Engine - Tuning Augmentation Parameters

By: Eugene Ou Yi Koh & Riley Beckett

Project Aim:

The main objective of this project was to develop a comprehensive document and implement code for tuning audio augmentation parameters. To achieve this, we conducted in-depth research and employed various optimization techniques, including **random search** and **grid search**, with a specific focus on fine-tuning parameters that significantly impact the model's performance. These methods were integrated into our audio classification system, which leverages **Convolutional Neural Networks (CNNs)** to differentiate and classify **environmental sounds** and **animal sounds**. Additionally, this project involved **collecting and preprocessing audio data** through techniques such as bandpass filtering, normalization, silence removal, and converting audio signals into spectrograms saved as RGB images. These spectrograms served as input to the CNN model, enabling efficient feature extraction and classification. The parameters optimized in this project were designed to improve a prototype system that can be refined and scaled for use in a real-world model. The methods and insights gained can effectively contribute to **Project Echo's goal** of accurately capturing and monitoring sounds from endangered species in real-world environments.

Hyperparameter Optimization Techniques:

Hyperpartmer optimization (or tuning) involves searching for the best hyperparameters for machine learning in order to maximize its performance on a given task. In this task, we employ both **grid search** and **random search** in our models (separately) to explore and evaluate different hyperparameter configurations.

Random Search:

What is Random Search?

Random search is a hyperparameter optimization technique that is used to randomly sample different combinations of hyperparameter values, either in both specified ranges

or distributions. This is a less computational heavy method compared to grid search (Which evaluates every possible combination) and instead can be utilized to select a subset of parameter combinations based on random sampling.

Why Did We Think of Using Random Search?

We chose random search due to its effectiveness in dealing with high-dimensional hyperparameter spaces, where an exhaustive search would be both computationally expensive and inefficient. Random search strikes an optimal balance between thoroughness and efficiency, ensuring comprehensive coverage of the parameter space and within a limited number of evaluations. Additionally, it allows for the exploration of a wide range of parameter combinations, potentially uncovering configurations that significantly improve performance—results that might not have been anticipated through manual tuning or other methods."

Cons of Random Search

Since the sampling is random the exact optimal combination might not be tested; especially with limited trials. Additionally thorough exploration of parameter interactions is not guaranteed, random search does not follow a structured approach. The randomness may be interpretable; it might not clearly explain why certain configurations work better than others.

Overall:

In the context of Project echo, random search allowed us to optimize key hyperparameters and improve the prototypes model performance. While random search is limited, it contributed to identifying effective combinations of hyperparameters for both the model and data augmentation methods.

Grid Search:

What is grid search?

Grid search is a way to find the optimal hyperparameters for a machine learning model. It works by evaluating all possible combinations of hyperparameter values in a grid, training and validating the model for each combination. It is rather slow due to how

computationally expensive it is due to the fact that it has to test everything. However this also makes it very thorough and allows one to find the optimal configuration.

Why use grid search?

The reason why we chose grid search is because it systematically tests all possible combinations of hyperparameters in a predefined range. This then allows us to search through the entirety of the hyperparameter space to find the optimal combination that would result in the best performance for our machine learning model.

Cons of grid search

Grid search is computationally expensive, taking long amounts of time to run especially with large datasets or many parameters. It struggles with small datasets as we personally found out as the small datasets led to overfitting the validation set and unreliable metrics. This in turn makes it significantly harder to identify the optimal settings. Grid search also lacks prioritization, as it treats all combinations equally and might miss optimal values between predefined points, thus leading to significant computational resources to implement effectively.

Overall:

Grid search is a thorough but computationally expensive method for tuning hyperparameters. It works best with a larger dataset but absolutely struggles with a smaller one. Grid search is actually rather straightforward to implement and easy to parallelize, making it practical for setups with decent hardware. Grid search is particularly useful for understanding how different hyperparameters influence model performance, as it provides detailed insights into the effects of each combination.

Key Features:

Directory Structure Setup

Purpose: A systematic and organized directory structure is created for storing raw audio files, spectrograms, and categorized datasets.

- Automatically generates subdirectories such as:
    - Raw audio categories (Animal Sounds, Environment).
    - Processed spectrogram categories (train, validation, Mixed Sounds).
    - There includes separate subdirectories for training and validation for both animal and environmental sounds.
- **Functions:**
    - **setup_directories(base_dir):** Sets up all required directories.
    - **create_directories(base_dir, subdirs):** Creates subdirectories if they don't already exist.

Audio Preprocessing

Purpose: Enhance audio data quality and diversity for effective analysis and classification. Ensures the audio is clean, normalized, and relevant for training.

- Bandpass Filtering: frequencies between 500 Hz and 8000 Hz are only retained, Important for distinguishing animal and environmental sounds.
    - **Example:** apply_bandpass(y, sr, low=500, high=8000)
- Normalization: Audio signals are scaled between -1 and 1, ensuring uniform amplitude and preventing distortions during training.
    - **Example:** normalize(y)
- Silence Removal: silent segments are detected and removed using amplitude thresholds (top_db=20), and this ensures classifier focuses only on meaningful parts of the audio data.
    - **Example:** remove_silence(y, sr)

Spectrogram Generation

Purpose: audio signals are converted into visual representations (Mel spectrograms) suitable for processing by convolutional neural networks.

- Mel spectrograms are generated with a fixed size (128x128) and this is important for compatibility with the input layer of the CNN.
- The power spectrograms are converted to the decibel scale (logarithmic) and this is important for dynamic range visualization.
- Spectrograms are saved RGB images.
- **Example:**

    - **saved_spectrogram(y, sr, output_path, size=(128, 128)):** Converts audio to spectrogram and saves as an image.
    - **test_spectrograms(input_folder, output_folder):** Generates spectrograms from test audio files.

Audio Augmentation

Purpose: The dataset diversity is expanded and ensures robustness through introducing realistic variations to the audio data.

- Uses the audiomentations library to apply transformations:
    - AddGaussianNoise: Simulates real-world background noise. TimeStretch: the tempo is altered while preserving the pitch.
    - PitchShift: the pitch is changed without affecting the tempo and this
    - mimics the variability in sound sources.
    - Shift: Time-shifts the signal, introducing temporal variations.
- Each audio file is augmented multiple times (e.g., 5 versions), and this helps expand the dataset size to include more variations and results in improving the model's generalization.
- **Example:**
    - **processing_audio(input_dir, output_dir, augmentations):** Applies augmentations and generates spectrograms.

Mixed Sound Generation

Purpose: Prepare the model to classify audio in noisy, real-world conditions by combining different sound types.

- The animal sounds and environmental sounds are combined to create mixed audio clips.
- The mixed signals are normalized to prevent amplitude imbalances.
- The mixed audio is converted into spectrograms for further processing.
- **Example:**
    - **generate_mixed_sounds(animal_dir, env_dir, output_dir):** Creates mixed audio samples and saves spectrograms.
    - **mixed_spectrograms(mixed_dir, target_dirs):** Distributes mixed spectrograms into specific directories.

Dataset Splitting

Purpose: The data is split into training and validation subsets to be able to evaluate the model's generalization ability.

- Divides automatically the augmented data into training and validation sets using the standard 80%-20% split.
- **Example:**
  - **split_data(input_dir, output_dirs, test_size=0.2):** Splits spectrogram data into training and validation directories.

Data Generators

Purpose: Prepare data for the CNN in real-time, reducing memory usage and enabling dynamic augmentations.

- **Training Data Generator:**
  - Real-time augmentations are utilized, for example: rotation, flipping, zoom, and shearing are used to simulate various perspectives of the spectrograms.
  - Rescales pixel values to the [0, 1] range for better gradient propagation.
- **Validation Data Generator:**
  - Only rescales pixel values without applying augmentations to ensure a fair evaluation of model performance.
- **Functions:**
  - **train_generator:** Data generator for training with augmentations.
  - **val_gen:** Data generator for validation.

Convolutional Neural Network (CNN) with Residual Connections

Purpose: A custom CNN architecture is implemented and further optimized for audio spectrogram classification.

- **Features:**
  - Residual Connections: Skip connections prevent gradient vanishing and enhance feature propagation.
  - Batch Normalization: Speeds up training and stabilizes the network by normalizing intermediate layer outputs.
  - Dropout: Helps reduce overfitting by randomly deactivating neurons during training.
  - Global Average Pooling (GAP): Reduces the number of parameters while preserving spatial features.

    - Output Layer: Softmax activation outputs probabilities for two classes (Animal Sounds, Environment).
- **Example:**
    - **cnn_model(input_shape=(128, 128, 3), l2_strength=0.001, dropout_rate=0.4):** Builds and returns the CNN model.

RandomSearch

Purpose:

Perform random sampling of hyperparameters for a Convolutional Neural Network (CNN).

- Identify the hyperparameter set that maximizes validation accuracy.

  Hyperparameters (learning_rate, l2_strength, dropout_rate) are sampled randomly within specified ranges.

- **Validation Evaluation**: Tracks the best-performing hyperparameters based on maximum validation accuracy.

Early Stopping: Monitors validation loss to stop training when no improvement is observed for a set number of epochs. Example: Patience=10

Model Handling: If a pre-trained best model exists then it's loaded and if not the script runs random search and determines the best hyperparameters and saves the best moel

GridSearch

Evaluation Metrics

Purpose: Assess the model's performance and visually display results.

- **Classification Report:** Precision, recall, and F1-score are displayed for both classes.
- **Confusion Matrix:** Correct and incorrect predictions across classes are visualized.
- **Example:**

    - **gen_classification_report(model, data_generator):** Generates the classification report and confusion matrix.

Model Saving and Loading

Purpose: Save and reload the trained model for future use.

- Saves the trained model in HDF5 format
- Implements a function to load the model for evaluating new audio data.
- **Example:**
    - **model.save(model_save_path):** Saves the trained model.
    - **load_trained_model(model_path):** Loads the saved model.

Validating Model

Purpose: Evaluate the model's ability to classify unseen audio data accurately.

- **Spectrogram Generation for Test Data:**
    - The new audio files are converted into spectrograms, ensuring consistency with the training process beforehand.
    - **Example: test_spectrograms(input_folder, output_folder)**
- **Prediction Functionality:**
    - Maps predictions to class labels (Animal Sounds, Environment).
    - Predictions are compared against ground truth labels (from a CSV file).
    - **Example:**
        - **predict(model, spectrogram_folder, ground_truth_csv, class_names)**
    - Provides confidence scores for each prediction.

- **Potential points of improvements:**
    - **Transfer Learning:** we can use pre trained models for feature extraction to boost accuracy while having minimal training.
    - **Learning Rate Scheduler:** we can dynamically adjust the learning rate to improve convergence to find the minimum point.

Random search

Results

Evaluation:

1. **Classification Report**

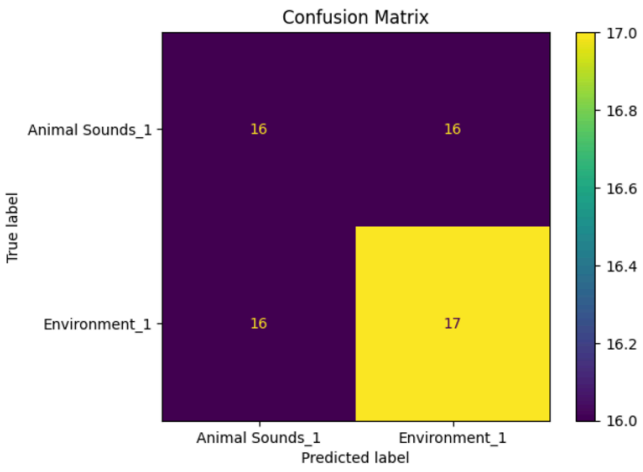
```
Classification Report:
                precision    recall  f1-score   support

Animal Sounds_1       0.50      0.50      0.50        32
  Environment_1       0.52      0.52      0.52        33

       accuracy                           0.51        65
      macro avg       0.51      0.51      0.51        65
   weighted avg       0.51      0.51      0.51        65
```

This classification report shows that the model is balanced but could further improvement performance in classifying "Animal Sounds" and "Environmental Sounds." Both classes have similar precision, recall, and F1-scores around 0.50–0.52. The overall accuracy is 51%, which means the model has somewhat limited discrimination between the two classes.

2. **Confusion Matrix**



This confusion matrix shows the performance of a classification model distinguishing between Animal Sounds" and "Environmental Sounds." The model has some confusion, as it misclassified 16 instances in both directions, but it correctly classified 16 "Animal Sounds" and 17 "Environmental Sounds.

3. Prediction Results

```
Model loaded successfully.
Processing test audio files: 100%|██████████| 12/12 [00:00<00:00, 12003.73it/s]
1/1 [==============================] - 0s 69ms/step
Correct: daintree bat.png -> Animal Sounds (0.95 confidence)
1/1 [==============================] - 0s 19ms/step
Correct: falling branches.png -> Environment Sounds (0.69 confidence)
1/1 [==============================] - ETA: 0s

1/1 [==============================] - 0s 21ms/step
Correct: Giant Banjo Frogs.png -> Animal Sounds (0.76 confidence)
1/1 [==============================] - 0s 19ms/step
Incorrect: Glider Squirrel.png -> Predicted: Environment Sounds (0.90 confidence), Actual: Animal Sounds
1/1 [==============================] - 0s 26ms/step
Incorrect: Golden Bowerbird.png -> Predicted: Environment Sounds (0.90 confidence), Actual: Animal Sounds
1/1 [==============================] - 0s 27ms/step
Correct: Light rain.png -> Environment Sounds (0.97 confidence)
1/1 [==============================] - 0s 29ms/step
Incorrect: night Frog.png -> Predicted: Environment Sounds (0.72 confidence), Actual: Animal Sounds
1/1 [==============================] - 0s 22ms/step
Correct: pademelon.png -> Animal Sounds (0.75 confidence)
1/1 [==============================] - 0s 24ms/step
Correct: shaking branch.png -> Environment Sounds (0.82 confidence)
1/1 [==============================] - 0s 20ms/step
Correct: storm wind in trees.png -> Environment Sounds (0.83 confidence)
1/1 [==============================] - 0s 21ms/step
Correct: walking in the mud.png -> Environment Sounds (0.75 confidence)
1/1 [==============================] - 0s 19ms/step
Correct: water stream.png -> Environment Sounds (0.98 confidence)

Accuracy: 75.00%
```

The first image shows predictions with confidence scores, highlighting some confident misclassifications (e.g., "Glider Squirrel" classified as "Environment Sounds" with 0.90 confidence). This suggests overlapping features between classes.

Grid search

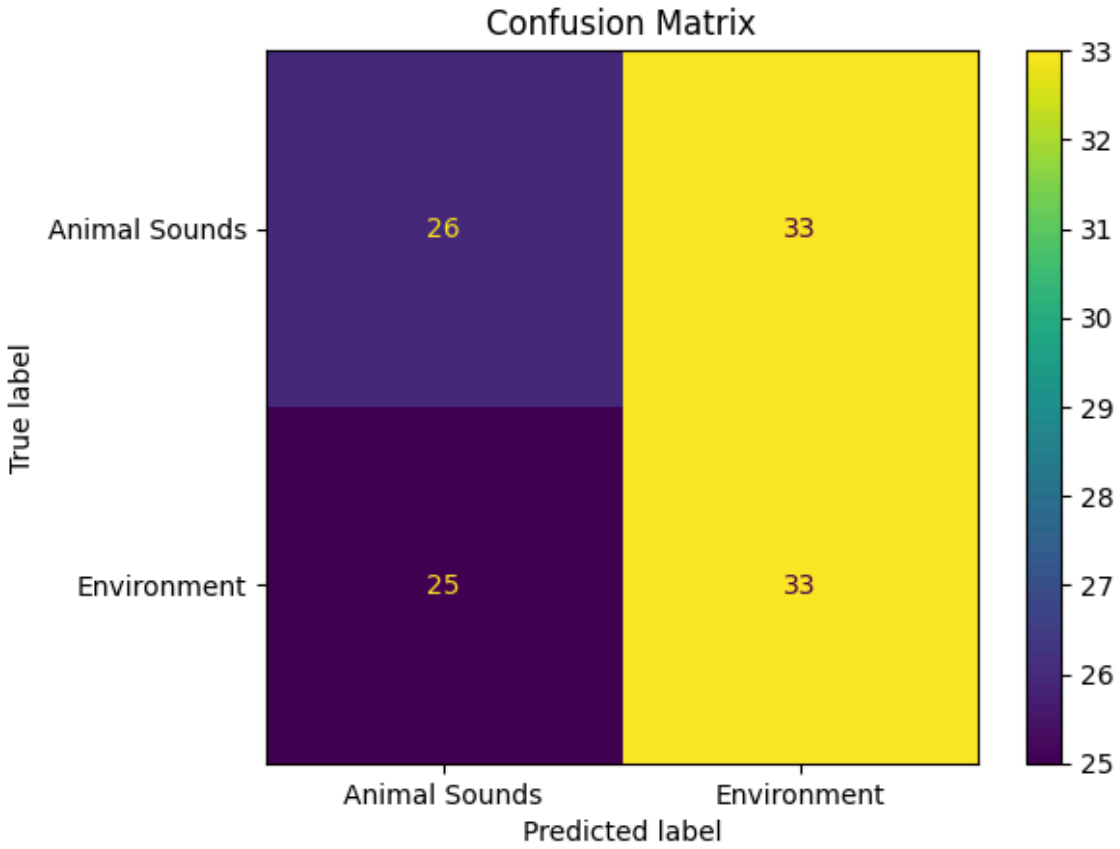
1. Classification Report

Classification Report:

	precision	recall	f1-score	support
Animal Sounds	0.51	0.44	0.47	59
Environment	0.50	0.57	0.53	58
accuracy			0.50	117
macro avg	0.50	0.50	0.50	117
weighted avg	0.50	0.50	0.50	117

This report shows us that our grid search model is not doing too well. The accuracy is only around 50 percent, the precision recall and F1 score are also all around 0.5 for both our Animal and Environment classes. This means that the model is unfortunately predicting randomly without properly learning the difference between the two classes thus showing us that the grid search did not find effective hyperparameters or that the data or feature may not be enough for the model to distinguish between the two classes properly. In the future when we are not scarce on time, we can have better and more abundant data to work with.

1. **Confusion Matrix**



The confusion matrix shows us that the grid search has room for improvement, it does manage to correctly classify 26 animal sounds and 33 environment sounds, demonstrating some ability to differentiate between the classes. However, it does struggle with misclassifications as 33 animal sounds are wrongly classified as environment and 25 environment sounds are classified as animal sounds. This tells us that the model does struggle with overlapping features. In the future, we can further improve our model through enhanced data preprocessing, augmentation or refined hyperparameter tuning.

1. **Prediction results**

```
Model loaded successfully.
Processing test audio files: 100%|██████████| 12/12 [00:00<00:00, 12003.73it/s]
1/1 [==============================] - 0s 69ms/step
Correct: daintree bat.png -> Animal Sounds (0.95 confidence)
1/1 [==============================] - 0s 19ms/step
Correct: falling branches.png -> Environment Sounds (0.69 confidence)
1/1 [==============================] - ETA: 0s

1/1 [==============================] - 0s 21ms/step
Correct: Giant Banjo Frogs.png -> Animal Sounds (0.76 confidence)
1/1 [==============================] - 0s 19ms/step
Incorrect: Glider Squirrel.png -> Predicted: Environment Sounds (0.90 confidence), Actual: Animal Sounds
1/1 [==============================] - 0s 26ms/step
Incorrect: Golden Bowerbird.png -> Predicted: Environment Sounds (0.90 confidence), Actual: Animal Sounds
1/1 [==============================] - 0s 27ms/step
Correct: Light rain.png -> Environment Sounds (0.97 confidence)
1/1 [==============================] - 0s 29ms/step
Incorrect: night Frog.png -> Predicted: Environment Sounds (0.72 confidence), Actual: Animal Sounds
1/1 [==============================] - 0s 22ms/step
Correct: pademelon.png -> Animal Sounds (0.75 confidence)
1/1 [==============================] - 0s 24ms/step
Correct: shaking branch.png -> Environment Sounds (0.82 confidence)
1/1 [==============================] - 0s 20ms/step
Correct: storm wind in trees.png -> Environment Sounds (0.83 confidence)
1/1 [==============================] - 0s 21ms/step
Correct: walking in the mud.png -> Environment Sounds (0.75 confidence)
1/1 [==============================] - 0s 19ms/step
Correct: water stream.png -> Environment Sounds (0.98 confidence)

Accuracy: 75.00%
```

This output shows that the model achieves 75% accuracy on the test set, which is an improvement. It correctly classifies most samples with high confidence but struggles with certain animal sounds misclassified as environment sounds. The high confidence in wrong predictions suggests the model is overconfident in some cases and may need further adjustments, like better data augmentation or refined training, to handle edge cases more effectively.