Power and Accuracy Trade-Offs for Machine Learning Methods Applied to Detection of Underwater Sound Sources

William Butler[1], Harrison Smith[2], Marios Impraimakis[1], Andrew Barnes[1], and Alan Hunter[1]

[1]University of Bath, Bath, BA2 7AY, United Kingdom
[2]Celtic Sea Power, Hayle, TR27 4DD, United Kingdom

Contact Author: William Butler, wmb34@bath.ac.uk

Keywords: *Machine Learning, CNN, Low-Power.*

1. INTRODUCTION

There are many autonomous marine drones currently deployed for a range of commercial, defense and marine conservation applications. Two categories commonly discussed are Uncrewed Surface Vehicles (USVs) and Uncrewed Underwater Vehicles (UUVs). These uncrewed vehicles often operate autonomously at sea for extended periods of time without refueling, yet due to their relatively small size they have limited power on board. For example, the Kongsberg SEAGLIDER may operate for up to 10 months, with a total power supply of only 17 MJ [1] (17 MJ $\approx$ 4722 Wh). In order to improve smart autonomous decision-making, as well as optimising the storage of data recorded by the onboard sensors, Artificial Intelligence (AI) can be used for rapid, real-time automated detection and classification of underwater noise-emitting objects. However, AI models often rely on large, deep neural networks that require extensive computational resources to run.

Much work has been done on edge computation, often utilising cloud resources [2]. For applications where cloud computing is not viable due to either a lack of internet connectivity at sea or security concerns associated with transmitting data, there has been less research. In addition, many of the models described above require pre-processing. In the case of audio classification, often spectrograms are used with Computer Vision models. These require the computation of Fourier transforms, which contribute to the overall energy cost of the system, not just the model itself.

However, while 2D CNN-based architectures such as ResNet have been widely successful [3], including for classifying spectrograms, CNNs with 1D-kernels can be used on time-series data such as the raw audio waveform. These models bypass the need for frequency decomposition via Fourier transform. In this work, two lightweight CNN models with similar architectures

Class	Count	Description
S1	741	Multiple Harmonic Tonals
S2	520	Non-tonal Bursts
S3	363	Harmonic Pulses
SP4	134	Dyadic Pulse Train
SP5	170	"Zipper" Pattern Pulse Train

Table 1: Seal call class densities and descriptions

are presented to compare the power and accuracy trade-offs between a 2D-kernel CNN model classifying spectrograms and a 1D-kernel CNN model classifying raw waveforms, as well as their respective pre-processing computational cost.

2. METHODOLOGY

The data used in this paper was provided by Celtic Sea Power (CSP). Multiple 2-hour recordings were taken with an AudioMoth [4] and HydroMoth [5] hydrophone with a sample rate of 96kHz. The audio was manually tagged by CSP employees and verified by a marine biologist. Seal calls were tagged with a start and end time and then split into 5 classes labeled S1, S2, S3, SP4, and SP5 according to a modified version of Nowak's seal vocalisation classes ([6], [7]). Table 1. shows the class densities in the dataset.

For each call tagged and labeled according to the five classes mentioned in Section 2., a 10-second slice of the 2 hour audio recordings is taken, randomly centering the call. Gaps between calls with a duration of 2 to 10 seconds are used to compute a background noise floor for each location by taking the median across each frequency. More information on the slicing and background noise floor generation, as well as the call types themselves can be found in the paper titled "Automated detection and classification of vocalisations from wild and captive seal populations" in this conference proceedings [7].

2.1. 1D CNN PRE-PROCESSING

Minimal pre-processing on the raw wav files is performed. 20 wav files are randomly sampled from each class, from the training set, and a mean and standard deviation of the amplitudes across those sampled calls is computed ($\mu \approx 0.00$, $\sigma \approx 0.07$). These values are then used to perform Z-score normalisation on the raw waveform. No other pre-processing is performed; this normalised time series is used as the input to the 1D CNN model. This input is a waveform sampled at 96kHz over a 10-second window, resulting in a single-channel input vector of shape (1,960000).

2.2. 2D CNN PRE-PROCESSING

To generate 2D inputs, a Short-Time Fourier Transform (STFT) is computed for each 10-second audio segment using a 2048-point window, a hop size of 1024 samples, a Hann window, and a sampling rate of 96kHz. The absolute value of the spectrogram, computed using the

Python Librosa package [8], is taken to obtain the magnitude of the frequencies. This returns a 1 channel, real-valued spectrogram of shape (1025, 936), after slicing the first and last time frame to account for any potential artifacts. A frequency-dependent background noise floor is estimated for each location by taking the median amplitude (in linear scale) from non-call segments as described above. This noise profile is then divided from the linear spectrogram for each input sample. After noise subtraction, the spectrogram is converted to the decibel (dB) scale with a reference of 1.0. From these dB spectrograms, a class-wise mean and standard deviation is computed across 20 randomly selected samples from each class ($\mu \approx -0.29, \sigma \approx 6.78$). These statistics are then used to perform Z-score normalisation on all spectrograms. The resulting normalised dB-scale spectrograms are used as input to the 2D CNN model.

2.3. MODEL ARCHITECTURES AND TRAINING

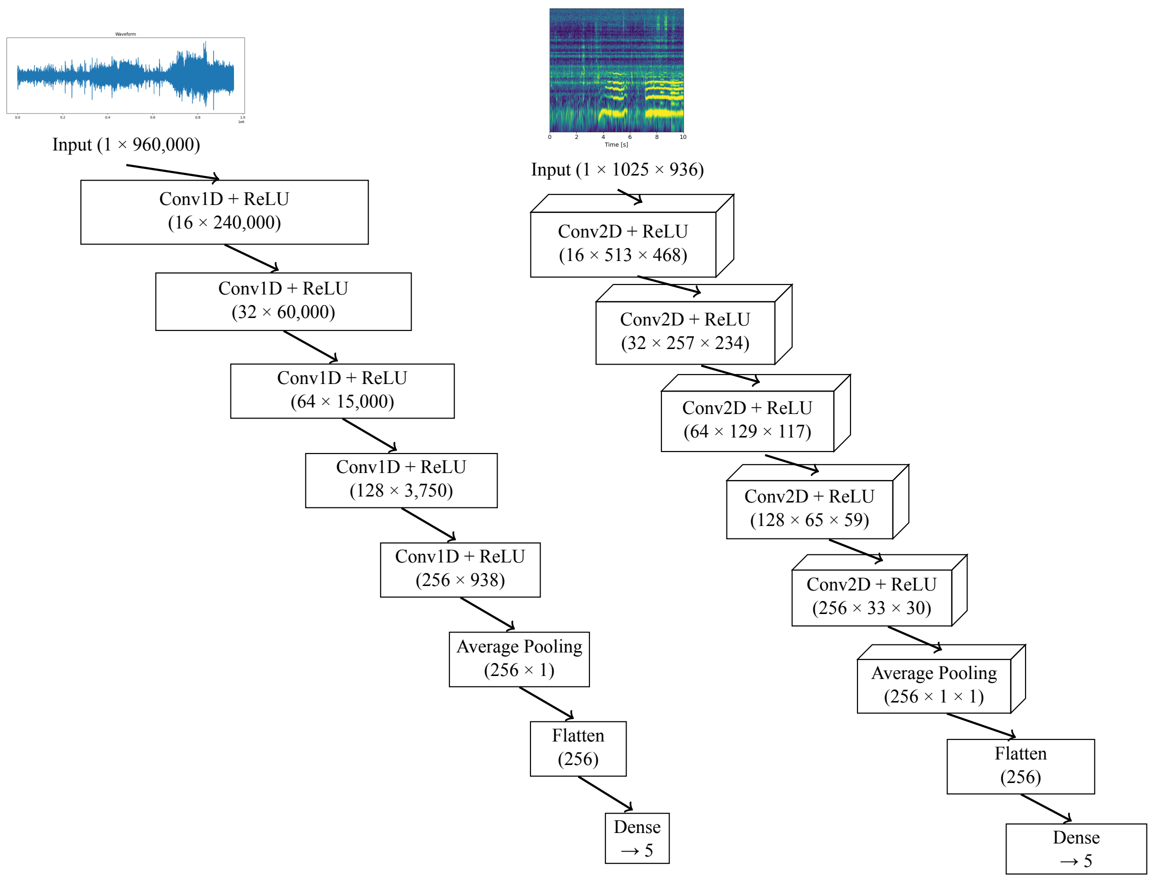

Waveform
Input (1 × 960,000)
Conv1D + ReLU (16 × 240,000)
Conv1D + ReLU (32 × 60,000)
Conv1D + ReLU (64 × 15,000)
Conv1D + ReLU (128 × 3,750)
Conv1D + ReLU (256 × 938)
Average Pooling (256 × 1)
Flatten (256)
Dense → 5
Time [s]
Input (1 × 1025 × 936)
Conv2D + ReLU (16 × 513 × 468)
Conv2D + ReLU (32 × 257 × 234)
Conv2D + ReLU (64 × 129 × 117)
Conv2D + ReLU (128 × 65 × 59)
Conv2D + ReLU (256 × 33 × 30)
Average Pooling (256 × 1 × 1)
Flatten (256)
Dense → 5

Figure 1: Comparison of 1D CNN for raw waveform input (left) and 2D CNN for spectrogram input (right). Dimensions indicate (channels × temporal length) for 1D and (channels × height × width) for 2D layers.

The raw waveform and spectrogram models share the same architecture, differing only in convolution type. The waveform model uses 1D convolutions with kernel size 9, stride 4 and 4 padding; the spectrogram model uses 2D convolutions with 3×3 kernels, stride 2 and 1 padding. This keeps both the parameter count and the downsampling rate consistent across models. Both models have five convolutional layers with ReLU activations at each layer and channels increasing from 16 to 256. After the convolutional stack, both models include an adaptive average pooling (Average Pooling) layer followed by a fully connected dense layer with 5 output nodes corresponding to the 5 classification categories. As PyTorch's CrossEntropyLoss integrates softmax internally, no explicit output activation was used. Fig 1. shows the architectures side by side with their channels and latent space dimensionality at each step.

Both models were trained under identical conditions to ensure a fair comparison. Cross-entropy loss was used with class weighting to address data imbalance. The dataset was split into training, validation, and test sets with ratios of 60:20:20. The Adam optimiser [9] with a learning rate of 0.0001 was used, and training was conducted over 100 epochs, with 10 iterations of training for each model. The best model was selected on the basis of the validation loss across epochs for each iteration of training.

2.4. METHODS FOR ESTIMATING/CALCULATING POWER CONSUMPTION

Floating Point Operations, or FLOPs was used to benchmark power consumption as it is agnostic to hardware specifications. To compute the FLOPs of the models themselves, two Python packages were used: ptflops [10] and fvcore from Facebook Research [11]. The results of these were compared and averaged to compute the estimated FLOPs of the model. The normalisation step consists of simply subtracting the mean and dividing by the standard deviation for each sample, giving a complexity of O(n) FLOPs.

The STFT computation is more challenging to benchmark as there are numerous implementations as well as hardware specific optimisations. Here, the number of FLOPs is estimated from the asymptotic computational cost of an FFT [12] over P samples within Q windows, i.e.

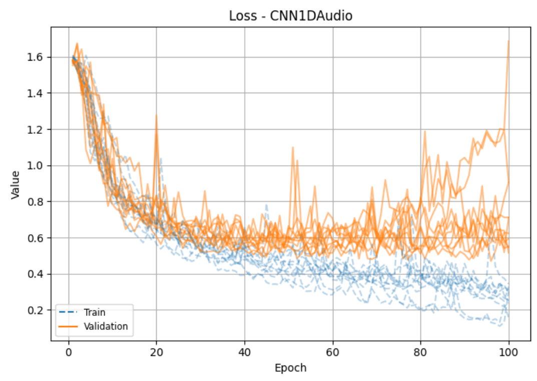
$$C_{\text{STFT}} \approx Q \times (2.5\, P \log_2 P) \tag{1}$$

is the estimated cost in FLOPs of the STFT.

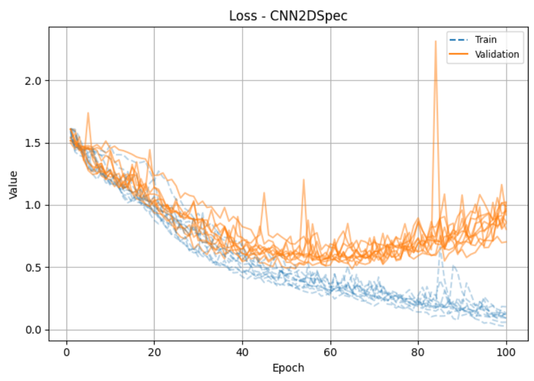
Finally, the background noise subtraction and conversion to dB scale need to be estimated. As background subtraction is just a division per sample; it is computed in O(n) FLOPs. The conversion to dB is not generally performed in FLOPs, depending on implementation, and may use methods such as lookup tables. However, it is assumed that this is generally O(n) complexity and so for this paper, only the STFT cost is used as all other costs are in the order of O(n) and thus are minimal compared to the model complexity and the Fourier transforms.

3. RESULTS

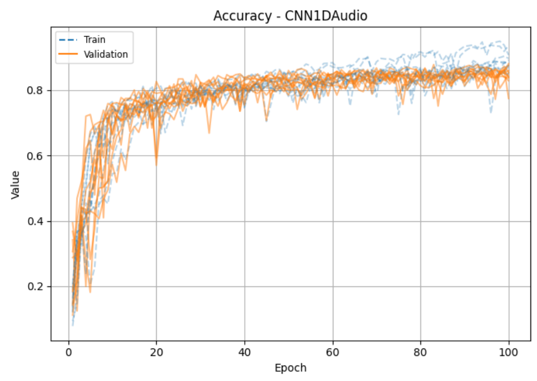
The total FLOPs for the STFT for $Q = 938$ windows and $P = 2048$ samples per window gives an estimated compute cost of just over 52.8 MFLOPs. This is an upper estimate, and hardware-specific implementations will very likely require fewer operations. Ptflops calculated 1163.43 MFLOPS and 1187.53 MFLOPs while fvcore calculated 1140.63 MFLOPS and 1164.85 for the 1D CNN and 2D CNN respectively. That gives an average of $\sim$1152 MFLOPs for the 1D model and $\sim$1176 for the 2D model.



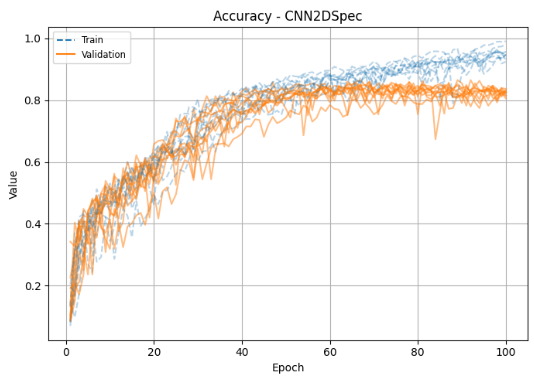
(a) 1D CNN training and validation loss plots over 10 iterations



(b) 2D CNN training and validation loss plots over 10 iterations



(c) 1D CNN training and validation accuracy plots over 10 iterations



(d) 2D CNN training and validation accuracy plots over 10 iterations

Figure 2: Comparison of 1D vs 2D CNN validation metrics.

Class	Precision	Recall	F1-score	Support
S1	**0.92**	0.77	0.84	149
S2	0.83	0.84	0.83	104
S3	0.65	0.78	0.71	72
SP4	0.83	0.87	0.84	27
SP5	0.82	**0.93**	**0.86**	34
Accuracy		*0.812*		386
Macro avg	0.812	0.837	0.816	386
Weighted avg	0.831	0.812	0.813	386

Table 2: Average test data classification report for 1D CNN trained on waveform input

Class	Precision	Recall	F1-score	Support
S1	**0.87**	0.83	**0.85**	149
S2	0.86	0.84	**0.85**	104
S3	0.68	0.70	0.69	72
SP4	0.78	**0.86**	0.82	27
SP5	0.80	0.84	0.82	34
Accuracy		*0.813*		386
Macro avg	0.799	0.816	0.804	386
Weighted avg	0.822	0.813	0.812	386

Table 3: Average test data classification report for 2D CNN trained on spectrograms

Model	**Test Accuracy**	**Model MFLOPs**	**Pre-Processing MFLOPs**	**Total MFLOPs**
1D-Kernel CNN	0.812	1152	Negligible	1152
2D-Kernel CNN	0.813	1176	<52.8	<1228

Table 4: Model accuracies and compute costs at inference, including pre-processing

4. DISCUSSION AND CONCLUSION

Despite the imbalanced dataset, the use of class weighting in the optimiser proved effective. With the exception of the S3 class, which both models are noticeably worse at classifying, performance was relatively consistent across all categories. The poorer performance of the S3 class is expected as this category may be more acoustically heterogeneous and comprised of multiple functionally distinct call types [7].

The 2D-kernel CNN trained on spectrogram inputs does not significantly outperform the 1D-kernel CNN trained on raw waveform inputs in terms of overall accuracy (0.813, 0.812 respectively) despite the increased pre-processing. As demonstrated in Fig. 2, which shows the performance of each model through training, both models exhibit initially relatively stable training dynamics without significant signs of overfitting. However, at around epoch 50-60 both models begin to spike in variance and begin to overfit.

Interestingly, despite similar architectures with nearly equal parameter counts and inference FLOPs, the 1D model converges faster, while the 2D model's validation performance improves more slowly, eventually achieving slightly better accuracy. However, given the class imbalance and challenges associated with S3, accuracy alone is insufficient to evaluate the models. Tables 2 and 3, which present class-level metrics, show that the 2D-kernel CNN slightly outperforms the 1D model on the two largest classes, S1 and S2, while the 1D model performs better on the smaller SP4 and SP5 classes. This trend may be due to the acoustic characteristics of the call types themselves, particularly if the SP4 and SP5 classes benefit from the phase information ommitted from the spectrograms in this paper. However, the slower training curve as well as the increased overfitting observed from in 2D CNN during later epochs may suggest that the architecture itself is less robust to imbalanced classification problems.

While the STFT adds a non-trivial computational cost of approximately 4–5% of total FLOPs at inference, this overhead is relatively minor. In scenarios where power consumption must be reduced such as in long-duration field deployments of UUVs or USVs, the loss of accuracy when choosing a 1D CNN is also minimal and may be justified. Furthermore, if primary target classes are sparsely represented, a waveform-based model may conserve energy while improving performance on critical but sparse acoustic signals since the 1D CNN appears to handle the minority classes (SP4 and SP5) more effectively. However, it remains unclear whether these performance differences stem from fundamental properties of the representation of the input signal and convolutional dimensionality (i.e. 1D vs. 2D inputs/kernels), if they are signal specific to SP4 and SP5, or if they are simply artifacts of variability due to the poorly performing and acoustically heterogeneous S3 class.

Further work is needed to determine whether the observed trends in minority class performance are due to signal-specific characteristics, class imbalance, incidental variability or other unknown factors. A clearer understanding of these factors could guide the development of more efficient, task-specific models that balance accuracy and resource constraints.

5. ACKNOWLEDGEMENTS

The authors would like to thank Seiche Water Technology Group and the ART-AI CDT for funding and supporting William Butler's PhD.

REFERENCES

[1] SEAGLIDER - kongsberg maritime - PDF catalogs | documentation | boating brochures. [Online]. Available: https://pdf.nauticexpo.com/pdf/kongsberg-maritime/seaglider/31233-105713.html

[2] D. Rosendo, A. Costan, P. Valduriez, and G. Antoniu, "Distributed intelligence on the edge-to-cloud continuum: A systematic literature review," *Journal of Parallel and Distributed Computing*, vol. 166, pp. 71–94, 2022.

[3] T. S. Prajwal and I. A K, "A comparative study of resnet-pretrained models for computernbsp;vision," in *Proceedings of the 2023 Fifteenth International Conference on Contemporary Computing*, ser. IC3-2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 419–425. [Online]. Available: https://doi.org/10.1145/3607947.3608042

[4] A. P. Hill, P. Prince, J. L. Snaddon, C. P. Doncaster, and A. Rogers, "Audiomoth: A low-cost acoustic device for monitoring biodiversity and the environment," *HardwareX*, vol. 6, p. e00073, 2019.

[5] T. A. Lamont, L. Chapuis, B. Williams, S. Dines, T. Gridley, G. Frainer, J. Fearey, P. B. Maulana, M. E. Prasetya, J. Jompa *et al.*, "Hydromoth: Testing a prototype low-cost acoustic recorder for aquatic environments," *Remote Sensing in Ecology and Conservation*, vol. 8, no. 3, pp. 362–378, 2022.

[6] L. J. Nowak, "Observations on mechanisms and phenomena underlying underwater and surface vocalisations of grey seals," *Bioacoustics*, vol. 30, no. 6, pp. 696–715, 2021.

[7] W. Butler, H. Smith, W. Lloyd, M. Impraimakis, A. Barnes, and A. Hunter, "Automated classification of vocalisations from wild and captive seal populations," June 2025, to appear in Proceedings of the Underwater Acoustics Conference and Exhibition (UACE 2025).

[8] B. McFee, C. Raffel, D. Liang, D. P. W. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th Python in Science Conference*, 2015, pp. 18–25.

[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization. arxiv [preprint](2014)," *arXiv preprint arXiv:1412.6980*, vol. 5, 2014.

[10] V. Sovrasov. (2018–2024) ptflops: A flops counting tool for neural networks in pytorch framework. Accessed: May 2025. [Online]. Available: https://github.com/sovrasov/flops-counter.pytorch

[11] Facebook AI Research, "fvcore: A library for efficient model analysis," https://github.com/facebookresearch/fvcore, 2025, accessed: May 2025. [Online]. Available: https://github.com/facebookresearch/fvcore

[12] M. Frigo and S. G. Johnson, "The design and implementation of fftw3," *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005.