EVALUATION OF A DEEP LEARNING-BASED ALGORITHMIC PIPELINE FOR REAL-TIME ONBOARD REMOVAL OF GLIDER SELF-NOISE FROM PASSIVE ACOUSTIC DATA

by

ANNA HU

(Under the Direction of Catherine R. Edwards and Guoming Li)

ABSTRACT

We created a dataset from passive acoustic data collected by autonomous underwater vehicles (AUV) called gliders and attempted to create a deep learning-based framework to identify instances of vehicle self-noise in the acoustic data. Data for this work was taken from a 2024 right whale monitoring mission in the Georgia/Florida calving grounds. The main sources of self-noise from the glider arise from the ballast pump, pitch motor, and air pump, which often co-occur as the glider moves in the water column. Various deep learning classifiers were investigated with different integrated sampling and data augmentation techniques to alleviate the data imbalance inherent in the dataset. All models performed poorly, likely due to the small minority amount in the datasets. ResNet50 and MobileNetV2 outperformed other deep learning models and were used as baselines for comparison. Results suggest that oversampling to achieve balanced labelsets improved model precision and recall most, but increased loss.

INDEX WORDS: Multilabel Classification, Audio Classification, Data Augmentation, Convolutional Neural Networks

EVALUATION OF A DEEP LEARNING-BASED ALGORITHMIC PIPELINE FOR REAL-TIME ONBOARD REMOVAL OF GLIDER SELF-NOISE FROM PASSIVE ACOUSTIC DATA

by

ANNA HU

B.S., University of Georgia, 2024

B.A., University of Georgia, 2025

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTERS OF SCIENCE

ATHENS, GEORGIA

2025

EVALUATION OF A DEEP LEARNING-BASED ALGORITHMIC PIPELINE FOR REAL-TIME ONBOARD REMOVAL OF GLIDER SELF-NOISE FROM PASSIVE ACOUSTIC DATA

by

ANNA HU

Major Professor:	Catherine R. Edwards
	Guoming Li
Committee:	Frederick Maier
	Jin Lu

Electronic Version Approved:

Ron Walcott

Vice Provost for Graduate Education and Dean of the Graduate School

The University of Georgia

December 2025

DEDICATION

For my friends and family for supporting me every step of the way.

ACKNOWLEDGEMENTS

I would like to thank my major professors, Catherine R. Edwards and Guoming Li, for agreeing to be my advisors after surprise emails from me, being so supportive all the time, teaching me both about research and the ways of the world, and for putting up with me even at the busiest parts of the year.

I would also like to thank everybody in Edwards Lab at Skidaway Institute. Thank you all for being so welcoming, teaching me about gliders and radar and letting me also participate and continue participating in glider missions. I learned more in the summer I was at Skidaway than I possibly ever could in any classroom or tiny dorm room staring at a screen.

A special thank you to my family for supporting me and coming up with (not always good) ideas for me. My biggest thanks goes to my sister, Emma – you taught me about things that I made a huge difference to how I approached my thesis, and your field of study isn't remotely related.

TABLE OF CONTENTS

Chapter 1

Introduction

ℵ 1.1 Introduction

The North Atlantic Right Whale (NARW, *Eubalaena glacialis*) is one of the most endangered species of whale in the world, with an estimated population of only 370 individuals in 2024 [1]. Because of climate change, the distribution and behaviors of NARW have changed greatly in the past few years [2], bringing them closer to areas with high density vessel traffic like major shipping lanes and commercial fishing grounds, which make them more vulnerable to vessel strikes and fishing gear entanglements [3]. Recent efforts to protect the species include aerial surveys, vessel-based surveys, and passive acoustic monitoring (PAM) to localize the animals for seasonal or dynamic management [3].

One of the important tools for PAM is underwater gliders. Gliders are underwater autonomous vehicles that swim in an up-down motion by changing their buoyancy and center of gravity [4]. Their quiet nature and ability to conduct long endurance missions of weeks to months make them ideal for PAM [5]. In recent years, gliders with integrated hydrophones have conducted NARW monitoring missions for the purposes of mitigating threats and observing habitat use [3] in near-real time using an integrated digital acoustic monitoring (DMON) hydrophone with onboard detection capability [5]. The onboard identification algorithm [6] segments out detections, which are transmitted to shore for analyst confirmation each time the

vehicle surfaces. Data sharing through the *robots4whales* network facilitates notification of stakeholders and alerts for dynamic management, such as vessel strike mitigation [3].

As part of these efforts, a pilot program was established by the Skidaway Institute of Oceanography and the University of South Carolina to conduct glider-based monitoring of NARW in their winter calving grounds in the southeastern United States [3]. In contrast to the deeper waters in which the whales spend most of their lives, the calving grounds off the FL, GA, and SC coasts lie in waters shoreward of the 20 m isobath, which is near the shallow limit of glider operation. Because of the sawtooth pattern of glider motion, operation in shallow water requires more frequent inflections, which reduces vehicle speed and thus controllability, requires greater power consumption to move motors more frequently, and produces more noise that can interfere with acoustic monitoring.

The main causes of glider self-noise arise from the actuation of the buoyancy pump and pitch motor used for propulsion, as well as the air pump that inflates at the surface for communications and the fin used for steering. As shown in Figure 1.1, the noise from these motors can be broadband and of much higher magnitude than the signal of the whale's call (70 vs 15 dB) and can mask the signal completely. Further, telemetering data to shore via satellite is cost-prohibitive and time at the surface is operationally risky, so detection data files transmitted at each surfacing are capped to approximately 46 kB. In addition to obscuring true positives, false positives from glider self-noise can fill up the telemetered file such that fewer true positives are transmitted in near real-time. Glider self-noise makes up roughly 10% of all passive acoustic data collected in a 30-day-long mission, meaning that approximately 3 days' worth of potential whale detections may be missed.

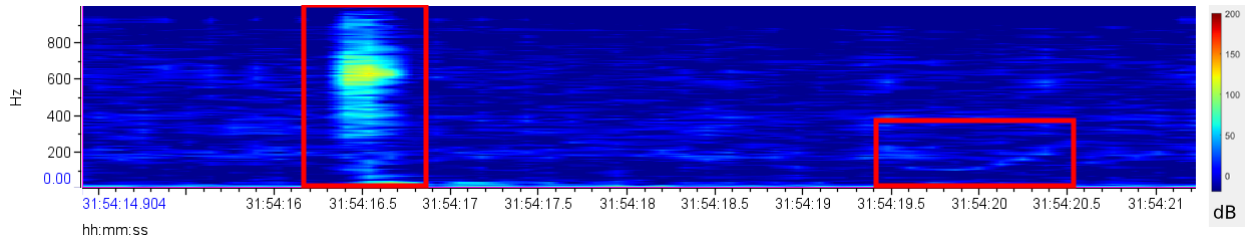


Figure 1.1. Spectrogram with pitch motor noise and whale upcall, with the signals each boxed in red on the left and right, respectively.

Because glider self-noise has very regular patterns, it is possible to develop a set of algorithms to filter out glider self-noise in real time. However, the instances of glider self-noise need to be identified in order to be able to apply these filters. Motor variable data from the glider engineering computer can be used as a reference to locate glider self-noise but the glider engineering data cannot be transferred to the hydrophone computer in real time when in the glider, causing a need for an algorithm to detect instances of glider self-noise.

In this thesis, passive acoustic data and glider engineering data from the January 2024 right whale mission were synced together to create a multi-label audio dataset for the identification of different glider motor self-noises. The resulting dataset was split into chunks and fed into a CNN for classification, which outputs an array of probabilities, with each probability in the array corresponding to a noise generating glider motor (Fig. 3.1b). Current deep learning classifiers are mainly trained on common scenes related to humans, and the effectiveness and performance of classifying self-noise via those models were optimized. Because of the imbalanced nature of the PAM data, various resampling and data augmentation techniques were tested to potentially improve model learning. The objective of this thesis is to develop a deep learning-based algorithmic framework to classify glider self-noise from passive acoustic data. The contributions

of this thesis include: 1) developing the first deep learning pipeline for glider self-noise detection with multilabel classification; 2) converting the audio signal processing problem into deep learning image classification problem to enhance computational efficiency and robustness; and 3) optimizing self-noise classification performance via multiple deep learning classifier comparison and image-/audio-based data augmentation techniques for solving data imbalance.

1.2 Structure of the Thesis

The thesis is organized as follows: Chapter 2 covers the background and related work to the thesis; Chapter 3 discusses the methods and experiments; Chapter 4 presents critical results of self-noise classification; and Chapter 5 is the conclusion and discussion.

Chapter 2

Background and Related Work

ℵ 2.1 Detection Algorithms for Passive Acoustic Monitoring

As recording and storage technology gets cheaper, larger amounts of data are collected with PAM; processing the collected data, the standard for which is analysis by humans who have trained on the procedure, becomes extremely difficult [6, 7]. Manual annotations of passive acoustic data and training analysts for the task require significant time and resources, which increases as the amount of data to process increases, prompting the development of automated detection and classification systems. Research has been conducted into developing detection algorithms for several fields, including monitoring wildlife [6, 8, 9, 10, 11] and environmental conditions [12, 13]. A wide array of methods has been applied to developed for the detection of marine mammals alone, such as feature detection, energy-based algorithms, Hidden Markov Models, and neural networks [8]. In this thesis, we focus on reviewing the detection algorithms created for detecting NARW.

Baumgartner et. al. [6] used feature detection to create a generalized detection algorithm for low frequency (< 1 kHz) baleen whale calls by analyzing the pitch tracks extracted from spectrograms created from the audio. Features were extracted from the pitch track and matched to a call library. The extracted pitch tracks and the classifications were compared with the detections marked by 3 different human analysts. For sei whale classification, the detection algorithm missed 38, 51, and 33% of calls for each analyst and incorrectly classified 62, 7, and

15% of the labeled calls. The detection algorithm missed 54, 49, and 63% of all the right whale calls annotated, and incorrectly classified 73, 59, and 58%.

Dugan et. al. [11, 14] used a cascade of several different classifiers including feature vector testing, classification regression tree, and artificial neural nets to detect and classify NARW upcalls. 58,624 NARW call samples and 81,880 noise samples were taken from two different datasets from 2000-2007 in three locations along the NARW's migration route were taken and inputted into the system as spectrograms. The system was tested on a database of 58,624 samples and achieved a 78.02% detection probability, a 3.25% error rate, with 1,072 and 2,324 false positives for each dataset used.

Although the work by Dugan et. al. [11, 14] appeared to outperform the algorithm developed by Baumgartner et. al. [6], the two studies have very different testing processes. Dugan et. al. [11, 14] measure performance against a database of NARW calls, while Baumgartner et. al. [6] measured algorithm performance against three groups of annotations analyzed by human analysts and took the subjective nature of manual annotation into account, providing several ways to adjust the proposed algorithm to fine-tune detections.

ℵ 2.2 Deep Learning for Passive Acoustic Monitoring

In the past decade, the success of deep learning methods triggered an increase in researching deep learning models applications in detection algorithms for processing passive acoustic data [7, 15]. As with earlier detection algorithms, many studies have been conducted on marine mammal research, often with the goal of reducing the false positives detection rate of earlier detection algorithms [7, 15, 16].

Shiu et. al. [15] trained 5 different CNNs and RNNs on NARW upcalls from 3 datasets collected in 2009 and 2012-2015 in various locations off the United States coastline. The datasets were labeled by human analysts and converted to 2 second spectrograms for training. Several 2D CNNs LeNet, BirdNET, VGG, ResNet, and Conv1D + GRU were used for training with 100 epochs and batch size of 1000. The 5 models yielded similar results, but LeNet achieved the highest precision and the lowest number of false positives. The models were tested on the DCLDE 2013 data and achieved a 70% recall with < 0.3 false positives detected per hour.

Kirsebom et. al. [16] trained a ResNet model on around 20,000 3-sec samples made from a combined dataset of NARW calls collected in 2009 and 2018-2019. The samples were converted to spectrograms and fed into a 2D CNN for a binary classification task (positive or negative for NARW calls), achieving a recall of 87.5% and a precision of 90.2%. The effect of increasing the size of the dataset using time shifting was also tested, with negligible improvement.

Similar work in other fields of ecology have compared the performance of using different audio representations as inputs. Dufourq et. al. [10] compare the performance of a 1D CNN with preprocessed amplitudes as the input with that of a 2D CNN with spectrograms as input. A total of 256 hours of recorded data were taken from recorders listening for Hainan gibbon calls, manually annotated, and segmented into 10 second segments. To balance the positive and negative, negative samples were randomly undersampled and more samples were created using data augmentation methods including time shifting and blending. Segments that were likely to be false positives were removed as a post-processing step. The resulting dataset had 18,992 samples (50% presence, 50% absence) and were split 60%-40% for training and validation. The 1D CNN test with data augmentation and without the post-processing step achieved an accuracy of 94.76% and a precision of 41.35%, the highest of all the 1D CNNs tests. This result was

outperformed by all the 2D CNN tests except the test without data augmentation and without the post-processing step. The best 2D CNN test was the test with data augmentation and the post-processing step, achieving an accuracy of 99.37% and a precision of 85.30%, which was significantly better than the performance of the 1D CNN.

A common problem faced when applying deep learning methods to PAM is the shortage of large, annotated, and balanced datasets with which to train deep learning models [7, 17]. Data augmentation is a popular and frequent solution to increase the number of samples, balance datasets, and improve generalizability [7, 15]. Several studies have been conducted to explore various data augmentation methods and their effects on model performance.

Padovese et. al. [17] explores two audio data augmentation packages, *SpecAugment* and *Mixup*, and their effects on performance on one of the datasets used by Kirsebom et. al [16]. *SpecAugment* created augmented samples by applying functions like time warping and masking to the segment, while *Mixup* created new samples by mixing two existing samples together. Tests were run on the full dataset (increased to a total of 40,000 samples from the baseline 3,888 samples) and an undersampled dataset (increased to a total of 8,000 samples from the baseline 200 samples). All of the resulting datasets had balanced samples for each class (positive and negative). Models trained on the full dataset achieved a performance increase as much as 89% to 93% in recall and 85% to 90% precision. On the other hand, models trained on the undersampled dataset were unable to learn properly and classified every sample as positive.

Nanni et. al. [18] tested the effects of different data augmentation methods applied to samples in different phases of the audio preprocessing stage. Ten-fold cross validation was run on CNNs based on GoogleNet and VGG with pretrained weights from datasets like ImageNet and Places365. Two datasets were used: BIRDZ – a multiclass dataset of bird calls with 3,101

samples created from the Xeno-canto Archive by Zhao et. al. [19]; and CAT – a multiclass dataset of around 3,000 cat calls collected from Kaggle, YouTube and Flickr by Pandeya et. al. [20]. Results show that models trained with datasets with spectrogram augmentation performed the best, with a best increase of 87% to 91%, while datasets with image augmentation caused the models to do worse than the baseline datasets without augmentation.

ℵ 2.3 Multi-label Classification

Multi-label classification (MLC) is a category of classification task in which each data point has the potential to be assigned multiple labels simultaneously. Multilabel datasets, like the one considered in this work, are distinct from other types of datasets in that each data point can be labeled with more than one class [21]. Each combination of labels (e.g. [0, 0, 0] or [1, 0, 0]) is called a labelset. Multilabel datasets and classification are often preferred because they reflect the complexity of real-world problems better than traditional classification tasks do [22].

One of the biggest problems in multilabel classification problems is data imbalance. Because of the label relationships in multi label datasets, many sampling methods that work for binary and multiclass datasets do not work for multilabel datasets. There are many ways to deal with multi-label dataset imbalance, such as sampling and ensemble methods [21]. Charte et. al. [23] introduces resampling algorithms, LP-RUS/OS and ML-RUS/OS, which is based on under/oversampling by labelsets (combinations of labels) for imbalance in multi label datasets. Tahir et. al. [24] discussed inverse random undersampling (IRUS) and its application to multi-label datasets. The approach requires binary relevance (BR), which would transform the dataset into multiple binary datasets (one for each class). One classifier would be trained for each dataset and then combined with an ensemble learning method. Cakir et. al. [25] evaluated the performance difference between running Combined Single Label DNNs (CSL DNNs) with

running one multi-label DNN. The multi-label DNN outperformed the CSL DNNs in accuracy by 2-3%. Furthermore, for multi-label problems with a large number of classes, CSL DNNs would use several times more computing resources than one multi-label DNN.

Multi-label audio classification has been employed for classifying polyphonic environmental sound, where each audio segment potentially has several different audio sources. Cakir et. al. [26] used a multi-label DNN with 2 hidden layers each with 800 units to classify polyphonic environmental sounds. The classifier was tested on a dataset of 1,133 minutes of recordings from everyday environments with 61 different classes and achieved an accuracy of 63.8%.

In summary, huge amounts of data collected in PAM necessitate automated detection classifiers because of the time and labor-intensive nature of manual annotation by trained human analysts, which has been the standard method for analyzing passive acoustic data. Early automated detection algorithms to identify whales in passive acoustic data sets often result in lower-than-ideal success rates, with a large number of false negatives and false positives. Deep learning methods perform better, but require large amounts of annotated and balanced data, which can be difficult to obtain. Data augmentation methods have been explored to mitigate this problem, with mixed results. Most deep learning-based detection algorithms use binary classification (positive or negative), but since multiple sources of vehicle self-noise often co-occur, the system examined here is a multi-label classification problem, and the literature provides a guide for potential solutions to the data imbalance issues associated with multi-label methods.

Chapter 3

Methodology

ℵ 3.1 Hardware

The Google Colab v2-8 TPU was used to create the datasets, and the Google Colab A100 GPU was used to train and run the models for this thesis.

ℵ 3.2 Dataset Building

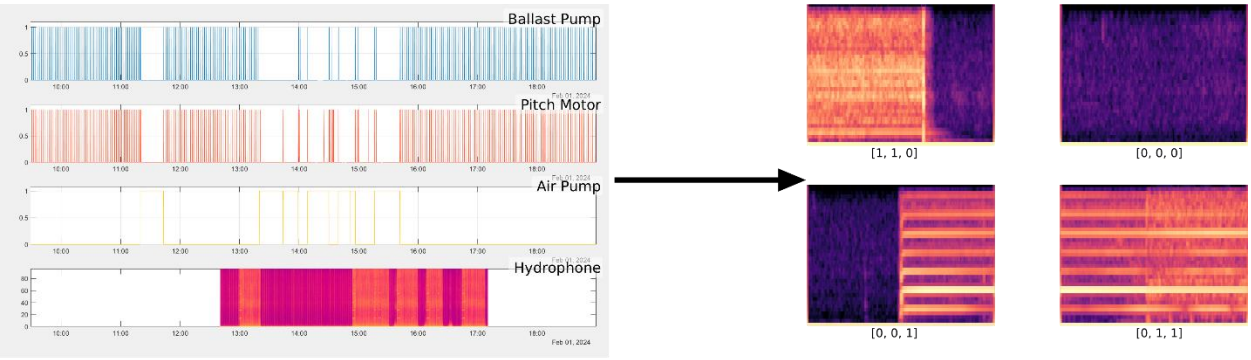


Figure 3.1. Pipeline for building the dataset.

Passive acoustic data and glider engineering data from the January-February 2024 glider mission off the coast of Georgia were used to create the training and testing datasets. The passive acoustic data were collected, processed, and stored on the DMON board; in real time, subsets of these data were sent through the glider's science and flight processor to shore, and the full data was downloaded post-deployment and are used in the analysis that follows. First, we calculated

the time difference between the hydrophone data and the glider engineering data and synced them together (Fig. 3.1a). This time difference (roughly 1 hour and 40 minutes) was caused by clock drift between the hydrophone and glider clocks, as well as an error in the glider header code. The hydrophone data was then split into several segments and converted into a mel-spectrogram (converted using the Python *librosa* package v 0.11.0 [27]). The glider engineering data was sampled at a rate of approximately 4 seconds, so the passive acoustic data was split into 4-second segments to match to reduce the number of potential labeling errors. Each spectrogram was then given a label per source of self-noise present in the segment. There were 3 main sources of glider self-noise in this data: ballast pump, pitch motor, and air pump. Labels were formatted as a three-element list in the order of ballast pump, pitch motor, and air pump to indicate true (1) or false (0) (Fig. 3.1b). Labels for the ballast pump and the pitch motor were taken from their respective motor variables in the glider engineering data. The air pump times were manually labeled and used to calculate air pump labels because the variables corresponding with the air pump did not correlate with self-noise generated by the air pump.

The downside of automatically labeling compared to manual labeling is that, due to the glider's flight computer's 4-second sampling cycle, automating the labeling process with the glider engineering data caused various instances of self-noise to not be labeled as such. Pitch motor self-noise was particularly vulnerable to this because of its short duration (1-3 seconds), causing many instances of pitch motor self-noise to be labeled as false. Despite this flaw, this method was still used for the process because it was much more efficient and saves a lot of time than a human annotator would require annotating the data. The automated labeling process took less than 1 hour to label 24 hours of PAM data, while a human annotator took 20 hours to annotate 15 hours of data. Although the air pump self-noise needed to be manually annotated, the

process took around 10 minutes to complete because there were only 6 occurrences of self-noise from the air pump – which only turned on every 4 hours when the glider surfaced, as opposed to every 5-30 seconds, like the ballast pump and pitch motor did. It is common for the pitch motor and buoyancy pump to co-occur, since both actions are required simultaneously for glider flight, and all three motors can co-occur at the beginning and end of glider surfacings.

We chose to create a multilabel dataset (where each motor was a class and each combination of motor) instead of multiclass (where each combination of motors was a class), which would increase the number of motor combinations for the model to predict.

3.3 Image Similarity Scores

To measure the correlations between each class and labelset, we calculated the image similarity of select images from the data. We used two methods of image similarity comparisons: histogram-based, which compares the histograms of two images, and the Structure Similarity Index (SSIM), which compares brightness, contrast, and structure and is more reflective of the difference seen by human eyes. We chose the histogram-based methods because each type of glider self-noise was expected to have their own feature shape and signal strength, which when mapped with a colormap, should result in samples in each class and labelset that are similar to each other and different from the others. SSIM was chosen because the segments with glider self-noise tend to be easily distinguished by the naked eye, which should in theory reflect in the SSIM scores. To evaluate the dataset as a whole, we randomly selected 700 images from the dataset and ran image similarity calculations for every pair in the selection. We also evaluated image similarity within each class and labelset by randomly selecting up to 700 images for each labelset/class and comparing images in the same class/labelset with each other. We ran the same

process on the dataset generated by the data and code provided by Dufourq et. al. [10] (hence dubbed as Gibbon dataset) as a benchmark to compare to our dataset.

ℵ 3.4 Data Imbalance

One of the main problems inherent in this dataset is the data imbalance, caused by the gliders' quiet nature, which can potentially affect model performance [28]. Glider self-noise makes up approximately 10% of the collected PAM data, and the amount in dataset is likely to be less due to labeling errors arising from the automatic labeling process. There are several ways to help with data imbalance, such as sampling and data augmentation [17]. Several methods surveyed in Chapter 2 are used to improve model performance.

3.4.1 Sampling

Sampling methods are usually grouped into two categories: undersampling and oversampling [29]. Undersampling involves decreasing the number of samples and oversampling involves increasing the number of samples. It is important to not oversample too much, as it can lead to lots of emphasis on the same sample after lots of duplication, which can lead to overfitting [30]. Undersampling has the opposite problem; removing too many data points leaves the model with too few data to actually learn anything [29].

Sampling is a more complicated task for multilabel datasets since each image can have multiple labels. Deleting or copying images by class affects the number of images in another class. One way to deal with this is to balance all the labelsets and treat the problem like a multiclass balancing problem, so the model gets a balanced set of labelsets to work with [21]. However, if care is not taken, doing so could cause the classes to be unbalanced.

Multilabel random oversampling (ML-ROS) is an oversampling technique designed for multilabel datasets [23]. ML-ROS is based off the percent increase of the entire dataset, where the number of samples in each labelset is increased based on the ratio with the majority labelset [23]. In this paper, we used ML-ROS10 (oversampling with an overall 10% increase in dataset size) and ML-ROS20 (oversampling with an overall 20% increase in dataset size). Because of the heavy imbalance in the training dataset, only the None labelset ([0, 0, 0]) was undersampled.

There were 4 different combinations of sampling methods used in this thesis: Undersample, Undersample + Oversample, Undersample + ML-ROS10, and Undersample + ML-ROS20.

- Undersample: The biggest problem of the training dataset was the huge difference between the None labelset ([0, 0, 0]) and the other labelsets (roughly 9:1). We designed the Undersample method to alleviate this problem by splitting the dataset into two groups: samples with glider self-noise, and samples without glider self-noise. The majority group, the group that did not have glider self-noise, was undersampled so both groups had an equal number of samples.
- Undersampling + Oversampling: The Undersampling + Oversampling was named as such because it used the Undersample method described previously and oversampled each labelset to be equal to the largest labelset. This sampling method was focused on achieving balance across all labelsets in the most direct way. This sampling method runs the risk of overfitting if labelsets are heavily imbalanced, because minority labelsets need to be drastically increased, which could potentially cause the model to fixate on the duplicates.
- Undersampling + ML-ROS10: The Undersampling + ML-ROS10 method is similar to the Undersampling + Oversampling method, with the difference of the algorithm used for

oversampling. Instead of trying to achieve perfect balance across all labelsets, the ML-ROS algorithm with a 10% dataset increase was run to achieve more balanced labelset distributions without drastically increasing the size of the dataset.

- Undersampling + ML-ROS20: The Undersampling + ML-ROS20 method is similar to the Undersampling + ML-ROS10 method, except instead of running ML-ROS with a 10% increase, we ran ML-ROS with a 20% increase. The goal of this sampling method was to test the effect of a larger percentage increase on model performance.

3.4.2 Data Augmentation

Another way to deal with data imbalance is to increase the size of the dataset by creating new data points using data augmentation methods. In this thesis, we used three types of data augmentation for audio classification described in section 2.2: image augmentation, spectrogram augmentation, and signal augmentation [18].

Image Data Augmentation

Albumentations is a library of image augmentation methods and was used to create both the image augmentation pipeline [31]. The image augmentation methods used in this paper were:

- Horizontal Flipping: 50% chance to horizontally flip the image
- Vertical Flipping: 50% chance to horizontally flip the image
- Horizontal Flipping: 50% chance to horizontally flip the image
- Random Rotate: 50% chance to rotate the image by a randomly selected angle
- Random Rotate: 50% chance to rotate the image by a randomly angle
- Random Scale: 50% chance to resize the image by a random scale

- Gaussian Blur: 50% chance to apply Gaussian blur to the image with a randomly sized kernel

Spectrogram Data Augmentation

The *Albumentations* library was also used to implement the spectrogram augmentation pipeline. While *Albumentations* is primarily a library for image augmentations, it contains augmentations such as time masking and frequency masking (also called *XY* masking for non-spectrogram images), which are key techniques for spectrogram augmentation [31, 32]. These two augmentations, each with a random place and range of masking and a 50% probability of occurring, make up the spectrogram augmentation pipeline used in this paper.

Previous studies have reported image augmentation causing the model to perform more poorly than the baseline model, which is likely related to how realistically the augmentation method reflects typical spectrogram behavior [18, 33]. Many image augmentation methods used in these studies, such as flipping and rotating, were unrealistic to spectrogram behavior, and could result in a spectrogram for a completely different sound, or result in an image that is not a spectrogram at all. To test this idea, the spectrogram augmentation pipeline was limited to techniques also used for image augmentation, and image augmentation pipeline was purposefully chosen to be unrealistic to spectrogram behavior.

Audio Data Augmentation

Audio augmentation methods were implemented with the *Audiomentations* package [34]. Augmentations were selected based on the plausibility of the resulting spectrogram being an accurate representation of glider self-noise. Augmentations that added noise were not used

because they can potentially mask the broadband glider self-noise in the spectrogram. A random number of the following augmentations were chosen to create a new audio.

- Time Shift: 50% chance to shift the audio segment by a random time, with rollover
- Time Mask: 50% chance to mask a random section of the audio segment, with fade-in/out of the masked section
- Tanh Distortion: 50% chance to add a random tanh distortion to the audio segment
- Gain Change: 50% chance to multiply the audio segment by a random amplitude to change the loudness
- Resample: 50% chance to change the sampling rate of the audio segment by a random frequency

3.5 Models

Twenty deep learning models from Keras [35] were selected to find an optimal model for the dataset (see Table 3.1). Smaller sized models were chosen to take into account the limited hard drive space onboard the glider. We also selected larger models like ResNet101, VGG16, and VGG19 as a comparison against smaller models.

Table 3.1. Summary of models used in this thesis.

Model and Reference	Model Version	Model Size (MB)	Depth
EfficientNet [36], [37]	EfficientNetB0	29	132
	EfficientNetB1	31	186
	EfficientNetB2	36	186
	EfficientNetB3	48	210
	EfficientNetB4	75	258
	EfficientNetB5	118	312
	EfficientNetV2B0	29	-
	EfficientNetV2B1	34	-
ResNet [38]	ResNet50	98	107
	ResNet50V2	98	103
	ResNet101	171	209
MobileNet [39]	MobileNet	16	55
	MobileNetV2	14	105
VGG [40]	VGG16	528	16
	VGG19	549	19
DenseNet [41]	DenseNet121	33	242
	DenseNet169	57	338
Inception [42], [43]	InceptionV3	92	189
	Xception	88	81
NASNet [44]	NASNetMobile	23	389

3.6 Experiment Design

The experiments were organized into three parts:

1. Models trained from scratch on the baseline datasets
2. Models trained with pretrained weights from the ImageNet dataset on the baseline datasets
3. Models that performed well in the experiment phases 1 and 2 was run on the sub-datasets created from the 24-hour and the 48-hour datasets

All models were trained with 30 epochs, a batch size of 64, 10^{-6} learning rate, ADAM optimizer, binary cross entropy loss (BCE), an output size of 3 (one for each source of self-noise) and a sigmoid activation function for multilabel classification. BCE loss is a loss function used for binary classification problems, and equates to the following:

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)$$

where y_i is the binary label (0, 1) for the i^{th} observation, N is the number of observations, and $p(y)$ is the probability of the i^{th} observation being positive (1).

Training datasets were split with an 80-20 training-validation split. Models were compared by their validation accuracy, loss, recall, and precision, and models that performed well were selected for further hyperparameter tuning. Epochs with the best validation recall was saved. Models were primarily evaluated by recall and precision over accuracy because over 90% of the baseline dataset had no self-noise, meaning that model could achieve an accuracy of over 90% just by assigning [0, 0, 0] to everything, which makes recall and precision more indicative of how well a model is performing than accuracy.

Sub-datasets were created by applying the 4 different sampling methods on the baseline datasets, then replacing the duplicating process used for oversampling with the 3 different data augmentation methods to create 13 different sub-datasets for training, shown in Table 3.2.

Table 3.2. List of augmentations and whether it applies to the input type.

Sub-datasets		**Sampling Methods**			
		Undersampling	Undersampling + Oversampling	Undersampling + ML-ROS10	Undersampling + ML-ROS20
Data Augmentation Methods	**Spectrogram Augmentation**	∅	Undersampling + Oversampling with spectrogram augmentation	Undersampling + ML-ROS10 with spectrogram augmentation	Undersampling + ML-ROS20 with spectrogram augmentation
	Audio Augmentation	∅	Undersampling + Oversampling with audio augmentation	Undersampling + ML-ROS10 with audio augmentation	Undersampling + ML-ROS20 with audio augmentation
	Image Augmentation	∅	Undersampling + Oversampling with image augmentation	Undersampling + ML-ROS10 with image augmentation	Undersampling + ML-ROS20 with image augmentation

Chapter 4

Results

The following chapter describes the data distributions of the created datasets and sub-datasets; the results of the three parts of the experiments described in section 3.6; and a comparison between the created dataset and a dataset from a successful study as a potential way to explain model results.

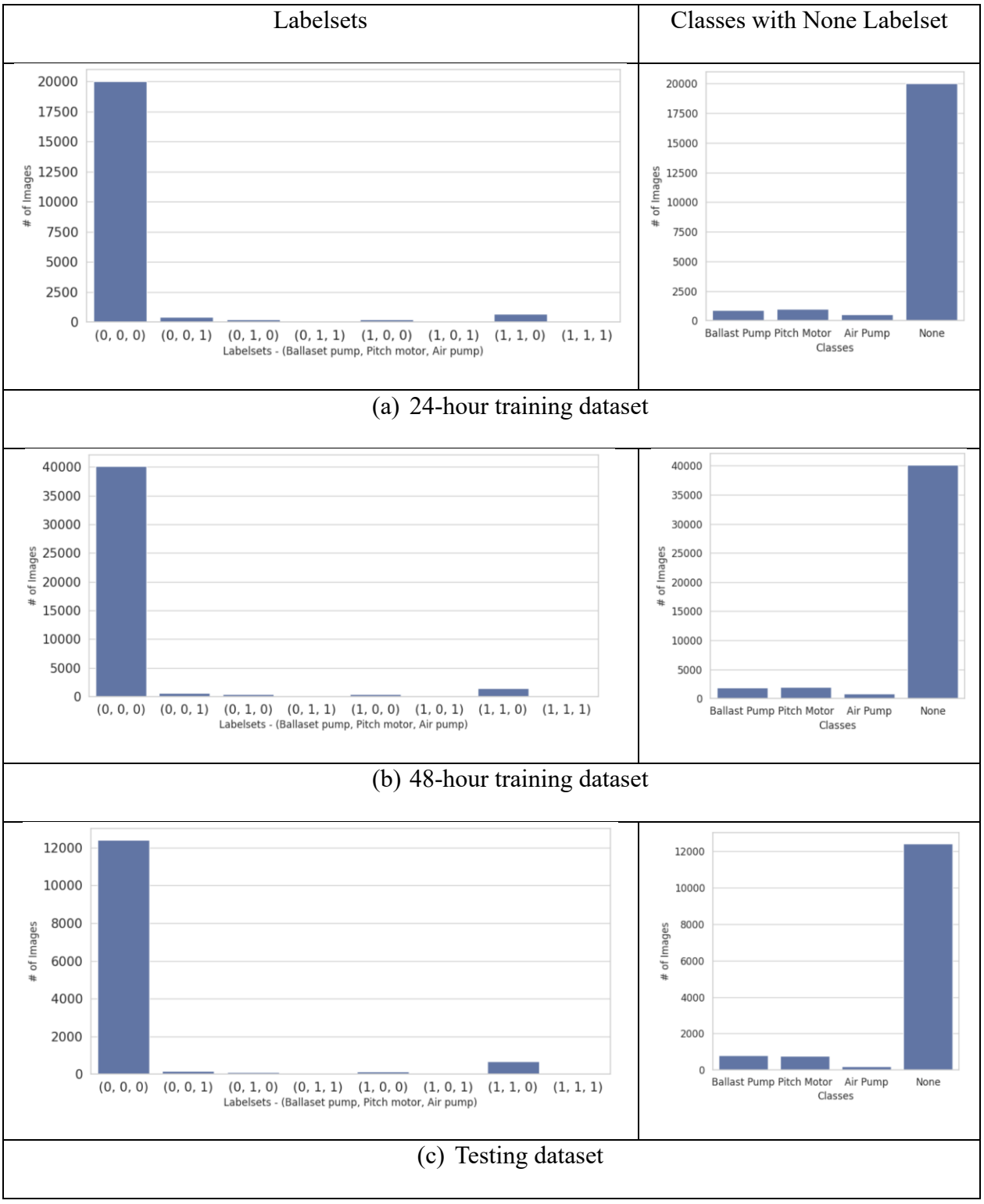
⁎ 4.1 Datasets

Two types of datasets were created for this work: baseline datasets and sub-datasets. Baseline datasets refer to the original training and testing datasets created from collected PAM data. Sub-datasets refer to the datasets created by applying various sampling and data augmentation methods to the baseline training datasets to deal with data imbalance issues.

4.1.1 Baseline Datasets

Three datasets were created from the mission data: two training datasets (24-hour and 48-hour) and one testing dataset. The 24-hour training dataset was created out of 24 hours of passive acoustic data from February 1st, 2024, and had 21,600 spectrograms. The 48-hour training dataset was created from data from 48 hours spanning February 1st and 2nd, 2024, and has 43,200 spectrograms. The testing dataset was created out of 15 hours of passive acoustic data from January 29th, 2024, and had 13,500 spectrograms. The resulting datasets had three binary classes, one for each source of self-noise (ballast pump, pitch motor, and air pump), and eight labelsets,

obtained by getting all the combinations of the classes. Figure 4.1 shows the class and labelset distribution of the training and testing datasets. The number of samples in the None labelset (0, 0, 0) greatly outnumbers the number of samples in any of the other labelsets and classes.



(a) 24-hour training dataset

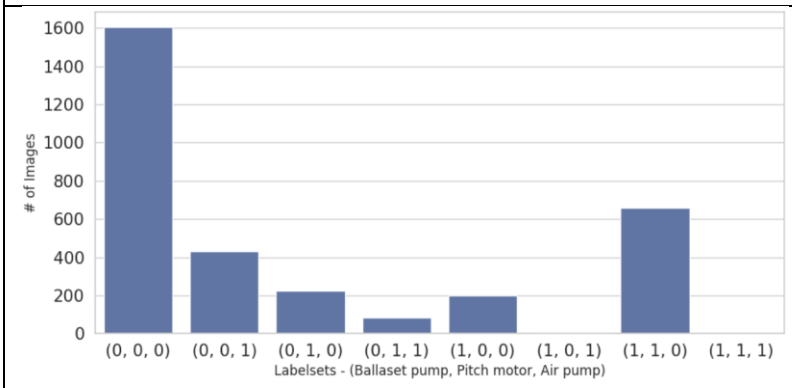
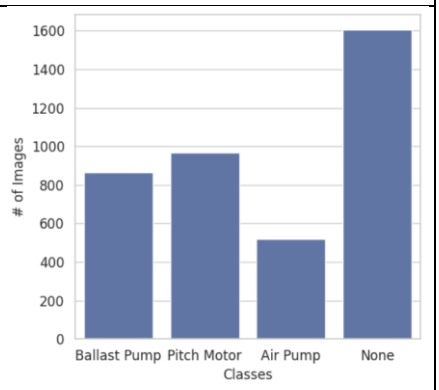
(b) 48-hour training dataset

(c) Testing dataset

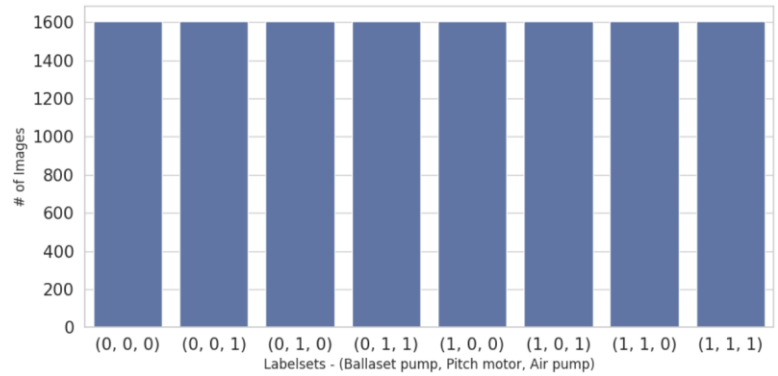
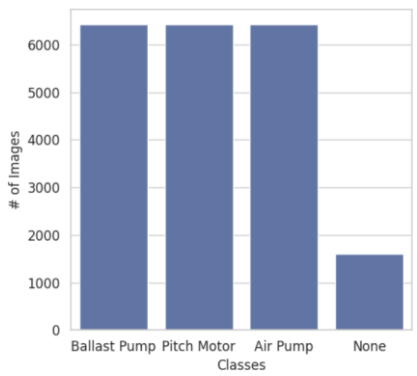
Figure 4.1. Class and labelset distributions for the 24-hour training dataset (a), 48-hour training dataset (b), and the testing dataset (c)

4.1.2 Sub-Datasets

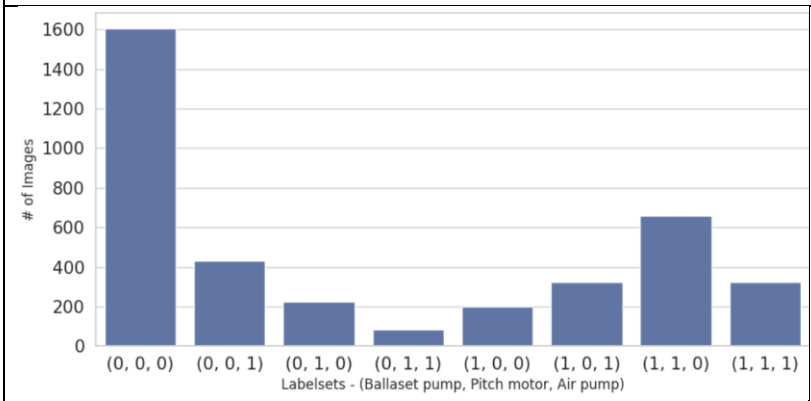
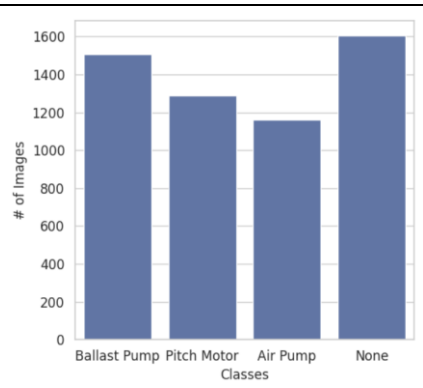
For each training dataset, thirteen sub-datasets were created, with three different classes and labelset distributions. Figure 4.2 and Figure 4.3 show the labelset and class distribution for each of the sampling methods used to create sub-datasets from the 24-hour training dataset and 48-hour training dataset. The Undersample sub-dataset reduces the None labelset by ~92% for both the 24-hour and 48-hour datasets so that the number of samples with no self-noise and the number of samples with self-noise are equal. One potential flaw with the Undersample sub-dataset is that the labelsets and classes are still very imbalanced. The Oversample + Undersample sub-datasets were created to make up for this flaw by oversampling all the labelsets with self-noise to match the size of the None labelset, resulting in a 300% sub-dataset size increase from Undersample sub-dataset. While the labelsets and classes in the Oversample + Undersample sub-datasets are completely balanced, 75% of the samples in the sub-datasets are either duplicated images or augmented images, which may cause the models trained on the Oversample + Undersample sub-datasets to be more susceptible to overfitting. The Undersample + ML-ROS10 and Undersample + ML-ROS20 sub-datasets (10% and 20% sub-dataset size increase from the Undersample sub-dataset) were created as a middle ground between the previous two types of sub-datasets described, achieving more balanced classes than the Undersample sub-dataset while maintaining the amount of duplicated or augmented samples in the sub-datasets at 20% and 40%.

Labelsets	Classes with None Labelset
	

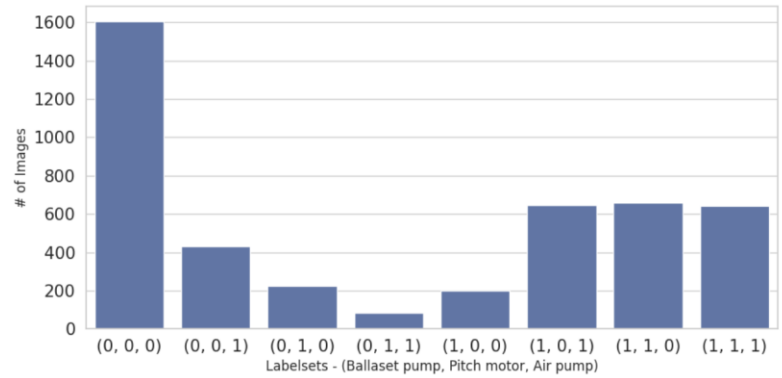
(a) Undersample sub-dataset: 3208 images

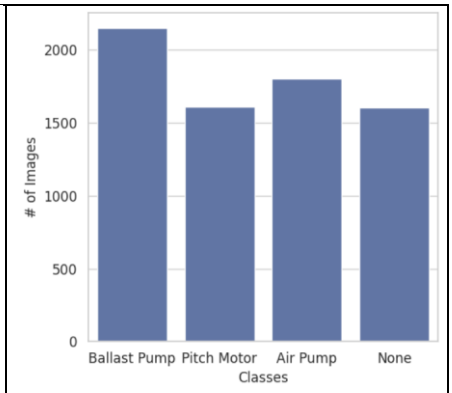
	

(b) Oversample + Undersample sub-datasets: 12832 images

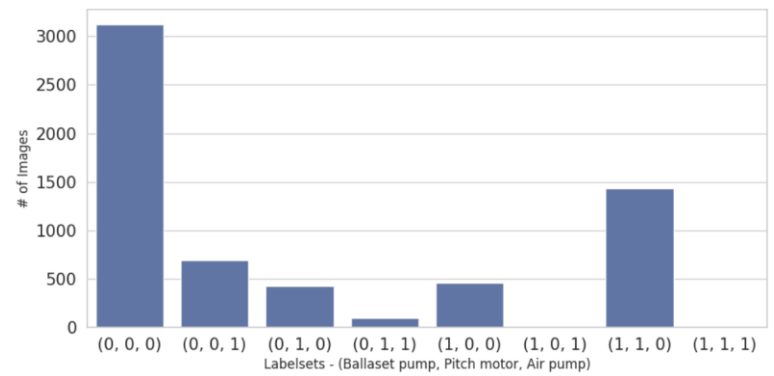
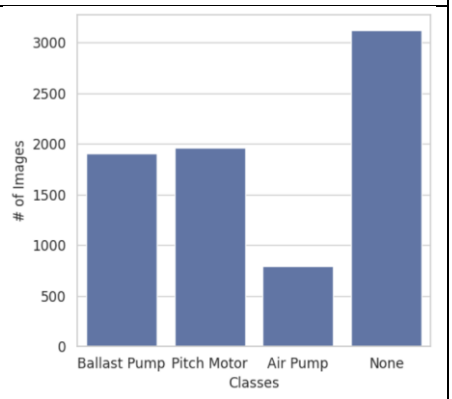
(c) Undersample + ML-ROS10 sub-datasets: 3848 images



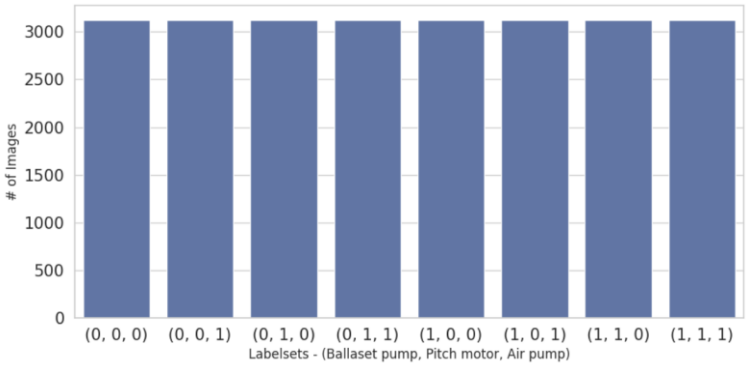


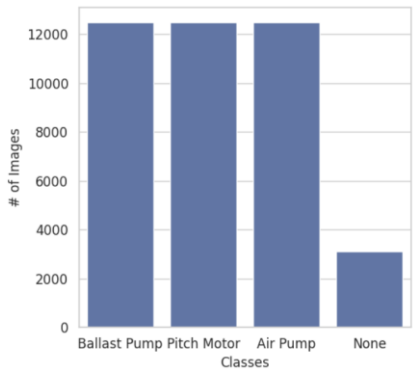
(d) Undersample + ML-ROS20 sub-datasets: 4490 images

Figure 4.2 Class and labelset distributions for the sub-datasets created from the 24-hour training dataset using the following sampling methods: Undersample (a), Oversample + Undersample (b), Undersample + ML-ROS10 (c), and Undersample + ML-ROS20 (d)

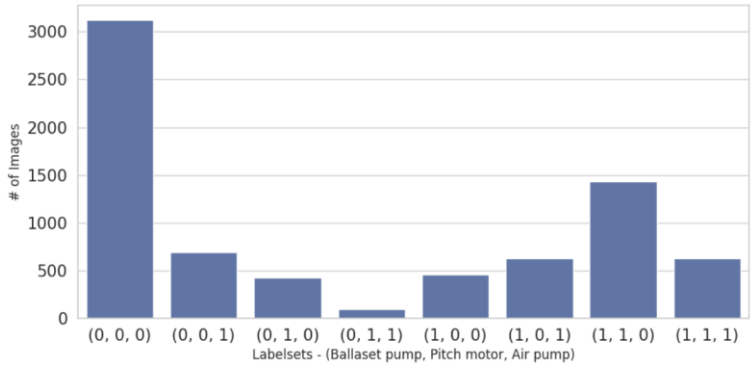
Labelsets	Classes
	

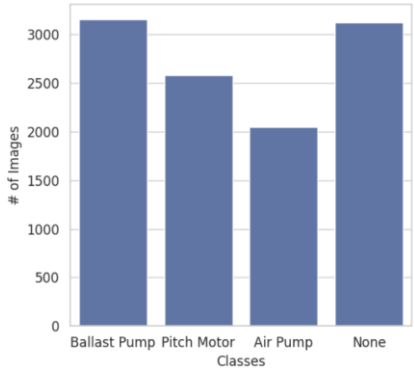
(a) Undersample sub-dataset: 6238 images



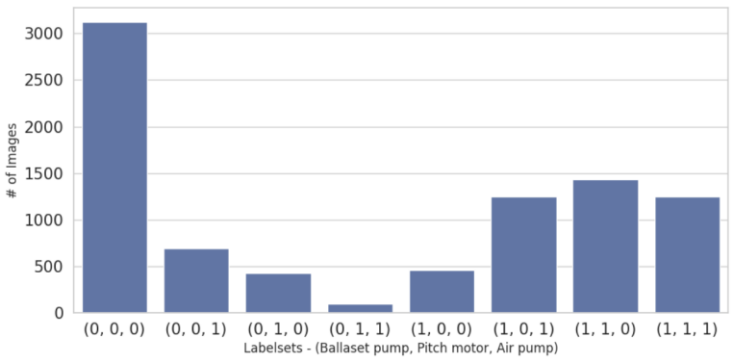


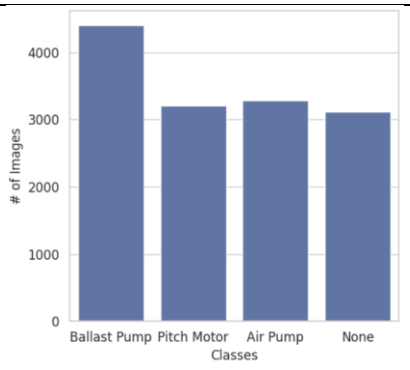
(b) Oversample + Undersample sub-datasets: 24952 images





(c) Undersample + ML-ROS10 sub-datasets: 7484 images





(d) Undersample + ML-ROS20 sub-datasets: 8732 images

Figure 4.3. Class and labelset distributions for the sub-datasets created from the 48-hour training dataset using the following sampling methods: Undersample (a), Oversample + Undersample (b), Undersample + ML-ROS10 (c), and Undersample + ML-ROS20 (d)

ℵ 4.2 Models Trained from Scratch

This section shows the results for twenty Keras models (Table 3.1) trained from scratch with randomly initialized weights on the 24-hour training dataset and the 48-hour training dataset. Models were compared primarily by validation recall and precision, which also taking validation loss into account to detect any overfitting. We expected the 48-hour training dataset to have better performing models overall due to the larger number of minority samples brought by the extra 24 hours of data.

4.2.1 24-hour Training Dataset

The validation results of training models with randomly initialized weights on the 24-hour training dataset are summarized in Figure 4.4. The *y* axis for the figures of validation loss is on a log scale due to the large range in validation loss. Most models had precision and recall scores around 1-2%, with the exception of NASNetMobile, which started with a much higher validation recall. However, NASNetMobile also had a consistently high validation loss and low validation accuracy while the validation recall was high. It is difficult to pick out the best performing model because there is no visible convergence in validation loss, and validation recall and accuracy tend to be very low.

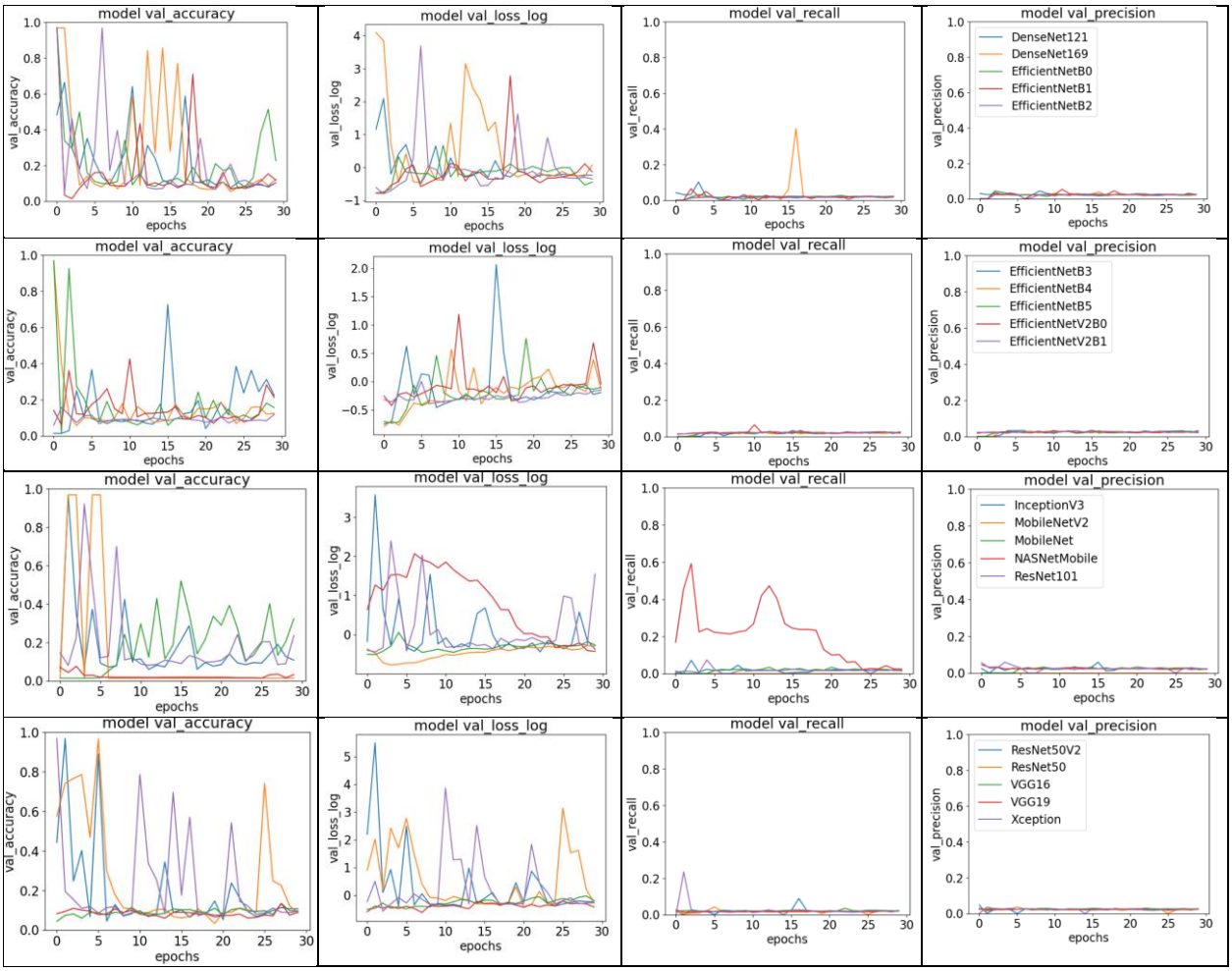


Figure 4.4. Validation results for models with randomized initial weights trained with the 24-hour training dataset

4.2.2 48-hour Training Dataset

Figure 4.5 shows the validation results for the 20 models trained on the 48-hour training dataset. Compared to the models trained on the 24-hour dataset, the validation recall and precision roughly doubled, increasing to stagnate at around 3-4%, which aligned with our expectations. NASNetMobile achieved a noticeably higher validation recall, but it coincides with a spike in the validation loss. DenseNet169 and EfficientNetB2 also shows a small increase in

validation recall and precision, but the increase for both coincides with a spike in the validation loss. Despite the slight increase in performance, there is still no obvious best performing model among the twenty models trained.

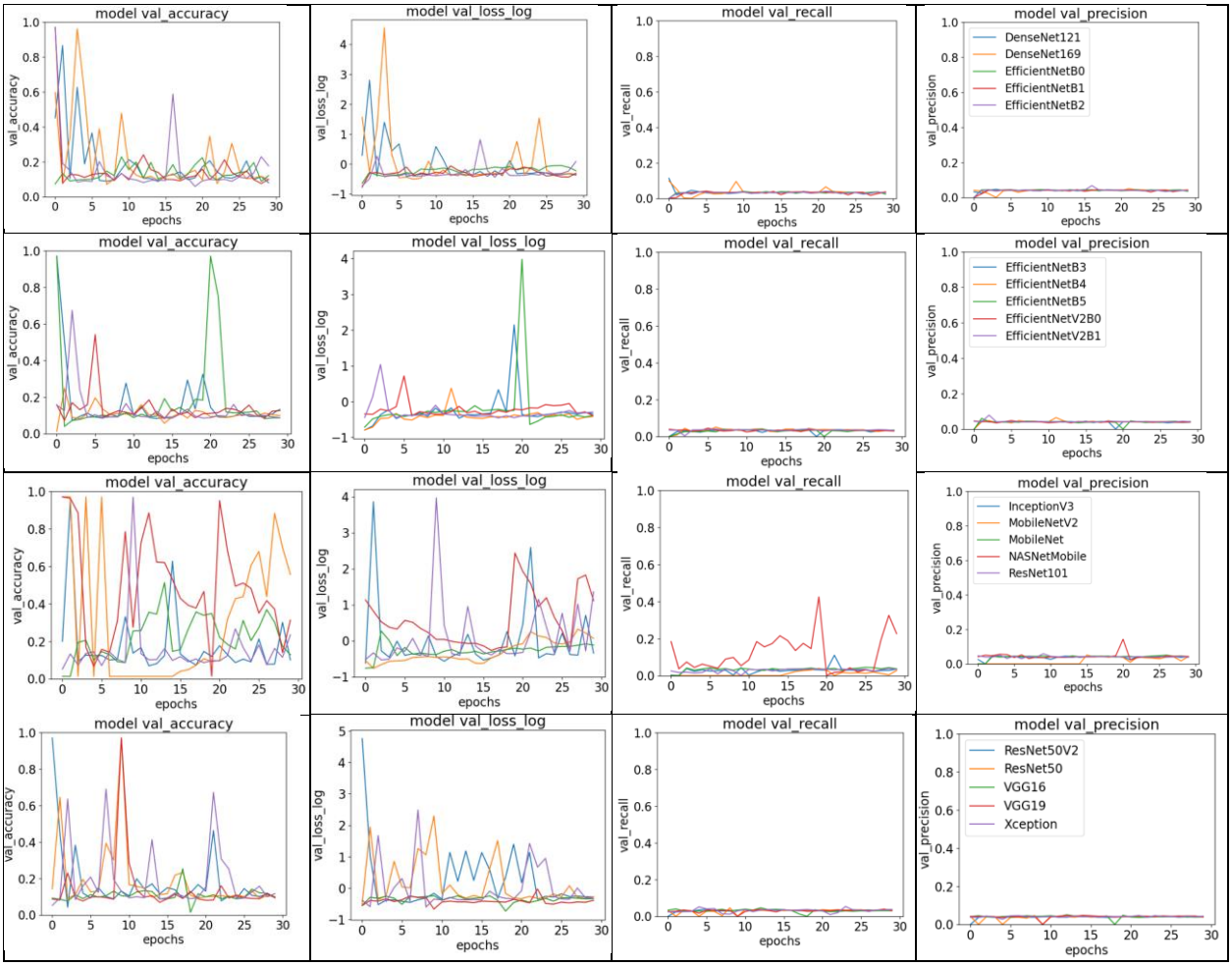


Figure 4.5. Validation results for 20 Keras models trained with the 48-hour baseline dataset

4.3 Models Trained with Pretrained Weights

To potentially achieve better performance than the models trained from scratch, the same twenty Keras models (Table 3.1) were trained on the same baseline training datasets with

pretrained ImageNet weights. The same metrics were used for this experiment as with the models trained from scratch. We did not expect higher performance due to the fact that models were trained on a dataset of images of everyday objects instead of spectrograms. However, because the performance of the models trained on scratch was so poor, there is a chance that pretrained ImageNet weights will help improve model performance.

4.3.1 24-hour Training Dataset

The validation results for the pretrained models trained on the 24-hour baseline training dataset are shown in Figure 4.6. Like with the 24-hour dataset models that were trained from scratch, the validation recall and precision hover at around 1-2% for most models, with no noticeable improvement in validation recall or precision compared to the models trained from scratch. Compared to the models trained from scratch, the validation loss for these models appeared to have a small overall upward direction, which mark the pretrained models as potentially worse than the models trained from scratch. Another aspect different from the models trained from scratch was the ability to distinguish better performing models. We found that MobileNet and ResNet50 ran relatively well compared to the other models. Compared to the other models, MobileNet and ResNet50 had relatively low and decreasing losses with no spikes in loss while also being able to achieve high accuracy at some epochs. While NASNetMobile achieved higher validation recall than MobileNet and ResNet50, the validation loss was higher than both models. EfficientNetB3 also had spikes with high validation recall, but those spikes coincided with spikes in validation loss.

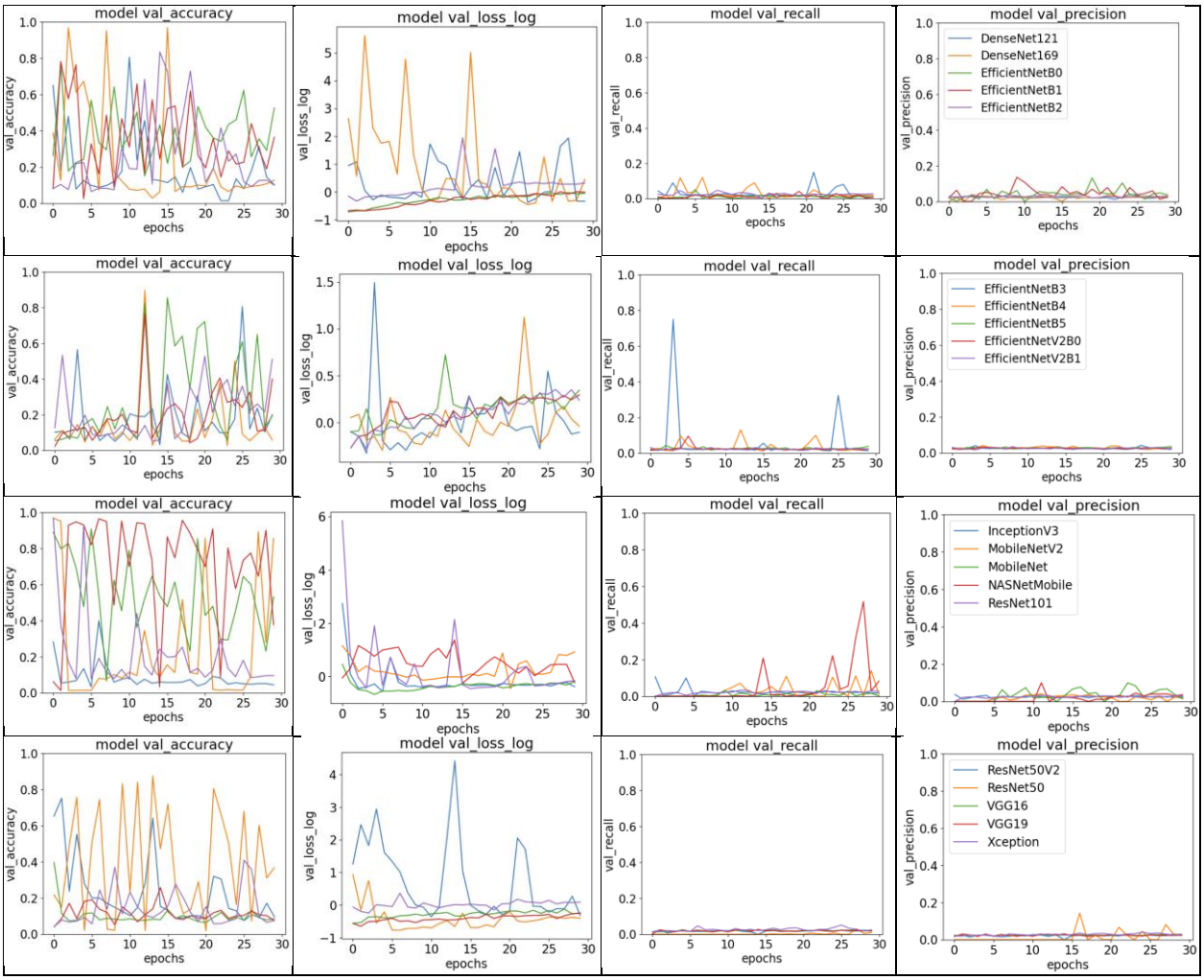


Figure 4.6 Validation results for 20 Keras models trained with the 24-hour baseline dataset. Each row shows the results for 5 different models, and each column shows the results for one metric.

4.3.2 48-hour Training Dataset

Figure 4.7 summarize the validation results for the pretrained models trained on the 48-hour baseline training dataset. The validation recall and precision stayed at around 3-4% for most of the training epochs, similar to the respective models trained from scratch. While these models also have a general upward trend in validation loss, there are less models with over two-digit loss

compared to the models trained from scratch. MobileNetV2 was found to have performed relatively well for this dataset. MobileNetV2 was able to reach high validation accuracy and a relatively high validation recall without having large increases in validation loss at the same time. While EfficientNetB5 also achieves a high validation recall at one epoch, it also has a five-digit loss in the same epoch, which makes it appear to perform more poorly than MobileNetV2. NASNetMobile has spikes in validation recall without corresponding spikes in validation loss, but the average is validation loss higher than that of MobileNetV2.

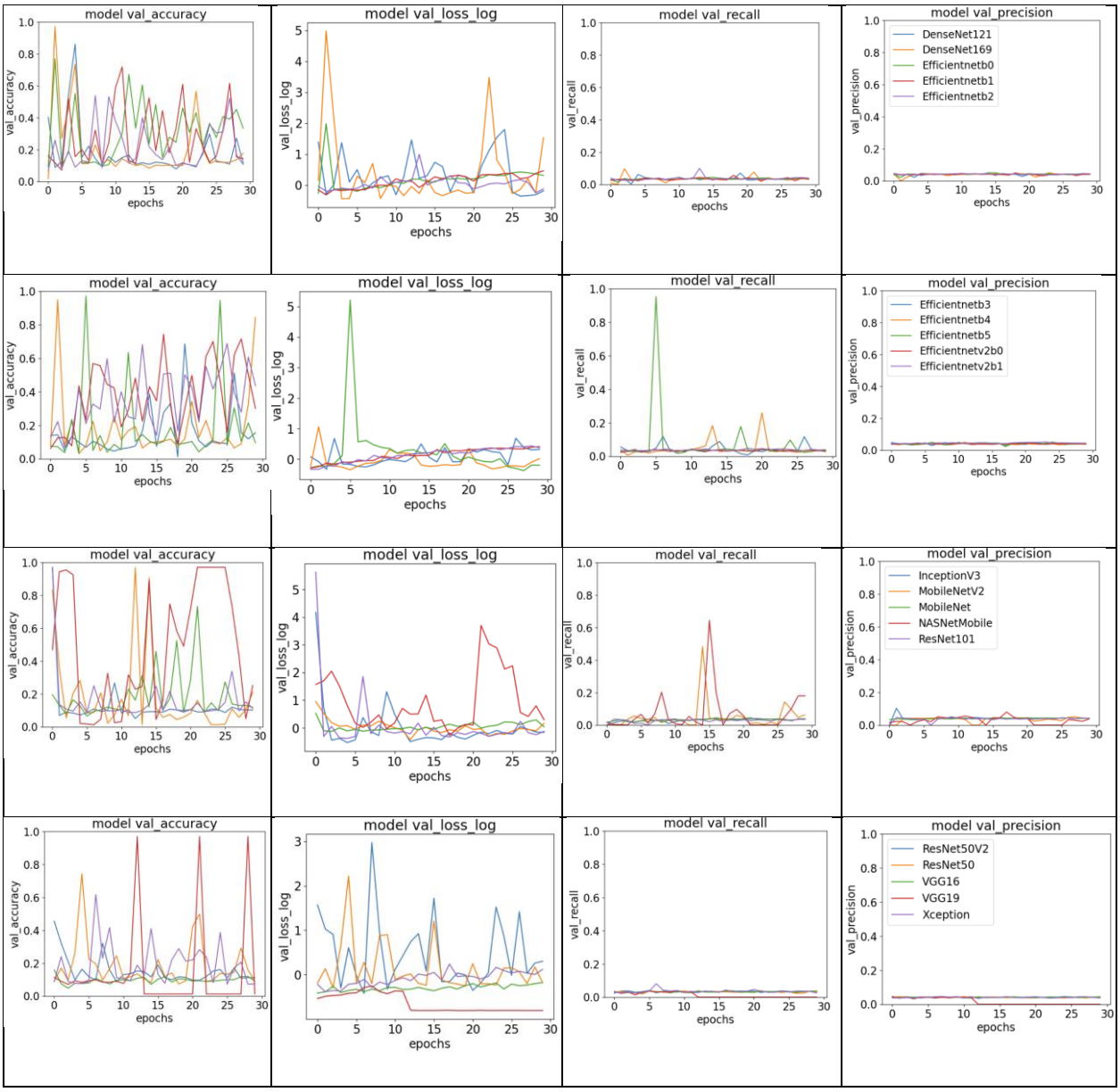


Figure 4.7. Validation results for 20 Keras models trained with the 48-hour baseline dataset

4.4 Models Trained on Sub-Datasets

One of the likely reasons for the poor performance by the models trained on the baseline datasets is the large data imbalance. Other studies that achieved high model performance [10, 15, 16, 17] had equal numbers of samples among their classes, while one of the labelsets in our

training dataset has over 90% of samples with the other seven labelsets unevenly splitting the remaining 10%. To test whether this was the case, we picked the best performing model for each baseline training dataset and trained the models on the more balanced sub-datasets to observe any improvement in model performance. We used the same metrics with the same considerations as the previous experiments to maintain consistency. Overall, we had several expectations for this experiment: 1) sub-dataset with more balance across labelsets and classes to perform better; 2) sub-datasets with higher amounts of duplicates and augmented images to perform worse; 3) sub-datasets with different data augmentation methods to perform differently from each other; and 4) 48-hour sub-datasets to perform better than 24-hour sub-datasets, reflecting the results in the previous two experiments.

4.4.1 24-hour Sub-datasets

We selected the pretrained ResNet50 model (baseline ResNet50 model) as the best model trained on the 24-hour dataset to test the 24-hour sub-datasets and compared the performance to the baseline ResNet50 model (Fig. 4.8). There was a large increase in validation recall and precision for all sub-datasets compared to the baseline ResNet50 model. However, there was also a large increase in validation loss for all the sub-datasets. Contrary to our expectations, there was no noticeable difference in the performance between the various types of data augmentation methods. Models trained on sub-datasets with the Oversampling + Undersampling sampling method, regardless of data augmentation method, achieved ~50% validation precision and recall, higher than baseline and the models trained on the other sub-datasets. However, the corresponding validation loss tends to spike up and down, unlike that of the models trained on the sub-datasets using ML-ROS10 and ML-ROS20, which start at a large number and curve downwards.

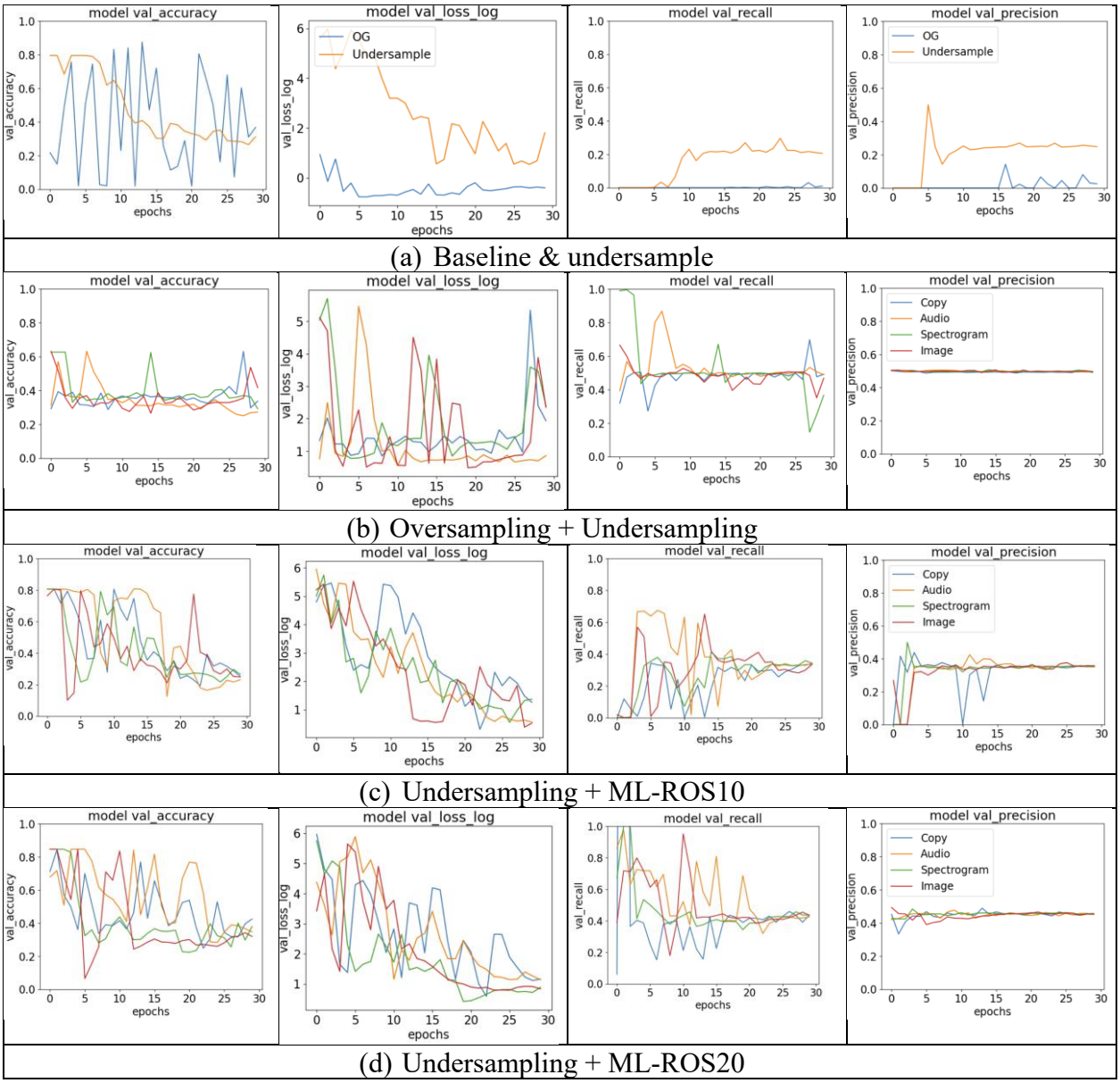


Figure 4.8. Validation results for ResNet50 models trained on the sub-datasets of the 24-hour baseline dataset. The first row shows the baseline dataset with the Undersample sub-dataset, the 2nd row shows results for the sub-datasets with Oversampling + Undersampling, the 3rd row shows results for the sub-datasets with Undersampling + ML-ROS10, and the 4th row shows results for the sub-datasets with Undersampling + ML-ROS20.

4.4.2 48-hour Sub-datasets

We tested the 48-hour sub-datasets on the pretrained MobileNetV2, which we found performed best among all the models trained on the 48-hour training dataset (Fig. 4.9). MobileNetV2 achieved less consistent validation recall and similar validation precision as the ResNet50 models trained on the 24-hour sub-datasets. Like the ResNet50 models, there was no noticeable way to rank the performance of the different data augmentation methods. Unlike the ResNet50 models in the previous section, the validation loss is significantly lower for all models trained on the 48-hour sub-datasets, highlighting the benefit of having more samples. However, there is a less obvious downwards curve for the validation loss and more spikes and upwards trajectories, which make it more difficult to compare the effect of the 24-hour sub-dataset and the 48-hour sub-datasets.

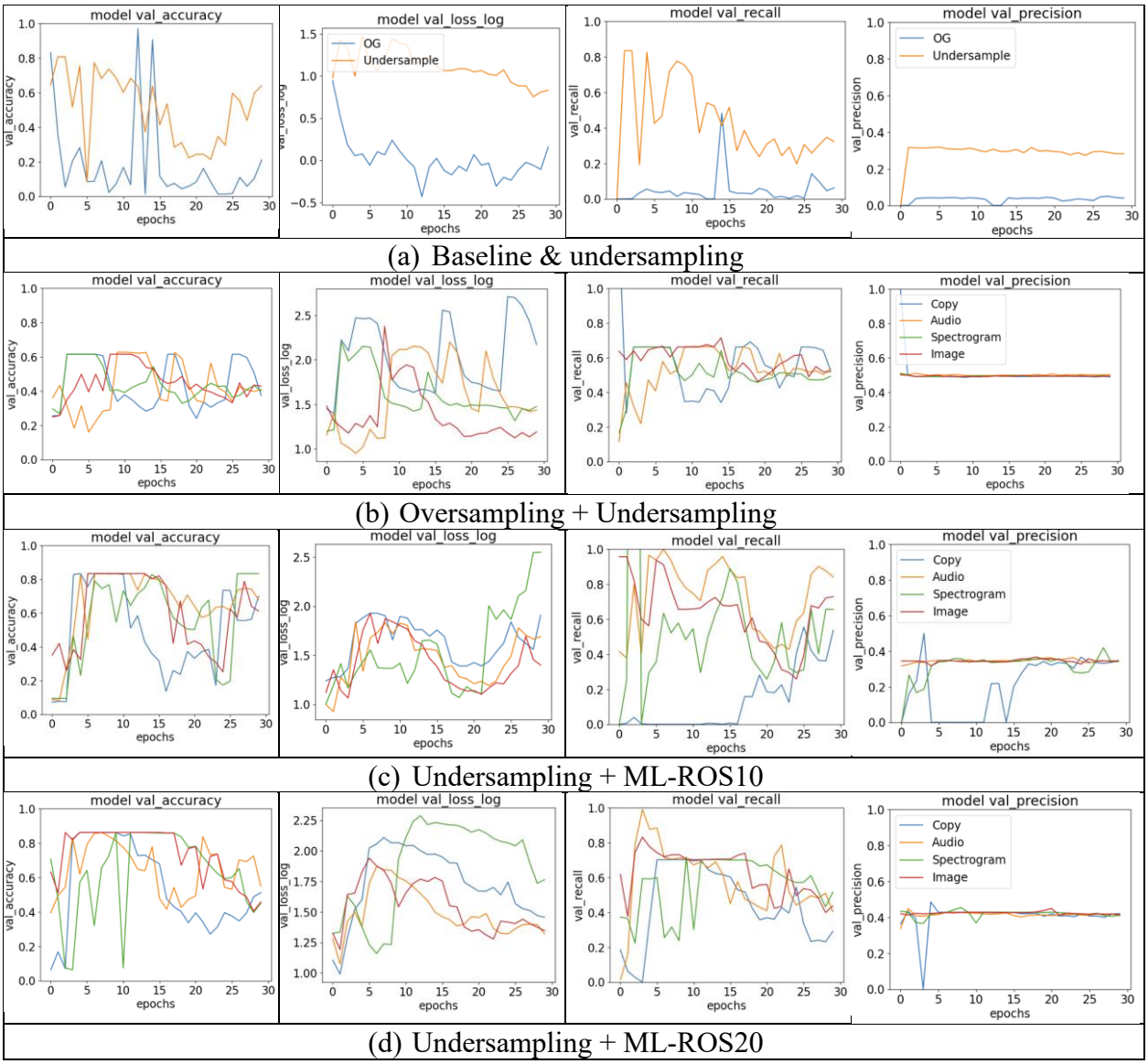


Figure 4.9. Validation results for MobileNetV2 models trained on the sub-datasets of the 48-hour baseline dataset. The first row shows the baseline dataset with the Undersample sub-dataset, the 2nd row shows results for the sub-datasets with Oversampling + Undersampling, the 3rd row shows results for the sub-datasets with Undersampling + ML-ROS10, and the 4th row shows results for the sub-datasets with Undersampling + ML-ROS20.

Consistent increase in validation recall and precision for both 24-hour sub-datasets and 48-hour sub-datasets suggest that having balanced labelsets and classes contributes more to model performance than the number of samples given to the model for training. The consistent performance in validation recall and precision over other sub-datasets by the Oversampling + Undersampling sub-datasets of both the 24-hour and 48-hour dataset indicate that having more balanced labelsets and classes improves model performance. The worse validation recall for the 48-hour Oversample + Undersample than the respective 24-hour sub-dataset suggest that numbers of duplicated and augmented images affect the validation loss more than the percentage of duplicated and augmented images. This is supported by the downwards trend of validation loss in the 24-hour Undersample ML-ROS10 and Undersample ML-ROS20 sub-datasets not present in the respective 48-hour datasets.

To confirm the theories suggested in the previous paragraph, we chose to test 20 pretrained models (to match the pretrained MobileNetV2 used for the sub-dataset experiment) on the two 48-hour sub-datasets (Undersample and Undersample + ML-ROS10). We chose the 48-hour Undersample sub-dataset to test whether balanced labelsets and classes applied to all models, and the 48-hour Undersample + ML-ROS10 sub-dataset to test whether choosing a different model might decrease the high validation loss. We chose not to test a sub-dataset that used a data augmentation technique because of the consistent lack of difference in performance.

The results for the models trained on the Undersampled 48-hour dataset are shown in Figure 4.10. Each model was compared to the baseline pretrained datasets in Figure 4.7. Having more balanced classes seemed to consistently improve the model performance regardless of the model, because validation recall for all models increased to around 25% and the validation precision increased to around 25-30%, while there is no obvious increase nor decrease in validation loss.

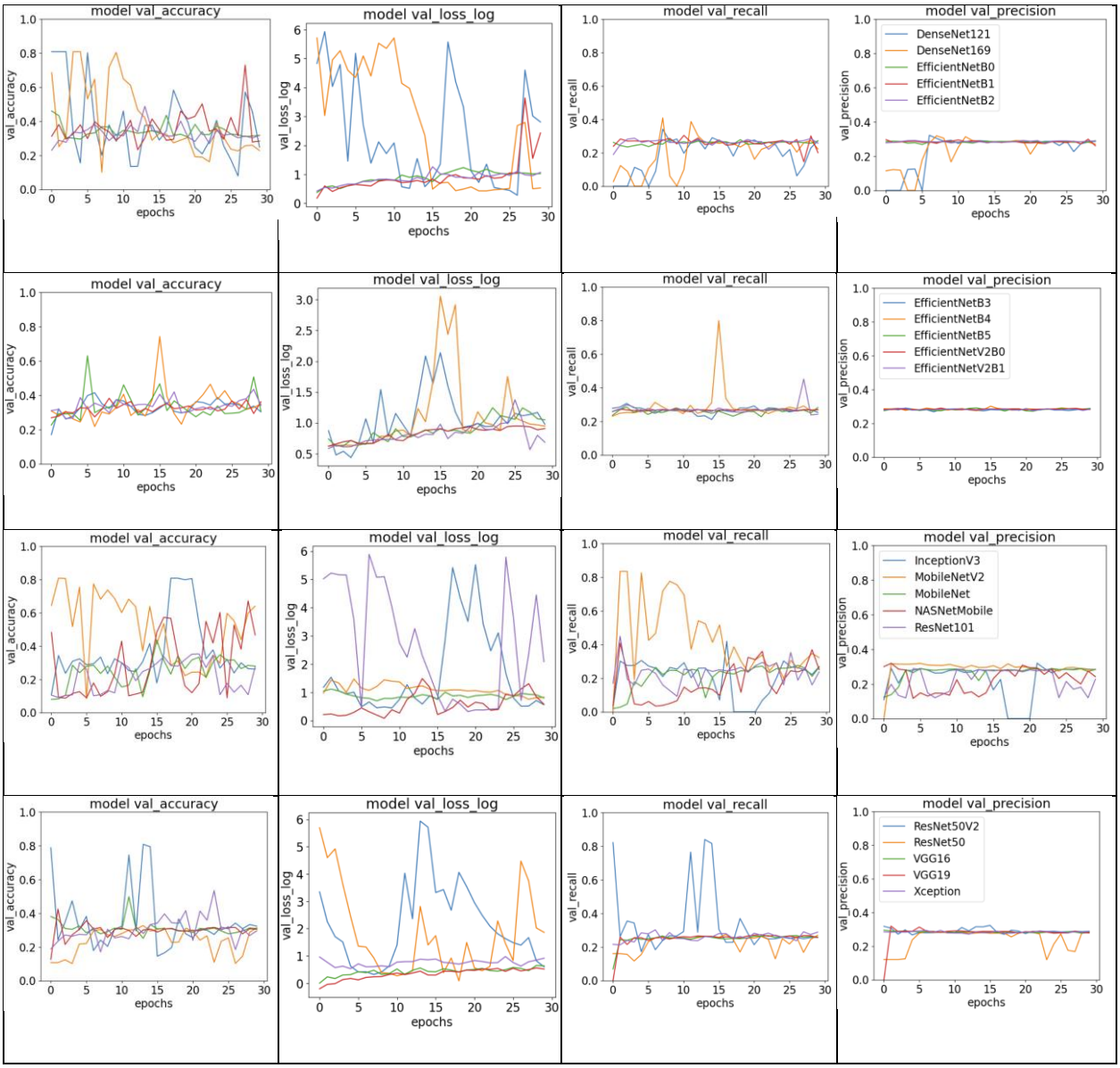


Figure 4.10. Validation results for 20 Keras models trained with the Undersampled 48-hour dataset

Figure 4.11 shows the results for the 20 pretrained models for the Undersampled + ML-ROS10 48-hour dataset. The results of these models were compared to the baseline results in Figure 4.7 and the Undersample sub-dataset results in Figure 4.10. Models trained on this dataset

had an improved overall validation recall of around 35% and validation precision of around 35-37% (~10% increase from the Undersample sub-datasets and ~30% increase to the baseline) with more obvious upwards trends in validation loss when compared to both the baseline and the Undersample sub-dataset models, likely caused by the duplicated images when oversampling.

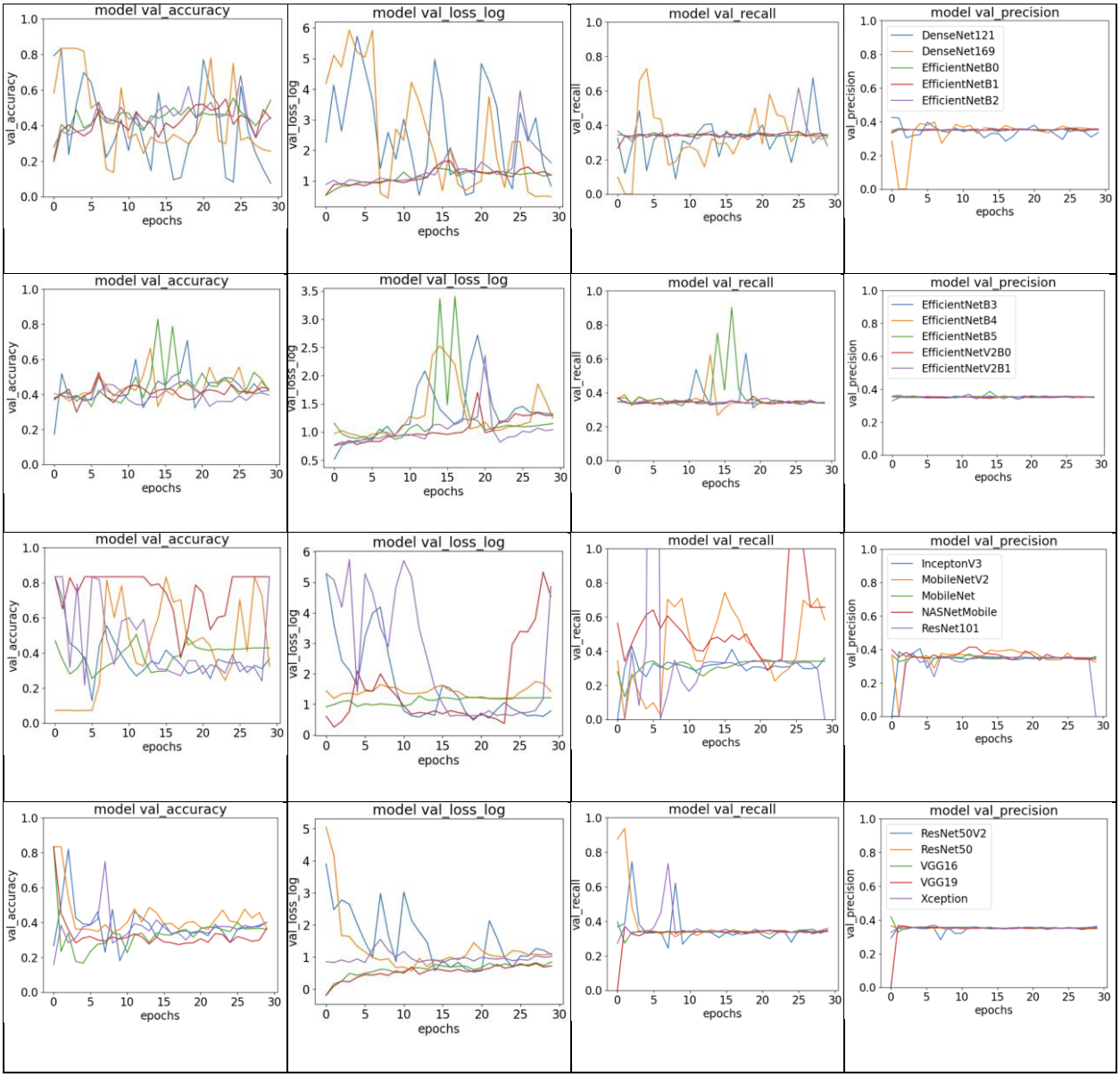


Figure 4.11. Validation results for 20 Keras models trained with the Undersampled + ML-ROS10 48-hour dataset

After confirming the effects of dataset balance and oversampling with large percentage increments, we tested whether model performance could be further increased with hyperparameter tuning. Due to time and resource constraints, we selected two hyperparameters with relatively small search spaces: optimizers and batch size. We chose the MobileNetV2 model trained on the 48-hour Undersample sub-dataset (which we found had the highest validation recall and precision with the smallest validation loss) to test ten different optimizers from Keras and six different batch sizes.

The results of experiments with optimizers are shown in Figure 4.12. Each models had the were run with 30 epochs, batch size 64, 224 × 224 image size, and 0.001 learning rate. Of the ten optimizers tested (Adadelta, Adagrad, AdamW, Adam, Adamax, Ftrl, Lion, Nadam, RMSprop, SGD), we found Adam to be the best optimizer with these default hyperparameters, having relatively high validation recall and precision, and relatively low and decreasing validation loss.

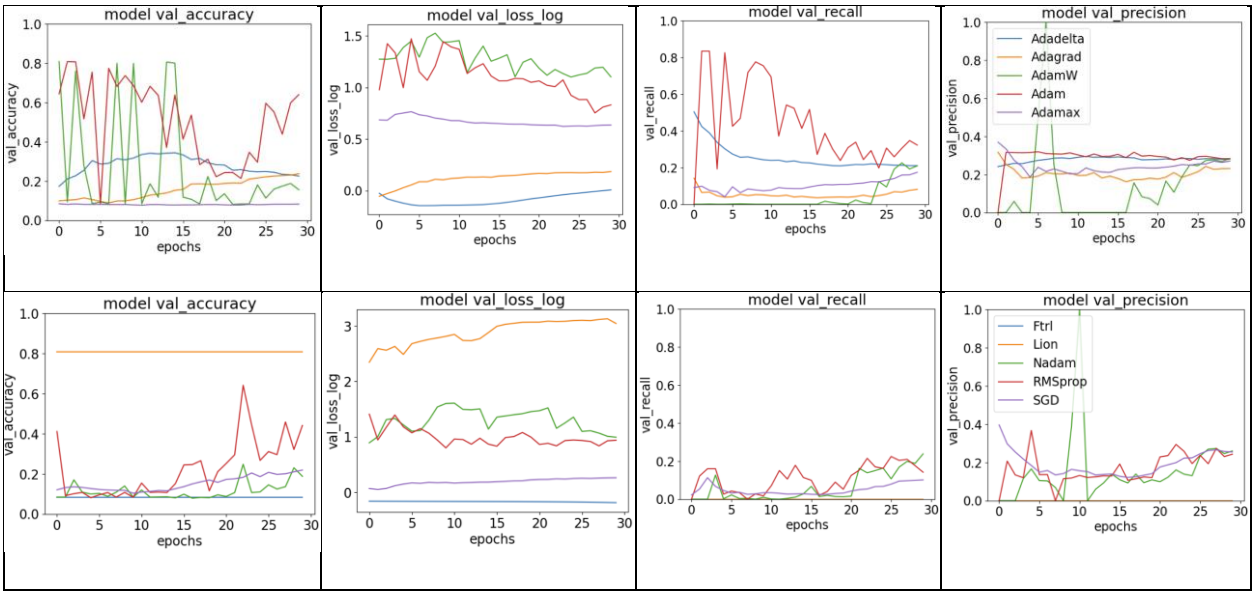


Figure 4.12. Validation results for optimizer testing of the best performing model trained on the Undersampled 48-hour baseline dataset

Figure 4.13 shows the results for the batch size experiments. The default hyperparameters were 30 epochs, Adam optimizer, 224 × 224 image size, and 0.001 learning rate. The batch size of 64 appeared to perform best, as it achieved the highest validation recall and relatively high validation accuracy and precision without having large spikes in validation loss.

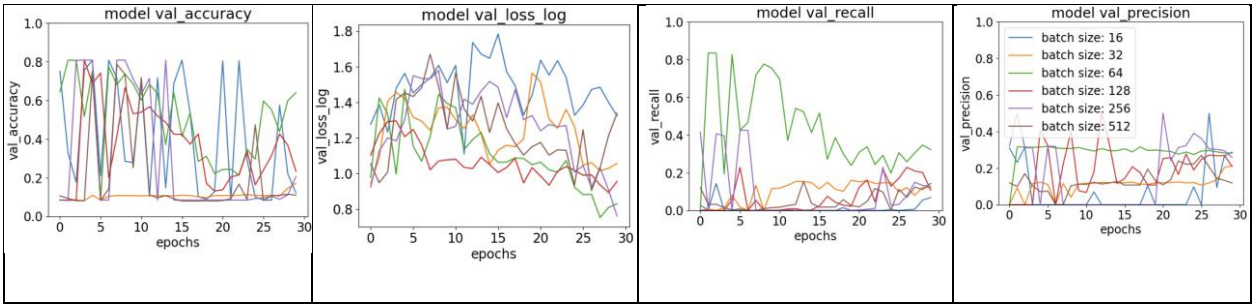


Figure 4.13. Validation results for batch size testing of the best performing model trained on the Undersampled 48-hour baseline dataset

In summary, we tested twenty different Keras models in various conditions with both the baseline datasets and sub-datasets. We found that although increasing the size of the dataset could help increase performance, dataset balance and percentage increase caused by overfitting and data augmentation affected model performance more. The models with the highest validation recall and accuracy were the ResNet50 and MobileNetV2 models trained on the 24-hour and 48-hour Oversample + Undersample sub-datasets with ~50% validation recall and precision, which had very high validation losses. The models with the best validation loss were the baseline ResNet50 and MobileNetV2 but had very low (1-4%) validation recall and precision.

ℵ 4.5 Image Similarity Score for Baseline Datasets

The poor performance of all models could potentially be explained by high image similarity across classes and labelsets. Image similarity measurements were calculated using both the histogram method and the SSIM method. The image similarity scores for the 48-hour training dataset were compared to those for the Gibbon dataset [10]. The Gibbon dataset was chosen because the authors of the research were able to train a 2D CNN and achieve good results with the dataset [10]. Furthermore, the data and code were available with no deprecated code, which allowed us to recreate the dataset used in the research [10]. The Gibbon dataset was organized into two classes (positive and negative for Gibbon calls), with each class having an equal number of samples [10]. The 48-hour training dataset was chosen out of the 3 datasets created because it was the largest and therefore had more samples in the minority labelsets.

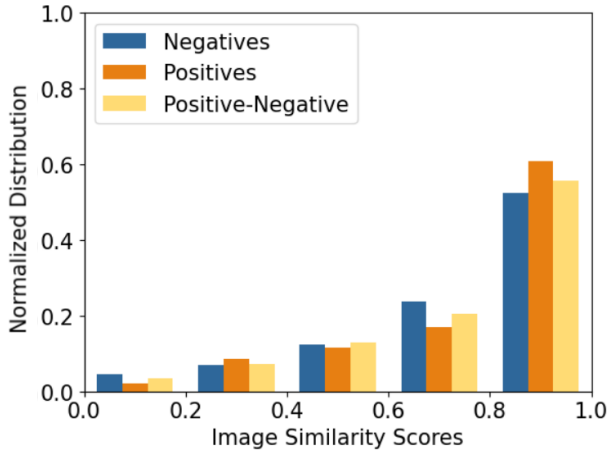
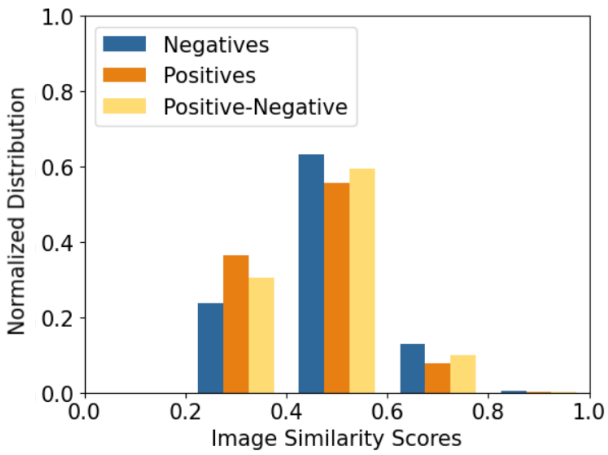
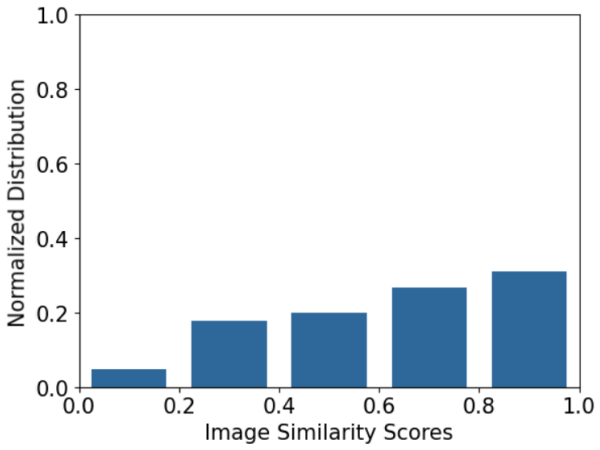
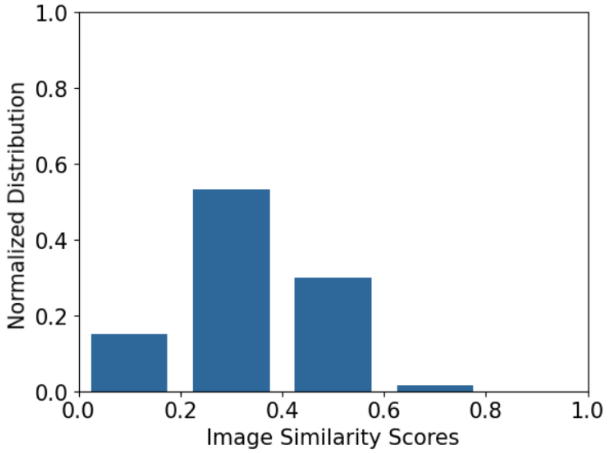
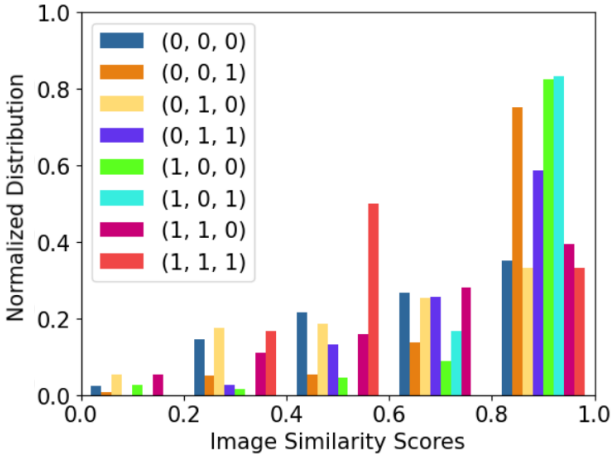
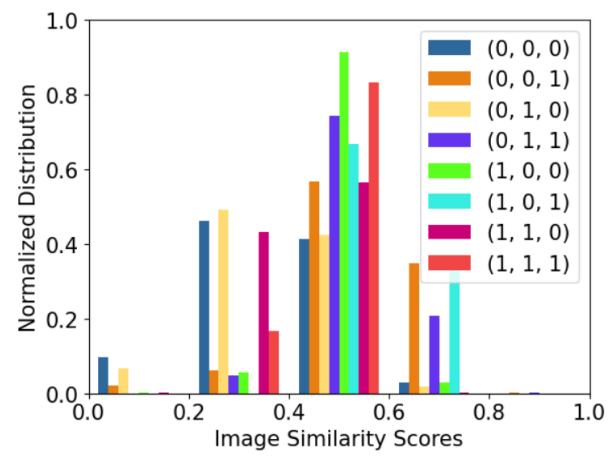
We expected image similarity scores to be low for intra-class comparisons in 48-dataset because each class (ballast pump, pitch motor, air pump) can have samples that belong to other classes, which would cause the samples that only belonged in one class to look different. We also

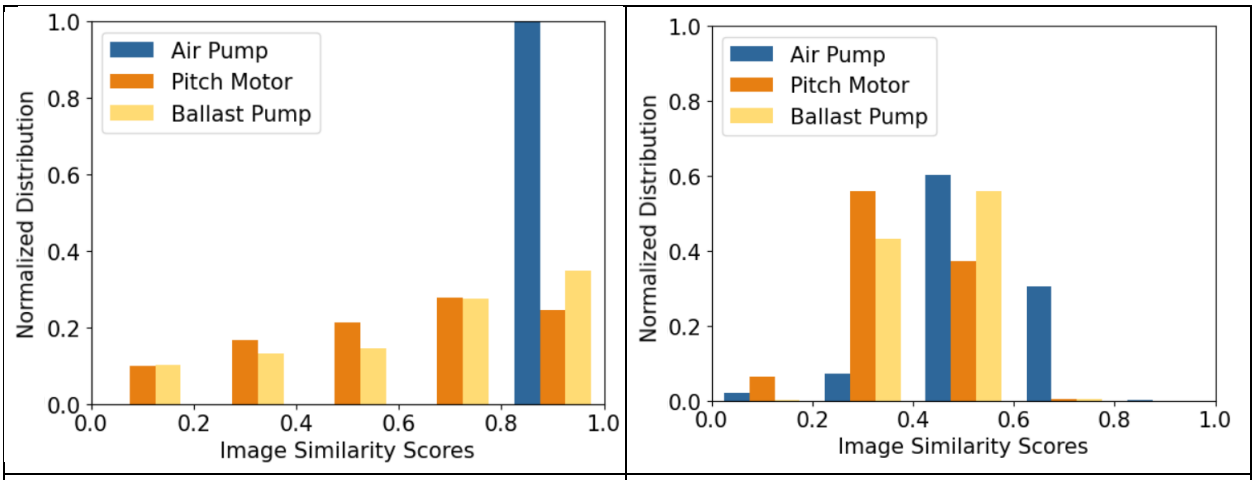
expected image similarity scores for samples in the None labelset to be low, because of the variety of other different features (boat noises, water surface sounds, etc.) present. Our reasoning was that the low image similarity scores inside classes and labelsets would cause the dataset to be harder for the model to learn. On the other hand, we expected datasets that were used to achieve high model performance like the Gibbon dataset to high image similarity scores within classes and low image similarity scores across different classes.

The results of the image similarity calculations are shown in Figure 4.14. All the graphs show in Figure 4.14 have the labelsets in the *x* axis and the distribution of image similarity scores calculated using the probability mass function in the *y* axis. Figure 4.14a and 4.14b shows the image similarities of the Gibbon dataset and was used as a standard to measure our dataset against. Figure 4.14c and 4.14d shows the image similarities of the overall 48-hour dataset. Figure 4.14e and 4.14 f shows the image similarities for each labelset in the 48-hour dataset. Figure 4.14g and 4.14 h shows the image similarities for each class in the 48-hour dataset.

To compare the image similarity scores for intra-class comparisons with inter-class comparisons in the Gibbon dataset, we separated the comparisons into three categories: 1) comparisons between two samples with positive Gibbon calls; 2) comparisons between two samples with negative Gibbon calls; and 3) comparisons with one positive sample and one negative sample. Contrary to our expectations, the class of the images being compared did not appear to significantly affect the distribution of image similarity scores (for both SSIM and histogram-based) in the Gibbon dataset. Intra-class and inter-class comparisons had similar proportions of image similarity scores, which indicates that SSIM and histogram-based image similarity were not important factors that the model used for detection and classification.

We were unable to group image similarity scores for the 48-hour dataset in the same way as the Gibbon dataset due to the larger number of classes and labelsets. We instead created three different views for the 48-hour dataset: 1) overall image similarity scores, regardless of class or labelset; 2) only intra-labelset comparisons; and 3) only intra-class comparisons. We chose these views because our predictions about image similarity were centered around comparisons within labelsets and within classes, and not across. Furthermore, with three classes and eight labelsets, displaying inter-labelset and inter-class comparisons would be complicated and difficult to understand. Image similarity scores for comparisons within the None labelset were relatively lower compared to other labelsets, as predicted. Image similarity scores for the [1, 0, 1] labelset (ballast pump and air pump) were high, likely because there were only four samples (out the 43,200 samples of the 48-hour dataset) which had similar features. Image similarity scores within classes were fairly low, with the exception of the air pump class, likely due to the manual annotation of air pump self-noise and the distinctive stripes in the spectrograms of air pump self-noise. Samples with pitch motor self-noise were the lowest, likely because of their vulnerability to being missed by the automatic labeling algorithm and its co-occurrences with other glider motors.

Histogram-Based Image Similarity	SSIM
	
(a)	(b)
	
(c)	(d)
	
(e)	(f)



(g) (h)

Figure 4.14. Image similarity distribution for 700 randomly selected images in the Gibbon dataset using the histogram method (a) and the SSIM method (b); image similarity distribution for 700 randomly selected images in the 48-hour training dataset using the histogram method (c) and the SSIM method (d); image similarity distribution for up to 700 randomly selected images each labelset of the 48-hour training dataset using the histogram method (e) and the SSIM method (f); image similarity distribution for up to 700 randomly selected images each class of the 48-hour training dataset using the histogram method (g) and the SSIM method (h)

To measure the impact of image similarity on model performance for our 48-hour dataset, we averaged the SSIM and histogram-based image similarity scores for each labelset; plotted them with the precision and recall for each labelset (Fig. 4.15); and calculated the correlation between image similarity values and model performance values (Table 4.1). We did not need to do so for the Gibbon dataset because the positive/negative binary classification made it easy to compare the image similarity scores among and across classes. Because the image similarity scores are calculated by selecting images from the labelsets in the 48-hour baseline dataset, we averaged

the recall and precision scores from the best-performing model for the 48-hour baseline dataset, which was the pretrained MobileNetV2 model.

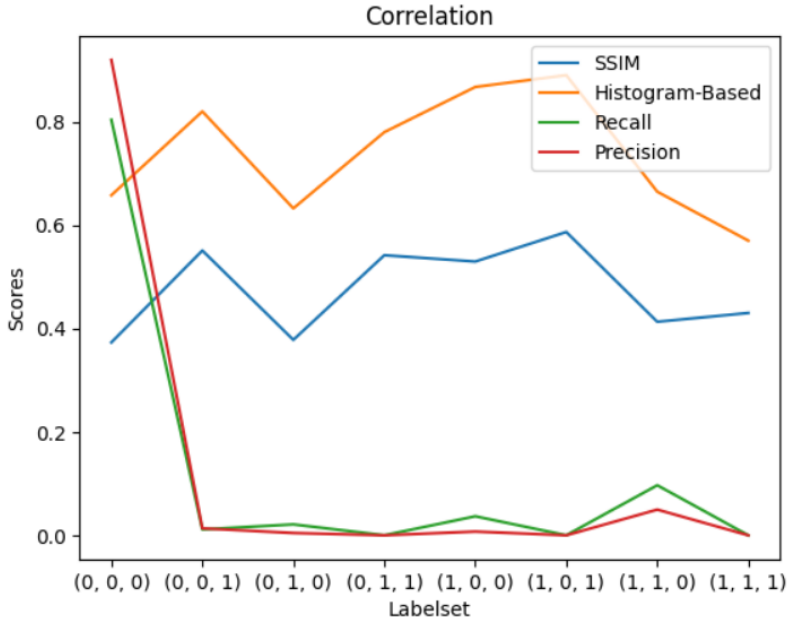


Figure 4.15. Averaged SSIM and histogram-base image similarity scores for each labelset matched with the recall and precision for the corresponding labelset

We calculated the *r* (Pearson correlation coefficient) and the R^2 (coefficient of determination) scores for the averaged SSIM and histogram-based image similarity scores with the averaged recall and precisions shown in Figure 4.9. The two similarity scores computed by the two different methods (SSIM and histogram) were correlated with each other but are not strongly correlated with the performance metrics. We also saw that R^2 scores were low for every comparison except for the comparisons between the SSIM and histogram-based similarity scores and between the performance metrics. We also calculated the scores after removing the None labelset, to test whether the presence of the None labelset affected the scores, but there was no

significant change in the correlation scores. These correlation scores show that image similarity for each labelset was not a large factor in helping the model achieve better performance for each labelset.

Table 4.1. Correlation scores for image similarity scores with model performance scores.

	r with the None labelset	*r* without the None labelset	R^2 with the None labelset	R^2 without the None labelset
SSIM & Histogram	0.89	0.90	0.79	0.816
SSIM & Recall	-0.53	-0.45	0.28	0.20
SSIM & Precision	-0.50	-0.37	0.25	0.14
Histogram & Recall	-0.28	-0.19	0.08	0.04
Histogram & Precision	-0.27	-0.20	0.07	0.04
Recall & Precision	0.99	0.94	0.99	0.89

Chapter 5

Discussion and Conclusion

5.1 Discussion

In this thesis, we attempted to create the first deep learning model for detecting glider self-noise using image classification. We automated the process of dataset creation by syncing the passive acoustic data with glider engineering data. We examined and combined different sampling and augmentation techniques and compared their results. We created multi-label datasets with balanced classes and balanced labelsets and investigated their significance in model performance.

The different sampling and augmentation methods were likely less effective because of the huge imbalance in our datasets among classes and labelsets (as much as 40,081 samples vs 4 samples across labelsets) caused large amounts of duplicates or augmented data in the dataset. Models with equally-balanced labelsets and classes tended to have high validation recall and precision, but high validation loss, a stark contrast to the relatively lower validation loss of the baseline models. High validation loss in the models trained on sub-datasets with high percentage increase by oversampling and data augmentation supported findings in the literature that suggested that oversampling a dataset by too much can potentially lead to overfitting [29, 30]. Contrary to results found in the literature [17, 18, 33] , using data augmentation to increase and diversify the dataset did not have any effect as compared to oversampling by duplication.

There are two likely underlying reasons for the poor model performance: the multi-label classification setup and the small (often single-digit) number of samples in minority labelsets. Our dataset differs from datasets used in similar research are that other datasets are set up as binary classification tasks, while our dataset is set up as a more complicated multi-label classification problem with eight labelsets and three binary classes. An alternate approach to this problem would be to perform multiclass classification with eight different classes, or create eight different binary classification problems (one for each labelset) and combine results. However, the number of classes needed for multiclass classification scales exponentially with the number of classes, which is less ideal for more complex problems. Splitting the problem into several binary classification problems is not ideal for deep learning applications, in which each model takes a large amount of computing resources. A potential solution to address the problem of small minority classes is to create the dataset by selecting the amount of passive acoustic data to use by the resulting number of samples in the minority labelsets instead of creating a dataset by the total amount of resulting samples. However, this solution may lead to potentially requiring a large amount of computing resources to create the dataset, and its level of computing power may not be readily available for integration onboard a glider due to power, space, and processing time considerations.

5.2 Future Work

The long-term goal for the larger project is to develop the capability of identifying glider self-noise in real-time, while keeping the model lightweight. Existing spectrogram based real-time detection and classification applications in other fields, such as the Merlin Bird ID, will serve as a guideline for future development. While this achievement was not possible within the time scale of a thesis, several short-term goals achieved were developing the pipeline for glider

self-noise detection and comparing various combinations of sampling and data augmentation methods.

There are several areas to improve on for future research. While our automated process of annotating data saved human and computing resources, it resulted in a dataset with less precise annotations and misclassified images. We can increase the accuracy and precision of our annotations without taking up as much time as manual labeling by training models or using existing libraries to automatically generate annotations using manually labeled data as a ground truth. This could help the model better identify the patterns inherent to each type of glider self-noise and improve the accuracy of classifications. Furthermore, having more precise annotations could allow us to use the dataset for other tasks such as image segmentation and bounding box detection, which are used in existing audio detection and classification models such as Merlin Bird ID and BirdNet.

Another way we could improve model performance is by preprocessing the data using classical signal processing methods to simplify the task for the model. Noise reduction methods could be applied to the hydrophone data to eliminate glider flow noise and continuous tonal noise from sources like ships and storms to reduce irrelevant noise and to reveal potentially obscured patterns [6]. Classical signal processing techniques could also be applied to filter out noise that have obvious patterns (such as air pump noise), which would allow the model to focus on classifying self-noise with less obvious patterns and reduce computing resources.

Model accuracy could be further improved by addressing the problem of the large imbalance between different classes and labelsets in a different way. Newer, model-based data synthesis methods might generate more realistic synthetic images than the data augmentation methods

used in this thesis, which could increase the diversity of the dataset, reduce overfitting and increase generalizability.

Once completed, this kind of work can have the potential to increase the accuracy and efficiency of detection algorithms in passive acoustic monitoring and reduce costs of false detections of NARW, ultimately leading to better protection for this critically endangered species.

Bibliography

[1] H. M. Pettis and P. K. Hamilton, "North Atlantic Right Whale Consortium 2024 Annual Report Card," 2025.

[2] E. L. Meyer-Gutbrod, K. T. A. Davies, C. L. Johnson, S. Plourde, K. A. Sorochan, R. D. Kenney, C. Ramp, J. F. Gosselin, J. W. Lawson and C. H. Greene, "Redefining North Atlantic right whale habitat-use patterns under climate change," *Limnology and Oceanography,* vol. 68, no. S1, pp. S71-S86, June 2023.

[3] K. L. Indeck, M. F. Baumgartner, L. Lecavalier, F. Whoriskey, D. Durette-Morin, N. R. Pettigrew, J. M. McSweeney, L. H. Thorne, K. L. Gallagher, C. R. Edwards, E. Meyer-Gutbrod and K. T. A. Davies, "GLIDER SURVEILLANCE FOR NEAR-REAL-TIME DETECTION AND SPATIAL MANAGEMENT OF NORTH ATLANTIC RIGHT WHALES," *Oceanography,* vol. 38, no. 1, pp. 13-21, 2025.

[4] D. C. Webb, P. J. Simonetti and C. P. Jones, "SLOCUM: an underwater glider propelled by environmental energy," *IEEE Journal of Oceanic Engineering,* vol. 26, pp. 447-452, 2001.

[5] M. F. Baumgartner, J. Bonnell, P. J. Corkeron, S. M. Van Parijs, C. Hotchkin, B. A. Hodges, J. Bort Thornton, B. L. Mensi and S. M. Bruner, "Slocum Gliders Provide Accurate Near Real-Time Estimates of Baleen Whale Presence From Human-Reviewed

Passive Acoustic Detection Information," *Frontiers in Marine Science,* Vols. Volume 7 - 2020, 2020.

[6] M. F. Baumgartner and S. E. Mussoline, "A generalized baleen whale call detection and classification system," *The Journal of the Acoustical Society of America,* vol. 129, no. 5, pp. 2889-2902, May 2011.

[7] D. Stowell, "Computational bioacoustics with deep learning: a review and roadmap," *PeerJ,* vol. 10, p. e13152, March 2022.

[8] M. Bittle and A. Duncan, "A review of current marine mammal detection and classification algorithms for use in automated passive acoustic monitoring," in *Proceedings of Acoustics*, 2013.

[9] L. Thomas and T. A. Marques, "Passive acoustic monitoring for estimating animal density," *Acoustics Today,* vol. 8, p. 35–44, 2012.

[10] E. Dufourq, I. Durbach, J. P. Hansford, A. Hoepfner, H. Ma, J. V. Bryant, C. S. Stender, W. Li, Z. Liu, Q. Chen, Z. Zhou and S. T. Turvey, "Automated detection of Hainan gibbon calls for passive acoustic monitoring," *Remote Sensing in Ecology and Conservation,* vol. 7, pp. 475-487, 2021.

[11] P. J. Dugan, A. N. Rice, I. R. Urazghildiiev and C. W. Clark, "North Atlantic right whale acoustic signal processing: Part II. improved decision architecture for auto-detection using multi-classifier combination methodology," in *2010 IEEE Long Island Systems, Applications and Technology Conference*, 2010.

[12] J. A. Nystuen and H. D. Selsor, "Weather Classification Using Passive Acoustic Drifters," *Journal of Atmospheric and Oceanic Technology,* vol. 14, pp. 656-666, 1997.

[13] B. B. Ma and J. A. Nystuen, "Passive Acoustic Detection and Measurement of Rainfall at Sea," *Journal of Atmospheric and Oceanic Technology,* vol. 22, pp. 1225-1248, 2005.

[14] P. J. Dugan, A. N. Rice, I. R. Urazghildiiev and C. W. Clark, "North Atlantic Right Whale acoustic signal processing: Part I. comparison of machine learning recognition algorithms," in *2010 IEEE Long Island Systems, Applications and Technology Conference*, 2010.

[15] Y. Shiu, K. J. Palmer, M. A. Roch, E. Fleishman, X. Liu, E. M. Nosal, T. Helble, D. Cholewiak, D. Gillespie and H. Klinck, "Deep neural networks for automated detection of marine mammal species," *Scientific Reports,* vol. 10, no. 1, pp. 1-12, December 2020.

[16] O. S. Kirsebom, F. Frazao, Y. Simard, N. Roy, S. Matwin and S. Giard, "Performance of a deep neural network at detecting North Atlantic right whale upcalls," *The Journal of the Acoustical Society of America,* vol. 147, no. 4, pp. 2636-2646, April 2020.

[17] B. Padovese, F. Frazao, O. S. Kirsebom and S. Matwin, "Data augmentation for the classification of North Atlantic right whales upcalls," *The Journal of the Acoustical Society of America,* vol. 149, no. 4, pp. 2520-2530, April 2021.

[18] L. Nanni, G. Maguolo and M. Paci, "Data augmentation approaches for improving animal audio classification," *Ecological Informatics,* vol. 57, p. 101084, May 2020.

[19] Z. Zhao, S.-h. Zhang, Z.-y. Xu, K. Bellisario, N.-h. Dai, H. Omrani and B. C. Pijanowski, "Automated bird acoustic event detection and robust species classification," *Ecological Informatics,* vol. 39, pp. 99-108, 2017.

[20] Y. R. Pandeya, D. Kim and J. Lee, "Domestic Cat Sound Classification Using Learned Features from Deep Neural Nets," *Applied Sciences,* vol. 8, 2018.

[21] A. N. Tarekegn, M. Giacobini and K. Michalak, "A review of methods for imbalanced multi-label classification," *Pattern Recognition,* vol. 118, p. 107965, October 2021.

[22] M.-L. Zhang and Z.-H. Zhou, "A Review on Multi-Label Learning Algorithms," *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING,* vol. 26, no. 8, p. 1819, 2014.

[23] F. Charte, A. J. Rivera, M. J. del Jesus and F. Herrera, "Addressing imbalance in multilabel classification: Measures and random resampling algorithms," *Neurocomputing,* vol. 163, pp. 3-16, September 2015.

[24] M. A. Tahir, J. Kittler and F. Yan, "Inverse random under sampling for class imbalance problem and its application to multi-label classification," *Pattern Recognition,* vol. 45, no. 10, pp. 3738-3750, October 2012.

[25] E. Cakir, T. Heittola, H. Huttunen and T. Virtanen, "Multi-label vs. combined single-label sound event detection with deep neural networks," in *2015 23rd European Signal Processing Conference (EUSIPCO)*, 2015.

[26] E. Cakir, T. Heittola, H. Huttunen and T. Virtanen, "Polyphonic sound event detection using multi label deep neural networks," in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015.

[27] B. McFee, C. Raffel, D. Liang, D. P. W. Ellis, M. McVicar, E. Battenberg and O. Nieto, "librosa: Audio and music signal analysis in python.," *SciPy,* vol. 2015, p. 18–24, 2015.

[28] K. Ghosh, C. Bellinger, R. Corizzo, P. Branco, B. Krawczyk and N. Japkowicz, "The class imbalance problem in deep learning," *Machine Learning,* vol. 113, p. 4845–4901, 2024.

[29] R. Mohammed, J. Rawashdeh and M. Abdullah, "Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results," *2020 11th International Conference on Information and Communication Systems, ICICS 2020,* pp. 243-248, April 2020.

[30] N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research,* vol. 16, pp. 321-357, June 2002.

[31] A. Buslaev, V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin and A. A. Kalinin, "Albumentations: Fast and Flexible Image Augmentations," *Information 2020, Vol. 11, Page 125,* vol. 11, no. 2, p. 125, February 2020.

[32] D. S. Park, W. Chan, Y. Zhang, C. C. Chiu, B. Zoph, E. D. Cubuk and Q. V. Le, "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition,"

Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH, Vols. 2019-September, pp. 2613-2617, 2019.

[33] Z. Mushtaq, S.-F. Su and Q.-V. Tran, "Spectral images based environmental sound classification using CNN with meaningful data augmentation," *Applied Acoustics,* vol. 172, p. 107581, 2021.

[34] I. Jordal, A. Tamazian, T. Dhyani, askskro, E. T. Chourdakis, N. Karpov, C. Landschoot, C. Angonin, O. Sarioglu, W. Drevo, BakerBunker, kvilouras, F. Catania, E. B. Çoban, E. Gritskevich, F. Mirus, J.-Y. Lee, K. Choi, L. Killingberg, MarvinLvn, SolomidHero, T. Alumäe, Z. R. Han and R. Solovyev, *iver56/audiomentations: v0.42.0,* Zenodo, 2025.

[35] F. Chollet, *Image preprocessing–Keras documentation. GitHub,* 2015.

[36] M. Tan and Q. V. Le, *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,* 2020.

[37] M. Tan and Q. V. Le, *EfficientNetV2: Smaller Models and Faster Training,* 2021.

[38] K. He, X. Zhang, S. Ren and J. Sun, *Deep Residual Learning for Image Recognition,* 2015.

[39] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto and H. Adam, *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,* 2017.

[40] K. Simonyan and A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition,* 2015.

[41] G. Huang, Z. Liu, L. van der Maaten and K. Q. Weinberger, *Densely Connected Convolutional Networks,* 2018.

[42] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna, *Rethinking the Inception Architecture for Computer Vision,* 2015.

[43] F. Chollet, *Xception: Deep Learning with Depthwise Separable Convolutions,* 2017.

[44] B. Zoph, V. Vasudevan, J. Shlens and Q. V. Le, *Learning Transferable Architectures for Scalable Image Recognition,* 2018.