Imperial College London
Department of Electrical and Electronic Engineering

The Diversity-Accuracy Duality in Ensembles of Classifiers

Rui Pedro Peixoto Cardoso

Supervisor: Prof. Jeremy V. Pitt

July 2022

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Electrical and Electronic Engineering of Imperial College London and
the Diploma of Imperial College London

Abstract

Horizontal scaling of Machine Learning algorithms has the potential to tackle concerns over the scalability and sustainability of Deep Learning methods, viz. their consumption of energy and computational resources, as well their increasing inaccessibility to researchers. One way to enact horizontal scaling is by employing ensemble learning methods, since they enable distribution. There is a consensus on the point that *diversity* between individual learners leads to better performance, which is why we have focused on it as the criterion for distributing the base models of an ensemble. However, there is no standard agreement on how diversity should be defined and thus how to exploit it to construct a high-performing classifier. Therefore, we have proposed different definitions of diversity and innovative algorithms which promote it in a *systematic way.*

We have first considered *architectural diversity* with an algorithm called WILDA: Wide Learning of Diverse Architectures. In a distributed fashion, this algorithm evolves a set of neural networks that are pretrained on the target task and diverse w.r.t. architectural feature descriptors. We have then generalised this notion by defining *behavioural* diversity on the basis of the divergence between the *errors* made by different models on a dataset. We have defined several *diversity metrics* and used them to guide a *novelty search* algorithm which builds an ensemble of behaviourally diverse classifiers. The algorithm promotes diversity in ensembles by explicitly searching for it, without selecting for accuracy. We have then extended this approach with a *surrogate diversity model*, which reduces the computational burden of this search by eliminating the need to train each network in the population with stochastic gradient descent at each step. These methods have enabled us to investigate the role that both architectural and behavioural diversity play in contributing to the performance of an ensemble.

In order to study the relationship between diversity and accuracy in classifier ensembles, we have then proposed several methods that extend the novelty search with accuracy objectives. Surprisingly, we have observed that, with the highest-performing diversity metrics, there is an *equivalence* between searching for diversity objectives and searching for accuracy objectives. This contradicts widespread assumptions that a trade-off must be found by balancing diversity and accuracy objectives. We therefore posit the existence of a *diversity-accuracy duality* in ensembles of classifiers. An implication of this is the possibility of evolving diverse ensembles without detriment to their accuracy, since it is implicitly ensured.

Statement of Originality

I hereby declare that the work presented in this thesis is my own. Where others' work has been used in any way, this has been duly acknowledged and referenced.

Copyright Declaration

Acknowledgements

The experience of the last four years as a PhD student has been life-changing and I would like to take this opportunity to express my gratitude to all those without whom none of this would have been possible.

To Prof. Jeremy Pitt, I am grateful for the opportunity to work with him and for my professional and — of course! — political growth. Thank you for all the illuminating discussions, all the fun chats, and for giving me plenty of freedom as a researcher. A PhD is as much a mental endeavour as it is a technical one. At times, it leaves us feeling lonely, inadequate, with all sorts of doubts and insecurities. However, I have always felt your unwavering support, even more so throughout the challenges posed by the pandemic. You have always pushed me in the right direction and helped me keep things in perspective. I cannot imagine going through this process without you as my supervisor. I am certain that this is not the end of our work together, but merely the beginning.

To Prof. Emma Hart, I am thankful for so kindly agreeing to be my unofficial second supervisor. I admire your technical expertise, your sharpness, and your ability to communicate ideas ever so clearly. Thank you for always helping me cut through the fog and get to the heart of things. I doubt I would have published half of my papers without your help and I do hope we can keep working together.

To my dear friend David Kurka[1], I am thankful for his collaboration and support throughout the last four years. I am sorry that life has put an ocean between us, but this is not the end of our story!

To all my friends from the Intelligent Systems and Networks group — the old and the new, those who were there throughout the whole thing, those who were just passing by, those who have arrived in the meantime, and those who have left already — I am grateful for all the fun moments, the happy memories, and for making me feel at home since the day I first set foot in that lab as a visiting MSc student four years ago. Thank you for showing this shy boy from North Portugal that there was a whole world beyond the end of his nose.

[1]David is the only friend I am naming here because he has worked with me during my PhD and is a co-author of several of my papers. I am very fortunate to be able to say that, were I to mention all the special people in my life, this would quickly become unmanageable.

To my friends and comrades of the IMT, I am thankful for showing to me with remarkable clarity what is to be done and for helping me find a deep-rooted sense of purpose. A special mention must be made of those comrades who welcomed me as a member in the (old) Kensington branch in 2020 and who have built with me the new Kensington branch and the youth work at Imperial College. I am certain that together we will change the world!

To all my other friends, whom I do not have to name because it is so obvious, I am thankful for all that which is hard to put into words. I feel that our friendship has so far defeated the odds and only become stronger and tighter with the passing years.

Of course, last but not least, I thank my family for their unconditional love and support throughout my life and for making me the man I am today.

‘The philosophers have only interpreted the world in various ways. The point, however, is to change it.’

Karl Marx

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Deep Learning (DL) techniques have revolutionised the field of Machine Learning (ML) in recent years, providing high-performing neural network models for solving a number of complex tasks, such as Computer Vision (CV) and Deep Reinforcement Learning (Deep RL). However, concerns have been raised over the scalability of these methods, as well as their consumption of computational resources. The best performing networks often have very large numbers of layers and millions of parameters, exhibiting a type of scaling that has been mostly vertical. Some DL algorithms run for days [SXZ$^+$20b, SWX$^+$19]. This raises important questions regarding the environmental sustainability of DL methods [SGM19]. In addition, the computational power required to run these algorithms also compromises their *democratic accessibility* to researchers [AW20]. Therefore, the main motivation behind our work has been the need for DL to be complemented with a paradigm that we call Wide Learning (WL), promoting horizontal scaling. *Distributed* learning techniques have been proposed as a way of enacting *horizontal* scaling [XHXD15]. Since ensemble methods enable distribution, they are especially suited to this task.

Ensemble models are capable of outperforming their individual learners w.r.t. predictive accuracy by taking an average of the individual predictions, thereby decreasing the variance of the final prediction. They can often be much more accurate than the individual classifiers they encompass if they are sufficiently *diverse* [Die00]; their individual members do not themselves have to be complex or high-performing [HP04]. Because of this, we have focused upon diversity as the *criterion* for distributing the base models of an ensemble. Within the context of Evolutionary Computing (EC) and Genetic Programming (GP), explicitly searching for *diversity*, i.e. evolutionary algorithms which favour divergence of solutions rather than convergence, has been found to be beneficial in some applications as it addresses issues with local optima and function plateaus (e.g. [LS08, LS11a, Mou11, LS11b, MC15]). Diversity can be measured in terms of a distance metric between solutions, as in [LS11a], or with respect to a set of *diversity features*, as in [MC15]. However, in the context of ensemble learning, there is no standard agreement on how diversity should be defined and thus how to exploit it to construct a high-performing classifier.

1.2 Research Programme

With the overarching goal of systematising the new Wide Learning paradigm, we sought to develop methods which systematically promote diversity amongst individual neural network architectures, serving as a criterion for distributing and constructing a high-performing ensemble of classifiers. When considering neural network models, one type of diversity is *architectural diversity*. We studied this type of diversity with an algorithm called WILDA (Wide Learning of Diverse Architectures) [CHP20, CHKP21a], which integrates ML and Evolutionary Computation (EC) to train a classification model. This approach first trains an ensemble of low-complexity artificial neural networks (ANNs), each of which extracts a feature vector from each data point. In a second phase, the features extracted from the ensemble are aggregated in a single shallow model to solve the classification task. In order to promote architectural diversity, we turned to a relatively recent class of algorithms from EC known as *quality diversity* (QD) algorithms [PSS16]. These algorithms return an archive of diverse, high-quality solutions to a problem in a single run, where diversity is defined by the user with respect to features of interest. In a distributed fashion, this algorithm exploits the QD approach to evolve a set of neural networks that are pretrained on the target task and diverse w.r.t. architectural feature descriptors. Each neural network extracts a feature vector that can be used in the final classification. After concatenating all the feature vectors extracted by the models in the ensemble, a single shallow neural network is trained using only these as input. The goal behind this algorithm was to investigate the role that architectural diversity plays in contributing to the performance of the ensemble.

Another type of diversity is *behavioural* diversity, which can be defined on the basis of the divergence between the *errors* made by different models on a dataset. Another goal of this research programme was to study the influence of behavioural diversity on ensemble accuracy and to compare it with architectural diversity. Because there is no standard definition for behavioural diversity, we proposed several error diversity metrics. These metrics were then used to guide a novelty search (NS) [LS11a] algorithm to build an ensemble of behaviourally diverse classifiers, searching over a space of neural network architectures by maximising a novelty score defined by the behavioural metric w.r.t to the other individuals in the population. Each individual network discovered is optimised using stochastic gradient descent (SGD) on a training dataset before its novelty score is calculated. Networks with high-scoring novelty are iteratively added to an archive which forms the final ensemble. The method therefore explicitly *searches* for diversity amongst learners. Diversity drives not only this search, but also the construction of the final ensemble.

With the goal of reducing the computational burden of the search, we then improved on this novelty search (NS) procedure with a *surrogate diversity model* that eliminates the need for training each neural network with stochastic gradient descent at each iteration. Another goal of our research has been to study the relationship between ensemble diversity and individual model accuracy. With that in mind, we extended the NS with several methods which combine diversity and accuracy objectives, ranging from favouring only diversity to favouring only accuracy.

1.3 Structure

Figure 1.1 illustrates the structure of this thesis and the flow between the chapters. Chapter 2 critically reviews the background literature relevant to our research programme, focusing on the limitations of typical Deep Learning methods, ensemble learning as a technique enacting the Wide Learning paradigm, and diversity as a potential criterion for distributing wide models. The result of this analysis informs our first attempt at tackling the problem of creating diversity *systematically*. In Chapter 3, we propose an algorithm called WILDA, which evolves an *architecturally* diverse ensemble. This was a first proof of concept, which we applied to simple problem domains. We then generalised this approach with a novelty search (NS) algorithm for creating explicit *behavioural* diversity, described in Chapter 4. Although this led to promising results, the algorithm was very computationally intensive because it required training all the neural networks in the current population with gradient descent at every step of the search in order to calculate diversity values. Chapter 5 explains how we improved on the NS approach with a *surrogate model* which estimates diversity, reducing the computational burden of the search. We then studied the influence of adding accuracy objectives to the NS with different methods combining diversity and accuracy, described in Chapter 6. It was as a result of the work presented in that chapter that the surprising observation was made of the equivalence between searching for diversity and accuracy objectives under certain conditions, i.e. the diversity-accuracy duality. Chapter 7 summarises the work reported in this thesis and points towards future research.

1.4 Contributions

The contributions of this thesis are twofold. We made *specific* contributions with innovative methods for *explicitly* and *systematically* searching for, and thereby creating, diversity. This included a number of *diversity metrics*, several algorithms, and their implementation — so that an empirical analysis could be carried out. This enabled us to investigate the role that both architectural and behavioural diversity play in contributing to the performance of an

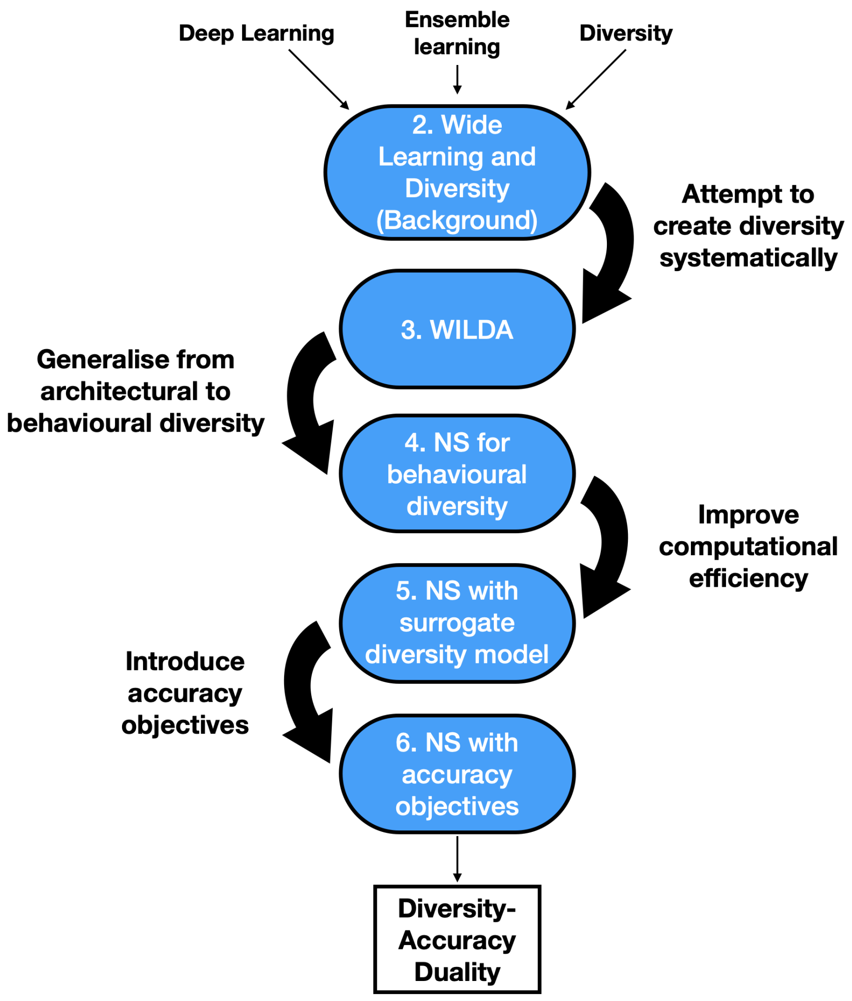

Deep Learning
Ensemble learning
Diversity
2. Wide Learning and Diversity (Background)
Attempt to create diversity systematically
3. WILDA
Generalise from architectural to behavioural diversity
4. NS for behavioural diversity
Improve computational efficiency
5. NS with surrogate diversity model
Introduce accuracy objectives
6. NS with accuracy objectives
Diversity-Accuracy Duality

Figure 1.1: Thesis structure and flow

ensemble. The experimental results show that running the NS with the *error diversity* metrics we proposed leads to higher-performing ensembles than other metrics commonly used in the literature and that the ensembles generated in this way significantly outperform those that use either only implicit measures to encourage diversity or random search approaches that simply reward it. By extending the NS with accuracy objectives, we contributed with insights into the relationship between diversity and accuracy in classifier ensembles.

Additionally, there is an *overall contribution* that emerges holistically from the work developed throughout this research programme: the notion of a *diversity-accuracy duality* in the ensembles evolved by our methods, under certain conditions and for the problem domains considered here. Following widespread assumptions, we were expecting that a trade-off must be found by balancing diversity and accuracy objectives. We were also expecting that improving individual model accuracy would lead to decreased diversity and vice-versa. Surprisingly, however, we observed that, for the highest-performing diversity metrics, there was an *equivalence* between searching for diversity objectives and searching for accuracy objectives, *using our generic search method*. While a further study of this equivalence is required in order for us to be able to draw stronger conclusions, this result is nevertheless significant because it challenges notions about the need to trade off diversity for accuracy and instead suggests that the two are interchangeable and correlated in particular conditions. An implication of this is the possibility of designing better algorithms which evolve diverse ensembles without detriment to their accuracy, since it is implicitly ensured.

This research programme contributed with several publications. The work presented in Chapter 3 resulted in publications in GECCO 2020 [CHP20] and EvoStar 2021 [CHKP21a]. The work of Chapter 4 was published in GECCO 2021 [CHKP21b]. Chapter 5 resulted in a paper in EvoStar 2022 [CHKP22], which was awarded best paper in the Evolutionary Machine Learning (EML) track. Finally, a paper sharing the results of Chapter 6 has been accepted at GECCO 2022.

Chapter 2

Wide Learning and Diversity

2.1 Introduction

In this chapter, we review background work relevant to this research programme. In Section 2.2, we discuss the achievements and shortcomings of Deep Learning (DL) techniques and compare them to what we call Wide Learning, i.e. methods which scale horizontally, growing wider rather than deeper. In Section 2.3, we review ensemble learning and how this technique readily lends itself to distribution and parallel computing, thus enacting the Wide Learning paradigm. In Section 2.4, we discuss the use of evolutionary methods in ensemble learning, since these methods have been used extensively to evolve *diverse* ensembles, which is precisely the focus of our work. In Section 2.5, we focus on a particular class of evolutionary algorithms called Quality Diversity algorithms, which are applied in a number of problem domains to search for solutions which are simultaneously diverse and accurate. In Section 2.6, we review current notions about the relationship between diversity and individual model accuracy in ensembles of classifiers and the trade-off between the two with regard to final ensemble performance. Finally, in Section 2.7, we review the use of surrogate modelling in Evolutionary Computing (EC) and in neural architecture search (NAS), since we have employed surrogate models throughout our work.

2.2 Deep Learning and Wide Learning

Deep Learning (DL) techniques have revolutionised the field of Machine Learning (ML) in recent years. [LBH15] reviews how DL methods have dramatically improved the state of the art in domains such as speech recognition, Computer Vision (CV), and video and audio processing. [Sch15] explains how deep neural networks are capable of learning better representations than shallow neural networks due to their ability to learn longer, deeper chains of causal links between actions and effects. [LKB+17] survey DL techniques used for medical image analysis, discussing current challenges as well as high-profile success cases where deep neural networks have outperformed human experts. The ability of DL to learn useful representations has made it an important tool for natural language processing (NLP) tasks, as reviewed by [YHPC18], since NLP often requires word *embeddings*. Another highly successful application of DL is object detection. [ZZXW19] review state-of-the-art techniques for generic object detection, salient object detection, face detection, and pedestrian detection. Deep neural networks have also been applied as a policy gradient Reinforcement Learning (RL) technique to the problem of learning to play challenging board games better than humans, with AlphaGo [SHM+16, SSS+17] and AlphaZero [SHS+18]. Convolutional neural networks (CNNs) are the typical design used in deep neural networks, especially in image processing tasks since they are especially suited to deal with two-dimensional shapes, thereby outperforming other techniques [LBBH98a].

Despite all their potential and successes, concerns have been raised over the scalability of these methods, as well as their consumption of computational resources. The best performing deep networks often have very large numbers of layers and millions of parameters, exhibiting a type of scaling that has been mostly vertical. It is not at all uncommon for state-of-the-art DL algorithms to run for days, such as [SXZ+20b, SWX+19], or even weeks, such as the original version of AlphaGo [SHM+16]. Amongst other things, this raises important issues regarding the environmental sustainability of DL methods. Training an instance of some of the most complex DL models can result in CO_2 emissions many times greater than an average car throughout its lifetime [SGM19]. The computational power required to run these algorithms also poses important questions regarding their *democratic accessibility* to researchers. There is a growing divide between the research outputs of large tech companies and that of even elite universities, with knowledge production being concentrated in a small number of big players in the interests of their own privatised profits [AW20].

In this research programme, we have striven to propose a new paradigm, which we call Wide Learning (WL), capable of complementing current DL techniques. The WL paradigm promotes horizontal scaling, with models that grow wider rather than deeper. Not only does the WL paradigm have the potential to tackle the above-mentioned issues concerning the sustainability and accessibility of DL methods, it can also address some technical challenges for DL with models which scale more effectively. [ZK16] show that Wide Residual Networks (WRNs) scale much better than deep residual networks: increasing width by a small factor results in similar performance gains as adding thousands more layers. Wide neural networks can overcome several difficulties that emerge when training deep neural networks, namely exploding/vanishing gradients and degradation [MTC18], i.e. the increase in both training and test error when increasing the depth of architectures. [ZK16] show that WRNs are capable of achieving state-of-the-art performance in image classification tasks while having considerably fewer layers than standard residual networks (ResNet) [HZRS16a] and being several times faster to train. Throughout our work, we focus on ensemble learning as one way to scale horizontally, i.e. in width, rather than vertically, i.e. in depth.

2.3 Ensemble Learning and Distributed Models

Distributed learning techniques have been proposed as a way of enacting *horizontal* scaling [QWD$^+$16, PG13, XHXD15]. [XHXD15] point out that some models are so complex, with millions to billions of parameters, that they need to be distributed across clusters with thousands of machines. Splitting a model into components and distributing these components is one way to carry out this task (e.g. [DCM$^+$12, HU16, CGDS20]). However, harnessing the power of very large data centres to distribute very complex models is *not* the point of our research programme. We are instead interested in developing methods which can serve as the basis for distributing models which are not very deep or complex across a set of reasonably priced machines. [ZK16] propose a change of paradigm whereby models are scaled by increasing their *width rather than their depth.* As we discussed in Section 2.2, they propose a new architecture based on residual networks [HZRS16b] called *wide residual network* (Wide ResNet). They show that these wide architectures result in better accuracy and efficiency than that of deep neural networks. Although they do not perform distribution of these neural networks, the fact that they scale wider rather than deeper suggests that such a strategy could be applied successfully here. Throughout our work, we have instead focused on ensemble models, since they are especially suited to distribution. We evolve ensembles which we then distribute across both multiple GPUs and multiple machines.

Ensemble methods consist of building a set of models and then aggregating their outputs to form a collective prediction, rather than relying on the predictions made by a single model. [Die00] reviews typical ensemble methods which result in classifiers whose accuracy is greater than that of their individual components and identifies three fundamental reasons why ensembles can perform better than singular models. The first reason is *statistical*: different models can induce different hypotheses which all adequately explain the training data, but which will generalise differently on test data. The second reason is *computational* and concerns the vulnerability of individual models performing local search to local optima. The third reason is *representational*: the underlying function explaining the data might not be representable by any of the individual models, but could be better approximated by an ensemble aggregating all those models.

The diversity of an ensemble is crucial to its performance and is the subject of extensive research [BWT05]. However necessary a condition, [Die00] points out that diversity itself is not sufficient to ensure that an ensemble outperforms its base learners and that these must be simultaneously diverse and accurate. The notion of diversity of interest here may be understood as the level of *disagreement* amongst the learners in an ensemble with regard to each data point, expressed by how different their *errors* are. Whereas for regression ensembles it is possible to derive an exact mathematical formulation for this diversity by calculating the *covariance matrix* of predictions [BWT05], which can be incorporated in the loss function, its definition for classification ensembles is less obvious, thus posing challenges. The literature proposes different metrics for measuring diversity (e.g. [Van05, KW03]), which attests to the vagueness of this concept when tackling ensembles of classifiers. Some of these metrics are pairwise, i.e. measured locally between each pair of learners, or non-pairwise, i.e. a single measure which expresses the global diversity of the ensemble. [BC19] propose a diversity metric for classification ensembles based on *error decomposition*, which is typically applied to regression ensembles. They conduct a theoretical investigation into the relationship between this metric and the generalisation of ensembles which informs the design of an ensemble pruning method. However, their analysis only applies to binary classification problems and the diversity metric, which results from their theoretical derivation of error decomposition, is just a weighted sum of the differences between the prediction made by the ensemble and the predictions made by individual classifiers, i.e. it is a non-pairwise metric. While a similar pairwise metric could easily be formulated, the issue here is that diversity between two models is taken to mean that one of them classifies the example correctly and the other does not. A useful metric in a real-world multinomial classification problem should adequately distinguish between *different errors* in order to be practical.

Generation of diversity in ensembles is mostly done *implicitly*: setting different initial weights in each model, using different training data, different architectures, or different learning algorithms [GCJ15]. While it is hoped that these approaches will generate enough diversity, this process is not directly controlled. Several multiobjective methods have been suggested which explicitly maintain a certain level of diversity in the Pareto fronts. For example, [PDCV10] present an immune-inspired multiobjective optimisation algorithm which optimises two objectives, the diversity of the ensemble and the output error, comparing it with another single-objective algorithm which only minimises errors and relies on implicit diversity. They use the *disagreement* metric, which we also apply in our work. They find that implicit diversity leads to a higher performance gain than their multiobjective approach, which suggests that this explicit diversity metric may not be fit for purpose. [SL15] explore the relationship between diversity and performance in the context of noise detection and observe that more diverse ensembles achieve higher precision but lower recall. [BM17] perform a similar analysis using random forests in the context of remote sensing classification. [ZZ13] present a semi-supervised method operating over unlabelled data to increase the diversity amongst the base learners in an ensemble. [GLL$^+$18] propose a diversity-based metric for performing *ensemble pruning*, i.e. reducing the size of an ensemble by selecting a subset for prediction. [DYL17a] also tackle ensemble pruning with a metric based on both diversity and accuracy. In another demonstration of how there is no standard definition of diversity in classifier ensembles, [BHBK05] introduce yet another diversity metric and use it in a method for decreasing the size of an ensemble, called "thinning". Our main research objective was to go beyond related work in the literature by both investigating multiple definitions of diversity, so as to obtain insights into which diversity metrics result in better-performing ensembles, and designing algorithms which *explicitly* create diversity in classifier ensembles.

2.4 Evolutionary Methods in Ensemble Learning

The use of evolutionary methods is commonplace in tackling the problem of ensemble learning. [ZB04] construct an ensemble of Genetic Programming (GP) classifiers trained on different small subsets of the training data. They ascribe the higher classification accuracy and less overfitting observed with their approach, in comparison with alternative methods, to the increased diversity provided by the GP search. [BSI19] propose an evolutionary algorithm to construct a committee of neural networks and hybridise it with *transfer learning* in an attempt to reduce computational time. They define diversity in terms of architectural diversity and incorporate it in the fitness function. [BJZY12] employ a multiobjective GP approach to evolve classifier ensembles that are both accurate and diverse in order to tackle the problem of unbalanced data, which calls for increased diversity amongst learners. They encourage diversity w.r.t. to the class output of learners by adding a correlation term in the fitness function, as well as a population-level penalty term. They refine their approach in [BJZY13]. [NP15] present yet another multiobjective approach for simultaneous feature selection and design of an ensemble of classifiers, but diversity is not defined as an explicit objective.

[GPHMOB05] evolve ensembles of neural network models using a cooperative coevolution algorithm for ensuring that the models in each ensemble perform well together. In a similar capacity to some of the other papers just mentioned, they use a multiobjective optimisation approach to incorporate four objectives of diversity: correlation, a metric of functional diversity, mutual information, and the Q statistic ([KW03]). Of particular interest to our work is *neuroevolution* [FDM08] and neural architecture search (NAS) [SZZ$^+$20, RAHL19, SM02a, LCS$^+$19], which refer to techniques evolving and searching for diverse architectures and sets of hyperparameters to construct neural network models. Optimising hyperparameters is an open problem which is often tackled in an *ad hoc* fashion; evolutionary methods can provide a solution to this problem by harnessing parallelisation to enhance the exploration of vast search spaces [SCLM19]. This is precisely what we have done in our work with a particular class of evolutionary algorithms called *quality diversity* algorithms, which return a set of diverse solutions rather than optimising a single objective fitness. This is discussed in the next section.

2.5 Quality Diversity Algorithms

As discussed before, diversity, in particular *behavioural* diversity, defined in terms of the *error diversity*, is key to the performance of ensemble models [Die00] and hence is a crucial factor in its design. The relatively recent paradigm of *quality diversity* (QD) [PSS16] algorithms within the evolutionary algorithms (EA) field — which aim to find a maximally diverse but high-performing collection of individuals for a given optimisation task — thus appears ideally suited to this goal. Specifically of interest to our work is the ability of QD methods to produce a diverse repertoire of optimised solutions in a single run. The methods have received much attention in the Evolutionary Robotics (ER) literature to evolve behaviourally or structurally diverse robots [CCTM15, MC15] but much less attention elsewhere, with the exception of a handful of papers in the combinatorial optimisation domain (e.g. [UH18]). A single example within ML [SMPS15a] uses a QD algorithm (novelty search) as an approach to unsupervised feature learning in a method that continually accumulates features that make novel discriminations amongst a training set (with no regard to the classification task), showing that after generating approximately 3000 features, a simple two-layer network performed well compared to other shallow architectures.

Multiple variants of QD algorithms exist which enact this paradigm of explicitly searching for diverse ensembles of classifiers. For example, MAP-Elites *illuminates* search spaces, i.e. it enables researchers to gain insight into how interesting characteristics of solutions combine to affect solution quality. For the first time, we propose that the MAP-Elites algorithm [MC15] be employed to evolve a set of optimised architectures that are diverse w.r.t. their structure, as explained in Chapter 3. The characteristics of interest we consider are therefore the hyperparameters describing the architecture of a network.

Another approach to evolving diverse solutions is to apply novelty search (NS) [LS11a]; this is an approach to evolutionary computation which, instead of rewarding objective fitness, rewards the novelty of a solution compared to those in the current population and an archive of previously discovered solutions. Novelty is domain-specific and can be determined w.r.t. the behaviour of a solution or its genotype. Variants of the method include an element of local competition (NSLC [LS11b]), which forces solutions which are close in the novelty space to compete with their neighbours based on objective fitness. This approach has been found to deal well with the problems posed by function plateaus and local optima, outperforming objective-based methods in some applications. [SMPS15b] use NS to accumulate divergent discriminative features in an unsupervised fashion. Starting in Chapter 4, we use NS to evolve an ensemble of classifiers that are diverse w.r.t. their behaviour, specifically the prediction errors on a validation dataset, searching in the space of hyperparameters. The methods of Chapters 4 and 5 do not explicitly select for accuracy, unlike the approaches mentioned before. Classification accuracy is obtained by optimising the parameters of each network with a stochastic gradient descent (SGD) procedure, but does not influence the novelty score. In contrast to previous multiobjective approaches which trade diversity against accuracy, we explicitly focus on the creation of a diverse ensemble. The NS uses behavioural diversity metrics to guide a search over the space of neural network architectures, constructing an ensemble made up of the most diverse models.

2.6 Accuracy-Diversity Trade-off in Ensemble Learning

[Die00] explains how the performance of an ensemble depends crucially on both the individual accuracy of base learners and the diversity between them. [KV94] formalise this by defining the generalisation error of an ensemble as $E = \bar{E} - \bar{A}$, where $\bar{E}$ is a weighted average of the generalisation errors of individual models and $\bar{A}$ is the weighted average of their ambiguities, which expresses their *disagreement*. Therefore, the more accurate and diverse the learners, the more accurate the predictions made by the ensemble. But the question is often posed regarding the trade-off between diversity and individual accuracy in ensemble learning, which has been the subject of extensive research.

[CCY06] present a review of the use of multi-objective evolutionary algorithms to find this trade-off. [ZZZ07] propose an artificial resampling method which groups the training set into crossed training sets. They claim that this method provides the best trade-off between diversity and accuracy and show that it outperforms Boosting [Sch90] and Bagging [Bre96] on several classification problems. [GJ14] propose a multi-objective evolutionary algorithm which maximises accuracy and diversity together. The Pareto-optimal solutions are analysed as trade-offs between diversity and accuracy. [OAWS15] propose an ensemble pruning method which utilises accuracy and diversity information simultaneously and show that it outperforms alternative approaches. [BJZY12] employ a multiobjective Genetic Programming (GP) approach to evolve classifier ensembles that are both accurate and diverse in order to tackle the problem of unbalanced data; they refine their approach in [BJZY13]. [SSC$^+$17] propose a niching evolutionary algorithm with adaptive negative correlation learning in which the adaptation strategy controls the diversity-accuracy trade-off. [HS18] study ensembles of optimisation algorithms, which have otherwise received little attention, and investigate the accuracy-diversity trade-off in that context. They apply their approach to the domain of bin-packing as an example. [Tsa14] analyses this trade-off with a multi-objective evolutionary system that combines partially trained learners, utilising a ranking formula which incorporates both diversity and accuracy. Many other approaches explore the balance between diversity and accuracy in ensembles, e.g. [MK21, BWT05, MB17, WY09, TQC09].

2.7 Surrogate Modelling

Combining an evolutionary algorithm (EA) with a surrogate modelling function has been common in the literature for many years, e.g. in single-objective optimisation [THMY21], multi-objective optimisation [RLDL20], and particularly in expensive optimisation [ZON$^+$06]. A first surrogate model for neural network optimisation was introduced by [GAM18] and used in conjunction with the NEAT [SM02b] algorithm for evolving the weights and topology of a neural network. This paper used a surrogate distance-based model, employing a genotypic compatibility distance metric that is part of NEAT. The approach has been quickly adopted in the literature using a range of surrogates and a variety of methods to evolve networks. There are several examples of approaches that use surrogates to estimate the performance of an architecture. For example, in 2017 [DYL17b] proposed the Peephole algorithm, which predicted the performance of a convolutional neural network based on its architecture information: a long-short term memory (LSTM) neural network was used to train the model. [SZBB19] extended a Cartesian Genetic Programming method called CPGANN to evolve neural networks using surrogate-based optimisation to reduce the number of fitness evaluations required. They used a Kriging model [CD18] as the surrogate. In [SWX$^+$19], a Random Forest algorithm (RF) was used as a surrogate to predict the performance of a convolutional neural network (CNN) architecture — the authors proposed a method for describing a CNN as a set of features which were used as input to the RF. In [SZZ$^+$20], the authors use a surrogate benchmark for neural architecture search (NAS).

In contrast, Hagg *et al.* [HZSG19] introduce a more flexible method for building a surrogate model that is independent of network topology: rather than describing the neural network architecture, they introduce a *phenotypic* metric which measures the difference in output between two neural networks given the same input sequence. The difference is used in a Kriging surrogate model. The approach that we propose in Chapter 5 is conceptually closest to that of Hagg. For a given neural network, we calculate a behavioural vector that describes its behaviour on a dataset. We then propose a Random Forest (RF) surrogate model that is used to estimate the distance between the behavioural vectors produced by any two neural networks, as this value is required to drive the NS algorithm.

2.8 Summary

This chapter provides the foundations upon which we have built our research programme. In Chapter 3, we propose an algorithm for building an ensemble using MAP-Elites [MC15] to maximise both the diversity and the accuracy of its members. In Chapter 4, we propose a new paradigm with a NS [LS08] method which *only* utilises *explicit* diversity objectives to construct an ensemble of neural network classifiers, with accuracy being implicitly ensured by training the networks with stochastic gradient descent (SGD). In Chapter 5, we then improve upon this NS method by introducing a surrogate model which estimates the distances between neural networks with the goal of avoiding a costly training step at each iteration of the procedure. In Chapter 6, we build upon this method by incorporating accuracy objectives into the explicit search for diversity, so as to investigate whether this could lead to a performance gain.

Chapter 3

Architectural Diversity

3.1 Introduction

In this chapter, we propose a definition of architectural diversity and present an algorithm which constructs an architecturally diverse ensemble of high-performing classifiers. This signifies a shift away from the most common Deep Learning (DL) techniques towards a new paradigm that we call Wide Learning (WL), by promoting the extraction of diverse features from the data in a parallel fashion. We aim to address the issues around the computational demand of DL algorithms by proposing a method which can be distributed across multiple reasonably-priced machines. The algorithm we propose is called WILDA (Wide Learning of Diverse Architectures) and it is described in Section 3.2. This is the key contribution of this chapter. It integrates ML and Evolutionary Computation (EC) to train a classification model applicable to the type of datasets used in DL. This approach first trains an ensemble of low-complexity neural networks, each of which extracts a feature vector from each data point. In a second phase, the features extracted from the ensemble are aggregated in a single shallow model to solve the classification task.

It is well understood that a necessary condition for good ensemble performance is that its members are both *accurate* and *diverse* [Die00]. Here we turn to the *quality diversity* (QD) paradigm [PSS16]. These algorithms return an archive of diverse, high-quality solutions to a problem in a single run, where diversity is defined by the user with respect to features of interest. We exploit the QD approach to generate an ensemble of shallow neural networks which are *architecturally diverse*, yet each *optimised* with respect to the classification task. The features extracted from each network in the first phase are then used in the second phase to train a single shallow network to output the final classification. The resulting architecture can be trained in a distributed way, unlike a typical DL model, which often uses a complex model on a single machine. Section 3.3 describes the experimental work carried out. The results, which are presented and discussed in Section 3.4, show that the method leads to ensembles that perform better than the single best individual model and that architectural diversity is key to improving performance. The work presented in this chapter resulted in two publications: [CHP20, CHKP21a].

3.2 The WILDA Algorithm

WILDA (Wide Learning of Diverse Architectures) uses a two-step approach to classification in which a diverse ensemble of shallow artificial neural networks (ANNs) is trained in the first step (Section 3.2.2) and the features extracted by the ensemble are used in the second step (Section 3.2.4) to train a small feedforward ANN to provide the final result. This is based on the conjecture that a set of diverse features can be extracted from an architecturally diverse repertoire of ANNs, which can then be used to efficiently train a single shallow network in the aggregation phase. The process can be summarised as follows:

1. Apply MAP-Elites to discover a *set* of neural network classifiers which are architecturally diverse, each optimised for accuracy using gradient descent (Section 3.2.1, Algorithm 3.2)
2. Repeat step 1 r times, and then merge the r sets into a single archive (Section 3.2.3)
3. Extract a single feature from each network in the merged archive for each data point (Section 3.2.2)

4. For each data point, concatenate the features extracted in step 3 to form a single input vector (Section 3.2.4)

5. Train a single shallow network to output the desired classification using the input vectors from step 4 as input (Algorithm 3.5)

WILDA uses a hybrid method that combines an evolutionary approach (MAP-Elites [MC15]) with a traditional ML approach (gradient descent) for training each ANN (Algorithm 3.1). MAP-Elites (Multi-dimensional Archive of Phenotypic Elites) explores a low-dimensional projection of the space of *hyperparameters* which describe the architecture of the networks and returns an archive of structurally diverse networks. A gradient descent procedure optimises the *parameters* of each ANN discovered. The MAP-Elites algorithm can be run simultaneously on multiple nodes, resulting in `n_nodes` archives at the end of the feature-extraction phase. These archives are merged before running the aggregation phase, which provides the final classification. We first give an overview of the MAP-Elites algorithm before describing each phase in detail.

Algorithm 3.1 WILDA algorithm (high-level view)

```
procedure WILDA
    for $n = 1 \rightarrow$ n_nodes do
        $MAP_n \leftarrow$ MapElites()                        ▷ generate $n$ archives of diverse networks
    end for
    $mergedMap \leftarrow$ merge($MAPs$)                       ▷ merge $n$ archives into a single archive
    $features \leftarrow$ extractFeatures($data$, $mergedMap$)   ▷ for each data point, extract
                                                              a feature vector from each
                                                              network in the archive
    $aggregatedModel \leftarrow$ trainShallowNetwork($features$)   ▷ train single network to
                                                                 classify data
end procedure
```

3.2.1 MAP-Elites

Fundamentally different to a traditional search algorithm, the MAP-Elites algorithm provides a holistic view of how high-performing solutions are distributed throughout a feature space [MC15]. The method creates an archive of high-performing solutions at each point in a space defined by dimensions of variation chosen by a user, according to characteristics of a solution that are of interest. The resulting archive enables the user to gain specific insight into how combinations of characteristics of solutions correlate with performance. As the approach encourages diversity, it has often been shown to be more capable of fully exploring a search space, outperforming state-of-the-art search algorithms which are given a single objective, and can be particularly helpful in overcoming deception [PSS16].

The standard algorithm is given in Algorithm 3.2. This is adapted for our purposes as follows. A solution consists of a PyTorch [PGC+19] representation of an ANN. We select three dimensions to characterise an architecture, namely the number of convolutional layers, the number of dense layers, and the maximum size (number of outputs) of any dense layer in the network, which together comprise the `featureDescriptor`. The algorithm begins by generating random solutions which are mapped to a grid that is discretised in each dimension into a fixed number of cells (representing possible values of each feature). The grid thus contains $|C| \times |D| \times |S|$ cells, where these values represent the total number of values permitted for the convolutional, dense and size dimensions, respectively. Following an initialisation phase, solutions are randomly selected from the grid, after which a *variation* operator is applied to generate new solutions. Child solutions are evaluated according to a performance metric and then mapped back to the grid according to their descriptor: a child solution replaces an existing solution in any cell if it is better according to its performance metric or may simply occupy an empty cell. The search process aims to fill the entire grid with solutions, each of which represents the best performing solution for a given feature descriptor. The precise implementation of each of the above steps is described in the next section.

3.2.2 Feature Extraction Phase

During the extraction phase, MAP-Elites attempts to find a set of diverse ANN architectures, each of which is optimised on a subset of the data towards solving a classification task of interest. At the end of this phase, a feature vector is extracted from each network for each data point, corresponding to the output of the second-to-last layer of each network as explained

Algorithm 3.2 MAP-Elites Algorithm, taken directly from [MC15]

```
procedure MAP-ELITES ALGORITHM
    ($\mathcal{P} \leftarrow \emptyset, \mathcal{X} \leftarrow \emptyset$)
    for $iter = 1 \rightarrow max\_iterations$ do
        if $iter < initialise\_iterations$ then
            $x' \leftarrow$ randomSolution()
        else
            $x \leftarrow$ randomSelection($\mathcal{X}$)
            $x' \leftarrow$ randomVariation(x)
        end if
        $b' \leftarrow$ featureDescriptor(x')
        $p' \leftarrow$ performance(x')
        if $\mathcal{P}(b') = \emptyset$ or $\mathcal{P}(b') < p'$ then
            $\mathcal{P}(b') \leftarrow p'$
            $\mathcal{X}(b') \leftarrow x'$
        end if
    end for
    return feature-performance map ($\mathcal{P}$ and $\mathcal{X}$)
end procedure
```

below.

Network Representation

An individual uses a list representation to describe a variable-length sequence of convolutional layers followed by a variable-length sequence of dense layers. Each layer has a random number of neurons, selected from a list of discrete values. Each dense layer uses a hyperbolic tangent [Mur12] activation function. The last hidden layer is designated as the *feature layer*: the output of this layer is a binary vector which represents a feature extracted by the network to be used in the second phase (Section 3.2.4). As a result, this layer always has a fixed number of neurons, *feature_size*, set by the user according to the desired size of the feature vector. Finally, an output layer is added which provides the classification of the data point.

Variation Operators

Three new individuals are generated at each iteration by the *crossover* and *mutation* operators. Two children are generated by applying *crossover* to a pair of randomly selected individuals. The third child is generated by applying *mutation* to a single randomly selected parent. Crossover randomly picks two individuals, selects random crossover points among their dense layers, and swaps them accordingly. Mutation randomly picks a mutation point among the dense layers of an individual and either adds or removes a layer at that position. Note that crossing over two sequences of dense layers or removing a layer from such a sequence will, in the general case, require changing the input and/or output sizes of layers at the crossover/mutation point; when adding a layer at a mutation point, its size is given by the output and input sizes of the previous and following layer, respectively. For simplicity, crossover and mutation only operate over the dense layers of an ANN; since convolutional layers tend to have a non-decreasing number of channels, these operations would require modifying all layers beyond the crossover/mutation point, defeating their purpose [LBBH98b]. However, as the convolutional layers generated in the initialisation process will be paired with different combinations of dense layers as a result of these two operations, this still ensures a diverse search process.

Performance Evaluation

To evaluate each individual, the single network encoded by the individual is *trained* for a fixed number of iterations (`eval_iters`) on a sample training set using a standard gradient descent procedure which minimises cross-entropy loss [Mur12]. Its classification accuracy is then calculated on a sample test set, and this value assigned as its fitness, as described in Algorithm 3.3. The sample train and test sets are drawn randomly from the training data at each iteration; they are both 20% the size of the complete training data, which encompasses 60000 examples in the two datasets we tested (MNIST and CIFAR-10). For this reason, each node uses 12000 examples at each evaluation and may therefore only ever have a partial view of the data required to solve the task.

3.2.3 Distribution of Computation

As described in the introduction, one of the goals of WL is to be able to distribute the computation over multiple nodes to enable the model to be run in parallel. One approach to achieving this would be to segment the $|C| \times |D| \times |S|$ grid into sub-partitions and run each sub-partition on a separate node. However, here we adopt an approach described in [HSP18], which proposes a fully distributed implementation of MAP-Elites designed to be run on a robot swarm. In this approach, each node runs its own instance of the MAP-Elites Algorithm 3.4. At the end of the extraction phase, all the maps returned are merged into a single map referred to in the QD literature as a global map of elites. In previous work [HSP18], we evaluated multiple options for performing the merge step which inform our choice of two strategies:

1. merging without overlap: for each cell in the map, select the highest-performing ANN model found in that cell from any of the `n_nodes` individual maps returned
2. merging with overlap: for each cell in the map, return all of the ANNs found in that cell across all `n_nodes` maps. This means that a maximum of `n_nodes` $\times |C| \times |D| \times |S|$ neural networks is returned

In both cases, the maximum number of models passed to the learning phase via the global map is N, where N is the size of the map (i.e. $|C| \times |D| \times |S|$). When merging without overlap, the procedure returns a maximum of N networks, maximising diversity. On the other hand, the merge with overlap procedure can return $>> N$ networks. In this case, the procedure selects the top N networks according to their fitness metric. This strategy can return multiple networks which map to the same cell, therefore favouring the quality of solutions over their diversity.

Algorithm 3.3 Calculating the fitness of an individual neural network and adding it to the map of elites

```
procedure TRAIN_AND_EVAL($m$, $sample\_train$, $sample\_test$, $ME$)
    $c, l, s \leftarrow$ architectural features of $m$
    train($m$, $sample\_train$)
    if $ME[c, l, s] = \emptyset$ OR
            accuracy($m$, $sample\_test$) $> ME[c, l, s]$.fitness then
        $ME[c, l, s] \leftarrow m$
    end if
end procedure
```

Algorithm 3.4 Training loop for each node in the extraction phase

```
create empty map of elites $ME$
draw $sample\_train$ and $sample\_test$ from training set $\mathcal{D}$
for $initial\_size$ do
    $m \leftarrow$ generate random ANN model
    train_and_eval($m, sample\_train, sample\_test, ME$)
end for
for $extraction\_epochs$ do
    draw $sample\_train$ and $sample\_test$ from training set $\mathcal{D}$
    draw individuals $x$, $y$ from map of elites $ME$
    $x', y' \leftarrow$ crossover($x, y$)
    train_and_eval($x', sample\_train, sample\_test, ME$)
    train_and_eval($y', sample\_train, sample\_test, ME$)
    draw individual $z$ from map of elites $ME$
    $z' \leftarrow$ mutate($z$)
    train_and_eval($z', sample\_train, sample\_test, ME$)
end for
```

3.2.4 Learning/Aggregation Phase

The learning phase uses the information learnt by the ANNs contained in the repertoire resulting from the first phase to train a single model to solve the classification task. This single model is a fixed-structure shallow ANN that has a single intermediate layer with `n_hidden_agg` neurons and a hyperbolic tangent activation function. The node where this model is trained is called the *root node*. Note that, even though there is a global merged repertoire at the end of the feature-extraction phase, as described in Section 3.2.3, this repertoire contains only references to the models which were generated and trained in separate nodes and each of these models will still be running in its corresponding node.

Algorithm 3.5 Learning phase of the procedure for the **root node**

```
$all\_MEs \leftarrow$ gather_all_maps()                    ▷ Root node receives all repertoires
$global\_ME \leftarrow$ merge($all\_MEs$).                  ▷ Repertoires merged into a global map
send $global\_ME$ to the other nodes                         ▷ Global map known by all nodes
initialise model $M$
for $learning\_epochs$ do
    for batched $data$ and $labels$ do
        $all\_features \leftarrow$ gather_all_features($data$)   ▷ Each node sends its feature vector extracted from $data$
        concatenate $all\_features$ into intermediate representation $\underline{f}$
        $M$.train_step $\left(\underline{f}, labels\right)$
    end for
end for
```

The phase begins with an extraction step: each data point in the training set is passed through each of the n networks contained in the merged map from the previous phase. This returns n binary vectors, each representing a feature (as described in Section 3.2.2), which are concatenated to form the input layer of the new model. This model is then trained with a standard gradient descent procedure by minimising cross-entropy loss. Algorithm 3.5 shows pseudocode for the learning phase specific to the root node and Algorithm 3.6 shows pseudocode for all nodes.

Algorithm 3.6 Learning phase of the procedure for all nodes (including root node)

```
send($ME$, root)                                ▷ Sends own repertoire to root node
$global\_ME$ ← receive(root)                    ▷ Receives global map from root node
$own\_models$ ← get_own_models($global\_ME$)
for $learning\_epochs$ do
    for batched $data$ do
        $own\_features$ ← get_features($own\_models$, $data$)
        send($own\_features$, root)             ▷ Sends feature vector to root node
    end for
end for
```

3.3 Experiments

Experiments have been conducted to: (1) evaluate the performance of WILDA as a classifier on two datasets providing varying levels of challenge; (2) explore the effects of encouraging diversity vs. quality within an ensemble; (3) explore the influence of the size of the ensemble used to execute the centralised learning step.

Two well-known benchmark datasets are used: MNIST [LBBH98b] and CIFAR-10 [Kri09]. MNIST is a set of 60000 hand-written digits, while the CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. MNIST is known to be relatively straightforward for ANN architectures, while the latter poses a significant challenge to "off-the-shelf" models; state-of-the-art DL models for CIFAR-10 require significant customisation and tuning. Four sets of experiments are conducted as described below.

The relevant parameter values are set as per Table 3.1, which lists all the parameter values

Table 3.1: Parameter settings

Parameter	Description	Value(s)
`n_nodes`	Number of distributed nodes	8
`initialise_iterations`	Number of ANNs in the initial maps	20
`C`	set of possible # of convolutional layers	{1,2}
`D`	set of of possible # of dense layers	{2,3,4}
`S`	set of possible values for layer size	{100,110,120, 130,140,150,170}
`max_iterations`	Number of iterations in the extraction phase	30
`eval_iters`	Number of iterations in gradient-descent training in extraction phase	5
`feature_size`	Size of binary vector produced by last hidden dense layer in extraction phase	100
`aggregation_iters`	Number of iterations in the learning phase	10
`n_hidden_agg`	Number of neurons in the intermediate layer of the centralised model (learning phase)	50
`merge`	Strategy used to merge the maps of elites evolved by each node	*with/without overlap*
`n_models`	Maximum number of models to use in the learning phase ($\leq$ the size of the map)	48

that are used throughout the experiments. All experiments, including the runs of the feature-extraction phase to get the baseline results, are repeated 30 times in order to evaluate statistical significance. Two-tailed Mann-Whitney significance tests are applied to compare experimental results, and noted as significant if the resulting p-value is < 0.01.

3.3.1 Baseline: Single Shallow ANN

As a baseline for comparing the quality of the ensemble-based solutions from WILDA, we use a single shallow ANN, trained in a similar fashion to the aggregated single-layer model used in the learning phase of the algorithm (Section 3.2.4). This ANN is the one that has achieved the best performance after a run of the feature-extraction phase (Section 3.2.2). Note that this is not a true "baseline" in the sense that we do not choose a random or otherwise uninformed architecture, but rather one that has been found to be the best. This is because we wish to understand the benefits of the ensembles built by our diversity-driven approach; outperforming individual neural networks with the highest fitness values is therefore a more interesting challenge. Out of interest, the shallow network trained with diverse features mentioned in [SMPS15a] achieves an accuracy of 98.75% on MNIST, while a shallow convolutional neural network (CNN) is reported to obtain 75.86% on CIFAR-10 in [MV15a].

3.3.2 Comparison of Different Merge Strategies

This set of experiments compares the two merge strategies (Section 3.2.2) to understand how the construction of the merged archive impacts the performance of the aggregated model. For reference, a full run of WILDA with the parameters of Table 3.1 takes around 50 minutes on the machine upon which the algorithm was tested. Recall that the two strategies represent different trade-offs between diversity and quality of the solutions. The size of the maps evolved by each node, as well as the global map, is $2 \times 3 \times 8 = 48$; this is also the maximum number of ANN models used in the learning phase (`n_models`).

3.3.3 Investigating the role of architectural diversity

The global map of elites which is constructed from the individual maps and used in the aggregation phase is essentially an ensemble of the best ANNs found for different types of architecture. This naturally raises the question of how useful it is to promote *architectural diversity* amongst the networks in the ensemble, and how the performance of such a diverse ensemble compares with ensembles which do not have *architectural* diversity, but are diverse in terms of their optimised *weights* due to training on different samples of the dataset. Thus, we compare the performance of architecturally diverse ensembles evolved by WILDA with two kinds of ensembles that lack architectural diversity:

- an ensemble of networks in which every individual has the *best* architecture found in the extraction phase but is trained using a different sample of the training data

Table 3.2: Median test set accuracy for the two merge strategies considered

	MNIST	CIFAR-10
Baseline	0.9899	0.646
Merge without overlap	0.99175	0.6982
Merge with overlap	0.9919	0.6983
Divergent Discriminative Feature Accumulation [SMPS15a]	0.9875	*n/a*

- an ensemble of networks in which each individual has the *worst* architecture found in the extraction phase but is trained using a different sample of the training data

3.3.4 Investigating the influence of ensemble size

After constructing the merged map, one question that arises is how to use it to solve the task. We can simply pick the single best-performing architecture, as per the baseline case, or use an ensemble selected from the map. This raises the question of how many networks to include in the ensemble. This set of experiments assesses the relevance of fine-tuning the number of ANN models used in the ensemble by comparing the test set performance of ensembles of different sizes. We vary `n_models` (the size of the ensemble) in the range {10, 20, 30, 40, 48}, selecting the best `n_models` ANNs in each experiment. All the other parameters are fixed as per Table 3.1.

3.4 Results and Discussion

This section presents the results of the experiments of Section 3.3 and discusses their significance. Table 3.2 presents the median test set accuracy for the two merge strategies and compares them to the baseline. Recall that the architecture of the baseline network is that of the best network found in a run of the feature-extraction phase, as explained earlier in this section.

3.4.1 Different Merge Strategies

By looking at the results of Table 3.2, we can see that both merge strategies significantly outperform the baseline individual best network for both datasets ($p << 0.01$). There is no significant difference between the two merge strategies, however. A possible reason for this is that these two merge strategies actually lead to similar global maps of elites. On the one hand, each node might be finding high-performing ANNs in different regions of their individual maps, thus leading to few overlaps at a same cell when merging. On the other, it is possible that networks which are mapped to the same cell are still significantly diverse. Further investigation is required to answer these questions. We also include for interest the result obtained by Szerlip et al. [SMPS15a] from first evolving 3000 divergent discriminative features, but note that the training procedure used in that paper differs from ours, which runs a two-phase procedure that first trains on a small training set before shifting to the full example set, using a single-layer network to classify. Our evolved ensemble of features obtained from 48 diverse networks outperforms both the single high-performing learner and the previously obtained result.

3.4.2 The Role of Architectural Diversity

Table 3.3 shows the accuracy results for ensembles trained with a fixed set of architectures (the best and worst architectures found in the extraction phase). All differences are significant compared to both the baseline and to both merge strategies ($p << 0.01$). It is clear that ensembles that do not have architectural diversity perform significantly worse than the results obtained by WILDA on both datasets. They also show that the fixed-architecture ensembles perform significantly worse than the baseline case. This is perhaps surprising given that the baseline case uses a feature vector from a single network obtained from the extraction phase. It appears that combining the predictions made by the best architecture trained on different subsets of the data leads to overall poorer performance than training an individual network on all of the data. This could be a particular characteristic of our procedure for aggregating the features extracted from the data by the ANNs in the ensemble during the learning phase, as described in Section 3.2.4. However, it is a clue that the ensembles may be accumulating and

Table 3.3: Median test set accuracy for ensembles without architectural diversity

	MNIST	**CIFAR-10**
Ensemble of instances of the best architecture	0.98685	0.62835
Ensemble of instances of the worst architecture	0.97815	0.557

reinforcing prediction errors when there is a lack of architectural diversity, i.e. errors made on the same data or on data from which similar features have been extracted. The performance of an ensemble depends on the *diversity of errors* made by each of its learners [Die00]; the results of this section enable us to suggest that promoting architectural diversity among the ANNs in the repertoires built during the extraction phase of the algorithm drives diversity of features extracted from the data and diversity of prediction errors, which in turn leads to higher test set accuracy. This observation is of the utmost importance in informing future research into how to increase the performance of the diverse ensembles evolved by WILDA.

3.4.3 Influence of Ensemble Size

Figure 3.1 presents the performance results when only the `n_models` top-performing models from the global map of elites are used in the learning phase. In all cases the algorithm significantly outperforms the baseline and changing this parameter only produces small variations in accuracy. For MNIST, using only 10 models outperforms all other cases. The difference is statistically significant when compared with cases using 30 or more models. These observations suggest that using fewer models in the learning phase leads to better performance on the MNIST dataset. This could be because the simplicity of MNIST leads to smaller error diversity among different learners, which would cause the reinforcement of errors in larger ensembles. On the other hand, using only 10 models leads to significantly worse performance on the CIFAR-10 dataset than all other cases. This disparity in the observations for both datasets suggests that the choice of number of models that brings optimal performance is domain-dependent and must therefore be fine-tuned to the problem being tackled.

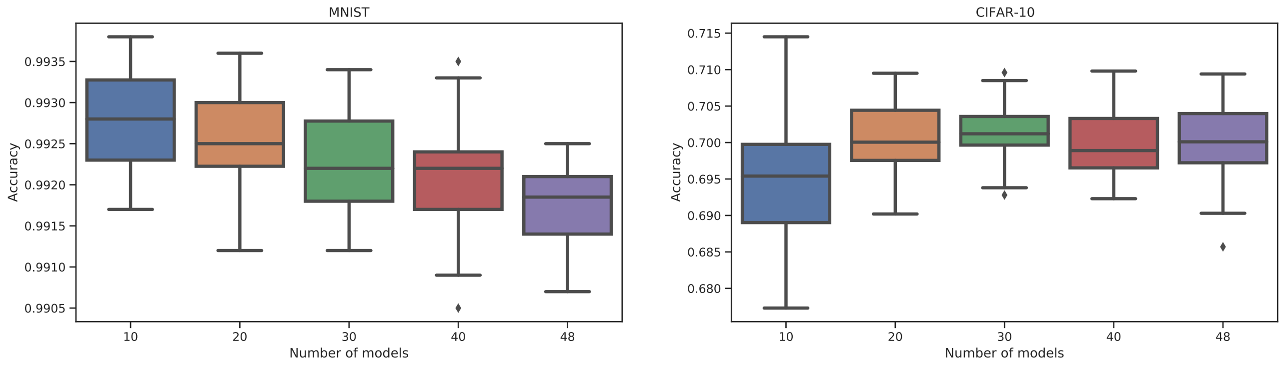


Figure 3.1: Test set accuracy for each number of models added to the ensemble in the learning phase

3.4.4 Search Space Illumination

Figure 3.2 shows an example of how the extraction phase of WILDA, which runs a version of MAP-Elites, can *illuminate* the search space of architectures. For each combination of number of dense layers and maximum size of any dense layer, the diagram shows the fitness (accuracy on sample test set) — averaged out along the other dimension, which is the number of convolutional layers — of the best-performing ANNs on the CIFAR-10 dataset which map to that cell after merging all individual maps into a global map *without overlap* (Section 3.2.2). Note that more runs of the extraction phase would be required in order to draw conclusions about which architecture leads to the best performance.

3.5 Summary

This chapter presented our first attempt at tackling the problem of constructing a diverse ensemble of neural network architectures in a systematic way. To this end, we proposed WILDA, an innovative diversity-driven distributed algorithm. The basic idea is to extract a representation for the input in a way that scales horizontally rather than vertically. The algorithm first trains a repertoire of *architecturally diverse* ANNs in parallel: each node constructs a repertoire of high-performing, architecturally diverse networks, accessing different subsets of the data, which are then merged together into a global map of elite networks, each of low complexity. The features extracted by each network from the data are then aggregated and fed

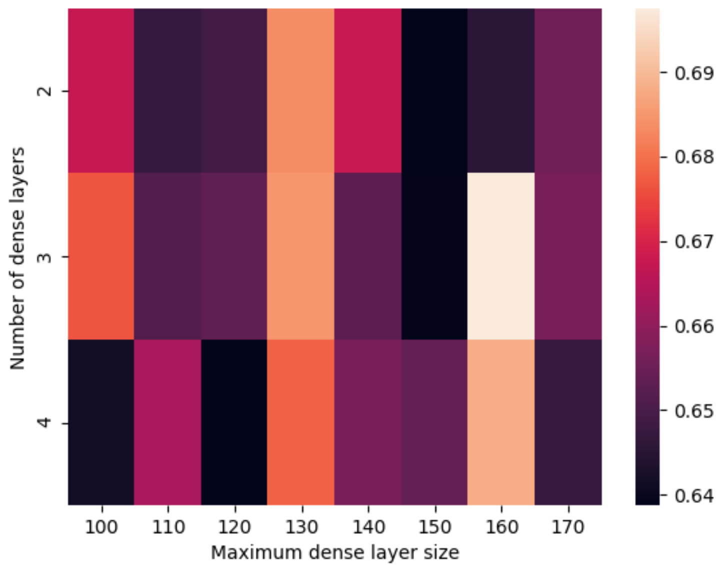


Figure 3.2: Illuminating the architectures' search space for the CIFAR-10 dataset - the colour shading indicates the accuracy of the best network found for each cell

to a centralised model which will solve the classification task. The approach relies on the assumption that networks constructed with diverse architectures and trained on diverse samples of data will extract diverse features from a dataset, ultimately improving classification.

With WILDA, we showed that a general method that is easily distributed and does not require either vast amounts of computational power or expert knowledge to design a network is capable of reasonable performance. Experimental results showed that this technique performs well on the MNIST and CIFAR-10 datasets and that a diverse ensemble performs better than the best individual model found in the extraction phase. We also showed that architectural diversity is key to improving performance: ensembles using fixed architectures were found to perform worse than an individual model, likely due to the accumulation and reinforcement of the same kind of errors. The results suggest that architectural diversity promotes error diversity, which in turn increases the performance of the ensembles evolved by the algorithm. The algorithm can also be a useful tool for exploring and illuminating the space of hyperparameters, exposing correlations between the characteristics of different architectures and their performance.

The approach presented in this chapter was the first step towards a general Wide Learning (WL) paradigm that can complement current DL techniques. But WILDA only *rewards* diversity, rather than actively *searching* for it. It also only uses a definition of architectural diversity. In the next chapter, we generalise and extend this notion of diversity in ensemble learning with a new method which searches for *behavioural* diversity.

Chapter 4

Explicit Behavioural Diversity

4.1 Introduction

After showing how architectural diversity can be promoted systematically to improve the performance of ensembles, we now turn our attention to *behavioural* diversity. [Die00] explains how, in order to ensure that ensembles perform better than their base learners, these must be *diverse* in terms of their *behaviour*. Behaviour is here broadly defined in terms of the *errors* that individual learners make in their predictions. However, there is no consensus on particular definitions of behavioural diversity or on how to exploit diversity to construct a high-performing classifier.

In this chapter, we first propose metrics which define diversity in terms of the *divergence of errors* made by different models and/or their level of disagreement, which characterises their *behaviour*. Novelty search (NS) [LS11a] is used to search over a space of neural network architectures, maximising a novelty score defined by the behavioural metric w.r.t. the other individuals in the population. Each individual network discovered is optimised using standard gradient descent on a training dataset before its novelty score is calculated. Networks with high-scoring novelty are iteratively added to an archive which forms the final ensemble. The method therefore explicitly *searches* for diversity amongst learners. Diversity drives not only this search, but also the construction of the final ensemble. Section 4.2 describes the individual network architectures — which are more complex than the simple architectures of Chapter 3 — and presents different definitions of behavioural diversity, the first of the two key contributions of this chapter. The second contribution is the NS algorithm, described in Section 4.3.

We test our approach on three benchmark datasets from Computer Vision — CIFAR-10, CIFAR-100 [Kri09], and SVHN [NW11]. The experimental work is described in Section 4.4. The results, which are presented and discussed in Section 4.5, show that the *error diversity* metrics we propose, used in conjunction with NS, lead to higher-performing ensembles than other metrics commonly used in the literature and that the ensembles generated by explicitly searching for diversity significantly outperform those that use either only implicit measures to encourage diversity or random search approaches that simply reward it. The main contributions in this chapter are twofold: (1) we describe a systematic NS method to evolve an ensemble of individual classifiers which are *behaviourally diverse* by explicitly searching for this diversity and (2) we provide new insights into how diversity impacts ensemble performance and which diversity metrics are most appropriate to defining behavioural diversity. We conclude that our empirical results signpost an improved approach to promoting diversity in ensemble learning. In a significant development from the approach of Chapter 3, we identify what sort of diversity is most relevant and propose an algorithm that explicitly searches for it *without selecting for accuracy*. The work presented in this chapter resulted in one publication: [CHKP21b].

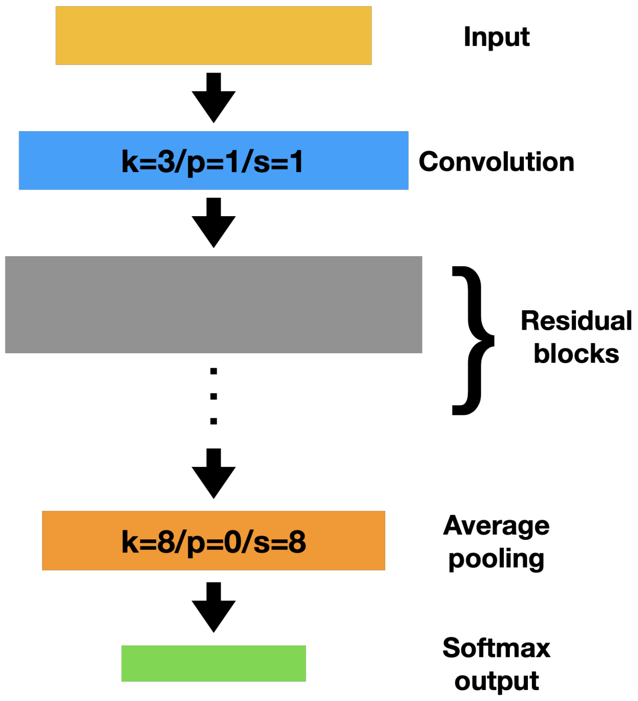
4.2 Architectures and Diversity Metrics

We use NS to evolve individual neural network models that are behaviourally diverse. The NS searches a space of architectures, whereas the parameters of the neural networks are optimised with a standard gradient descent procedure. The most diverse models are added to an ensemble, which is used for prediction on a test set. In this section, we describe the individual neural network architectures evolved by our algorithm and the metrics we propose to measure diversity.

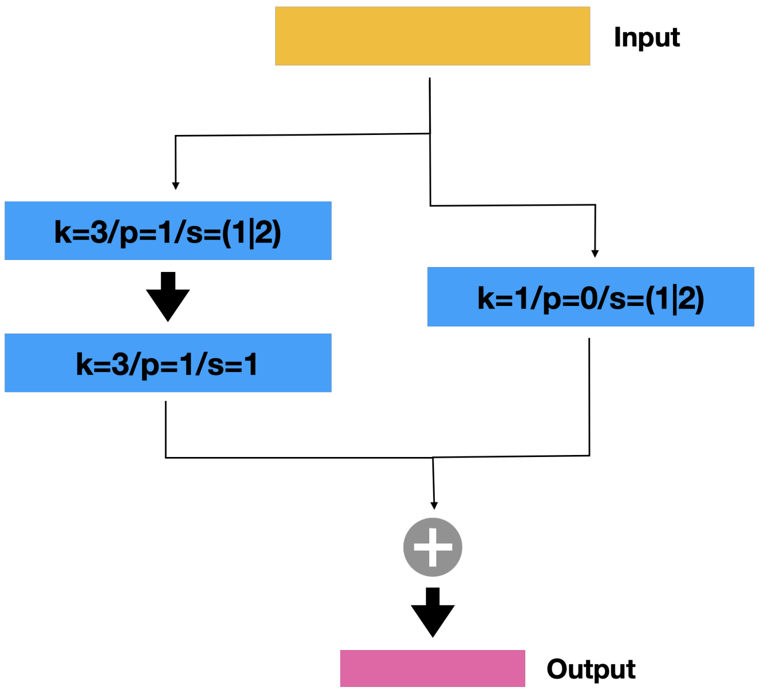
4.2.1 Individual Neural Networks

The individual models evolved by our procedure are *residual neural networks* [HZRS16a] based on the wide architectures proposed by [ZK16]. Figure 4.1a shows a generic neural network such as those evolved by our NS algorithm. They are made up of a *convolutional layer* with kernel size 3, padding 1, and stride 1; a variable-length sequence of *residual blocks*; an *average pooling* layer with kernel size 8, padding 0, and stride 8; and a final linear output layer with a *softmax* activation function. The output size, i.e. number of channels, of the convolutional layer and each residual block is variable. Figure 4.1b illustrates a generic residual block. It is a block of kernel size 3 such as those used by [ZK16], meaning it is made up of two sequential convolutional layers with kernel size 3, padding 1, and the same, though variable, output size. The output of a block is the sum of its input with the output of the second of the two sequential convolutional layers; note that if their size and shape do not coincide, an extra convolutional layer must first be applied to the input. The stride of the second convolution in the block is always 1; the stride of the first convolution is 2 for the *last two* residual blocks in the network and 1 for all other blocks. If r is the number of residual blocks in the network, the effect of this is that the first $r-2$ blocks *do not* reduce the *dimensionality* of the input data, while the last two halve both the width and height of the input feature planes. Batch normalisation is applied before the average pooling layer and each convolution bar the very first one in the network. Each convolutional layer is followed by a ReLU activation function [NH10]. Furthermore, the residual blocks apply a *dropout* layer between each convolutional layer.

The *hyperparameters* of each network are evolved by NS. The relevant degrees of freedom are the output size of the first convolution (Figure 4.1a); the number of residual blocks in the network; the output size of each residual block (i.e. the output size common to all its convolutional layers); and the *probability* of dropout at each residual block. Each individual in the population is then defined by a variable-length vector, depending on the number of blocks r: $[C, O_1, ..., O_r, P_1, ..., P_r]$, where C is the output size of the first convolution, O_i is the output size of block i, and P_i its dropout probability. Each individual is mapped to a Pytorch module [PGC+19] for implementation purposes. The *parameters* of each network are randomly initialised and then optimised by a standard gradient descent procedure.



(a) Generic residual neural network as those evolved by our procedure (k = kernel size; p = padding; s = stride)



(b) Generic residual block. Note that the convolution on the right-hand side is only necessary when the number of channels and/or dimensions of the input are not the same as the output

Figure 4.1: Generic topology of individual neural networks

Diversity Metrics

The key element of NS is the definition of the metric to calculate novelty. We consider three different behavioural metrics, two of which we define ourselves, calculated between pairs of individuals, that are used by the NS procedure to calculate novelty scores. We also consider two selection metrics for adding members to the final ensemble, one of which is also defined by us. Both are calculated in a non-pairwise fashion.

Behavioural metrics for guiding the NS

Let $\boldsymbol{y}_i$ be the vector of predictions for model i with each prediction y_i^n for data point $\boldsymbol{x}^n$ being a class label in $\{1..C\}$. Let $\boldsymbol{p}_i$ be a binary vector where $p_i^n = 1$ if the prediction y_i^n is correct and $p_i^n = 0$ otherwise. Let N^{11}, N^{00}, N^{01}, and N^{10}, respectively, be the total number of test instances where two models are both correct, both incorrect, and when one is correct and the other is not. The first diversity metric we consider is the *proportion of different errors* between two models when at least one of them is incorrect. We propose this metric since it provides insight into the divergence between the errors made by two models. We have defined it as:

$$\text{prop}_{i,j} = \frac{N^{01} + N^{10}}{N^{00} + N^{01} + N^{10}} \tag{4.1}$$

Consider now the *two's complement* of the binary vector of correct predictions $\boldsymbol{p}_i$, $\boldsymbol{w}_i$ (i.e. the binary vector of wrong predictions). The next metric we propose is the *cosine distance* between the binary vectors of wrong predictions made by two models i and j. Like $\text{prop}_{i,j}$, we consider this metric because it is a measure of the distance between the errors made by two models. We have defined it as:

$$\text{cos\_dist}_{i,j} = 1 - \frac{\boldsymbol{w}_i \cdot \boldsymbol{w}_j}{\|\boldsymbol{w}_i\| \|\boldsymbol{w}_j\|} \tag{4.2}$$

Finally, we consider a widely used metric (e.g. [Van05, KW03, PDCV10]) defined as the *disagreement* between two models, i.e. the proportion of test instances where one of them is correct and the other is not. We take this metric into account since it enables the judgement of how commonly two models disagree on any test instance. It is defined as:

$$\text{dis}_{i,j} = \frac{N^{01} + N^{10}}{N^{00} + N^{01} + N^{10} + N^{11}} \tag{4.3}$$

We note here that the two metrics we propose focus more closely on the instances where there was at least one error, i.e. which at least one of the two models has misclassified, and are calculated with respect to those instances. On the other hand, the last metric is simply a proportion of the instances where the two models disagree, calculated w.r.t. the entire test set; it is thus less informative with regard to errors.

Ensemble selection metrics

The first selection metric for adding members to the final ensemble which we propose is simply the mean behavioural diversity metric measured between a candidate ensemble member i and each of the current ensemble members. Let S be the set of ensemble members. This metric is thus defined as:

$$\text{Mean b. d.} = \overline{\text{div\_metric}_i} = \frac{1}{\|S\|} \sum_{j \in S} \text{div\_metric}_{i,j} \tag{4.4}$$

Where $\text{div\_metric}_{i,j}$ is one of $\text{prop}_{i,j}$, $\text{cos\_dist}_{i,j}$, and $\text{dis}_{i,j}$. The higher this mean, the more diversity there is among the candidate model and the current ensemble members.

The other selection diversity metric we consider is the *entropy* of predictions among the ensemble members, as defined in [Van05]. Let S_i be the candidate ensemble created by adding i to S. The entropy is then:

$$E = \frac{1}{N} \sum_{n=1}^{N} \sum_{c=1}^{C} -\frac{N_c^n}{\|S_i\|} \log_C \left(\frac{N_c^n}{\|S_i\|} \right) \tag{4.5}$$

Where N is the number of test instances, C is the number of classes and N_c^n is the number of models which assign class c to instance $\boldsymbol{x}^n$. The higher this entropy value, the higher the diversity amongst the members of the candidate ensemble. We note that this entropy metric does not focus on instances where there were errors in the classification, but rather on the entire test set. On the other hand, the mean behavioural metric might focus on those instances if the behavioural metric is one of $\text{prop}_{i,j}$ or $\text{cos\_dist}_{i,j}$, as discussed before.

The previous definitions assume S to be a non-empty set. When S is empty, i.e. when no models have been added to the ensemble yet, the first model to be added is the one with the best *novelty score*, as explained further down in the description of the algorithm.

4.3 Novelty Search Algorithm

Our algorithm for building an ensemble implements NS as described by [LS11a], applying it to our problem domain. Algorithm 4.1 presents the pseudocode for this procedure. The original training data is split into two sets, one for training and one for validation. The parameters of the models are optimised on the training set with standard gradient descent and the validation set is used to get predictions by each model which will enable the calculation of diversity metrics.

Selection in NS is driven by the novelty score, which computes the sparseness at any point in the behaviour space, defined by the behavioural metric. Areas with denser clusters of visited points are considered less novel and therefore rewarded less. This is defined as the average distance to the K-nearest neighbours of a point, calculated with respect to the other individuals in the current generation and to a stored *archive* of previously sampled solutions. Hence the novelty score is calculated as:

$$NS_i = \frac{1}{k}\sum_{k=0}^{K} \text{div\_metric}(x_i, \mu_k) \tag{4.6}$$

where μ_k is the kth-nearest neighbour of x_i with respect to the *behavioural* diversity metric $\text{div\_metric}_{i,j}$, selected from the metrics defined in Section 4.2.1.

Algorithm 4.1 Ensemble evolution through NS

```
randomly initialise population $pop$
$archive \leftarrow \emptyset$
$ensemble \leftarrow \emptyset$
draw $train\_split$ and $val\_split$ from training set $\mathcal{D}$
set evolution iterations $epochs$
set training iterations $iters$
select behavioural $\text{div\_metric}_{i,j}$ from Section 4.2.1
$ns_i$ is the fitness value (novelty score) of model $m_i$
for $epochs$ do
    for model $m_i \in pop$ do
        train($m_i, train\_split, iters$)    ▷ standard gradient descent optimisation
    end for
    $all \leftarrow pop \cup archive$
    for pair $m_i, m_j \in all \times all$ where $m_i \neq m_j$ do
        $dist_{i,j} \leftarrow \text{div\_metric}(m_i, m_j)$    ▷ calculated on $val\_split$
    end for
    for model $m_i \in pop$ do
        $ns_i \leftarrow \frac{1}{k} \sum_{m_n \in \mathcal{N}_i} dist_{i,n}$    ▷ where $\mathcal{N}_i$ is the set of k-nearest neighbours of $m_i$ in $all$
    end for
    $archive \leftarrow archive \cup \text{sample}(pop, sample\_size)$
    adds new member to $ensemble$    ▷ the one with the highest ensemble selection metric
    $s \leftarrow \text{select}(pop)$    ▷ tournament selection w.r.t. novelty score
    $pop' \leftarrow \emptyset$
    for model $m_i \in s$ do
        $m'_i \leftarrow \text{mutate}(m_i)$    ▷ as described in the text
        add $m'_i$ to $pop'$
    end for
    $pop \leftarrow pop'$
end for
```

Individuals are selected for reproduction on the basis of their novelty scores using a tournament selection procedure. In the interests of promoting divergence and avoiding convergence, reproduction only uses mutation. Mutation either adds or removes a randomly chosen residual block from an individual, modifying input/output sizes at the mutation point as necessary; changes the output size and dropout probability of a random block; or swaps two consecutive blocks chosen at random. After a new individual is produced, its parameters are optimised with gradient descent.

After evaluation of the entire population, n_A randomly chosen individuals are added to the *archive*, following the method suggested in [GMC15]. In addition, the individual from the population that scores the highest ensemble selection metric (Section 4.2.1) is added to the ensemble; the size of the final ensemble is therefore the number of iterations of the NS. Ensemble selection metrics are calculated w.r.t. current ensemble members; if the ensemble is the empty set (i.e. in the first iteration), then the individual with the highest novelty score is selected as its first member.

4.3.1 Evaluation of the Evolved Ensemble

In order to evaluate the performance of the evolved ensemble, we use the stacking technique [Wol92], which trains a linear model to weight the predictions of each individual learner. This linear model is trained for a configurable number of iterations on the validation set mentioned in Section 4.3. This is to avoid overfitting the test set.

4.3.2 Baseline Methods

We consider two baseline methods for comparing the results produced by our approach. The first one is a simple random search algorithm. This algorithm is identical to Algorithm 4.1, except that each new generation is initialised at random, rather than being the result of reproduction based on the novelty scores of the individual members of the previous generation. The point of this baseline is to determine the performance gain added by the NS. Models are added to the final ensemble based on the ensemble selection metric (Section 4.2.1), as previously. Note that in this case we only use behavioural metrics when calculating a mean behavioural metric between a candidate model and current ensemble members, which is used as an ensemble selection metric. There is therefore no need to calculate behavioural metrics amongst the individuals of the population, neither to keep an archive of past individuals, as these metrics are not used to guide the NS as before.

The second baseline method is an ensemble generated with mechanisms that only *implicitly* promote diversity, as described in Section 4.2.1, namely an ensemble of *architecturally diverse* neural networks, initialised with different random weights. This ensemble is created in the same fashion as the random initialisation that is used for both the first population in the NS and throughout the random search. The members of this ensemble are trained on different subsets of the data.

We do not compare our approach to any baseline that simply searches for accuracy as the importance of diversity is already established in the literature (e.g. [Die00]). We are instead interested in comparing methods which realise this notion of diversity in different ways.

4.4 Experiments

Experiments were conducted on three datasets — CIFAR-10, CIFAR-100, and SVHN. Note again that our goal with this work was not to produce models competitive with state-of-the-art results. Our methods construct ensembles made of small and shallow neural networks trained for a limited number of epochs; in contrast, the methods in the literature which achieve the best results on these datasets require considerably greater computational effort and/or extensive fine-tuning of hyperparameters (e.g. [KBZ+20, HCB+19, TL19, CZM+18, DT17, LKY18]), with some requiring at least dozens of GPU days (e.g. [SXZ+20a]). We were instead interested in studying the effects of diversity upon ensemble performance. Table 4.1 lists the common parameters whose values remain fixed throughout these experiments and Table 4.2 details the variables (hyperparameters) that are changed to generate different neural network models and

Table 4.1: Common parameters fixed throughout the experiments per method(s)

Parameter	**Method(s)**	**Value**
Evolution iterations	NS and random	10
Training epochs	NS and random	40
Training epochs	Implicit ensemble	80
Stacking epochs	All	10
Data split (CIFAR)	All	40000 train / 10000 validation
Data split (SVHN)	All	43257 / 30000
Size of subsets	Implicit ensemble	20000
Batch size	All	128
Population size	NS and random	30
K (nearest neighbours)	NS	3
Size of tournament	NS	10
Archive sample size n_A	NS	5

Table 4.2: Range of hyperparameters

Parameter	**Value**[1]
Number of blocks	2:6
Size of the first convolution	4:16:4
Size of residual blocks	24:32:4
Dropout prob. in blocks	0.1:0.4:0.1

[1] Notation is *start*:*end* or *start*:*end*:*step*

the ranges of their respective values. These hyperparameters have been chosen in order to produce small architectures that are easy to train in parallel. The NS algorithm and the random search baseline have been tested with different combinations of parameter settings, namely by instantiating the behavioural diversity metric used for calculating the novelty score of each individual and the ensemble selection diversity metric used for selecting a member of each generation to be added to the final ensemble. This is shown in Table 4.3. Note that, when running the baseline with entropy as the ensemble selection metric, the behavioural metric becomes irrelevant, as novelty scores are only ever calculated in the very first iteration in order to select the first ensemble member. For this reason, we consider only the combination with $\text{prop}_{i,j}$ when entropy is used with the random search baseline. All experiments have been carried out 10 times in order to assert statistical significance when comparing results. We next describe the hypotheses that this experimental work has put to the test.

Table 4.3: Combinations of parameter settings

Selection metric	**Mean behavioural metric**		**Entropy**	
Behavioural metric	NS	Random	NS	Random
$\mathbf{prop}_{i,j}$	✓	✓	✓	✓
$\mathbf{cos}_{i,j}$	✓	✓	✓	✗
$\mathbf{dis}_{i,j}$	✓	✓	✓	✗

Hypothesis 4.1 (Ensemble vs Single Best Individual)**.** *Ensembles constructed from a set of individual classifiers chosen to maximise behavioural diversity outperform their single best member.*

This is tested by comparing the ensembles generated by NS and the random search baseline to their single best member. Recall that our procedures only search for diversity; accuracy is obtained by training each neural network with standard gradient descent. We aim to understand whether searching for diversity alone still ensures that the selected models are accurate or if that might have a negative impact on ensemble performance.

Hypothesis 4.2 (Novelty Search vs Random Search)**.** *Ensembles evolved by the NS algorithm (Section 4.3) lead to higher test set accuracy than that of those found by the random search (Section 4.3.2).*

We expect the NS algorithm to lead to higher-performing ensembles since, unlike the random search, it not only rewards diversity, but also actively searches for it.

Hypothesis 4.3 (Novelty Search vs Implicit Diversity Ensembles)**.** *Ensembles evolved by the NS algorithm have higher test set accuracy than ensembles generated with standard methods that only implicitly promote diversity.*

In order to test this, we generate ensembles of *architecturally diverse* neural networks which are trained on *different subsets* of the data and are initialised with different random weights (Section 4.3.2). We expect the experimental work to confirm that explicitly searching for diversity leads to better ensemble accuracy than relying on implicit mechanisms for generating it.

Hypothesis 4.4 (Error Diversity Metrics vs Generic Diversity Metrics). *$prop_{i,j}$ and $cos_{i,j}$ lead to better-performing ensembles than $dis_{i,j}$. The mean behavioural metric, when used in conjunction with the first two of these, leads to better results than entropy.*

We formulate this hypothesis because error diversity has been argued to be the most important type of diversity in ensemble learning [Die00]. Both the disagreement metric and entropy are more generic metrics of diversity than those we propose, which focus more closely on error instances.

4.5 Results and Discussion

This section presents the results for the three datasets considered — CIFAR-10, CIFAR-100, and SVHN — and discusses whether or not they reject the hypotheses of Section 4.4. Table 4.4 shows the accuracy results on all datasets for the three approaches we have considered: the NS algorithm, the random search, and the ensemble built with implicit diversity mechanisms. With minimal hyperparameter fine-tuning, these results are in line with figures reported in the literature for more specialised models of similar complexity (e.g. [MV15b, CJG+15, MKHS14, JHD12, MF13, SCL12]).

4.5.1 Hypothesis 4.1

Hypothesis (Ensemble vs Single Best Individual). *Ensembles constructed from a set of individual classifiers chosen to maximise behavioural diversity outperform their single best member.*

Table 4.5 shows the results for all datasets of paired Mann-Whitney significance tests comparing the accuracy of the ensembles with that of their respective highest-performing individual. We observe that the ensembles typically outperform their single best individual in a statistically significant way. The only exceptions to this rule have been observed with the $\text{dis}_{i,j}$ behavioural metric from the literature, for which statistical significance is not observed on the CIFAR-10

Table 4.4: Accuracy results for the NS algorithm and the two baseline methods. Median values over 10 runs

Dataset		**Novelty Search**				**Random Search**				**Implicit**
		Mean b. d.		**Entropy**		**Mean b. d.**		**Entropy**		
		Ensemble	Single best	Ensemble	Single best	Ensemble	Single best	Ensemble	Single best	
CIFAR-10	$\mathbf{prop}_{i,j}$	83.51%	79.155%	81.46%	77.965%	80.54%	77.285%	73.885%	75.495%	76.315%
	$\mathbf{cos}_{i,j}$	83.29%	78.845%	81.31%	77.87%	80.61%	77.325%	✗	✗	
	$\mathbf{dis}_{i,j}$	70.30%	71.135%	67.48%	64.845%	77.13%	77.205%	✗	✗	
CIFAR-100	$\mathbf{prop}_{i,j}$	45.42%	41.405%	43.68%	40.855%	40.895%	38.855%	35.245%	37.54%	35.99%
	$\mathbf{cos}_{i,j}$	44.695%	41.065%	43.045%	40.34%	41.305%	39.155%	✗	✗	
	$\mathbf{dis}_{i,j}$	43.53%	40.03%	41.875%	39.655%	39.875%	38.33%	✗	✗	
SVHN	$\mathbf{prop}_{i,j}$	91.435%	88.805%	87.48%	85.035%	91.77%	89.72%	85.555%	84.165%	90.65%
	$\mathbf{cos}_{i,j}$	91.285%	88.555%	87.955%	84.82%	91.96%	89.395%	✗	✗	
	$\mathbf{dis}_{i,j}$	86.19%	84.345%	81.505%	74.675%	91.01%	89.035%	✗	✗	

Table 4.5: Ensemble vs single best individual. Best case shown when statistical significance at the 1% level is observed

Method	**Metrics**		**Best CIFAR-10**	**Best CIFAR-100**	**Best SVHN**
Novelty search	$\text{prop}_{i,j}$	Mean b. d.	Ensemble	Ensemble	Ensemble
		Entropy	Ensemble	Ensemble	Ensemble
	$\cos_{i,j}$	Mean b. d.	Ensemble	Ensemble	Ensemble
		Entropy	Ensemble	Ensemble	Ensemble
	$\text{dis}_{i,j}$	Mean b. d.	N/a	Ensemble	Ensemble
		Entropy	N/a	Ensemble	Ensemble
Random search	$\text{prop}_{i,j}$	Mean b. d.	Ensemble	Ensemble	Ensemble
		Entropy	Single best	Single best	N/a
	$\cos_{i,j}$	Mean b. d.	Ensemble	Ensemble	Ensemble
	$\text{dis}_{i,j}$	Mean b. d.	N/a	Ensemble	Ensemble

dataset, and with the random search baseline when entropy is the selection metric, the only case where the single best individual outperforms the ensemble. These observations meet the expectations and justify the claim of Hypothesis 4.1 that the neural networks in the evolved ensembles are both diverse and accurate and that explicitly searching for diversity alone without rewarding accuracy, at least with the NS algorithm, does not impact negatively upon ensemble performance. On the contrary, given how the ensembles evolved with NS outperform those evolved with random search, as discussed below, we argue that it is precisely this explicit search for diversity that could lead to better ensemble accuracy.

4.5.2 Hypothesis 4.2

Hypothesis (Novelty Search vs Random Search)**.** *Ensembles evolved by the NS algorithm lead to higher test set accuracy than that of those found by the random search.*

Table 4.6 shows the results of Mann-Whitney significance tests between the NS and the random search baseline. The ensembles evolved by the NS significantly outperform those evolved by the random search for all cases on CIFAR-100, as well as on CIFAR-10 except when the baseline $\text{dis}_{i,j}$ metric is the behavioural metric. The NS does not do better than the random search on SVHN. The accuracy results between the NS and the random search are very similar for the error diversity metrics $\text{prop}_{i,j}$ and $\cos_{i,j}$, as well as when entropy is the selection metric, as per Table 4.4. Interestingly, the random search considerably outperforms the NS when the disagreement metric $\text{dis}_{i,j}$ is used. As discussed later in this section, it is clear that the error diversity metrics that we propose lead to better results than the disagreement metric. The reason why the NS with these error diversity metrics does not outperform random search on SVHN requires further investigation but is likely a characteristic of the problem domain. SVHN is a simpler dataset and therefore finding the best-performing networks within the ranges defined in Table 4.2 is easier and a random search could prove the better strategy over NS. It is possible that the error diversity metrics lead the NS to trade accuracy for diversity on this dataset. In other words, if most neural network models are accurate but with similar behaviour, forcing the search to keep finding more behaviourally diverse models could result in less accurate learners, impacting ensemble performance negatively; this appears to be the case on SVHN, especially with the disagreement metric commonly used in the literature. On the hardest dataset of the three, CIFAR-100, the NS always outperforms the random search and the difference in accuracy that results from each is largest, as per Table 4.4; on the second hardest, CIFAR-10, the NS is only outperformed by the random search in combination with the underperforming disagreement metric. It is thus possible that the NS is more useful on harder problems, upon which the search must be guided by some non-random criterion. These observations justify the claim we make in Hypothesis 4.2 that, at least in some cases, explicitly searching for diversity, which the NS algorithm does, leads to better accuracy than simply rewarding it, as in the random search. However, there remain open questions regarding the conditions under which this can be observed, namely those pertaining to the problem domain and to the diversity metrics which define the search space of the NS. Further investigation into the trade-off between diversity and accuracy is required as well.

4.5.3 Hypothesis 4.3

Hypothesis (Novelty Search vs Implicit Diversity Ensembles)**.** *Ensembles evolved by the NS algorithm have higher test set accuracy than ensembles generated with standard methods that*

Table 4.6: NS vs random search. Best case shown when statistical significance at the 1% level is observed

Metrics		**Best CIFAR-10**	**Best CIFAR-100**	**Best SVHN**
$\mathrm{prop}_{i,j}$	Mean b. d.	Novelty search	Novelty search	Random search
	Entropy	Novelty search	Novelty search	N/a
$\cos_{i,j}$	Mean b. d.	Novelty search	Novelty search	Random search
$\mathrm{dis}_{i,j}$	Mean b. d.	Random search	Novelty search	Random search

only implicitly promote diversity.

Regarding the ensembles which are generated with only an implicit definition of diversity (last column of Table 4.4), significance tests show that, on CIFAR-100, both the NS and the random search significantly outperform this baseline in all cases except when random search is used with entropy; on CIFAR-10, no statistical significance is observed when the random search uses the disagreement metric $\mathrm{dis}_{i,j}$, but the baseline significantly outperforms both search methods in all other cases where this metric is used and also when compared to the random search with entropy; on SVHN, the baseline significantly outperforms the search methods in more cases, namely all those using entropy and when the NS uses the disagreement metric, with no statistically significant difference observed when the random search uses this same metric. We can therefore attest the claim of Hypothesis 4.3 by observing that the methods tend to outperform the implicit diversity baseline when the error diversity metrics that we propose in Section 4.2.1 are used; using the disagreement and entropy metrics often leads to worse performance than this baseline, likely because, as discussed before, they lead the search to excessively trade the accuracy of its individual learners for diversity of behaviours. We also note that the performance of this baseline on SVHN is close to that of the best cases produced by the NS; given its smaller complexity w.r.t. the other search methods, this means that it could be advantageous on easier datasets such as SVHN.

Table 4.7: Different diversity metrics. Best case shown when statistical significance at the 1% level is observed

Method	**Fixed metric**	**Varied metrics**		**Best CIFAR-10**	**Best CIFAR-100**	**Best SVHN**
Novelty search	Mean behavioural diversity	$\text{prop}_{i,j}$	$\cos_{i,j}$	N/a	N/a	N/a
		$\text{prop}_{i,j}$	$\text{dis}_{i,j}$	$\text{prop}_{i,j}$	$\text{prop}_{i,j}$	$\text{prop}_{i,j}$
		$\cos_{i,j}$	$\text{dis}_{i,j}$	$\cos_{i,j}$	$\cos_{i,j}$	$\cos_{i,j}$
	Entropy	$\text{prop}_{i,j}$	$\cos_{i,j}$	N/a	N/a	N/a
		$\text{prop}_{i,j}$	$\text{dis}_{i,j}$	$\text{prop}_{i,j}$	N/a	$\text{prop}_{i,j}$
		$\cos_{i,j}$	$\text{dis}_{i,j}$	$\cos_{i,j}$	N/a	$\cos_{i,j}$
	$\text{prop}_{i,j}$	Mean b. d.	Entropy	Mean b. d.	Mean b. d.	Mean b. d.
	$\cos_{i,j}$	Mean b. d.	Entropy	Mean b. d.	Mean b. d.	Mean b. d.
	$\text{dis}_{i,j}$	Mean b. d.	Entropy	Mean b. d.	N/a	Mean b. d.
Random search	Mean behavioural diversity	$\text{prop}_{i,j}$	$\cos_{i,j}$	N/a	N/a	N/a
		$\text{prop}_{i,j}$	$\text{dis}_{i,j}$	$\text{prop}_{i,j}$	N/a	$\text{prop}_{i,j}$
		$\cos_{i,j}$	$\text{dis}_{i,j}$	$\cos_{i,j}$	N/a	$\cos_{i,j}$
	$\text{prop}_{i,j}$	Mean b. d.	Entropy	Mean b. d.	Mean b. d.	Mean b. d.

4.5.4 Hypothesis 4.4

Hypothesis (Error Diversity Metrics vs Generic Diversity Metrics)**.** *$prop_{i,j}$ and $cos_{i,j}$ lead to better-performing ensembles than $dis_{i,j}$. The mean behavioural metric, when used in conjunction with the first two of these, leads to better results than entropy.*

Table 4.7 shows the results of significance tests between the accuracy results produced by different diversity metrics. We can observe that the behavioural metrics $\text{prop}_{i,j}$ and $\cos_{i,j}$ perform similarly well, with no statistically significant difference found between the results produced by these two metrics in any parameter setting. They both significantly outperform the $\text{dis}_{i,j}$ metric in the NS with the mean behavioural diversity metric as the ensemble selection metric; however, both when entropy is the selection metric and in the random search, this is not observed on all datasets. Settings that use the mean behavioural metric as an ensemble selection metric tend to statistically outperform those using entropy, for both the NS and the random search; the only case where this is not observed with statistical significance is on the

CIFAR-100 dataset when the NS uses the $\text{dis}_{i,j}$ metric.

We note that the metrics we propose, $\text{prop}_{i,j}$ and $\cos_{i,j}$, tend to lead to better results than the disagreement metric commonly found in the literature, $\text{dis}_{i,j}$. In addition, calculating the mean behavioural metric w.r.t. to the current ensemble members and using it as a selection metric tends to work better than using entropy, which as mentioned before might lead to worse ensemble performance than that of the single best individual. As we have mentioned before, the behavioural metrics we propose focus more closely on error instances, i.e. instances that at least one of the models has misclassified, unlike the common disagreement metric, which is calculated w.r.t. to all test instances. Additionally, entropy is also calculated over all instances in the test set and broadly measures the general divergence amongst the predictions made by the ensemble members. This justifies the claim we make in Hypothesis 4.4 about the diversity metrics which lead to better-performing ensembles: the error diversity metrics we propose seem to be correlated with better ensemble accuracy and are thus better suited to the task of searching for diversity. Further research into the implications of this finding is required to understand how it can be fully leveraged to evolve high-performing ensembles.

4.6 Summary

This chapter described an innovative NS algorithm which evolves ensembles by explicitly searching for behavioural diversity, unlike other methods found in the literature, which typically either rely on *implicit* mechanisms for promoting diversity or only reward it by including it as an objective while searching for models which represent trade-offs between diversity and accuracy. Our procedure not only rewards diversity, but rather actively searches for it by guiding the NS with a definition of novelty score which is based on how diverse each individual neural network is w.r.t. the other individuals in the population. The accuracy of the individual learners is ensured by a standard gradient descent procedure, but it is not taken into account in the NS. We investigate three *behavioural diversity* metrics, two of which we propose ourselves, and two metrics for selecting individuals to be added to the final ensemble, one of which is also defined by us as the mean of behavioural diversity metrics calculated w.r.t. current ensemble members and the other is the entropy of candidate ensembles.

The results showed that this approach succeeds at evolving an ensemble by explicitly searching for behavioural diversity, significantly outperforming, particularly on harder datasets, a random search baseline which merely rewards diversity and a baseline ensemble generated with implicit diversity. They also showed that the ensembles almost always outperform their best individual learner, meaning that the method is able to generate and select diverse enough learners while maintaining accuracy. Of particular relevance is the observation that, amongst the diversity metrics we considered, the error diversity metrics we proposed led to better results, i.e. they pushed the NS towards better areas of the search space. All these observations provide valuable insights into the problem of promoting diversity in ensembles of classifiers, suggesting not only that *explicit* methods such as the one presented in this chapter should be adopted on harder problems that implicit methods struggle to solve, but also that diversity metrics should focus directly and closely on the *errors* made by individual learners.

A drawback of the work presented in this chapter was that it used only small and shallow neural network models. This was a consequence of the computational complexity of the procedure: training each neural network architecture in the population with gradient descent at every step of the procedure is a resource-intensive step, but one which was necessary to calculate diversity metrics. This limited the scope of practical applicability of our algorithm to unrealistically specific and simplistic use cases. In the next chapter, we address this issue by augmenting this procedure with a surrogate model [SZZ+20] which removes the need for the costly step mentioned above. This significantly reduces the computational burden of the search procedure and expands its scope of usability by enabling larger search spaces to be considered.

Chapter 5

Surrogate Diversity Modelling

5.1 Introduction

In Chapter 4, we proposed a novelty search (NS) method that *explicitly* searches for diversity amongst a set of base learners by making use of metrics for measuring *behavioural* diversity. However, a fundamental limitation of this approach is its computational complexity, with a costly step of training all the neural network models in the population at each step of the search. Such time and computational demands compromise the ultimate goal of that approach, which is to develop learning algorithms which scale horizontally, namely with models that can be distributed across many low-cost machines. Any claim to tackling the unsustainability of Deep Learning (DL) algorithms with more scalable solutions is undermined by the fact that this approach is so computationally intensive. In order to overcome the costly step of training each model, we introduce a surrogate model [SZZ+20] into our method. We use this surrogate model, pretrained on a sample drawn from the search space of neural network architectures, to get an estimate of the *error* distance between two neural networks given architectural descriptors, *without* training these networks. Whereas this calculation was previously a very costly step, this technique renders it instantaneous.

There are two key contributions in this chapter. In Section 5.2, we introduce new diversity metrics in addition to those we proposed in Chapter 4. The second and major contribution is an improvement of our previous NS algorithm by augmenting it with a surrogate model, which is employed as mentioned above; this is detailed in Section 5.3. The experimental work is described in Section 5.4. The results, which are presented in Section 5.5, show that our new method achieves a speedup of 10 times compared to the previous approach when the same parameters are used, *without loss of performance*. By changing the parameters to expand the search space of neural network architectures, we considerably improve on the results of Chapter 4 when testing the method on three benchmark datasets from Computer Vision — CIFAR-10, CIFAR-100, and SVHN. Using a surrogate model enables us to search a wider space of neural network architectures and run the Novelty Search procedure for longer. This signifies an improved paradigm for implementing horizontal scaling of learning algorithms. The new approach makes an explicit search for diversity considerably more tractable *for the same bounded resources*. The work presented in this chapter resulted in one publication: [CHKP22].

5.2 Architectures and New Diversity Metrics

The NS operates over a space of architectures defined by a set of hyperparameters and calculates novelty scores based on the behavioural distance between the neural network models. This section describes the architectures evolved by the procedure, which are of the same type as in Chapter 4, and the diversity metrics that we propose to measure distance, some of which are new w.r.t. the previous chapter.

5.2.1 Neural Network Architectures

The architectures evolved by our procedure are *residual neural networks* [HZRS16a] based on the wide architectures proposed by [ZK16]. They are of the same kind as those we used in Chapter 4, where a full description can be found.

The *hyperparameters* of each network are evolved by NS. Each individual in the population is defined by a variable-length vector, depending on the number of blocks r: $[J, C, O^1, ..., O^r, D^1, ..., D^r]$, where J is a Boolean value indicating whether the network should be trained jointly or separately if it is in the final ensemble, C is the output size of the first convolution, O^i is the output size of block i, and D^i its dropout probability. Each individual is mapped to a Pytorch module [PGC⁺19] for implementation purposes. The *parameters* of each network are randomly initialised and then optimised by a standard gradient descent procedure.

In order to preprocess the input to the surrogate model, we *normalise* the representation described above in the following way: we first *rescale* the elements in all positions so that they lie between 0 and 1. The first element is the Boolean value indicating whether the neural network should be trained jointly or separately, so it need not be normalised. Then, given that the representations have variable length depending on the number of residual blocks in each neural network, we *pad* the vector so that it has fixed length, corresponding to the maximum possible number of residual blocks, by adding an appropriate number of elements equal to 0 before the sequence of block output sizes and before the sequence of dropout probabilities. Therefore, if the number of residual blocks in the network is r and the maximum number of blocks is R; if the maximum and minimum sizes of the first convolution in the network are C_{max} and C_{min}, respectively; if the maximum and minimum sizes of each residual block are O_{max} and O_{min}, respectively; and if the maximum and minimum dropout probability of each block are D_{max} and D_{min}; then the normalised representation of neural network m_i is:

$$\begin{aligned}\textbf{norm\_rep}_i = \Bigg[J_i, & \frac{C_i - C_{min}}{C_{max} - C_{min}}, \\ & 0, ..., 0, \frac{O_i^{R-r} - O_{min}}{O_{max} - O_{min}}, ..., \frac{O_i^{R} - O_{min}}{O_{max} - O_{min}}, \\ & 0, ..., 0, \frac{D_i^{R-r} - D_{min}}{D_{max} - D_{min}}, ..., \frac{D_i^{R} - D_{min}}{D_{max} - D_{min}} \Bigg]\end{aligned} \tag{5.1}$$

Where there are $R - r$ elements equal to 0 before the sequence of block output sizes and before the sequence of dropout probabilities.

5.2.2 Diversity Metrics

In order to calculate novelty scores, which are used as the objective function by the NS, we have considered six different diversity metrics, five of which we have defined ourselves. These metrics are calculated between each pair of individual neural network architectures. We used three of these metrics in the previous version of our procedure, which is described in Chapter 4.

Let $\boldsymbol{y}_i$ be the vector of predictions for model m_i with each prediction y_i^n for data point $\boldsymbol{x}^n$ being a class label in $\{1..C\}$. Let $\boldsymbol{p}_i$ be a binary vector where $p_i^n = 1$ if the prediction y_i^n is correct and $p_i^n = 0$ otherwise. Let N^{11}, N^{00}, N^{01}, and N^{10}, respectively, be the total number of test instances where two models are both correct, both incorrect, and when one is correct and the other is not. The first diversity metric we consider is the *proportion of different errors* between two models when at least one of them is *correct*. We expect it to provide insight into the divergence between the errors made by two models. It is defined as:

$$\mathrm{prop}^1_{i,j} = \frac{N^{01} + N^{10}}{N^{11} + N^{01} + N^{10}} \tag{5.2}$$

The second diversity metric we consider is very similar and is the *proportion of different errors* between two models when at least one of them is *incorrect*. We first proposed this metric in Chapter 4. We repeat it here for clarity:

$$\mathrm{prop}^2_{i,j} = \frac{N^{01} + N^{10}}{N^{00} + N^{01} + N^{10}} \tag{5.3}$$

The third metric we propose is the *harmonic mean* between these two proportion metrics. This is a sound way of averaging the two proportion metrics into a single metric so that they are both taken into account. It is defined as:

$$\mathrm{prop}^{\mathrm{harm}}_{i,j} = \frac{2 \cdot \mathrm{prop}^1_{i,j} \cdot \mathrm{prop}^2_{i,j}}{\mathrm{prop}^1_{i,j} + \mathrm{prop}^2_{i,j}} \tag{5.4}$$

We also consider a widely used metric (e.g. [Van05, KW03, PDCV10]) defined as the *disagree-*

ment between two models, i.e. the proportion of test instances where one is correct and the other is not. We take this metric into account since it expresses how commonly two models disagree on any test instance. It is defined as:

$$\mathrm{dis}_{i,j} = \frac{N^{01} + N^{10}}{N^{00} + N^{01} + N^{10} + N^{11}} \tag{5.5}$$

Consider now the *two's complement* of the binary vector of correct predictions $\boldsymbol{p}_i$, $\boldsymbol{w}_i$, i.e. the binary vector of *wrong* predictions. The next metric we propose is the *cosine distance* between the binary vectors of wrong predictions made by two models m_i and m_j. Like $\mathrm{prop}^1_{i,j}$ and $\mathrm{prop}^2_{i,j}$, we consider this metric because it is a measure of the distance between the errors made by two models. We have defined it as:

$$\mathrm{cos\_dist}_{i,j} = 1 - \frac{\boldsymbol{w}_i \cdot \boldsymbol{w}_j}{\|\boldsymbol{w}_i\| \|\boldsymbol{w}_j\|} \tag{5.6}$$

Finally, we consider a metric of *architectural* diversity. Take the *normalised* vector which represents each individual neural network, as described in Section 5.2.1. Let its size be L. To obtain an architectural representation, we simply remove the first element from the normalised representation, i.e. the Boolean value indicating whether or not the neural network should be trained separately or jointly. Thus, referring to Equation 5.1, the architectural representation of model m_i is:

$$\mathbf{arch\_rep}_i = \mathbf{norm\_rep}_i^{\{1..L-1\}} \tag{5.7}$$

We then define *architectural distance* between neural networks m_i and m_j as the cosine distance between their normalised architectural representations:

$$\mathrm{arch\_dist}_{i,j} = 1 - \frac{\mathbf{arch\_rep}_i \cdot \mathbf{arch\_rep}_j}{\|\mathbf{arch\_rep}_i\| \|\mathbf{arch\_rep}_j\|} \tag{5.8}$$

These metrics determine the *behavioural distance* between two neural network models, which is used to calculate the novelty scores that guide the NS procedure, as explained in Section 5.3.3. Note that the metrics $\text{prop}^2_{i,j}$ and $\text{cos\_dist}_{i,j}$ focus more closely on the instances where the models made a *prediction error*. In the work of Chapter 4, these two metrics have led to better performance than the others. Here we are interested in learning whether the same pattern can be observed with our new improved version of the NS procedure.

5.3 Novelty Search Augmented with a Surrogate Model

We use NS to evolve an ensemble of behaviourally diverse neural network models. Unlike the work of Chapter 4, where the neural networks in a generation had to be trained with gradient descent at each iteration of the NS in order to calculate the behavioural distances between each pair of architectures, these distances are now *estimated* by a surrogate model which is pretrained on a sample drawn from the space of neural network architectures. The most diverse models are added to the final ensemble, which is then trained on the input data. This section provides detail into these steps.

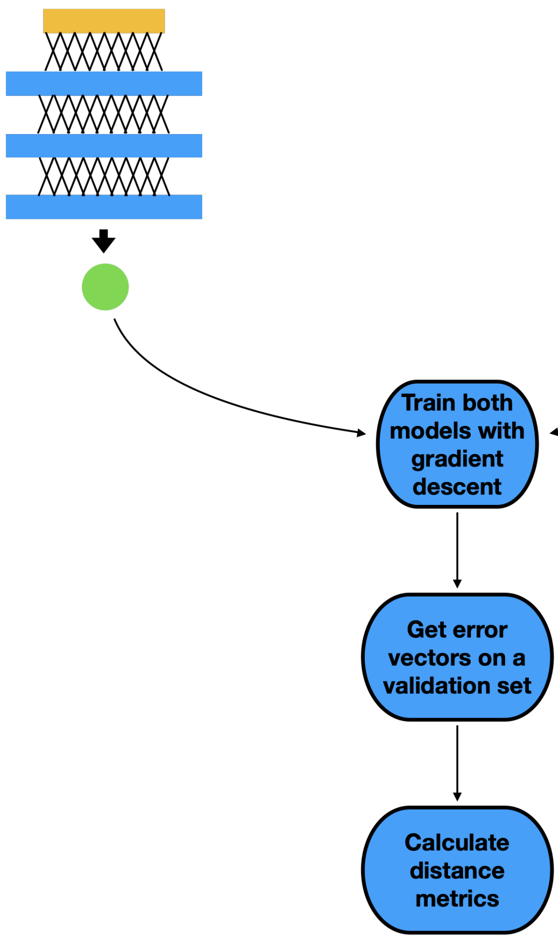
5.3.1 Surrogate Model to Estimate Distances

The NS requires novelty scores to be determined, which in turn require the distances between pairs of neural networks in the current population to be calculated. However, calculating the exact distance values between two neural network models entails first training the models on the input data with gradient descent and then evaluating them on a validation dataset, as we did in Chapter 4. This can be a very costly step if computational resources are limited, which constrains the NS to only a few iterations and the population to a small size — as the neural networks have to be trained in parallel for efficiency. Here we overcome this limitation by pretraining a Random Forest [Bre01] surrogate model which estimates the behavioural distances between a pair of neural network models.

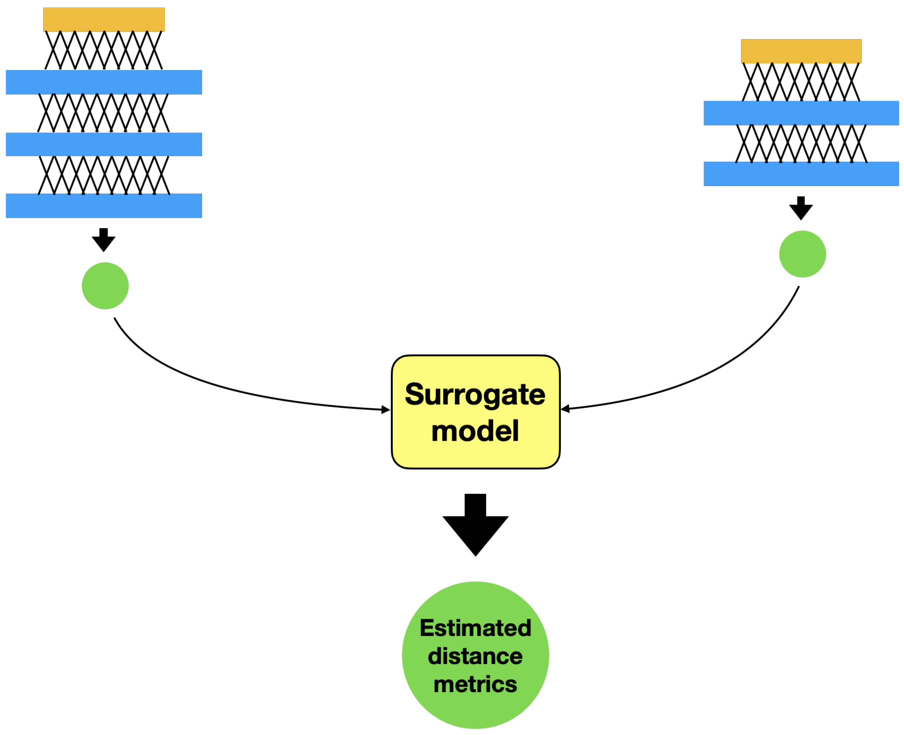
Note that the estimates of behavioural distances produced by the surrogate model do not need to be very accurate. This is because, when calculating the novelty score of a particular individual neural network, we only need to know relative distances in order to determine nearest neighbours. This means that the surrogate model need only capture the general trends of growth of the distance values, even if the actual values are not very precise. This makes the use of a surrogate model very appropriate with no need for a very complex model. Figure 5.1 shows the differences between the previous method for calculating exact distance values, shown in Figure 5.1a, and the current method using a surrogate model, shown in Figure 5.1b. Calculating exact distance values is a very costly step, potentially requiring several GPU hours depending on the length of training. In contrast, estimating these distances by means of the surrogate model is an instantaneous process, once the surrogate model has been trained on sample data beforehand.

5.3.2 Pretraining the Surrogate Model

The surrogate model must be trained beforehand so that it can be used effectively during the NS to estimate the distance values between two neural network models. To do this, we draw a sample of neural networks from the search space of architectures defined by the set of hyperparameters used with the NS method. We first train each of these neural networks with gradient descent and calculate their error vectors on a validation data set. We then build random pairs of neural networks and calculate the exact distance values, for all six metrics considered, between them as a function of either their error vectors or their architectural descriptors, as explained in Section 5.2.2. Finally, we construct a data set on which we fit a Random Forest regressor [Bre01] which takes as input the normalised representations of two neural network architectures, as per Section 5.2.1, and has six outputs: the estimates of the distance values for all six metrics considered. We have selected a Random Forest model due to its low complexity and because we expect it to generalise well on new data, given that it is an ensemble model. Algorithm 5.1 describes the process of training this surrogate model in pseudocode.


Train both models with gradient descent
Get error vectors on a validation set
Calculate distance metrics

(a) Steps required to calculate exact distance values between two neural networks



(b) Estimating the distance values between two neural networks with a surrogate model

Figure 5.1: Difference between calculating and estimating distance values

Algorithm 5.1 Pretraining the surrogate model on sample architectures

```
draw a sample $S$ of neural networks from the search space defined by $J$, $[C_{min}, C_{max}]$, $[O_{min}, O_{max}]$, and $[D_{min}, D_{max}]$;
train($S$);        ▷ Models trained jointly or separately according to the value of $J_i$
for neural network model $m_i \in S$ do
    get error vector $e_i$ on validation set $\mathcal{D}_{val}$;
end for
build $\frac{\|S\|^2 - \|S\|}{2}$ unique pairs of neural networks;
initialise dataset $\mathcal{D}_{dists} \leftarrow \emptyset$;
for each pair $m_i$, $m_j$ do
    $\boldsymbol{d} \leftarrow$ all 6 distance values;        ▷ calculated as per Section 5.2.2
    norm_rep$_i$, norm_rep$_j$ are the normalised representations of $m_i$ and $m_j$;
    add data point $x \leftarrow \{\text{norm\_rep}_i, \text{norm\_rep}_j, \boldsymbol{d}\}$ to $\mathcal{D}_{dists}$;
end for
train random forest model $rf$ on $\mathcal{D}_{dists}$;
return random forest model $rf$;
```

5.3.3 Novelty Search Algorithm

Our algorithm for building an ensemble implements NS as described by [LS11a], applying it to our problem domain. Algorithm 5.2 presents the pseudocode for this procedure. The original training data is split into two sets, one for training and one for validation. The training set is used to train the final ensemble; it is also used to train the sample of neural network architectures drawn from the search space that is in turn used to pretrain the surrogate model. Whereas training each of these sample neural networks makes use of the entire training set, pretraining the surrogate model only requires the validation set, which is used to calculate exact distance values between pairs of neural networks.

Selection in NS is driven by the novelty score, which computes the sparseness at any point in the behavioural space. This sparseness is defined by one of the distance metrics of Section 5.2.2. Areas with denser clusters of visited points are considered less novel and therefore rewarded less. This is defined as the average distance to the K-nearest neighbours of a point, calculated with respect to the other individuals in the current generation and to a stored *archive* of previously sampled solutions. Hence, the novelty score is calculated as:

$$NS_i = \frac{1}{k} \sum_{k=0}^{K} \text{div\_metric}(m_i, \mu_k) \tag{5.9}$$

Where μ_k is the kth-nearest neighbour of m_i with respect to the diversity metric $\text{div\_metric}_{i,j}$, selected from the metrics defined in Section 5.2.2.

Individuals are selected for reproduction on the basis of their novelty scores using a tournament selection procedure. In the interests of promoting divergence and avoiding convergence, reproduction only uses mutation. Mutation either adds or removes a randomly chosen residual block from an individual, modifying input/output sizes at the mutation point as necessary; changes the output size and dropout probability of a random block; or swaps two consecutive blocks chosen at random.

After evaluating the entire population, n_A randomly chosen individuals are added to the *archive*, following the method suggested in [GMC15]. In addition, the individual from the population with the highest *elite score*, calculated in a similar fashion to the novelty score, is added to an *elite archive*. After running the NS for the specified number of iterations, a subset of this elite archive is selected as the final ensemble. This subset is chosen so as to maximise the average distance amongst its members. The final ensemble is then trained by gradient descent, the *only time* when this parameter optimisation takes place.

5.3.4 Evaluation of Evolved Ensembles

In order to evaluate the performance of the evolved ensemble, we use the stacking technique [Wol92], which trains a linear model to weight the predictions of each individual learner. This linear model is trained for a configurable number of iterations on the validation set mentioned in Section 5.3.3. This is to avoid overfitting the test set.

5.3.5 Baseline: Previous Method

The approach we present here is an improvement of the method that we propose in Chapter 4. We use this as a baseline against which we compare the new method. In its main aspects, the previous method is similar to the new method, with the notable difference that it does not

Algorithm 5.2 Ensemble evolution through NS

```
randomly initialise population $pop$;
$archive \leftarrow \emptyset$;
$elite\_archive \leftarrow \emptyset$;
draw $\mathcal{D}_{train}$ and $\mathcal{D}_{val}$ from training set $\mathcal{D}$;
set evolution iterations $epochs$;
set archive sample size $n_A$;
set final ensemble size $ensemble\_size$;
surrogate model $s_{div}$ pretrained as per Algorithm 5.1;
select diversity $\text{div\_metric}_{i,j}$ from Section 5.2.2;
for $epochs$ do
    for $m_i, m_j \in pop \times pop \cup archive : m_i \neq m_j$ do
        $\text{div\_metric}_{i,j} \approx s_{div}(m_i, m_j)$
    end for
    for $m_i, m_j \in pop \times elite\_archive$ do
        $\text{div\_metric}'_{i,j} \approx s_{div}(m_i, m_j)$
    end for
    for $m_i \in pop$ do
        $NS_i \leftarrow \frac{1}{k}\sum_{k=0}^{K} \text{div\_metric}(m_i, \mu_k)$        ▷ Equation 5.9
        $NS'_i \leftarrow \sum_{m_j \in elite\_archive} \text{div\_metric}'(m_i, m_j)$
    end for
    $sample \leftarrow \text{random\_sample}(pop, n_A)$
    $archive \leftarrow archive \cup sample$
    $el\_best \leftarrow \max(pop, NS'_i)$
    $elite\_archive \leftarrow elite\_archive \cup \{el\_best\}$
    $s \leftarrow \text{tournament\_select}(pop, NS_i)$
    $pop \leftarrow \text{mutate}(s)$
end for
for $m_i, m_j \in elite\_archive \times elite\_archive : m_i \neq m_j$ do
    $\text{div\_metric}^*_{i,j} \approx s_{div}(m_i, m_j)$
end for
for $m_i \in elite\_archive$ do
    $NS^*_i \leftarrow \sum_{m_j \in elite\_archive : m_i \neq m_j} \text{div\_metric}^*(m_i, m_j)$
end for
$ensemble \leftarrow \max(elite\_archive, NS^*_i, ensemble\_size)$
train($ensemble$);        ▷ Models trained jointly or separately according to the value of $J_i$
```

make use of a surrogate model to estimate the distance between two neural network models. As discussed previously, this original method calculates exact distances between each pair of neural networks by first training all the models in the current generation with gradient descent and then getting their error vectors by evaluating them on a validation set. As an additional difference, the previous method would calculate at each iteration an *ensemble selection metric* for each member of the population and then add to the final ensemble the single best-scoring neural network in each generation. The new method maintains an *elite archive*, to which a sample of neural networks from each generation with highest novelty score with respect to this archive, which we call *elite score*, is added at each iteration; novelty scores with respect to the final elite archive are calculated at the end for each of its members and this *ensemble score* is used to select a subset of neural networks which will make up the final ensemble.

We compare the new method proposed in this paper with our previous approach along two main lines. Firstly, we seek to understand whether there is a speedup with the new method as a result of increased efficiency when looking for solutions of similar complexity to those found with the previous method. Secondly, we investigate whether the new method can be used to produce better solutions, i.e. solutions of higher complexity and leading to better final performance. This means that we are interested in investigating whether there is both a quantitative improvement, i.e. being able to do *more of* what could be done with the previous method thanks to a more efficient use of computational resources, and a qualitative improvement, i.e. being able to do *more than* what could be done with the previous method by tackling solutions that were previously unfeasible or intractable.

5.4 Experiments

This section describes the two sets of experiments carried out for comparing the new NS method, which makes use of a surrogate model to estimate the distance between models as described previously, with the previous method, which instead calculates exact distance values by first training all the models in the population with gradient descent and then determining their error vectors on a validation set. We compare the methods based on resource usage, namely runtime, for similar parameter settings and model complexity, as well as on their ability to scale to larger search spaces and search for more complex models.

5.4.1 Test set 1: Resource Usage for Similar Complexity

In this set of tests, we investigate the total time required to run each of the two methods when they are looking for *solutions of the same complexity* and running for the same number of iterations. We wish to determine the speedup that can be gained with the new method, which makes use of a surrogate model to overcome the need for training all the models in the current generation with gradient descent in order to calculate novelty scores. We run both the new and the previous methods on CIFAR-10 and fix the parameters, as shown in the second column of Table 5.1. We conjecture that in these conditions our new method not only results in a speedup due to the use of a Random Forest surrogate model, but also outputs ensembles of similar performance. This is expressed by Hypotheses 5.1 and 5.2.

Hypothesis 5.1 (Runtime of previous NS method vs new method enhanced with a surrogate model)**.** *Enhancing the NS procedure with a Random Forest surrogate model pretrained to estimate the distance between models, and thereby their novelty scores, results in a speedup compared with our previous method, which calculates exact distance values and novelty scores, when constructing ensembles of the same complexity.*

Hypothesis 5.2 (Performance achieved with the previous NS method vs the new method with a surrogate model)**.** *When looking for solutions of the same complexity, the new NS procedure, using a surrogate model, outputs ensembles which do not perform worse than those constructed by our previous method, even though the new method only* estimates *distance values and novelty scores.*

Table 5.1: Novelty search parameters for both test sets

Parameter	**Test set 1: runtime comparison (both methods)**	**Test set 2: expanded search space (new method only)**
Iterations	10	100
Final ensemble size	11	40
Population size	30	100
Diversity metric	$\text{cos\_dist}_{i,j}$	All from Section 5.2.2
Number of blocks	2:6	2:6
Number of channels in the first convolution	4:16	4:16
Number of channels in residual blocks	24:32	16:64
Dropout probability in residual blocks	0.1:0.4	0.1:0.9
Number of neighbours K	3	15
Size n_A of archive sample	5	10
Size of tournament for selection	10	50

5.4.2 Test set 2: Expanding the Search Space

Using a surrogate model to speed up the procedure has enabled us to both search for solutions of higher complexity and run the NS for longer. In this set of experiments, we apply the new method to three benchmark datasets from the Computer Vision (CV) literature — CIFAR-10, CIFAR-100, and SVHN — and test it with all diversity metrics previously defined in Section 5.2.2. We also compare the results achieved with the new method to the best results observed with the previous method. The parameters that we use with the new method are shown in the third column of Table 5.1; they correspond to the *expanded* search space made possible by the use of a surrogate model. We expected to see further evidence of what we observed in previous work, presented in Chapter 4, regarding *error diversity* metrics, namely that those diversity metrics which focus more closely on the instances where the models make prediction errors lead to higher-performing ensembles. This is expressed by Hypothesis 5.3. We also expect the new method to lead to higher-performing ensembles than those constructed with the previous method, since the use of a surrogate model makes it feasible to expand the search space and run the NS for longer. This is expressed by Hypothesis 5.4.

Hypothesis 5.3 (Better performance with metrics that focus on error instances)**.** *In a similar fashion to what we have observed with our previous method, running the NS procedure with the distance metrics that focus more closely on the instances where the models make prediction errors leads to higher-performing ensembles than when more generic diversity metrics are employed.*

Hypothesis 5.4 (Performance achieved with the previous NS method vs the new method with a surrogate model)**.** *The new NS method enhanced with a surrogate model makes it possible to search a larger space of more complex neural network architectures and, therefore, outputs higher-performing ensembles than the best ones constructed by our previous method.*

5.5 Results and Discussion

In this section, we present the results of the two sets of experiments described in Section 5.4. We then discuss these results and whether the hypotheses formulated above can be rejected.

5.5.1 Hypothesis 5.1

Hypothesis (Runtime of previous NS method vs new method enhanced with a surrogate model)**.** *Enhancing the NS procedure with a Random Forest surrogate model pretrained to estimate the distance between models, and thereby their novelty scores, results in a speedup compared with our previous method, which calculates exact distance values and novelty scores, when constructing ensembles of the same complexity.*

Table 5.2 shows the median value, calculated after 10 independent runs, of the time required to run both the previous NS method and the new method, which makes use of a surrogate model, with the same parameters. These results show that the new method is about 10 times faster than the original NS method. A Mann-Whitney significance test shows that this difference is significant at the 1% level. This supports the claim of Hypothesis 5.1 that enhancing the NS method with a Random Forest surrogate model to estimate the distances between models speeds

Table 5.2: Median results over 10 runs of the previous NS method and the new NS method with a surrogate model on CIFAR-10 (test set 1)

Runtime of NS	48760.5 s
Runtime of NS with surrogate model	4871 s
Training a sample of architectures	28970.5 s
Building a dataset and training the Random Forest surrogate model	18113.5 s
Accuracy achieved by NS	82.245%
Accuracy achieved by NS with surrogate model	83.885%

up the search for diverse models and the construction of a diverse ensemble. For reference, we also report in Table 5.2 the median time, over 10 runs, required to train a sample of 40 neural network architectures on CIFAR-10, as well as to build a dataset and train the Random Forest surrogate model as per Algorithm 5.1. Note that these two runtimes are a *one-off cost* and that, in order to pretrain the surrogate model for our experiments, we have trained a total of 3200 sample architectures by running several processes in parallel on a cluster, each training 40 architectures.

5.5.2 Hypothesis 5.2

Hypothesis (Performance achieved with the previous NS method vs the new method with a surrogate model). *When looking for solutions of the same complexity, the new NS procedure, using a surrogate model, outputs ensembles which do not perform worse than those constructed by our previous method, even though the new method only* estimates *distance values and novelty scores.*

Table 5.2 also shows the median accuracy, calculated after 10 independent runs, achieved by ensembles constructed by both the previous NS method and the new method, when these are executed with the same parameters. The results show that the ensembles constructed by the new method do not perform worse than those constructed by the original method, which calculates exact values for the distance metrics and novelty scores. In fact, we observe that the new method leads to slightly better performance. A Mann-Whitney significance test shows that this difference is significant at the 1% level. This corroborates Hypothesis 5.2, which claims that there is no loss in performance when using the new method and its surrogate estimates. Besides the use of surrogate models, the major difference between the previous and the new method is the way a subset of all the models is selected to be in the final ensemble. As explained before, the previous method applies an *ensemble selection metric* at each iteration of the NS, whereas the new method keeps an *elite archive*, from which the final ensemble is selected in an additional step at the end of the procedure. It seems that the ensemble selection procedure of the new method is the cause behind the better performance achieved by its ensembles.

5.5.3 Hypothesis 5.3

Hypothesis (Better performance with metrics that focus on error instances). *In a similar fashion to what we have observed with our previous method, running the NS procedure with the distance metrics that focus more closely on the instances where the models make prediction errors leads to higher-performing ensembles than when more generic diversity metrics are employed.*

Table 5.3 shows the median accuracy, after 10 runs, of ensembles evolved by the new NS procedure extended with a surrogate model, for all six diversity metrics of Section 5.2.2 and all three datasets considered. We observe that on CIFAR-10 and SVHN, the metrics $\text{prop}^2_{i,j}$ and $\text{cos\_dist}_{i,j}$ lead to the highest-performing ensembles. Mann-Whitney tests show that the difference to the other metrics is statistically significant. On CIFAR-100, this is observed additionally with the metrics $\text{prop}^{\text{harm}}_{i,j}$ and $\text{dis}_{i,j}$.

The metrics $\text{prop}^2_{i,j}$ and $\text{cos\_dist}_{i,j}$ are the two that focus more closely on the instances where the two models being compared make prediction errors. Additionally, the metric $\text{prop}^{\text{harm}}_{i,j}$ depends on the value of $\text{prop}^2_{i,j}$. These observations back the claim of Hypothesis 5.3 that error diversity metrics lead to better-performing ensembles compared to more generic diversity metrics. This confirms our observations in Chapter 4.

Table 5.3: Median accuracy over 10 runs of ensembles constructed by the new method (test set 2). Best results highlighted. Best results with the original NS shown for comparison

Dataset	**Diversity metric**	**Final ensemble accuracy**	**Best accuracy with original NS (from Chapter 4)**
CIFAR-10	$\text{prop}^1_{i,j}$	67.295%	83.51%
	$\text{prop}^2_{i,j}$	90.605%	
	$\text{prop}^{\text{harm}}_{i,j}$	83.975%	
	$\text{dis}_{i,j}$	86.28%	
	$\text{cos\_dist}_{i,j}$	90.11%	
	$\text{arch\_dist}_{i,j}$	80.4%	
CIFAR-100	$\text{prop}^1_{i,j}$	28.725%	45.42%
	$\text{prop}^2_{i,j}$	63.05%	
	$\text{prop}^{\text{harm}}_{i,j}$	63.41%	
	$\text{dis}_{i,j}$	63.18%	
	$\text{cos\_dist}_{i,j}$	63.035%	
	$\text{arch\_dist}_{i,j}$	49.83%	
SVHN	$\text{prop}^1_{i,j}$	78.825%	91.435%
	$\text{prop}^2_{i,j}$	94.8%	
	$\text{prop}^{\text{harm}}_{i,j}$	89.775%	
	$\text{dis}_{i,j}$	90.675%	
	$\text{cos\_dist}_{i,j}$	94.79%	
	$\text{arch\_dist}_{i,j}$	90.68%	

5.5.4 Hypothesis 5.4

Hypothesis (Performance achieved with the previous NS method vs the new method with a surrogate model)**.** *The new NS method enhanced with a surrogate model makes it possible to search a larger space of more complex neural network architectures and, therefore, outputs higher-performing ensembles than the best ones constructed by our previous method.*

The last column of Table 5.3 shows the best performance achieved by ensembles evolved with our previous NS method. These results show very clearly that the new method constructs higher-performing ensembles than our previous procedure, with the most considerable difference being observed on CIFAR-100 and CIFAR-10. Mann-Whitney tests reveal that, for each dataset, the difference between the best results achieved by the new method and the best achieved by the previous method is indeed statistically significant. This difference results from the fact that the new method, thanks to its use of a surrogate model, is able to *search a wider space of neural network architectures*, even though it runs on *the same bounded resources*. We conclude that this supports Hypothesis 5.4.

5.6 Summary

This chapter described a significant extension to the work presented in Chapter 4, where we proposed an innovative NS method to build behaviourally diverse ensembles of classifiers. The previous method had signposted an innovative way to construct high-performing ensembles by explicitly searching for diversity. However, its application in practice had been hampered by limitations in the amount of available computational resources, since it involved a time-consuming step of training all networks in each generation of the NS with gradient descent. The new method overcomes this limitation by using a pretrained surrogate model to estimate the distance between neural network architectures, necessary to calculate novelty scores, without the need to train them. In this way, we obtained an approximate speedup of 10 times w.r.t. the previous method when running them both with the same parameters, *without loss of* *classification accuracy*. We were also able to construct better-performing ensembles thanks to the expanded architecture search space facilitated by using a surrogate. We confirmed previous observations that error diversity metrics lead to better-performing ensembles than more generic metrics.

This new method thus represented an improved paradigm for implementing horizontal scaling of learning algorithms. It made an explicit search for diversity considerably more tractable than the approach of Chapter 4 *for the same bounded resources*. In the next chapter, we extend this algorithm with four new methods which incorporate *accuracy* objectives into the NS, with the goal of studying the relationship between diversity and classification accuracy and to characterise the trade-offs between the two.

Chapter 6

The Diversity-Accuracy Duality in Ensembles of Classifiers

6.1 Introduction

As we have seen, defining a classifier ensemble requires two phases: creating a large set of potential classifiers, then selecting an appropriate subset to form an ensemble. Accuracy can be achieved by training the learners with stochastic gradient descent (SGD), while diversity can be achieved using implicit or explicit techniques. The former include techniques such as training the models on different subsets of the data or starting from different random initialisations [GCJ15], while the latter can be achieved by methods that explicitly create architectural or behavioural diversity, such as the ones we propose in this thesis.

The trade-off between diversity and accuracy in ensemble learning has been the subject of extensive research, e.g. [GJ14, OAWS15, ZZZ07]. In this chapter, we consider several strategies for combining diversity and accuracy objectives along the two phases mentioned above, ranging the full spectrum between favouring only explicit diversity and only explicit individual model accuracy, with different combinations in between. We measure diversity with a number of *diversity metrics* and make use of surrogate models — following the method described in Chapter 5 — to reduce computational burden, which facilitates an extensive search for potential architectures and, therefore, ensembles. The surrogate models are used to estimate (1) the distance between neural network architectures, which is required to drive the novelty search (NS) method, and (2) the accuracy of a network. In this way, we are able to conduct a thorough study to investigate whether there is indeed a fundamental tension between accuracy and diversity. A high-level view of the generic NS method is given in Section 6.2 and a detailed description is provided in Section 6.4.

The major contribution of this chapter arises from the experimental work, described in Section 6.5. Results on three problems from Computer Vision (CV) — CIFAR-10, CIFAR-100 [Kri09], and SVHN [NW11] — are presented in Section 6.6. They show that incorporating accuracy objectives significantly improves ensemble accuracy for the worst-performing diversity metrics, but not for the best ones. However, they also reveal the surprising result that there are multiple equivalent ways of combining the best diversity metrics with accuracy objectives that lead to ensembles of similar measured diversity and average individual accuracy. This includes even a method which only makes use of explicit accuracy objectives, with diversity being generated in an implicit manner by the evolutionary procedure. This means that, in the cases we study here, there is no dichotomy between diversity and accuracy in ensembles of classifiers; each contributes to final performance without detriment to the other and weighting one more does not impact negatively upon the other. For the problem domains we tackle, there appears to be a fundamental *equivalence* between searching for diversity and searching for individual model accuracy. This equivalence between utilising diversity or accuracy objectives suggests that the two are interchangeable in some conditions and, therefore, enables us to posit the existence of a *diversity-accuracy duality* in ensembles of classifiers. Although further investigation is required, this result is nevertheless significant because it challenges notions about the need to trade off diversity for accuracy. An implication of this is the possibility of designing better algorithms which evolve diverse ensembles while the accuracy of base learners is implicitly ensured. The work presented in this chapter resulted in one paper which has been accepted at GECCO 2022.

6.2 A High-Level Conceptual View

Figure 6.1 shows a high-level view of a generic population-based search method for constructing ensembles of classifiers, which extends the NS method of Chapter 5. This method has two phases: a search phase and an ensemble selection phase. During the search phase, a large set of candidate classifiers is created. Firstly, the distance between each pair of neural network models in the population, and between each member of the population and each member of the *search archive*, is estimated by a *surrogate diversity* model. The accuracy of each neural network architecture is also estimated by a *surrogate accuracy* model. These distance and accuracy estimates are then used to calculate three scores: (1) a fitness $score_i$, which is used to evolve a new generation at each iteration to replace the current population; (2) an archive score $arch\_score_i$, which is used to select a sample of models to be added to the search archive at each iteration; and (3) an elite score el_score_i, which determines a single neural network to be added to an *elite archive* at each iteration. The way these scores are calculated is determined by the particular method from Section 6.4.2 with which this generic search procedure is instantiated.

In the ensemble selection phase, a subset of the neural networks in the elite archive is selected to become the final ensemble. These are the top S networks which score the highest ensemble score en_score_i. Again, the way this score is calculated depends on the particular search method. The methods we propose in this paper represent different combinations of diversity and accuracy objectives along the two phases of the generic search algorithm. Those different combinations are reflected in the four scores mentioned above. As the last step, the final ensemble is trained on a training set with stochastic gradient descent (SGD) and a linear stacking model [Wol92] is trained on a validation set to weight the predictions of each individual learner.

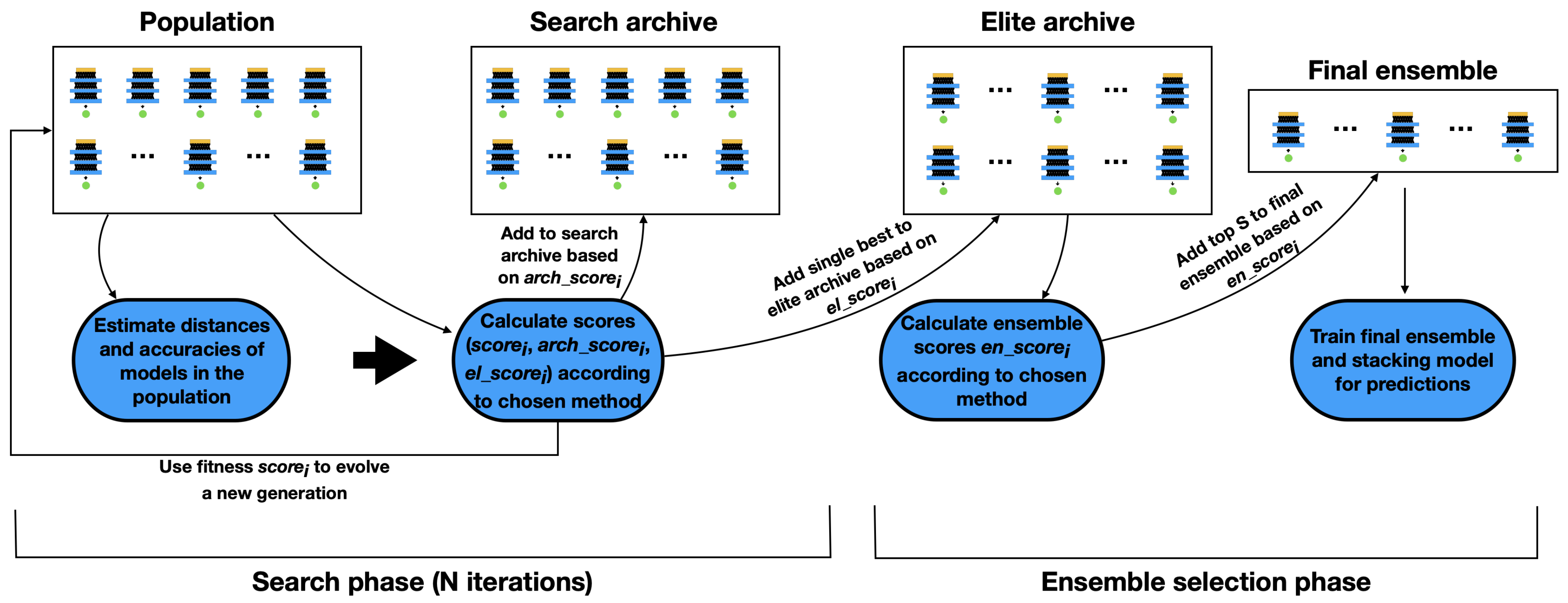

Population
Search archive
Elite archive
Final ensemble
Estimate distances and accuracies of models in the population
Calculate scores ($score_i$, $arch\_score_i$, el_score_i) according to chosen method
Add to search archive based on $arch\_score_i$
Add single best to elite archive based on el_score_i
Calculate ensemble scores en_score_i according to chosen method
Add top S to final ensemble based on en_score_i
Train final ensemble and stacking model for predictions
Use fitness $score_i$ to evolve a new generation
Search phase (N iterations)
Ensemble selection phase

Figure 6.1: A high-level view of the two phases of a generic search method for constructing a classifier ensemble. First, a large elite archive is created as a result of a population-based evolutionary search. Then a subset of the elite archive is selected to form an ensemble

6.3 Architectures and Diversity Metrics

Our NS procedure augmented with accuracy objectives evolves architectures of the same type as in Chapters 4 and 5. The diversity metrics that we employ here are the same as those defined in Chapter 5. In the interests of clarity, this section provides a brief summary of both the architectures and the diversity metrics.

6.3.1 Neural Network Architectures

Here we evolve neural network architectures — which are then trained with a stochastic gradient descent (SGD) procedure — of the same type as in previous chapters, based on the wide residual network architectures proposed by [ZK16]. We take the same representation for each individual in the population as in Chapter 5. This is a variable-length vector, depending on the number of blocks r: $[J, C, O^1, ..., O^r, D^1, ..., D^r]$, where J is a Boolean value indicating whether the network should be trained jointly or separately if it is in the final ensemble, C is the output size of the first convolution, O^i is the output size of block i, and D^i its dropout probability. Before being mapped to a Pytorch module [PGC+19], this representation is *normalised* as in Chapter 5 by rescaling all elements so that they lie between 0 and 1 and padding the vector so that it becomes fixed-length.

6.3.2 Diversity Metrics

In order to calculate novelty scores, we employ the diversity metrics proposed in Chapter 5. These metrics are calculated between each pair of individual neural network architectures. Let $\boldsymbol{y}_i$ be the vector of predictions for model m_i with each prediction y_i^n for data point $\boldsymbol{x}^n$ being a class label in $\{1..C\}$. Let $\boldsymbol{p}_i$ be a binary vector where $p_i^n = 1$ if the prediction y_i^n is correct and $p_i^n = 0$ otherwise. Let N^{11}, N^{00}, N^{01}, and N^{10}, respectively, be the total number of test instances where two models are both correct, both incorrect, and when one is correct and the other is not. The first diversity metric we consider is the *proportion of different errors* between two models when at least one of them is *correct*:

$$\text{prop}^1_{i,j} = \frac{N^{01} + N^{10}}{N^{11} + N^{01} + N^{10}} \tag{6.1}$$

The second diversity metric we use is very similar and is the *proportion of different errors* between two models when at least one of them is *incorrect*:

$$\text{prop}^2_{i,j} = \frac{N^{01} + N^{10}}{N^{00} + N^{01} + N^{10}} \tag{6.2}$$

The third metric we use is the *harmonic mean* between these two proportion metrics:

$$\text{prop}^{\text{harm}}_{i,j} = \frac{2 \cdot \text{prop}^1_{i,j} \cdot \text{prop}^2_{i,j}}{\text{prop}^1_{i,j} + \text{prop}^2_{i,j}} \tag{6.3}$$

We also consider the *disagreement* metric [Van05, KW03, PDCV10], i.e. the proportion of test instances where one of the models is correct and the other is not:

$$\text{dis}_{i,j} = \frac{N^{01} + N^{10}}{N^{00} + N^{01} + N^{10} + N^{11}} \tag{6.4}$$

Let $\boldsymbol{w}_i$ be the two's complement of $\boldsymbol{p}_i$, i.e. the binary vector of *wrong* predictions of model m_i. We consider the *cosine distance* between the binary vectors of wrong predictions made by two models m_i and m_j:

$$\text{cos\_dist}_{i,j} = 1 - \frac{\boldsymbol{w}_i \cdot \boldsymbol{w}_j}{\|\boldsymbol{w}_i\| \|\boldsymbol{w}_j\|} \tag{6.5}$$

Finally we consider a metric of *architectural* diversity. Take the *normalised* vector which represents each individual neural network, as described in Section 6.3.1. Let its size be L. To obtain an architectural representation, we simply remove the first element, J, from the normalised representation:

$$\mathbf{arch\_rep}_i = \mathbf{norm\_rep}_i^{\{1..L-1\}} \tag{6.6}$$

We then define the *architectural distance* between neural networks m_i and m_j as the cosine distance between their normalised architectural representations. It is defined thus:

$$\mathrm{arch\_dist}_{i,j} = 1 - \frac{\mathbf{arch\_rep}_i \cdot \mathbf{arch\_rep}_j}{\|\mathbf{arch\_rep}_i\| \|\mathbf{arch\_rep}_j\|} \tag{6.7}$$

6.4 Generic Novelty Search with Accuracy Objectives

We extend the NS procedure that we propose in Chapter 5, which constructs an ensemble of behaviourally diverse neural networks by evolving their architectures. We augment this method with accuracy objectives and make use of the same technique which employs a surrogate model for estimating novelty scores, as this reduces the computational burden of the *explicit* search for diversity. In addition, to preserve this greater efficiency, we deploy another surrogate model for *estimating* what the accuracy of a neural network architecture will be if it is trained with gradient descent on the input data. This surrogate accuracy model is pretrained on a sample of architectures drawn from the search space. We then propose multiple ways to incorporate explicit accuracy objectives into the NS.

6.4.1 Surrogate Models for Estimating Distance and Accuracy

A common technique in the optimisation and neural architecture search (NAS) literature (e.g. [THMY21, RLDL20, ZON$^+$06, GAM18, SZZ$^+$20]) is to use a surrogate modelling function to estimate certain parameters, rather than calculating exact values. In Chapter 5, we followed this trend by employing a surrogate model that produces estimates of the values of the diversity metrics of Section 6.3.2 between two neural network architectures. This reduces the computational burden of the search, as, in order to calculate exact values for these diversity metrics, the original NS procedure of Chapter 4 required exact error vectors to be determined by evaluating the architectures on a validation set. This involved a costly step of training all the neural networks in the population with gradient descent at every iteration. Following this approach, for each dataset we pretrain a Random Forest regressor [Bre01] to be the *surrogate diversity* model.

In addition, we also pretrain a *surrogate accuracy* model for each of the three datasets considered. This is so that accuracy objectives can be incorporated in the search for an ensemble of neural network architectures without having to train these architectures at every step of the procedure and calculate exact accuracy values on a validation set. Instead, estimates are produced for the expected accuracy given the normalised representation of each neural network individual, described in Section 6.3.1. The surrogate accuracy model is pretrained by first drawing a sample of 3200 neural networks from the search space of architectures, training them on the input data, and calculating their exact accuracy values on a validation set. We then construct a dataset on which we fit a Random Forest regressor. This model takes as input the normalised representation of a neural network architecture, as per Section 6.3.1, and outputs its estimated accuracy. The reason for selecting a Random Forest model lies in its low complexity and expected good generalisation, as it is an ensemble model. Algorithm 6.1 provides a pseudocode description of the procedures for training both the surrogate diversity and the surrogate accuracy models.

Algorithm 6.1 Pretraining the surrogate accuracy model on sample architectures

```
draw a sample $S$ of neural networks from the search space defined by $J$, $[C_{min}, C_{max}]$,
$[O_{min}, O_{max}]$, and $[D_{min}, D_{max}]$;
train($S$);        $\triangleright$ Models trained jointly or separately according to the value of $J_i$
for neural network model $m_i \in S$ do
    calculate accuracy $a_i$ on validation set $\mathcal{D}_{val}$;
end for
initialise dataset $\mathcal{D}_{accuracies} \leftarrow \emptyset$;
for each model $m_i$ do
    norm_rep$_i$ is the normalised representation of $m_i$;
    add data point $x \leftarrow \{$norm_rep$_i, a_i\}$ to $\mathcal{D}_{accuracies}$;
end for
train random forest model $rf$ on $\mathcal{D}_{accuracies}$;
return random forest model $rf$;
```

6.4.2 Novelty Search Extended with Accuracy Objectives

In this paper, we propose a number of *alternative* methods which extend the NS method of Chapter 5 by combining diversity and accuracy objectives in different ways across the two phases outlined in Figure 6.1. We refer to the generic search method described in Section 6.2, which is *instantiated* according to each of the selected search methods proposed herein and described below, namely by calculating *in different alternative ways* the scores described before: the fitness $score_i$, the archive score $arch\_score_i$, and the elite score el_score_i, all three during the *search phase*; and the ensemble score en_score_i, during the *ensemble selection phase.* These scores affect the way individuals are selected to both the search and the elite archives, whose purpose is described in Section 6.2, and to the final ensemble. A pseudocode description of this generic search method is given in Algorithm 6.2.

Combining Method 1 (CM1): Local Competition

Local Competition (LC) [LS11b] extends NS [LS08] by adding fitness objectives, which in our case are explicit accuracy objectives. It weights diversity and accuracy according to a parameter α. We expect variations of this parameter to produce different results. The distance between two models, div_metric(m_i, m_k), and the accuracy acc_i of model m_i are *estimated* by surrogate models as described before. In addition to the novelty score NS_i, a local competition score LC_i is calculated as the proportion of neighbours that a model outperforms:

$$NS_i = \frac{1}{k}\sum_{k=0}^{K} \text{div\_metric}(m_i, m_k) \tag{6.8}$$

$$LC_i = \frac{1}{k}\sum_{k=0}^{K} c(m_i, m_k) \tag{6.9}$$

For all K neighbours m_k of m_i. The diversity metric div_metric$_{i,j}$ is selected from the metrics defined in Section 6.3.2. Note that, while the paper which originally proposed Novelty Search with Local Competition (NSLC) [LS11b] defines LC_i as a count, we define it as a *proportion*

Algorithm 6.2 Generic search method for constructing a classifier ensemble

```
randomly initialise population $pop$;
$search\_archive \leftarrow \emptyset$;
$elite\_archive \leftarrow \emptyset$;
draw $\mathcal{D}_{train}$ and $\mathcal{D}_{val}$ from training set $\mathcal{D}$;
set evolution iterations $epochs$;
set search archive sample size $S_A$
set final ensemble size $S$;
surrogate diversity model $s_{div}$ pretrained as per Algorithm 5.1;
surrogate accuracy model $s_{acc}$ pretrained as per Algorithm 6.1;
select diversity $\text{div\_metric}_{i,j}$ from Section 6.3.2;
$score_i$ is the fitness score of model $m_i$;
$arch\_score_i$ is the archive score of model $m_i$;
$el\_score_i$ is the elite score of model $m_i$;
$en\_score_i$ is the ensemble score of candidate model $m_i$;
for $epochs$ do
    for $m_i, m_j \in pop \times pop \cup search\_archive : m_i \neq m_j$ do
        $\text{div\_metric}_{i,j} \approx s_{div}(m_i, m_j)$
    end for
    for $m_i \in pop$ do
        $acc_i \approx s_{acc}(m_i)$
    end for
    calculate $score_i$, $arch\_score_i$, and $el\_score_i$ according to the selected search method;
    $sa\_sample \leftarrow \text{sample}(pop, arch\_score_i, S_A)$
    $search\_archive \leftarrow search\_archive \cup sa\_sample$
    $el\_best \leftarrow \max(pop, el\_score_i)$
    $elite\_archive \leftarrow elite\_archive \cup \{el\_best\}$
    $s \leftarrow \text{tournament\_select}(pop, score_i)$
    $pop \leftarrow \text{mutate}(s)$
end for
for $m_i \in elite\_archive$ do
    calculate $en\_score_i$ according to the selected search method;
end for
$ensemble \leftarrow \max(elite\_archive, en\_score_i, S)$
train($ensemble$);
```

calculated w.r.t. the number of neighbours of m_i. This is to ensure that a single score can be appropriately calculated that mixes both LC_i and the novelty score NS_i, which will be of the same order of magnitude due to the fact that distances are scaled to lie between 0 and 1 when pretraining the surrogate diversity model. $c(m_i, m_k)$ is defined thus:

$$c(m_i, m_k) = \begin{cases} 1 & \text{if } acc_i > acc_k \\ 0 & \text{if otherwise} \end{cases} \tag{6.10}$$

The *fitness score* of model m_i is then calculated by mixing NS_i and LC_i according to the mixing parameter α:

$$score_i = (1 - \alpha) \times NS_i + \alpha \times LC_i \tag{6.11}$$

No archive score $arch\_score_i$ is calculated for LC since a *random sample*, of size S_A, of the individuals in the current population is added to the search archive, as in Chapter 5. Consider now a novelty score for model m_i calculated w.r.t. *all the individuals in the elite archive*, NS_i^*. Consider the equivalent local competition score, LC_i^*. The elite score el_score_i is calculated in a similar fashion to $score_i$, with the same parameter α but using these two scores instead. The individual in each generation with the highest el_score_i is added to the elite archive. At the end of the procedure, an ensemble score en_score_i is calculated in a very similar way for all the neural network models in the elite archive — w.r.t. all other individuals in this archive. The top S individuals with the highest ensemble scores will make up the final ensemble.

Combining Method 2 (CM2): Search for Diversity with Accuracy in Archives

This variant uses a novelty score to guide the search procedure, namely the selection of individuals from the population for reproduction, while storing the neural network models in the archives, including the search archive, according to their accuracy. Thus, we expect this method to maintain diverse populations whilst selecting the most accurate models in an elitist fashion. The scores are defined in the following way:

$$score_i = NS_i \tag{6.12}$$

$$arch\_score_i = el\_score_i = en\_score_i = acc_i \tag{6.13}$$

While the single individual with highest el_score_i is added to the elite archive at every step of the search, the top S_A individuals with highest $arch\_score_i$ are added to the search archive.

Combining Method 3 (CM3): Search for Accuracy with Diversity in Archives

This variant does the opposite of the previous combining method, i.e. it uses accuracy to guide the search procedure, but stores individuals in the elite archive based on how novel/diverse they are. The final ensemble is also selected based on novelty. We thus expect it to maintain accurate populations and the final ensemble to be a set of diverse models selected from a high-performing region of the search space. Recall that NS_i^* is the novelty score calculated for model m_i in the current generation w.r.t. all individuals in the elite archive. Let NS_i^{**} be a similar novelty score calculated for model m_i in the elite archive w.r.t. all other models in this archive. The scores are then defined as:

$$score_i = acc_i \tag{6.14}$$

$$el\_score_i = NS_i^* \tag{6.15}$$

$$en\_score_i = NS_i^{**} \tag{6.16}$$

This method does not keep a search archive as the search is guided by accuracy only and, therefore, there is no need to keep an archive of past solutions with respect to which a novelty score is calculated.

Explicit Accuracy with Implicit Diversity

The last method we consider uses only an implicit definition of diversity. The search is guided by accuracy and individuals are stored in the elite archive and selected for the final ensemble also based exclusively on their estimated accuracy. Diversity is generated implicitly by the evolutionary procedure, namely the mutation operator applied in the reproduction step. We expect this method to produce accurate but not very diverse ensembles. The scores are then determined as:

$$score_i = el\_score_i = en\_score_i = acc_i \tag{6.17}$$

As in the case of the previous method, no search archive is kept, since the purpose of such an archive is for novelty scores to be calculated w.r.t. past solutions.

6.5 Experiments

In this section, we describe the experiments carried out on three datasets — CIFAR-10, CIFAR-100, and SVHN. We compare the results reported in Chapter 5 with the four modified methods that we present in Section 6.4.2. Running each of these methods to construct an ensemble whose performance is then evaluated requires four steps: (1) running the modified NS procedure using the surrogate diversity and surrogate accuracy models, *without training* the neural network architectures during the search; (2) training the ensemble of neural network architectures resulting from the previous step on the training set $\mathcal{D}_{train}$, using a standard stochastic gradient descent (SGD) procedure; (3) training a stacking model [Wol92] on the validation set $\mathcal{D}_{val}$, so as to learn a weighted average of the predictions made by each member of the ensemble; and (4) calculating the classification accuracy of the ensemble on a test set. The surrogate diversity and surrogate accuracy models are pretrained as described before. Table 6.1 shows

Table 6.1: Common parameters fixed throughout the experiments for all the NS methods extended with accuracy objectives

Parameter	**Value**
Iterations	100
Final ensemble size S	40
Population size	100
Number of residual blocks	2:6
Number of channels in the first convolution	4:16
Number of channels in residual blocks	16:64
Dropout probability in residual blocks	0.1:0.9
Number of neighbours K	15
Size n_A of archive sample	10
Size of tournament for selection	50

the parameters used throughout the experiments. Each experiment, i.e. each sequence of the steps (1)-(4) described above, is run 10 times in order to ensure statistical significance in our observations. The following four hypotheses are tested as part of our empirical analysis.

Hypothesis 6.1 (Performance Gain by Adding Accuracy Objectives)**.** *Taking individual model accuracy into account as an objective, by means of the methods presented in Section 6.4.2, leads to better ensemble accuracy than what can be achieved with a plain NS method.*

This hypothesis expresses the expectation that accuracy objectives can improve the results of a plain search for explicit diversity alone. We test it by comparing the results achieved by the NS method of Chapter 5 with the final ensemble accuracy resulting from the three methods from Section 6.4.2 which combine diversity and accuracy objectives: local competition, search for diversity with accuracy in the archives, and search for accuracy with diversity in the archives.

Hypothesis 6.2 (Different Performance with Different Combinations of Diversity and Accuracy)**.** *Selecting different combinations of diversity and individual model accuracy, along the spectrum that ranges from favouring only diversity to favouring only accuracy, results in different ensemble accuracy.*

This hypothesis expresses the notion that multiple ways of mixing diversity and accuracy objectives lead to different ensemble accuracy and that, therefore, an optimal middle ground between searching only for one or the other can be found. We test it by comparing the ensemble performance resulting from varying the mixing weight α when deploying the LC approach (see Equation 6.11), as well as from the other methods of Section 6.4.2.

Hypothesis 6.3 (Diversity and Accuracy Must Be Balanced)**.** *Assigning greater importance to diversity in an ensemble leads to worse individual model accuracy. Conversely, weighting individual accuracy more leads to less diverse ensembles. There is a trade-off to be found between the two.*

This hypothesis claims that there is a fundamental tension between diversity and individual model accuracy and that one can only be improved at the expense of the other. We test it by calculating the values of diversity metrics and average individual model accuracy for the ensembles resulting from applying both the previous NS method and the various methods of combining diversity and accuracy objectives.

Hypothesis 6.4 (Worse Performance Without Explicit Diversity)**.** *Removing explicit diversity objectives, keeping only individual accuracy objectives when searching for an ensemble, leads to a decrease in ensemble diversity and, consequently, worse ensemble accuracy.*

This hypothesis expresses the importance of explicit diversity objectives for constructing a high-performing classifier ensemble. This follows from the results of Chapters 4 and 5. We test it by comparing the performance resulting from the last method presented in Section 6.4.2, which considers only accuracy objectives with diversity being generated implicitly, with that resulting from the previous NS approach and the methods which combine both diversity and accuracy objectives.

6.6 Results and Discussion

This section presents the results of the experimental work that we have carried out and conducts an empirical analysis by discussing those results in connection with the hypotheses laid out in Section 6.5. Table 6.2 shows the results of running each of the methods proposed in

Section 6.4.2, as well as the NS method of Chapter 5. This is the mean final ensemble accuracy over 10 runs for each method, parameter setting, and diversity metric, as applicable. As observed in Chapter 5, the metrics that lead to the best results are $\text{prop}^2_{i,j}$ and $\text{cos\_dist}_{i,j}$. While for LC we run the method with all diversity metrics for a direct comparison with the previous NS approach, for the other two combining methods we confine ourselves to these two metrics in the interests of clarity. The method that only utilises explicit accuracy objectives does not make use of any diversity metric as diversity is generated implicitly.

6.6.1 Hypothesis 6.1

Hypothesis (Performance Gain by Adding Accuracy Objectives)**.** *Taking individual model accuracy into account as an objective, by means of the methods presented in Section 6.4.2, leads to better ensemble accuracy than what can be achieved with a plain NS method.*

Table 6.2 shows accuracy results both for the NS method of Chapter 5 and the four methods we propose herein. The cells highlighted in red correspond to results which are significantly better at the 1% level, determined by Mann-Whitney statistical significance tests over 10 runs, than the NS method for the respective diversity metric (not applicable to the last method). We can see that both LC and the other two combining methods of Section 6.4.2 improve on the NS for some of the metrics considered, but that the only consistent improvement on all three datasets is observed for LC with metrics $\text{prop}^1_{i,j}$ and $\text{arch\_dist}_{i,j}$, which are the ones that tend to perform the worst in the NS. We note that, on CIFAR-10, the second combining method produces statistically significant improvements over NS, but that these improvements are not only too small to be considered relevant, but also inconsistent as they are not observed on CIFAR-100 or on SVHN. A similar inconsistent improvement is observed on CIFAR-100 for the third method, with metric $\text{prop}^2_{i,j}$. These results therefore only partially support Hypothesis 6.1, since introducing accuracy objectives only improves on the results of the NS for the worst-performing diversity metrics. The hypothesis must be rejected since no significant improvement is observed for the best-performing ones, which suggests that the choice of a good diversity metric plays

Table 6.2: Median accuracy over 10 runs for the previous NS method and all the methods extending it with accuracy objectives. Results significantly better than NS at the 1% level for the respective diversity metric highlighted in red

Method		**Diversity Metric**	**Accuracy CIFAR-10**	**Accuracy CIFAR-100**	**Accuracy SVHN**
NS (from Chapter 5)		$\text{prop}^1_{i,j}$	67.295%	28.725%	78.825%
		$\text{prop}^2_{i,j}$	90.605%	63.05%	94.8%
		$\text{prop}^{\text{harm}}_{i,j}$	83.975%	63.41%	89.775%
		$\text{dis}_{i,j}$	86.28%	63.18%	90.675%
		$\text{cos\_dist}_{i,j}$	90.11%	63.035%	94.79%
		$\text{arch\_dist}_{i,j}$	80.4%	49.83%	90.68%
CM1: LC	$\alpha = 0.1$	$\text{prop}^1_{i,j}$	80.485%	34.605%	89.975%
		$\text{prop}^2_{i,j}$	90.655%	63.46%	94.98%
		$\text{prop}^{\text{harm}}_{i,j}$	86.12%	63.935%	91.635%
		$\text{dis}_{i,j}$	84.615%	63.63%	91.63%
		$\text{cos\_dist}_{i,j}$	90.26%	63.415%	94.87%
		$\text{arch\_dist}_{i,j}$	87.745%	53.865%	93.27%
	$\alpha = 0.5$	$\text{prop}^1_{i,j}$	90.67%	63.45%	94.565%
		$\text{prop}^2_{i,j}$	90.715%	63.18%	94.87%
		$\text{prop}^{\text{harm}}_{i,j}$	86%	63.755%	92.67%
		$\text{dis}_{i,j}$	86.62%	63.99%	93.76%
		$\text{cos\_dist}_{i,j}$	90.005%	63.685%	94.925%
		$\text{arch\_dist}_{i,j}$	88.635%	58.615%	94.25%
	$\alpha = 0.9$	$\text{prop}^1_{i,j}$	90.295%	63.695%	94.82%
		$\text{prop}^2_{i,j}$	90.735%	63.47%	94.99%
		$\text{prop}^{\text{harm}}_{i,j}$	85.31%	63.755%	93.78%
		$\text{dis}_{i,j}$	86.145%	63.835%	92.23%
		$\text{cos\_dist}_{i,j}$	89.9%	63.265%	94.955%
		$\text{arch\_dist}_{i,j}$	88.635%	59.045%	94.455%
CM2: Search for Div., Acc. in Archives		$\text{prop}^2_{i,j}$	90.83%	63.125%	94.915%
		$\text{cos\_dist}_{i,j}$	90.83%	62.545%	94.91%
CM3: Search for Acc., Div. in Archives		$\text{prop}^2_{i,j}$	90.73%	64.16%	94.865%
		$\text{cos\_dist}_{i,j}$	90.565%	63.93%	94.89%
Explicit Accuracy with Implicit Diversity			90.895%	63.755%	94.915%

a more important role and can make a more considerable difference than explicit accuracy objectives.

6.6.2 Hypothesis 6.2

Hypothesis (Different Performance with Different Combinations of Diversity and Accuracy)**.** *Selecting different combinations of diversity and individual model accuracy, along the spectrum that ranges from favouring only diversity to favouring only accuracy, results in different ensemble accuracy.*

As observed above, introducing accuracy objectives only considerably improves on the NS results for the two worst-performing metrics. To analyse the influence of different combinations of diversity and accuracy objectives, we now focus more closely on the results achieved by LC, the other two combining methods, and the explicit accuracy search method. For LC, we see that there is an improvement for the two worst-performing metrics, $\text{prop}^1_{i,j}$ and $\text{arch\_dist}_{i,j}$, when increasing α from 0.1 to 0.5 or 0.9. Mann-Whitney tests confirm that this improvement is indeed statistically significant: for both metrics on CIFAR-100 and SVHN; and for $\text{prop}^1_{i,j}$ on CIFAR-10. However, no statistically significant difference is observed for any of the other metrics.

If we look at the second combining method of Section 6.4.2, we observe a slight improvement, which is nonetheless statistically significant, on CIFAR-10 over LC for the metric $\text{cos\_dist}_{i,j}$, but this is not observed on CIFAR-100 or SVHN and, therefore, not a consistent result. And finally, if we look at the final method we propose, the explicit accuracy search with implicit diversity, we again see that, although improving on the worst metrics, there is no significant improvement observed consistently on all three datasets over NS or any of the combining methods for the two best diversity metrics, $\text{prop}^2_{i,j}$ and $\text{cos\_dist}_{i,j}$. Statistical significance is observed in particular cases — e.g. on CIFAR-10 and CIFAR-100 for NS with the metric $\text{cos\_dist}_{i,j}$ or on CIFAR-10 for LC with that same metric — but in any case the improvements in accuracy are very small. We therefore reject Hypothesis 6.2 since the results do not consistently back the claim that different combinations of diversity and accuracy objectives lead to significantly different ensemble accuracy. As observed before, the choice of diversity metric seems to play a more crucial role than the choice of a particular combination between diversity and accuracy.

6.6.3 Hypothesis 6.3

Hypothesis (Diversity and Accuracy Must Be Balanced)**.** *Assigning greater importance to diversity in an ensemble leads to worse individual model accuracy. Conversely, weighting individual accuracy more leads to less diverse ensembles. There is a trade-off to be found between the two.*

Table 6.3 shows the average individual accuracy and the values of distance metrics for the final ensemble, measured on the test data for each of the datasets considered, with different methods and parameter settings. In the interests of clarity and due to limitations of space we only include results for some of the methods and diversity metrics, since other results do not contribute with any additional insight. The values of different diversity metrics are scaled *for the same dataset* so that the magnitude of variations across rows may be directly compared, hence the negative values. We have utilised the same min-max scalers that are fitted on the training data when pretraining the surrogate diversity model for each dataset.

The first thing we note is the clear correspondence between similar average individual accuracy and similar values for each diversity metric. The more similar the accuracy values, the more similar the diversity values. This is observed across different methods on all three datasets. For example, if we compare the rows for the NS method with metrics $\text{prop}^2_{i,j}$ and $\text{cos\_dist}_{i,j}$, we find no statistically significant difference in average individual accuracy or the values of diversity metrics, measured on the test set. If we take a closer look on the results of both the NS and LC, we see that for the metric $\text{prop}^1_{i,j}$, one of the worst-performing ones as discussed previously, increasing the weight α of the local competition score LC_i (Equation 6.11) leads to a clear increase in the average individual accuracy of the models in the ensemble and a decrease in the observed value for this metric w.r.t. the test set. This is observed consistently on all three datasets and Mann-Whitney tests confirm that this difference is statistically significant at the 1% level between rows corresponding to different values of α and w.r.t. to plain NS. However, the exact opposite is observed for the rows corresponding to the metric $\text{cos\_dist}_{i,j}$, with the measured individual accuracy and diversity values remaining approximately constant for different values of α and even for the method that only uses accuracy objectives; statistically significant differences are not observed. This counter-example allows us to reject Hypothesis 6.3 since it is not the case in general that weighting diversity more will lead to worse individual accuracy or vice-versa, which is a surprising result. The observations suggest this is highly dependent on the choice of a diversity metric, rather than being a general rule. *Running the methods with a high-performing diversity metric does not seem to require a trade-off with individual model accuracy.*

6.6.4 Hypothesis 6.4

Hypothesis (Worse Performance Without Explicit Diversity). *Removing explicit diversity objectives, keeping only individual accuracy objectives when searching for an ensemble, leads to a decrease in ensemble diversity and, consequently, worse ensemble accuracy.*

Looking at Table 6.2, we can see that the last method, which searches only for explicit accuracy, with implicit diversity being generated by the evolutionary procedure, does not do worse than the best amongst the other methods. In fact, our analysis reveals that in some cases it achieves better accuracy than some of these other methods in a statistically significant way, although this is not observed consistently — i.e. for all methods and diversity metrics across all three datasets — and at any rate the differences are small. This means that Hypothesis 6.4 must be rejected, as removing explicit diversity objectives does not lead to worse ensemble accuracy. This is a surprising result given the findings of Chapters 4 and 5, regarding the explicit search for diversity when compared to common methods that only promote it implicitly.

Table 6.3: Average individual accuracy and diversity metrics for the final ensemble after training, calculated on test sets. Median values over 10 runs. Distance values scaled to facilitate comparison

	Method		**Diversity Metric**	**Avg. Ind. Acc.**	**prop^1**	**prop^2**	**prop$^{\text{harm}}$**	**dis**	**cos_dist**	**arch_dist**
CIFAR-10	NS		prop$^1_{i,j}$	19.53%	0.895	0.203	0.309	0.234	0.132	0.015
			prop$^2_{i,j}$	85.51%	-0.195	0.569	-0.119	-0.075	0.437	0.018
			cos_dist$_{i,j}$	85.50%	-0.195	0.571	-0.120	-0.076	0.439	0.028
	CM1: LC	$\alpha = 0.1$	prop$^1_{i,j}$	23.78%	0.872	0.283	0.396	0.312	0.190	0.115
			cos_dist$_{i,j}$	85.89%	-0.201	0.570	-0.129	-0.081	0.438	0.010
		$\alpha = 0.5$	prop$^1_{i,j}$	84.35%	-0.173	0.580	-0.085	-0.054	0.447	0.038
			cos_dist$_{i,j}$	85.41%	-0.196	0.567	-0.122	-0.077	0.435	0.023
		$\alpha = 0.9$	prop$^1_{i,j}$	85.05%	-0.191	0.563	-0.113	-0.072	0.431	0.054
			cos_dist$_{i,j}$	85.59%	-0.198	0.569	-0.123	-0.078	0.436	0.022
	Explicit Acc., Implicit Div.			85.87%	-0.199	0.574	-0.126	-0.079	0.442	0.025
CIFAR-100	NS		prop$^1_{i,j}$	4.85%	0.923	0.086	0.159	0.113	0.059	0.027
			prop$^2_{i,j}$	53.75%	-0.806	0.536	0.474	0.378	0.434	0.040
			cos_dist$_{i,j}$	53.76%	-0.801	0.538	0.473	0.377	0.435	0.029
	CM1: LC	$\alpha = 0.1$	prop$^1_{i,j}$	7.07%	0.892	0.143	0.245	0.186	0.100	0.073
			cos_dist$_{i,j}$	54.41%	-0.812	0.544	0.476	0.380	0.442	0.047
		$\alpha = 0.5$	prop$^1_{i,j}$	52.61%	-0.730	0.552	0.497	0.402	0.448	0.045
			cos_dist$_{i,j}$	54.52%	-0.810	0.544	0.476	0.380	0.442	0.050
		$\alpha = 0.9$	prop$^1_{i,j}$	54.88%	-0.816	0.549	0.477	0.380	0.447	0.044
			cos_dist$_{i,j}$	53.81%	-0.809	0.537	0.474	0.378	0.435	0.033
	Explicit Acc., Implicit Div.			54.68%	-0.818	0.544	0.475	0.379	0.441	0.041
SVHN	NS		prop$^1_{i,j}$	16.23%	0.851	0.126	0.144	0.138	0.087	0.055
			prop$^2_{i,j}$	92.10%	-0.173	0.521	-0.292	-0.171	0.386	0.036
			cos_dist$_{i,j}$	92.07%	-0.172	0.520	-0.291	-0.171	0.385	0.039
	CM1: LC	$\alpha = 0.1$	prop$^1_{i,j}$	23.15%	0.867	0.266	0.307	0.274	0.187	0.134
			cos_dist$_{i,j}$	92.24%	-0.174	0.525	-0.294	-0.172	0.390	0.029
		$\alpha = 0.5$	prop$^1_{i,j}$	86.02%	-0.017	0.642	-0.057	-0.002	0.495	0.081
			cos_dist$_{i,j}$	92.39%	-0.177	0.516	-0.300	-0.175	0.381	0.020
		$\alpha = 0.9$	prop$^1_{i,j}$	92.23%	-0.175	0.514	-0.297	-0.174	0.379	0.062
			cos_dist$_{i,j}$	92.40%	-0.176	0.519	-0.298	-0.174	0.384	0.027
	Explicit Acc., Implicit Div.			92.49%	-0.178	0.516	-0.302	-0.176	0.381	0.025

6.6.5 The Diversity-Accuracy Duality

For an explanation of these surprising results, we now turn again to Table 6.3 and focus on the last row for each dataset, corresponding to this explicit accuracy search method. We have already noted that a trade-off between accuracy and diversity is not required when the best metrics are utilised. We can also see that, for the last method, the values for the individual accuracy and each of the diversity metrics are very similar to those in the rows corresponding to the best diversity metrics, $\text{prop}^2_{i,j}$ and $\text{cos\_dist}_{i,j}$. The difference between the values for this explicit accuracy search and the rows corresponding to the worst metric, $\text{prop}^1_{i,j}$, is naturally statistically significant in most cases, with some exceptions observed for LC with $\alpha = 0.9$. For the other cases, a statistically significant difference is at times observed, as this method tends to result in slightly higher average individual accuracy, probably a result of *only* favouring accuracy during the search. However, these differences are very small and the key observation is that the average individual accuracy and the diversity values are very similar when comparing this method with all the other ones using the two best metrics, including the NS, *which only favours diversity* explicitly. These results suggest that, contingent on the choice of a high-performing diversity metric — in this case, $\text{prop}^2_{i,j}$ or $\text{cos\_dist}_{i,j}$ — there is an *equivalence* between searching for diversity and searching for accuracy. Regardless of the importance assigned to each of these two properties, the resulting ensembles will have similar average individual accuracy and diversity and it is for this reason that their accuracy on test data is similar. We therefore hypothesise that, for these two diversity metrics, there is an *accuracy-diversity duality*, in the sense that these two properties appear to be interchangeable by means of an underlying process which is not yet understood, but which our methods nevertheless seem to approximate. This is highly significant because it challenges notions in the literature about the need to find a trade-off between diversity and accuracy in ensemble learning.

6.7 Summary

This chapter described an extension of a previous NS method with the goal of incorporating accuracy objectives when searching for behaviourally diverse ensembles. Our initial research question was whether these accuracy objectives could lead to a performance gain in terms of final ensemble accuracy. We investigated a range of search methods that span the full spectrum of favouring only accuracy, only diversity, or different combinations of both. We found that accuracy objectives lead to significant improvements in ensemble accuracy, but only for the worst-performing diversity metrics. For the best metrics, performance was not improved upon regardless of the importance/weight assigned to accuracy objectives. But the most surprising result was the observation that there is an equivalence between searching for diversity — when defined by the two best metrics, $\text{prop}^2_{i,j}$ and $\text{cos\_dist}_{i,j}$ — and searching for accuracy, with multiple ways of combining these two objectives leading to ensembles of similar diversity and average individual accuracy. When we considered the highest-performing metrics, there was no dichotomy between diversity and accuracy; each contributed to ensemble performance without detriment to the other and weighting one more did not impact negatively upon the other.

The observed equivalence between utilising diversity or accuracy objectives potentially means that the two are interchangeable and correlated in some conditions. This is a rather counter-intuitive result which suggests the existence of a diversity-accuracy duality in ensembles of classifiers. While further investigation of this equivalence is required so that stronger conclusions may be drawn, this result is significant because it challenges widespread assumptions about the need to trade off diversity for accuracy. An implication of this is the possibility of designing better algorithms which evolve diverse ensembles *without detriment to their accuracy*, since it is implicitly ensured.

Chapter 7

Conclusion

7.1 Summary

In this research programme, we sought to develop innovative methods that create explicit diversity in ensembles of classifiers, rather than relying on techniques that only implicitly promote it. There is a consensus on the point that diversity between individual members of an ensemble leads to better performance, but no standard and consistent "rulebook" for defining and exploiting diversity to construct a high-performing classifier. By leveraging diversity as the *criterion* for both instantiating and distributing the base learners of a classifier ensemble, we aimed to use the techniques described in this thesis to develop alternative methods to current deep learning (DL) techniques which are more sustainable, less resource-intensive, and more accessible to researchers and practitioners.

We began by describing an algorithm called WILDA, which builds an *architecturally* diverse ensemble for solving a classification task. In a distributed fashion, this algorithm evolves a set of neural networks that are pretrained on the target task and diverse w.r.t. architectural feature descriptors. WILDA implements a quality diversity (QD) algorithm called MAP-Elites to evolve this set of base learners, each of which extracts a feature vector from the data. These feature vectors are then aggregated and used as input to train a shallow centralised model for classification. This was our first attempt at a method which promotes diversity *systematically*

— even though this algorithm only rewards it, rather than actively searching for it.

In an effort to generalise this notion of architectural diversity and extend upon it, we then described a NS approach to evolving *behaviourally* diverse ensembles, proposing a number of *error* diversity metrics. The NS uses these metrics to guide a search over a large space of neural network architectures, *without* selecting them based on their accuracy, which is only implicitly ensured by a gradient descent procedure. Interestingly, we found that the error diversity metrics we proposed lead to better results than others commonly found in the literature, i.e. they push the NS towards better areas of the search space. However, the application in practice of this method was hampered by limitations in the amount of available computational resources, since it involves a very time-consuming step of training all networks in each generation of the NS with gradient descent.

We then presented an extension to this NS method which overcomes this limitation by using a pretrained surrogate model to estimate the distance between neural network architectures, necessary to calculate novelty scores, without the need to train them. In this way, we obtained an approximate speedup of 10 times w.r.t. the previous method when running them both with the same parameters, *without loss of classification accuracy.* We were able to construct better-performing ensembles thanks to the expanded architecture search space facilitated by using a surrogate. We also confirmed the above-mentioned observations regarding the error diversity metrics that we proposed. This new method thus represented an improved paradigm for implementing horizontal scaling of learning algorithms, making an explicit search for diversity considerably more tractable than the previous approach *for the same bounded resources.*

In order to study the relationship between diversity and accuracy in classifier ensembles, we then proposed several methods that extend the novelty search with accuracy objectives. Following widespread assumptions, we were expecting that a trade-off must be found by balancing diversity and accuracy objectives. Surprisingly, however, we observed that, for the highest-performing diversity metrics, there is an *equivalence* between searching for diversity objectives and searching for accuracy objectives. We therefore posit the existence of a *diversity-accuracy duality* in ensembles of classifiers, although more investigation is required to characterise this duality. An implication of this is the possibility of designing better algorithms which evolve diverse ensembles without detriment to their accuracy, since it is implicitly ensured.

7.2 Limitations

We identify ample opportunity for improving our work. The main limitation of our results is the fact that the neural network architectures evolved by our procedures are much less complex than state-of-the-art approaches. While it was not our aim to develop solutions capable of outperforming the state of the art, it was mainly due to limitations of the available hardware that we have not considered more complex architectures which could have led to a performance potentially capable of challenging the state of the art. In general, the architectures we have employed are quite rigid sequences of residual blocks, with only a few parameters being allowed to vary. This has made it hard to explore other solutions which could have led to better performance.

Another limitation is that, while we have proposed methods that explicitly search for diversity, we do not have a systematic way of *defining* diversity beforehand. The definitions we present have either been taken from the literature or specified by us in an *ad hoc* fashion, albeit with reasonable justifications for their use. While our results provide some insight into what makes a useful diversity metric, namely that metrics which focus on the errors made by the models lead to better-performing ensembles, this relationship is not yet completely clear and has precluded us from conducting a more in-depth study into how to exploit appropriate definitions of diversity.

7.3 Future Work

Future work will focus on new research questions arising from the results of this research programme, in addition to addressing the limitations pointed out in Section 7.2. We will improve on our current methods by making them more flexible w.r.t. the architectures that can be evolved. While the complexity of these architectures will remain bounded by the limitations in hardware and compute power available to us, widening the search space of candidate neural network architectures could prove beneficial. This includes going beyond convolutional neural network (CNN) and residual neural network (ResNet) architectures, considering new design paradigms. We would also like to develop a systematic solution to the very problem of defining diversity by evolving diversity metrics *automatically*, using simple operations as building blocks.

The surprising result described in Chapter 6, regarding the observed equivalence between searching for diversity or searching for accuracy, deserves a more in-depth study. In future work, we aim to provide more insight into this equivalence, which will enable a better characterisation of the diversity-accuracy duality. Going beyond the empirical analysis provided in this thesis, we will investigate what underlying process exists which could explain the equivalence between accuracy objectives and the two highest-performing diversity metrics. Ultimately, we seek to leverage this insight into the relationship between diversity and accuracy in ensembles of classifiers to design better evolutionary search methods capable of constructing simultaneously diverse and high-performing ensembles, which can be used as the basis for horizontal scaling and distribution of neural network models.

7.4 Final Remarks

The work carried out during the course of this research programme has resulted in a number of publications. In the beginning of this PhD programme, we produced papers whose content is outside the scope of this thesis, namely work on a multi-agent simulation of a community energy system published in ISoLA 2018 [CRH$^+$18], eCAS 2018 [PCHO19], and GECCO 2019 [CHP19]. The work on architectural diversity of Chapter 3 resulted in publications in GECCO 2020 [CHP20] and EvoStar 2021 [CHKP21a]. The work that resulted in the NS algorithm of Chapter 4 was published in GECCO 2021 [CHKP21b]. The extension of this method with a surrogate model, described in Chapter 5, resulted in a paper in EvoStar 2022 [CHKP22], which was awarded best paper in the Evolutionary Machine Learning (EML) track. Finally, a paper sharing the results of Chapter 6 has been accepted at GECCO 2022.

I am pleased with the fact that this thesis contributes an unexpected result — the diversity-accuracy duality — which emerged from our work on designing algorithms that construct diverse ensembles of classifiers systematically. Far from being the end point, this result opens up the possibility of many other research avenues. I have taken on a position as Research Assistant in the Computing department at Imperial College and I am glad that the work I carried out during this research programme has proved useful. I am certain that the experience of the last four years will remain very relevant.

Appendix A

Additional Figures

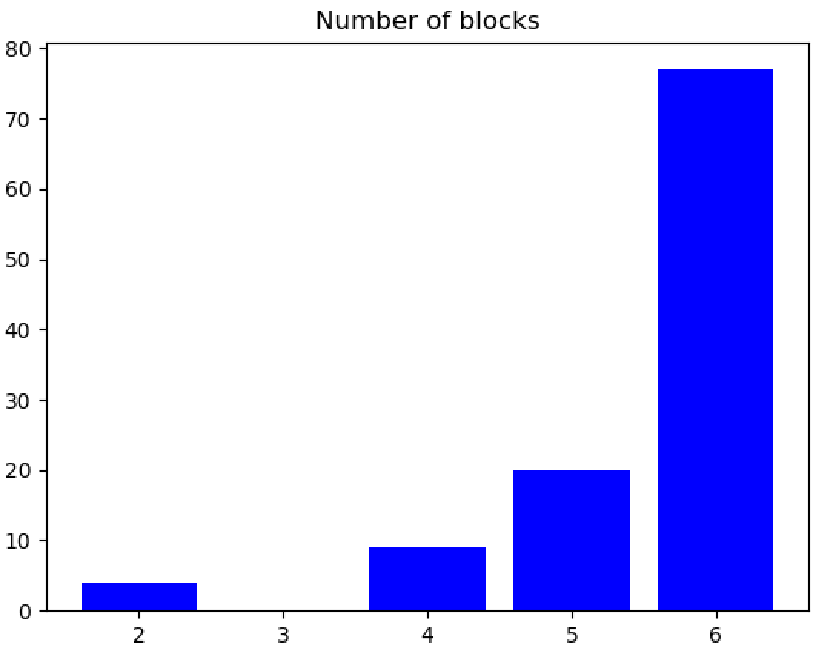


Figure A.1: Distribution of number of blocks in the final ensemble for the original NS method (Chapter 4)

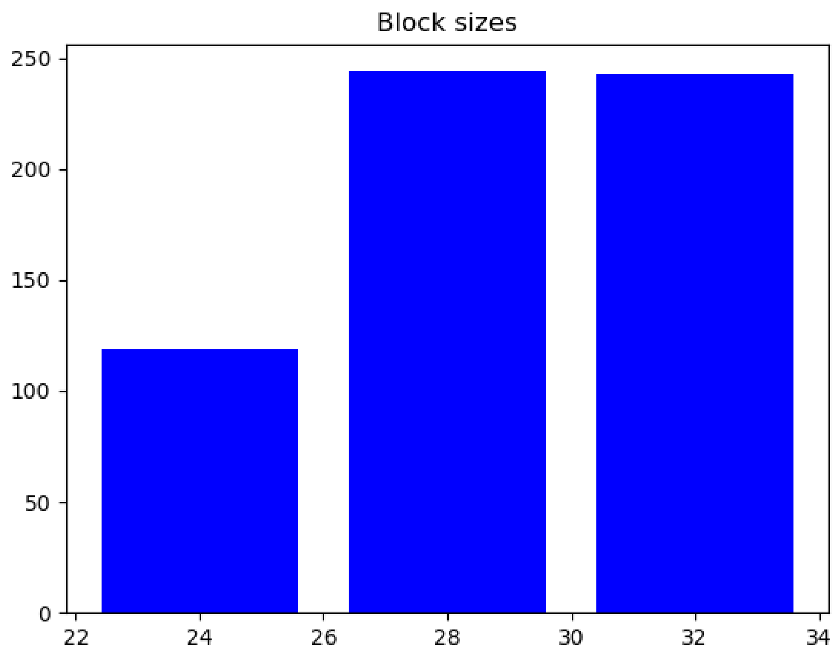


Figure A.2: Distribution of sizes of the residual blocks in the final ensemble for the original NS method (Chapter 4)

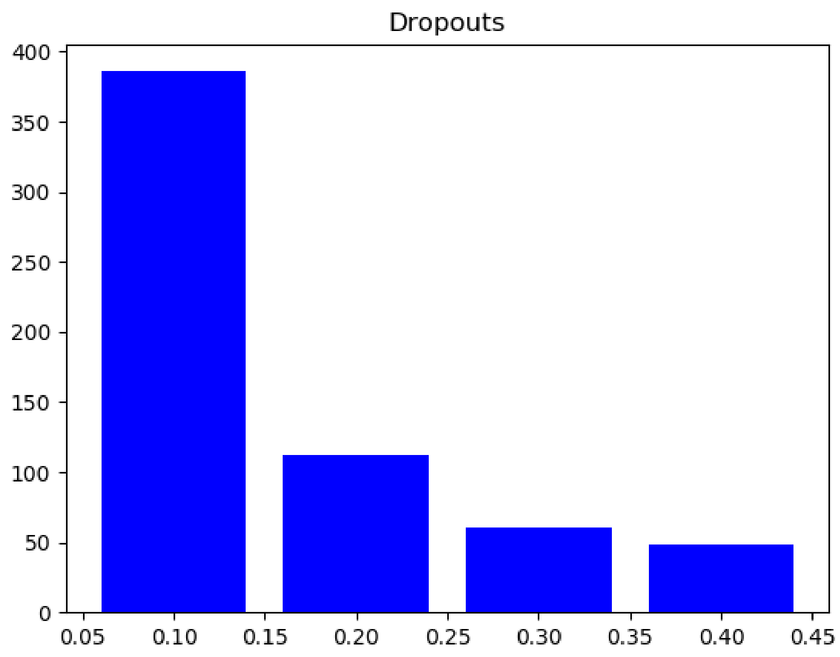


Figure A.3: Distribution of dropout probabilities in the final ensemble for the original NS method (Chapter 4)

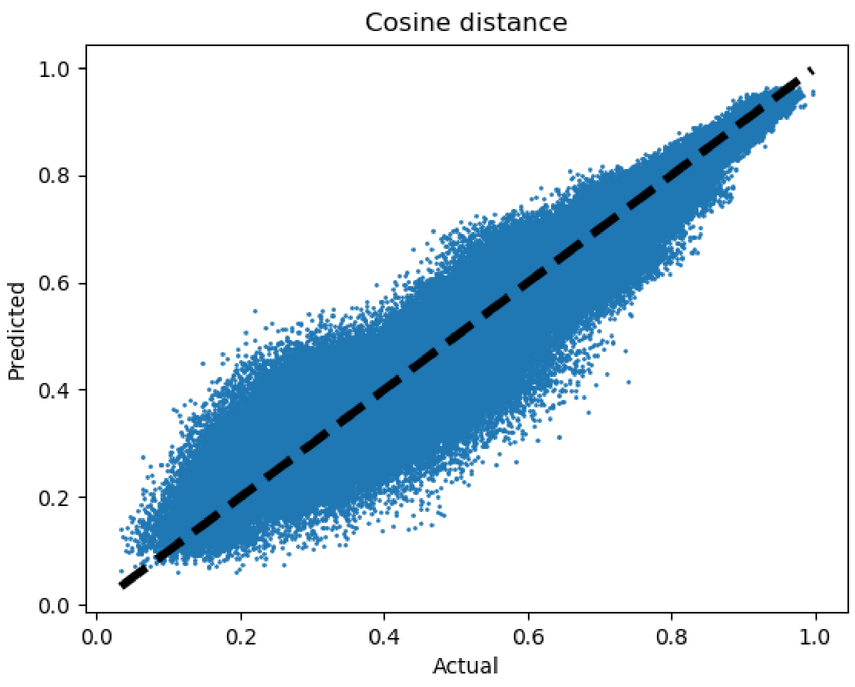


Figure A.4: Performance of surrogate model for predicting the cosine distance on CIFAR-10 (Chapter 5)

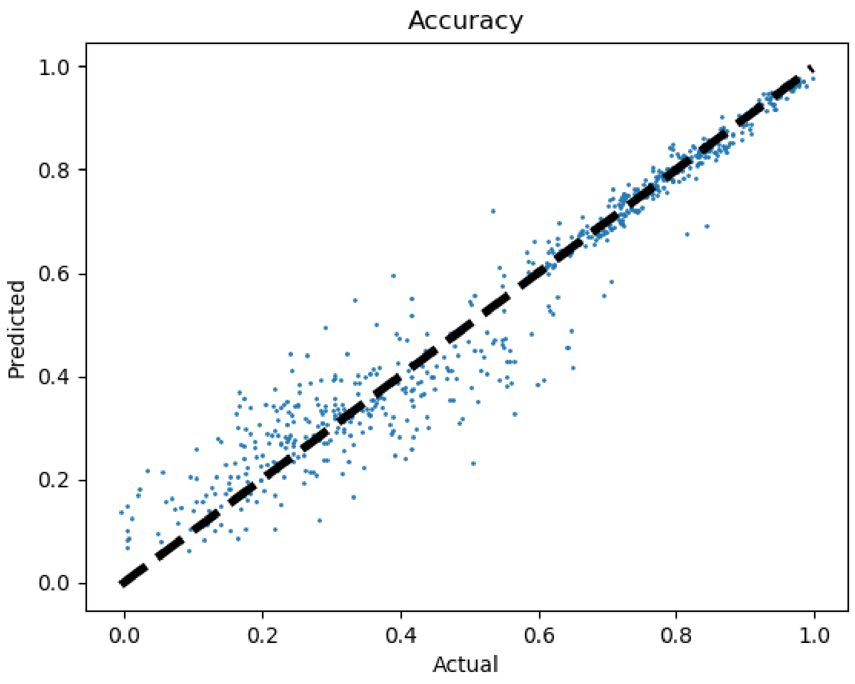


Figure A.5: Performance of surrogate model for predicting classification accuracy on CIFAR-10 (Chapter 6)

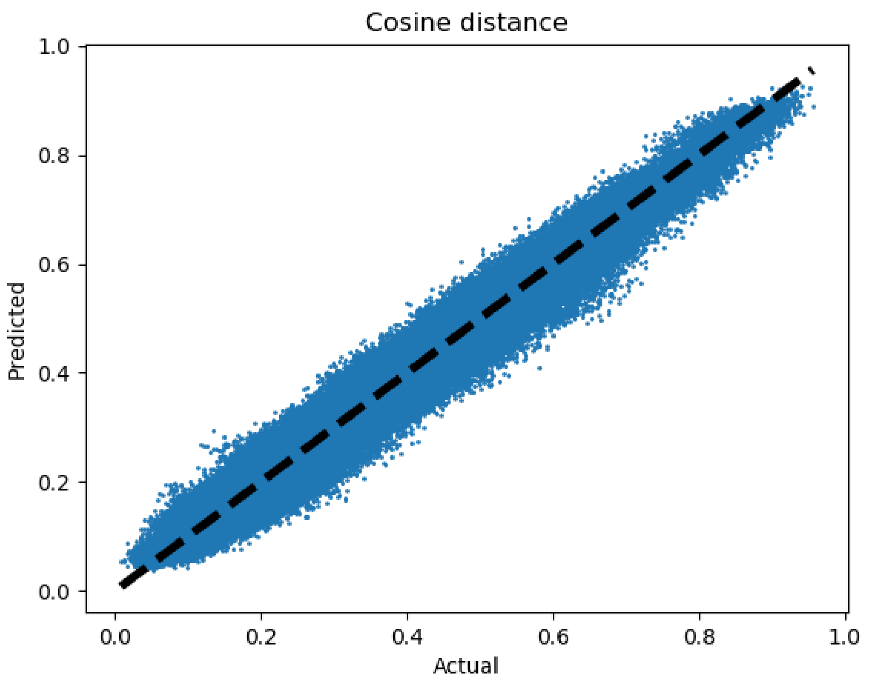


Figure A.6: Performance of surrogate model for predicting the cosine distance on CIFAR-100 (Chapter 5)

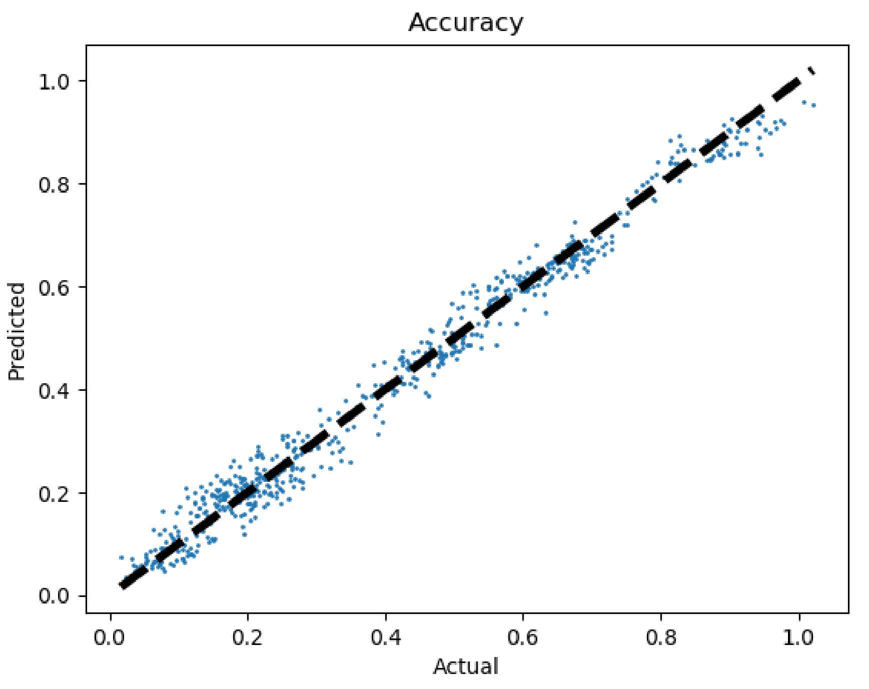


Figure A.7: Performance of surrogate model for predicting classification accuracy on CIFAR-100 (Chapter 6)

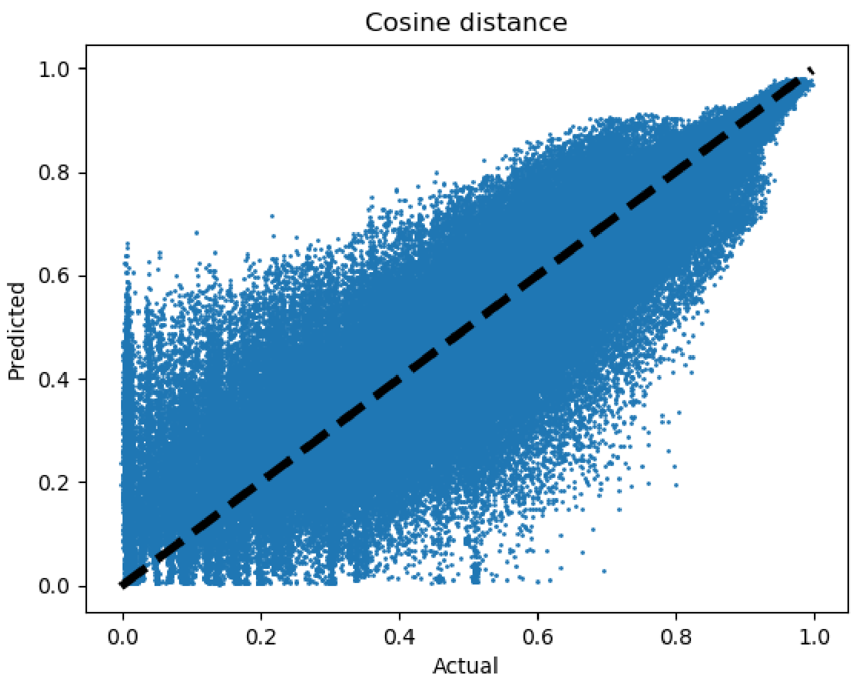


Figure A.8: Performance of surrogate model for predicting the cosine distance on SVHN (Chapter 5)

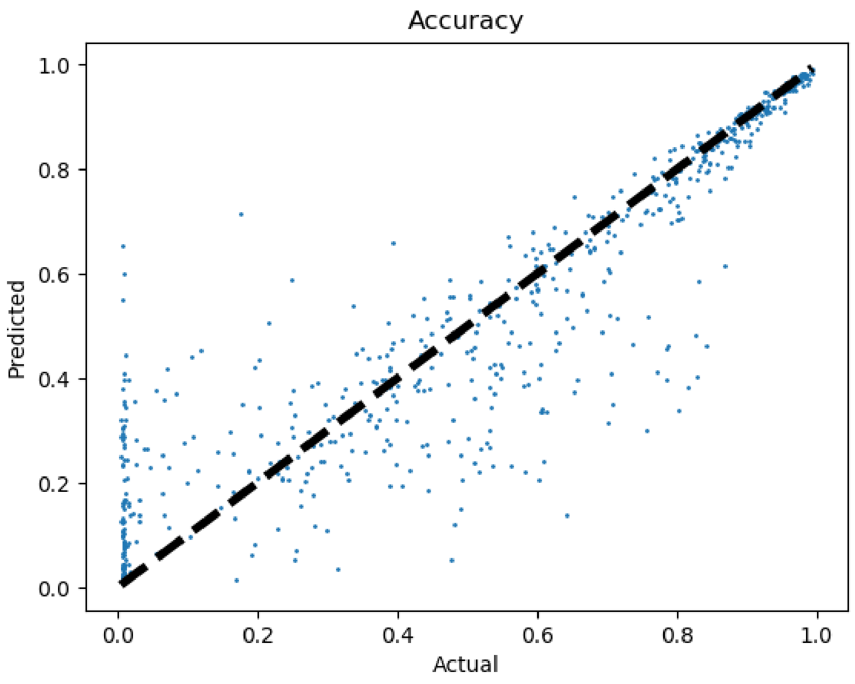


Figure A.9: Performance of surrogate model for predicting classification accuracy on SVHN (Chapter 6)

Bibliography

[AW20] Nur Ahmed and Muntasir Wahed. The de-democratization of AI: deep learning and the compute divide in artificial intelligence research. *CoRR*, abs/2010.15581, 2020.

[BC19] Yijun Bian and Huanhuan Chen. When does diversity help generalization in classification ensembles? *CoRR*, abs/1910.13631, 2019.

[BHBK05] Robert E. Banfield, Lawrence O. Hall, Kevin W. Bowyer, and W. Philip Kegelmeyer. Ensemble diversity measures and their application to thinning. *Information Fusion*, 2005.

[BJZY12] Urvesh Bhowan, Mark Johnston, Mengjie Zhang, and Xin Yao. Evolving diverse ensembles using genetic programming for classification with unbalanced data. *IEEE Transactions on Evolutionary Computation*, 17(3):368–386, 2012.

[BJZY13] Urvesh Bhowan, Mark Johnston, Mengjie Zhang, and Xin Yao. Reusing genetic programming for ensemble selection in classification of unbalanced data. *IEEE Transactions on Evolutionary Computation*, 18(6):893–908, 2013.

[BM17] S Boukir and A Mellor. Ensemble diversity analysis on remote sensing data classification using random forests. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 1302–1306, 2017.

[Bre96] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[BSI19] Alejandro Baldominos, Yago Saez, and Pedro Isasi. Hybridizing evolutionary computation and deep neural networks: an approach to handwriting recognition using committees and transfer learning. *Complexity*, 2019, 2019.

[BWT05] Gavin Brown, Jeremy L. Wyatt, and Peter Tiño. Managing diversity in regression ensembles. *J. Mach. Learn. Res.*, 6:1621–1650, 2005.

[CCTM15] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503, 2015.

[CCY06] A. Chandra, H. Chen, and X. Yao. Trade-off between diversity and accuracy in ensemble generation. *Studies in Computational Intelligence*, 16:429–464, 2006.

[CD18] Jean-Paul Chilès and Nicolas Desassis. *Fifty Years of Kriging*, pages 589–612. Springer International Publishing, Cham, 2018.

[CGDS20] Karanbir Singh Chahal, Manraj Singh Grover, Kuntal Dey, and Rajiv Ratn Shah. A hitchhiker's guide on distributed training of deep neural networks. *Journal of Parallel and Distributed Computing*, 137:65–76, 2020.

[CHKP21a] Rui P. Cardoso, Emma Hart, David Burth Kurka, and Jeremy Pitt. WILDA: wide learning of diverse architectures for classification of large datasets. In Pedro A. Castillo and Juan Luis Jiménez Laredo, editors, *Applications of Evolutionary Computation - 24th International Conference, EvoApplications 2021, Held as Part of EvoStar 2021, Virtual Event, April 7-9, 2021, Proceedings*, volume 12694 of *Lecture Notes in Computer Science*, pages 649–664. Springer, 2021.

[CHKP21b] Rui P. Cardoso, Emma Hart, David Burth Kurka, and Jeremy V. Pitt. Using novelty search to explicitly create diversity in ensembles of classifiers. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '21, page 849–857, New York, NY, USA, 2021. Association for Computing Machinery.

[CHKP22] Rui P. Cardoso, Emma Hart, David Burth Kurka, and Jeremy Pitt. Augmenting novelty search with a surrogate model to engineer meta-diversity in ensembles of classifiers. In Juan Luis Jiménez Laredo, J. Ignacio Hidalgo, and Kehinde Oluwatoyin Babaagba, editors, *Applications of Evolutionary Computation*, pages 418–434, Cham, 2022. Springer International Publishing.

[CHP19] Rui P. Cardoso, Emma Hart, and Jeremy V. Pitt. Evolving Robust Policies for Community Energy System Management. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, pages 1120–1128, New York, NY, USA, 2019. Association for Computing Machinery.

[CHP20] Rui P. Cardoso, Emma Hart, and Jeremy V. Pitt. Diversity-Driven Wide Learning for Training Distributed Classification Models. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, GECCO '20, pages 119–120, New York, NY, USA, 2020. Association for Computing Machinery.

[CJG+15] Tsung-Han Chan, Kui Jia, Shenghua Gao, et al. Pcanet: A simple deep learning baseline for image classification? *IEEE Transactions on Image Processing*, 24(12):5017–5032, Dec 2015.

[CRH+18] Rui P. Cardoso, Rosaldo J. F. Rossetti, Emma Hart, et al. Engineering Sustainable and Adaptive Systems in Dynamic and Unpredictable Environments. In Tiziana Margaria and Bernhard Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation. Distributed Systems*, pages 221–240, Cham, 2018. Springer International Publishing.

[CZM+18] Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, et al. Autoaugment: Learning augmentation policies from data. *CoRR*, abs/1805.09501, 2018.

[DCM+12] Jeffrey Dean, Greg Corrado, Rajat Monga, et al. Large scale distributed deep networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[Die00] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[DT17] Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017.

[DYL17a] Qun Dai, Rui Ye, and Zhuan Liu. Considering diversity and accuracy simultaneously for ensemble pruning. *Applied Soft Computing Journal*, 2017.

[DYL17b] Boyang Deng, Junjie Yan, and Dahua Lin. Peephole: Predicting network performance before training. *CoRR*, abs/1712.03351, 2017.

[FDM08] Dario Floreano, Peter Dürr, and Claudio Mattiussi. Neuroevolution: from architectures to learning. *Evol. Intell.*, 1(1):47–62, 2008.

[GAM18] Adam Gaier, Alexander Asteroth, and Jean-Baptiste Mouret. Data-efficient neuroevolution with kernel-based surrogate models. In *Proceedings of the genetic and evolutionary computation conference*, pages 85–92, 2018.

[GCJ15] S Gu, R Cheng, and Y Jin. Multi-objective ensemble generation. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(5):234–245, 2015.

[GJ14] Shenkai Gu and Yaochu Jin. Generating diverse and accurate classifier ensembles using multi-objective optimization. In *2014 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM)*, pages 9–15, 2014.

[GLL$^+$18] Huaping Guo, Hongbing Liu, Ran Li, et al. Margin & diversity based ordering ensemble pruning. *Neurocomputing*, 2018.

[GMC15] Jorge Gomes, Pedro Mariano, and Anders Lyhne Christensen. Devising effective novelty search algorithms: A comprehensive empirical study. In *GECCO 2015*

- *Proceedings of the 2015 Genetic and Evolutionary Computation Conference*, 2015.

[GPHMOB05] Nicolás García-Pedrajas, César Hervás-Martínez, and Domingo Ortiz-Boyer. Cooperative coevolution of artificial neural network ensembles for pattern classification. *IEEE transactions on evolutionary computation*, 9(3):271–302, 2005.

[HCB$^+$19] Yanping Huang, Youlong Cheng, Ankur Bapna, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, et al., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 103–112, 2019.

[HP04] Lu Hong and Scott E. Page. Groups of diverse problem solvers can outperform groups of high-ability problem solvers. *Proceedings of the National Academy of Sciences*, 101(46):16385–16389, 2004.

[HS18] E. Hart and K. Sim. On constructing ensembles for combinatorial optimisation. *Evolutionary Computation*, 26(1):67–87, 2018. cited By 3.

[HSP18] Emma Hart, Andreas S. W. Steyven, and Ben Paechter. Evolution of a functionally diverse swarm via a novel decentralised quality-diversity algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, pages 101–108, New York, NY, USA, 2018. ACM.

[HU16] Vishakh Hegde and Sheema Usmani. Parallel and Distributed Deep Learning. *Tech Report*, 2016.

[HZRS16a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[HZRS16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and*

Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pages 770–778. IEEE Computer Society, 2016.

[HZSG19] Alexander Hagg, Martin Zaefferer, Jörg Stork, and Adam Gaier. Prediction of neural network performance by phenotypic modeling. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '19, page 1576–1582, New York, NY, USA, 2019. Association for Computing Machinery.

[JHD12] Yangqing Jia, Chang Huang, and Trevor Darrell. Beyond spatial pyramids: Receptive field learning for pooled image features. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2012.

[KBZ+20] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, et al. Big transfer (bit): General visual representation learning. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part V*, volume 12350 of *Lecture Notes in Computer Science*, pages 491–507. Springer, 2020.

[Kri09] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. *... Science Department, University of Toronto, Tech. ...*, 2009.

[KV94] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation and active learning. In *Proceedings of the 7th International Conference on Neural Information Processing Systems*, NIPS'94, page 231–238, Cambridge, MA, USA, 1994. MIT Press.

[KW03] Ludmila I. Kuncheva and Christopher J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 2003.

[LBBH98a] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LBBH98b] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998.

[LBH15] Y. Lecun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. cited By 33590.

[LCS$^+$19] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, et al. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 82–92. Computer Vision Foundation / IEEE, 2019.

[LKB$^+$17] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, et al. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 2017.

[LKY18] Senwei Liang, Yuehaw Khoo, and Haizhao Yang. Drop-activation: Implicit parameter reduction and harmonic regularization. *CoRR*, abs/1811.05850, 2018.

[LS08] Joel Lehman and Kenneth O Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, pages 329–336, 2008.

[LS11a] Joel Lehman and Kenneth O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 2011.

[LS11b] Joel Lehman and Kenneth O Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 211–218. ACM, 2011.

[MB17] A. Mellor and S. Boukir. Exploring diversity in ensemble classification: Applications in large area land cover mapping. *ISPRS Journal of Photogrammetry and Remote Sensing*, 129:151–161, 2017. cited By 19.

[MC15] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *CoRR*, abs/1504.04909, 2015.

[MF13] Mateusz Malinowski and Mario Fritz. Learning smooth pooling regions for visual recognition. In Tilo Burghardt, Dima Damen, Walterio W. Mayol-Cuevas, and Majid Mirmehdi, editors, *British Machine Vision Conference, BMVC 2013, Bristol, UK, September 9-13, 2013*. BMVA Press, 2013.

[MK21] Akanksha Mukhriya and Rajeev Kumar. Building outlier detection ensembles by selective parameterization of heterogeneous methods. *Pattern Recognition Letters*, 146:126–133, 2021.

[MKHS14] Julien Mairal, Piotr Koniusz, Zaïd Harchaoui, and Cordelia Schmid. Convolutional kernel networks. In Zoubin Ghahramani, Max Welling, Corinna Cortes, et al., editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2627–2635, 2014.

[Mou11] Jean Baptiste Mouret. Novelty-based multiobjectivization. In *Studies in Computational Intelligence*, 2011.

[MTC18] Ricardo Pio Monti, Sina Tootoonian, and Robin Cao. Avoiding degradation in deep feed-forward networks by phasing out skip-connections. In Vera Kurková, Yannis Manolopoulos, Barbara Hammer, et al., editors, *Artificial Neural Networks and Machine Learning - ICANN 2018 - 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III*, volume 11141 of *Lecture Notes in Computer Science*, pages 447–456. Springer, 2018.

[Mur12] Kevin P Murphy. *Machine learning: a probabilistic perspective.* 2012.

[MV15a] Mark D. McDonnell and Tony Vladusich. Enhanced image classification with a fast-learning shallow convolutional neural network. In *2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015*, pages 1–7. IEEE, 2015.

[MV15b] Mark D. McDonnell and Tony Vladusich. Enhanced image classification with a fast-learning shallow convolutional neural network. In *2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015*, pages 1–7. IEEE, 2015.

[NH10] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve Restricted Boltzmann machines. In *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, 2010.

[NP15] Kaustuv Nag and Nikhil R Pal. A multiobjective genetic programming-based ensemble for simultaneous feature selection and classification. *IEEE transactions on cybernetics*, 46(2):499–510, 2015.

[NW11] Yuval Netzer and Tao Wang. Reading digits in natural images with unsupervised feature learning. *Nips*, 2011.

[OAWS15] S. Özöğür Akyüz, T. Windeatt, and R. Smith. Pruning of error correcting output codes by optimization of accuracy–diversity trade off. *Machine Learning*, 101(1-3):253–269, 2015. cited By 18.

[PCHO19] Jeremy Pitt, Rui Cardoso, Emma Hart, and Josiah Ober. Relevant expertise aggregation for policy selection in collective adaptive systems. In *Proceedings - 2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems, FAS*W 2018*, 2019.

[PDCV10] R Pasti, L N De Castro, G P Coelho, and F J Von Zuben. Neural network ensembles: Immune-inspired approaches to the diversity of components. *Natural Computing*, 9(3):625–653, 2010.

[PG13] Diego Peteiro-Barral and Bertha Guijarro-Berdiñas. A survey of methods for distributed machine learning. *Prog. Artif. Intell.*, 2(1):1–11, 2013.

[PGC$^+$19] Adam Paszke, Sam Gross, Soumith Chintala, et al. Automatic differentiation in PyTorch. In *Advances in Neural Information Processing Systems 32*, 2019.

[PSS16] Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40, 2016.

[QWD$^+$16] Junfei Qiu, Qihui Wu, Guoru Ding, et al. A survey of machine learning for big data processing. *EURASIP J. Adv. Signal Process.*, 2016:67, 2016.

[RAHL19] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4780–4789, Jul. 2019.

[RLDL20] Xiaoran Ruan, Ke Li, Bilel Derbel, and Arnaud Liefooghe. Surrogate assisted evolutionary algorithm for medium scale multi-objective optimisation problems. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 560–568, 2020.

[Sch90] Robert E Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[Sch15] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[SCL12] Pierre Sermanet, Soumith Chintala, and Yann LeCun. Convolutional neural networks applied to house numbers digit classification. In *Proceedings of the 21st International Conference on Pattern Recognition, ICPR 2012, Tsukuba, Japan, November 11-15, 2012*, pages 3288–3291. IEEE Computer Society, 2012.

[SCLM19] Kenneth O. Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen. Designing neural networks through neuroevolution. *Nat. Mach. Intell.*, 1(1):24–35, 2019.

[SGM19] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August*

2, 2019, Volume 1: Long Papers, pages 3645–3650. Association for Computational Linguistics, 2019.

[SHM$^+$16] D. Silver, A. Huang, C.J. Maddison, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. cited By 6891.

[SHS$^+$18] David Silver, Thomas Hubert, Julian Schrittwieser, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[SL15] Borut Sluban and Nada Lavrač. Relating ensemble diversity and performance: A study in class noise detection. *Neurocomputing*, 2015.

[SM02a] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

[SM02b] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[SMPS15a] Paul A. Szerlip, Gregory Morse, Justin K. Pugh, and Kenneth O. Stanley. Unsupervised feature learning through divergent discriminative feature accumulation. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 2979–2985. AAAI Press, 2015.

[SMPS15b] Paul A. Szerlip, Gregory Morse, Justin K. Pugh, and Kenneth O. Stanley. Unsupervised feature learning through divergent discriminative feature accumulation. In Blai Bonet and Sven Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 2979–2985. AAAI Press, 2015.

[SSC$^+$17] W. Sheng, P. Shan, S. Chen, et al. A niching evolutionary algorithm with adaptive negative correlation learning for neural network ensemble. *Neurocomputing*, 247:173–182, 2017. cited By 20.

[SSS$^+$17] D. Silver, J. Schrittwieser, K. Simonyan, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017. cited By 3427.

[SWX$^+$19] Yanan Sun, Handing Wang, Bing Xue, et al. Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor. *IEEE Transactions on Evolutionary Computation*, 2019.

[SXZ$^+$20a] Y. Sun, B. Xue, M. Zhang, et al. Automatically designing cnn architectures using the genetic algorithm for image classification. *IEEE Transactions on Cybernetics*, 50(9):3840–3854, 2020.

[SXZ$^+$20b] Yanan Sun, Bing Xue, Mengjie Zhang, et al. Automatically designing CNN architectures using the genetic algorithm for image classification. *IEEE Trans. Cybern.*, 50(9):3840–3854, 2020.

[SZBB19] Jörg Stork, Martin Zaefferer, and Thomas Bartz-Beielstein. Improving neuroevolution efficiency by surrogate model-based optimization with phenotypic distance kernels. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, pages 504–519. Springer, 2019.

[SZZ$^+$20] Julien Siems, Lucas Zimmer, Arber Zela, et al. Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. *CoRR*, abs/2008.09777, 2020.

[THMY21] Hao Tong, Changwu Huang, Leandro L Minku, and Xin Yao. Surrogate models in evolutionary single-objective optimization: A new taxonomy and experimental study. *Information Sciences*, 562:414–437, 2021.

[TL19] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 2019.

[TQC09] Krzysztof Trawinski, Arnaud Quirin, and Oscar Cordón. On the combination of accuracy and diversity measures for genetic selection of bagging fuzzy rule-based multiclassification systems. In *Ninth International Conference on Intelligent Systems Design and Applications, ISDA 2009, Pisa, Italy , November 30-December 2, 2009*, pages 121–127. IEEE Computer Society, 2009.

[Tsa14] Athanasios Tsakonas. An analysis of accuracy-diversity trade-off for hybrid combined system with multiobjective predictor selection. *Appl. Intell.*, 40(4):710–723, 2014.

[UH18] Neil Urquhart and Emma Hart. Optimisation and illumination of a real-world workforce scheduling and routing application (wsrp) via map-elites. In *International Conference on Parallel Problem Solving from Nature*, pages 488–499. Springer, 2018.

[Van05] Rick Van Krevelen. *Error Diversity in Classification Ensembles.* PhD thesis, 2005.

[Wol92] David H. Wolpert. Stacked generalization. *Neural Networks*, 1992.

[WY09] Shuo Wang and Xin Yao. Diversity analysis on imbalanced data sets by using ensemble models. In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2009, part of the IEEE Symposium Series on Computational Intelligence 2009, Nashville, TN, USA, March 30, 2009 - April 2, 2009*, pages 324–331. IEEE, 2009.

[XHXD15] Eric P. Xing, Qirong Ho, Pengtao Xie, and Wei Dai. Strategies and principles of distributed machine learning on big data. *CoRR*, abs/1512.09295, 2015.

[YHPC18] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing [review article]. *IEEE Computational Intelligence Magazine*, 13(3):55–75, 2018.

[ZB04] Yifeng Zhang and Siddhartha Bhattacharyya. Genetic programming in classifying large-scale data: an ensemble method. *Information Sciences*, 163(1-3):85–101, 2004.

[ZK16] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Richard C. Wilson, Edwin R. Hancock, and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. BMVA Press, 2016.

[ZON$^+$06] Zongzhao Zhou, Yew Soon Ong, Prasanth B Nair, et al. Combining global and local surrogate models to accelerate evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(1):66–76, 2006.

[ZZ13] Min Ling Zhang and Zhi Hua Zhou. Exploiting unlabeled data to enhance ensemble diversity. *Data Mining and Knowledge Discovery*, 2013.

[ZZXW19] Zhong-Qiu Zhao, Peng Zheng, Shou-Tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, 2019.

[ZZZ07] Xiaofei Zhu, Jianmin Zhong, and Lixia Zhuo. Optimization of the trade-off by artificially re-sampling for ensemble learning. In *Third International Conference on Natural Computation (ICNC 2007)*, volume 5, pages 49–53, 2007.