CURTIN UNIVERSITY

DISSERTATION

Applying Deep Neural Networks to Classify Humpback Whale Song Units

Author:
Joannes Karmel
GANDAHUSADA

Supervisor:
Christine ERBE, Qilin LI

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Data Science (Honours)

in

Curtin University

January 11, 2024

Contents

Acknowledgements

I would like to acknowledge Christine Erbe from CMST on providing me the data and explaning the technicalities of spectrograms. I was able to develop a basic understanding of audio data as a waveform and spectrogram, which was much needed for this dissertation. I would like to also thank Qilin Li for proving me guidance in training the Neural Networks, and assisting with the direction of the code for this project. Thank you for being in this one year journey with me, I appreciated the great help from you guys looking out for me.

CURTIN UNIVERSITY

Abstract

Marine Science and Technology
Department or School Name

Bachelor of Data Science (Honours)

Applying Deep Neural Networks to Classify Humpback Whale Song Units

by Joannes Karmel GANDAHUSADA

Scientists have been using PAM technology to study and monitor Humpback Whale acoustics for many years. While this method does reduce man hours on the field, scientists still have to spend strenuous weeks annotating Humpback Whale sounds, which not only takes time, but is prone to error. This paper details a workflow using a program scripted in Python to conduct a study on training neural networks on classifying different Humpback Whale vocalisations, which scientists without a computing background can replicate themselves. The study found that models of the Vision Transformers architecture were very robust and performed well when classifying individual spectrogram images, with each performance accuracy being greater than 84%. Augmenting the data using on-the-fly random horizontal translations would further aid the performance of each model by preventing them from overfitting and achieving a validation accuracy representative to the model's testing accuracy. These models can also be deployed on classifying streamed audio, although more work needs to be done to improve these models in a live setting. Without classifying too many non-existent sounds, a ViT model can accurately classify 54.9% of all sounds, with an incorrect rate of 10.1% and false negative rate of 32.8%. The codebase for this project can be found at *https://github.com/kasmello/Humpback-Neural-Network*.

Chapter 1

Introduction

The need to actively monitor Australian marine life is essential to maintain the health and diversity of its ecosystem. Marine scientists do this in order to make frequent and scientific observations of animals, such as their current population at a certain location, their migration patterns, and their communication amongst themselves and other species. In order for Australia to become a country with thriving aquatic wildlife, active steps need to be taken to frequently monitor marine activity and respond quickly to any environmental threats. Humpback Whales (*Megaptera novaeangliae*) are considered essential to Australia's marine life, and are critical to monitor in order to maintain the ocean's ecosystem because they are able to recycle nutrients by releasing fecal matter to the surface of the ocean after feeding[1]. This helps fertilise the oceanic plankton which are responsible for capturing carbon and producing more than half the oxygen on Earth. It can be said that these species play a vital part in maintaining a sustainable ocean and planet. While the Humpback whale used to be under heavy threat to whaling, a massive effort has been made to preserve the whales in Australia. They are still considered to be endangered in Tasmania, but their population in other regions has recovered, listed Vulnerable in New South Whales and South Australia, and Conservation Dependent in Western Australia[2]. One way to continue Australia's efforts in the recovery of Humpback Whales is to conduct thorough studies on their vocal behaviour, whether it be for mating rituals, parenting, or general communication with whales or other species. The analysis of such behaviour to a widespread sample of Humpback whales may have many benefits, including but not limited to a better understanding of Humpback Whale communication and action, or learning to spot for distress signals. A popular approach to studying Humpback Whale vocalisations is the use of Passive Acoustic Monitoring (PAM).

PAM is a monitoring technique which focuses on using recording devices to record and monitor the acoustics of underwater wildlife without interfering with them(Zimmer, 2011), which scientists use to monitor the soundscape of marine life in order to build a better understanding (Erbe, 2013). PAM is very effective, as many aquatic animals use sound as their main sense to communicate, navigate the waters, or locate prey. This is a product of acoustic signals being able to travel quickly and far in the water. In the case of Humpback Whales, the males are known to sing, by repeating patterns of different calls, typically in the range of 20Hz to 8kHz (Payne and McVay, 1971). They make these vocalisations for a myriad of reasons, such as foraging, distress signaling, communication, and as a mating signal (Nicklin, Darling,

[1] How Whales Help the Ocean - https://www.greatwhaleconservancy.org/how-whales-help-the-ocean - accessed 21 Nov, 2022

[2] *Megaptera novaeangliae* - Humpback Whale - https://www.environment.gov.au/cgi-bin/sprat/public/publicspecies.pl?taxon_id=38 - accessed 21 Nov, 2022

and Jones, 2006). With PAM technology, scientists are able to gain insight to the behaviours and movement of Humpback Whales through the acoustic signals they produce. However, this recording technology still has notable caveats. Autonomous records are collecting data at thousands of samples per second, which quickly adds up to many terabytes every year (McCauley et al., 2017). As such, scientists will have to manually comb through weeks, months, and years of this data in order to collect a reasonable amount of data to analyse. This process still requires many hours, and is subject to human errors such as the misclassification of animals, or even ambience as a form of life. Considering the terabytes of data that can accumulate, it is natural to use some sort of data-driven system capable of automating the detection and annotation process for species such as Humpback Whales.

Many data-driven approaches tackling similar problems are in the form of machine learning. A paper in 2007 was driven by the necessity of non-invasive, automated surveying of species in a water and river ecosystem, and as such used support vector machines (SVM) to classify seven different Humpback Whales using the song units they produced. The study followed and recorded song units from these different humpback whales, recording up to 28,000 song units for training, and was able to classify the correct whale up to 99% of the time (Mazhar, Ura, and Bahl, 2007). Another paper trained a supervised Hidden Markov Model (HMM) to classify Humpback Whale calls (Tolkova et al., 2017). This was aided by the processing of spectrograms of whale calls using principal component analysis (PCA) and connected-component based methods to remove the background noise. This paper was able to achieve an accuracy of 68% when being tested on Humpback Whale 'squeaks', 'low yaps' and empty sounds. While both of these methods are valid and make very powerful tools, the technical knowledge one would require to try these methods may be too much for a marine scientist who has very little computing knowledge. An ideal system should enable a sciencist to construct their own system using the data they have, and tweak their system to suit their own needs. For an ideal balance of high performance and ease of set-up and use, a preferable machine learning method lies in the convolutional neural network (CNN).

CNNs are primarily used in vision classification tasks and have achieved State of The Art (SOTA) performances across different problems. This has been helped by the model pre-trained on datasets containing millions of images being available to download and use for anyone. To utilise the prior training of the model audio classification tasks, With a problem similar to the one presented in this paper, one would convert their audio data into spectrogram images, and utilised 'transfer learning' by downloading the pretrained models and fine-tuning them on the spectrograms for audio classification tasks. There are many cases of researchers using vision classification neural networks over audio signal transformers due their large performance gains and development in recent history, such as a recent study published in October 2022 which used the EfficientNet-B0 architecture (White et al., 2022). The scientists in this paper were able to build a model to distinguish between image spectrograms of Delphinoid vocalisations, biological clicks, vessel noises and ambient noise, achieving an 80% accuracy in noisy data, and 95% in low-noise data. This study chose a small CNN architecture as the intended solution was for a light-weight solution able to be used and deployment on a local machine, a motive which has strongly inspired the direction of this paper. The scientists annotated 25,075 audio clips, each being 3s long, and assigned 13,198 spectrograms to the training, 7918 on the validation set and 3959 on the testing set. Each audio clip was reviewed manually to ensure the sounds

were never time-centered (assumed to mitigate overfitting), and then converted to spectrograms, which the model would then be trained on.

The newest architecture of Neural Networks is the transformer architecture. Another study documents the testing and creation of an Audio Spectrogram Transformer (AST), an architecture derived from the attention-based Vision Transformer (ViT) architecture (Gong, Chung, and Glass, 2021). As transformers typically need large amounts of data to train, this paper applied transfer-learning techniques by adapting several Vision Transformer models pretrained on ImageNet to AST, and finetuned further using AudioSet, an audio dataset with over two million 10 second clips from Youtube. The outcome of this paper had many SOTA performing models, including an AST models pretrained on just ImageNet achieving 95.6% on the, ESC-50 dataset and an AST model pretrained on both ImageNet and Audio Set achieving 98.1% on the Speech Commands V2 set. Due to the promise of high accuracy in audio classification, the ViT architecture will be one of the neural network models featured in this paper.

The expectation of this study's outcome was a procedure or system which can assist with the monitoring of Australia's aquatic wildlife. To achieve this, the goal of this study was to construct a workflow and program for marine scientists to use to build their own neural networks for Humpback Whale sound classification. These models not only should classify individual Humpback Whale sounds neatly formatted into spectrogram images, but also perform well with streamed, live data. One of the main challenges of this goal was to design the workflow to be as efficient and easy to understand that other scientists without immense computing knowledge can follow using their own data. Studies show the vocalisations of humpback whales differ between regions across the world, and are prone to change either through whales picking up songs from other whales in different regions, or simply due to 'cultural evolution' over time (Allen et al., 2018). As such, it is very important for scientists to know how to retrain their models with ease to recognise different groups of Humpback Whales. Additionally, the inference time, which refers to the amount of time taken to classify unseen data, needs to be quick and achievable without supercomputers. While newer models like ViT and EfficientNet may converge better than other models, their larger architectures logically lead to longer inference times of the validation-set. Should a model's speed in predicting streamed data be impacted and cause lag, it will make the model useless for classifying live data. The final issue was whether a model trained on singular images was able to be utilised on live data in the first place, as classifying live data has significantly more hurdles than singular images. They include recognising the absence/presence of sound and dealing with dirty data, in the form of high noise to signal segments, and multiple whale vocalisations being played over one another. These problems were addressed in this paper through meticulous training procedures and practices which explored optimal parameter values to mitigate these issues.

Chapter 2

Methods

2.1 Models Used

ViT, ResNet-18, and EfficientNet-B0 were the neural network architectures chosen for this paper. These were chosen due to their differing size and layers. These differences will be analysed in this paper on whether they have any benefits or deficits in classifying Humpback Whale vocalisations.

2.1.1 Vision Transformers Architecture

The ViT architecture is a neural network architecture containing no convolutional layers and instead uses attention layers (Vaswani et al., 2017). The advantages of using attention layers in the context of images is the model being able to draw connections between two features far apart from each other. A regular CNN draws better connections between nearby pixels, whereas a ViT will calculate the relationship between each pair of image patches to derive the attention of each patch, which is how important a patch is to the image. Figure 1 details the layer composition of the architecture.

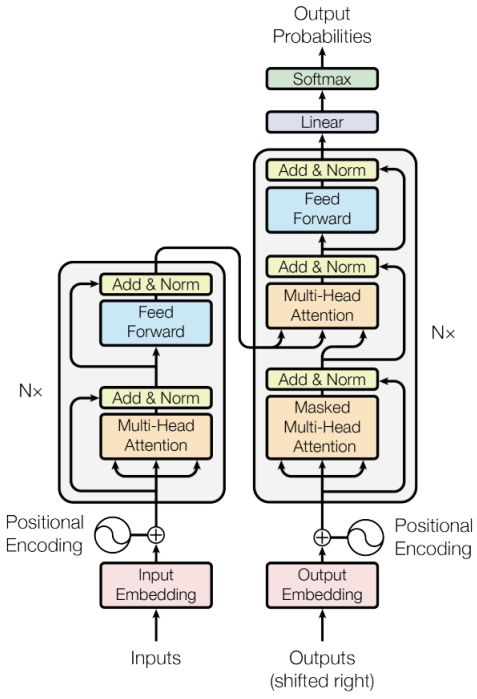


Figure 2.1: ViT Architecture (Vaswani et al., 2017)

The images were first divided into same-size patches before being flattened to a sequence and being assigned positional encoding in the input embedding layer. The sequence of patches were then fed through multiple self-attention modules before being transformed into a probability distribution by the softmax output layer. A vital layer of the self-attention modules is the multi-head attention layers, composed of parallel attention mechanism modules that calculate the query, key, and values of all possible pairs of inputs, which were used to derive the attention scores of each patch using the weighted sum of the values.

$$c_i = \sum_j \alpha_{ij} h_j$$

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_k exp(e_{ik})}$$

$$e_{ij} = a(s_i, h_j)$$

Figure 2.2: Equations to derive dot-product attention. c refers to the attention, s refers to the query, which maps a key that is used to retrieve the value (h). (Bahdanau, Cho, and Bengio, 2014)

It is with these mechanisms that vision transformers often generalizes faster than other neural networks. However, computing all the possible pair-wise calculations is why the ViT is computationally more intensive than other neural network architectures. Theoretically, one could limit the pair-wise calculation of each input to a neighbourhood of size r (Vaswani et al., 2017), but this paper will not investigate this approach. The ViT this paper will investigate is the base model with 16 x 16 patches as a starting point.

2.1.2 ResNet-18 Architecture

ResNet-18 Neural Network models are widely used for their efficiency and great generalisation. The architecture contains 18 deep layers and modelled to have many convolutional layers functioning effectively. It is common when training deep networks that information is forgotten in the deeper layers, leading to exploding or vanishing gradients which would degrade the performance of the model (Zagoruyko and Komodakis, 2016). To work around this the ResNet architecture makes use of residual blocks, which have shortcut connections mapping inputs or hidden states forward in the network, forcing subsequent layers to retain information learnt from the previous layers (Targ, Almeida, and Lyman, 2016).

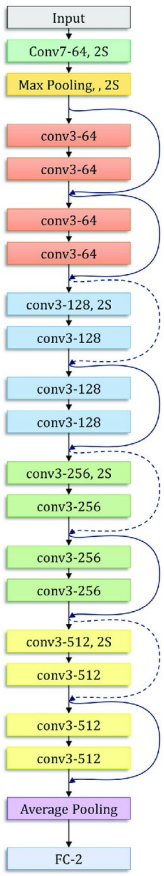


Figure 2.3: ResNet-18 Architecture (Kundu et al., 2021a)

In the original residual learning paper (He et al., 2016), instead of making stacked layers fit an identity mapping, denoted as $H(x)$, they were instead were set to fit a residual mapping, denoted as $F(x) = H(x) - x$. This was found to be a simpler procedure for the layers, which can be explained in Figure 2.4.

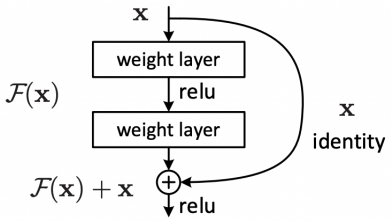


Figure 2.4: Example of a residual block (He et al., 2016)

Figure 4 is an example of a residual block. While the layers are fitting the residual mapping, the shortcut connection simply performs an identity mapping of the inputs and adds them to the outputs one or more layers ahead. With this procedure, the stacked layers have an easier task to optimise and fit the residual mapping to 0 to achieve $H(x) = x$.

2.1.3 EfficientNet Architecture

The EfficientNet family of models is a product of the optimized scaling of neural networks in network depth, width and resolution (Tan and Le, 2019). While there

have previously been neural networks which have been scaled up, such as a ResNet-18 model scaled up to ResNet200 with more deep layers, there was never a proper procedure to scale a network before this paper. This paper has not only procured a method of compound scaling, a method to scale all three dimensions of a network, but also a new network architecture as a base to scale up. This architecture would go to achieve many SOTA performances across different classification problems, while being simultaneously one of the most smallest and efficient neural networks (8.4x smaller and 6.1x faster). Its family ranges from EfficientNet-B0, the lightest model, and finishes at EfficientNet-B7, the largest and most complex model. Each subsequent network is scaled up using compound scaling, which scales the number of layers, channels and input size with a fixed coefficient.

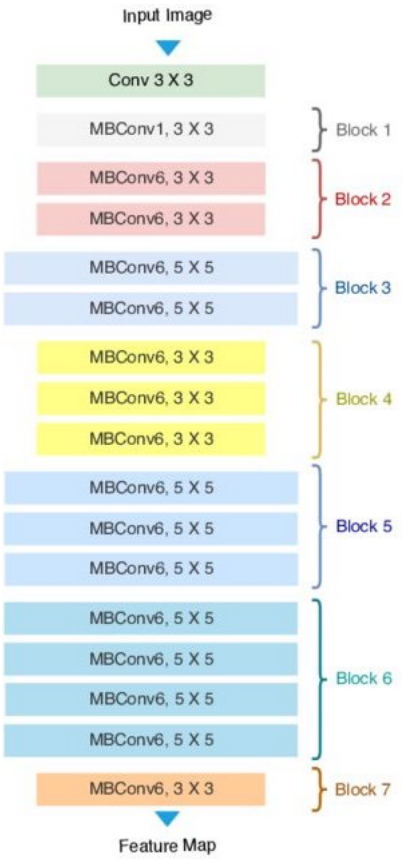


Figure 2.5: EfficientNet-B0 Architecture (Kundu et al., 2021b)

The smallest architecture in the family, EfficientNet-B0 was designed to be as efficient as possible and optimised to have great accuracy and floating point operations per second (FLOPS). It was due to a primary goal of this study being workflow efficiency, that this model was chosen to be tested and compared to ViT or ResNet-18.

	ViT-Base/16	**ResNet-18**	**EfficientNet-B0**
FLOPS (Billions)	33.03	1.82	0.02

Figure 2.6: Comparison of FLOPS across ViT, ResNet-18, EfficientNet-B0 when trained on ImageNet.

Taking a look at the FLOPS per architecture[1], ViT has the largest at 33 billion FLOPS, followed by ResNet-18 and finally Efficient, with 20 million FLOPS. The expectation for this paper is that while ViT will have the longest inference time, its performance will be marginally better compared to the other models due to the substantial number of FLOPS.

2.2 Workflow

The methods in this investigation can be divided into three categories: data preparation, training, and testing. Data preparation contains data collection, and annotating and processing methods in preparation for the training of the model. Training consists of training many models on many different combinations of parameter values, and evaluating the performance of trained models using their performances on the validation and testing data sets. The last section, testing, sees the most optimal models being applied on streamed data. The flowchart in Figure 2.6 details the workflow procedure carried out in this paper.

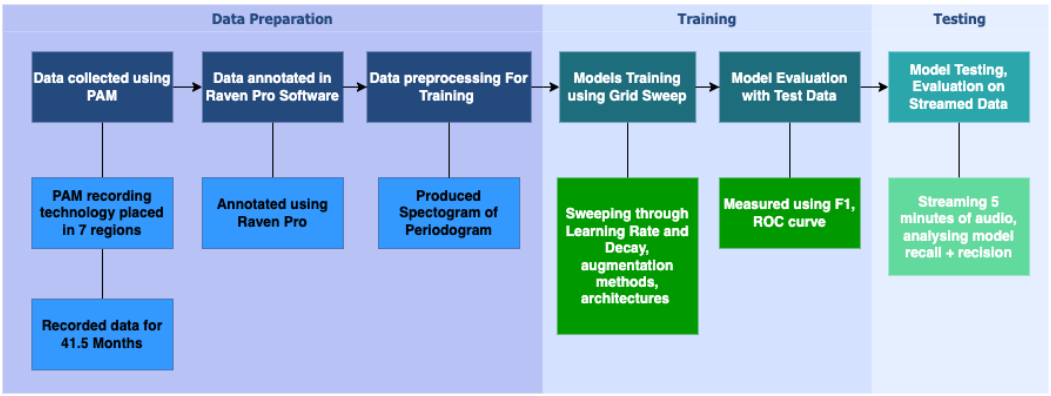


Figure 2.7: Workflow diagram, detailing each step and the contents of each step.

2.2.1 Data Collection

With PAM technology, sound sensors were placed underwater across seven different sites, and collected audio recordings for a 42-month period at either 10,000Hz or 6,000 Hz. These sensors would record 5 minutes of audio signals over 15 minutes, as this way allows the recorders to capture audio spread over a wider range of time while consuming three times less storage. Each audio was initially stored as a DAT audio file, and converted to five minute Waveform Audio (*WAV*) files using CHORUS, a MATLAB toolbox developed for underwater sound analysis by Curtin's Centre for Marine Science and Technology (CMST).

[1]MODEL ZOO - https://mmclassification.readthedocs.io/en/latest/model_zoo.html - accessed 3 Dec, 2022

2.2.2 Data Annotation

A total of 23,026 samples of audio data were annotated. This was done inside the *Raven Pro* program for Sound Analysis, developed by *Cornell Lab of Ornithology*. The program provided the capabilities of reading *WAV* files, converting waveforms to spectrograms for the ease of sound identification, and annotating sounds of interest in the file by selecting the desired song units using a selection box. Sounds were carefully annotated, with the goal of providing the best training data, and followed the principles below:

1. Each selection was to contain only one vocalisation. There could be no overlapping acoustics unless data for a class was hard to find
2. While there may be background noise in each sound selection, it should not obscure the visibility of the vocalisation in the spectrogram
3. Vocalisations were categorized by their approximate pitch, length, shape and harmonics
4. All other sounds that were neither Humpback Whales or Minke Whales were marked as 'Blank'. This includes ambient noise and ship vessel noise

After careful consideration, 25 different categories of audio were procured, with one category of ambience, one category for Minke Whale vocalisations, and the other 23 containing different Humpback Whale acoustics. The reason for including Minke Whale acoustics was to prevent the Neural Network from overfitting to only Humpback Whale acoustics, which may result in the Neural Network classifying any sort of vocalisation as a Humpback Whale call. By including Minke Whale vocalisations, which are similar in sound to Humpback Whales, the networks were trained to differentiate between Minke Whale acoustics and Humpback Whale acoustics. In addition to this, 'Blank' sounds were as such as an easy way to group together vessel sounds, ambience, and any other non-whale vocalisations, and train the neural network models to classify them as such.

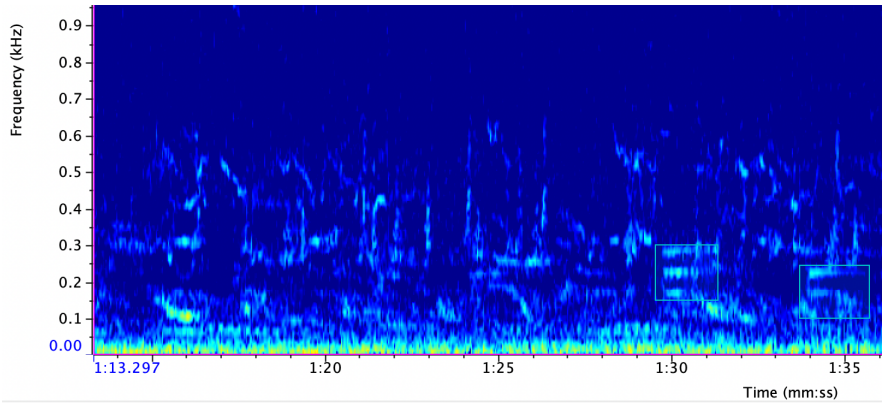


Figure 2.8: Spectrogram of Minke Whale (marked in a box) and Humpback Whale vocalisations. To the untrained eye, this can be difficult to differentiate.

2.2.3 Data Pre-processing

As the model can only perform as well as the data it is given, necessary steps were taken to ensure that all the spectrograms distinctively showed the sounds. A Python

script was able to read in the 5 minute *WAV* files and crop the appropriate sounds using the start and end times of the annotated tables generated in Raven Pro. Due to the limitations of the neural network accepting fixed-size images, sounds either had to be cropped or lengthened to reach 2.7s. The number 2.7 was chosen, as it was slightly longer than the median length of all collected vocalisations, meaning it would be able to contain the majority of sounds. Should a sound be x seconds too short, the start and end time of the sound would be moved outwards by $\frac{x}{2}$ seconds, and $\frac{x}{2}$ seconds inwards if the sound was x seconds too long.

Each 2.7s wave signal was split into windows of 512 samples, with each window having an overlap of 256 samples. A fast fourier transform (FFT) was then applied to the waveform, which is a function used to break the waveform down to the sum of all frequency sinusoids to shift the *WAV* signal from a time domain to a frequency domain plotted against the amplitude (Brigham, 1988). This is followed by a periodogram method to produce a power spectral density, highlighted the strong signals over background noise, and then stitching the resulting windows together to create spectrograms. The resulting values were then normalised between a range of 0 and 1 for each image to identical ranges, converted into monochrome images, and resized to a dimension of 224 x 224 to match the input dimensions of the neural networks. It is also important to note that frequencies below 50Hz and above 3000Hz have been removed from the spectrogram, as sounds belonging in these ranges tended to be anomalies.

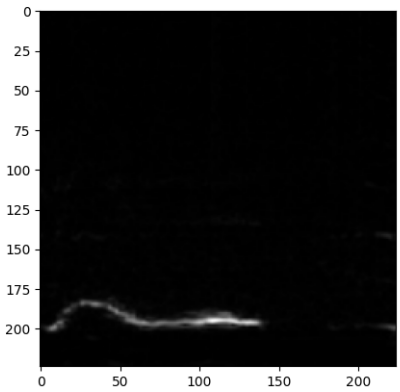


Figure 2.9: Example of Spectrogram Image calculated from a power spectral density.

The same process is repeated for the validation and testing data, with the exception of data augmentation, detailed in section 2.2.5.

2.2.4 Training Methods

This section highlights the methods and parameter values used to train each model in order to find a set of high performing models with low inference times. Each model was trained on an single-GPU system with an RTX3080.

Each model was trained using a different combination of parameter values. A grid search was used to sweep through a grid of values which controlled how the model learnt, and how the data was augmented. As of now, there is no manual way

to calculate based off the parameter values whether a neural network will perform well, and as such, a grid of parameter values were swept through to train the neural networks and allow for the identification of optimal combination of parameter values. Each batch of training will have its own grid of values to be swept through. Below were the list of parameters in the grid search:

- Learning Rate - the rate which the weights of the models changed.
- Learning Rate Scheduler - the schedule for how the learning rate changes. Can be set to cosine annealing, cosine annealing with warm restarts, or no scheduling.
- SpecAugment Activation - Set to on or off.
- Pink Noise Overlay Activation - Set to on or off.
- Random Horizontal Translation Set to on or off.

Each neural network model was trained on the spectrogram images formed in section 2.2.3. These pictures were propagated forward through each models' layers, with the final layer producing a probability distribution of K probabilities, each probability corresponding to the K^{th} class. These prediction probabilities were produced using a *LogSoftmax* function. The category with the highest probability score will be considered as the model's prediction. The loss function was calculated from the probability distribution, indicating how correct and 'confident' a model was, determined by how close the probabilities were to 0 or 1. The gradient of the loss function was then propagated back to the inital layers of the model to fine-tune the weights of each layer.

$$LogSoftmax(x) = log(\frac{e^x}{\sum_{j=1}^{n} e^x})$$

Figure 2.10: Formula for LogSoftmax. x is the tensor output produced by the model(Gibbs, 1902).

The spectrograms were split in a 80:10:10 ratio for the training, validation and testing sets respectively. Each model was trained with a randomly augmented version of the base training set, and judged based on their validation and testing accuracies. It was ensured that data leakage did not occur during model validation by disabling back propagation of the loss gradient when being validated. Drop-out layers were disabled during validation and testing to achieve maximum performance out of each model, but enabled during training to prevent overfitting.

Other features implemented to combat overfitting included a time limit, and a patience mechanism. Each training run would stop training should a run take longer than an hour, or run out of 'patience'. Inspired by previous recommendations for network training (Bengio, 2012), the patience mechanism in this experiment was used to stop the network from training too long by to monitoring the validation performance of model, and stopping the training when performance gains started to diminish. The patience mechanism in this experiment was set to monitor the F score, which takes into account precision and recall. Should the F-score fail to increase by 0.3% from the previous maximum, the patience level would decrease by one. Eventually the patience level would drop to 0 and stop training. Another feature which implemented was learning rate schedulers, which would lower the learning

rate on subsequent epochs. The two schedulers used in the batches were cosine annealing (cosineAN), and cosine annealing with warm restarts (cosineANW), concepts first introduced in a 2017 study exploring scheduling with restarts Loshchilov and Hutter, 2016. CosineAN is a scheduling method which decays the learning rate every epoch using a cosine function. CosineANW is an almost identical scheduler with the addition of learning rate restarts every few epochs, meaning the learning rate would be restored partially or fully after every few epochs. Some papers used these methods to great effect, such as Dosovitskiy's 2020 paper that trained and finetuned their vision transformers with cosineAN (Dosovitskiy et al., 2020). This paper utilised cosineAN, which would decay until the learnig rate approached 0, and cosineANW, which would decay abruptly, and restart every three epochs. Not only did these features mitigate overfitting, they also saved a lot of necessary time for experiments. With these features, one could train a powerful model in under an hour.

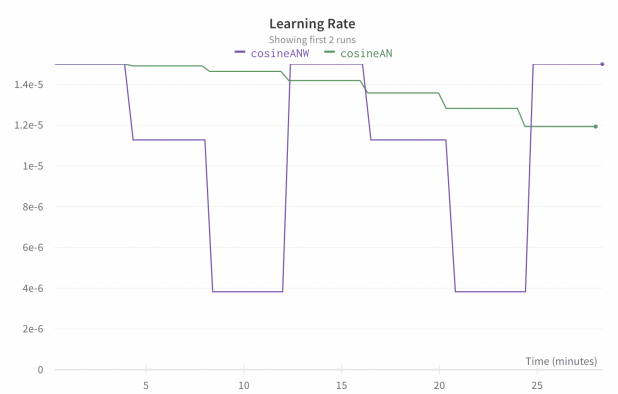


Figure 2.11: Example of the different cosine annealing decay methods

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})(1 + cos(\frac{T_{cur}}{T_{max}}\pi))$$

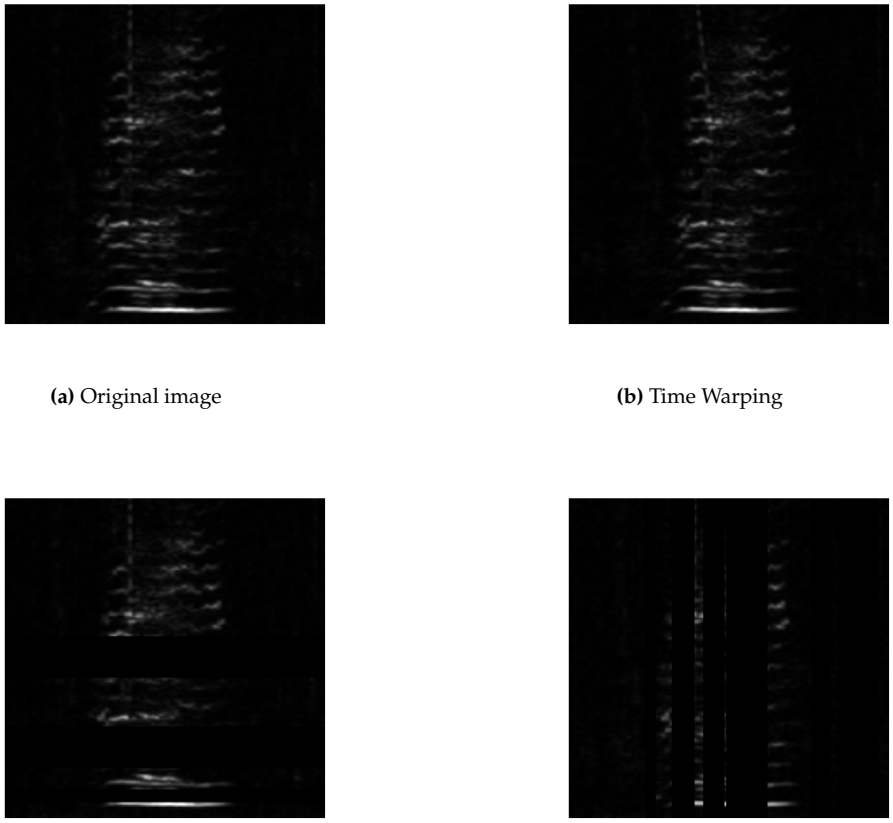
Figure 2.12: Equation of cosineANW (Loshchilov and Hutter, 2016). η refers to learning rate, T_{cur} refers to the number of epochs since the last restart, and T_{max} refers to maximum number of epochs between a restart. For cosineAN, t_{max} was set to the number of maximum epochs (20).

Each model was trained up to 20 epochs using the AdamW optimiser with a weight decay of 0.0001. The values for learning rate, decay, augmentation methods, and architecture were changed depending on the results after each batch.

2.2.5 Data Augmentation

On-the-fly random augmentation practices were employed during training to create synthetic data, which prevented the models overfitting on one set of training data. The expected effect of each augmentation was the slight decrease in validation accuracy, which will either result in a better representation or even an improvement in testing accuracy. The practices used were based on the *SpecAugment* process described in Park's *SpecAugment* proposal designed to enhance neural networks trained for speech recognition (Park et al., 2019). This process involved the horizontal distortion of the spectrogram on the time scale (time-warping), placing horizontal masks on the spectrogram at random frequency bands (frequency masking) and placing vertical masks on the spectrogram at random points of time (time masking). Each of

these transformations had a 35% chance of being carried out at random magnitudes. Another augmentation implemented was the layering of pink noise on top of the spectrograms. This was done by distributing random noise across the frequencies with the equation. $P(f) = \frac{c}{f}$, with f being the frequency (Ward and Greenwood, 2007). Unlike white noise which distributes static noise equally across all frequencies, pink noise is louder in decibels in the lower frequencies, which can be seen by the inverse relationship between the amplitude and frequency. Pink noise was used over white noise due to an observed larger density of random noise under 100Hz in obtaines recordings, simulating dirty data. When enabled, there was a 70% probability of pink noise being overlayed over the spectrogram.



(a) Original image

(b) Time Warping

(c) Frequency masking

(d) Time masking

Figure 2.13: All SpecAugment processes

(a) Original image **(b)** Pink Noise Overlay

Figure 2.14: Pink Noise Augmentation

One last method of data augmentation used was a random horizontal translation applied to the spectrogram. This was introduced in an attempt to improve the model's performance in the detection of Humpback Whale vocalisations on streamed data, as the vocalisations may appear anywhere in the frame, as opposed to the centre of the image, which was framed this way during annotation as it helped capture longer vocalisations. This augmentation has a 50% probability of happening.

(a) Original image **(b)** Horizontal Translation

Figure 2.15: Horizontal Translation Augmentation

2.2.6 Live Detection

After training and testing, the most optimal models produced from training were then tested on streamed audio. The neural networks were tasked with selecting the instances where sound was located in the audio clip, classify the correct vocalisation, and avoid marking ambience as Humpback Whale vocalisations.

Before any prediction was done, the entire sound data will be fed into a voice activity detection function (VAD) and marked in places where sound was detected. This process has been implemented to work with the neural networks to minimize false positives. This was done by splitting the audio into 25ms windows with 10ms overlap, and calculating the log energy of these frames. The algorithm for this system derived from a voice activity detection system for smart earphones (Lezzoum, Gagnon, and Voix, 2014), which also used 25ms windows with a 20ms overlap to

calculate energy. This paper calculates energy from three different frequency bands to generate ratios for thresholds, whereas this paper calculated the energy for only 50Hz to 3000Hz for a simpler approach. The code used for this process was derived from an open-source VAD repository on GitHub (Shokouhi, 2020).

Afterwards, windows of 2.7s were swept through the audio segments, simulating the setting of streaming audio. Each 2.7s window was converted to a spectrogram and fed to the model if the window contained enough sound, which was determined by the VAD. As the output of the model was a probability distribution, users will be able to select another probability threshold indicating the lowest probability a category can have for the program to indicate its presence in the audio file. Should two categories have a probability higher than the threshold, they will both be marked by the program.

This test will be ran with two 5-minute audio segments of data, with a total of 326 Humpback Whale vocalisations naturally scattered throughout the recording. All marked sounds by the program and model will then be written to a selection table file which may be opened by Raven Pro to visualise the selection boxes around vocalisations.

2.2.7 Model Evaluation

To decide on the most optimal models, each model generated from the grid search was compared using the following metrics:

1. Validation Loss Trends - Loss of a model is defined how 'confident' a model is, which can be measured by how close to 1 or 0 its output predictions were. A model with a lower validation loss is a more optimal model. This metric was used to gauge overfitting, which can be seen if the trend of validation losses becomes divergent or starts increasing instead of decreasing during training.

2. Validation Accuracy - Models were optimised to have the highest validation accuracy as possible. A validation accuracy above 80% was an ideal starting point for this study, but a validation accuracy above 90% suggests some overfitting.

3. Testing Accuracy - While a high testing accuracy does mean an optimal model, the main reason these metrics were recorded was to observe if the most optimal models from training can reproduce similar results on unseen data only shown to the model at the end of training, or if the models were simply overfitting on the validation and training data.

4. Inference Time - While bigger models have the capacity of outperforming the smaller models, it was in this paper's best interest to find a model that has a great prediction performance. If a model had a high inference time, it would render the model useless on live data, which needs to classify many frames of the audio per second. An ideal inference time was 10 images per second, which would mean an attempt to classify a new sound every 100ms on the the streamed audio. For the 2331 sounds in the validation set, this would mean a maximum of 230 seconds.

5. Performance on streamed data - As the models were designed to be deployed onto streamed data, their performance on unseen streamed data was taken into consideration. Similarly to the model's testing performance, this metric was looked at to analyse whether the time and resources spent training each model will be reflected on application. There were five separate metrics which will be used to gauge the model's performance here.

   (a) False positives - Sounds predicted by the model where there were none. An optimal model for streamed data was expected to have a rate close to 0%. Calculated by proportion of false positives out of all sounds predicted.

   (b) Sounds classified correctly. An optimal model was expected to have a rate close to 80%. Calculated by proportion of sounds correctly classified out of sounds in the audio.

   (c) Sounds misclassified as other sounds. An optimal model was expected to have a rate close to 0%. Calculated by proportion of sounds incorrectly classified out of sounds in the audio.

   (d) Sounds misclassified as non-sounds. An optimal model was expected to have a rate close to 0%. Calculated by proportion of sounds incorrectly classified as blanks out of sounds in the audio.

   (e) Ratio of items predicted against items in the table. An optimal model would have this model as close to 1 as possible.

   Similarly to the model training procedure, another grid search will be conducted to obtain the variety of results when using different parameter values which decide when a selection of the audio should be classified as a sound or not. These parameters included the maximum number of items predicted in one window, the decimal threshold used with the proportion distributions from the *LogSoftmax* function, and the VAD threshold. The values for these parameters will be manually chosen by a scientist for their unique use case, and this grid search has determined which parameters would be useful for specific use cases.

Chapter 3

Results

3.1 Validation Performance

Many of the models trained were highly accurate in classification, even with the limitations of training time, and patience. While there was not a single 'best' combination of parameters, there was a range of parameter values and augmentation practices that lead to a more robust model that achieves high accuracy on the validation and testing set. Three batches of testing were ran, and each batch was trained on a different combination of parameter values and architectures, which were decided based on results in the previous batch. This would ensure that not too many tests were done in a single batch before finding out that no parameters or values in the batch corresponded to a robust model. Each batch followed 1 hour of training time and a patience level of 3 using the mechanism defined in section 2.2.4.

12 models were produced in the first batch, and treated as a test for whether the combination of parameter values were in the right range. There was no learning rate schedule or random augmentation implemented, while the SpecAugment process was always implemented on each run. In the first batch, all models performed relatively well, with a final accuracy ranging between 91% and 95% recall. The top two performers were of ResNet-18 architecture, achieving accuracies of 94.64% and 94.58% respectively, with the ViT model being close behind at 94.42%. As the scores were all close, no conclusive evidence can be found favoring one model over the other. An unexpected finding was the non-difference between models trained on spectrograms overlayed with pink noise, and spectrograms without pink noise.

There were multiple signs which point to these models overtraining on the data. Due to the fact that all vocalisations were annotated with the sound being centered,

LR	Decay	Pink Aug.	Spec Aug.	Moved.	Arch.
5e-05, 1e-04, 2e-04	None	T, F	T	F	ViT, ResNet-18
1e-06, 1e-05	CosineANW	T, F	T, F	T	EN-B0, ViT, ResNet-18
1e-05, 1e-04	CosineAN	T, F	T, F	T	EN-B0, ViT, ResNet-18

Figure 3.1: This table shows the combination of parameters tested per batch. Only ViT and ResNet-18 were used in the first batch to limit the number of runs and gauge if the other parameter values were optimal.

		ViT	ResNet-18
LR=2e-04	Pink-Noise	0.9171	0.9418
	No Pink-Noise	0.9169	0.9405
LR=1e-04	Pink-Noise	0.9366	0.9464
	No Pink-Noise	0.9387	0.9405
LR=5e-05	Pink-Noise	0.9363	0.9458
	No Pink-Noise	0.9442	0.9427

Figure 3.2: Accuracy results of first batch, each model trained with no learning decay, and Spectrogram Augmentation. Every model in this batch performed well with every combination of parameter values. All ResNet-18 models have over 94% accuracy, and implies this range of learning rates was optimal for this architecture. The ViT models were also seen to be robust, with all accuracy scores being over 91%. The top four ResNet-18 and ViT model performances were highlighted.

these models may be overfitted on sounds centered in the viewing window, which would not optimal when used on streamed audio as the sound could be positioned anywhere. The stochastic trends of the validation loss in Figure 3.3 were also indicative of overfitting, with some models even seeing upwards trends in final epochs, where the expectation would have been the loss gradually decreasing.

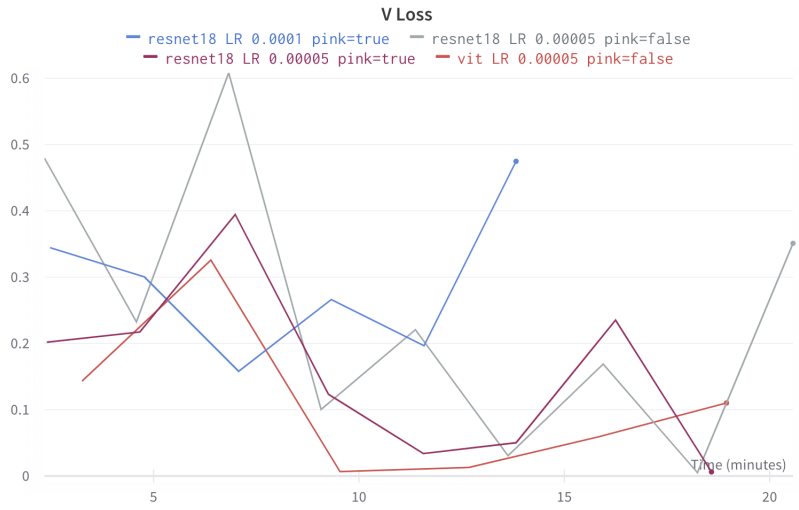


Figure 3.3: Validation loss for the top four performers. For reference, the loss trend for a non-overfitting CNN would gradually converge towards the minima loss (Rice, Wong, and Kolter, 2020). The trend lines shown in this diagram were shown to be erratic, at times shooting up and in the next epoch abruptly decreasing. This was a sign of the model learning too much in one epoch, and indicates the start of overfitting and a learning rate too large.

Twenty four models were trained in the second batch of tests, where the EfficientNet-B0 architecture was introduced, and twenty four more were trained in the third. In these batches, the starting learning rates were lowered and, cosine annealing was implemented to minimise divergence of the loss functions (Dauphin, De Vries, and Bengio, 2015). Random horizontal translation was also implemented in an attempt to guide the overfitting issues in the previous batch. The goal of these two batches was

to determine the more optimal learning rate scheduler, between slowly minimising the learning rate with cosineAN in the second batch or rapidly decreasing the learning rate before restarting it with cosineANW in the third batch.

		ViT	ResNet-18	EfficientNet-B0
LR=1e-05	Pink+SpecAug.	0.8686	0.879	0.8441
	Pink	0.8712	0.8642	0.8459
	SpecAug.	0.8686	0.8707	0.8284
	No Aug.	0.8847	0.8703	0.838
LR=1e-06	Pink+SpecAug.	0.8681	0.7699	0.5341
	Pink	0.8703	0.7856	0.5127
	SpecAug.	0.8651	0.7856	0.5616
	No Aug.	0.8707	0.7943	0.5157

Figure 3.4: Validation Accuracy table for batch 2 with horizontal translation and cosineANW. The best models in this batch were both ViT models with learning rates of 1e-05. The ResNet-18 model struggled to reach 80% with a learning rate of 1e-06 and the EfficientNet-B0 model barely reached 50% with the same learning rate.

		ViT	ResNet-18	EfficientNet-B0
LR=1e-04	Pink+SpecAug.	0.8489	0.8572	0.8624
	Pink	0.8799	0.8742	0.8642
	SpecAug.	0.8598	0.8603	0.8563
	No Aug.	0.8616	0.8454	0.8555
LR=1e-05	Pink+SpecAug.	0.8716	0.8821	0.8533
	Pink	0.8751	0.8786	0.8489
	SpecAug.	0.8668	0.8803	0.8511
	No Aug.	0.8712	0.8825	0.859

Figure 3.5: Validation Accuracy table for batch 3 with horizontal translation and cosineAN. In this batch, the best two models were of ResNet-18 architecture, although they did not perform better than the ViT models from the previous batch. ResNet-18 and EfficientNet-B0 performed better with cosineANW, especially when comparing the models with LR=1e-05. The performance for ViT appeared to be more or less similar.

While the two best performing models in both batches used the ViT architecture, there was no single model with a significantly higher accuracy than the other models. However, there were some models that clearly struggled to converge to an acceptable accuracy, such as ResNet-18 trained at a rate 1e-06, and EfficientNet-B0 trained at a rate 1e-06, which struggled to even reach 50% accuracy. As these models performed well with the higher learning rate of 1e-05, it appears that using the learning rate of 1e-06 made the models converge too slowly, and overstep the one hour training limit. The ViT model however consistently converged to a high accuracy with all learning rates, with only one of its models barely dipping below 85% accuracy. ResNet-18 and EfficientNet-80 on the other hand seem to require a learning rate of over 1e-05 to be able to converge at a high accuracy. Additionally, with random horizontal translations now implemented, the peak performance for both batches was 88%,

which was approximately 6% lower in performance to peak models without random horizontal translation. This fit the expectation of the augmentation, which was to prevent overfitting. This effect is seen in the validation loss trends, which did not reach as low of values as the validation loss trends from the first batch due to less overfitting, yet fluctuated less. In contrast, the other combinations of augmentations did not make a difference in validation performance.

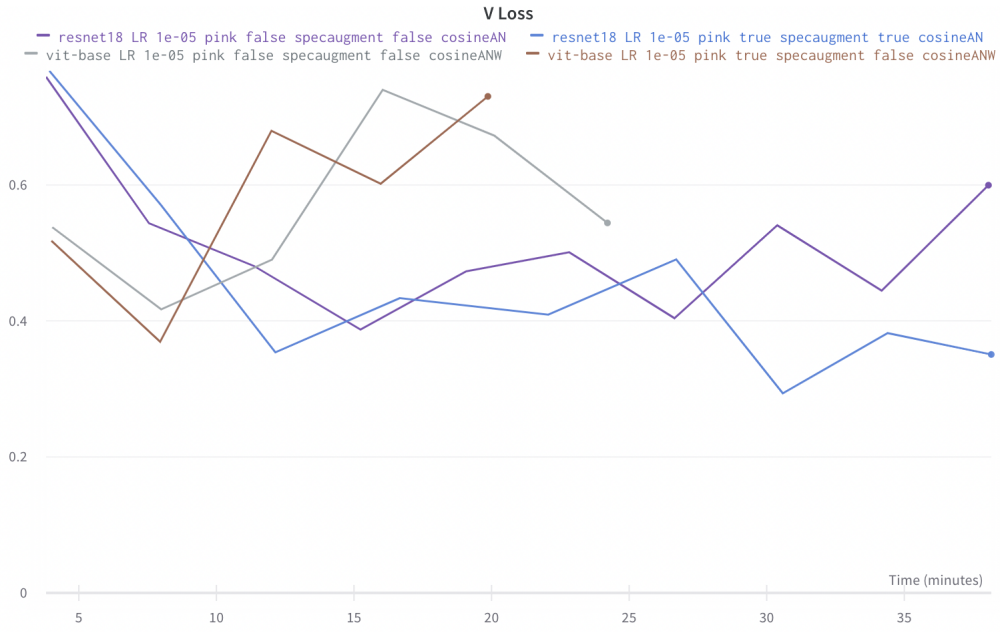


Figure 3.6: Validation loss for 4 best performers from batch 2 3. Although loss was higher across the board, stochastic nature of trends was noticeably muted compared to the trends in Figure 3.3.

3.2 Testing Performance

Based on validation performance, the best performing ResNet-18 model without horizontal translation and a ViT model with horizontal translation and suitable learning rate schedule were benchmarked using the testing data. Despite their differences in validation performance, their testing accuracy, recall and precision were very similar, with the metrics of accuracy, recall and precision only having a 1% difference. The ViT model has performed to the expectation set during validation, while the ResNet-18 model has suffered from slight overfitting to the validation set. When inspecting each model's performance across each category using a confusion matrix, results were also very good across all categories. These models were more than sufficient in classifying the correct Humpback Whale sounds.

Model	Loss	Epochs	Accuracy	Recall	Precision
ViT cosineANW LR=1e-06	0.56	12	0.89	89	0.89
ResNet-18 LR=5e-05	0.62	7	0.9	0.9	0.9

Figure 3.7: Testing results for the best performing ResNet-18 and a ViT model. Despite a moderate significance in validation performance, testing metrics for both models were comparable, implying that the ResNet-18 model was overfit on training/validation data. Pink noise augmentation and horizontal translation have been implemented for both models, due to their theoretical improving of model performance.

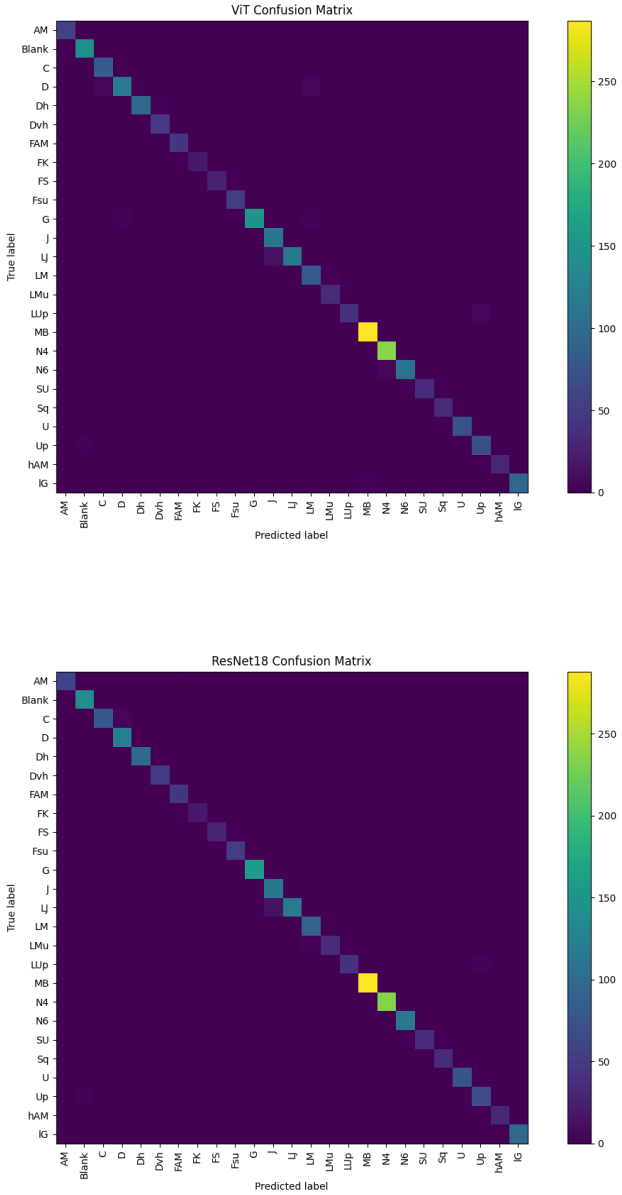


Figure 3.8: Confusion Matrix for both models. The optimal result for this diagram is for the cells along the diagonal to be as close to yellow as possible and the surrounding areas to stay as shades of purple, which means the model is correctly classifying the sounds. When used on testing data, both models performed at a high level in differentiating sounds.

3.3 Performance on Streamed Audio

Both models exhibited passable classification accuracy of the correct sounds in streamed data. This is seen by the moderate correct classification rate and low false positive rate across most audio streaming parameter values. Both models were effective at avoiding false positives to the point where using a voice activity detection threshold did not change the metrics notably. However, the rate of misclassifying vocalisations as blank sounds was problematically high, and the main reason the

ResNet-18 Streamed Audio Performance

Max	SM T.	VAD T.	Correct	Incorrect	False Neg.	False Pos.	Pred:Real
3	0.6	0.5	0.319	0.006	0.644	0.053	1.092
3	0.2	0.5	0.469	0.012	0.5	0.05	1.963
1	0.2	0.5	0.405	0.018	0.558	0.049	1.555
8	0	0.5	0.862	0.003	0.117	0.063	11.46
3	0	0.5	0.66	0.006	0.316	0.056	4.626
1	0	0.5	0.405	0.018	0.558	0.049	1.561
3	0.6	0	0.377	0.018	0.549	0.044	1.31
3	0.2	0	0.528	0.064	0.38	0.056	2.209
1	0.2	0	0.454	0.067	0.445	0.054	1.813
8	0	0	0.911	0.018	0.052	0.072	11.687
3	0	0	0.733	0.074	0.175	0.059	5.107
1	0	0	0.454	0.067	0.445	0.054	1.822

Figure 3.9: Results taken from ResNet-18 model deployment. Max represents maximum number of sounds predicted in one window, SM T. represents the *LogSoftmax* threshold, while VAD T. represents the VAD threshold. Performance improved when increasing the maximum amount of predicted items, at the cost of more sounds being predicted. The result with the most balanced correct classification rate, false negatives, and ratio of predicted to real sounds was coloured teal, while the result with the largest predicted-to-real ratio was colored red, and is an undesirable result.

correct classification rate can be quite low. In fact, for the majority of tests, the expectations to classify a model as optimal were not met, with the correct classification rates lower than the expectation of 80%, and false negative rates far from 0%. The correct classification rate and false negative rate can be improved by raising the maximum number of items allowed in a single window, with the best rates being a correct classification rate at 91.1% and a false negative rate of 5.2%. However, the ratio of items predicted would drastically increase, resulting in the model classifying a lot of extra sounds that do not exist.

The ViT models slightly edges out the ResNet-18 models in performance, due to generally higher correct classification rates and lower false negative rate. When comparing the most balanced performances for each model (marked by teal colouring in the tables), the ViT model achieved marginally better scores, with a correct classification rate of 54.9%, false negative rate of 32.8%, and a predicted-to-real ration of 1.939. The rates for incorrect classification and false positive rates were always low, and as such, were not a heavy point of concern for the comparison, but it is interesting to note that ViT models produce slightly higher incorrect classifications than the ResNet-18 models.

ViT Streamed Audio Performance

Max	SM T.	VAD T.	Correct	Incorrect	False Neg.	False Pos.	Pred:Real
3	0.6	0.5	0.408	0.009	0.555	0.05	1.049
3	0.2	0.5	0.518	0.018	0.445	0.05	1.905
1	0.2	0.5	0.472	0.018	0.488	0.049	1.555
8	0	0.5	0.856	0.006	0.12	0.064	10.957
3	0	0.5	0.644	0.015	0.322	0.062	4.485
1	0	0.5	0.472	0.018	0.488	0.049	1.555
3	0.6	0	0.436	0.028	0.491	0.047	0.969
3	0.2	0	0.549	0.101	0.328	0.057	1.939
1	0.2	0	0.512	0.101	0.359	0.06	1.537
8	0	0	0.88	0.031	0.071	0.076	10.383
3	0	0	0.693	0.083	0.206	0.071	4.859
1	0	0	0.512	0.101	0.359	0.06	1.537

Figure 3.10: Results taken from ViT model deployment. The same patterns from the ResNet-18 model were also observed here. Using the same parameter values from the ResNet-18 table, ViT marginally outperformed ResNet-18, especially when comparing the results in teal.

3.4 Speed and Practicality

Training these models was proven to be capable on a single-GPU computer with a one hour restriction. Inference time was low across all models, with the mean time of inference starting at 7 seconds (333 images per second) for the first epoch, and slowly decreasing after every epoch.

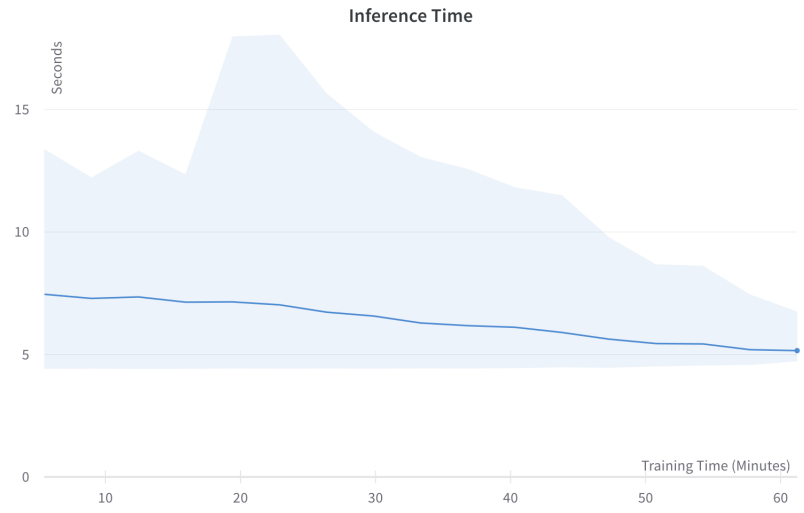


Figure 3.11: Inference time recorded on validation data against training time.

Chapter 4

Discussion

4.1 Validation and Testing Performance Comparison

Most of the trained models performed very well and achieved a validation accuracy of 85% or higher, despite imposing a one hour training limit. However due to the limitation, there is a large possibility that each model has not had the sufficient time to converge to converge to their potential performance peaks. This was an expected result, and one of the risks taken when choosing to have training time restrictions. Investigations in the future will have to require more computing power and longer training limits in order to train the models to their maximum validation performance.

ViT models were the most robust in this investigation, being the least influenced by parameter changes and always outputting high performances. This was to the expectation set at the beginning of the experiment due to the many more FLOPS it processes during training over the other networks. While ResNet-18 and EfficientNet performances were strongly dictated by the learning rate, ViT models were able to achieved a validation accuracy of over 84% across all learning rates and parameters. As such, ViT would be a great architecture to use for scientists who do not necessarily have the time or knowledge to tinker with many parameter values.

Comparing the effects of difference augmentation processes, while models which were not trained with horizontal translation had worse validation accuracies, these performances closer resembled their performances on the testing set, while models trained without horizontal translation were not as close in validation and testing performance. It is much better for a validation set to closer represent the testing so that scientists have a better idea of what to expect from their models, which is why random horizontal translation is recommended when replicating this experiment, alongside the additional robustness it gives the model. What was not expected however was the other augmentations (SpecAugment and pink noise overlay) not affecting the validation performances in a noticeable manner. Future investigations may require the magnitude of theses transformations to be increased to observe any differences between models trained with and without them, and compared in regards to their testing performance, which was not done in this paper.

During classification on streamed data, the ViT model with horizontal translation showed superior performance, where more often than not, ViT models seemed would have a higher correct classification rate and lower false negative rate than the ResNet-18 model without horizontal translation. The ViT models do have marginally higher incorrect classification rates, though the increases were not as large as the decreases in false negative rates. It appears that the ResNet-18 model was more likely to predict

a blank noise than the ViT, hence the lower incorrect classification rate and higher false negative rate.

4.2 Applications and Practicality

Assuming one had access to similar computing resources an RTX3080 provides, It is possible to use this workflow to train a model with a low inference time (as the times recorded during validation far surpassed the expectation of 10 images classified per second) and good validation performances. Due to many models achieving similar performances despite being trained on different parameter values, this suggests that the models have not converged to their peak performances, and as such, a longer training time is recommended. While performance on the streamed audio is passable, none of the models produced can be considered optimal for streamed audio as they did not perform to the expectations for an optimised model. More improvement is required in future studies.

The optimal parameter values to be used when classifying live data are dependent on the goals of the visions the scientists have of the project. If one wanted to simply detect the presence of Humpback Whales and put less attention on identifying the correct song unit classification, one would use a high maximum item prediction threshold, and low thresholds for the prediction probability and energy threshold. Despite setting the thresholds to extreme values, the false positive and incorrect rates still remain low. The side effect of heavily raising the maximum sounds to predict at once is the increase of the predicted sound to actual sounds ratio. This is terrible for scientists that want to accurately label each sound, as this setting will label too many categories instead of the accurate category sound. If a scientist wanted to do this, perhaps for a song analysis, a higher maximum prediction threshold (such as 0.2 used in Figure 3.10) should to be set to decrease the prediction-to-real ratio while maintaining respectable correction classification and false negative rates. It is more important for the model to not over classify the audio database compared to slightly improving with correct classification or avoiding false negatives. It is also notable that models with low *log softmax* thresholds have a large prediction-to-real ratio, which in turn inflates the correction classification rate.

This paper provides a set of procedures which can be followed by other scientists without too much computing power or knowledge in CNNs. The program used in this paper is open-source, and can be downloaded and used by others. It contains all the data pre-processing modules, model training procedures, and data augmentation functions so users will not have to produce their own. Basic coding knowledge is required however to know where to change the parameter values and file paths however. When fully optimised, a CNN model will be able to be deployed to enhance the capabilities of PAM and assist scientists with the monitoring of Australian Humpback Whales, and possibly other underwater species if the models are trained with their audio.

Chapter 5

Conclusion

In this study, many competent models were trained through careful experimentation using a program that can be used by scientists to follow the efficient workflow outlined in this paper. Using the ViT architecture with horizontal translation, one is able to achieve great results on a wide variety of parameter values without overfitting, meaning it is the most convenient architecture to use, requiring the least time to optimise. The effects of SpecAugment and pink noise overlay were not seen in this paper howerver, and require further investigation by amplifying their augmentations and observing their effects on model performance. Despite time and resource constraints on model training, the performances of each model still ranged from good to great, proving that high computation power was not required. However, a time limit of over 1 hour was required to allow these models to converge to their potential performances. Future studies based on this work are definitely required to polish each model's performance on streamed audio however, as the correct classification and false negative rates leave a lot to be desired. For a scientist to fully rely on such a model, one would need to implement smart algorithms which can assist the neural network with accurate classifications and little false positives or negatives, and tightly bind the start and end times around the sounds rather than the default 2.7s window used in this paper. The code repository for the program developed for this paper can be found at *https://github.com/kasmello/Humpback-Neural-Network*.

Bibliography

Allen, Jenny A et al. (2018). "Cultural revolutions reduce complexity in the songs of humpback whales". In: *Proceedings of the Royal Society B* 285.1891, p. 20182088.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). *Neural Machine Translation by Jointly Learning to Align and Translate*. DOI: 10.48550/ARXIV.1409.0473. URL: https://arxiv.org/abs/1409.0473.

Bengio, Yoshua (2012). "Practical recommendations for gradient-based training of deep architectures". In: *Neural networks: Tricks of the trade*. Springer, pp. 437–478.

Brigham, E. Oran (1988). *The Fast Fourier Transform and Its Applications*. USA: Prentice-Hall, Inc. ISBN: 0133075052.

Dauphin, Yann, Harm De Vries, and Yoshua Bengio (2015). "Equilibrated adaptive learning rates for non-convex optimization". In: *Advances in neural information processing systems* 28.

Dosovitskiy, Alexey et al. (2020). "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929*.

Erbe, Christine (Jan. 2013). "Underwater passive acoustic monitoring noise impacts on marine fauna - a workshop report". In: *Acoustics Australia* 41, pp. 113–119.

Gibbs, Josiah Willard (1902). *Elementary principles in statistical mechanics: developed with especial reference to the rational foundations of thermodynamics*. C. Scribner's sons.

Gong, Yuan, Yu-An Chung, and James Glass (2021). "Ast: Audio spectrogram transformer". In: *arXiv preprint arXiv:2104.01778*.

He, Kaiming et al. (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Kundu, Rohit et al. (Sept. 2021a). "Pneumonia detection in chest X-ray images using an ensemble of deep learning models". In: *PLOS ONE* 16, e0256630. DOI: 10.1371/journal.pone.0256630.

— (Sept. 2021b). "Pneumonia detection in chest X-ray images using an ensemble of deep learning models". In: *PLOS ONE* 16, e0256630. DOI: 10.1371/journal.pone.0256630.

Lezzoum, Narimene, Ghyslain Gagnon, and Jérémie Voix (2014). "Voice activity detection system for smart earphones". In: *IEEE Transactions on Consumer Electronics* 60.4, pp. 737–744. DOI: 10.1109/TCE.2014.7027350.

Loshchilov, Ilya and Frank Hutter (2016). "SGDR: Stochastic Gradient Descent with Restarts". In: *CoRR* abs/1608.03983. arXiv: 1608.03983. URL: http://arxiv.org/abs/1608.03983.

Mazhar, Suleman, Tamaki Ura, and Rajendar Bahl (2007). "Vocalization based Individual Classification of Humpback Whales using Support Vector Machine". In: *OCEANS 2007*, pp. 1–9. DOI: 10.1109/OCEANS.2007.4449356.

McCauley, Robert D et al. (2017). "Developing an Underwater Sound Recorder: The Long and Short (Time) of It..." In: *Acoustics Australia* 45.2, pp. 301–311.

Nicklin, Charles P., James D. Darling, and Meagan E. Jones (2006). "Humpback whale songs: Do they organize males during the breeding season?" In: *Behaviour* 143.9,

pp. 1051 –1101. DOI: https://doi.org/10.1163/156853906778607381. URL: https://brill.com/view/journals/beh/143/9/article-p1051_1.xml.

Park, Daniel S et al. (2019). "Specaugment: A simple data augmentation method for automatic speech recognition". In: *arXiv preprint arXiv:1904.08779*.

Payne, Roger S and Scott McVay (1971). "Songs of Humpback Whales: Humpbacks emit sounds in long, predictable patterns ranging over frequencies audible to humans." In: *Science* 173.3997, pp. 585–597.

Rice, Leslie, Eric Wong, and Zico Kolter (2020). "Overfitting in adversarially robust deep learning". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 8093–8104. URL: https://proceedings.mlr.press/v119/rice20a.html.

Shokouhi, Navid (2020). *py_vad_tool [Source Code]*. URL: https://github.com/idnavid/py_vad_tool/blob/master/unsupervised_vad.py.

Tan, Mingxing and Quoc V. Le (2019). "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: DOI: 10.48550/ARXIV.1905.11946. URL: https://arxiv.org/abs/1905.11946.

Targ, Sasha, Diogo Almeida, and Kevin Lyman (2016). *Resnet in Resnet: Generalizing Residual Architectures*. DOI: 10.48550/ARXIV.1603.08029. URL: https://arxiv.org/abs/1603.08029.

Tolkova, Irina et al. (2017). "Automatic classification of humpback whale social calls". In: *The Journal of the Acoustical Society of America* 141.5, pp. 3605–3605.

Vaswani, Ashish et al. (2017). "Attention is all you need". In: *Advances in neural information processing systems* 30.

Ward, L. M. and P. E Greenwood (2007). "1/f noise". In: *Scholarpedia* 2.12. revision #137265, p. 1537. DOI: 10.4249/scholarpedia.1537.

White, Ellen L. et al. (2022). "More than a whistle: Automated detection of marine sound sources with a convolutional neural network". In: *Frontiers in Marine Science* 9. ISSN: 2296-7745. DOI: 10.3389/fmars.2022.879145. URL: https://www.frontiersin.org/articles/10.3389/fmars.2022.879145.

Zagoruyko, Sergey and Nikos Komodakis (2016). *Wide Residual Networks*. DOI: 10.48550/ARXIV.1605.07146. URL: https://arxiv.org/abs/1605.07146.

Zimmer, Walter MX (2011). *Passive acoustic monitoring of cetaceans*. Cambridge University Press.