Optimising the Engine Model for Overlapping Audio

Jessica Stinson
224576666

Introduction

Founded in 2022, Project Echo is dedicated to developing a sound classification model to facilitate the monitoring of endangered species in the Otways, Victoria. The current model can classify 118 species within the area, with ongoing efforts to expand the database to include all regional species. In the long term, the project aims to broaden the model's capabilities to recognise a diverse range of animal vocalisations and environmental sounds. Project Echo provides a non-intrusive approach to tracking both endangered species and their predators to support the work of animal conservationists. While the model performs with high accuracy under optimal conditions, its effectiveness declines significantly when processing audio with overlapping sounds. In addition, all previous versions of the Echo model have used multi-class classification, which limits detection to a single species per audio file and prevents recognition of multiple simultaneous vocalisations. This analysis aims to evaluate the current model pipeline and explore potential improvements to enhance model performance when evaluating overlapping audio files.

Creating a Synthetic Dataset

The current Project Echo database consists of short audio clips each containing a single animal vocalisation. When dealing with real-world audio, there are often overlapping calls and environmental noises which interfere with the performance of the classification model. To address this issue, a simple script was developed to generate synthetic audio files by combining multiple vocalisations. The dataset used to train the updated version of the engine model was composed of approximately two-thirds synthetic audio files (containing two to three overlapping vocalisations) and one-third original single-species audio. Due to memory constraints, the dataset was limited to six bird species from the Otways region, though the script can easily be scaled to include the full Project Echo database.

Labelling the Dataset

When testing the existing training pipeline, classification of single-species audio files yields strong results with an accuracy of 98.68%. While this achievement provides a promising outlook for the future success of Project Echo, real-world audio collected from the Otways National Park frequently contains background noises and overlapping vocalisations. Tests conducted on the model using real-world audio files have found that the introduction of background noise and additional species hinders the performance of the Echo model, with the model frequently failing to correctly identify any species in the audio clips. The multi-class classification method utilised for the current pipeline also forces the model to make a prediction, even on audio clips that lack any animal vocalisations. To combat these challenges a range of encoding techniques were tested. Initial testing utilised a multi-hot encoding technique where more than one species could be present in the labels. Unfortunately, this allowed the model to become biased toward predicting the absence of species, and it avoided making any meaningful predictions.

The decision was made to move away from multi-hot encoding, and the updated model pipeline has now been optimised to make predictions from class embeddings. Each species is assigned an embedding with a length of 64 which does not increase with the addition of more species names. This change encourages the model to make positive predictions and allows for continued expansion of the dataset without uncontrolled expansion of label lengths. The code for creating class embeddings can be seen in figure 1 below.

```
embedding_dim = SC['EMBEDDING_SIZE']

def create_class_embeddings(class_names, embedding_dim=embedding_dim):
    """
    Creates a dictionary mapping each class name to a unique, randomly initialized embedding.
    """
    embeddings = {}
    random.seed(42)
    for class_name in class_names:
        embeddings[class_name] = np.random.rand(embedding_dim).astype(np.float32)
    return embeddings
```

Figure 1. New function to create class embeddings

For files containing more than one vocalisation, multi-label embeddings are created by averaging the individual species embeddings (see figure 2, below). While this method has yielded the best results so far, predictions were still primarily focused on a single dominant species. Future work should continue to explore methods for combining class embeddings to better predict less prominent species.

```
# Create multi-label embeddings by averaging individual embeddings
multi_embeddings = []
for file_labels in labels:
    file_embedding = np.zeros(embedding_dim, dtype=np.float32)
    for i, is_present in enumerate(file_labels):
        if is_present:
            present_embeddings = [class_embeddings[class_names[i]] for i,
                                  present in enumerate(file_labels) if present]
            file_embedding = np.mean(present_embeddings, axis=0)

    multi_embeddings.append(file_embedding)
multi_embeddings = np.array(multi_embeddings)
```

Figure 2. Creating embeddings for multi-label audio files

Building and Training the Model

With the changes implemented to the model pipeline, it was noted that the training process performed differently than previous versions of the engine model. Previous models included both batch normalisation and dropout layers during training, however, testing of the new model revealed that its performance significantly improved with the removal of all batch normalisation layers. The model build was simplified by removing these layers and the dropout layer was retained to avoid overfitting (Figure 3).

```
def build_model(trainable, embedding_dim=embedding_dim):
    model = tf.keras.Sequential(
        [
            # Input layer with specified image dimensions
            tf.keras.layers.InputLayer(input_shape=(SC['MODEL_INPUT_IMAGE_HEIGHT'],
                                                    SC['MODEL_INPUT_IMAGE_WIDTH'],
                                                    SC['MODEL_INPUT_IMAGE_CHANNELS'])),

            # Use the EfficientNetV2 model as a feature generator (needs 260x260x3 images)
            hub.KerasLayer("https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet1k_b0/feature_vector/2", trainable=trainable),
            # Add the classification layers
            tf.keras.layers.Flatten(),

            # Fully connected layer with multiple of the number of classes
            tf.keras.layers.Dense(len(class_names) * 8,
                                  activation="relu"),

            # Another fully connected layer with multiple of the number of classes
            tf.keras.layers.Dense(len(class_names) * 4,
                                  activation="relu"),

            # Add dropout to reduce overfitting
            tf.keras.layers.Dropout(0.50),

            # Output layer with one node per class, without activation
            tf.keras.layers.Dense(embedding_dim, activation='linear'),
        ]
    )
```

Figure 3. Building the Model

To ensure that changes to the model pipeline did not introduce overfitting, the model was monitored using the loss function of the validation dataset. The number of training epochs was increased from 50 to 100, though early stopping was still implemented to prevent the model from training for too long. Figure 4 below demonstrates the changes in the loss functions of both the training and validation datasets as well as the trends observed in performance metrics over the course of the training process. With the change from multi-hot labels to embeddings, the key metrics previously used to monitor the training process were no longer appropriate for evaluating the model's performance. To tackle this challenge, the loss function was altered to use the mean squared error and performance was primarily evaluated using the cosine similarity between the predicted embeddings and the true embeddings of each audio file. By the conclusion of the training process, the loss functions of both datasets had steadily decreased closer to zero and both cosine similarity values were greater than 0.95.

```
Epoch 1/100
34/34 [==============================] - 64s 1s/step - loss: 0.2811 - mse: 0.2811 - mae: 0.4564 - cosine_similarit
y: 0.3747 - val_loss: 0.1719 - val_mse: 0.1719 - val_mae: 0.3413 - val_cosine_similarity: 0.7259 - lr: 0.0010
Epoch 2/100
34/34 [==============================] - 40s 1s/step - loss: 0.1861 - mse: 0.1861 - mae: 0.3524 - cosine_similarit
y: 0.6697 - val_loss: 0.1089 - val_mse: 0.1089 - val_mae: 0.2655 - val_cosine_similarity: 0.8648 - lr: 0.0010
Epoch 3/100
34/34 [==============================] - 39s 1s/step - loss: 0.1334 - mse: 0.1334 - mae: 0.2942 - cosine_similarit
y: 0.7897 - val_loss: 0.0794 - val_mse: 0.0794 - val_mae: 0.2275 - val_cosine_similarity: 0.9065 - lr: 0.0010
Epoch 4/100
34/34 [==============================] - 36s 971ms/step - loss: 0.1088 - mse: 0.1088 - mae: 0.2650 - cosine_simila
rity: 0.8412 - val_loss: 0.0625 - val_mse: 0.0625 - val_mae: 0.2020 - val_cosine_similarity: 0.9185 - lr: 0.0010
Epoch 5/100
34/34 [==============================] - 35s 964ms/step - loss: 0.0926 - mse: 0.0926 - mae: 0.2432 - cosine_simila
rity: 0.8710 - val_loss: 0.0539 - val_mse: 0.0539 - val_mae: 0.1875 - val_cosine_similarity: 0.9226 - lr: 0.0010
Epoch 96/100
34/34 [==============================] - 8s 217ms/step - loss: 0.0308 - mse: 0.0308 - mae: 0.1352 - cosine_similar
ity: 0.9534 - val_loss: 0.0344 - val_mse: 0.0344 - val_mae: 0.1425 - val_cosine_similarity: 0.9475 - lr: 7.5000e-0
4
Epoch 97/100
34/34 [==============================] - 8s 219ms/step - loss: 0.0304 - mse: 0.0304 - mae: 0.1352 - cosine_similar
ity: 0.9543 - val_loss: 0.0314 - val_mse: 0.0314 - val_mae: 0.1382 - val_cosine_similarity: 0.9517 - lr: 7.5000e-0
4
Epoch 98/100
34/34 [==============================] - 8s 214ms/step - loss: 0.0284 - mse: 0.0284 - mae: 0.1298 - cosine_similar
ity: 0.9575 - val_loss: 0.0312 - val_mse: 0.0312 - val_mae: 0.1373 - val_cosine_similarity: 0.9520 - lr: 7.5000e-0
4
Epoch 99/100
34/34 [==============================] - 8s 218ms/step - loss: 0.0290 - mse: 0.0290 - mae: 0.1317 - cosine_similar
ity: 0.9568 - val_loss: 0.0318 - val_mse: 0.0318 - val_mae: 0.1359 - val_cosine_similarity: 0.9505 - lr: 7.5000e-0
4
Epoch 100/100
34/34 [==============================] - 8s 215ms/step - loss: 0.0290 - mse: 0.0290 - mae: 0.1315 - cosine_similar
ity: 0.9563 - val_loss: 0.0320 - val_mse: 0.0320 - val_mae: 0.1393 - val_cosine_similarity: 0.9510 - lr: 7.5000e-0
4
```

Figure 4. Tracking the loss function and performance metrics during training

Testing the Model

Once the model had been built, it was tested on a small test dataset which yielded promising results. The model demonstrated perfect accuracy for single-species audio files, and one of the species in each of the mixed audio files was successfully identified (see figure 5 for example results output). While the model still requires future improvement to identify all species within an audio clip, the ability to make accurate predictions on a clip with conflicting vocalisations is a significant improvement from previous engine models. Future work should continue to explore the classification of multi-label audio files to also identify less prominent species in these mixed vocalisation recordings.

```
--------------------------------------------------
Sample 1
True classes: ['Daphoenositta chrysoptera']
Predicted Classes:
  - Daphoenositta chrysoptera (sim = 0.9660)
Sim with true: 0.9656

--------------------------------------------------
Sample 2
True classes: ['Falco peregrinus']
Predicted Classes:
  - Falco peregrinus (sim = 0.9940)
Sim with true: 0.9937

--------------------------------------------------
Sample 3
True classes: ['Stomiopera unicolor']
Predicted Classes:
  - Stomiopera unicolor (sim = 0.9830)
Sim with true: 0.9826

--------------------------------------------------
Sample 4
True classes: ['Stomiopera unicolor']
Predicted Classes:
  - Stomiopera unicolor (sim = 0.9790)
Sim with true: 0.9795

--------------------------------------------------
Sample 5
True classes: ['Acanthiza chrysorrhoa', 'Daphoenositta chrysoptera']
Predicted Classes:
  - Daphoenositta chrysoptera (sim = 0.9351)
Sim with true: 0.8913

--------------------------------------------------
```

Figure 5. Sample test results from the updated engine model pipeline

Conclusion

The updated Project Echo model shows significant improvement in handling overlapping animal vocalisations, marking a crucial step toward real-world application in wildlife monitoring. By shifting from multi-class classification to class embeddings and incorporating synthetic mixed-species audio, the model can now accurately identify at least one species in complex recordings. While further work is needed to improve detection of less prominent species, this version lays a strong foundation for future expansion and enhanced conservation efforts in the Otways region.