# Term Project Report Template

Duygu Onder (21COMP1014)  
Isa Berk Geriler (20SOFT1034)

## Title:

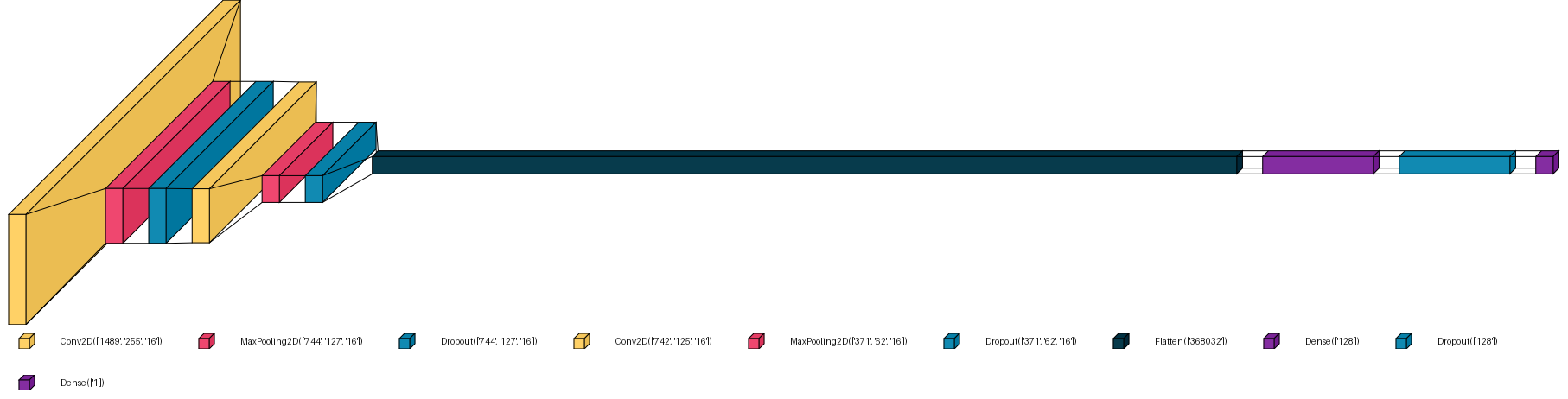
Audio Analysis of Capuchin Bird Calls Using CNNs

## 1. Introduction

Capuchin birds, native to South and Central America, play a significant role in their ecosystems, yet their behavior and communication patterns remain under-researched. Understanding these vocalizations is essential for studying their ecology, monitoring populations, and identifying potential threats such as habitat loss and climate change. The primary challenge lies in the accurate classification and analysis of these calls from complex audio recordings, often plagued by environmental noise and overlapping signals. Traditional bioacoustics analysis methods require significant manual effort and expertise, which limits scalability and real-time application.

## 2. Methodology

This project aims to leverage deep learning, specifically Convolutional Neural Networks (CNNs), to classify and analyze capuchin bird calls from audio data. CNNs have shown remarkable success in pattern recognition tasks, including image and audio signal processing. By transforming audio signals into spectrograms and feeding them into CNN models, we can enable automated, high-accuracy classification of bird calls. This solution has the potential to streamline bioacoustics research and contribute to the conservation of capuchin birds by facilitating large-scale monitoring efforts.  
  
Starting by audio preprocessing, audio recordings of Capuchin and non-Capuchin calls were labeled and converted from stereo 48 kHz to mono-channel at a 16 kHz sampling rate. Also, unnecessary dimensions from the data tensor were removed. This preprocessing step ensures uniformity in the data and reduces computational complexity, making the data suitable for efficient analysis. Then audio waveforms were transformed into spectrograms by using STFT (Short-Time Fourier Transform) technique. This technique converts time-domain audio signals into a time-frequency representation, highlighting patterns specific to Capuchin bird calls. For CNN input compatibility normalize and reshaping steps were done. By visualizing audio as spectrograms, CNNs can leverage their proven capability in image recognition tasks to process and classify audio data. CNN architecture model used for classification comprises these components:

  
Figure: Architecture of the CNN, created by visualkeras

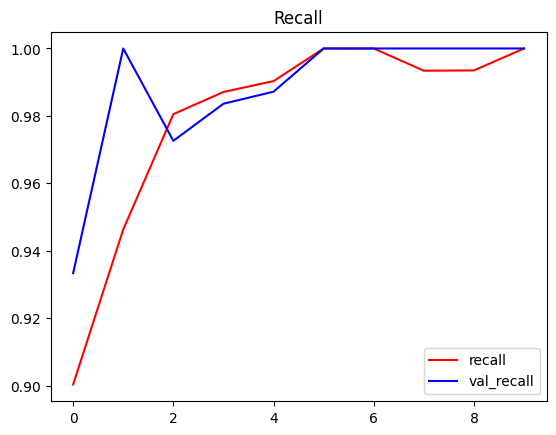
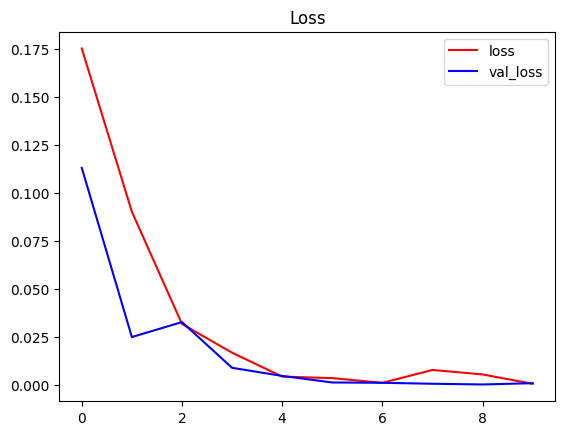
**- Convolution Layers**: Extract spatial features from spectrograms using 16 3x3 kernels (filters). **- Pooling Layers**: Perform down-sampling through max-pooling to reduce dimensionality and computational load.   
**- Dropout Layers**: Introduce regularization to mitigate overfitting.  
**- Dense Layers**: Fully connected layers interpret extracted features for classification.  
**- Sigmoid Activation (last Dense Layer)**: Outputs a probability for binary classification.

The model is optimized using the Adam optimizer, known for its efficiency in handling sparse gradients, and Binary Crossentropy as the loss function, which is well-suited for binary classification tasks like this project. For metrics, accuracy, precision, recall observed. Also, EarlyStopping is used for stopping training if validation loss doesn’t improve for 2 consecutive epochs. Data augmentation techniques, such as adding white noise can be applied to enhance model robustness. This method helps simulate various environmental conditions and improve the generalizability of the model.   
  
The choice of CNNs is motivated by their demonstrated success in pattern recognition tasks. Fully Connected Networks like MLPs are sensitive to location shifts and since this project is about finding the bird calls, CNNs make the perfect choice. Also, the use modern optimization techniques like Adam optimizer ensures faster converges and higher accuracy during training.

## 3. Experiments

To execute this project, we will use the following tools and frameworks:   
 • Python for data preprocessing, model training, and evaluation.   
 • TensorFlow/Keras for building and training the CNN models.  
 • Google Colab for exploratory analysis and documentation, and scalable cloud-based training (using GPUs like T4 or A100).   
 • Pre-existing capuchin bird audio recordings, such as those provided in the [Unlocked Challenge](https://www.hp.com/us-en/workstations/industries/data-science/unlocked-challenge.html) or related GitHub repositories ([Z-Unlocked Challenge 3](https://github.com/Z-Unlocked/Unlocked_Challenge_1)). The dataset consists of 100 forests, 217 recordings of capuchin birds calls and 593 recordings of other bird species calls. The data was split into as follows: **Training Data:** 70%, **Validation Data:** 15%, **Test Data:** 15%. Also, Augmentation techniques applied to dataset randomly. These techniques include adding white noise in the background, applying High-pass and Low-pass filters to filter out low and high frequencies respectfully, and normalizing the audios.

• Scikit-learn for confusion matrix and classification report.  
 • VisualKeras for visualizing the CNN architecture.  
 • Numpy and Matplotlib

• Receiver Operating Characteristic (ROC) curve: Finds the optimal threshold for a binary classifier by maximizing the trade-off between the true positive rate (TPR) and false positive rate (FPR) using the ROC curve. The threshold that maximizes "tpr - fpr" is considered optimal, helping improve classification performance.  
  
Model configuration is a CNN with two convolutional layers, max pooling and dropout layers to prevent overfitting. Adam optimizer and Binary Crossentropy loss function for efficient binary classification. The following metrics were used to assess model performance:  
 • **Accuracy:** Proportion of correct predictions.  
 • **Precision:** Proportion of true positives among all predicted positives.  
 • **Recall:** Proportion of true positives among all actual positives.  
 • **Loss:** Binary Crossentropy loss to measure prediction errors.  
  
Here is the accuracy, precision, recall and loss plots that obtained: A graph with a line and a line

Description automatically generated with medium confidenceA graph with red and blue lines

Description automatically generated

## 4. Results and Analysis

**Key Findings:**  
During the training process, the model demonstrated the following:  
 - Training and validation accuracy improved over the epochs, with an early stopping mechanism to avoid overfitting.  
 - Loss generally decreased for both training and validation datasets.  
 - Early stopping ensured that the model did not overfit by taking into account when the validation loss result in 2 epoch is greater than the minimum validation loss, maintaining generalization across dataset.  
  
As for the metrics shown these followings:  
 - **Accuracy:** The model achieved an overall accuracy of approximately 98% on the test dataset, confirming its effectiveness in distinguishing Capuchin calls from non-Capuchin audio.  
 - **Precision**: High precision (approximately %98) indicates the model effectively minimizes false positives, crucial for applications where misclassification could lead to unnecessary alerts.   
 - **Recall**: With a recall of approximately 97%, the model successfully detects most of the Capuchin bird calls, reducing false negatives.  
 - **F-1 Score**: The balance between precision and recall highlights the model’s overall reliability.

metin, ekran görüntüsü, renklilik, dikdörtgen içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure: Confusion Matrix and Classification Report

Data augmentation techniques randomly such as adding white noise to background, normalizing the sounds (between 0 and 1) and adding high-pass/low-pass filters to eliminate low/high frequencies enhanced the robustness of the model. These augmentations simulated real-world audio conditions, leading to better generalization.

Confusion matrices displayed accurate classification with high true positive rate for Capuchin calls and effective distinction from non-Capuchin sounds. Spectrograms and their CNN feature maps indicate that the model effectively extracted meaningful patterns from audio signals.

ekran görüntüsü, mor içeren bir resim

Açıklama otomatik olarak oluşturuldu

Figure: Capuchin Call Spectrogram  
  
ekran görüntüsü, mor, leylak, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu  
 Figure: Non-Capuchin Call Spectrogram

**Challenges Faced:  
  
- Input Shape Issues:** During the initial stages, mismatches in data shapes/dimensions caused errors in model training (model.fit) because our spectrograms would be generated in 3D, but the model requires 4D inputs. This was resolved by setting and reassuring the input dimensions (4D) during the spectrogram generation.  
- **Model Overfitting**: Without dropout layers, the model showed poor generalization due to overfitting. Incorporating dropout and max pooling significantly improved the results.  
- **Parameter Optimization**: Early iterations of the model without max pooling led to higher parameter, high computational complexity and slower convergence. Introducing max pooling reduced the parameter count and improved training efficiency.  
- **Library Choices**: While libraries like Librosa are popular for audio processing, TensorFlow-IO was chosen for its seamless integration with TensorFlow models, also Librosa works with numpy arrays but when building the data pipeline, we deal with tensors, so it makes the integration process difficult.  
-**Random Data Augmentation:** Since we are applying these augmentations randomly, each notebook run can alter augmented sounds, overwrite, and affect the results. Once a suitalbe augmentation is done, it must be commented out or the previous findings may vanish/be slightly phrone to error (e.g. 0 calls on recording\_18 can become 1 in worse case). **-Noise filtering:** Further refinement is needed for complex overlapping sounds. **Analysis and Conclusion:**Similar studies using CNNs have reported accuracy in the range of 85–90% for similar tasks. This project’s model achieves comparable or superior performance while incorporating data augmentation to handle diverse environmental conditions effectively. Here are some of the repos and Kaggle notebooks which we found like our model:

<https://www.kaggle.com/code/lkergalipatak/bird-audio-classification-with-tensorflow> <https://www.kaggle.com/code/hakim11/z-by-hp-unlocked-challenge-3-audio-processing/notebook#Build-Preprocessing-Function> <https://github.com/nicknochnack/DeepAudioClassification/blob/main/results.csv>  
<https://github.com/manvix404/Audio-Classification/blob/master/results.csv>  
  
**Key Findings**- The model performed consistently, with accurate call counts for key recordings like e.g. recording\_08 (25 calls) and recording\_87 (24 calls).  
- It correctly identified recording\_18 with 0 calls, demonstrating robustness against noisy data.  
- Slight undercounting occurred in recording\_11 (2 calls in our model vs. 3 actual), showing room for improvement with sparse signals.  
- Most of the recordings had less than 5 calls per minute, this can indicate change in their habitat, or lessening species (below is a reference histogram for frequencies).

metin, ekran görüntüsü, ekran, görüntüleme, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu  
   
Figure: Histogram of Capuchin Calls Frequencies/Recordings

**Comparison with Existing Models**  
- **Manvix404’s Model:** Underestimated in key recordings and misclassified recording\_18 (6 calls).  
- **Hakim’s Model**: Overestimated noise in recording\_18 (5 calls).  
- **Ilker’s Model:** Overestimated recording\_18 (7 calls) and showed variability in results.  
-Robust noise handling: Correctly identified recording\_18 with 0 calls.  
-Consistent performance on key recordings.  
-Enhanced generalization via data augmentation.  
**Conclusion**Our model’s robustness and accuracy position it as a reliable tool for bioacoustic research, outperforming other models in handling noisy data. The model successfully classified Capuchin bird calls with 98% accuracy and can be used for detecting capuchin bird calls on longer recordings. Future improvements will focus on noise filtering, sparse signal detection, and training on diverse datasets**.**