Introduction to Deep Learning Project
Deep Networks for Possum Call Detection

Dwane van der Sluis Richard Sterry Filip Svoboda Hadrien de Vaucleroy

Abstract

Identifying the presence of a targeted species in a pest trap is a much sought-after component of smart pest control in New Zealand. In this project we develop the case for using, and a prototype of deep learning for the task of detecting possums. We collected and annotated a small dataset of short audio clips of possums and other labelled creatures. We use it to investigate the comparative performance of three network architectures. We address key questions about noise to signal ratio and the use of MFCC features. Finally, we discuss our plans and progress on a prototype real-time possum call detector based on deploying a trained GoogleConv network to a Raspberry Pi 3 device.

1. Introduction

1.1. The Possum Problem

New Zealand's ecosystems have, due to 80 million years of separate evolution, a disproportionally large composition of endemic fauna and flora species (Cooper & Millener, 1993). Roughly 80% plants and 40% animals are unique to the islands, making their ecosystems particularly vulnerable to alien life (Lindsey & Morris, 2000). Ironically, the most vulnerable region in the world was introduced to the largest number of these species (Williams & Timmins, 2002).

The foreign species are causing very significant damage both to the environment and to the economy of New Zealand. Almost half of the indigenous plant species, and numerous animal species went extinct as a direct consequence of the aliens predation or competition (Holdaway, 2009). Every year, the economy suffers 3.3 billion NZD, roughly 2% of GDP, in lost productivity. While the government spends circa 70 million NZD (93% of the Environments appropriations) on pest-control programs (McGlone, 2014).

The brushtail possum was introduced to New Zealand by settlers in 1837 as an economic animal for its fur. It spread to the forest almost immediately and since then has been one of the prime contributors to the devastation of New Zealands nature. It prays on the chicks of the endangered kokako, kereru, kiwi, piwakawaka, titi, tui birds and on the harrier hawk. It consumes 21,000 tonnes of vegetation each night, and it spreads the highly contagious bovine tuberculosis. Its population, despite extensive hunting, stabilised at around 30 million (Meyer, 2000), as shown in Figure 1.

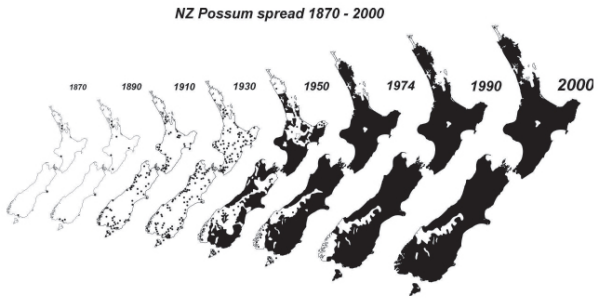

Figure 1. Possum spread 1870 - 2000. *Source: Landcare Research*

1.2. Pest Control Techniques

The main possum control methods are poisoning and trapping. These either kill, maim, or capture any animal lured to the bait. They cause substantial collateral damage as the flightless birds (e.g. kiwi, weka) and domestic companions are equally likely to take the bait. Poisons, leach into water systems, and are subject to diminished effectiveness as immunity eventually arises in surviving populations. Because of these side effects, the methods are deployed only in remote areas and are not available around human dwellings and in nature reserves. These then become safe havens for pests.

Given the present state of the pest-control technology, saving New Zealands ecosystem is impossible (Griffiths et al., 2015). Therefore, novel approaches are actively being sought by the New Zealand government. Two main approaches are emerging. The conservationists are most excited about the deployment of biocontrols. These are artificially created adversarial genes introduced in the population through genetically altered individuals. They take roughly 10 generations to spread to the full population and through

gradual mutation cause eventual extinction (Gantz, 2015). This method, while extremely effective, carries even larger risks of collateral damage as infected possums might cross to Australia where even a single individual could cause serious harm to the, there indigenous, possum population. Finally, it has not been tested outside of the mosquito species and the laboratory environment.

Our method follows the suggestion of Dr. Griffiths, of the Island Conservation environmental group, and focuses on automated traps that target specified animals only (Griffiths et al., 2015). Specifically, we focus on the machine-learning aspect of such a trap. Given the significantly decreased collateral damage to other animals, these traps would be suitable for extensive deployment throughout the islands. Furthermore, the technology needed for these traps is widely used in variety of hot key recognition applications (e.g. personal assistants).

1.3. Our Project

Our project was inspired by the experience of one of our team members with NZ possums. We then contacted the Cacophony Project[1], an NGO partner to the governmental pest-control program. They provided us with a small sample of possum calls, which were augmented, on which we are developing and testing the architectures. Our contribution will serve as a preliminary proof of concept and as a foundation for further development and refinement of integrated analytic and mechanical solutions. These will not be limited to the smart trap, but will also include a variety of audio-based scanners, locators, and lures. Nevertheless, to scope the project well, we focus on identifying possum calls only in this paper.

Our paper is organised as follows.

In Section 2, we introduce the methodology and code we adapted from Google's 'Simple Audio Recognition' Tutorial[2]. We discuss how we assembled our dataset, and present the architectures we used: GoogleConv, AlexNet and DeepEar.

In Section 3, we present the results of our experiments, comparing the individual networks and discussing some experiments designed to test robustness.

Section 4 discusses the work we have been doing to move from this proof-of-concept network to a working prototype deployed on a mobile device.

We follow in Sections 5 and 6 with suggestions for the future development of the project and conclude with final remarks.

[1] https://cacophony.org.nz/

[2] https://www.tensorflow.org/versions/master/tutorials/audio_recognition

2. The Methods and Methodology

Recognising the presence of a possum call in a sample of recorded bush sounds can be framed as in instance of the hot key recognition task. The data come as a continuous streak of n-second long overlapping windows which are transformed into visual representations the spectrograms. These are then analysed for the presence of the sought-after call, which is often detectable as a visual distortion of a particular shape and intensity in the background image.

First, we describe the candidate methods. Then, we present an overview of the data. Finally, we detail the training and evaluation sample generation process.

2.1. Network Architectures

The presented solution investigates three well-understood and well-documented architectures: GoogleConv, AlexNet, and DeepEar.

GoogleConv is our benchmark implementation. It's the default network used in the 2017 Google Tutorial, and is a simple convolutional network based on (Sainath & Parada, 2015). Its architecture consists of 2 convolutional layers followed by a fully connected layer. It was original deployment as a simple key word recognition network.

Layer	Layer Type	Size	Output Shape
1	Conv/ReLU	64x20x8	
2	Max Pooling	2x2, stride 2	(99,20,64)
3	Conv/ReLU	64x10x4	(99,20,64)
4	FC/ReLU	126720 hidden units	6

Table 1. GoogleConv architecture: 940,000 parameters

AlexNet, a network originally conceived to analyse images, is an architecture consisting of 5 convolutional layers followed by 3 fully connected layers. It has delivered state-of-the-art performance on standard image recognition datasets such as the ImageNet (Krizhevsky et al., 2012). Since then, it has been also used for analysing the spectrograms in auditory tasks similar to ours. Our implementation of AlexNet differs slightly from the original to compensate for the different image size and aspect ratio.

Layer	Layer Type	Size	Output Shape
1	Conv/ReLU/Norm	64x20x8, stride 4	
1	Max Pooling	3x3, stride 2	(25,5,64)
2	Conv/ReLU/Norm	128 10 x 4 filters	
2	Max Pooling	3x3, stride 2	(13,3,128)
3	Conv/ReLU	128x10x4	(13,3,128)
4	Conv/ReLU	128x10x4	(13,3,128)
5	Conv/ReLU	128x10x4	
5	Max Pooling	3x3, stride 2	(7,2,128)
6	FC + ReLU	1792 hidden units	512
7	FC +ReLU	512 hidden units	512
8	FC +ReLU	512 hidden units	6

Table 2. AlexNet architecture: 3,488,326 parameters

DeepEar, is a model designed for audio recognition task with constrained use. It consists of multiple coupled DNN networks, each focusing on a different task in the overall analysis of the auditory input (e.g. emotion, speaker, or ambient sound recognition) (Lane et al., 2015). In our implementation we use the emotion recognition DNN of the full system.

Layer	Layer Type	Size	Output Shape
1	FC/ReLU	4096	4096
2	FC/ReLU	4096	4096
3	FC/ReLU	4096	4096
4	FC	4096	6

Table 3. DeepEar architecture: 82,460,672 parameters

2.2. Dataset

Data collection was the biggest challenge we faced. It took a number of iterations before we arrived at a dataset of acceptable quality.

Given the lack of access to New Zealand wildlife in the UK we were forced to synthetically compose our dataset from various sources. Almost without exception, the data was downloaded from web sources: we are very grateful to those who kindly make such data publicly available under Creative Commons licences[3]. There are some excellent audio libraries on the internet[4] but generally their data is not standardised or easily amenable for bulk downloading. Most of our data therefore came from Kaggle competitions and an academic competition called the Bird Audio Challenge (Stowell et al., 2016).

A dataset where each sound category comes from a separate source (i.e. one source for possums, one for birds...) carries the serious risk that the classifiers will learn the features of the *sources* of the recordings, rather than the features of the recorded calls. We attempted to limit this by ensuring comparable sound quality and clear recordings.

2.2.1. Possum Data

Possum call records were built up from a number of sources, these included the Australian National Film and Sound Archive: nfsa.gov.au, and filed recordings from New Zealand made by the Cacophony Project. Each recording contained multiple calls; each call was extracted into an individual file up to 2 seconds long. This produced 194 mono 16KHz recordings.

2.2.2. Non-Possum Data

As well as a set of possum call samples, we needed similar-sized sets of audio samples of other creatures so that the network could be trained to discriminate between possums and other classes. Ideally, our non-possum classes would have consisted of animal and bird sounds that are likely to be encountered in the environments of New Zealand where the possums need to be identified. In practice, we had to settle for classes where we could find sufficient quality, labelled data: **cats**, **dogs** and **birds**.

These files came from a number of sources (see table), and all needed cutting and annotation to yield 2 second clips in a standardised format. The original files were mostly around 10 seconds long and generally included multiple signals, e.g. a typical cat file was 12 seconds long with four clearly identifiable miaows separated by quiet sections. We wrote a simple algorithm to extract signals by splitting around sustained amplitude peaks in each file, and then manually spot-checked the quality of the results. This worked well for the cat and dog data, but the bird data was of much lower quality and there were many mis-labellings; we therefore set up a process to manually accept or reject the proposed chunks and only used samples that wed checked. We also tagged files carefully to ensure that all signals extracted from a single source file would appear in the same dataset split (training/validation/testing.)

All data was converted to mono, 16KHz, 16-bit .wav files.

We also needed a larger set of samples of other types of audio events so that the network can train an **unknown** class: its important the network is trained to recognise that sounds other than those from the known classes are possible, otherwise it is likely to generate too many false positives in out-of-sample testing. We assembled a set of clips that in-

[3]https://creativecommons.org/licenses/

[4]Xeno Canto; Macaulay Library; British Library; Animal Sound Archive Berlin

cluded chainsaws, aeroplanes, church bells, car horns, sheep, frogs, crickets, human speech and footsteps among many others. Again, all files were processed into standardised 2s .wav files centred on amplitude peaks.

2.2.3. Background Noise

Finally, we needed a range of longer recordings of background sound scenes that could be used for data augmentation. These included 1-10 minute recordings of (amongst others) rainfall, wind, thunder, creaking trees, traffic noise, and (a personal favourite) a gents public toilet in Dartmoor. These were also converted to mono, 16KHz .wav files.

2.2.4. Final Dataset

Our final dataset is summarised in Table 4. It was not comparable to the dataset used in the Google Tutorial in either size or quality, but with the aid of aggressive data augmentation we believe its big enough to provide some insight and to motivate further research. There is some residual class imbalance but its within tolerance.

Class	Count	Source
Possum	194	NFSA Cacophony Project
Bird	441	Machine Listening Challenge freefield1010 Warblr
Cat	419	Kaggle: Cats & Dogs
Dog	240	Kaggle: Cats & Dogs
Unknown	855	Kaggle ESC50 Kaggle ground parrots Google speech (small subset)
Background Noise	-	British Library Sound Archive 50 files, 3.5 hours audio in total
Total	**2199**	

Table 4. Dataset summary

2.3. Code & Data Pipeline

We used the code from Google's TensorFlow 'Simple Audio Recognition' Tutorial[5] as the benchmark implementation for the project.

The dataset is split into training, validation and testing subsets using a randomization procedure that ensures that samples extracted from each source file are assigned to the same fold. This helps to avoid the unrealistic results that could be caused by having very similar samples in training and testing sub-sets. The randomization is seeded in a deterministic way such that we used the same dataset splits for every experiment, which make the results comparable.

The Google code uses the standard TensorFlow SGD optimiser with two training rate steps, and optimises the cross-entropy loss metric. In the initial stages of our project we didn't change these components.

The network is trained using mini-batches of 100 samples drawn randomly from the training sub-set, with each sample subjected to the data augmentation process described in section 2.3.1 followed by the spectrogram and MFCC transformations discussed in sections 2.3.2 and 2.3.3.

2.3.1. Data Augmentation

In order to minimise overfitting, and maximize utility of the training sample data augmentation can be use. Data augmentation is a regularization technique that artificially increases the size of the dataset by using transformations that do not alter the label to add more similar examples. The data augmentation techniques used on the training samples were a vertical translation of up to 100ms (1600 samples), zero padding, and adding background noise to the samples (described below). By default, background noise augmentation was added to 80% of samples in each training mini-batch. A 2-second clip is selected from one of the noise files and mixed in with a randomly-selected volume up to 10% of sample volume. No noise is added when measuring the validation accuracy.

2.3.2. Spectrograms

The pipeline begins with the time/amplitude-domain waveform which need to be converted into a spectrogram (time/frequency domain). These spectrograms are comprised of spectrums, which are created by 480 samples, and are taken every 160 sample (10ms) interval. These 480 samples are then passed through a fast Fourier transformer (FFT) based Mel-scale filterbank. This is an algorithm which samples signals over a time period and transforms them into their frequency components, which are then non-linearly (log function) transformed to improve recognition performance. Passing the 480 samples through the FTT based Mel-scale filter back returns 40 derivatives that are then combined into a feature vector of 40 'frequency' bins by 198 time-periods. This feature vector is then used in the various neural networks described in this paper. The first two graphs in Figure 2 show the original time/amplitude-domain wave form and the resulting spectrogram.

2.3.3. MFCC

The spectrograms are then applied a further transformation into Mel-Frequency Cepstral Coefficients (MFCCs). This

[5] https://www.tensorflow.org/versions/master/tutorials/audio_recognition

final MFCC transformation step reflects human hearing sensitives. The last graph of Figure 2 shows the output of this stage. Note that both spectrograms look like 'images', with intensities across a 2D domain, which is why applying convolutional layers makes sense. The resolution of the original spectrogram images is of size 198 x 257; the MFCC images are of size 198 x 40.

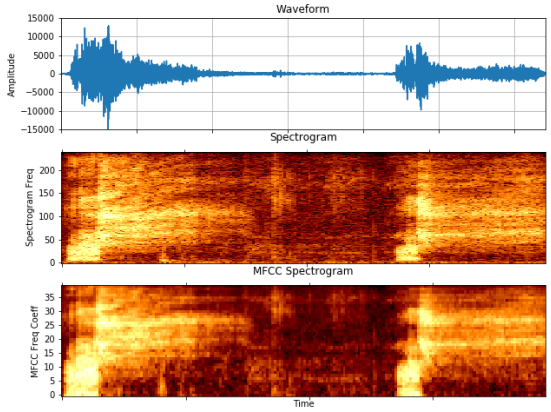

Figure 2. Audio processing pipeline for a possum call sample

The MFCC transformation makes sense when trying to discriminate between different human speech signatures. However, there's no reason to assume that possum calls are well-described by a transformation that represents human hearing. We therefore experimented with removing the MFCC step, and ran the network directly on the spectrogram outputs instead. We examine this assumption more closely in Section 3.5.

3. Performance Analysis

Our experiments were designed to test the feasibility of utilising the very restricted number of possum recordings in the data augmentation methodology developed by Google. First, we compared the relative performance of AlexNet, DeepEar, and GoogleConv architectures in a default setting of the data augmentation procedure described above. Then, we investigated the robustness of the winning algorithm's performance under varying levels of the environmental noise. Finally, we recognised that the anthropocentric MFCC features need not be the optimal features for analysing animal-to-animal communication. Thus, we conclude this section by analysing the relative merits of working with and without the MFCC transformation.

3.1. Benchmark Implementation: GoogleConv

The GoogleConv network achieved more than 80% training accuracy within 1000 steps, which is roughly equivalent to 60 epochs in a setup without data augmentation[6]. There was limited improvement after that; we stopped training after 2000 steps for a final training accuracy of 86%. Validation accuracy was evaluated every 100 steps and also reached 86%. Although validation metrics didn't improve much beyond 1000 steps, there was no evidence of overtraining, which may be due to the 50% dropout probability and data augmentation used in training.

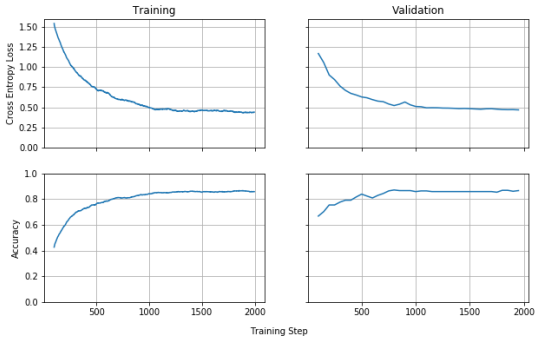

Figure 3. Rolling average loss and accuracy (100 step window) for benchmark GoogleConv network with 30% unknowns

Figure 4 shows the confusion matrix on the validation set. GoogleConv correctly predicted 13 out of the 17 possum samples in the validation set, and made three false positive predictions, which corresponds to 76% recall, 81% precision and an F1-score of 79% on the possum class. It incorrectly classified three gold possum samples as cats, which is worth further investigation. There were no major misclassification difficulties across other classes.

[6]Each step corresponds to a batch size of 100 samples drawn randomly from the training dataset.

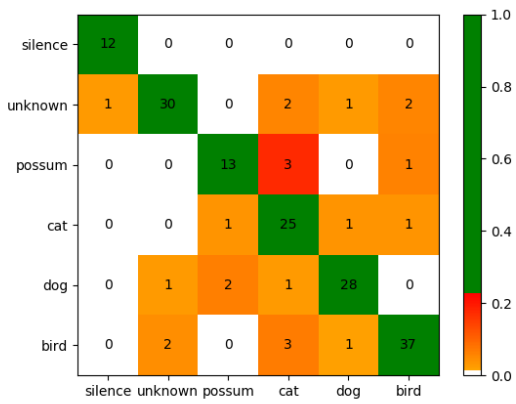

Figure 4. Confusion matrix for GoogleConv network

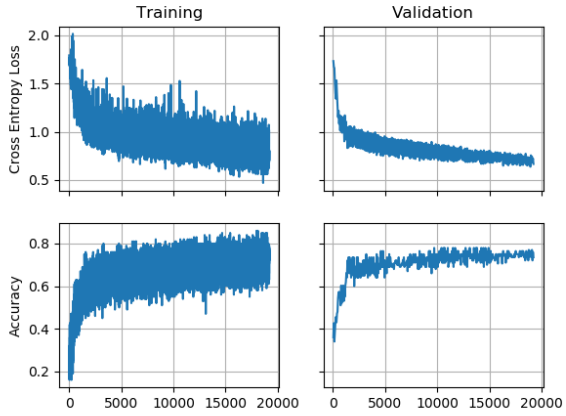

Figure 5. Rolling average loss and accuracy (100 step window) for benchmark DeepEar network with 30% unknowns

3.2. Alternatives: AlexNet & DeepEar

In our initial investigations we aimed to apply AlexNet and DeepEar in exactly the same framework as GoogleConv, so that we could isolate the impact of changing the network architecture. However, in practice it proved difficult to train these architectures using `tf.train.GradientDescentOptimizer` (stochastic gradient descent), which is used by the benchmark implementation. We spent significant amounts of time experimenting with learning rates and when we did eventually manage to train the networks the process took many more steps than for GoogleConv.

We stopped training **DeepEar** after 19200 steps for a final training accuracy of 71%, as shown in Figure 5. Validation accuracy was evaluated every 100 steps and reached 75%.

Figure 6 shows the confusion matrix on the validation set. DeepEar correctly predicted 15 out of the 16 possum samples in the validation set, and made 9 false positive predictions, which corresponds to 93% recall, 62% precision and an F1-score of 75% on the possum class. It incorrectly classified 6 cats samples as possums, which is worth further investigation.

We found **AlexNet** even harder to train than DeepEar, and although we did eventually achieve passable result we decided to switch to using the adam optimiser `tf.train.AdamOptimizer` for all three networks. Training was much faster using adam, and all three networks achieved sensible levels of performance.

3.3. Model Comparison

Figure 7 shows the training and validation set accuracy metrics of the three networks using the adam optimizer.

We see the convolution based methods outperform the DNN DeepEar by a substantial margin. This is explained by the superior ability of the convolution input layer to capture spatial features in the 2D spectrogram representation of the data.

The two CNN-based solutions show relatively similar performance. AlexNets performance takes a couple of more steps to take off, but after that point the two performances move in a close tandem. This delay is due to the much larger parameter space AlexNet is optimizing relative to GoogleConvs (3.5m parameters vs. 1m).

Given the superior performance of the CNNs and the smaller parameter count of the GoogleConv we chose the smaller, more parsimonious, model for our subsequent experiments. Efficiency will be of great importance for the real-world deployment of this technology.

3.4. Noise Level Robustness

Having established that GoogleConv delivers strong performance relative to its number of parameters, we turn to showing its behaviour with respect to the noisiness of the generated data. We will now change the upper bound on the uniform distribution we used before to be in turn 0.1, 0.3, 0.5, 0.8, 1.0, and 2.0. For example, the case 0.3 corresponds to each noise intensity being drawn from U(0, 0.3). Figure 8 shows the observed performance.

Increasing the noise reduces the training accuracy (from 87% at zero noise to 73% at 200%) in a predictable ordered fashion. This is much less so for the validation accuracy where we see that the performances under varying noise intensities are much lees decoupled. Smaller noise levels do

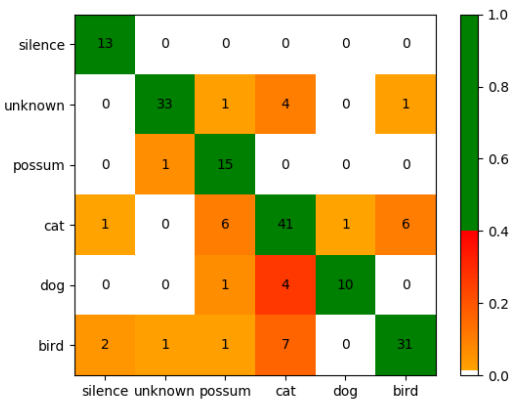

Figure 6. Confusion matrix for DeepEar network

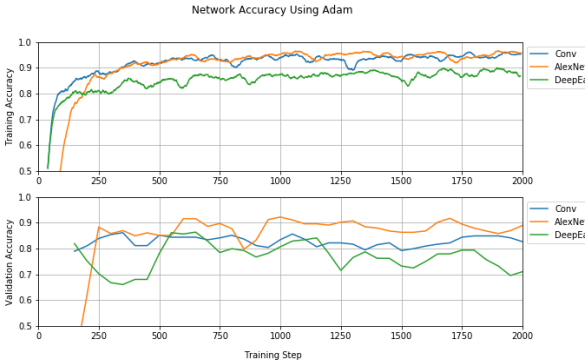

Figure 7. Model comparison using Adam optimiser

tend to be associated with higher validation accuracy, but there are many more violations of the strict ordering we see in the top graph. This suggests that the predictive power of the data feature representations learned by the Google CONV architecture is rather less sensitive to the presence of noise in the training examples. Therefore, we have a reason to expect the GoogleConv model to handle data of heterogeneous origin rather well which could be important if future data collection was handled by local volunteers.

Having established that GoogleConv is preferred, and that it can be expected to work relatively well with data obtained in heterogeneous fashion we now turn to the biggest yet unaddressed issue to what extent are possum calls suitable for the humans-inspired MFCC setup.

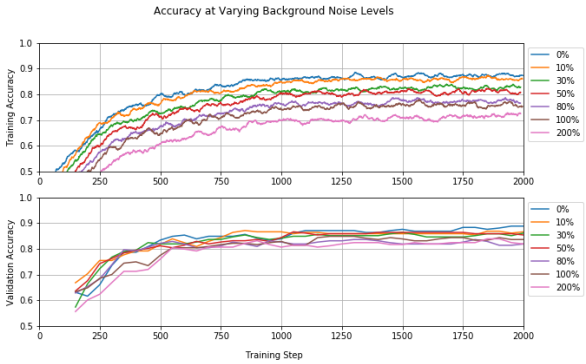

Figure 8. Noise experiments

3.5. MFCC Robustness

The MFCC features are designed to simulate the distortions in the perception of sound waves by humans. They strengthen frequencies humans hear best, and dampens those we are less sensitive to. Possum hearing range is from 0.1 kHz to 10kHz (Osugi et al., 2011), which is slightly lower based than the human range of 0.2 to 20kHz. We can therefore suspect that their optimal hearing frequency range need not be in line with ours. In such a case the MFCCs would be adversely affecting the information contained in the sample, and thus should not be used.

As the original spectrograms have significantly more features than the MFCC version (50,886 vs. 7,920), the training process was much slower to run. We halted it after 1400 steps. As shown in figure 9, the training accuracy grew more slowly than before, but still achieved 85%. However, the validation accuracy never got above 65%. It appears that the extra parameters allow the no-MFCC network to overfit. In our task, the MFCC stage may be valuable because it compresses the inputs, rather than because it transforms frequencies in a certain way.

3.6. Findings

Our analysis shows how even a small non-professional dataset can yield models that generalize rather well. All three models showed encouraging performance, while GoogleConv was chosen as the preferred choice based on improved accuracy relative to its number of parameters. These results hold robust under varying levels of noise in the data generation process. As we showed, even 2:1 noise ratio produces better than 70% accuracy on this small dataset with acceptable rates of false positives for the possum class. The current traps have accuracy of 3%. This, while an unfair comparison, motivates why the observed modest (relative to the canonical Deep Learning applications) performance is in fact a major encouragement to the beneficiaries of this

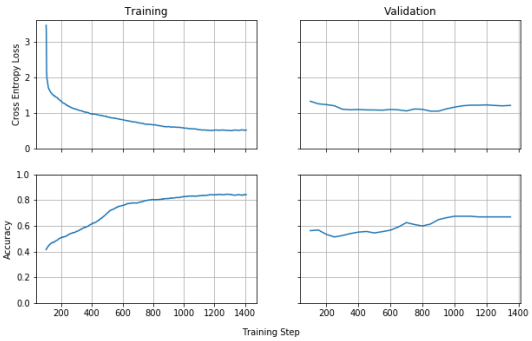

Figure 9. GoogleConv on raw spectrograms

project. We thus conclude that the present analysis gives justification to a professional data collection and, later, targeted architecture design.

4. Raspberry Pi 3 Deployment

After initial success with a laptop deployment, we moved to the target platform; the Raspberry Pi 3. This is because it is cheap, USD$35, and powerful. The Raspberry Pi 3 contains a Broadcom BCM2837 quad-core 64-bit ARM Cortex A53 running at 1.2 GHz, and 1Gb of memory. Non-volatile storage is provided via a micro SD card. As some libraries have no ARM distribution available and 64-bit versions of several libraries are not yet ready for the Pi3, many libraries will need to be compiled from source. We compiled the Bazel 5.4 and Python 3.6.2. Tensor-flow r1.5 source code was modified to use alternative 32-bit libraries. This is still in progress. When finished our prototype inputs real time audio recording of its surroundings and signals each time it hears a possum call. This signal can be used by the smart trap, lure system, or other yet to be developed applications.

5. Conclusions

This project sought to demonstrate the potential Deep Neural Networks hold for the Environmental Conservation efforts in the New Zealand, and more broadly around the world. It set out to investigate the limits data availability and computational resources place on the Cacophony Project in their mission to develop a smart possum trap. It used a basic data set on which it trained three canonical deep architectures (GoogleConv, DeepEar, AlexNet). A series of tests and experiments revealed a marginal preference for the GoogleConv; this was then further investigated for noise level robustness and reliance on the MFCC spectrogram transformation.

We found that GoogleConv held to its satisfactory performance under a wide range of noise to signal ratios. Also, we found that the MFCC features were necessary for its sustained performance. These results encouraged us to seek prototypical implementation on a Raspberry Pi 3 device.

6. Next Steps

This project presented a beta version of an analytical engine which is meant to sit in a device deployed in the wild, catching possums. In the constraints of our setup, it showed very good performance in identifying possums and other animals. Nevertheless, several issues ought to be addressed before the concept can be deployed.

We worked with limited and noisy samples. Moreover, our example classes came from different sources and different recording devices. This carries the potential that the network learned to recognise unique footprint of the recording method or devices, rather than the calls themselves. A professional sample ought to be collected. Such collection should be carried out using the same equipment for all classes, and ideally one which will be later used in the smart traps. Using the same microphones should align the training sample with the deployment data the traps will encounter on their missions. Given the large cost to obtain large enough sample, we suggest fitting the current traps with recording devices and construct the sample based on what animals are being trapped at present. Recording the sounds before the trap was closed should save labour otherwise spent on annotation. The sample would still carry some potential for noise, however as we saw even in small datasets GoogleConv shows considerable robustness to these. Most importantly, this way the collected sample of animal calls will correspond in composition to the fauna of the forests in which the traps are meant to be deployed in the future. Therefore, the model will consider true alternatives, rather than their proxies as was the case here.

Our prototype was not optimised with respect to its resources. First, the Raspberry Pi 3 gave the model a generous 1GB of memory. In larger scale, this could prove wasteful. Therefore, alternative, less parameter intensive, architectures should be experimented with. Also, model compression techniques such as the node pruning and SVD encoding should be tried. Second, energy consumption was neither optimised nor measured in this project. This, probably even more than the memory size is a key consideration since the traps are meant to be deployable for long periods of time in remote areas.

Consequently, we suggest automated fauna sounds dataset collection and subsequent, thorough, experimentation with various architectures focusing not only on the accuracy, but also on the memory requirement and energy the consumption.

References

Amos, W., Nichols, H. J., Churchyard, T., and de L. Brooke, M. Rat eradication comes within a whisker! a case study of a failed project from the south pacific. *Royal Society Open Science*, 3(4), 2016. doi: 10.1098/rsos.160110. URL http://rsos.royalsocietypublishing.org/content/3/4/160110.

Cooper, R. and Millener, P. The New Zealand biota: Historical background and new research. *Trends in Ecology & Evolution*, 8:429–433, 1993.

Gantz, V. M. et al. Highly efficient Cas9-mediated gene drive for population modification of the malaria vector mosquito Anopheles stephensi. In *Proc. Natl Acad. Sci. USA*, 2015.

Griffiths, R., Buchanan, F., Broome, K., Neilsen, J., Brown, D., and Weakley, M. Successful eradication of invasive vertebrates on Rangitoto and Motutapu Islands, New Zealand. *Biological Invasions*, 17:1355–1369, 2015.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet Classification with Deep Convolutional Neural Networks. In *NIPS*, 2012.

Lane, N. D., Georgiev, P., and Qendro, L. Deepear: Robust Smartphone Audio Sensing in Unconstrained Acoustic Environments using Deep Learning. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp 15)*, 2015.

Lindsey, T. and Morris, R. (eds.). *Collins Field Guide to New Zealand Wildlife*. HarperCollins (New Zealand) Limited, 2000. ISBN 978-1-86950-300-0.

McGlone, W. Challenges for Pest Management in New Zealand, 2014. online: accessed 04-Jan-2018.

New Zealand Department of Conservation, Te Papa Atawbai. Predator Free 2050. URL http://www.doc.govt.nz/predator-free-2050. online: accessed 05-Dec-2017.

New Zealand Parliamentary Commissioner for the Environment, Te Kaitiaki Taiao a Te Whare Paremata. Evaluating the use of 1080: Predators, Poisons & Silent Forests, 2011. URL www.pce.parliament.nz/assets/Uploads/PCE-1080.pdf. online: accessed 02-Jan-2018.

Osugi, M., Foster, T. M., Temple, W., and Poling, A. Behavior-Based Assessment of the Auditory Abilities of Brushtail Possums. *Journal of the Experimental Analysis of Behavior*, 96(1):123–138, 2011.

Owens, B. Behind New Zealands wild plan to purge all pests, 2017. online: accessed 04-Jan-2018.

Russell, J. C., Towns, D. R., Anderson, S. H., and Clout, M. N. Intercepting the first rat ashore. *Nature*, 437, 2005.

Sainath, T. N. and Parada, C. Convolutional neural networks for small-footprint keyword spotting. In *INTERSPEECH*, 2015.

Stowell, D., Wood, M., Stylianou, Y., and Glotin, H. Bird detection in audio: a survey and a challenge. *CoRR*, abs/1608.03417, 2016. URL http://arxiv.org/abs/1608.03417.

Treasury, New Zealand. Budget 2017, 2017. URL http://www.treasury.govt.nz/budget/2017/summarytables/estimates/02.htm. online: accessed 04-Jan-2018.

Williams, P. A. and Timmins, S. Economic impacts of weeds in New Zealand. *Biological Invasions*, 2002.