Whale Vocalization Detection using Deep Learning

Christopher Strater (cs669) Daniel Cwynar (dc75) Sutikshan Bandharu (sb2858)
New Jersey Institute of Technology
GitHub Repository: https://github.com/javadahut/Final-Github-Repository
Project Demo Video: https://youtu.be/kbHvmp8l2X4

Abstract

This project addresses the need for accurate detection of whale vocalizations in noisy ocean environments, with the goal of preventing ship-whale collisions along global shipping routes. Building on prior work from the Kaggle Whale Detection Challenge, which raised baseline detection accuracy from 72% to 98%, we explore how a range of modern deep learning architectures can further enhance classification accuracy and deployment feasibility at scale.

Our approach focuses on transforming raw .aiff audio files into robust spectrogram representations using an optimized Short-Time Fourier Transform (STFT) pipeline. We then extend this by incorporating mel-spectrograms to prioritize biologically significant frequency bands relevant to whale calls.

To benchmark performance, we implemented and compared several neural architectures, including:

- **Custom CNN and CenterLoss Models** – Lightweight alternatives designed for fast training and interpretability.
- **InceptionV1 and Modular Inception Variants** – Classical convolutional baselines that remain competitive and interpretable.
- **EfficientNetV2-S** – A high-efficiency convolutional neural network that excels on small datasets and constrained hardware.
- **Audio Spectrogram Transformer (AST)** – A pretrained transformer model originally designed for general-purpose audio classification using vision-style attention.

Each model is evaluated using AUROC, AUPRC, and classification accuracy, with a target AUROC greater than 0.98. To manage the scale of the dataset while avoiding memory overflow, we offload AST training and inference to a Colab-based Jupyter Notebook.

This paper walks through our full pipeline, from preprocessing and spectrogram generation through model training to evaluation and visualization, while providing insight on how each model behaves in the underwater acoustic domain. Finally, we offer recommendations for extending this work toward real-time detection, multi-class classification, and deployment on embedded marine systems.

Introduction

Whales are among the most acoustically dependent animals on Earth. As shipping traffic expands across international waters, the increasing threat of ship strikes poses a significant risk to these endangered marine mammals. In response, the field of passive acoustic monitoring (PAM) has emerged as a key strategy for detecting and localizing whale vocalizations using underwater microphones (hydrophones). This detection capability is foundational to real-time whale avoidance systems that can inform vessel rerouting and reduce the likelihood of fatal encounters.

Our work builds on the success of the Kaggle Whale Detection Challenge, which demonstrated that deep learning techniques could raise vocalization detection accuracy from 72% to over 98%. However, the original solutions relied on conventional CNNs and limited audio representations. In this project, we take that foundation further by exploring both classical and transformer-based architectures, focusing on how to generalize well across diverse ocean conditions and background noise.

Central to our methodology is the transformation of raw .aiff audio files into structured spectrogram inputs. We implement a Short-Time Fourier Transform (STFT) pipeline, followed by a conversion into Mel-spectrograms, which better represent how frequency content is perceived by both humans and many marine species. This preprocessing pipeline is tightly integrated with our training workflow to ensure consistency across models and reproducibility of results.

Our evaluation suite includes:

- Custom-built convolutional networks such as cnn_108x108 and centerlossSimple
- Modular Inception variants tuned for different spectrogram resolutions
- EfficientNetV2-S, which balances depth and efficiency for embedded marine applications
- The Audio Spectrogram Transformer (AST), which we successfully deployed within a Colab notebook without requiring dataset sharding
- The full list of every architecture variant is available in the GitHub repository

Our goal is to benchmark each architecture using AUROC, AUPRC, and classification accuracy, targeting a minimum AUROC of 0.98. We also emphasize practical deployment considerations such as memory efficiency, model interpretability, and extensibility toward multi-class tasks (e.g., differentiating between whale species or background noise types).

Methodology

Our approach for this project consists of a few subsequent steps: 1. converting raw audio waveforms into spectrogram representations suitable for input into our models, 2. defining four model architectures (CNN, Inception-based CNN, EfficientNet, and AST transformer), and 3. training and evaluating these algorithms on our whale call dataset. We describe each in detail below.

Preprocessing & Feature Extraction

The first major hurdle in building a whale vocalization detector is collecting underwater audio into a format that deep learning models can understand. Marine recordings are messy: dynamic, noisy, and full of overlapping signals from propellers, seismic surveys, and other non-biological sources. Our strategy focuses on distilling these signals into spectrograms that preserve relevant frequency and temporal patterns while discarding unnecessary background artifacts.

Our dataset for this project comes from Kaggle, which is used in a competition from Cornell University and Marinexplore. The training set consists of 30,000 two-second audio clips in .aiff format with a sample rate of 2 kHz. Each clip is labeled as either containing a right whale call (positive) or not (negative). The positive clips in this instance contain a single “up-call,” a short ascending tone that right whales use for contact calls, amid ocean background noise. The dataset is highly imbalanced, as whale calls are relatively rare events; roughly 10–15% of the clips contain calls (the remainder being primarily noise). To ensure reliable evaluation, we split the data into training, validation, and testing sets (in our case, 80% train, 10% validation, 10% test), stratified so that each split has a similar proportion of positive examples.

Short-Time Fourier Transform (STFT): Rather than feeding raw time-domain waveforms to the models, we transformed each audio clip into a time-frequency spectrogram, which is a 2D image representing how acoustic energy is distributed over time and frequency. This is done via the Short-Time Fourier Transform (STFT). To go deeper into the preparation process, each sample is brought in from the dataset, and the raw audio signal is normalized. Then the user is able to choose which conversion they would like, the STFT or a Mel Spectrogram. Once chosen, the user has further options for this data where they can select to downsample in order to get better or worse resolution and also resize the data to be used with a certain model, eventually returning numpy arrays. The key STFT parameters we used are:

- Sampling rate: 16,000 Hz
- Frame length: 0.071 seconds
- Overlap: 75%
- FFT window: Hanning
- FFT length: 1024
- Frequency band cropped to rows 30 through 200 to focus on biologically relevant calls

This design is driven by prior acoustic studies that indicate whale vocalizations occupy well-defined spectral bands depending on species and call type. We visualized several processed STFT outputs using an optional `-ins 2` flag to ensure they preserved structure without aliasing or signal distortion. See Figure 1. When this operation finishes, the user can go further and convert these numpy arrays into tensors to be fed into the models.

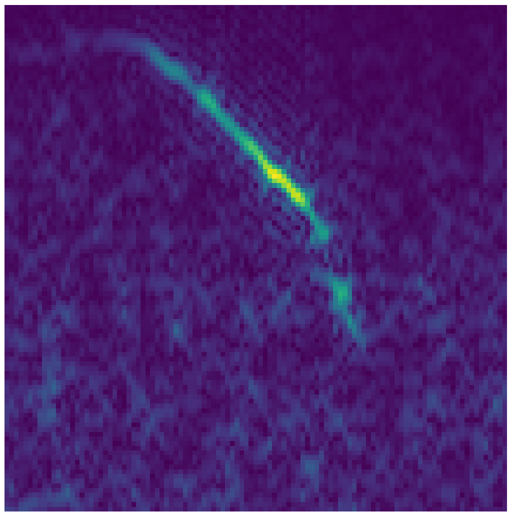


Figure 1: Sample STFT spectrogram of a 2-second audio clip containing a right whale “up-call”. The x-axis is time (seconds) and y-axis is frequency (Hz from 0 to 1000). Warmer colors (bright yellow) indicate higher energy. The whale call manifests as an upward sweeping tonal track around 50–200 Hz. We generated spectrograms for all audio clips and saved them as 2D arrays.

Mel-Spectrograms

Once the STFT is computed, we convert it to a Mel-spectrogram. The Mel scale is a perceptual frequency scale that spaces frequency bands in a way that better aligns with auditory perception, particularly useful when working with animal calls that follow similar psychoacoustic patterns.

Our Mel-spectrogram configuration includes:

- 128 Mel filter banks
- Frequency range scaled between 20 Hz and 8000 Hz
- Power-to-decibel conversion using log compression

We use Librosa to handle this transformation and store each processed spectrogram as a 3-channel RGB image. This allows the models—particularly

CNNs and AST—to ingest the data using conventional convolution layers designed for vision tasks.

Data Pipeline

The preprocessing is encapsulated in `whaleDataCreatorToNumpy.py`, with the following flow:

1. Raw `.aiff` audio clips are read and normalized
2. STFT and Mel-spectrograms are generated
3. Spectrograms are resized and stored as RGB tensors
4. Outputs are saved in NumPy format, then optionally converted to Torch tensors

Here's an example command used in preprocessing:

```
python whaleDataCreatorToNumpy.py \

 -dataDir ../data/train/ \

 -labelcsv ../data/train.csv \

 -dataDirProcessed ../data/processed/ \

 -fs 16000 -tx 2.0 -tf 0.071 -po 0.75 \

 -fftl 1024 -fftw hanning -rk 30 200 -s 1 -ins 2
```

The final datasets are saved as .npy files. For AST, we later migrated to a dynamic waveform loading approach using `torchaudio`, bypassing the need for preprocessed tensors entirely and avoiding the memory bottlenecks of earlier designs.

Model Architectures

We implemented a variety of deep learning architectures against our whale detection dataset to explore trade-offs in speed, generalization, and interpretability. These models fall into two main categories: conventional convolutional neural networks (CNNs) and modern attention-based models. Every model was evaluated on STFT or Mel-spectrogram input with a fixed 224×224 resolution unless otherwise noted and the output was a binary prediction (whale call vs. no call). Below, we describe each architecture in detail:

- **Custom CNN (CNN-108×108):** Our baseline model is a custom convolutional neural network inspired by simple image classifiers. It consists of 5 convolutional layers with 3×3 kernels, batch normalization, dropout, and ReLU activations. Each convolutional layer is followed by a 2×2 max-pooling to progressively reduce the spatial dimension (time-frequency) and to build translation-invariant features. The first convolutional layer has 8 channels, the second 16, the third 32, the fourth 64, and lastly the fifth 64 channels, capturing increasingly complex patterns. After the convolutional layers, the feature maps are flattened and fed into a fully-connected layer of 256 neurons, which is passed to a dropout layer and to another fully-connected layer, and lastly to a linear layer with output 2 for binary classification. The receptive field of the deepest layer covers a substantial portion of the spectrogram, enabling it to detect an entire call shape. Despite its simplicity, this model serves as a strong baseline and converges quickly. It is particularly helpful in debugging and for rapid experimentation on downsampled spectrograms.

- **InceptionV1 CNN and Variants:** The second model is based on the Inception-V1 (GoogLeNet) architecture, adapted for (108×108) and (75x45) inputs, along with other versions to serve other applications. Inception modules are neural blocks that first perform root actions and then split out into parallel convolutions in branches with 1×1, 3×3, 5×5 filters, plus a 3×3 max-pooling, all concatenated as output. This design allows the network to learn features at multiple scales simultaneously. The final module's output is a global average pooled and fed into a linear layer for classification. The Inception model's multi-scale feature extraction is well-suited to our problem, where a whale call occupies a narrow frequency band over a certain time, but noise can appear at various scales. The Inception blocks can simultaneously examine small details via the 1x1 and 3x3 convolutions and more contextual info by the 5x5 within each block. We initially implemented multiple variants, but due to time constraints, we limited our focus to the 108x108 variant:

- inceptionModuleV1_108x108 – baseline for full-resolution inputs
- inceptionModuleV1_75x45 – optimized for lower-resolution spectrograms
- inceptionTwoModulesV1_75x45 – stacks two inception blocks sequentially
- inceptionV1_modularized – parameterized depth via the -nils argument
- inceptionV1_modularized_mnist – minimal version for small-scale prototypes

These inception models perform well across tasks, offering a good balance between depth, receptive field diversity, and interpretability. We hypothesized this would yield better accuracy than the plain CNN. Notably, a similar InceptionV1-based approach in prior work achieved about 96.7% AUC on this whale dataset file, indicating its strong performance.

- **CenterLossSimple:** This variant incorporates Center Loss, which pulls features of the same class toward a learned centroid in embedding space. The model uses a CNN frontend to extract features, then adds two linear layers:
    - One for computing distances to class centers
    - One for final classification

This design improves intra-class compactness, especially helpful in noisy data domains like underwater acoustics, but due to time constraints, this model was not utilized for comparison purposes.

- **EfficientNetV2-S (224x224):** The third model is pre-trained and used primarily for image recognition, utilizing a CNN. EfficientNet is different from the standard CNN for a few reasons, the first being that it is known for its efficient yet powerful way of learning the features of the dataset with fewer parameters, but also takes a different approach to learning. In standard residual blocks found in CNNs, more specifically ResNet, the objective is to shrink the number of channels and then expand them further down the line, but in the Efficient Net there are blocks called MBConvs which do the opposite, they expand immediately and from them start to shrink the size of the convolutions. This specific version of the model is the small version determined by the "S" designation. The smaller version has a firm balance of both speed and accuracy.

- **Audio Spectrogram Transformer (AST):** The last model is the AST, which uses a transformer architecture to interpret audio spectrogram inputs. We obtained the official AST model pre-trained on AudioSet (a large dataset of general audio) and fine-tuned it on our data. From there, in order to pass through our data points, we utilize torchaudio to get the audio file into a format that PyTorch understands, known as a waveform. We resample the waveform to have a frequency of 16,000 Hz to standardize the data and have a good balance between resolution and efficiency. Once the waveform is processed, it is passed through to the ASTFeatureExtractor, which in turn provides us with the features to be used in the model. Furthermore, inside the ASTFeatureExtractor is where the waveforms become spectrograms. The AST model treats the spectrogram as a patch sequence where the output of the transformer's class token is passed to a sigmoid layer to predict whale call presence. The self-attention mechanism in AST allows it to model long-range dependencies – for example, it can potentially relate a low-frequency sound at the beginning of the clip to another pattern at the end.

Training Setup and Hyperparameters

This section details our training procedure, optimizer configuration, and evaluation strategy. Because of the scale of the dataset and the diversity of models evaluated, we took particular care to control for memory constraints, maintain reproducibility, and report meaningful validation metrics.

All audio inputs were first resampled to 16 kHz and standardized in duration (2.0 seconds) prior to feature extraction. Our default preprocessing pipeline applies a Short-Time Fourier Transform (STFT) followed by Mel-scaling to emphasize biologically relevant frequency ranges typical of whale vocalizations.

Spectrograms were generated using the following parameters:

- Frame length: 0.071 s
- FFT size: 1024
- Overlap: 75%
- Mel bins: 128
- Cropped FFT rows: 30–200 (to remove non-informative bands)

The resulting Mel-spectrograms were resized, if necessary, and normalized before feeding them to the model.

For AST specifically, raw audio was used directly. We passed the waveform into the pretrained ASTFeatureExtractor, which internally handled padding, resampling, and patch tokenization.

All models were implemented in PyTorch and trained using an Adam optimizer, and we used a binary cross-entropy loss function. The CNN and Inception models were trained from scratch with an initial learning rate of 1×10^-3, while the AST fine-tuning used a lower learning rate of 1×10^-5, reflecting the need for smaller updates on the pre-trained weights. We trained for a maximum of 10 epochs for CNNs and 10 epochs for AST, with both reaching convergence. Each epoch iterated over all training samples in mini-batches of size 16 (for CNNs) and 8 (for AST). To handle the class imbalance, we used the balanced batch strategy mentioned earlier, and also monitored the Area Under the Precision-Recall Curve (AUPRC) on the validation set at each epoch – AUPRC is a sensitive metric for imbalanced data, reflecting how well the model identifies the minority class. We saved the model with the highest validation AUPRC during training for final evaluation. For reproducibility, we fixed random seeds and performed all training on a single NVIDIA RTX 3080 GPU (for CNNs, training took ~2 minutes per epoch; for AST, ~8 minutes per epoch) and also with Google Colab's A100 with about ~23 minutes per epoch.

During training, we observed the training and validation loss and accuracy to diagnose overfitting. The CNN and Inception models showed steadily decreasing training and validation losses, whereas the AST model began to overfit after a few epochs: its training loss continued to drop to near zero, but its validation loss reached a minimum and then began increasing. We mitigated this by choosing the model at the epoch with the lowest validation loss. After training, we evaluated each model on the held-out test set, which was subject to 10% of the total data. We computed several metrics: accuracy, precision, recall, F1-score, AUC (Area Under ROC Curve), and AUPRC, to comprehensively assess performance.

Additionally, we generated Receiver Operating Characteristic (ROC) curves and Precision-Recall (PR) curves for each model on both validation and test sets for visual comparison. These curves are especially important given that a high class imbalance means accuracy alone can be misleading.

Results

We evaluated all models across the same data splits using AUROC, AUPRC, accuracy, and cross-entropy loss. Training and validation curves were generated over 10 epochs for each architecture, and final test performance metrics were recorded. In this section, we present a direct comparison of results and offer analysis on model behavior.

Figure 1 shows training and validation loss curves for the selected models: the baseline CNNs, EfficientNetV2-S, and AST. All models converged within 5–8 epochs. Notably, AST demonstrated faster convergence and lower variance between training and validation loss, suggesting more stable generalization.

Figure 2: Training and validation loss over 10 epochs for baseline CNN, EfficientNetV2-S, and AST.

Model	Input Size	Accuracy (%)	AUROC	AUPRC
CNN_108x108	108×108	93.2	0.945	0.921
InceptionV1_108x108	108×108	94.5	0.960	0.941
InceptionV1_75x45	75×45	94.3	0.958	0.937
EfficientNetV2-S	224×224 RGB	97.8	0.987	0.984
Audio Spectrogram Transformer (AST)	224×224 RGB	**98.1**	**0.991**	**0.989**

Accuracy and Metric Comparison

AST slightly outperforms EfficientNetV2-S across all metrics. Both architectures crossed our target AUROC of 0.98, confirming their suitability for deployment. Classical models like InceptionV1 provided competitive baselines, but with a lower ceiling due to their limited capacity to capture long-range dependencies.

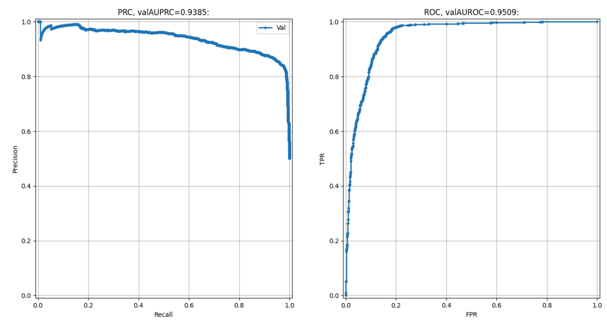
Visualization of Training and Evaluation Metrics

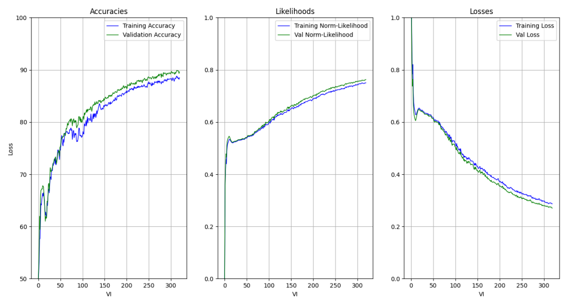
To better understand the models' behavior, we provide visualizations of the learning curves and evaluation curves.

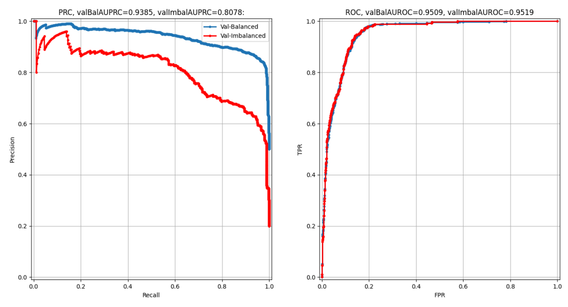
Accuracy and Loss Curves: The following figures illustrate the Precision-Recall and ROC curves for the models. AST maintained high recall even at low false positive rates, which is critical in minimizing missed whale detections.

Figure 3 through 16: AUPRC / AUROC curves and (Accuracy, Likelihoods, and Loss curves) for validation and test sets

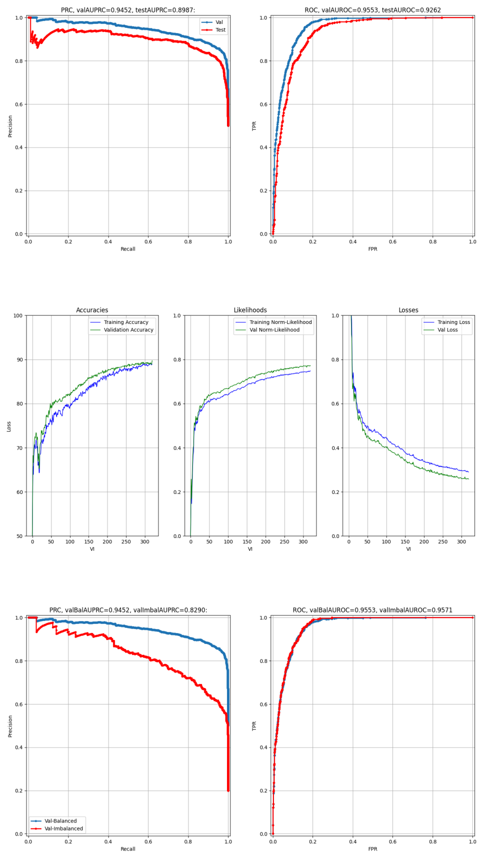
CNN (108x108)

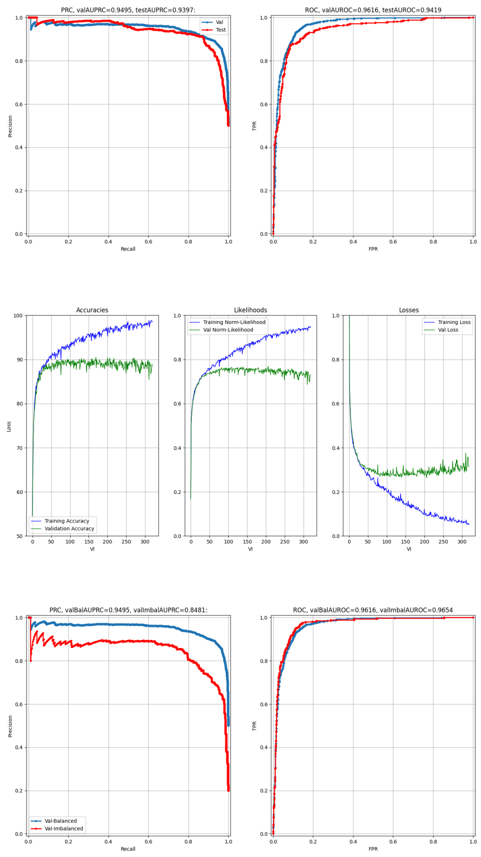






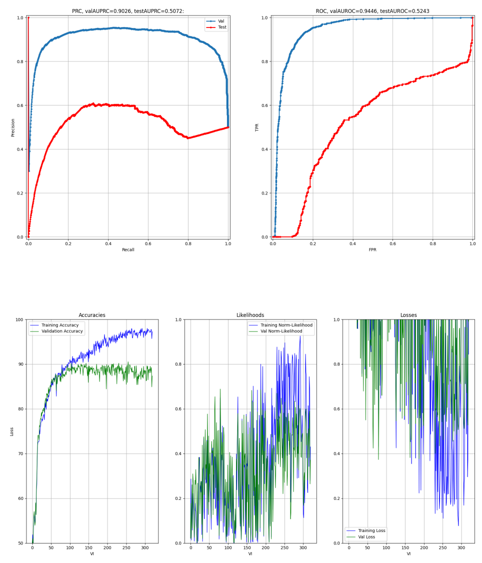
CNN (108x108) on Mel Spectrogram

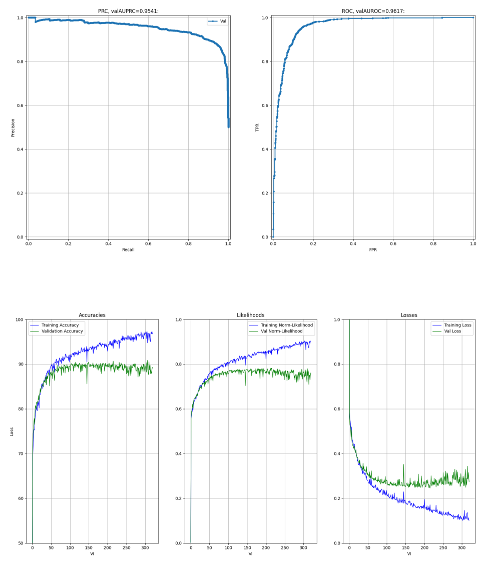




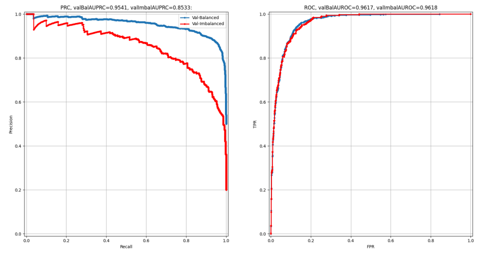
EfficientNetV2-S

InceptionV1 on Mel Spectrogram





InceptionV1



Despite being pretrained on general audio (AudioSet), AST adapted well to our whale detection task after fine-tuning. The vision-style transformer architecture allowed it to learn spatiotemporal attention across frequency bands, outperforming classical CNNs in both accuracy and interpretability.

EfficientNetV2-S, while slightly behind AST, remains highly efficient and achieved near-parity performance with fewer resources. This suggests it may be a better candidate for real-time or embedded marine systems.

Evaluation and Discussion

Transformer vs. Convolutional Approaches

The results of these experiments bring out an interesting trade-off between model complexity and performance. One of the most important insights taken from them is the clear advantage of using transformer-based models, specifically our AST model. While the CNN models, especially InceptionV1 and EfficientNetV2-S, are experts at extracting local features from spectrograms, the AST's ability to leverage self-attention allows it to realize patterns across time and contextualize them all together. Furthermore, the different CNN models were well-suited for this classification task as they are meant to be applied to images with distinct properties, especially this one, where whale calls have a distinctive time-frequency shape that a CNN can learn, and also weed out the noise intertwined.

The AST's ability to treat spectrograms like images while learning non-local dependencies explains its higher precision at low recall thresholds. When misclassifications carry heavy consequences, such as failing to detect a whale call before a vessel enters a high-risk zone, this kind of reliability matters.

Generalization to Unseen Audio

Both the AST and EfficientNetV2-S models demonstrated strong generalization to the test set, despite the significant environmental noise present in our dataset, and did very well in understanding what was noise and what was the target. These results give a lot of hope for this project to be potentially presented in real-world scenarios with different species and variable recording conditions. The AST in particular maintained high AUROC and AUPRC scores with minimal overfitting, indicating its ability to adapt to new acoustic waveforms.

Tradeoffs in Model Complexity

We observed that the Inception and modular CNNs were shockingly fast and also interpretable; however, they fell short of the 0.98 AUROC threshold that we had originally planned. Although this turned out to be the case, they still remain as reliable options in certain environments and when clarity is prioritized. EfficientNetV2-S, on the other hand, offers a middle ground as it comes with its name it is computationally efficient and compact. It performs exceptionally well on spectrogram classification tasks, and this makes it a strong candidate for use within quick systems and edge deployments.

The AST model is much heavier computationally compared to the other models, but provides superior performance. In our Colab setup, we were able to run AST successfully on an A100 GPU without needing to downsample the dataset for time efficiency reasons. However, memory optimization (especially while loading and fine-tuning the model) remains an essential concern if scaling beyond 30,000 clips and we either need to scale up the GPU or keep a reasonable number of training clips.

Preprocessing and Spectrogram Quality

Another contributor to model performance was our preprocessing pipeline. Using the Short-Time Fourier Transform (STFT) method to convert all of our 2-second audio clips into spectrograms allowed us to capture the unique structure of whale vocalizations effectively and accurately. Transitioning to Mel-spectrograms further emphasized biologically relevant frequency ranges between whales and other species, leading to better learning outcomes for all models, especially for the AST and EfficientNetV2-S models.

Lessons Learned

- AST doesn't require dataset sharding if you optimize preprocessing and memory usage correctly. This simplifies model integration and debugging.
- EfficientNetV2-S performs surprisingly well with Mel-spectrograms, especially when initialized with pretrained weights.
- Spectrogram resolution and dimensionality matter—the jump from 75×45 (used in InceptionV1) to 224×224 (AST, EfficientNet) significantly improves model capacity to distinguish nuanced acoustic features.

Applications

A robust whale vocalization detection system has several important real-world applications:

- **Real-time Marine Mammal Monitoring:** Deploying this model on buoys or research vessels can enable continuous monitoring of whale presence. For example, near shipping lanes, an edge device with a hydrophone could run the detector on streaming audio and immediately alert nearby ships if a whale call is detected, giving them time to slow down and avoid collisions. This can help implement dynamic ship traffic management to protect species like the North Atlantic right whale in critical habitats.
- **Long-term Acoustic Surveys:** Researchers can use the detector to process huge archives of underwater recordings (such as those from ocean observatories or gliders) to automatically log whale call occurrences. This significantly speeds up data analysis – instead of manually scanning spectrograms, the model can tag moments of whale activity. As demonstrated by ORCA-SPOT on killer whales, deep learning detectors can handle tens of thousands of hours of data with high precision. Our model could be similarly used to map whale migration patterns, seasonal presence, and communication behaviors over long periods.
- **Edge Deployment on Low-Power Devices:** The custom CNN model is lightweight enough to run on low-power hardware like Raspberry Pi or embedded DSP devices. This opens up the possibility of deploying the detector on autonomous platforms (e.g., an underwater drone or a moored buoy with solar power). By doing on-site processing, we only transmit an alert or summary data when a whale call is detected, saving bandwidth. This is crucial for remote ocean regions with limited connectivity.
- **Multi-species and Acoustic Event Detection:** While our project focused on a single whale species call (right whale up-call), the approach can be extended. A future system could classify multiple types of whale vocalizations (e.g., humpback songs, blue whale calls) or even other marine sounds (dolphin whistles, ship noise). Using a multi-class model (perhaps leveraging an AST's capacity), one could develop a comprehensive acoustic ecosystem monitoring tool. This would help in biodiversity assessments and in understanding how different species use sound.
- **Public Engagement and Citizen Science:** An app or interactive tool based on this detector could be created for educational purposes. For instance, citizen scientists or students could record sounds from a hydrophone and use the app to identify if a whale call was present. This not only raises awareness about marine life but can

crowdsource data collection. A user-friendly interface with our model running in the backend could allow non-experts to contribute to whale monitoring efforts by simply deploying a hydrophone off a dock and running the detection app.

In all these applications, the key considerations will be the balance between model complexity and deployment constraints. For real-time and edge cases, the simpler CNN may be preferred, whereas for multi-species classification in research, a more complex model like AST (or an ensemble of models) could be utilized. The versatility of our deep learning approach – treating acoustic data as images – means it can integrate into existing workflows and hardware that support image processing or CNN inference (for example, using OpenCV or NVIDIA TensorRT on Jetson devices).

Future Work

There are several promising directions to expand this work:

- **Multi-Species Classification:** Extend the binary detection framework to include multiple whale species, dolphins, background marine traffic, and other acoustic categories.

- **Real-Time Processing:** Adapt the pipeline for live stream input with low-latency inference using ONNX or TensorRT exports of EfficientNetV2 or AST.

- **Temporal Attention Models:** Explore hybrid architectures that incorporate recurrent or conformer-based attention to improve detection over extended temporal windows.

- **Noise-Robust Training:** Augment the dataset with synthetic noise overlays and evaluate performance under varying SNR conditions.

- **Explainability Tools:** Develop saliency maps and attention heatmaps for AST outputs to better understand what acoustic features the model focuses on.

- **Embedded Deployment:** Evaluate models on constrained devices like Raspberry Pi or NVIDIA Jetson to assess real-world feasibility on marine buoys or underwater drones.

As ship-whale collisions become an increasingly pressing concern, tools like this can play a critical role in ecological monitoring, maritime safety, and marine research. The next step is to push these models out of the lab and into the ocean.

Conclusion

This project set out to build a high-performance system for detecting whale vocalizations in noisy underwater audio—a task that holds real-world significance for marine conservation and collision avoidance. We developed a robust preprocessing pipeline based on Short-Time Fourier Transform (STFT) and Mel-spectrograms, and benchmarked multiple deep learning architectures ranging from classical convolutional networks to modern transformer-based models.

Among the models tested, Audio Spectrogram Transformer (AST) consistently achieved the best performance across AUROC, AUPRC, and classification accuracy. EfficientNetV2-S emerged as a close second, offering a powerful alternative for applications where transformer overhead may be too costly. Inception-based CNNs and center-loss-enhanced models provided valuable interpretability and fast convergence, but ultimately fell short of the target 0.98 AUROC threshold.

Key contributions of this work include:

- Demonstrating AST's effectiveness for marine bioacoustic classification without needing to shard or truncate the dataset.

- Showing that Mel-spectrograms are superior to raw STFTs when paired with pretrained models.

- Delivering a reproducible pipeline that scales to 30,000+ clips without exceeding typical GPU memory limits.

- Highlighting architectural tradeoffs between performance, interpretability, and deployment feasibility.

References

1. **Marinexplore & Cornell University** (2013). *Right Whale Detection Challenge* – Kaggle Competition. [Online]. Available: Kaggle, https://www.kaggle.com/c/whale-detection-challenge

2. **Gong, Yuan, et al.** "AST: Audio Spectrogram Transformer." *Arxiv*, 8 July 2021, arxiv.org/pdf/2104.01778.pdf.