DEEP LEARNING PROJECT

# HUMAN SCREAM DETECTION

July 10, 2024

Anastasia Drakou - 2022202304006
Aikaterini Karathanou - 2022202304009

# Contents

## Role of the Project

The Human Scream Detection Dataset is designed to identify and analyze human screams, which is vital for developing machine learning models used in security, healthcare, and emergency response systems. Accurate scream detection can significantly enhance safety by alerting authorities to emergencies, improving video surveillance, and enabling automatic detection of violent incidents. In healthcare, it supports monitoring patients who may be in distress, particularly when immediate human intervention is not possible. This dataset is crucial for advancing safety technologies and ensuring timely assistance in critical situations.

**Figure 1:** Human Scream

Our objective is to determine the most effective deep learning model for detecting screaming sounds, with the goal of applying it to real, unseen data. Ultimately, we aim for the model to function as an effective surveillance tool in the future.

## Files of the Project

- **Load_and_Preprocess_EDA.ipynb** : Load and preprocess our audio data into 17 features for EDA.
- **EDA.ipynb** : Exploratory Data Analysis of our main dataset.
- **Load_and_Preprocess.ipynb** : Load and Preprocess of our audio data into 128 me spectograms
- **SVM.ipynb** : A machine learning algorithm, SVM, will be used as a baseline to compare the metrics with deep learning models
- **FNN.ipynb** : Fully Connected Neural Network
- **CNN.ipynb** : Convolutional Neural Network
- **CNN_Regularization.ipynb** : Convolutional Neural Network with Regularizations

- **CNN_f1_score_callbacks.ipynb** : Convolutional Neural Network woth F1 score in callbacks
- **ResNet.ipynb** : Transfer learning model
- **Deploy_Load_and_Preprocess.ipynb** : Load and Preprocess of the deployment dataset
- **Deploy_Resnet.ipynb** : Running a transfer learning model to the deployment dataset

## Dataset

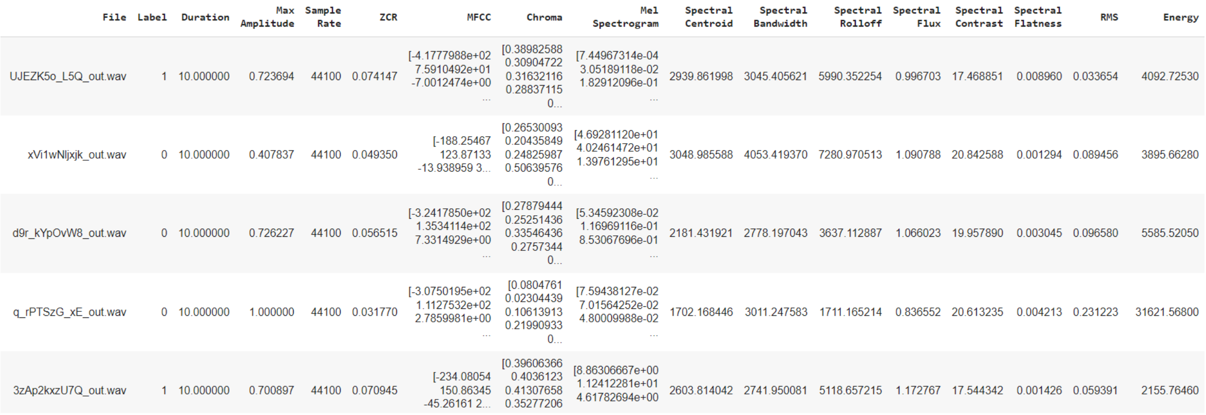
**Dataset :** The dataset utilized in this analysis was sourced from Kaggle. The link to access the dataset is provided: Human Scream Detection Dataset

**Noise Dataset :** The noise dataset utilized in this project for data augmentation was obtained from a specialized sound engineer.

**Deployment Dataset :** The dataset utilized in this analysis was sourced from two different Kaggle datasets. The link to access the dataset is provided: for Screaming : Screaming Dataset and for Not Screaming : Urban Sounds + Emergency Vehicle Siren Sounds.

## Feature Extraction for EDA

In detail, the extracted features include the duration of the audio signal, the maximum amplitude, the sample rate, the zero-crossing rate (ZCR), the Mel-Frequency Cepstral Coefficients (MFCC), chroma features, the Mel spectrogram, the spectral centroid, the spectral bandwidth, the spectral rolloff, the spectral flux, the spectral contrast, the spectral flatness, the root mean square (RMS) value, and the energy of the signal. This process ensures that the machine learning model has all the necessary information to understand and recognize different types of sounds. Detailed extraction and analysis of these features ensure that the model can achieve high performance, accurately identifying screams. The extracted features are then used for training and evaluating machine learning models. During training, the models learn to recognize patterns and differences in features between screams and non-screams. In the evaluation phase, we test the models to confirm their accuracy and effectiveness. This approach ensures the development of a reliable scream detection system, which can be applied in various practical uses.



| File | Label | Duration | Max Amplitude | Sample Rate | ZCR | MFCC | Chroma | Mel Spectrogram | Spectral Centroid | Spectral Bandwidth | Spectral Rolloff | Spectral Flux | Spectral Contrast | Spectral Flatness | RMS | Energy |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UJEZK5o_L5Q_out.wav | 1 | 10.000000 | 0.723694 | 44100 | 0.074147 | [-4.1777988e+02<br>7.5910492e+01<br>-7.0012474e+00<br>... | [0.38982588<br>0.30904722<br>0.31632116<br>0.28837115<br>0... | [7.44967314e-04<br>3.05189118e-02<br>1.82912096e-01<br>... | 2939.861998 | 3045.405621 | 5990.352254 | 0.996703 | 17.468851 | 0.008960 | 0.033654 | 4092.72530 |
| xVi1wNljxjk_out.wav | 0 | 10.000000 | 0.407837 | 44100 | 0.049350 | [-188.25467<br>123.87133<br>-13.938959 3... | [0.26530093<br>0.20435849<br>0.24825987<br>0.50639576<br>0... | [4.69281120e+01<br>4.02461472e+01<br>1.39761295e+01<br>... | 3048.985588 | 4053.419370 | 7280.970513 | 1.090788 | 20.842588 | 0.001294 | 0.089456 | 3895.66280 |
| d9r_kYpOvW8_out.wav | 0 | 10.000000 | 0.726227 | 44100 | 0.056515 | [-3.2417850e+02<br>1.3534114e+02<br>7.3314929e+00<br>... | [0.27879444<br>0.25251436<br>0.33546436<br>0.2757344<br>0... | [5.34592308e-02<br>1.16969116e-01<br>8.53067696e-01<br>... | 2181.431921 | 2778.197043 | 3637.112887 | 1.066023 | 19.957890 | 0.003045 | 0.096580 | 5585.52050 |
| q_rPTSzG_xE_out.wav | 0 | 10.000000 | 1.000000 | 44100 | 0.031770 | [-3.0750195e+02<br>1.1127532e+02<br>2.7859981e+00<br>... | [0.0804761<br>0.02304439<br>0.10613913<br>0.21990933<br>0... | [7.59438127e-02<br>7.01564252e-02<br>4.80009988e-02<br>... | 1702.168446 | 3011.247583 | 1711.165214 | 0.836552 | 20.613235 | 0.004213 | 0.231223 | 31621.56800 |
| 3zAp2kxzU7Q_out.wav | 1 | 10.000000 | 0.700897 | 44100 | 0.070945 | [-234.08054<br>150.86345<br>-45.26161 2... | [0.39606366<br>0.4036123<br>0.41307658<br>0.35277206 | [8.86306667e+00<br>1.12412281e+01<br>4.61782694e+00 | 2603.814042 | 2741.950081 | 5118.657215 | 1.172767 | 17.544342 | 0.001426 | 0.059391 | 2155.76460 |

**Figure 2:** Dataset visualization

# EDA

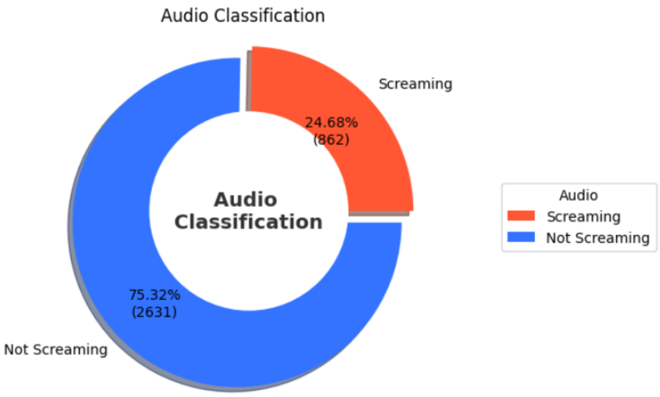


**Figure 3**

The dataset consists of 862 screaming audio files and 2631 non-screaming audio files, indicating an imbalance. The screaming audio files include a variety of screams, both with and without background noises. The non-screaming files feature sounds like music, speech, human voices, and environmental noises. Since this is a binary classification problem, we will define class 1 as the screaming audios and class 0 as the non-screaming audios. Below, we present a plot showing the duration of audio files, displaying the counts of audios per duration. It is evident that 92% of the audio files have a duration of 9 to 10 seconds, with the maximum length being 10 seconds.

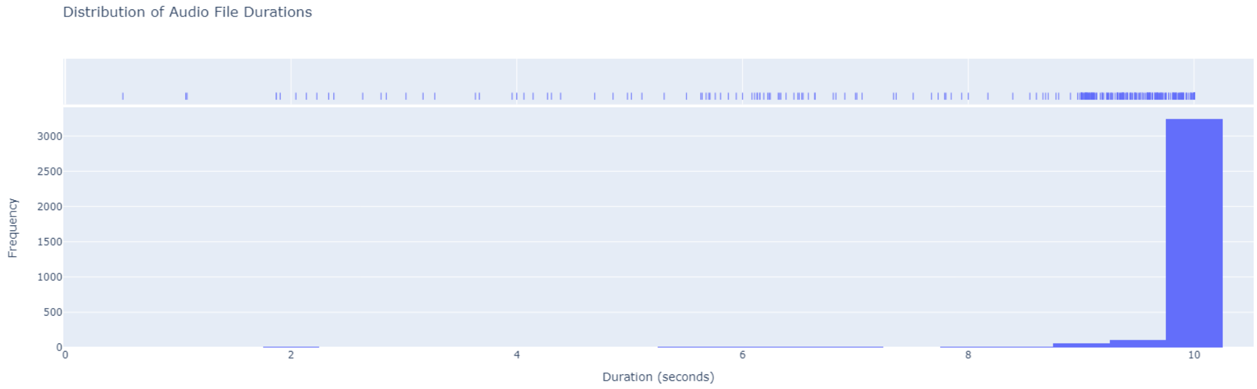


**Figure 4:** Duration bar plot

It is evident from the feature extraction in our exploratory data analysis that we obtained 17 different features, providing valuable insights into the differences between the screaming audio files and the non-screaming audio files. For instance, we observed distinct variations in energy and Zero Crossing Rate (ZCR).

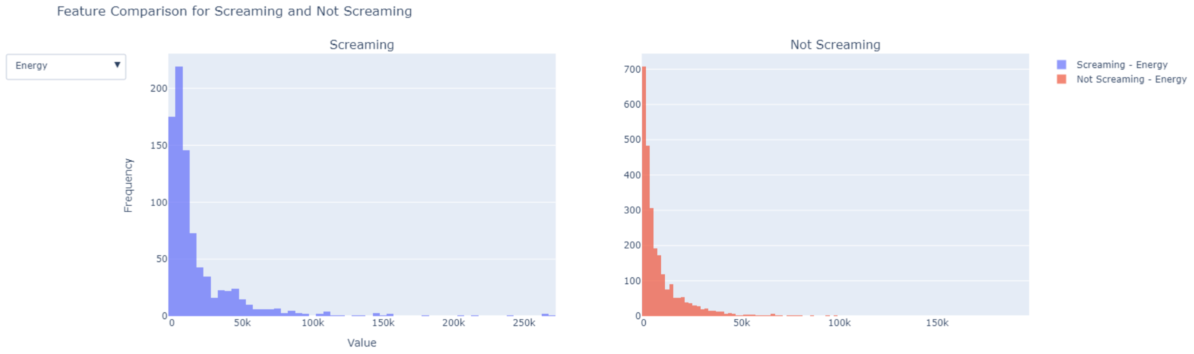


**Figure 5:** Energy

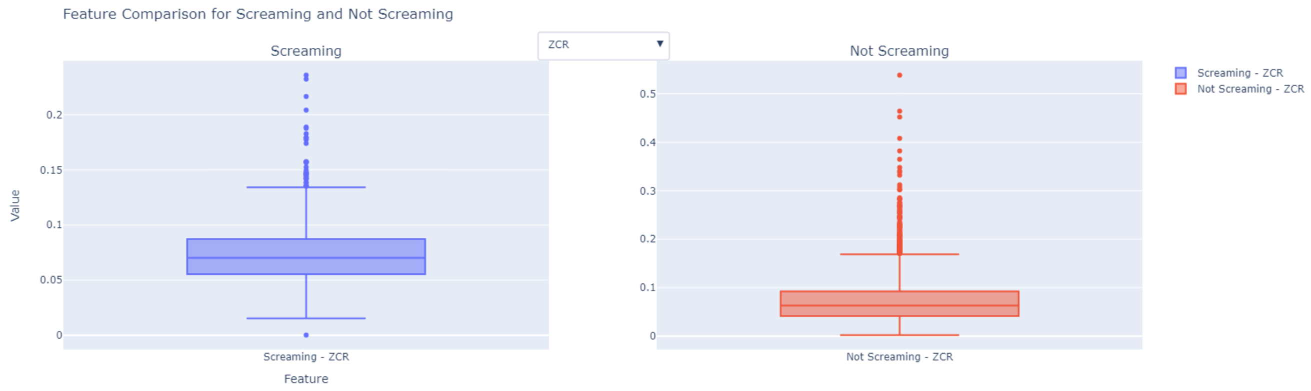


**Figure 6:** ZCR

# Load and Preprocess

The audio files are adjusted to a maximum duration of max_length by trimming or padding with silence at the end, ensuring consistency across samples. Subsequently, the power spectrogram is computed using mel-frequency transformation, representing the sound's energy across 128 frequency bins, covering the spectrum of the audio signal's frequencies. The choice of using 128 frequency bins in the power spectrogram covers a broad spectrum of frequencies, typically sufficient to represent the fundamental frequency characteristics of the sound with adequate resolution. This allows for capturing essential frequency components across the audio signal's spectrum, ensuring effective analysis and representation of its energy distribution. We created bar plots to illustrate the class distributions across the training, validation, and test sets after randomly splitting our data into these subsets (80% training, 10% validation, 10% test) using a random process.

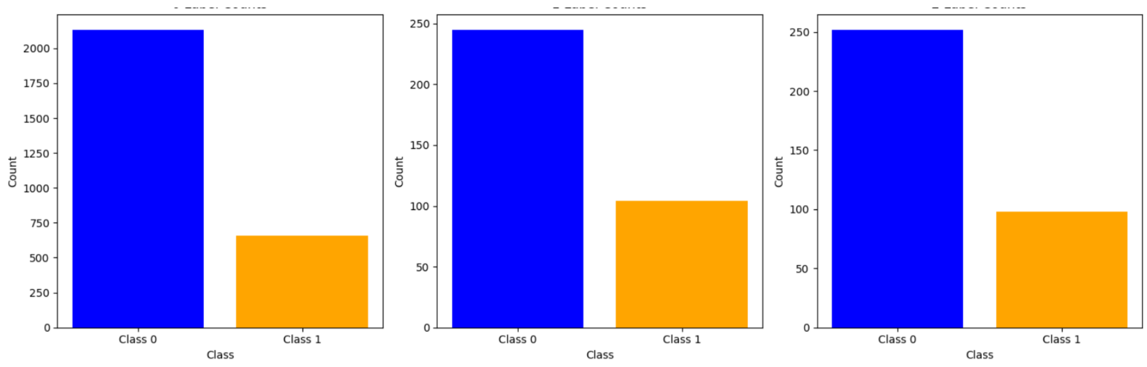


**Figure 7:** Train data, Validation data, and Test data

## Data Augmentation

To expand our dataset and generate more examples for class 1, we applied data augmentation techniques. Such as noise input and time shift.

### Noise

To increase our dataset and generate more data for our minority class, we added background noises to the audio files. The background noises used include traffic sounds, rain sounds, bird sounds, and crowd sounds. Each audio file in class 1 was randomly assigned a background noise at a noise level of 0.4. To prevent bias towards noise, we also added background noise to 20% of the audio files in class 0.

### Time Shift

Time-shifting in screaming audio files involves altering the starting point of the audio signal, effectively moving it forward or backward in time. This process creates a new version of the audio where the timing of the content is changed but the overall sound remains the same, helping to generate variations for data augmentation or analysis.

The time shift technique was applied in 50% of train scream dataset. Following the data augmentation, our training dataset remains imbalanced, but the imbalance has been significantly reduced compared to before. Furthermore, applying time shift can aid in improving the training or evaluation of machine learning models for sound recognition. It can be used to create customized training data that includes variations of sound events that the model needs to learn. This approach can lead to more accurate and generalized learning because the model will be trained to recognize the same categories of sounds regardless of their temporal position. This technique enhances the model's ability to generalize across different instances of the same sound categories, thus improving overall performance in real-world applications.

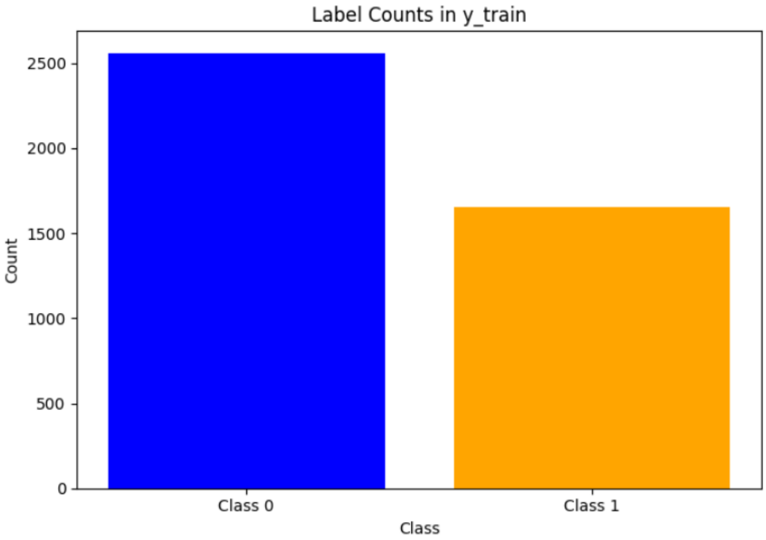
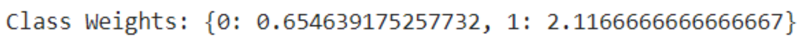


**Figure 8:** Augmented Data of training dataset

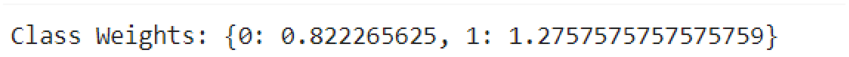
## Class of Weights

Class weights are used in classification problems to address class imbalance in the data. They are automatically computed based on the frequency of each class, giving more weight to less frequent classes to ensure effective training across all classes of the problem. This approach is crucial in cases where the data exhibit class imbalance and helps optimize the model's performance in real-world applications. Initially, we computed the weights on the baseline dataset, and subsequently, we repeated the process incorporating augmented data.



```
Class Weights: {0: 0.654639175257732, 1: 2.1166666666666667}
```

**Figure 9:** Class weights on Baseline



```
Class Weights: {0: 0.822265625, 1: 1.2757575757575759}
```

**Figure 10:** Class weights included augmented data

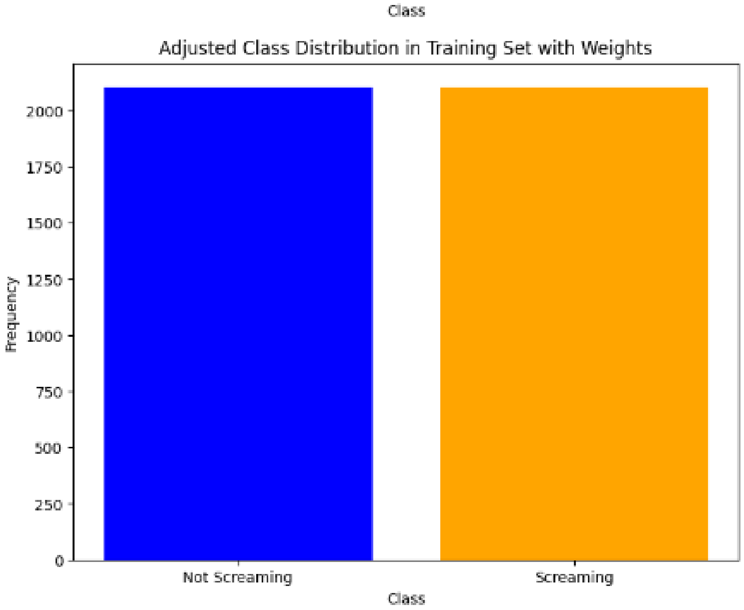
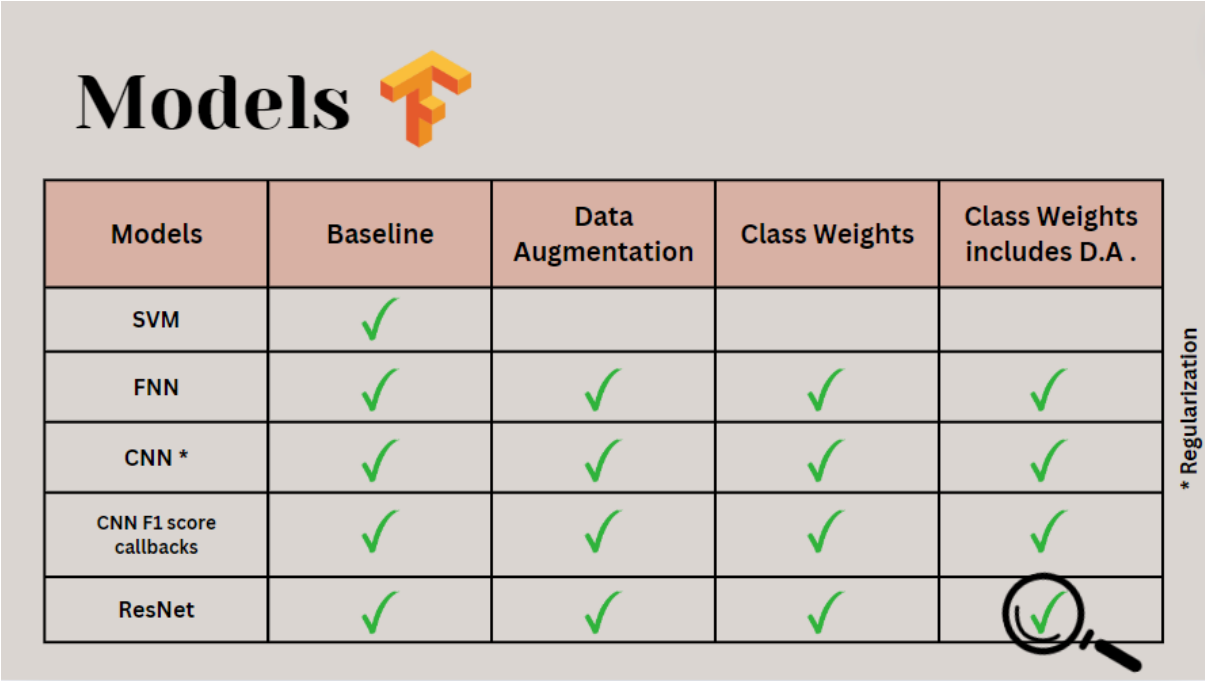


**Figure 11:** Distribution of Classes after Class weights

## Models

Initially, we applied an SVM model to our raw (baseline) dataset to establish benchmark metrics for comparison. All deep learning models followed a consistent process: first, they were trained on the raw dataset, then on the augmented dataset, followed by the raw dataset with a weighted loss function, and finally on the augmented dataset with a weighted loss function.



Models

| Models | Baseline | Data Augmentation | Class Weights | Class Weights includes D.A . |
|---|---|---|---|---|
| SVM | ✓ | | | |
| FNN | ✓ | ✓ | ✓ | ✓ |
| CNN * | ✓ | ✓ | ✓ | ✓ |
| CNN F1 score callbacks | ✓ | ✓ | ✓ | ✓ |
| ResNet | ✓ | ✓ | ✓ | ✓ |

* Regularization

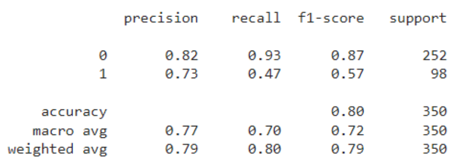
**Figure 12:** Models

As we moved from model to model, improvements in performance were observed. The results, particularly F1 scores and the number of true positives, were compared across models. Notably, the CNN model's F1 score with callbacks did not perform as well. However, applying regularization to the CNN model led to a significant performance boost. Despite this, the ResNet model ultimately outperformed all others in terms of performance.

## Models' Results

### Fully connected Neural Network

Here, we can see the results of the fully connected neural network. It is evident that this model performs better than our baseline SVM machine learning model. However, it does not show significant improvement over the results from our raw data, despite following the necessary steps. Here are the results of the raw data.



```
              precision    recall  f1-score   support

           0       0.82      0.93      0.87       252
           1       0.73      0.47      0.57        98

    accuracy                           0.80       350
   macro avg       0.77      0.70      0.72       350
weighted avg       0.79      0.80      0.79       350
```

**Figure 13:** Results

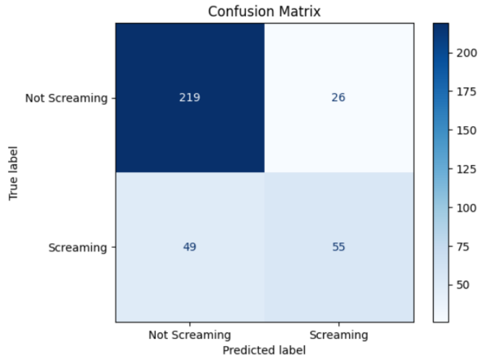
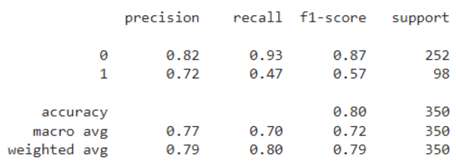


**Figure 14:** Confusion Matrix

Here, are the results of augmented data with the weight loss function.



```
              precision    recall  f1-score   support

           0       0.82      0.93      0.87       252
           1       0.72      0.47      0.57        98

    accuracy                           0.80       350
   macro avg       0.77      0.70      0.72       350
weighted avg       0.79      0.80      0.79       350
```

**Figure 15:** Results

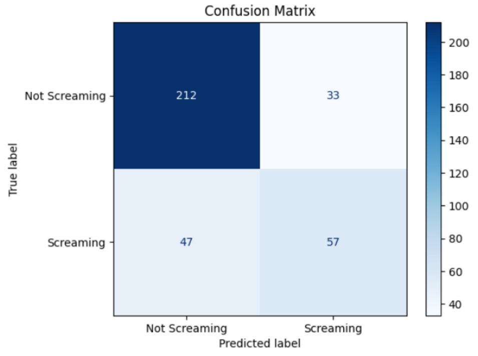
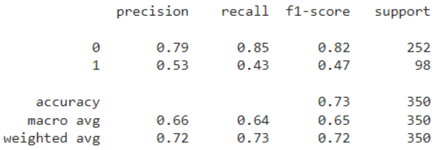


**Figure 16:** Confusion Matrix

## Convolutional Neural Network

The Convolutional Neural Network (CNN) shows better results than the Fully Connected Neural Network (FNN). By following the process, we can see continuous improvement in the results.



```
              precision    recall  f1-score   support

           0       0.79      0.85      0.82       252
           1       0.53      0.43      0.47        98

    accuracy                           0.73       350
   macro avg       0.66      0.64      0.65       350
weighted avg       0.72      0.73      0.72       350
```

**Figure 17:** Results

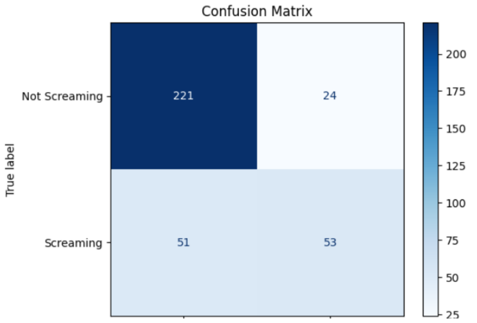


**Figure 18:** Confusion Matrix

We also applied regularization to this model, leading to significant improvements in its performance. Below are the results, along with graphs of loss and accuracy, demonstrating that this CNN model performs exceptionally well.

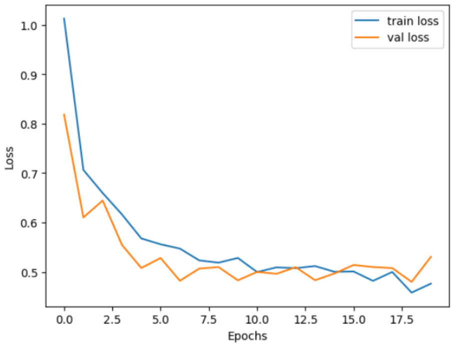


**Figure 19:** Loss

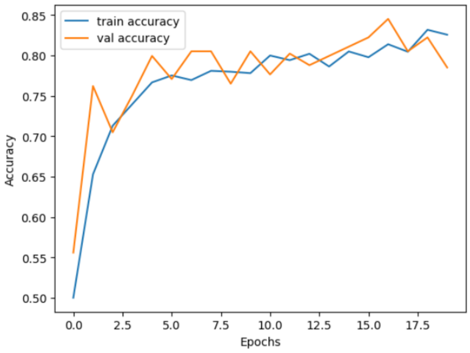
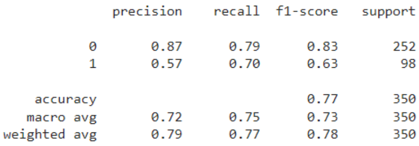


**Figure 20:** Accuracy



```
              precision    recall  f1-score   support

           0       0.87      0.79      0.83       252
           1       0.57      0.70      0.63        98

    accuracy                           0.77       350
   macro avg       0.72      0.75      0.73       350
weighted avg       0.79      0.77      0.78       350
```

**Figure 21:** Results

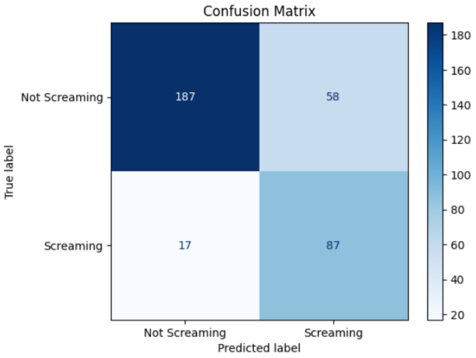
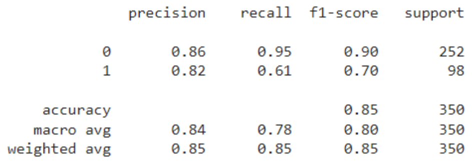


**Figure 22:** Confusion Matrix

### ResNet

We also applied a transfer learning model, ResNet, which yielded surprisingly better results than any other model, even when applied to the raw dataset.



```
              precision    recall  f1-score   support

           0       0.86      0.95      0.90       252
           1       0.82      0.61      0.70        98

    accuracy                           0.85       350
   macro avg       0.84      0.78      0.80       350
weighted avg       0.85      0.85      0.85       350
```

**Figure 23:** Results

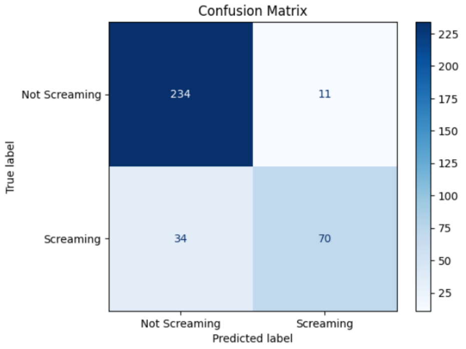
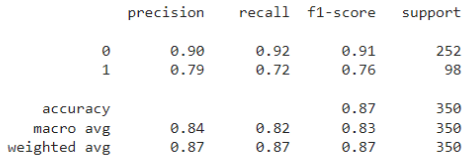


**Figure 24:** Confusion Matrix

By following the flow, we can see that the results improve.



```
              precision    recall  f1-score   support

           0       0.90      0.92      0.91       252
           1       0.79      0.72      0.76        98

    accuracy                           0.87       350
   macro avg       0.84      0.82      0.83       350
weighted avg       0.87      0.87      0.87       350
```

**Figure 25:** Results

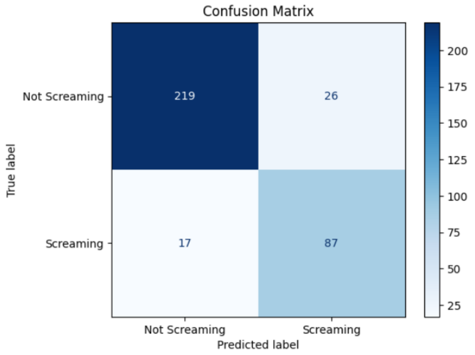


**Figure 26:** Confusion Matrix

## Best Model

Our top-performing model is ResNet. Next, let's review the architecture of this transfer learning model.

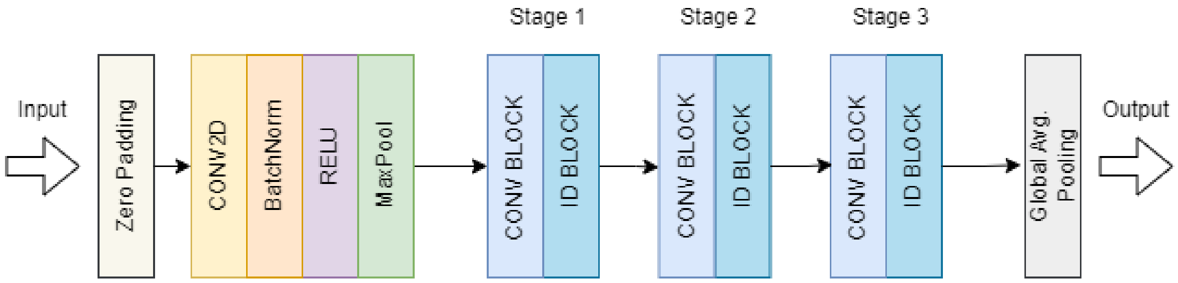


**Figure 27:** ResNet Architecture

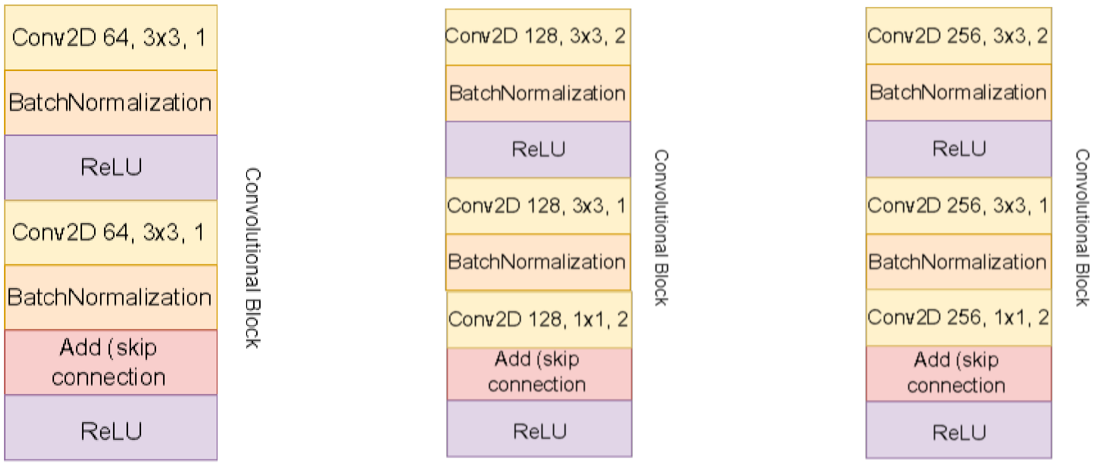


**Figure 28:** Stage 1

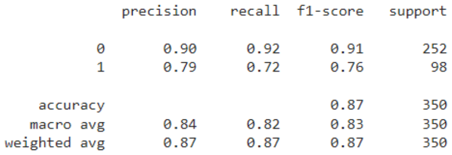
**Figure 29:** Stage 2

**Figure 30:** Stage 3

**Figure 31:** Convolutional Blocks

## Resnet Results

It's worth noting that the F1 scores are very high for both classes, and the number of false positives is exceptionally low.



```
              precision    recall  f1-score   support

           0       0.90      0.92      0.91       252
           1       0.79      0.72      0.76        98

    accuracy                           0.87       350
   macro avg       0.84      0.82      0.83       350
weighted avg       0.87      0.87      0.87       350
```

**Figure 32:** Results

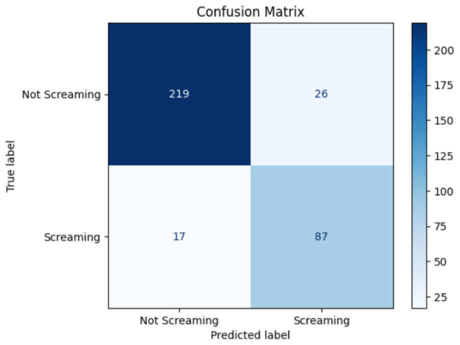


**Figure 33:** Confusion Matrix

# Deloyment

Specifically, we extracted the spectrogram feature with 128 frequency bins, while previously adjusting the audio files to a duration of 10 seconds (padding). We then split the data into training, validation, and test sets in the same proportions as the first dataset, namely 80% for training, 10% for validation, and 10% for testing. The reason we used a second dataset is to ensure that the model is trained correctly and can be effectively applied in different cities around the world. This way, our model can generalize better and recognize screams and non-screams in various urban environments, enhancing its accuracy and reliability in real-world conditions. By employing this approach, the training process becomes more comprehensive, and the model gains greater flexibility in recognizing sound events in diverse geographical contexts. In the training phase, we applied the following strategy to ensure our model Resnet achieves its objective.

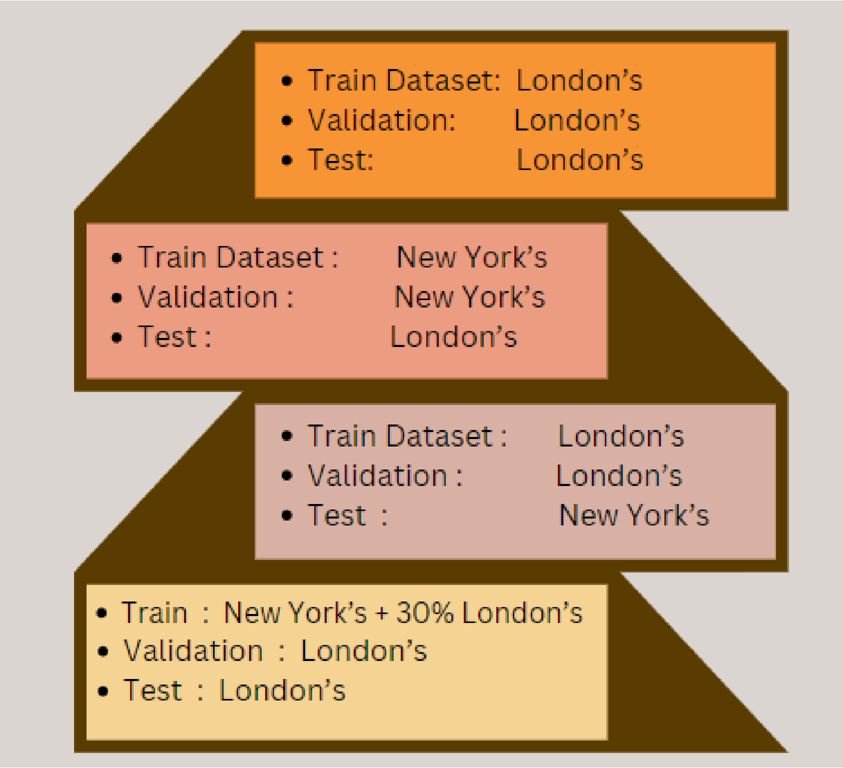
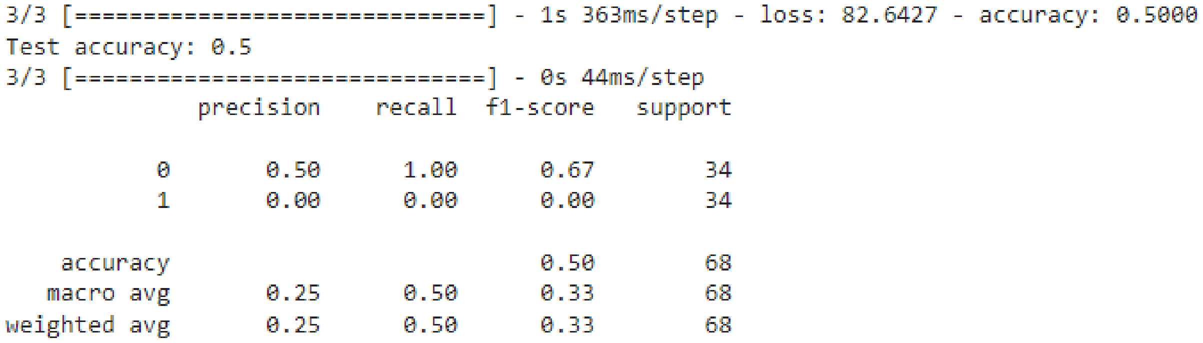

• Train Dataset: London's
• Validation: London's
• Test: London's
• Train Dataset : New York's
• Validation : New York's
• Test : London's
• Train Dataset : London's
• Validation : London's
• Test : New York's
• Train : New York's + 30% London's
• Validation : London's
• Test : London's

**Figure 34:** Strategic of Deployment

1. The results of A' phase :



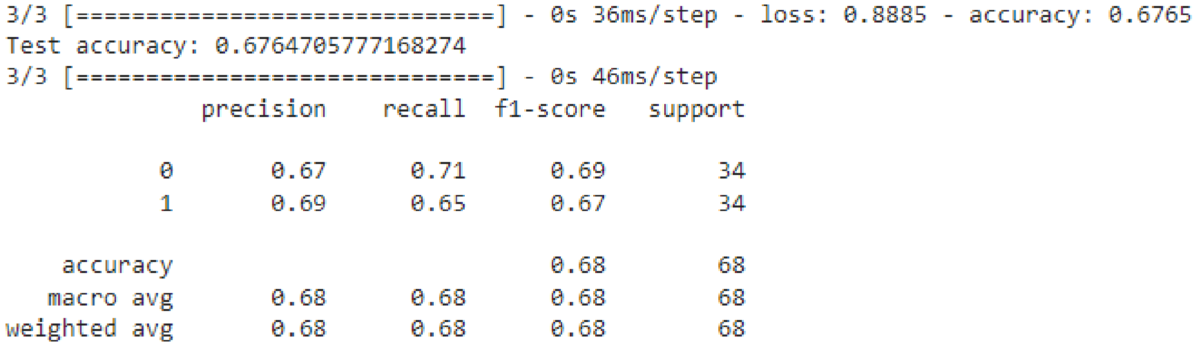
```
3/3 [==============================] - 1s 363ms/step - loss: 82.6427 - accuracy: 0.5000
Test accuracy: 0.5
3/3 [==============================] - 0s 44ms/step
              precision    recall  f1-score   support

           0       0.50      1.00      0.67        34
           1       0.00      0.00      0.00        34

    accuracy                           0.50        68
   macro avg       0.25      0.50      0.33        68
weighted avg       0.25      0.50      0.33        68
```

**Figure 35:** A' phase

2. The results of B' phase:



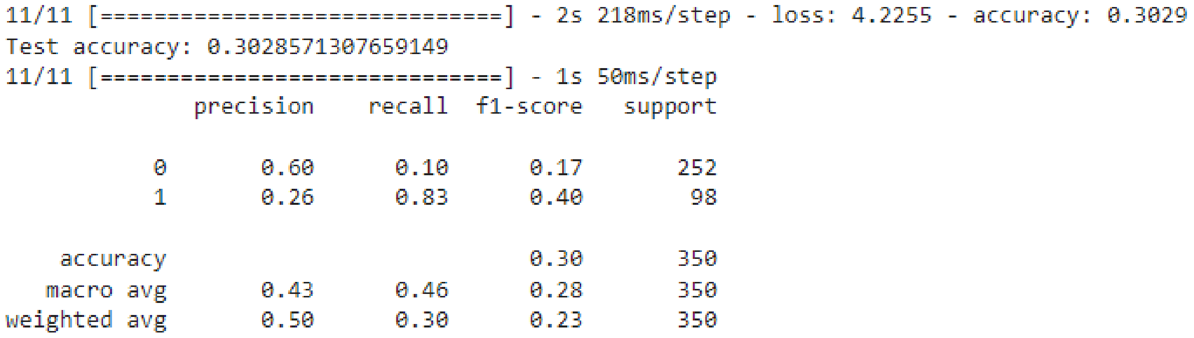
```
3/3 [==============================] - 0s 36ms/step - loss: 0.8885 - accuracy: 0.6765
Test accuracy: 0.6764705777168274
3/3 [==============================] - 0s 46ms/step
              precision    recall  f1-score   support

           0       0.67      0.71      0.69        34
           1       0.69      0.65      0.67        34

    accuracy                           0.68        68
   macro avg       0.68      0.68      0.68        68
weighted avg       0.68      0.68      0.68        68
```

**Figure 36:** B'phase

3. The results of C' phase:



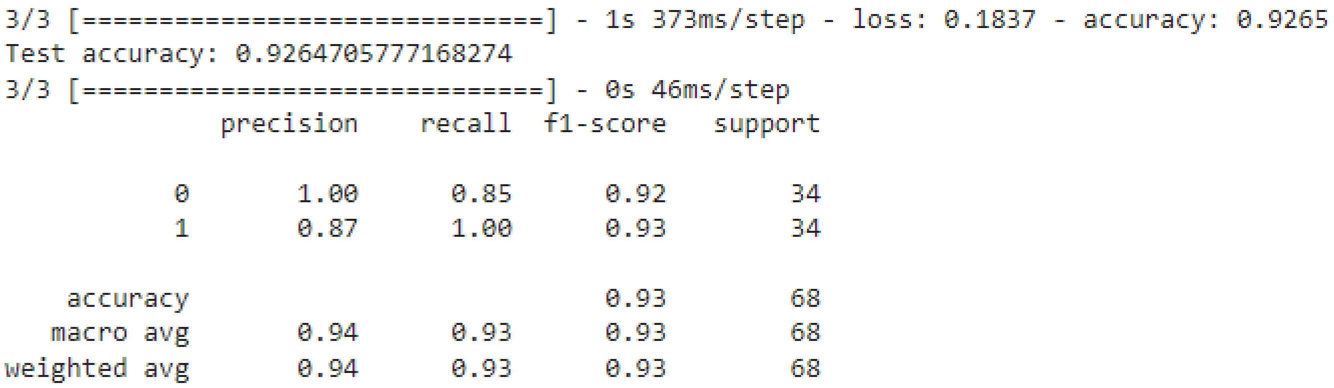
```
11/11 [==============================] - 2s 218ms/step - loss: 4.2255 - accuracy: 0.3029
Test accuracy: 0.3028571307659149
11/11 [==============================] - 1s 50ms/step
              precision    recall  f1-score   support

           0       0.60      0.10      0.17       252
           1       0.26      0.83      0.40        98

    accuracy                           0.30       350
   macro avg       0.43      0.46      0.28       350
weighted avg       0.50      0.30      0.23       350
```

**Figure 37:** C' phase

4. The results of D' phase, Resnet Model gave us the best percentage of Metrics , especially F1-Score:



```
3/3 [==============================] - 1s 373ms/step - loss: 0.1837 - accuracy: 0.9265
Test accuracy: 0.9264705777168274
3/3 [==============================] - 0s 46ms/step
              precision    recall  f1-score   support

           0       1.00      0.85      0.92        34
           1       0.87      1.00      0.93        34

    accuracy                           0.93        68
   macro avg       0.94      0.93      0.93        68
weighted avg       0.94      0.93      0.93        68
```

**Figure 38:** D' phase

We are following a strategic approach to achieve the best possible results with our model, with the ultimate goal of ensuring it generalizes equally well to the London dataset. In each phase of our strategy, we systematically move from one phase to the next. In the final phase of our strategy, we incorporated 30% of the training data from the London dataset into the New York training dataset. This approach led us to achieve f1-scores of 92% and 93% for classes 0 and 1, respectively, which we find very satisfactory. This improvement confirms that incorporating data from the London dataset into the New York training dataset significantly enhanced the model's performance, making it more capable of generalizing to new data. This methodology demonstrates the value of data augmentation strategies in increasing the accuracy and generalizability of machine learning models.

By following these steps and incorporating ResNet into the project flowchart, you can ensure that the model generalizes well and meets the project's requirements.

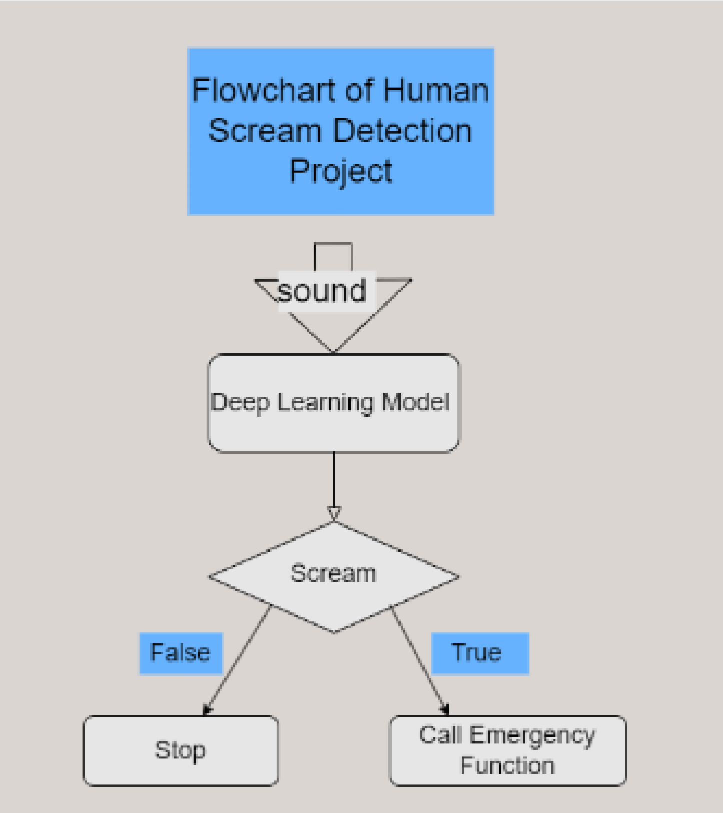


**Figure 39:** Flowchart