



**Calhoun: The NPS Institutional Archive**  
**DSpace Repository**

---

Theses and Dissertations

1. Thesis and Dissertation Collection, all items

---

2021-12

**MACHINE LEARNING OPERATIONS (MLOPS)  
ARCHITECTURE CONSIDERATIONS FOR DEEP  
LEARNING WITH A PASSIVE ACOUSTIC  
VECTOR SENSOR**

Villemez, Nicholas R.

Monterey, CA; Naval Postgraduate School

---

<http://hdl.handle.net/10945/68759>

---

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

*Downloaded from NPS Archive: Calhoun*



Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

**Dudley Knox Library / Naval Postgraduate School**  
**411 Dyer Road / 1 University Circle**  
**Monterey, California USA 93943**

<http://www.nps.edu/library>



**NAVAL  
POSTGRADUATE  
SCHOOL**

**MONTEREY, CALIFORNIA**

**THESIS**

**MACHINE LEARNING OPERATIONS (MLOPS)  
ARCHITECTURE CONSIDERATIONS FOR DEEP LEARNING  
WITH A PASSIVE ACOUSTIC VECTOR SENSOR**

by

Nicholas R. Villemez

December 2021

Thesis Advisor:  
Co-Advisor:

Marko Orescanin  
Paul Leary

**Approved for public release. Distribution is unlimited.**

THIS PAGE INTENTIONALLY LEFT BLANK

|  |   |  |   |
|--|---|--|---|
| <b>REPORT DOCUMENTATION PAGE</b>   |   |  | <i>Form Approved OMB<br/>No. 0704-0188</i>                      |
| Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC, 20503.  |   |  |   |
| <b>1. AGENCY USE ONLY<br/>(Leave blank)</b>  | <b>2. REPORT DATE</b><br>December 2021                                  | <b>3. REPORT TYPE AND DATES COVERED</b><br>Master's thesis             |   |
| <b>4. TITLE AND SUBTITLE</b><br>MACHINE LEARNING OPERATIONS (MLOPS) ARCHITECTURE<br>CONSIDERATIONS FOR DEEP LEARNING WITH A PASSIVE ACOUSTIC<br>VECTOR SENSOR  |   |  | <b>5. FUNDING NUMBERS</b>                                       |
| <b>6. AUTHOR(S)</b> Nicholas R. Villemez   |   |  |   |
| <b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b><br>Naval Postgraduate School<br>Monterey, CA 93943-5000  |   |  | <b>8. PERFORMING<br/>ORGANIZATION REPORT<br/>NUMBER</b>         |
| <b>9. SPONSORING / MONITORING AGENCY NAME(S) AND<br/>ADDRESS(ES)</b><br>N/A  |   |  | <b>10. SPONSORING /<br/>MONITORING AGENCY<br/>REPORT NUMBER</b> |
| <b>11. SUPPLEMENTARY NOTES</b> The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.  |   |  |   |
| <b>12a. DISTRIBUTION / AVAILABILITY STATEMENT</b><br>Approved for public release. Distribution is unlimited.   |   |  | <b>12b. DISTRIBUTION CODE</b><br>A                              |
| <b>13. ABSTRACT (maximum 200 words)</b><br><br>As machine learning augmented decision-making becomes more prevalent, defense applications for these techniques are needed to prevent being outpaced by peer adversaries. One area that has significant potential is deep learning applications to classify passive sonar acoustic signatures, which would accelerate tactical, operational, and strategic decision-making processes in one of the most contested and difficult warfare domains. Convolutional Neural Networks have achieved some of the greatest success in accomplishing this task; however, a full production pipeline to continually train, deploy, and evaluate acoustic deep learning models throughout their life cycle in a realistic architecture is a barrier to further and more rapid success in this field of research. Two main contributions of this thesis are a proposed production architecture for model life cycle management using Machine Learning Operations (MLOps) and evaluation of the same on live passive sonar stream. Using the proposed production architecture, this work evaluates model performance differences in a production setting and explores methods to improve model performance in production. Through documenting considerations for creating a platform and architecture to continuously train, deploy, and evaluate various deep learning acoustic classification models, this study aims to create a framework and recommendations to accelerate progress in acoustic deep learning classification research. |   |  |   |
| <b>14. SUBJECT TERMS</b><br>acoustic, deep learning, automated identification system, AIS, classification, live inference, model deployment, machine learning architecture, machine learning operations, production, machine learning, artificial intelligence, Bayesian, neural network   |   |  | <b>15. NUMBER OF<br/>PAGES</b><br>85                            |
|  |   |  | <b>16. PRICE CODE</b>   |
| <b>17. SECURITY<br/>CLASSIFICATION OF<br/>REPORT</b><br>Unclassified   | <b>18. SECURITY<br/>CLASSIFICATION OF THIS<br/>PAGE</b><br>Unclassified | <b>19. SECURITY<br/>CLASSIFICATION OF<br/>ABSTRACT</b><br>Unclassified | <b>20. LIMITATION OF<br/>ABSTRACT</b><br>UU                     |

THIS PAGE INTENTIONALLY LEFT BLANK

**Approved for public release. Distribution is unlimited.**

**MACHINE LEARNING OPERATIONS (MLOPS) ARCHITECTURE  
CONSIDERATIONS FOR DEEP LEARNING  
WITH A PASSIVE ACOUSTIC VECTOR SENSOR**

Nicholas R. Villemez  
Lieutenant, United States Navy  
BS, United States Naval Academy, 2016

Submitted in partial fulfillment of the  
requirements for the degree of

**MASTER OF SCIENCE IN COMPUTER SCIENCE**

from the

**NAVAL POSTGRADUATE SCHOOL  
December 2021**

Approved by: Marko Orescanin  
Advisor

Paul Leary  
Co-Advisor

Gurminder Singh  
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

## **ABSTRACT**

As machine learning augmented decision-making becomes more prevalent, defense applications for these techniques are needed to prevent being outpaced by peer adversaries. One area that has significant potential is deep learning applications to classify passive sonar acoustic signatures, which would accelerate tactical, operational, and strategic decision-making processes in one of the most contested and difficult warfare domains. Convolutional Neural Networks have achieved some of the greatest success in accomplishing this task; however, a full production pipeline to continually train, deploy, and evaluate acoustic deep learning models throughout their life cycle in a realistic architecture is a barrier to further and more rapid success in this field of research. Two main contributions of this thesis are a proposed production architecture for model life cycle management using Machine Learning Operations (MLOps) and evaluation of the same on live passive sonar stream. Using the proposed production architecture, this work evaluates model performance differences in a production setting and explores methods to improve model performance in production. Through documenting considerations for creating a platform and architecture to continuously train, deploy, and evaluate various deep learning acoustic classification models, this study aims to create a framework and recommendations to accelerate progress in acoustic deep learning classification research.



THIS PAGE INTENTIONALLY LEFT BLANK

---

---

# Table of Contents

---

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introduction</b>  | <b>1</b>  |
| 1.1      | Research Objectives and Contribution . . . . .   | 2         |
| 1.2      | Organization . . . . .   | 3         |
| <b>2</b> | <b>Background and Related Work</b>   | <b>5</b>  |
| 2.1      | Artificial Intelligence, Machine Learning, and Deep Learning. . . . .                    | 5         |
| 2.2      | Neural Networks . . . . .  | 9         |
| 2.3      | Convolutional Neural Networks . . . . .  | 11        |
| 2.4      | Bayesian Convolutional Neural Networks. . . . .  | 13        |
| 2.5      | Machine Learning Operations . . . . .  | 16        |
| 2.6      | Machine Learning in Acoustics. . . . .   | 22        |
| 2.7      | Significance of Applying Machine Learning Operations to Acoustic Deep Learning . . . . . | 23        |
| <b>3</b> | <b>Methodology</b>   | <b>25</b> |
| 3.1      | Dataset and Processing Pipeline . . . . .  | 25        |
| 3.2      | Model Architecture . . . . .   | 30        |
| 3.3      | Performance Metrics . . . . .  | 31        |
| 3.4      | Experiment Methodology . . . . .   | 34        |
| <b>4</b> | <b>Results</b>   | <b>37</b> |
| 4.1      | Machine Learning Operations Architecture . . . . .                                       | 37        |
| 4.2      | Model Performance Comparisons. . . . .   | 52        |
| <b>5</b> | <b>Conclusion</b>  | <b>57</b> |
| 5.1      | Discussion . . . . .   | 57        |
| 5.2      | Future Work . . . . .  | 58        |

|                                  |           |
|----------------------------------|-----------|
| <b>List of References</b>        | <b>61</b> |
| <b>Initial Distribution List</b> | <b>67</b> |

---

---

## List of Figures

---

|             |  |    |
|-------------|--|----|
| Figure 2.1  | Artificial Intelligence Venn Diagram . . . . .                               | 6  |
| Figure 2.2  | Artificial Intelligence Winters Timeline . . . . .                           | 6  |
| Figure 2.3  | Multi-Layer Perceptron . . . . .   | 10 |
| Figure 2.4  | Rectified Linear Activation Function . . . . .                               | 11 |
| Figure 2.5  | Convolutional Neural Network Feature Maps . . . . .                          | 12 |
| Figure 2.6  | Typical Convolutional Neural Network Architecture . . . . .                  | 13 |
| Figure 2.7  | Neural Network and Bayesian Neural Networks . . . . .                        | 14 |
| Figure 2.8  | The Machine Learning Operations (MLOps) Lifecycle . . . . .                  | 17 |
| Figure 2.9  | The Machine Learning Operations Process . . . . .                            | 18 |
| Figure 2.10 | Machine Learning System Diagram . . . . .                                    | 20 |
| Figure 2.11 | Machine Learning Platform Components . . . . .                               | 21 |
| Figure 3.1  | Monterey Accelerated Research System . . . . .                               | 25 |
| Figure 3.2  | Acoustic Deep Learning Application Physical Architecture . . . . .           | 26 |
| Figure 3.3  | Acoustic Data Processing Pipeline . . . . .                                  | 27 |
| Figure 3.4  | Mel Log Spectrograms . . . . .   | 28 |
| Figure 3.5  | Custom Convolutional Neural Network Architecture . . . . .                   | 31 |
| Figure 4.1  | Acoustic Deep Learning Machine Learning Operations Process Diagram . . . . . | 37 |
| Figure 4.2  | Application Home Page . . . . .  | 39 |
| Figure 4.3  | Live Spectrogram and Model Predictions User Interface . . . . .              | 40 |
| Figure 4.4  | Live Model Uncertainty User Interface . . . . .                              | 41 |

|             |  |    |
|-------------|--|----|
| Figure 4.5  | Automated Identification System User Interface . . . . .       | 42 |
| Figure 4.6  | Total Model Performance Metrics . . . . .                      | 42 |
| Figure 4.7  | Model Performance Classification Reports . . . . .             | 43 |
| Figure 4.8  | Daily Model Performance . . . . .                              | 44 |
| Figure 4.9  | Model Deployment Form . . . . .                                | 45 |
| Figure 4.10 | Stop or Reactivate Production Models . . . . .                 | 46 |
| Figure 4.11 | Average Ship Class Size . . . . .                              | 47 |
| Figure 4.12 | Average Size of Designations with Unknown Ship Class . . . . . | 47 |
| Figure 4.13 | Assign Ship Class Form . . . . .                               | 48 |
| Figure 4.14 | Average Size of Designations with a Known Ship Class . . . . . | 49 |
| Figure 4.15 | Alter Ship Class Form . . . . .                                | 49 |
| Figure 4.16 | Application Database Schema . . . . .                          | 51 |

---

---

## List of Tables

---

|           |   |    |
|-----------|---|----|
| Table 3.1 | Automatic Identification System Formatting . . . . .          | 29 |
| Table 3.2 | Ship Class and Designation Mapping . . . . .                  | 29 |
| Table 3.3 | True Label Assignment Process . . . . .                       | 30 |
| Table 4.1 | Pre-Trained Model Accuracy Over Time . . . . .                | 52 |
| Table 4.2 | Newly Trained Model Accuracy Over Time . . . . .              | 53 |
| Table 4.3 | Model Performance of Different Shuffle Groupings . . . . .    | 54 |
| Table 4.4 | Model Accuracy Over Time Trained on Additional Data . . . . . | 55 |

THIS PAGE INTENTIONALLY LEFT BLANK

---

---

## List of Acronyms and Abbreviations

---

|               |                                      |
|---------------|--------------------------------------|
| <b>AI</b>     | Artificial Intelligence              |
| <b>AIS</b>    | Automatic Identification System      |
| <b>ANN</b>    | Artificial Neural Network            |
| <b>API</b>    | Application Programming Interface    |
| <b>BDL</b>    | Bayesian Deep Learning               |
| <b>BNN</b>    | Bayesian Neural Network              |
| <b>CD</b>     | Continuous Delivery/Deployment       |
| <b>CE</b>     | Continuous Evaluation                |
| <b>CI</b>     | Continuous Integration               |
| <b>CNN</b>    | Convolutional Neural Network         |
| <b>DevOps</b> | Development and Operations           |
| <b>DL</b>     | Deep Learning                        |
| <b>DNN</b>    | Deep Neural Network                  |
| <b>DoD</b>    | Department of Defense                |
| <b>DOT</b>    | Department of Transportation         |
| <b>ELBO</b>   | Negative Evidence Lower Bound        |
| <b>IoT</b>    | Internet of Things                   |
| <b>JAIC</b>   | Joint Artificial Intelligence Center |
| <b>JSON</b>   | JavaScript Object Notation           |



**KL** Kullback-Leibler

**MARS** Monterey Accelerated Research System

**MC** Monte Carlo

**ML** Machine Learning

**MLOps** Machine Learning Operations

**MMSI** Maritime Mobile Service Identity

**NLP** Natural Language Processing

**NN** Neural Network

**NPS** Naval Postgraduate School

**ONI** the Office of Naval Intelligence

**ReLU** Rectified Linear Activation Function

**REST** Representational State Transfer

**ROC** Receiver Operating Characteristics

**STFT** Short Time Fourier Transform

**U.S.** United States

**UWDC** Undersea Warfighting Development Center

---

---

## Acknowledgments

---

First, I would like to thank my advisors, Dr. Marko Orescanin and Dr. Paul Leary, for the mentorship they contributed to enable me to complete this thesis. I would also like to thank former students LT Sabrina Atchley, LCDR Brandon Beckler, and LT Andrew Pfau whose previous work and assistance helped make this possible. I have built upon the previous work of many and am grateful for their efforts, which have allowed me to explore a problem that I personally find interesting.

Finally, I would like to thank my wife, Alana Villemez, for staying up late and pulling all-nighters with me countless times, for sharing space at home throughout the COVID lockdowns, and for being such a supportive partner. All those times we were too busy to do the dishes or had to decline plans with friends are what allowed me to achieve this. I am so thankful for your encouragement and understanding, and I could not have completed this without them.

THIS PAGE INTENTIONALLY LEFT BLANK

---

---

# CHAPTER 1:

## Introduction

---

The rapid acceleration of Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL) technologies in modern society has created opportunities for growth and efficiency in many areas such as business operations, Internet of Things (IoT), and numerous software applications. These opportunities also extend to the Department of Defense (DoD) where AI/ML offers to improve and accelerate warfighting capabilities and decision-making processes. In order to take advantage of these opportunities, the DoD has invested resources into various AI/ML initiatives. For example, the increased emphasis on AI/ML development has led to the publication of the first AI strategy in 2018, as well as the establishment of the Joint Artificial Intelligence Center (JAIC) to guide the DoD and United States (U.S.) government to more effectively harness AI/ML capabilities [1], [2]. These initiatives, among others, seek to leverage emerging technology to ultimately provide warfighting advantages and shorten decision-making timelines within the DoD and broader U.S. government.

New applications for ML to augment or replace existing processes in the DoD are continually being discovered. One such opportunity that is being developed is the application of ML to acoustic classification, which is the identification of particular objects from their acoustic signature. Currently, human-operated ashore and afloat workstations are monitored by sonar analysts to identify significant activity based on acoustic signatures [3]. ML offers potential to augment these efforts to increase their scale and efficiency.

While the Navy is pursuing various efforts to augment acoustic analysis with AI/ML capabilities within organizations such as the Office of Naval Intelligence (ONI) and Undersea Warfighting Development Center (UWDC), they are still nascent and have not yet achieved their full potential [4], [5]. These organizations also still need to fully address complex data labeling and pipeline considerations that span multiple organizations and stakeholders across the globe [5]. As the number of acoustic sensors and their capabilities increase, the amount of corresponding data also increases, and the number of human analysts is already insufficient to process currently available data [5]. Automating elements of this currently manual process offers promise to increase the scale and accuracy at which analysts can

process data and discover insights by filtering out insignificant data and focusing efforts towards significant activity. Using ML to classify acoustic data for this purpose can assist greatly in automating and accelerating these tasks.

However, rapid and unforeseen changes are prevalent in acoustic environments, which affects the data feeding into ML models and may degrade model accuracy [6]. Due to dynamic environments, ML models must be researched and improved. However, to reliably deliver accurate ML models, a reliable process and platform to continually monitor model performance, cue new development, and update old models is necessary. Due to the frequently changing conditions, processes must support continuous evaluation, improvement, and deployment of ML models. Machine Learning Operations (MLOps) is a set of processes that address this need for managing the lifecycle of ML models to ensure their continued effectiveness. Applying MLOps principles to acoustic ML efforts stands to enhance the quality and quantity of research efforts by more effectively managing the ML lifecycle between researchers and end-users.

## **1.1 Research Objectives and Contribution**

This work builds off of previous research conducted by LT Andrew Pfau in his thesis "Multi-Label Classification of Underwater Soundscapes Using Deep Convolutional Neural Networks," LT Sabrina Atchley in her thesis "Active Bayesian Deep Learning with an Acoustic Vector Sensor," and LCDR Brandon Beckler in his thesis "Enhanced Multi-Label Classification of Heterogeneous Underwater Soundscapes by Convolutional Neural Networks Using Bayesian Deep Learning" [3], [7], [6].

This work seeks to outline a realistic MLOps process that can be emulated for operational use to continually train, evaluate, and deploy ML models for live acoustic inference. This work also emphasizes the importance of implementing MLOps processes by continuously evaluating model performance over time and showing how it can improve model research and development. This work achieves this by outlining a live inference architecture for deploying acoustic classification models into production to augment sonar operator decision-making alongside other data streams for fusion analysis. Additionally, this work seeks to propose a holistic methodology for continuously developing, deploying, updating, and replacing production ML models to account for performance degradation and rapidly replacing deployed

models with improved versions. In order to emphasize the importance of having a process for Continuous Integration (CI) and Continuous Delivery/Deployment (CD) of ML models, this work compares the performance of various models trained over varying time periods in a live-inference environment to display degradation that occurs over time in a real-world environment. Additionally, this work seeks to establish a baseline human-machine teaming interface through a platform that connects MLOps processes to a production environment and integrates information to increase model interpretability and trustworthiness for end-users AI.

Specific research questions include:

1. What is the performance difference between acoustic ML models in production and in development?
2. What is the performance difference between acoustic deep learning models trained on smaller datasets and larger datasets?
3. How does concept drift affect production ML models?
4. What is an adequate live-inference production architecture for acoustic ML models?
5. What is an adequate MLOps architecture and process to accelerate acoustic ML research?

## **1.2 Organization**

Chapter II introduces MLOps, ML software applications, Bayesian Neural Networks, Convolutional Neural Networks, multilabel and multiclass classification, MLOps, and previous related work for deep learning with acoustics.

Chapter III discusses the data processing pipelines, Bayesian and Convolutional Neural Network model architectures, and the experiment methodology.

Chapter IV outlines the proposed live-inference MLOps platform and the resulting application and model performance.

Chapter V explores the conclusions that can be drawn from the results discussed in Chapter IV and how any assumptions may affect these conclusions. It ends with a discussion of

opportunities for future work and improvements that can be made to the platform.

---

---

## CHAPTER 2: Background and Related Work

---

### **2.1 Artificial Intelligence, Machine Learning, and Deep Learning**

The terms AI, ML, and DL are often used interchangeably to refer to a broad set of automated capabilities enabled by computing technology that can mimic or exceed human performance. However, it is important to differentiate between these terms in order to better understand their applications and relevancy. For the purposes of this work, the following definitions have been taken from the *Oxford English Dictionary*, which states that AI refers to “the capacity of computers or other machines to exhibit or simulate intelligent behaviour” [8]. ML is a subset of AI, defined as “the capacity of computers to learn and adapt without following explicit instructions, by using algorithms and statistical models to analyse and infer from patterns in data” [9]. DL is the most specific subset of the three terms, and is defined as “a type of machine learning considered to be in some way more dynamic or complete than others, especially machine learning based on artificial neural networks in which multiple layers of processing are used to extract progressively higher level features from data” [10]. Goodfellow et al. propose an additional categorization of Representation Learning between ML and DL, which uses ML to learn a representation of the input data, which this work does not discuss [11]. This concept of the progression of AI to ML to DL is depicted in Figure 2.1, outlining the increasingly more specific and complex nature of each term, with DL generally referring to the most complex AI model architectures.



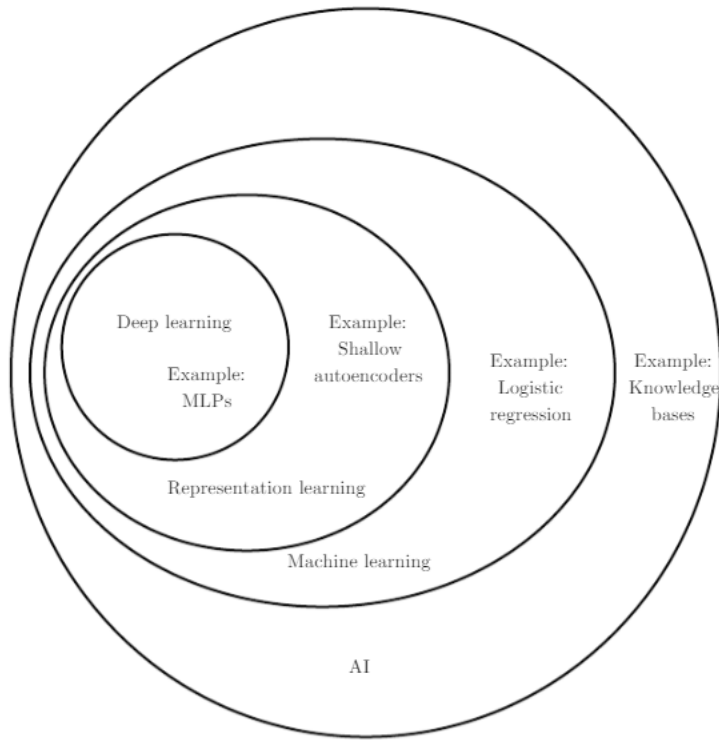


Figure 2.1. Venn diagram displaying the relationship between AI, ML, and DL. Source: [11].

### 2.1.1 History of Artificial Intelligence

AI technologies have experienced several periods of popularity throughout history, with the periods of disinterest categorized as “AI Winters,” as depicted in Figure 2.2 [12].

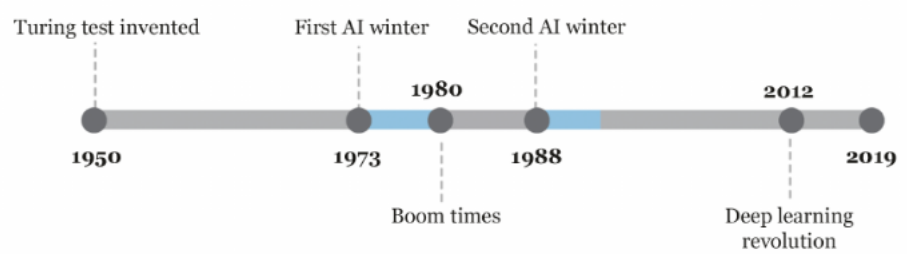


Figure 2.2. Timeline of AI winters. Source: [12].

AI was first introduced as a concept in the 1940s where prominent figures such as the mathematician Alan Turing explored creating intelligent machines [13]. Warren McCulloch and Walter Pitts presented the first Artificial Neural Network (ANN) in 1943 as a method of using propositional logic to represent how biological neurons in animal brains interact with each other [14]. The term AI itself was first used in 1956 by Marvin Minsky and John McCarthy when they hosted the Dartmouth Summer Research Project on Artificial Intelligence [13]. Another key achievement that progressed the field of AI and DL was the use of back-propagation to successfully train Neural Networks (NNs) in 1986 [11]. Rather than just sending inputs through a NN for a prediction and manually tuning the model parameters, the difference between the actual output and the desired output could be measured via a loss function and the parameters of the model updated automatically through the back-propagation algorithm. Achievements in the past decade have led to renewed interest in the potential for AI applications, particularly using DL methods due to their displayed ability to outperform humans and superior accuracy [11]. One achievement that significantly increased interest in AI was when Google's AlphaGo program defeated the world champion in the Go board game in 2015 [13]. More recent examples include OpenAI's development of one of the most advanced Natural Language Processing (NLP) models, GPT-3, in 2020 and the DALLE text-to-image discrete variational autoencoder in 2021 [15], [16]. The field of AI has progressed significantly in recent history, and as a result, a plethora of techniques and methods have been discovered to develop AI capabilities.

### **2.1.2 Supervised Versus Unsupervised Learning**

Typically, ML models are trained in either a supervised or unsupervised manner. Supervised ML refers to structured data inputs that contains a label as the target variable for the model to predict. Supervised models learn from labeled datasets in order to make predictions on structured, unlabeled datasets. An example of a supervised ML task is classification, in which the model takes features as input variables to infer a predicted target variable as the class categorization [14].

Unsupervised ML takes unstructured data as inputs which do not contain labels. An example of an unsupervised ML task is clustering, in which a model takes unstructured data as input and creates clusters of the features based on how alike they are to identify patterns [14].

### 2.1.3 Multiclass Versus Multilabel Models

Multiclass classification refers to models that have the ability to discriminate between more than two classes of target variables. This means that if a set of features corresponds to three different possible categories, the model has the ability to predict any of the three categories, as opposed to binary classifiers which are only able to predict two classes. However, multiclass classifiers are only capable of predicting a single output. The final classification layer in multiclass NNs contains a softmax activation function to achieve this, as shown in Equation 2.1.

$$\sigma(x)_i = \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}} \quad (2.1)$$

The input vector of the layer is referred to by  $x$ , with  $K$  as the number of classes and  $x_j$  as the output vector [14]. The softmax function results in each neuron in the final layer outputting a probability of its corresponding class with all probabilities summing to one [14]. The neuron with the highest probability can then be used as the predicted class [14].

Multilabel classifiers are able to simultaneously predict multiple classes, and are usually comprised of multiple binary classification problems using a cross-entropy loss function [14]. For instance, if more than one class is present in a particular set of input features, a trained multilabel classifier could predict all present classes simultaneously. Similar to multiclass models, each output neuron corresponds to a class and will output a probability value on the likelihood that a class is present. This architecture uses a sigmoid activation function on the output layer, which transforms each neuron's output to a probability between zero and one [14].

$$\sigma(t) = \frac{1}{1 + e^{-t}} \quad (2.2)$$

The weighted sum of inputs from the previous layer is referred to by  $t$ , and the output  $\sigma(t)$  serves as the input to the next layer [14]. A probability threshold, typically set to 0.5, can be

applied to the outputs of each neuron to determine which classes are predicted as present in the input features [14].

### **2.1.4 Overfitting and Underfitting Models**

While training models, there remains a danger for both overfitting and underfitting the training data [14]. Overfitting the training data results in models that perform well on the training data but they do not perform well on any other data [14]. This can occur as a result of model architectures that are too complex for the training data [14]. Underfitting is another concern that occurs when a model architecture is too simple compared to the training data [14]. The result of underfitting is that the model cannot adequately learn the patterns in the training data [14]. Both underfitting and overfitting result in lower model performance [14].

## **2.2 Neural Networks**

NNs are composed of elements called neurons and connections between those neurons [14]. The neurons are organized into layers, with each layer's neurons interconnected with the next layer's [14]. NNs contain at least an input layer, where features are passed into a hidden layer, and an output layer, which outputs the model's predictions [14]. Multi-layer perceptrons are the simplest form of neural networks, which contain at least one hidden layer between the input and output layers, as depicted in Figure 2.3.

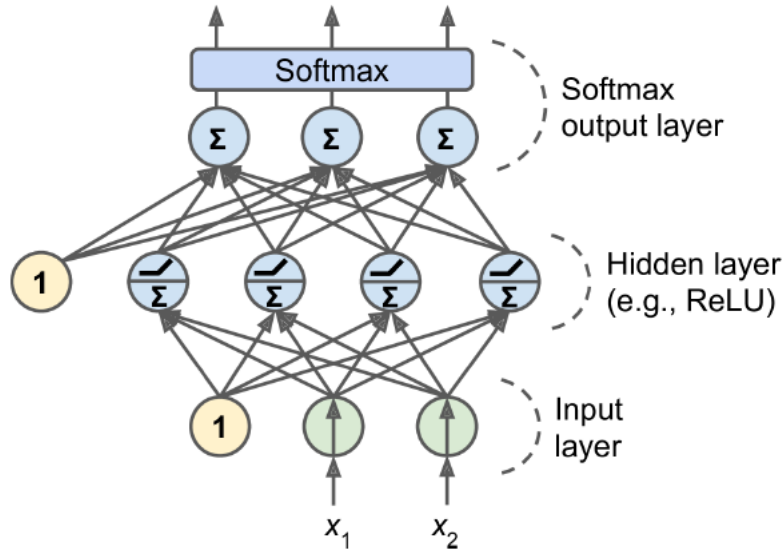


Figure 2.3. A multi-layer perceptron. Source: [14].

A neuron refers to a node in the model architecture, which takes all input signals from the previous layer and computes their weighted sum, as depicted by Equation 2.3, where  $w$  refers to the weight and  $x$  refers to the input feature. The output is then sent through an activation function to exit the neuron and traverse the output synapses to the next layer in the model [14].

$$g(x) = \sum_{i=1}^n w_i x_i \quad (2.3)$$

Features are initially passed through the input layer, and as they traverse the connections to the first hidden layer, they are multiplied by a weight and bias value associated with the connection to the next neuron [14]. The neurons in the first hidden layer then compute the weighted sum of the input connections and sends the result through the activation function [14]. The activation function's output is then sent to the next layer of neurons, where this process is repeated until the features reach the output layer [14]. A commonly used activation function in modern NNs is the Rectified Linear Activation Function (ReLU) activation function shown in Figure 2.4, where  $z$  is the output from Equation 2.3 [11]. ReLU

allows models to perform nonlinear transformations, but because it is still similar to linear functions, it retains the benefits of how well linear functions are able to generalize and be optimized [11].

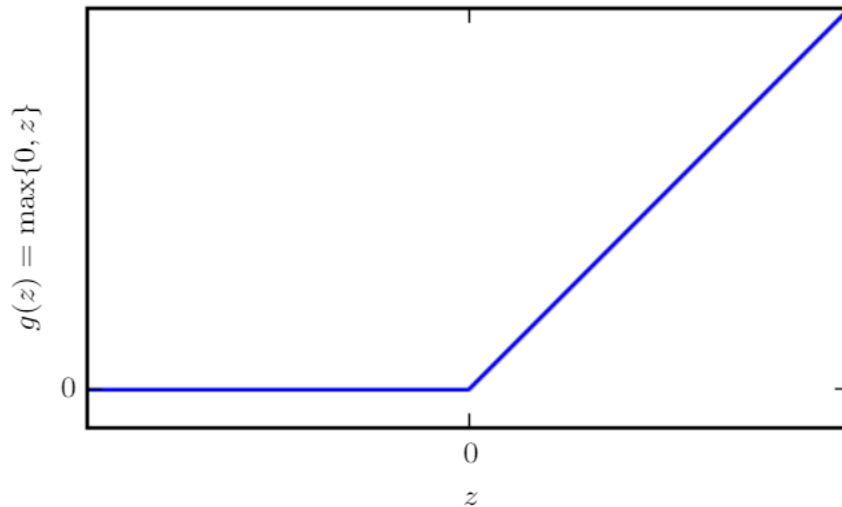


Figure 2.4. ReLU activation function. Source: [11].

The output layer contains the same number of neurons as the number of properties the model is built to predict. The output of the final layer in the network is used for the model's prediction. For a trained model, the output can be used in a production environment or for further research and testing. However, if the model is training on a dataset, a cost or loss function is then used to update the weights and biases. This is accomplished through back-propagation, where the chosen loss function computes an error value, which is used to update the model's trainable parameters and decrease error on the next epoch [14].

## 2.3 Convolutional Neural Networks

Researchers discovered that adding more layers resulted in significant improvements to accuracy and increased a model's ability to handle complicated datasets [6]. However, there is a trade-off in that the more complicated a model's architecture is, the more computing and time resources are required to train it. Models with a deep number of hidden layers are

referred to as Deep Neural Networks (DNNs), which comprise the field of DL research. Convolutional Neural Networks (CNNs) in particular, as a subset of DNNs, proved to perform well on tasks such as pattern recognition and image classification [17]. CNNs contain convolutional layers, which consist of filters that are applied to the input and activate on certain features [14]. The filters contain trainable weights that are multiplied and summed on the input features [14]. Filters are moved across input features by a defined step size [14]. A step size of one results in an output that is the same size as the input [3]. Larger step sizes result in a smaller output than the input, which is a technique known as pooling that lessens the number of trainable parameters to decrease computation requirements and prevent overfitting [3]. The filtering process in a convolutional layer results in highlighted regions of the input features, creating a feature map [14]. These feature maps are well-suited to detecting objects in images or other patterns in an input [3], [6]. Figure 2.5 illustrates multiple convolutional layers and filters being applied to an image to create feature maps.

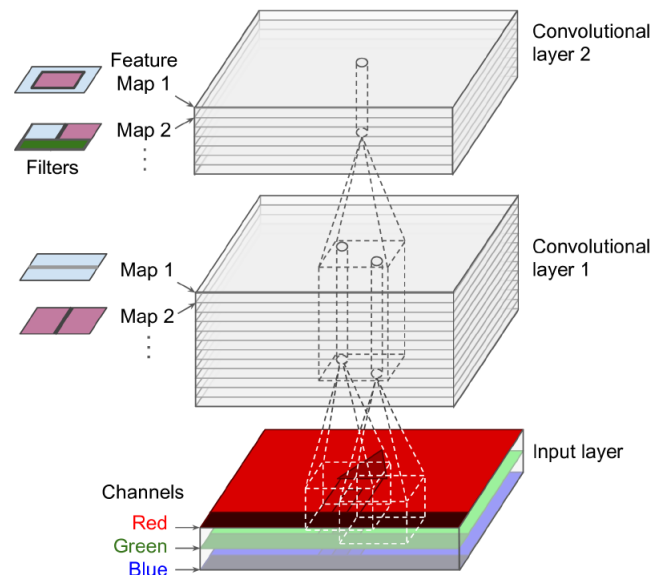


Figure 2.5. Convolutional layers with multiple feature maps applied to an image with three color channels. Source: [14].

Combining convolutional and pooling layers with fully connected neural network layers results in a typical CNN architecture such as the one displayed in Figure 2.6 [6], [14]. For

example, if one of the output nodes corresponded to pagodas in Figure 2.6, then a trained model would activate that node the most with the depicted input image.

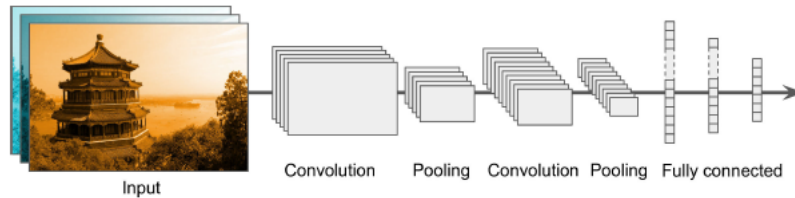


Figure 2.6. Typical CNN architecture and example image input. Source: [14].

## 2.4 Bayesian Convolutional Neural Networks

Bayesian Deep Learning (BDL) is a method of DL that uses Bayesian probabilistic models to make predictions using Deep Neural Networks [18]. BDL enables better model interpretability because it allows for the estimation of uncertainty measures in the predictions. Uncertainty measures identify which predictions have relatively higher or lower levels of confidence and how likely a particular prediction is to be correct. BDL is able to estimate uncertainty, because the model uses various methods of sampling probabilistic distributions, one which is depicted in Figure 2.7.



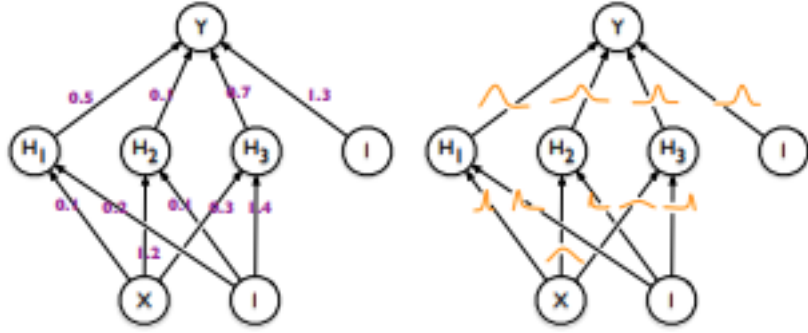


Figure 2.7. A simple neural network with one hidden layer. The left image is a deterministic neural network and the right is a Bayesian Neural Network. Source: [19].

In this method, the model learns probabilistic distributions as the weights, which results in a different NN for each prediction that is made and, therefore, potentially different predictions [7]. When making more than one prediction on the same data, uncertainty can be quantified by measuring the amount of variation in the results [7]. When the model is more certain about a set of predictions, it will contain less variation, and when it is less certain, it will contain greater variation [7]. Presenting uncertainty measures to end-users increases a model’s interpretability and ultimately helps improve end-user trust in the model’s outputs [20], [7].

The loss function used to minimize the probability distribution of the weights during training is the Negative Evidence Lower Bound (ELBO) loss function depicted in Equation 2.4 [21], [6]. The first term in the equation represents the expected likelihood or value, and the second term is the Kullback-Leibler (KL) divergence, which is used to measure the proximity between two probability densities [21], [6]. Minimizing the KL divergence results in minimizing the ELBO function during backpropagation [21], [6].

$$\mathcal{L}(\theta) = -\mathbb{E}_{q_{\theta}}[\log p(\mathcal{D} | \omega)] + \text{KL}(q_{\theta}(\omega) || p(\omega)) \quad (2.4)$$

This work primarily deals with a method of BDL by modeling distributions over weights

using a Monte Carlo (MC) dropout approach, where dropout layers are introduced to the model that randomly exclude neurons from the network while training [22]. Dropout layers are usually employed to combat overfitting during training, but in this context, the randomness serves to form a probability distribution for the models [7]. This method of approximating uncertainty has been shown to require fewer computational resources than KL divergence without sacrificing performance [22]. When passing a sample through a dropout Bayesian Neural Network (BNN) multiple times, neurons are excluded at random, resulting in different predictions from which uncertainty can be calculated [7]. Equation 2.5 represents a Bayesian Neural Network using an MC Dropout approach as implemented in [7].

$$p(y = c|x) = \int p(y = c|x, \omega)p(\omega)d\omega \quad (2.5)$$

The input to the network is  $x$ , and  $c$  represents the output of the model. The integral of all possible weights must be taken to estimate an output class  $c$ , as depicted in Equation 2.6 [7].

$$\approx \int p(y = c|x, \omega)q^*(\omega)d\omega \quad (2.6)$$

The unknown value  $p(\omega)$  is assumed to be derived from a Bernoulli distribution that is created from the active dropout layers [7].

$$\approx \frac{1}{T} \sum_t p(y|x, \omega_t) = \frac{1}{T} \sum_t p_c^t \quad (2.7)$$

After running the network on the same sample multiple times, the sum of the results is then used to calculate the MC integration [7].

## 2.4.1 Uncertainty

Uncertainty can be quantified as a result of measurements from the data or in the model parameters [7]. A plethora of methods are available to calculate uncertainty; however, there is no consensus on the most effective measurement [6]. This work primarily discusses entropy, aleatoric, and epistemic uncertainty measures.

Entropy is computed by taking the average amount of information in the predictive distribution, as shown in Equation 2.8 [7], [21].

$$H_p(\hat{y} | \mathbf{x}^*) = - \sum_c \bar{p}_c \log \bar{p}_c \quad (2.8)$$

Dividing  $H_p$  by  $\log 2^c$  results in a value between zero and one to normalize the output, resulting in the number of classes  $c$ , as depicted in Equation 2.9 [21].

$$H_p^*(\hat{y} | \mathbf{x}^*) = - \sum_c \bar{p}_c \frac{\log \bar{p}_c}{\log 2^c} \quad (2.9)$$

Aleatoric and epistemic uncertainties are two alternative calculations to entropy, where aleatoric uncertainty reveals information about noise in the data and epistemic uncertainty is in reference to uncertainty of model parameters [7]. Both the aleatoric and epistemic uncertainty calculations are depicted in Equation 2.10 [7].

$$\underbrace{\frac{1}{T} \sum_{t=1}^T \text{diag}(\hat{p}_{c_t}) - \hat{p}_{c_t}^2}_{\text{aleatoric}} + \underbrace{\frac{1}{T} \sum_{t=1}^T (\text{diag}(\hat{p}_{c_t}) - \bar{p}_c)^2}_{\text{epistemic}} \quad (2.10)$$

## 2.5 Machine Learning Operations

### 2.5.1 Machine Learning Operations Process

According to Google’s “Practitioner’s Guide to MLOps,” MLOps is “a set of standardized processes and technology capabilities for building, deploying, and operationalizing ML

systems rapidly and reliably” [23]. Essentially, MLOps is a set of processes that attempts to minimize the costs incurred by the technical debt described by Sculley et al. and to account for the effects of concept drift in ML systems [24]. MLOps is a term derivative from Development and Operations (DevOps), which “integrates the tasks, knowledge and skills pertaining to planning, building, and running software product activities in a joint cross-functional team within the IT function” [25]. Similarly, MLOps is a framework that attempts to manage the continuous flow of ML model lifecycles as efficiently as possible so that model performance and utility is maximized to account for concept drift, system improvements, and ongoing maintenance [23]. As annotated in Figure 2.8, MLOps accounts for the entire lifecycle of managing ML models, to include creating, deploying, and replacing models and their accompanying data.



Figure 2.8. The MLOps lifecycle. Source: [23].

Figure 2.9 depicts how the different elements of the MLOps process interact with each other throughout the MLOps lifecycle. The MLOps process begins with researching and

developing new models and data processing pipelines where improvements and baselines are established for a particular capability [23]. After new models are developed, testing and preparation of the models for deployment is conducted to ensure they are ready for a real-world environment [23]. Once models are prepared and deployed into production, Continuous Evaluation (CE) of model performance must be monitored so their accuracy remains above acceptable thresholds for their intended purposes [23]. Finally, as model performance drops or research and development efforts produce enhancements, CI/CD of ML models must be integrated to ensure timely replacement of outdated models in production [23]. Interwoven between each step in the MLOps process is continuous management and curating of the data used to train new models and used by deployed models [23]. Ultimately, a holistic MLOps process seeks to automate each step in the lifecycle to the greatest extent possible [23]. A successful, continuous MLOps process ensures that less time is spent on ephemeral tasks to transition between different stages in a model’s lifecycle so that rapid improvements and corrections can be made, ensuring the best performance and experience possible for end-users [23]. Using MLOps as holistic approach to manage model lifecycles is important to decrease technical debt in ML systems and to account for decreasing model performance as a result of causalities such as concept drift.

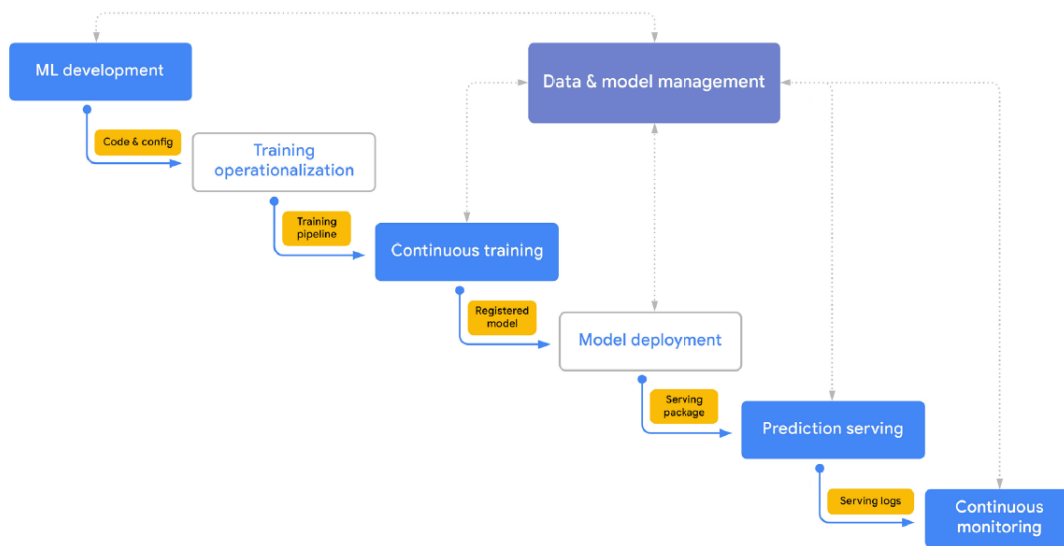


Figure 2.9. The MLOps process. Source: [23].

## 2.5.2 Technical Debt

The advent of MLOps developed from Google’s 2015 paper, “Hidden Technical Debt in Machine Learning Systems,” where Sculley et al. explore the hidden costs of ongoing maintenance for ML systems [24]. Sculley et al. assert that over time, organizations tend to incur significant technical debt, especially when increasingly adopting ML systems [24]. Technical debt refers to the general observation that developing and deploying ML models is easy, while maintaining them is difficult and the requirements compound significantly over time [24]. Sculley et al. attempt to categorize various types of debt into eroding boundaries, data dependencies, feedback loops, design anti-patterns, configuration debt, and changes in the external world [24].

Software engineering practices have shown that building software in a modular, separate fashion results in easier debugging and improvements [24]. However, this modular approach is difficult to enforce in ML systems because they are built to combine signals together for their inputs, making isolated improvements impossible [24]. Additionally, transfer learning to adapt or fine-tune models that receive inputs from other models creates chained dependencies known as “correction cascades” [24]. Undeclared consumers of a model’s output can also create additional debt that link a model’s output to other parts of a tech stack [24].

Data dependencies in ML systems can result in unstable inputs to a model, where the data qualitatively or quantitatively changes over time [24]. If changes or improvements are made to data streams without accounting for these changes in the model, the improvement can actually have a detrimental impact on the model performance [24].

ML systems can also affect their own performance if they update over time, such as those configured for active learning [24]. These effects manifest in both direct and hidden feedback loops that can influence a model’s behavior or guide the underlying predictors [24].

As depicted in Figure 2.10, only a small proportion of code in an ML system is actually related to the act of ML [24]. The other code is typically related to orchestrating the system and its data pipelines [24]. Poor approaches to managing this supporting code can result in over complicated features or generalized code that limits possibilities in the ML system [24].

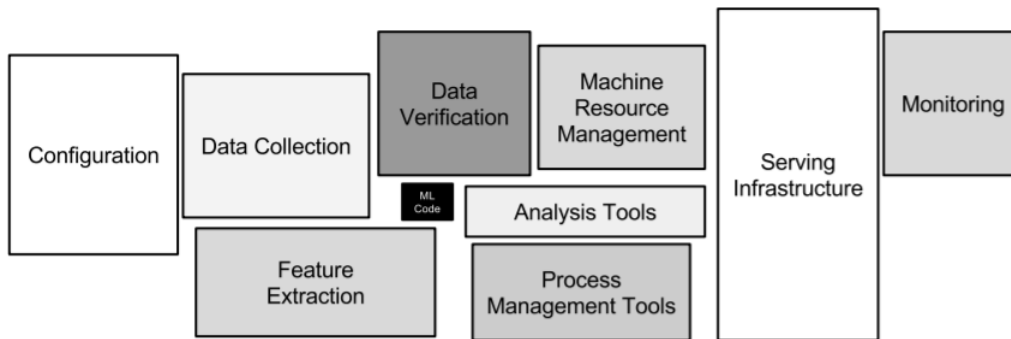


Figure 2.10. Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex. Source: [24].

Configuration debt is another area in which ML systems can accumulate costs [24]. Typically, ML systems contain a set of configurable options related to data processing and model parameters [24]. Failing to properly manage or keep track of configuration options so they are clear to developers and end-users can be costly [24].

Finally, since ML systems often interact directly with the real-world, changes or instabilities to these external factors result in ongoing maintenance costs to keep models updated and functional [24].

### 2.5.3 Concept Drift

One way external changes manifest in ML systems is through concept drift, where the input data changes over time due to hidden contexts that are not present in predictor variables [26]. Change that results in a shifting distribution of the target variable is commonly referred to as real concept drift [27]. Model performance can also be affected by shifting data distributions of the input variables, which is a case known as virtual concept drift [27]. Each variant of concept drift can happen individually or simultaneously, but all cases eventually require that a model be updated [26].

For example, take a CNN classifier that is built to predict the model of a car in an image. Over time, car manufacturers release newer versions of the model that may look slightly different. The underlying data changing over time may result in degraded model performance due to

the concept drift in the system. Concept drift can be accounted for by retraining the original model or training a new model with training data that accounts for the new distributions.

Various attempts have been made to account for concept drift with ML in acoustic environments. Ntalampiras architects a system that incorporates online learning and an automated mechanism for updating class labels under a holistic concept drift framework [28]. Daqiqil id et al. developed a Kernel Density Estimation method that evaluates the probability density function to measure the degree of difference between values in corresponding variables and quantify concept drift in an acoustic environment [29]. This work attempts to explore the effects that concept drift has on a live acoustic ML system’s performance using a passive acoustic vector sensor.

### 2.5.4 Previous Work in Machine Learning Operations Architectures

Baylor et al. architect an example Tensorflow-based, production ML platform for the Google Play Store in which they apply a framework that supports building one machine learning platform for multiple machine learning tasks, continuous training and serving, an exposed interface for engineers and end-users to interact with the platform, and production-level reliability and scalability [30]. Baylor et al. display how their platform enables teams to easily deploy ML models and limit technical debt while doing so for a holistic system [30]. In their work, they outline the typical components of a ML platform, as depicted in Figure 2.11.

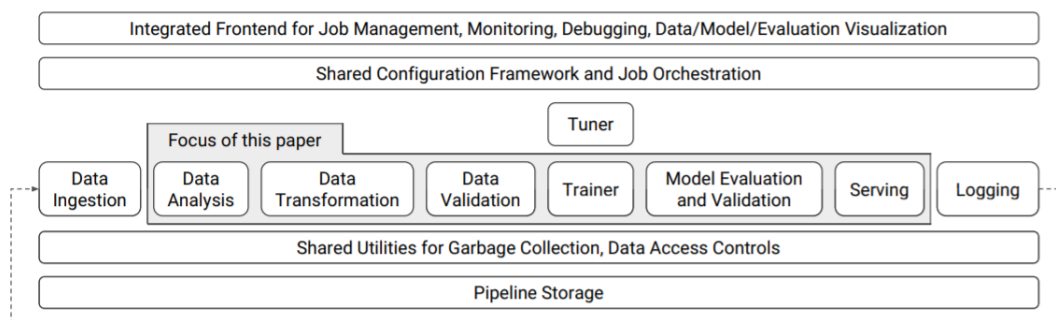


Figure 2.11. High-level component overview of a machine learning platform. Source: [30].



Min et al. develop a platform for multi-tenant model serving for acoustic and vision models in an MLOps architecture for distributed edge devices [31]. In this work, Min et al. outline the importance of minimizing redundancies by maintaining singular pipelines for multi-tenant model serving and employing a modular approach to increase the platform’s flexibility [31]. Min et al. also explore the trade-offs between performance latency and system attributes such as the number of models serving and batch serving in that latency is increased as the computation requirements increase in a system. Latency can be reduced and throughput maximized by using methods such as adaptive scheduling, feature caching, shared data operations, and system-aware model containers [31].

## **2.6 Machine Learning in Acoustics**

Underwater soundscapes in the context of this work refers to an ocean environment, where a plethora of biological and man-made sound sources are present. ML has been increasingly used for acoustic applications, and can achieve higher performance compared to traditional signal processing methods [32]. ML in the underwater, acoustic domain has applications such as source localization to identify the origin of the noise or acoustic scene classification to identify the source entity [32].

Most ML research in underwater acoustic environments is concerned with identifying biological sounds rather than man-made sounds such as those originating from ships or submarines [33], [34], [6]. This is likely due to the lack of available labeled data for research [6]. Existing ML research on classifying acoustic signals of man-made objects such as ships consist of relatively small datasets, whereas large datasets are typically required for acoustic classification tasks [32], [6]. Zak used a dataset containing only five ships [35], Santos-Domínguez et al. used two hours of recorded acoustic signals [36], Niu et al. used 30 minute recordings of three ships [37], and Berg et al [38] and Neilsen et al. [39] needed to synthetically generate their own examples to make up for the lack of data [6].

### **2.6.1 Underwater Sounds and Environmental Impacts**

Ship sounds are created and affected by a wide variety of factors in the ocean. The sources of ship noise can vary, as a confluence of sound from machinery, propeller cavitation, propeller shafts, and reduction gears all result in underwater noise [3], [6]. The resulting

noise from a ship's machinery creates an acoustic signature that can be detected by sonar sensors [3], [6]. In addition to the diverse range of noise sources, environmental effects such as temperature, salinity, and pressure, and physical interference such as biological objects or topography can change the environment through which sound propagates [3], [6]. Temperature, salinity, and pressure in particular change throughout the seasons of the year, and can affect the distance and manner in which sound propagates [6]. These environmental changes can result in virtual concept drift, which degrades model performance over time.

## **2.7 Significance of Applying Machine Learning Operations to Acoustic Deep Learning**

Due to the dynamic nature of underwater environments, input values to a NN can change due to shifting class distributions, varying classification schemes, and environmental impacts resulting in real and virtual concept drift. Due to increasing DoD investment in AI/ML, as these capabilities and the amount of available data improve and increase, analysts will become more reliant upon deployed ML models to assist in classifying acoustic scenes [5]. The concept drift inherent in acoustic environments creates a concern for significant technical debt in acoustic ML systems in the DoD as models will require ongoing maintenance and updating throughout their lifecycle. When considering the fact that these models will be distributed across various platforms, environments, sensors, and organizations, the risk of compounding technical debt increases significantly. In order to account for these risks, this work attempts to explore the impact of concept drift in a production, live-inference acoustic ML system, and apply the principles outlined by Sculley et al. through employing a basic MLOps architecture and framework [24].

THIS PAGE INTENTIONALLY LEFT BLANK

---

## CHAPTER 3: Methodology

---

### 3.1 Dataset and Processing Pipeline

#### 3.1.1 Vector Sensor

The acoustic sensor used to collect and stream the data in this work is the GeoSpectrum M20-105 vector sensor from GeoSpectrum Technologies Inc [40]. The vector sensor is comprised of 4 channels and records frequencies up to 3kHz [40]. The first channel measures acoustic pressure, and the other 3 are directional particle velocity fields that can assist in determining the bearing of acoustic data [40]. The M20-105 vector sensor is located at the Monterey Accelerated Research System (MARS) observatory at a latitude and longitude of 36.712465, -122.187548, 891 meters below the surface of the ocean in the Monterey Bay National Marine Sanctuary in California [41], [42].

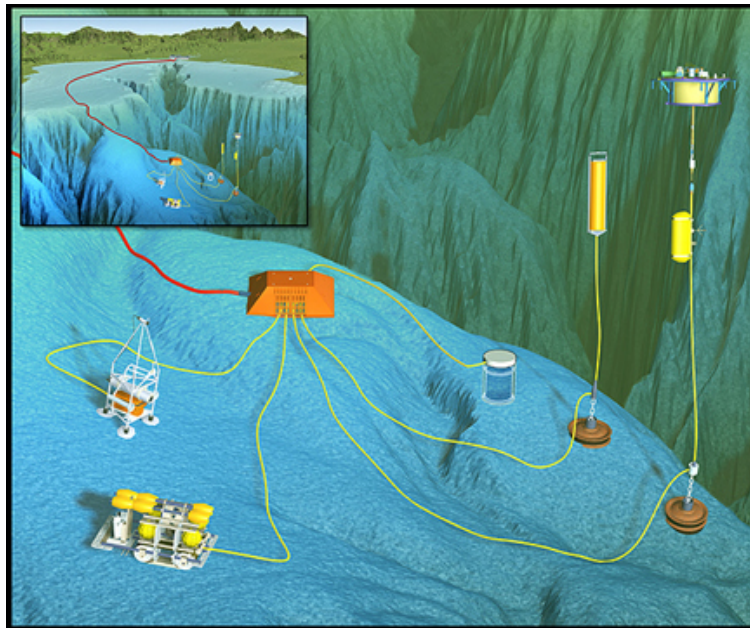


Figure 3.1. The MARS sensor is located 891 meters below the surface of Monterey Bay and connected to the shore via a 52km cable. Source: [43].

### 3.1.2 Physical Architecture and Storage

Recordings from the MARS sensor are saved in a .wav format every 30 minutes to a server located at the Naval Postgraduate School (NPS) via an Rsync process. These .wav files are then indexed every 5 minutes into a MySQL database to support data query packages developed by Dr. Paul Leary in [44] that streamline work with the raw acoustic data. The data query packages directly return raw acoustic data in response to queries defining start and end times, obscuring the intermediate layer of working with .wav files [44]. As depicted in Figure 3.2, once the acoustic data is stored on the server and indexed by the database, it is available for access via the data query tools. The MLOps platform then uses the available data for visualization, live inference, and generation of new datasets.

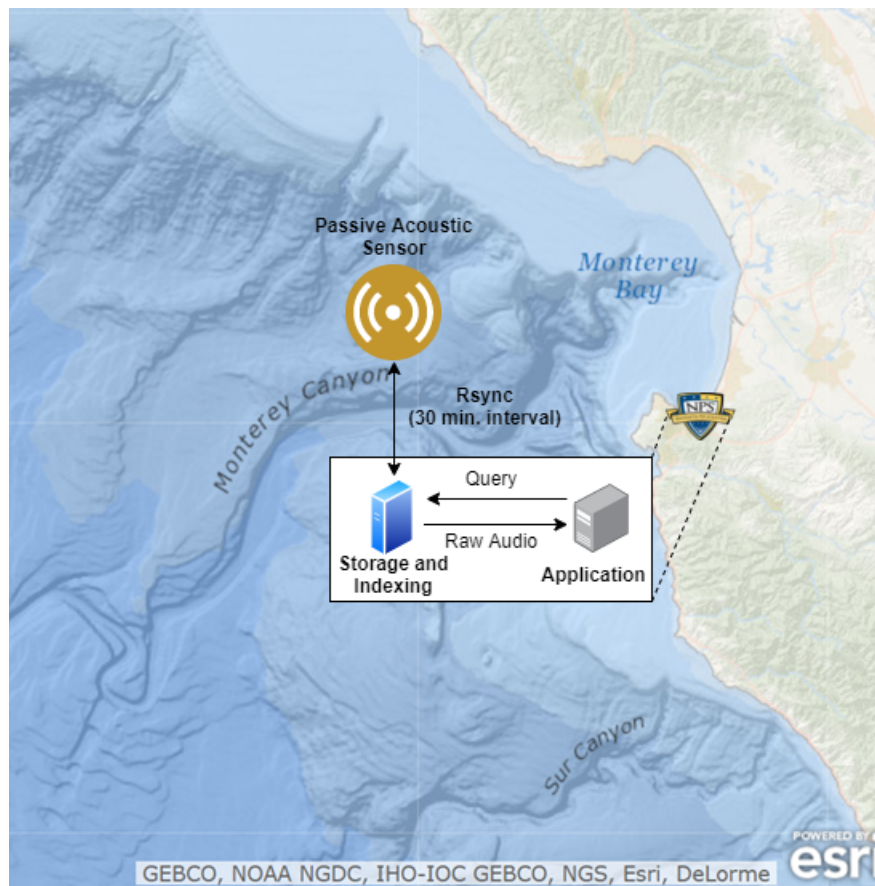


Figure 3.2. Physical architecture of the acoustic deep learning platform.

### 3.1.3 Data Preparation and Processing

#### Acoustic Processing

The acoustic data is used for two purposes. First, it is used to perform live inference inside the web application. Second, it is used to create preprocessed datasets for improving or retraining models and developing new models. As recommended by [31], the platform employs shared data operations in which all developed models, the live inference platform, and the pipeline for storing preprocessed data for research use the same pipeline code. Figure 3.3 displays the pipeline to prep the data, which is adapted from the original pipeline developed by Pfau in [3].

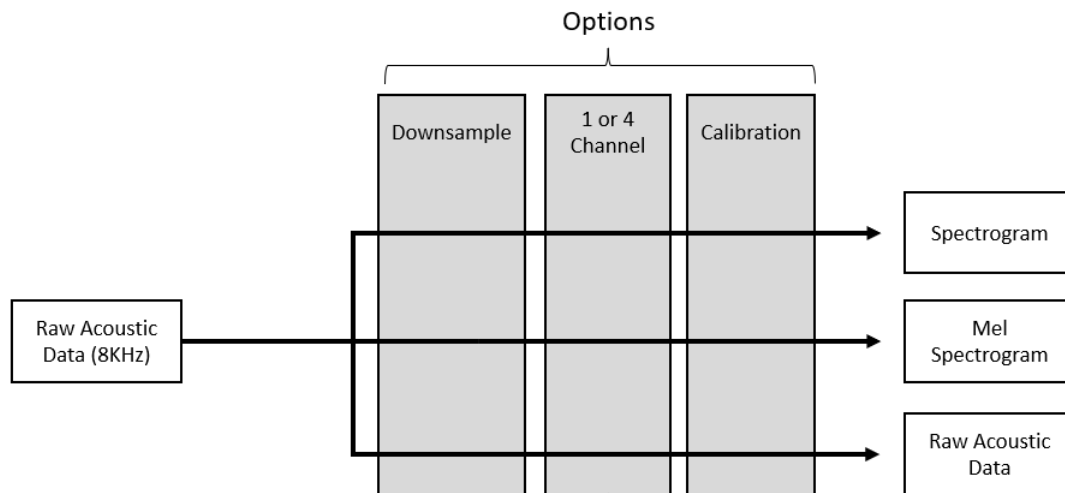


Figure 3.3. Acoustic data preprocessing options used to develop new models and apply to live-stream acoustic data for deployed models.

First, the acoustic signal is downsampled from 8KHz by a configurable amount to decrease the number of parameters in the machine learning models; in this work the signal is downsampled to 4KHz [3]. Downsampling decreases the complexity of the model and necessary computing power by decreasing the number of trainable parameters [3]. After downsampling, users can opt to retain all four channels or only the first acoustic pressure channel. A low pass calibration filter can then be optionally applied to filter noise from the acoustic data. This work only uses uncalibrated data and retains all four channels during preprocessing.

After the optional calibration filter has been applied, the acoustic signal can be stored in a raw data format, as a spectrogram created using a Short Time Fourier Transform (STFT), or a mel-log spectrogram [3]. Mel-log spectrograms are commonly used in deep learning acoustics research and are computed by applying a mel-filterbank over the magnitude of the complex output of the STFT [7], [3], [6]. Mel-log spectrograms are designed to emulate the non-linear scale of human hearing by better discriminating lower frequencies [3]. The mel-log scale is logarithmic over 1000Hz and linear under 1000Hz [3]. This work solely uses data in a mel-log spectrogram format. To perform this processing, an STFT is calculated for 30-second time periods with a 500ms frame size and 125ms frame hop using a Hann window function [3]. A 128 band mel-filterbank is run on the STFT magnitude and then a log-compression is computed to produce the mel-log spectrogram as depicted in Figure 3.4 [3]. Once the acoustic data is formatted as 30-second mel-log spectrograms, it is ready to be merged with Automatic Identification System (AIS) true labels.

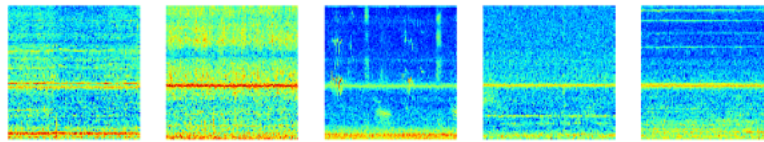


Figure 3.4. Mel-log spectrograms containing frequency on the y axis and time on the x axis. Pixel color indicates sound intensity, with red being the highest levels and blue as the lowest. Source: [7].

### **Automatic Identification System Processing**

After the acoustic data is processed, true labels are applied to each 30-second period. This research uses AIS to determine true labels for the recorded time periods and processes AIS broadcasts in a manner adapted from Pfau’s work in [3]. Individual AIS positions are recorded, each position’s distance and bearing from the acoustic sensor are calculated, and then metadata, if available, for each ship is queried from VesselFinder to ensure the proper ship designation and sizes are used [45], [3]. The AIS data is then processed into an optimized format to apply true labels to the acoustic data, as shown through the example in Table 3.1. The formatting records the time each individual ship, identified by their unique Maritime Mobile Service Identity (MMSI), entered and exited a defined radius from the acoustic sensor. This work uses 20km as the radius to define the entry and exit times to

ensure detection by the acoustic sensor in a majority of conditions.

Table 3.1. AIS formatting process.

| MMSI      | Designator | Latitude  | Longitude   | Time                | Sensor Distance (km) |
|-----------|------------|-----------|-------------|---------------------|----------------------|
| 123456789 | Tanker     | 36.88037  | -122.159342 | 08/11/2021 05:30:14 | 18.802               |
| 123456789 | Tanker     | 36.88037  | -122.159392 | 08/11/2021 05:31:32 | 18.801               |
| 123456789 | Tanker     | 36.880378 | -122.159377 | 08/11/2021 05:33:08 | 18.802               |
| ...       | ...        | ...       | ...         | ...                 | ...                  |

↓

| MMSI      | Designator | Start Time          | End Time            | Radius (km) |
|-----------|------------|---------------------|---------------------|-------------|
| 123456789 | Tanker     | 08/11/2021 05:25:51 | 08/11/2021 06:44:32 | 20          |

Class labels are applied to the formatted AIS data to group each ship into one of four categories based on their designations in accordance with Table 3.2. Each designation is based on the ship’s approximate size. In this work, all time periods where no ships were present within 40km are labeled with Class E which means there is no ship present.

Table 3.2. Ship classes. Source: [21].

| Class | Ship Designators                                    |
|-------|---|
| A     | Fishing Vessel, Tug, Towing Vessel                  |
| B     | Pleasure Craft, Sailboat, Pilot                     |
| C     | Passenger ship, Cruise Ship                         |
| D     | Tanker, Container Ship, Military Ship, Bulk Carrier |
| E     | No ship present, background noise                   |



## Data Labeling

Once individual ships have been assigned to a class, time periods of recorded acoustic data are labeled with the ship classes that were present within a 20km radius during that time period, as depicted in Table 3.3.

Table 3.3. True label assignment process.

| MMSI      | Designator     | Start Time          | End Time            | Radius (km) | Ship Class |
|-----------|----------------|---------------------|---------------------|-------------|------------|
| 123456789 | Tanker         | 08/11/2021 05:25:51 | 08/11/2021 06:44:32 | 20          | D          |
| 194729432 | Sailboat       | 08/11/2021 06:31:16 | 08/11/2021 06:57:09 | 20          | B          |
| 493872943 | Fishing Vessel | 08/11/2021 06:22:57 | 08/11/2021 08:19:46 | 20          | A          |



| Time Period | 08/11/2021 06:33:00 - 08/11/2021 06:33:30 |                 |     |
|-------------|---|-----------------|-----|
| Channel 1   | -8.27551354e-04                           | -5.69582451e-04 | ... |
| Channel 2   | 6.91413879e-05                            | 1.30891800e-04  | ... |
| Channel 3   | -1.17540825e-04                           | -2.09689606e-04 | ... |
| Channel 4   | -2.27928627e-04                           | -1.00255478e-04 | ... |
| True Label  | Class A, Class B, Class D                 |                 |     |

Once the dataset has been processed and formatted, it is available for use in the ML components of the architecture. The dataset processing pipeline is used for two purposes, preprocessing data for live inferencing in the production architecture and creating large preprocessed datasets to train or retrain models for research and development. These datasets are stored in a .tfrecords format in order to optimize performance during model training since the ML architecture is based on the Tensorflow library [46].

## 3.2 Model Architecture

The model used in this work is a custom CNN architecture developed in [3], adapted as a BNN in [7] and for multilabel classification by [6], as shown in Figure 3.2. The

custom architecture employs a rectangular 10x5 kernel size for each convolutional layer, as opposed to the square kernels typically used for image classification in order to vary the time and frequency dimensions of the input spectrograms [3]. The 10x5 kernel size was developed through a hyperparameter search of kernel ratios of 1:1, 2:1, 3:1, and 4:1 that found a 2:1 ratio performed the best [3]. The mel spectrograms are normalized through a batch normalization layer at the input, then passed through four convolutional blocks, each containing two convolutional layers [3]. The first convolutional block contains 16 filters, and each subsequent block contains 16 more filters than the previous, as suggested by the findings in [17], [3]. A batch normalization layer is employed after each convolutional layer, and a max pooling layer at the end of each block halves the number of inputs to the next block [3]. The architecture employs the ReLU activation function, a dropout layer, and a final classification layer that employs a softmax or sigmoid activation function depending on whether the model is multiclass or multilabel [3].

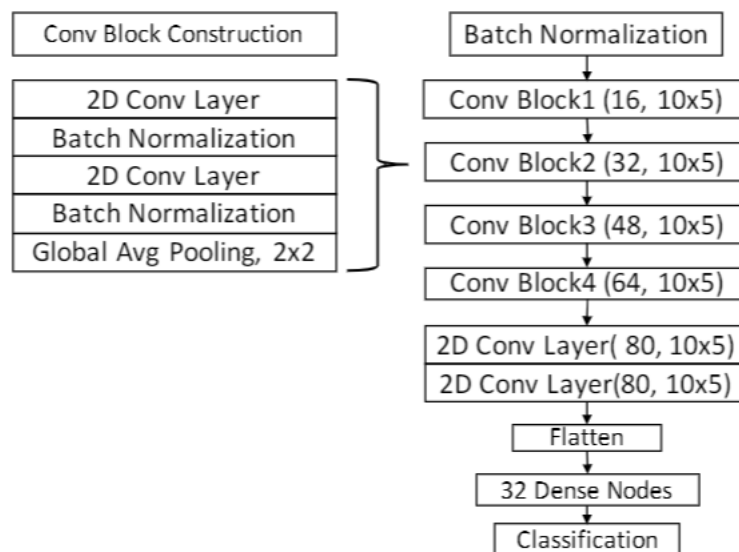


Figure 3.5. Custom CNN Model Architecture: Each block is described by (number of filters, filter shape). Source: [3].

### 3.3 Performance Metrics

Standard metrics used to calculate performance of ML models include accuracy ( $Acc$ ), precision ( $Prec$ ), recall ( $Rec$ ),  $F^1$ , and area under the Receiver Operating Characteristics (ROC)

curve [11]. The methods employed to calculate these metrics differ between multiclass and multilabel models. This work employs the same calculations and code to compute performance metrics as in [7] and [6] outlined in Equations 3.1, 3.2, 3.3, and 3.4.

Equation 3.1 displays the methods used to calculate performance metrics for the multiclass models used in this work. TP and TN represent the true-positives and true-negatives from the model’s predictions, and FP and FN similarly represent the false-positives and false-negatives [11].

$$\begin{aligned}
 Acc &= \frac{TP + TF}{TP + TF + FP + FN} \\
 Prec &= \frac{TP}{TP + FP} \\
 Rec/TPR &= \frac{TP}{TP + FN} \\
 F^1 &= 2 * \frac{Prec * Rec}{Prec + Rec}
 \end{aligned}
 \tag{3.1}$$

Similar metrics are used in multilabel performance evaluation; however, the calculations require some adaptations to account for the comparison of multiple labels. Performance can be calculated for each possible output class, and an average taken of these results through a method known as macro-averaging [47]. Micro-averaging is another method by which all the classes are aggregated first and then the average is calculated [47].

Metrics can be calculated by each class or by each example, but to do so, the number of true and false positives and negatives are required [47]. Equation 3.2 depicts how to calculate these for a dataset  $D^*$  with size  $M$ ,  $D^* = (x_i^*, Y_i)_{i=1}^M$  where  $x_i$  is the  $i$ -th feature vector and  $Y_i$  are the true labels for each feature vector [6].

$$\begin{aligned}
TP_j &= \{|x_i^*|y_j \in Y_i \wedge y_j \in f(x_i^*), 1 \leq i \leq M\} \\
FP_j &= \{|x_i^*|y_j \notin Y_i \wedge y_j \in f(x_i^*), 1 \leq i \leq M\} \\
TN_j &= \{|x_i^*|y_j \notin Y_i \wedge y_j \notin f(x_i^*), 1 \leq i \leq M\} \\
FN_j &= \{|x_i^*|y_j \in Y_i \wedge y_j \notin f(x_i^*), 1 \leq i \leq M\}
\end{aligned} \tag{3.2}$$

$TP$ ,  $FP$ ,  $TN$ , and  $FN$  are calculated where  $f(x_i^*)$  is the set of predictions from model  $f$  on  $x_i^*$  [6]. Accuracy, precision, recall, and  $F^1$  can be calculated with the outputs from Equation 3.2 through either micro or macro-averaging in Equation 3.3, where  $B \in Acc, Prec, Rec, F^1$  and  $c$  is the number of classes [47].

$$\begin{aligned}
B_{micro} &= B \left( \sum_{j=1}^c TP_j, \sum_{j=1}^c FP_j, \sum_{j=1}^c TN_j, \sum_{j=1}^c FN_j \right) \\
B_{macro} &= \frac{1}{c} \sum_{j=1}^c B (TP_j, FP_j, TN_j, FN_j)
\end{aligned} \tag{3.3}$$

To compute instance-based performance metrics, an average of the total can be taken after each example is evaluated individually [6]. Equation 3.4 depicts these calculations for each performance metric [6].

$$\begin{aligned}
Acc_{subset} &= \frac{1}{M} \sum_{i=1}^M [|f(x_i^*) = Y_i|] \\
Prec_{inst} &= \frac{1}{M} \sum_{i=1}^M \frac{|Y_i \cap f(x_i^*)|}{|f(x_i^*)|} \\
Rec_{inst}(m) &= \frac{1}{M} \sum_{i=1}^M \frac{|Y_i \cap f(x_i^*)|}{|Y_i|} \\
F_{inst}^1 &= \frac{2 \cdot Prec_{inst} \cdot Rec_{inst}}{Prec_{inst} + Rec_{inst}}
\end{aligned} \tag{3.4}$$

## **3.4 Experiment Methodology**

### **3.4.1 Tools**

The tools used to build the MLOps architecture and conduct experiments include code written in Python, JavaScript, HTML, and CSS. The Python Flask web framework is used to build the application in accordance with the methodology described in [48] and [49], and Bokeh is the Python library used to build the embedded visualizations in the Flask application [50]. The ML and data pipeline code is written using Tensorflow [46]. The runtime environment consists of multiple Docker containers with Linux-based images that are managed and networked using Docker-Compose [51]. Databases used in the architecture include PostgreSQL and MySQL [52], [53].

### **3.4.2 Dataset Characteristics**

The static dataset generated for this work's experiments contains 967 days of recorded acoustic data, totalling 2.7 TB. The acoustic data was collected from January 2019 until August 2021 and labeled with 1,438 unique ships from AIS broadcasts. The data in the live stream architecture and the static dataset are configured for use with both one and four channel models, uncalibrated, and formatted as mel-log spectrograms.

### **3.4.3 Approach**

A production, MLOps architecture is proposed and implemented in this work to manage ML model lifecycles and data pipelines. The architecture is then used to deploy production ML models from work in [3] and [7]. for live inference. Concept drift can manifest in this system through changing relations between ship class designations and their emanated acoustic signatures or through environmental factors such as temperature, salinity, and pressure. To identify how this drift may affect model performance, this work explores the performance over time of a multiclass, four channel custom CNN and MC Dropout BNN using approaches from [3] and [7]. This work further uses the MLOps architecture to train multilabel MC Dropout BNNs on varying time periods to test them on data increasingly distant in time from the training dataset to assess performance degradation rates. This work uses the findings from these experiments to explore methods to optimize model performance in a production architecture and identify possible factors that contribute to concept drift.

All models were trained to between 600 and 1000 epochs until convergence using NVIDIA Quadro RTX 8000, Python 3.6, Tensorflow 2.2, and Tensorflow probability 0.11.

THIS PAGE INTENTIONALLY LEFT BLANK

# CHAPTER 4: Results

## 4.1 Machine Learning Operations Architecture

This work proposes a system-level production architecture and MLOps process for acoustic deep learning systems. Figure 4.1 outlines the interaction between the various components of this purpose-built MLOps architecture for ML with live acoustic signals. The holistic process consists of elements for researching and developing new ML models, retraining active learning models, deploying the models for live inference on streaming acoustic data, automated true labeling of the acoustic data using an AIS stream, and a user interface for managing the MLOps process. Each component runs in a containerized environment to maintain a modular architecture as recommended in [24] and to be hardware agnostic.

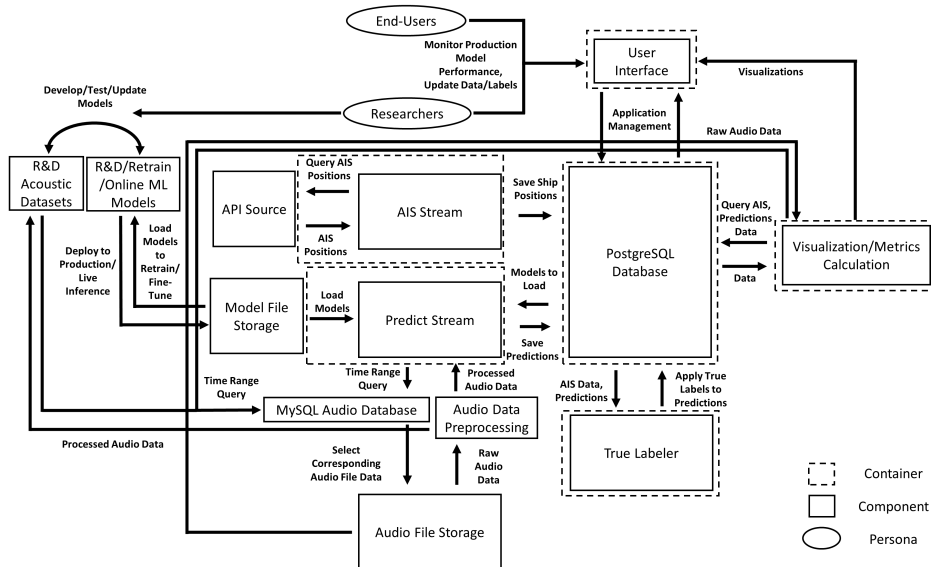


Figure 4.1. An architecture for an acoustic deep learning MLOps process for production of live-inference models into an application, research and development of new models, CI/CD and CE of deployed machine-learning models, and developing new datasets for continuous research.



### **4.1.1 Model Lifecycle**

In accordance with the MLOps lifecycle outlined in Figure 2.8, the proposed system outlines methods for accomplishing all phases of model management. Static, labeled datasets created from the pipeline described in Section 3.1 are available for researching novel models, hyper-parameter tuning for increased performance, and retraining and updating previously deployed models. The operations using these datasets are done separately from the live prediction architecture, which primarily interacts with the live data streams. However, the static datasets can be continually updated from the live data streams. Trained models can be deployed into production after training on a static dataset via the user interface for managing production ML models as shown in Section 4.1.2. Continuous training, if desired, can also be done using the static datasets to perform transfer learning to update and redeploy models. While running in the application, functionality is available for both prediction serving and continuous monitoring of performance, as outlined in Section 4.1.2.

### **4.1.2 User Interface and Machine Learning Operations Functionality**

The web application maintains a user interface for various MLOps functions within the architecture through four modules as depicted in Figure 4.2. The first module is the live inference display where analysts can view the live-stream of model predictions, AIS ship tracks, and acoustic spectrograms. The second module is for monitoring the performance of the production ML models. The third module's function is to manage the deployed ML models. The fourth module serves as an interface to edit and curate the data used for the platform.

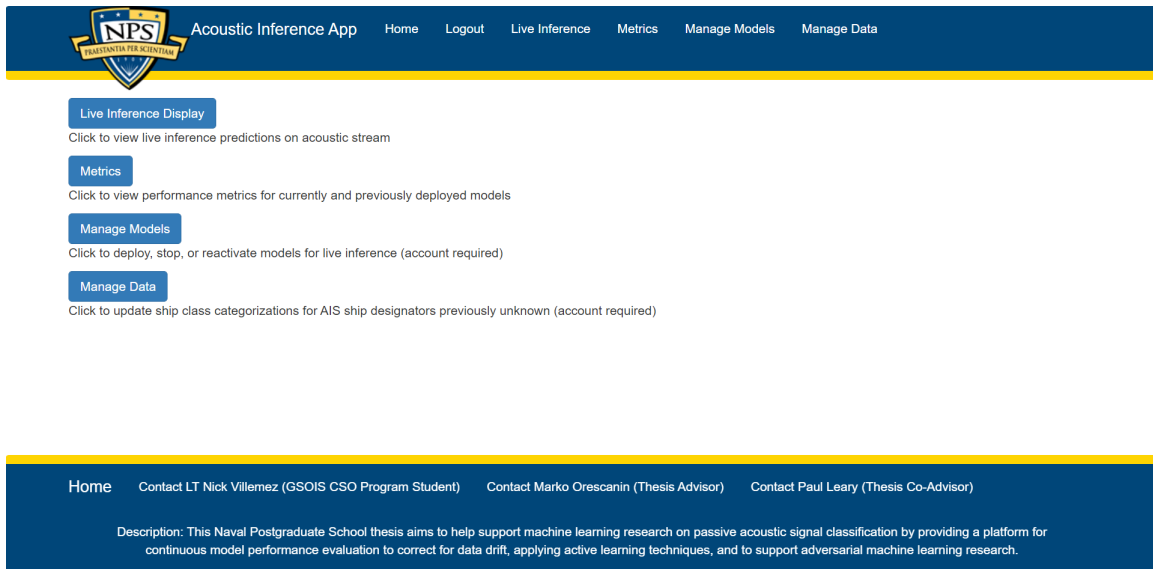
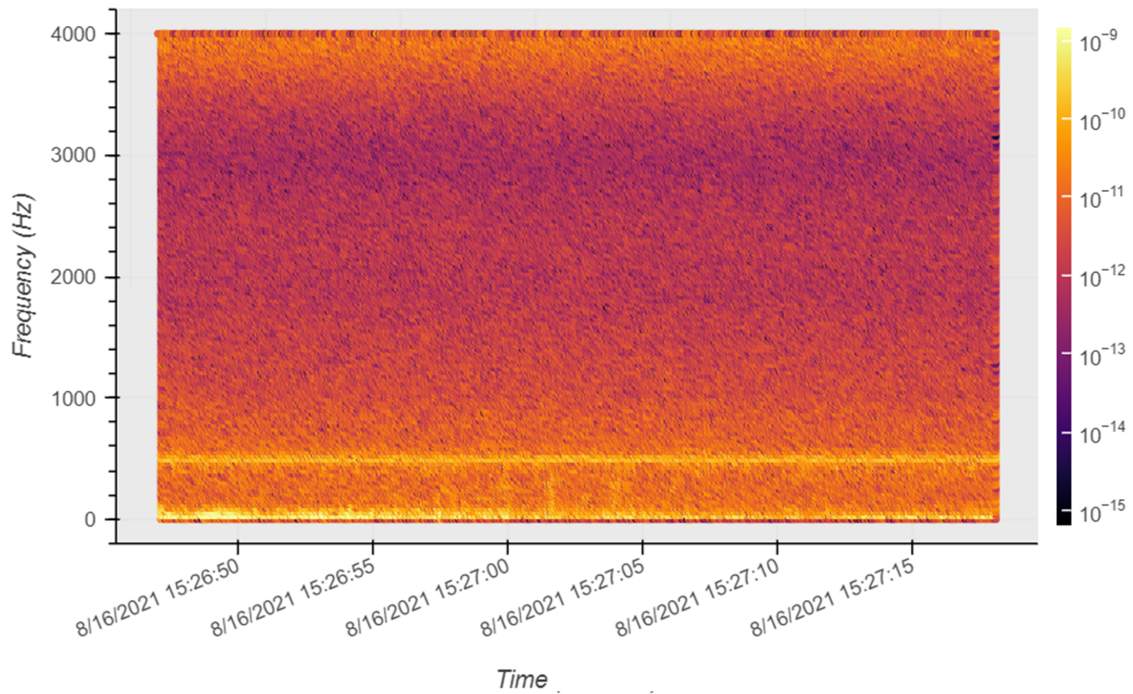


Figure 4.2. Web application home page.

### Live Prediction Module

The live prediction display shows a live scrolling spectrogram to depict the acoustic data. Below the spectrogram are the corresponding predicted classes for each deployed model combined with metadata about the model's attributes, as depicted in Figure 4.3. A multi-tenant model display provides insight to end-users for higher confidence when multiple models output the same predicted class. Conversely, if multiple models display a variety of predicted classes, this indicates uncertainty in the predictions. Displaying multiple model predictions serves to loosely emulate an ensemble process to increase trust and interpretability of the system. Additionally, multi-tenant model serving is beneficial because this architecture enables model deployments that serve different functions and can make different types of predictions, as noted by [31]. Additionally, multi-tenancy better supports research and development to evaluate different models in production alongside one another.



| Model ID | Model Name                   | Model Type  | Channels | Model Prediction |
|----------|------------------------------|-------------|----------|------------------|
| 11       | Custom CNN v1                | multi_class | 1        | Class C          |
| 12       | Custom CNN v1                | multi_class | 4        | Class C          |
| 13       | Deterministic Dropout BNN v1 | multi_class | 1        | Class B          |
| 14       | Probabilistic Dropout BNN v1 | multi_class | 1        | Class B          |
| 15       | Probabilistic Dropout BNN v1 | multi_class | 4        | Class E          |
| 16       | Probabilistic Flipout BNN v1 | multi_class | 1        | Class B          |

Figure 4.3. Live spectrogram and model predictions.

Similarly, for the probabilistic BNN models, users can select an uncertainty metric to display, as depicted by Figure 4.4. The uncertainty metric provides insight into which models are relatively more certain in their prediction. The primary uncertainty measures available include aleatoric uncertainty, epistemic uncertainty, entropy, and variance as implemented by [7].

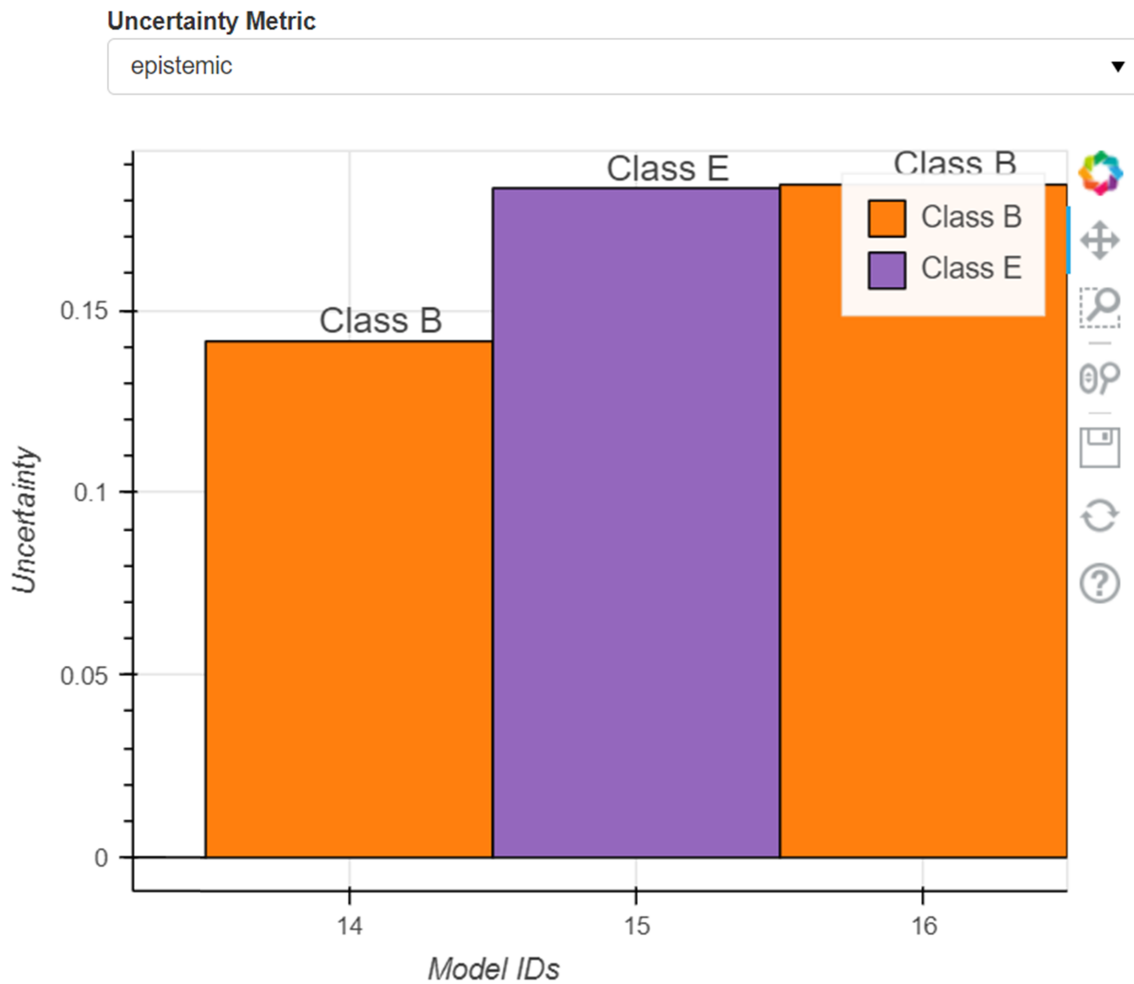


Figure 4.4. Live uncertainty graph for BNN models.

In order to verify the actual presence of predicted ship classes, the user interface also includes a visualization of an AIS stream from the U.S. Department of Transportation (DOT) SeaVision Application Programming Interface (API) [54]. As depicted in Figure 4.5, the display visualizes the sea bottom contours, ship positions relative to the acoustic sensor, as well as other information contained in the AIS broadcast on mouse-over of the ship's positions. The addition of an AIS visualization enables correlation, fusion, and characterization of model predictions in a single user interface.

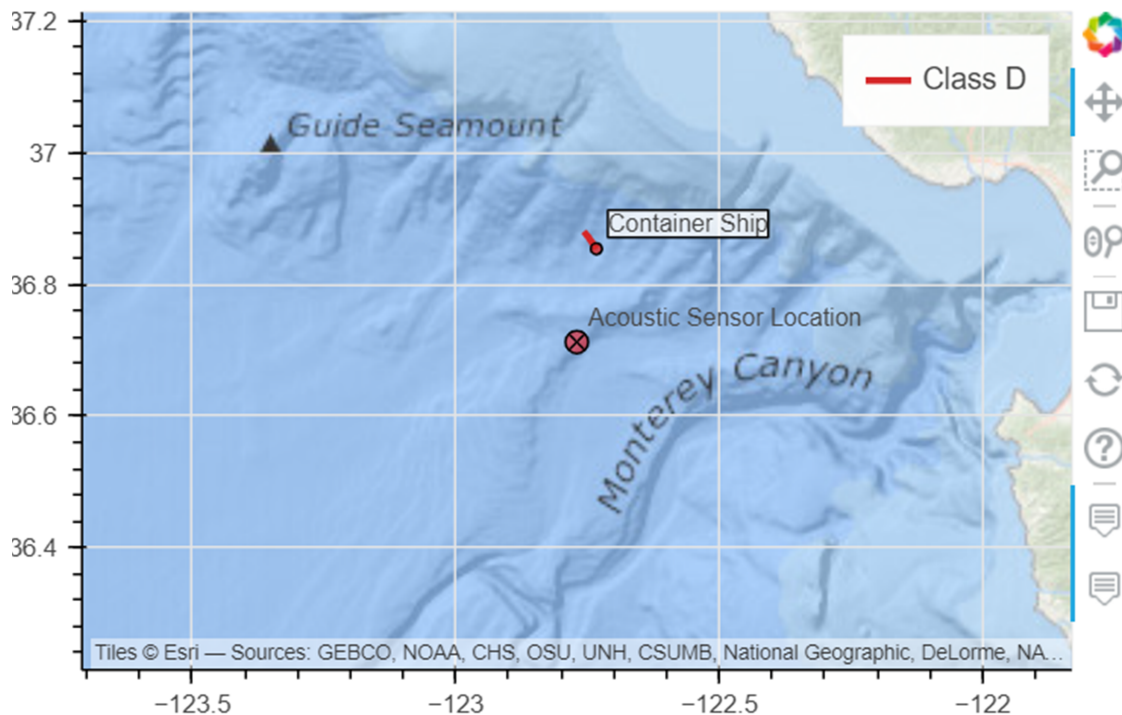


Figure 4.5. Live AIS stream display of ship positions.

### Model Performance Module

The model performance module provides insight to researchers and end users about the relative performance of each ML model. Figure 4.6 calculates the Accuracy, Precision, Recall, and F1 scores for the entire lifetime of while each model was active.

| Model ID | Active (T/F) | Model Name                   | Model Type  | Channels | Accuracy           | Precision (Weighted) | Recall (Weighted)  | F1 (Weighted)    |
|----------|--------------|------------------------------|-------------|----------|--------------------|----------------------|--------------------|------------------|
| 11       | true         | Custom CNN v1                | multi_class | 1        | 0.506586976249335  | 0.5179091183275119   | 0.506586976249335  | 0.51208863719581 |
| 12       | true         | Custom CNN v1                | multi_class | 4        | 0.0625528053321650 | 0.2052549857019330   | 0.0625528053321650 | 0.08380707149200 |
| 13       | true         | Deterministic Dropout BNN v1 | multi_class | 1        | 0.2302214693519041 | 0.4292281409951544   | 0.2302214693519041 | 0.29026913182655 |
| 14       | true         | Probabilistic Dropout BNN v1 | multi_class | 1        | 0.2328375106135410 | 0.5956477687154099   | 0.2328375106135410 | 0.32866507188244 |
| 15       | true         | Probabilistic Dropout BNN v1 | multi_class | 4        | 0.0399698103713950 | 0.4334359419287680   | 0.0399698103713950 | 0.01288494211419 |
| 16       | true         | Probabilistic Flipout BNN v1 | multi_class | 1        | 0.1241862951665140 | 0.425999205588404    | 0.1241862951665140 | 0.18700267679034 |

Figure 4.6. Calculation of total performance for production ML models.

The table in Figure 4.7 offers a more in-depth view of each model's performance by displaying the metrics for each possible ship class prediction and their micro and macro-

averages. The classification report metrics allows users to identify which models perform better in regard to particular ship classes.

| #  | Class/Measure | Precision            | Recall              | F1 Score            | Support | Model ID |
|----|---------------|----------------------|---------------------|---------------------|---------|----------|
| 0  | Class A       | 0                    | 0                   | 0                   | 1426    | 11       |
| 1  | Class B       | 0                    | 0                   | 0                   | 215     | 11       |
| 2  | Class C       | 0                    | 0                   | 0                   | 0       | 11       |
| 3  | Class D       | 0.5271312243280248   | 0.5316906527111499  | 0.5294011217658767  | 13758   | 11       |
| 4  | Class E       | 0.561574484242501    | 0.5359342915811088  | 0.5484548825710754  | 16558   | 11       |
| 5  | micro avg     | 0.506586976249335    | 0.506586976249335   | 0.506586976249335   | 31957   | 11       |
| 6  | macro avg     | 0.21774114171410516  | 0.21352498885845175 | 0.21557120086739046 | 31957   | 11       |
| 7  | weighted avg  | 0.5179091183275119   | 0.506586976249335   | 0.5120886371958193  | 31957   | 11       |
| 8  | samples avg   | 0.506586976249335    | 0.506586976249335   | 0.506586976249335   | 31957   | 11       |
| 9  | Class A       | 0.049475993175725076 | 0.2847124824684432  | 0.08430232558139535 | 1426    | 12       |
| 10 | Class B       | 0                    | 0                   | 0                   | 215     | 12       |
| 11 | Class C       | 0                    | 0                   | 0                   | 0       | 12       |
| 12 | Class D       | 0                    | 0                   | 0                   | 13758   | 12       |
| 13 | Class E       | 0.3918819188191882   | 0.09620727140959053 | 0.15448770789894778 | 16558   | 12       |
| 14 | micro avg     | 0.06255280533216509  | 0.06255280533216509 | 0.06255280533216509 | 31957   | 12       |

Figure 4.7. Calculation of classification reports for production ML models.

Concept drift is an additional concern in production ML architectures that will degrade model performance over time. Figures 4.6 and 4.7 will not show this performance degradation since they are total summary statistics, so a visualization of performance metrics over time is necessary to observe concept drift. Time domain visualization enables users to determine a performance threshold beyond which a model becomes unusable or necessary to replace or retrain. This component is significant to enable active-learning configurations, where a degradation beyond a certain point could initiate automatic retraining of the associated model. Time domain considerations also allow users to observe if models perform worse at certain times of year due to environmental changes. In these cases of concept drift, models could be adapted accordingly to account for specific periods of time. To enable this type of analysis, Figure 4.8 depicts model performance by each day.

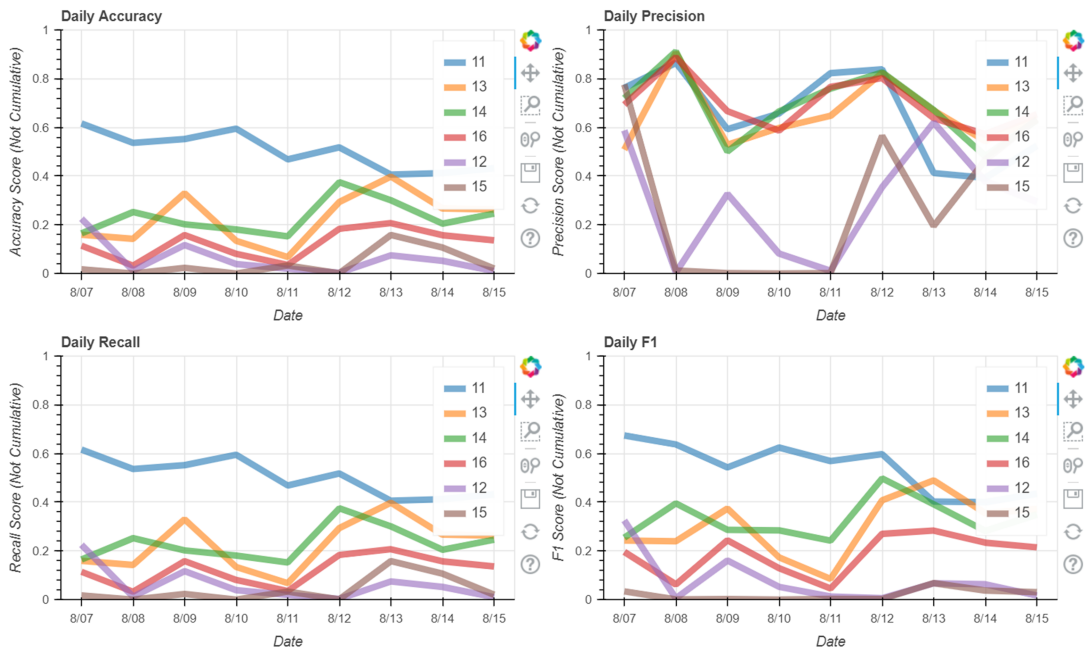


Figure 4.8. Calculation of daily model performance metrics.

## Model Management Module

To abstract the use of code in the deployment and management of ML models, the application offers a user interface to perform these functions. Figure 4.9 depicts the form that is used for the custom deployment of Tensorflow model checkpoints into production. The form includes the ability to upload the model checkpoint file in a .h5 format and a params.txt file containing metadata about how the model was trained, attributes of the model, and the input data types. Users must then specify a model name by which to refer to this deployment, whether the model is multiclass or multilabel, the number of input channels of acoustic data it is configured for, the formatting process for the data stream, the model architecture, and the accuracy score that the checkpoint achieved during training on the test set.

# Deploy a Model

Upload .h5 model checkpoint file

No file chosen

Upload corresponding params.txt. Please verify all information is correct, inference will not perform correctly otherwise (e.g. bnn=True)

No file chosen

Model Name (e.g. Joes Probabilistic Four Channel Model)

Model Type

Channels (1 or 4 channels)

Model Input (stft not supported)

Model Choice

Prediction Classes

Training Accuracy Score (Enter the score the model achieved during training, testing, and validation)

Figure 4.9. Web form to deploy new ML models into production.

In addition to deploying new ML models, the Model Management Module allows users to stop models in production or to reactivate models that have been previously stopped through the web form displayed in Figure 4.10.



## Stop a Model

Select a model to stop live inference for (Must be linked to your user account)

## Reactivate a Model

Select a model to reactivate (does not delete previous predictions, just halts new inference)

Figure 4.10. Form to stop active model predictions or to reactivate stopped models.

When the Model Management Module is used in concert with the Model Metrics Module, users can quickly determine which models need to be replaced, deactivate them, and deploy new models as a replacement. The user interface for managing multi-tenant model deployments results in less required code for researchers to move their models from development to production.

### Data Management Module

As AIS data is continually saved and each ship is assigned to a particular class, it is possible that the characteristics of each ship class change due to the shifting attributes of the ships that comprise that class. Changes in size for instance would result in changes in the general acoustic signatures associated with a particular class and degrade the performance of deployed ML models. In order to combat this form of real concept drift, the Data Management Module offers insight into the statistics of each ship size as they correlate to their assigned class. As an overall reference, the page first contains a table to show the

average dead weight, length, and beam for each ship class category as depicted by Figure 4.11.

| # | Ship Class | Average Dead Weight | Average Length     | Average Beam       |
|---|------------|---------------------|--------------------|--------------------|
| 0 | Class A    | 579.0576579715193   | 46.971040369289724 | 12.635239298001132 |
| 1 | Class B    | 166.8759302939029   | 29.3802705020438   | 9.3183784020871948 |
| 2 | Class C    | 78.79393223010244   | 40.2772524297347   | 11.029419490412398 |
| 3 | Class D    | 67796.34688161209   | 238.11847858942065 | 35.40268010075567  |

Figure 4.11. Table to display the average ship class sizes.

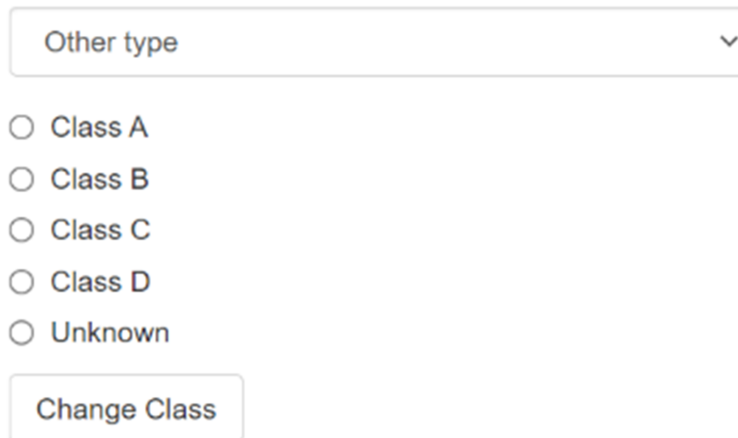
As additional ship designations are recorded that have not yet been assigned to a ship class, they are marked as having an “Unknown” class. Using the sizing information from Figure 4.11, users can determine which category a new designation belongs to and assign it via the form depicted in Figure 4.13. Due to limitations in the data from AIS broadcasts, some broad designations such as “Unknown” or “Other type” can not be assigned to a ship class and are discarded during the data labeling process.

| # | Ship Designation   | Average Dead Weight | Average Length     | Average Beam      |
|---|--------------------|---------------------|--------------------|-------------------|
| 0 | Other type         | NaN                 | 24.384615384615383 | 7.461538461538462 |
| 1 | Unknown            | NaN                 | 19.692307692307693 | 7.153846153846154 |
| 2 | HSC                | NaN                 | 12.75              | 5                 |
| 3 | Resolution 18 ship | NaN                 | 15                 | 6                 |
| 4 | -1                 | NaN                 | NaN                | NaN               |
| 5 | Local type         | NaN                 | 13                 | 10                |
| 6 | Port tender        | NaN                 | 10                 | 6                 |
| 7 | Cargo              | NaN                 | NaN                | NaN               |

Figure 4.12. Table to display the average size of each ship designation with an unassigned ship class.

# Assign Ship Class

Select a ship designation to update the class for.  
Some designations may need to remain Unknown due to vague descriptors



The form consists of a dropdown menu at the top with the text "Other type" and a downward arrow. Below the dropdown are five radio button options: "Class A", "Class B", "Class C", "Class D", and "Unknown". At the bottom of the form is a button labeled "Change Class".

Figure 4.13. Form to assign a class to uncategorized ship designations.

The attributes of ship designations already assigned to a ship class may also drift over time, so the information provided in the table of Figure 4.14 allows users to monitor the summary statistics of ship designations and correlate them with the summary statistics in Figure 4.11. In order to account for concept drift in the relation between the categorization of classes and the ship classes statistical attributes, the form depicted in Figure 4.15 provides a means to reassign class categories as necessary.

| #  | Ship Designation         | Average Dead Weigh | Average Length                        | Average Beam       | Ship Class |
|----|--------------------------|--------------------|---------------------------------------|--------------------|------------|
| 0  | Landings Craft           | NaN                | NaN                                   | NaN                | Class A    |
| 1  | Military ops             | NaN                | 40                                    | 14                 | Class A    |
| 2  | Fishing vessel           | NaN                | 14.74285714285714; 5.371428571428571  |                    | Class A    |
| 3  | Fishing Vessel           | NaN                | NaN                                   | NaN                | Class A    |
| 4  | Fishing Support Vess 718 |                    | 64                                    | 15                 | Class A    |
| 5  | Tug                      | 377.57142857142856 | 28.56818181818181                     | 9.954545454545455  | Class A    |
| 6  | Pusher Tug               | 711.625            | 48.5                                  | 13.1               | Class A    |
| 7  | Dredging or UW ops       | NaN                | 26.57142857142857; 11.14285714285714; |                    | Class A    |
| 8  | Towing vessel            | NaN                | 24                                    | 9.363636363636363  | Class A    |
| 9  | Crew Boat                | NaN                | NaN                                   | NaN                | Class A    |
| 10 | Buoy/Lighthouse Ves 350  |                    | 68                                    | 13                 | Class A    |
| 11 | Salvage Ship             | NaN                | NaN                                   | NaN                | Class A    |
| 12 | Research Vessel          | 1107               | 59                                    | 14.666666666666666 | Class A    |
| 13 | Anti-polution            | NaN                | 20                                    | 7                  | Class A    |
| 14 | Offshore Tug/Supply 1422 |                    | 59                                    | 12                 | Class A    |

Figure 4.14. User interface to display the average size of each ship designation.

## Change Ship Class

Select a ship designation to update the class for.

Some designations may need to remain Unknown due to vague descriptors

- Class A
- Class B
- Class C
- Class D
- Unknown

Figure 4.15. Form to change assigned ship classes.

The Live Prediction, Model Management, Data Management, and Model Performance modules in the application are key to enabling an MLOps architecture in which the continuous management of model life cycles is possible to optimize performance.

### **4.1.3 Databases**

Each module is dependent upon a database back-end to manage the overall architecture. This work also defines a schema outlined in Figure 4.16 that is used as the application's back-end database. The primary application uses a PostgreSQL object-relational database, which has a few differences compared to other types of databases in that it has a wide variety of available data types such as JavaScript Object Notation (JSON), which this work uses extensively, and can store more complex data than relational databases [52]. The MySQL database used in this work for the acoustic data querying tools is another database used for storing simpler data in a relational format [53].

The AIS table stores the raw data stream of ship tracks from the API. The Ship Classes table is used as a reference to assign each ship to a ship class. Metadata associated with each ship is saved in the MMSI table to decrease the number of web-scraping operations with VesselFinder [45]. During the true label assignment process, the data in the AIS table is iterated through and stored in a processed format in the AIS Times table, which is used to assign true labels to periods of acoustic data.

Each model and its metadata is stored in the Models table to track versions, data formats, and architectures of each model to use during the loading process and whether or not the model is currently active to conduct live-inferencing operations. As predictions are continuously made on 30-second time periods, they are stored in the Predictions table, which contains a JSON-formatted set of predictions from each model.

The MySQL database developed by Dr. Paul Leary in [44] allows for development of tools to seamlessly query raw audio data without requiring the manipulation of .wav files. Each .wav file is indexed and the metadata is stored in the Audio and Orientation tables to support the query tools discussed in Section 3.1.2.

| PostgreSQL              |                 |                           |                 |
|-------------------------|-----------------|---------------------------|-----------------|
| AIS                     |                 | AIS Times                 |                 |
| Column Name             | Data Type       | Column Name               | Data Type       |
| id                      | serial          | id                        | serial          |
| mmsi                    | int             | mmsi                      | int             |
| imo_number              | int             | radius                    | int             |
| ship_name               | varchar         | start_time                | timestamp (utc) |
| callsign                | varchar         | end_time                  | timestamp (utc) |
| cargo                   | varchar         | closest_point_of_approach | float           |
| heading                 | float           | cpa_time                  | timestamp (utc) |
| nav_status              | varchar         | designation               | varchar         |
| speed_over_ground       | float           | ship_class                | varchar         |
| latitude                | float           | record_timestamp          | timestamp (utc) |
| longitude               | float           |                           |                 |
| time_of_fix             | timestamp (utc) | Ship Classes              |                 |
| dist_from_sensor        | float           | Column Name               | Data Type       |
| dead_weight             | float           | id                        | serial          |
| length                  | int             | designation               | varchar         |
| beam                    | int             | ship_class                | varchar         |
| designation             | varchar         | record_timestamp          | timestamp (utc) |
| ship_class              | varchar         |                           |                 |
| bearing_from_sensor     | float           | MMSI                      |                 |
| record_timestamp        | timestamp (utc) | Column Name               | Data Type       |
|                         |                 | id                        | serial          |
| Models                  |                 | mmsi                      | int             |
| Column Name             | Data Type       | dead_weight               | float           |
| id                      | serial          | length                    | int             |
| file_name               | varchar         | beam                      | int             |
| model_name              | varchar         | designation               | varchar         |
| model_type              | varchar         | record_timestamp          | timestamp (utc) |
| channels                | int             |                           |                 |
| model_input             | varchar         | Predictions               |                 |
| model_input             | varchar         | Column Name               | Data Type       |
| model_choice            | varchar         | id                        | serial          |
| prediction_classes      | varchar         | start_time                | timestamp (utc) |
| params                  | json            | end_time                  | timestamp (utc) |
| training_accuracy_score | float           | true_label                | json            |
| user_id                 | int             | model_predictions         | json            |
| active                  | boolean         | record_timestamp          | timestamp (utc) |
| record_timestamp        | timestamp (utc) |                           |                 |

| MySQL           |           |
|-----------------|-----------|
| Orientation     |           |
| Column Name     | Data Type |
| filename        | text      |
| start_time_sec  | int       |
| start_time_usec | int       |
| end_time_sec    | int       |
| end_time_usec   | int       |
| num_samples     | int       |
| Audio           |           |
| Column Name     | Data Type |
| filename        | text      |
| start_time_sec  | int       |
| start_time_usec | int       |
| end_time_sec    | int       |
| end_time_usec   | int       |
| num_samples     | int       |
| sample_rate     | int       |

Figure 4.16. Application database schema.

## 4.2 Model Performance Comparisons

In addition to the proposed live-inference architecture, this work explores model performance in a simulated production setting by using the MLOps architecture to generate new datasets and evaluate model performance on them. This work tests models on time periods of data outside of those that were used to train the models. By performing inference on these new datasets, the evaluation simulates model performance in a production environment. The test data was selected from increasingly distant time periods from the original training dataset, which spanned from March to August of 2019. The models were tested on subsequent data from the following week, month, and 3 months of data from after the period used for training, testing, and validation. The models referenced in Table 4.1 were developed from work in [7] and [3].

Table 4.1. Pre-trained model accuracy over time in a simulated production environment.

| Model                 | Validation | Test | 1 Week | 1 Month | 3 Months |
|-----------------------|------------|------|--------|---------|----------|
| 4 Channel Custom CNN  | 0.69       | 0.70 | 0.45   | 0.20    | 0.201    |
| 4 Channel Dropout BNN | 0.81       | 0.80 | 0.29   | 0.10    | 0.231    |

The high performance of the models on the validation and test datasets in comparison to their relatively low performance on subsequent time periods indicates the models are possibly overfitting on the training dataset. In order to further validate this, probabilistic MC dropout BNN models were trained on 6-month periods from varying times and tested in the same manner on time ranges increasingly distant from the training dataset. Table 4.2 depicts the models' resulting performance.

Table 4.2. Newly trained model accuracy over time in a simulated production environment.

| Trained Time Period | Validation | Test  | 1 Week | 1 Month | 3 Months |
|---------------------|------------|-------|--------|---------|----------|
| 01-06 2019          | 0.92       | 0.914 | 0.357  | 0.368   | 0.278    |
| 07-12 2019          | 0.87       | 0.887 | 0.336  | 0.350   | 0.356    |
| 01-06 2020          | 0.94       | 0.978 | 0.096  | 0.191   | 0.247    |
| 07-12 2020          | 0.85       | 0.961 | 0.773  | 0.77    | 0.404    |

The test results for the following week and month of the model trained on July through December of 2020 are likely outliers due the fact that 80% of the samples were Class E for no ship. The Class E predictions accounted for the majority of accurate predictions, with the other classes maintaining performance within similar ranges to the other test results.

After reviewing the results, it was evident that the datasets were likely overfitting during training, regardless of the time period the data was trained on. However, this wasn't immediately evident, because the models displayed a high performance on the test datasets. This means that the methodology taken to shuffle the dataset possibly resulted in increased performance on the test dataset so it wasn't indicative of performance in a production environment. Since the dataset is comprised of .tfrecords, each containing approximately 64 30-second acoustic samples, shuffling these files or the 30-second samples directly results in a high likelihood that the same ship target variable is present in the training, validation, and test datasets. Models would then be able to learn a ship in the training dataset and perform well on that same ship in the test and validation datasets, inflating the performance metrics. To explore this possibility, an attempt is made to correct overfitting during training by splitting the dataset in different ways that results in fewer ships being present in all 3 dataset splits.

The first experiment was to conduct a sequential split to ensure no ship targets crossed over between the test, validation, and training datasets. The sequential split orders the acoustic data chronologically, then identifies an 80% training dataset and two 10% datasets for



validation and testing. However, with this approach, it is possible that concept drift from the changing environment during different times of year affects performance results. The second experiment involved grouping the data by each day and shuffling the grouped data before conducting the splits. This approach attempted to account for the changing environment while minimizing the number of targets present in all 3 datasets, since fewer ships crossover between days. The final two approaches apply the same shuffling method after grouping the data by week and month. The results of model performance using each of these splits is depicted in Table 4.3.

Table 4.3. Model performance of different shuffle groupings.

| Shuffle Grouping | Validation | Test  | 1 Week | 1 Month | 3 Months |
|------------------|------------|-------|--------|---------|----------|
| Sequential       | 0.56       | 0.164 | 0.356  | 0.09    | 0.358    |
| Day              | 0.54       | 0.353 | 0.291  | 0.089   | 0.18     |
| Week             | 0.39       | 0.49  | 0.00   | 0.095   | 0.364    |
| Month            | 0.33       | 0.134 | 0.188  | 0.123   | 0.124    |

The various approaches to shuffling methodologies did not result in better sustained performance on data outside of the training datasets. However, the performance on the test dataset is more indicative of the performance in production, so this concern has been addressed in the new shuffling approaches. Of note, the models began to rapidly overfit during training within the first 10 epochs, indicating the model architectures may be too deep or complex. Despite the concern for overfitting, it is possible that the resulting low performance from the models in production is due to the amount of available data for training. To explore this possibility, models were trained on year-long periods to see if it resulted in higher, sustained performance. The results of this experiment are depicted in Table 4.4.

Table 4.4. Models trained on year-long time periods accuracy over time.

| Trained Time Period | Validation | Test  | 1 Week | 1 Month | 3 Months |
|---------------------|------------|-------|--------|---------|----------|
| 2019                | 0.76       | 0.853 | 0.292  | 0.298   | 0.292    |
| 2020                | 0.69       | 0.808 | 0.833  | 0.831   | 0.404    |

Similar to the results for the 07-12 2020 model in Table 4.2, the 1 week and 1 month test results for the model trained on 2020 are outliers due to the high number of Class E “No Ship” designations that account for the majority of samples. Generally, the models trained on year-long time periods seem to have similar or worse performance compared to the models trained on 6-month time periods. As such, the models do not generalize well and do not seem to perform much better than random choice.

THIS PAGE INTENTIONALLY LEFT BLANK

---

## CHAPTER 5: Conclusion

---

### **5.1 Discussion**

The main contribution of this research is the proposed system architecture for a production MLOps platform for acoustic signals. It enables researchers and end-users to train new models, deploy them, observe ML models in production, retrain models, and manage the data. The MLOps architecture also enables development of new, progressively larger datasets than ever before used in literature for more robust testing and training new models on passive sonar signals. Integrating the various components of this system ultimately serves the purpose of providing end-to-end functionality for ML and accelerating the pace at which research can be conducted by housing this functionality in a single platform.

Having this functionality of the system enabled testing the model performance in a production setting to identify that it degraded. This further enabled investigation of the source of this degradation. Additionally, training models on larger datasets did not improve performance in production. The insights gained from using the production architecture emphasize the importance of maintaining a holistic MLOps process to support the full model lifecycle, because not doing so leaves significant gaps that could affect the interpretation of results or leave important factors undiscovered.

There are a variety of possible reasons that could have contributed to the models' degraded performance in production. One possibility is concept drift in the form of the dynamic environmental factors or changing statistics in the relation between ship classes and their acoustic signatures. These drifts could be happening so rapidly that each models' ability to generalize degrades almost immediately after the training period. Another possibility could involve unaccounted-for noise that distorts ship sounds, including ships not transmitting AIS that are present in the acoustic signals or biological sources. Unbalanced classes could have additionally contributed to the model performance issues as well due to underrepresented and overrepresented classes. This concern is particularly relevant in multilabel classification approaches, as each combination of classes is also treated as a separate label. Ultimately,

model performance degradation on time periods outside of the training dataset could have been a result of any of these factors or a combination of all of them.

## 5.2 Future Work

Additional work should be performed to identify how to prevent model overfitting during training and the best way to prepare the data to prevent this. This could include boosting underrepresented classes in the dataset and then further determining the best approaches to perform training, testing, and validation splits to minimize the number of targets present in all three groups. Further model tuning and hyperparameter searches will also likely be required in order to progress this research. Since the models overfit so quickly when testing the new shuffling approaches, the model architecture may be too deep, so research should experiment with less complex architectures.

Concept drift is another area research could focus on exploring further. In order to better quantify the source of concept drift in the system, more work should be done to look at the changing statistical relationship over time between the ship classes, ship designations, and acoustic signatures. Efforts could also include identifying the extent of unaccounted-for noise that is present in the acoustic data. Once concept drift is better quantified and identified, automated methods of updating models could be developed to deploy a new model once drift has exceeded a specified threshold.

The MLOps platform also contains opportunities to build additional features. Active learning can be automated in the live-inference system to automatically cue the retraining and redeployment of an active learning model based on uncertainty thresholds as executed in [7]. Further exploration of active learning methods can help determine whether this approach combats concept drift more effectively. User interfaces can also be developed to create and manage new datasets and training for new ML models using no-code approaches. The platform can also be adapted so each deployed model exists within a container and is exposed via a Representational State Transfer (REST) framework to increase modularity as recommended in [24]. Additionally, timelines from acoustic events until model prediction and user display should be explored and optimized to the extent possible.

Holistic security considerations for the application can also be explored, to include the

physical architecture, software, and security of the ML models. ML model security can encompass developing standardized testing to ensure the model is ready for production and exploring durability to adversarial attacks. Tests can also be developed to run on the production models to ensure continuous testing and security.

Finally, new approaches and model architectures can be developed and added to the system for performance comparisons. For example, a ship/no ship binary classifier was implemented in [3] that can be adapted for this platform. Models can be developed for multi-instance classification, which would be able to account for multiple instances of the same ship class. Object detection models can be explored as another approach to serve as multi-instance, multilabel classifiers. Approaches can also be explored to build classifiers for ship designations directly rather than ship classes to see if they perform better than ship class groupings and prevent concept drift in the target variable. Additionally, approaches using acoustic data from multiple sensors can be developed to identify how models perform across different physical environments and sensor characteristics. Finally, models can be developed to compare the relative effectiveness of different data processing approaches such as calibrated versus uncalibrated datasets.

THIS PAGE INTENTIONALLY LEFT BLANK

---

---

## List of References

---

- [1] Department of Defense, “Summary of the 2018 Department of Defense Artificial Intelligence Strategy,” p. 17, 2018. Available: <https://media.defense.gov/2019/Feb/12/2002088963/-1/-1/1/SUMMARY-OF-DOD-AI-STRATEGY.PDF>
- [2] Department of Defense Chief Information Officer. Joint Artificial Intelligence Center. [Online]. Available: <https://dodcio.defense.gov/About-DoD-CIO/Organization/jaic/>. Accessed September 30, 2021.
- [3] M. Pfau, Andrew, “Multi-label classification of underwater soundscapes using deep convolutional neural networks,” M.S. Thesis, Dept. of Operational and Information Sciences., NPS, Monterey, CA, USA, 2020 [Online]. Available: <https://calhoun.nps.edu/handle/10945/66705>
- [4] Office of Naval Intelligence Data Science Team, private communication, December 2020.
- [5] J. Anderson, private communication, September 2021.
- [6] B. Beckler, “Enhanced multi-label classification of heterogenous underwater soundscapes by convolutional neural networks using bayesian deep learning,” M.S. Thesis, Dept. of Operational and Information Sciences., NPS, Monterey, CA, USA, 2021.
- [7] B. Atchley, “Active bayesian deep learning with an acoustic vector sensor,” M.S. Thesis, Dept. of Operational and Information Sciences., NPS, Monterey, CA, USA, 2021.
- [8] *Oxford English Dictionary*. “Artificial intelligence”. [Online]. Available: <https://www.oed.com/viewdictionaryentry/Entry/271625>. Accessed October 02, 2021.
- [9] *Oxford English Dictionary*. “Machine learning”. [Online]. Available: <https://www.oed.com/view/Entry/111850?redirectedFrom=machine+learning#eid38479194>. Accessed October 02, 2021.
- [10] *Oxford English Dictionary*. “Deep learning”. [Online]. Available: <https://www.oed.com/view/Entry/48625?redirectedFrom=deep+learning#eid1295227860>. Accessed October 02, 2021.
- [11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016 [Online]. Available: <http://www.deeplearningbook.org>



- [12] S. Schuchmann, “Analyzing the prospect of an approaching AI winter,” Ph.D. dissertation, May 2019.
- [13] M. Haenlein and A. Kaplan, “A brief history of artificial intelligence: On the past, present, and future of artificial intelligence,” *California Management Review*, vol. 61, no. 4, pp. 5–14, Aug. 2019 [Online]. Available: <https://doi.org/10.1177/0008125619864925>
- [14] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, & Tensorflow. Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd ed. Sebastopol, CA: O’Reilly Media Inc., 2019.
- [15] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, “Language models are few-shot learners,” in *34th Conference on Neural Information Processing Systems*, July 2020 [Online]. Available: <http://arxiv.org/abs/2005.14165>
- [16] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, “Zero-shot text-to-image generation,” in *Proceedings of the 38th International Conference on Machine Learning*, Feb 2021 [Online]. Available: <http://arxiv.org/abs/2102.12092>
- [17] B. M. Ozyildirim and S. Kartal, “Comparison of deep convolutional neural network structures: The effect of layer counts and kernel sizes,” in *Fourth International Conference on Advances in Information Processing and Communication Technology - IPCT 2016.*, Aug. 2016, pp. 16–19 [Online]. Available: <https://www.seekdl.org/conferences/paper/details/8220>
- [18] H. Wang and D.-Y. Yeung, “A survey on bayesian deep learning,” *ACM Computing Surveys*, vol. 53, no. 5, pp. 1–37, Oct. 2020. Available: <https://dl.acm.org/doi/10.1145/3409383>
- [19] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, “Weight uncertainty in neural network,” in *Proceedings of the 32nd International Conference on Machine Learning (Proceedings of Machine Learning Research)*, F. Bach and D. Blei, Eds. Lille, France: PMLR, 07–09 Jul 2015, vol. 37, pp. 1613–1622. Available: <http://proceedings.mlr.press/v37/blundell15.html>
- [20] R. Marcinkevičs and J. E. Vogt, “Interpretability and explainability: A machine learning zoo mini-tour,” *arXiv preprint arXiv:2012.01805*, Dec. 2020 [Online]. Available: <http://arxiv.org/abs/2012.01805>

- [21] M. Orescanin, S. Atchley, B. Beckler, N. Villemez, A. Pfau, P. Leary, and K. Smith, “Active bayesian deep learning with vector sensor for passive sonar sensing of the ocean,” *in review: IEEE Journal of Oceanic Engineering*, 2021.
- [22] Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: representing model uncertainty in deep learning,” in *Proceedings of The 33rd International Conference on Machine Learning*, Oct. 2016, pp. 1050–1059, arXiv: 1506.02142. Available: <http://arxiv.org/abs/1506.02142>
- [23] K. Salama, J. Kazmierczak, and D. Schut, “Practitioners guide to MLOps: A framework for continuous delivery and automation of machine learning.” p. 37, May 2021 [Online]. Available: [https://services.google.com/fh/files/misc/practitioners\\_guide\\_to\\_mlops\\_whitepaper.pdf](https://services.google.com/fh/files/misc/practitioners_guide_to_mlops_whitepaper.pdf)
- [24] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, “Hidden technical debt in machine learning systems,” in *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2015, vol. 28 [Online]. Available: <https://proceedings.neurips.cc/paper/2015/hash/86df7dcfd896fcf2674f757a2463eba-Abstract.html>
- [25] A. Wiedemann, N. Forsgren, M. Wiesche, H. Gewalt, and H. Krcmar, “Research for practice: the DevOps phenomenon,” *Communications of the ACM*, vol. 62, no. 8, pp. 44–49, July 2019 [Online]. Available: <https://doi.org/10.1145/3331138>
- [26] A. Tsymbal, “The problem of concept drift: definitions and related work,” *Computer Science Department, Trinity College Dublin*, vol. 106, no. 2, p. 58, 2004 [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.58.9085&rep=rep1&type=pdf>
- [27] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A survey on concept drift adaptation,” *ACM Computing Surveys*, vol. 46, no. 4, pp. 44:1–44:37, Mar. 2014 [Online]. Available: <https://doi.org/10.1145/2523813>
- [28] S. Ntalampiras, “Automatic analysis of audiostreams in the concept drift environment,” in *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, Sep. 2016, pp. 1–6.
- [29] I. D. Id, M. Abe, and S. Hara, “Evaluation of concept drift adaptation for acoustic scene classifier based on Kernel Density Drift Detection and Combine Merge Gaussian Mixture Model,” in *TENCON 2020 - 2020 IEEE Region 10 Conference (TENCON)*, Mar. 2021 [Online]. Available: <http://arxiv.org/abs/2105.13220>

- [30] D. Baylor, E. Breck, H.-T. Cheng, N. Fiedel, C. Y. Foo, Z. Haque, S. Haykal, M. Ispir, V. Jain, L. Koc, C. Y. Koo, L. Lew, C. Mewald, A. N. Modi, N. Polyzotis, S. Ramesh, S. Roy, S. E. Whang, M. Wicke, J. Wilkiewicz, X. Zhang, and M. Zinkevich, “TFX: A TensorFlow-based production-scale machine learning platform,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*. New York, NY, USA: Association for Computing Machinery, Aug. 2017, pp. 1387–1395 [Online]. Available: <https://doi.org/10.1145/3097983.3098021>
- [31] C. Min, A. Mathur, U. G. Acer, A. Montanari, and F. Kawsar, “SensiX++: Bringing MLOPs and multi-tenant model serving to sensory edge devices,” *arXiv:2109.03947 [cs]*, Sep. 2021 [Online]. Available: <http://arxiv.org/abs/2109.03947>
- [32] M. J. Bianco, P. Gerstoft, J. Traer, E. Ozanich, M. A. Roch, S. Gannot, and C.-A. Deledalle, “Machine learning in acoustics: Theory and applications,” *The Journal of the Acoustical Society of America*, vol. 146, no. 5, pp. 3590–3628, Nov. 2019 [Online]. Available: <http://asa.scitation.org/doi/10.1121/1.5133944>
- [33] D. Gillespie, “Detection and classification of right whale calls using an ‘edge’ detector operating on a smoothed spectrogram,” *Canadian Acoustics*, vol. 32, no. 2, pp. 39–47, June 2004.
- [34] M. Thomas, B. Martin, K. Kowarski, B. Gaudet, and S. Matwin, “Marine mammal species classification using convolutional neural networks and a novel acoustic representation,” in *Machine Learning and Knowledge Discovery in Databases*, Würzburg, Germany, July 2019, arXiv: 1907.13188 [Online]. Available: <http://arxiv.org/abs/1907.13188>
- [35] A. Zak, “Ship’s hydroacoustics signatures classification using neural networks,” in *Self Organizing Maps - Applications and Novel Algorithm Design*, J. I. Mwasiagi, Ed. InTech, Jan. 2011 [Online]. Available: <http://www.intechopen.com/books/self-organizing-maps-applications-and-novel-algorithm-design/ship-s-hydroacoustics-signatures-classification-using-neural-networks>
- [36] D. Santos-Domínguez, S. Torres-Guijarro, A. Cardenal-López, and A. Pena-Gimenez, “ShipsEar: An underwater vessel noise database,” *Applied Acoustics*, vol. 113, pp. 64–69, Dec. 2016 [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0003682X16301566>
- [37] H. Niu, E. Ozanich, and P. Gerstoft, “Ship localization in Santa Barbara Channel using machine learning classifiers,” *The Journal of the Acoustical Society of America*, vol. 142, no. 5, pp. EL455–EL460, Nov. 2017 [Online]. Available: <http://asa.scitation.org/doi/10.1121/1.5010064>

- [38] H. Berg, K. T. Hjelmervik, D. H. S. Stender, and T. S. Sastad, “A comparison of different machine learning algorithms for automatic classification of sonar targets,” in *OCEANS 2016 MTS/IEEE Monterey*. Monterey, CA, USA: IEEE, Sep. 2016, pp. 1–8 [Online]. Available: <http://ieeexplore.ieee.org/document/7761112/>
- [39] T. B. Neilsen, C. D. Escobar-Amado, M. C. Acree, W. S. Hodgkiss, D. F. Van Komen, D. P. Knobles, M. Badiy, and J. Castro-Correa, “Learning location and seabed type from a moving mid-frequency source,” *The Journal of the Acoustical Society of America*, vol. 149, no. 1, pp. 692–705, Jan. 2021 [Online]. Available: <https://asa.scitation.org/doi/10.1121/10.0003361>
- [40] “Geospectrum technologies inc., *M20-105*,” 2018 [Online]. Available: <https://geospectrum.ca/wp-content/uploads/2018/11/M20-105-v7.pdf>
- [41] J. P. Ryan, D. E. Cline, J. E. Joseph, T. Margolina, J. A. Santora, R. M. Kudela, F. P. Chavez, J. T. Pennington, C. Wahl, R. Michisaki, K. Benoit-Bird, K. A. Forney, A. K. Stimpert, A. DeVogelaere, N. Black, and M. Fischer, “Humpback whale song occurrence reflects ecosystem variability in feeding and migratory habitat of the northeast Pacific,” *PloS one*, vol. 14, no. 9, p. e0222456, Sep. 2019 [Online]. Available: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0222456>
- [42] J. P. Ryan, J. E. Joseph, T. Margolina, L. T. Hatch, A. Azzara, A. Reyes, B. L. Southall, A. DeVogelaere, L. E. Peavey Reeves, Y. Zhang, D. E. Cline, B. Jones, P. McGill, S. Baumann-Pickering, and A. K. Stimpert, “Reduction of low-frequency vessel noise in monterey bay national marine sanctuary during the covid-19 pandemic,” *Frontiers in Marine Science*, vol. 0, 2021 [Online]. Available: <https://www.frontiersin.org/articles/10.3389/fmars.2021.656566/full>
- [43] Monterey Bay Aquarium Research Institute. Monterey Accelerated Research System (MARS). [Online]. Available: <https://www.mbari.org/at-sea/cabled-observatory/>
- [44] P. Leary, “Vector sensor processing,” 2019 [Online]. Available: <https://gitlab.nps.edu/vector-sensor-processing>
- [45] VesselFinder, “Vesselfinder map,” 2011 [Online]. Available: <https://www.vesselfinder.com/>
- [46] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas,

- O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015 [Online]. Available: <https://www.tensorflow.org/>
- [47] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: An overview," *arXiv*, Aug. 2020 [Online]. Available: <https://arxiv.org/abs/2008.05756>
- [48] M. Grinberg, "The flask mega-tutorial," Dec. 2017 [Online]. Available: <https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world>
- [49] M. Herman, "Test-Driven IO," June 2021 [Online]. Available: <https://testdriven.io/courses/tdd-flask/>
- [50] B. D. Team, "Bokeh: Python library for interactive visualization," 2018 [Online]. Available: <https://bokeh.pydata.org/en/latest/>
- [51] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.
- [52] PostgreSQL Global Development Group, *PostgreSQL, Version 13.3*, 1996 [Online]. Available: <https://www.postgresql.org/>
- [53] Oracle Corporation, *MySQL, Version 8.0.26*, 1995 [Online]. Available: <https://www.mysql.com>
- [54] U.S. Department of Transportation, "Seavision application programming interface," 2019 [Online]. Available: <https://info.seavision.volpe.dot.gov/node/131>

---

---

## Initial Distribution List

---

1. Defense Technical Information Center  
Ft. Belvoir, Virginia
2. Dudley Knox Library  
Naval Postgraduate School  
Monterey, California