Robots," in *Proceedings of the 4th International Conference on Computing, Communications and Control Technologies (CCCT)*, vol. II, Orlando, FL, July 2006, pp. 329–333.

[6] E. L. Akers, R. S. Stansbury, A. Agah, and T. L. Akins, "Mobile Robots for Harsh Environments: Lessons Learned from Field Experiments," in *Proceedings of the 11th International Symposium on Robotics and Applications (ISORA)*, Budapest, Hungary, July 2006, pp. 1–6.

[7] A. Al-Ani and M. Deriche, "A New Technique for Combining Multiple Classifiers using The Dempster-Shafer Theory of Evidence," *Journal of Artificial Intelligence Research*, vol. 17, pp. 333–361, November 2002.

[8] S. A. A. Al-Faraj, "Reservoir Formation Facies Identification using Decision Tree Learning," Master's thesis, King Fahd University of Petroleum and Minerals, 1998.

[9] G. T. Albanis and R. A. Batchelor, "Combining Heterogeneous Classifiers for Stock Selection," *International Journal of Intelligent Systems in Accounting and Finance Management*, vol. 15, no. 1, pp. 1–21, January 2007.

[10] K. M. Ali and M. J. Pazzani, "Error Reduction through Learning Multiple Descriptions," *Machine Learning*, vol. 24, no. 3, pp. 173–202, 1996.

[11] E. Alpaydin, "Techniques for Combining Multiple Learners," in *Proceedings of the Engineering of Intelligent Systems Conference*, vol. 2. Tenerife, Spain: ICSC Press, 1998, pp. 6–12.

[12] R. C. Arkin, Y. Endo, B. Lee, D. MacKenzie, and E. Martinson, "Multistrategy Learning Methods for Multirobot Systems," Defense Technical Information Center OAI-PMH Repository (United States), Tech. Rep., 1998.

[13] R. C. Arkin, Y. Endo, B. Lee, D. C. MacKenzie, , and E. Martinson, "Multistrategy Learning Methods for Multirobot Systems," in *Multi-Robot Systems: From Swarms to Intelligent Automata (Proceedings Second International Workshop on Multi-Robot Systems)*, vol. 2. Springer, 2003, pp. 137–150.

[14] D. Bahler and L. Navarro, "Methods for Combining Heterogeneous Sets of Classifiers," in *Proceedings of the National Conference on Artificial Intelligence Workshop, New Research Problems for Machine Learning*, 2000, pp. 5–10.

[15] T. Balch, "Learning Roles: Behavioral Diversity in Robot Teams," Georgia Institute of Technology, Tech. Rep. GIT-CC-97-12, 1997.

[16] T. Balch, "Behavioral Diversity in Learning Robot Teams," Ph.D. Thesis, College of Computing, Georgia Institute of Technology, 1998.

[17] T. Balch and A. Ram, "Integrating Robotics Research with JavaBots," in *Proceedings of the AAAI Spring Symposium*. Stanford, CA, USA: AAAI, 1998.

[18] A. Bhatt, "Reservoir Properties from Well Logs using Neural Networks," Ph.D. dissertation, Department of Petroleum Engineering and Applied Geophysics, Norwegian University of Science and Technology, 2002.

[19] A. Bhatt, H. B. Helle, and B. Ursin, "Application of Committee Machines in Reservoir Characterisation While Drilling: A Novel Neural Network Approach in Log Analysis," in *Proceedings of the Nordic Symposium on Petrophysics*, Trondheim, Norway, May 2001, pp. 1–15.

[20] B. Bhattacharya and D. Solomatine, "Machine Learning in Soil Classification," *Neural Networks, Special Issue on Earth Sciences and Environmental Applications of Computational Intelligence*, vol. 19, no. 2, pp. 186–195, March 2006.

[21] G. Bohling and M. Dubois, "An Integrated Application of Neural Network and Markov Chain Techniques to Prediction of Lithofacies from Well Logs, Tech. Rep. 2003-50, 2003.

[22] M. Bowling and M. Veloso, "Simultaneous Adversarial Multi-Robot Learning," in *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, August 2003, pp. 699–704.

[23] M. H. Bowling and M. M. Veloso, "Multiagent Learning Using a Variable Learning Rate," *Artificial Intelligence*, vol. 136, no. 2, pp. 215–250, 2002.

[24] P. Brazdil, M. Gams, S. Sian, L. Torgo, and W. van de Velde, "Learning in Distributed Systems and Multi-Agent Environments," in *Lecture Notes in Artificial Intelligence: Machine Learning-EWSL-91*, 1991, pp. 412–423.

[25] P. Brazdil and L. Torgo, "Knowledge Acquisition via Knowledge Integration," in *Current Trends in AI*. Amsterdam: IOS Press, 1990, pp. 90–104.

[26] P. Brazdil and L. Torgo, "Knowledge Integration and Learning," Laboratory of AI and Computer Science, LIACC, Machine Learning Group, University of Puerto Rico, Tech. Rep. 91.1, 1991.

[27] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, October 2001.

[28] S. L. Buchardt and D. Dahl-Jensen, "Estimating the Basal Melt Rate at NorthGRIP using a Monte Carlo Technique," *Annals of Glaciology*, vol. 45, no. 1, pp. 137–142, October 2007.

[29] S. L. Buchardt and D. Dahl-Jensen, "Predicting the Depth and Thickness of the Eemian Layer at NEEM Using a Flow Model and Internal Layers," in *Geofysikdag*, 2007.

[30] A. Castano, A. Fukunaga, J. Biesiadecki, L. Neakrase, P. Whelley, R. Greeley, M. Lemmon, R. Castano, and S. Chien, "Automatic Detection of Dust Devils and Clouds on Mars," *Machine Vision and Applications*, vol. 19, pp. 467–482, 2008.

[31] R. Castano, D. Mazzoni, N. Tang, T. Doggett, S. Chien, R. Greeley, B. Cichy, and A. Davies, "Learning Classifiers for Science Event Detection in Remote Sensing Imagery," in *Proceedings of the ISAIRAS Conference*, Munich, Germany, September 2005.

[32] R. Castano, D. Mazzoni, N. Tang, R. Greeley, T. Doggett, B. Cichy, S. Chien, and A. Davies, "Onboard Classifiers for Science Event Detection on a Remote Sensing Spacecraft," in *Proceedings of the KDD Conference*, Philadelphia, Pennsylvania, August 2006, pp. 845–851.

[33] S. Celis and D. R. Musicant, "Weka-Parallel: Machine Learning in Parallel," Department of Mathematics and Computer Science, Carleton College, Tech. Rep., 2003, URL: http://sourceforge.net/projects/weka-parallel/.

[34] P. K. Chan and S. J. Stolfo, "Toward Parallel and Distributed Learning by Meta-Learning," in *Working Notes of the AAAI Workshop on Knowledge Discovery in Databases*, 1993, pp. 227–240.

[35] K. Chen, L. Wang, and H. Chi, "Methods of Combining Multiple Classifiers with Different Features and Their Applications to Text-Independent Speaker Identification," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 11, no. 3, pp. 417–445, 1997.

[36] J. G. Cleary and L. E. Trigg, "K*: An Instance-Based Learner Using an Entropic Distance Measure," in *Proceedings of the 12th International Conference on Machine learning*, 1995, pp. 108–114.

[37] W. W. Cohen, "Fast Effective Rule Induction," in *Proceedings of the 12th International Conference on Machine Learning*, 1995, pp. 115–123.

[38] M. T. Cox and A. Ram, "Introspective Multistrategy Learning: On the Construction of Learning Strategies," *Artificial Intelligence*, vol. 112, no. 1-2, pp. 1–55, 1999.

[39] CReSIS, "Center for Remote Sensing of Ice Sheets," 2009. [Online]. Available: http://www.cresis.ku.edu/

[40] P. Darbyshire and D. Wang, "Learning to Survive: Increased Learning Rates by Communication in a Multi-agent System," in *Proceedings of the Australian Conference on Artificial Intelligence*, 2003, pp. 601–611.

[41] T. G. Dietterich, "Ensemble Methods in Machine Learning," *Lecture Notes in Computer Science: Multiple Classifier Systems*, vol. 1857, pp. 1–15, 2000.

[42] R. V. dos Santos, F. Artola, S. da Fontoura, and M. Vellasco, *Lithology Recognition by Neural Network Ensembles*, ser. Lecture Notes in Computer Science: Advances in Artificial Intelligence. Springer-Verlag, 2002, vol. 2507, pp. 302–312.

[43] M. K. Dubois, G. C. Bohling, and S. Chakrabarti, "Comparison of Four Approaches to a Rock Facies Classification Problem," *Computers & Geosciences*, vol. 33, no. 5, pp. 599–617, May 2007.

[44] M. Dubois, G. Bohling, and S. Chakrabarti, "Comparison of Rock Facies Classification Using Three Statistically Based Classifiers, Tech. Rep. 2004-64, 2004.

[45] M. Dubois, A. Byrnes, G. Bohling, S. Seals, and J. Doveton, "Statistically-Based Lithofacies Predictions for 3-D Reservoir Modeling: Examples from the Panoma (Council Grove) Field, Hugoton Embayment, Southwest Kansas," in *Proceedings of the American Association of Petroleum Geologists Annual Convention*, vol. 12, A44, Salt Lake City, Utah, 2003.

[46] S. Dzeroski and B. Zenko, "Is Combining Classifiers Better than Selecting the Best One?" *Machine Learning*, vol. 54, no. 3, pp. 255–273, March 2004.

[47] N. H. R. Ebrahimpour, "Combining Multiple Classifiers: Diversify with Boosting and Combining by Stacking," *International Journal of Computer Science and Network Security*, vol. 7, no. 1, pp. 127–131, January 2007.

[48] R. M. Felder and R. Brent, "Effective Strategies for Cooperative Learning," *Journal of Cooperation and Collaboration in College Teaching*, vol. 10, no. 2, pp. 69–75, 2001.

[49] R. Fikes, M. Cutkosky, T. Gruber, and J. V. Baalen, "Knowledge Sharing Technology Project Overview," Stanford University, Knowledge Systems Laboratory, Tech. Rep. KSL 91-71, November 1991.

[50] E. Frank and I. H. Witten, "Generating Accurate Rule Sets Without Global Optimization," in *Proceedings of the 15th International Conference on Machine Learning*. San Francisco, CA: Morgan Kaufmann, 1998, pp. 144–151.

[51] T. Froehner, M. Nickles, G. Weiß, W. Brauer, and R. Franken, "Integration of Ontologies and Knowledge from Distributed Autonomous Sources," *Journal Kuenstliche Intelligenz (KI)*, vol. 19, no. 1, pp. 18–25, 2005.

[52] H. Gardner, *Frames of Mind: The Theory of Multiple Intelligences*. New York: Basic Books, 1983.

[53] C. M. Gifford, E. L. Akers, R. S. Stansbury, and A. Agah, *Mobile Robots for Polar Remote Sensing*, ser. The Path to Autonomous Robots. Heidelberg, Germany: Springer-Verlag, February 2009, pp. 3–24.

[54] D. Gordon, "A Multistrategy Learning Scheme For Agent Knowledge Acquisition," *Informatica*, vol. 17, no. 4, pp. 331–346, 1993.

[55] A. Großmann and R. Poli, "Continual Robot Learning with Constructive Neural Networks," in *Proceedings of the Sixth European Workshop on Learning Robots*. London, UK: Springer-Verlag, 1998, pp. 95–108.

[56] C. R. Haller, V. J. Gallagher, T. L. Weldon, and R. M. Felder, "Dynamics of Peer Education in Cooperative Learning Workgroups," *Journal of Engineering Education*, vol. 89, no. 3, pp. 285–293, 2000.

[57] H. P. Harmon, R. S. Stansbury, E. L. Akers, and A. Agah, "Sensing and Actuation for a Polar Mobile Robot," in *Proceedings of the International Conference on Computing, Communications and Control Technologies (CCCT)*, vol. IV, August 2004, pp. 371–376.

[58] T. Haynes and S. Sen, "Evolving Behavioral Strategies in Predators and Prey," in *(IJCAI)-95 Workshop on Adaptation and Learning in Multiagent Systems*, 1995, pp. 32–37.

[59] T. Haynes and S. Sen, "Cooperation of the Fittest," in *Late Breaking Papers at the Genetic Programming Conference, Stanford University*, July 1996, pp. 47–55.

[60] T. Haynes, S. Sen, D. Schoenefeld, and R. Wainwright, "Evolving a Team," in *Working Notes for the AAAI Symposium on Genetic Programming*. AAAI, 1995, pp. 23–30.

[61] D. Hitchcock, "Dispelling Myths about Teams," *AXIS Advisory*, Spring 2000.

[62] H. Hu, I. Kelly, D. A. Keating, and D. Vinagre, "Coordination of Multiple Mobile Robots via Communication," in *Proceedings of the SPIE Mobile Robots XIII and Intelligent Transportation Systems*, vol. 3525. Boston, MA, USA: SPIE, January 1999, pp. 94–103.

[63] Y. Huang and C. Sueng, "A Method of Combining Multiple Experts for the Recognition of Unconstrained Handwritten Numerals," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 405–410, April 1997.

[64] G. H. John and P. Langley, "Estimating Continuous Distributions in Bayesian Classifiers," in *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann, 1995, pp. 338–345.

[65] D. W. Johnson, R. T. Johnson, and M. B. Stanne, "Cooperative Learning Methods: A Meta-Analysis," *Methods of Cooperative Learning: What Can We Prove Works*, 2000. [Online]. Available: www.clcrc.com

[66] U. Kartoun, H. Stern, Y. Edan, C. Feied, J. Handler, M. Smith, and M. Gillam, "Collaborative Q(lambda) Reinforcement Learning Algorithm - A Promising Robot Learning Framework," in *Proceedings of the IASTED International Conference on Robotics and Applications*, Oct. 31 - Nov. 2 2005, pp. 13–19.

[67] I. Kawaishi, S. Yamada, and J. Toyoda, "Experimental Comparison of a Heterogeneous Learning Multi-Agent System with a Homogeneous One," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*. Beijing, China: IEEE, 1996, pp. 613–618.

[68] D. A. Keating and I. D. Kelly, "Increasing Mobile Robot Learning Rates Through Sharing of Experiences," in *Proceedings of the International Conference on Control*. Swansea, Wales: IEEE, September 1998, pp. 1664–1669.

[69] I. D. Kelly, "The Development of Shared Experience Learning in a Group of Mobile Robots," Ph.D. Thesis, University of Reading, 1997.

[70] I. D. Kelly, D. A. Keating, and K. Warwick, "Mutual Learning by Autonomous Mobile Robots," in *Proceedings of the 1st Workshop on Telepresence and Robotics Applications in Science and Art*, Linz, Austria, June 1997, pp. 103–116.

[71] I. D. Kelly and D. A. Keating, "Faster Learning of Control Parameters Through Sharing Experiences of Autonomous Mobile Robots," *International Journal of Systems Science, Special Issue on Developments in Intelligent Control Systems*, vol. 29, no. 7, pp. 783–793, 1998.

[72] KGS, "Kansas Geological Survey," University of Kansas, 2009. [Online]. Available: http://www.kgs.ku.edu/

[73] J. H. Kim, K. K. Kim, and C. Y. Suen, "Hybrid Schemes of Homogeneous and Heterogeneous Classifiers for Cursive Word Recognition," in *Proceedings of the International Workshop on Frontiers in Handwriting Recognition*, Amsterdam, September 2000, pp. 433–442.

[74] D. Klein, K. Toutanova, H. T. Ilhan, S. D. Kamvar, and C. D. Manning, "Combining Heterogeneous Classifiers for Word-Sense Disambiguation," in *Proceedings of the ACL Workshop on Word Sense Disambiguation: Recent Successes and Future Directions*, vol. 8, 2002, pp. 74–80.

[75] R. Kohavi, "The Power of Decision Tables," in *Proceedings of the European Conference on Machine Learning*, 1995, pp. 174–189.

[76] R. E. Korf, "A Simple Solution to Pursuit Games," in *Working Papers of the 11th International Workshop on Distributed Artificial Intelligence*, February 1992, pp. 183–194.

[77] S. Kotsianti and D. Kanellopoulos, *Combining Bagging, Boosting and Dagging for Classification Problems*, ser. Lecture Notes in Computer Science. Springer-Verlag, 2007, pp. 493–500.

[78] L. I. Kuncheva, "Combining Classifiers: Soft Computing Solutions," *S.K. Pal and A. Pal (Ed.), Pattern Recognition: From Classical to Modern Approaches, World Scientific*, pp. 427–451, 2001.

[79] L. I. Kuncheva, "Switching Between Selection and Fusion in Combining Classifiers: An Experiment," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 32, no. 2, pp. 146–156, April 2002.

[80] D. LaBelle, J. Bares, and I. Nourbakhsh, "Material Classification by Drilling," in *Proceedings of the International Symposium on Robotics and Automation in Construction*, Taipei, Taiwan, September 2000.

[81] S. Lander, V. R. Lesser, and M. E. Connell, "Conflict Resolution Strategies for Cooperating Expert Agents," in *Proceedings of the International Working Conference on Cooperating Knowledge Based Systems*. Springer-Verlag, 1991, pp. 183–200.

[82] S. le Cessie and J. van Houwelingen, "Ridge Estimators in Logistic Regression," *Applied Statistics*, vol. 41, no. 1, pp. 191–201, 1992.

[83] L. Li, A. Martinoli, and Y. S. Abu-Mostafa, "Learning and Measuring Specialization in Collaborative Swarm Systems," *Adaptive Behavior - Animals, Animats, Software Agents, Robots, Adaptive Systems*, vol. 12, no. 3-4, pp. 199–212, 2004.

[84] Y. Liu and M. D. Sacchi, "Mapping Rock Mechanical Properties with Seismic Attribute-based Support Vector Machine (SVM) Technique," in *Proceedings of the CSPG/CSEG Joint Convention*, 2003.

[85] A. Lohoefener, "Design and Development of a Multi-Channel Radar Depth Sounder," Master's Thesis, Department of Electrical Engineering and Computer Science, University of Kansas, Lawrence, KS, November 2006.

[86] M. J. Mataric, "Interaction and Intelligent Behavior," Ph.D. Thesis, Massachusetts Institute of Technology, 1994.

[87] M. Mataric, "Learning in Multi-Robot Systems," *Adaption and Learning in Multi-Agent Systems*, pp. 152–163, 1996.

[88] S. Moysey, R. Knight, H. Jol, R. Allen-King, and D. Gaylord, "Quantification of Reflection Patterns in Ground-Penetrating Radar Data," *Eos Trans. AGU*, vol. 86, no. 52, Fall Meet. Suppl., Abstract U23C-04 2005.

[89] National Climatic Data Center (NCDC), "The GRIP Ice Coring Effort," 2009. [Online]. Available: http://www.ncdc.noaa.gov/paleo/icecore/greenland/summit/document/

[90] Netflix, Inc., "Netflix Prize." [Online]. Available: http://www.netflixprize.com//index

[91] G. Oswald and S. Gogineni, "Recovery of Subglacial Water Extent from Greenland Radar Survey Data," *Journal of Glaciology*, vol. 54, no. 184, pp. 94–106, January 2008.

[92] L. Panait and S. Luke, "Cooperative Multi-Agent Learning: The State of the Art," *Autonomous Agents and Multi-Agent Systems*, vol. 11, no. 48, pp. 387–434, November 2005.

[93] L. Parker, "A Case Study for Life-long Learning and Adaptation in Cooperative Robot Teams," in *Proceedings of the SPIE Sensor Fusion and Decentralized Control in Robotic Systems II*, vol. 3839, 1999, pp. 92–101.

[94] L. E. Parker, "Lifelong Adaptation in Heterogeneous Multi-Robot Teams: Response to Continual Variation in Individual Robot Performance," *Autonomous Robots*, vol. 8, no. 3, pp. 239–267, 2000.

[95] L. E. Parker, M. J. Mataric, M. Veloso, and T. Balch, "Status of Robotics in the U.S. Workshop," in *National Science Foundation*, Washington, D.C., July 2004.

[96] L. E. Parker, C. Touzet, and F. Fernandez, *Techniques for Learning in Multi-Robot Teams*. A K Peters, 2002, pp. 191–236.

[97] J. Payne-Anderson, "Training and Learning in Teams," University of North Texas, 1998. [Online]. Available: http://www.workteams.unt.edu/old/reports/anderson.html

[98] M. E. Peters, D. D. Blankenship, D. E. Smith, J. W. Holt, and S. D. Kempf, "The Distribution and Classification of Bottom Crevasses from Radar Sounding of a Large Tabular Iceberg," *IEEE Geoscience and Remote Sensing Letters*, vol. 4, no. 1, pp. 142–146, January 2007.

[99] D. Prebble and H. Frederick, "10 Ways to Distinguish between a Team and a Group: Molding Your People into a Pro-active and Productive Team," Australian Business Limited and Australian Commonwealth ITR Department. [Online]. Available: http://www.1000ventures.com/business_guide/crosscuttings/team_vs_group.html

[100] F. J. Provost and D. N. Hennessy, "Scaling Up: Distributed Machine Learning with Cooperation," in *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1996, pp. 74–79.

[101] F. J. Provost and D. N. Hennessy, "Scaling Up: Distributed Machine Learning with Cooperation," in *Proceedings of the National Conference on Artificial Intelligence*, 1996, pp. 74–79.

[102] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.

[103] P. Ritthipravat, T. Maneewarn, and D. Laowattana, "Sharing Learning Policies between Multiple Mobile Robots," in *Proceedings of the International Conference on Integration of Knowledge Intensive Multi-Agent Systems*. Boston, MA, USA: IEEE, October 2003, pp. 616–621.

[104] G. L. Rogova, M. I. Bursik, and S. Hanson-Hedgecock, "Interpreting the Pattern of Volcanic Eruptions: Intelligent System for Tephra Layer Correlation," in *Proceedings of the International Conference on Information Fusion*, July 2007, pp. 1–7.

[105] T. Roush, M. Shipman, R. Morris, P. Gazis, and L. Pedersen, "Essential Autonomous Science Inference in Rovers (EASIR)," in *Proceedings of the IEEE Aerospace Conference*, vol. 2, Big Sky, MT, 2004, pp. 790–800.

[106] M. Saerens and F. Fouss, "Yet Another Method for Combining Classifiers Outputs: A Maximum Entropy Approach," in *Multiple Classifier Systems*, 2004, pp. 82–91.

[107] M. Saggaf and E. L. Nebrija, "Estimation of Lithologies and Depositional Facies from Wire-Line Logs," *American Association of Petroleum Geologists Bulletin*, vol. 84, no. 10, pp. 1633–1646, October 2000.

[108] R. Santos, M. Vellasco, F. Artola, and S. da Fontoura, *Neural Net Ensembles for Lithology Recognition*, ser. Lecture Notes in Computer Science: Multiple Classifier Systems. Springer-Verlag, 2003, vol. 2709, pp. 246–255.

[109] S. R. Schach, *Object-Oriented and Classical Software Engineering.* Seventh Edition, McGraw-Hill Science/Engineering/Math, 2007.

[110] S. Sen and M. Sekaran, "Multiagent Coordination with Learning Classifier Systems," in *Proceedings of the IJCAI Workshop on Adaption and Learning in Multi-Agent Systems*, vol. 1042, 1996, pp. 218–233.

[111] S. Sen, M. Sekaran, and J. Hale, "Learning to Coordinate without Sharing Information," in *Proceedings of the Twelfth National Conference on Artificial Intelligence.* Seattle, WA, USA: AAAI, 1994, pp. 426–431.

[112] F. Shang, X. Zhang, and T. Zhao, "Application of a Mixed Kernel in the Oil Water-Flooded Layer Identification," *International Journal of Computer Science and Network Security*, vol. 8, no. 1, pp. 102–106, January 2008.

[113] Z. Shi, P. Luo, Y. Hao, G. Li, M. Stumptner, Q. He, and G. Quirchmayr, "Intelligent Technology for Well Logging Analysis," pp. 373–382, 2005.

[114] S. S. Sian, "Adaptation Based on Cooperative Learning in Multi-Agent Systems," *Decentralized Artificial Intelligence 2*, pp. 257–272, 1991.

[115] S. S. Sian, "The Role of Cooperation in Multi-Agent Learning," in *Proceedings of the International Working Conference on Cooperating Knowledge Based Systems.* Berlin, Heidelberg: Springer, 1991, pp. 164–180.

[116] S. S. Sian, "Extending Learning to Multiple Agents: Issues and a Model for Multi-Agent Machine Learning (MA-ML)," in *Proceedings of the European Working Session on Learning.* Porto, Portugal: Springer-Verlag, 1991, pp. 440–456.

[117] K. A. Smith, "Cooperative Learning: Effective Teamwork for Engineering Classrooms," in *Proceedings of the Frontiers in Education Conference*, vol. 1, no. 1-4, 1995, pp. 2b5.13–2b5.18.

[118] L.-K. Soh and C. Tsatsoulis, "ARKTOS: A Knowledge Engineering Software Package for Satellite Sea Ice Classification," in *Proceedings of the International Conference on Geoscience and Remote Sensing Symposium*, 2000, pp. 696–698.

[119] A. Srivastava and J. Stroeve, "Onboard Detection of Snow, Ice, Clouds and Other Geophysical Processes Using Kernel Methods," in *Proceedings of the Workshop on Machine Learning Technologies for Autonomous Space Applications*, 2003.

[120] R. S. Stansbury, E. L. Akers, H. P. Harmon, and A. Agah, "Survivability, Mobility, and Functionality of a Rover for Radars in Polar Regions," *International Journal of Control, Automation, and Systems*, vol. 2, no. 3, pp. 334–353, 2004.

[121] B. M. Steele, "Combining Multiple Classifiers: An Application Using Spatial and Remotely Sensed Information for Land Cover Type Mapping," *Remote Sensing of Environment*, vol. 74, pp. 545–556, 2000.

[122] P. Stone and M. Veloso, "Multiagent Systems: A Survey from a Machine Learning Perspective," *Autonomous Robots*, vol. 8, no. 3, pp. 345–383, July 2000.

[123] S. Takamuku and R. C. Arkin, "Multi-Method Learning and Assimilation," *Robotics and Autonomous Systems*, vol. 55, no. 8, pp. 618–627, 2007.

[124] V. Tamma, "An Ontology Model Supporting Multiple Ontologies for Knowledge Sharing," Ph.D. Thesis, The University of Liverpool, 2002.

[125] M. Tan, "Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents," in *Proceedings of the Tenth International Conference on Machine Learning*. Amherst, MA, USA: Morgan Kaufmann Publishers, 1993, pp. 330–337.

[126] M. Tan, "Multi-Agent Reinforcement Learning: Independent vs. Cooperative Learning," in *Readings in Agents*. Morgan Kaufmann, 1997, pp. 487–494.

[127] P. Tangamchit, J. Dolan, and P. Kosla, "The Necessity of Average Rewards in Cooperative Multirobot Learning," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, 2002, pp. 1296–1301.

[128] P. Tangamchit, J. Dolan, and P. Khosla, "Crucial Factors Affecting Cooperative Multirobot Learning," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, October 2003, pp. 2023–2028.

[129] D. Tartakovsky and B. Wohlberg, "Delineation of Geologic Facies with Statistical Learning Theory," *Geophysical Research Letters*, vol. 31, no. 18, p. L18502, 2004.

[130] D. M. Tax, M. van Breukelen, R. P. Duin, and J. Kittler, "Combining Multiple Classifiers by Averaging or by Multiplying," *Pattern Recognition*, vol. 33, pp. 1475–1485, 2000.

[131] R. Thakur, R. Rabenseifner, and W. Gropp, "Optimization of Collective Communication Operations in MPICH," *International Journal of High Performance Computing Applications*, vol. 19, no. 1, pp. 49–66, 2005.

[132] The MathWorks, "MATLAB - The Language of Technical Computing," 2006. [Online]. Available: http://www.mathworks.com/products/matlab/

[133] D. R. Thompson, S. Niekum, T. Smith, and D. Wettergreen, "Automatic Detection and Classification of Geological Features of Interest," in *Proceedings of the IEEE Aerospace Conference*, Big Sky, Montana, March 2005, pp. 1–12.

[134] D. R. Thompson, T. Smith, and D. Wettergreen, "Data Mining During Rover Traverse: From Images to Geologic Signatures," in *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space*, September 2005, pp. 1–8.

[135] K. Ting and B. Low, "Theory Combination: An Alternative to Data Combination, Tech. Rep. 96/19, 1996.

[136] C. F. Touzet, "Robot Awareness in Cooperative Mobile Robot Learning," *Autonomous Robots*, vol. 8, no. 1, pp. 87–97, 2000.

[137] C. F. Touzet, "Distributed Lazy Q-Learning for Cooperative Mobile Robots," *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, pp. 5–13, 2004.

[138] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Effective Voting of Heterogeneous Classifiers," in *Proceedings of the European Conference on Machine Learning*, September 2004, pp. 465–476.

[139] G. Tsoumakas, L. Angelis, and I. Vlahavas, "Selective Fusion of Heterogeneous Classifiers," *Intelligent Data Analysis*, vol. 9, no. 6, pp. 511–525, November 2005.

[140] K. Tumer and J. Ghosh, "Error Correlation and Error Reduction in Ensemble Classifiers," *Connection Science*, vol. 8, no. 3, pp. 385–404, December 1996.

[141] E. Uchibe and K. Doya, "Reinforcement Learning with Multiple Heterogeneous Modules: A Framework for Developmental Robot Learning," in *Proceedings of the 4th International Conference on Development and Learning*, 2005, pp. 87–92.

[142] University of Copenhagen, "NGRIP - North Greenland Ice Core Project," Niels Bohr Institute, 2009. [Online]. Available: http://www.gfy.ku.dk/~www-glac/ngrip/index_eng.htm

[143] M. van der Baan and C. Jutten, "Neural Networks in Geophysical Applications," *Geophysics*, vol. 65, no. 4, pp. 1032–1047, August 2000.

[144] C. Watkins, "Learning from Delayed Rewards," Ph.D. Thesis, King's College, Cambridge, UK, 1989.

[145] C. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[146] G. Weiß, "Adaptation and Learning in Multi-Agent Systems: Some Remarks and a Bibliography," in *Adaptation and Learning in Multi-Agent Systems*. Springer-Verlag, 1996, pp. 1–21.

[147] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd ed. Morgan Kaufmann, San Francisco, 2005.

[148] B. Wohlberg, D. Tartakovsky, and A. Guadagnini, "Subsurface Characterization with Support Vector Machines," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 1, pp. 47–57, 2006.

[149] D. H. Wolpert and W. G. Macready, "No Free Lunch Theorems for Optimization," *IEEE Transactions on Evolutionary Computing*, pp. 67–82, 1997.

[150] P. Wong, T. Gedeon, and I. Taggart, "An Improved Technique in Porosity Prediction: A Neural Network Approach," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 33, no. 4, pp. 971–980, July 1995.

[151] K. Woods, W. P. K. Jr., and K. Bowyer, "Combination of Multiple Classifiers Using Local Accuracy Estimates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 405–410, April 1997.

[152] L. Xu, A. Krzyzak, and C. Y. Suen, "Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 3, pp. 418–435, May/June 1992.

[153] E. Yang and D. Gu, "Multiagent Reinforcement Learning for Multi-Robot Systems: A Survey," Department of Computer Science, University of Essex, Tech. Rep. SS-96-01, 2004.

[154] H. Zou and F. Yang, *Study on Signal Interpretation of GPR Based on Support Vector Machines*, ser. Lecture Notes in Computer Science: Bio-Inspired Computational Intelligence and Applications. Springer-Verlag, 2007, vol. 4688, pp. 533–539.

[155] X. Zuo, "High Level Support for Distributed Computation in WEKA," Master of Computational Science, University College, Dublin, Ireland, 2004, URL: http://userweb.port.ac.uk/ khusainr/weka/index.html.