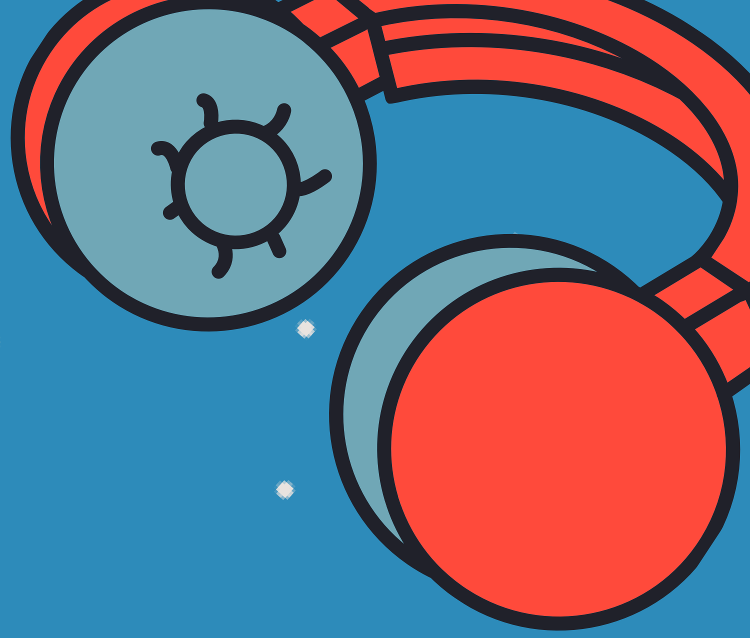
FOUNDATIONS OF DEEP LEARNING

TUT ACOUSTIC SCENES 201

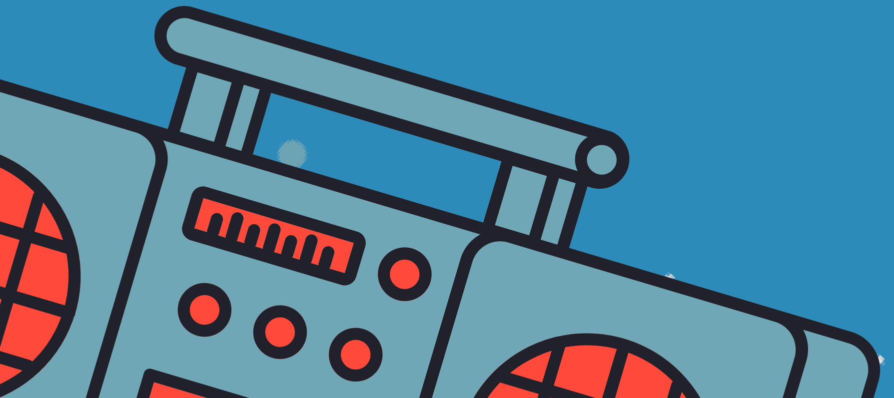
Matteo Mondini 902873 - Patrick Costa 858156



PRESENTATION OUTLINE

OUR DISCUSSION POINTS

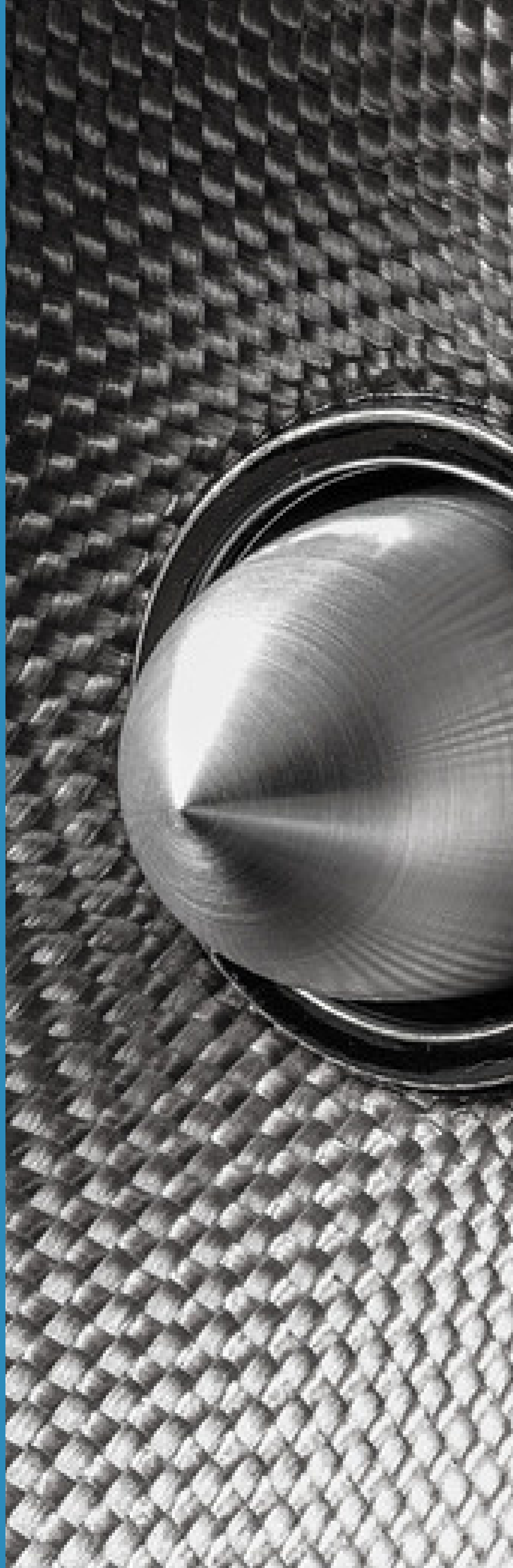
- Problem this project try to solve
- Analysis of the available data
- Solutions
- Positive and negative results of the solutions
- Conclusions



OBJECTIVE

SOUND CLASSIFICATION

The primary objective of this project is to ensure the accurate classification of 4680 audio files that capture ambient sound originating from 15 distinct acoustic scenes. . We tried different types of spectrogram representations, different augmentation approaches as well as different networks in order to find the best combination



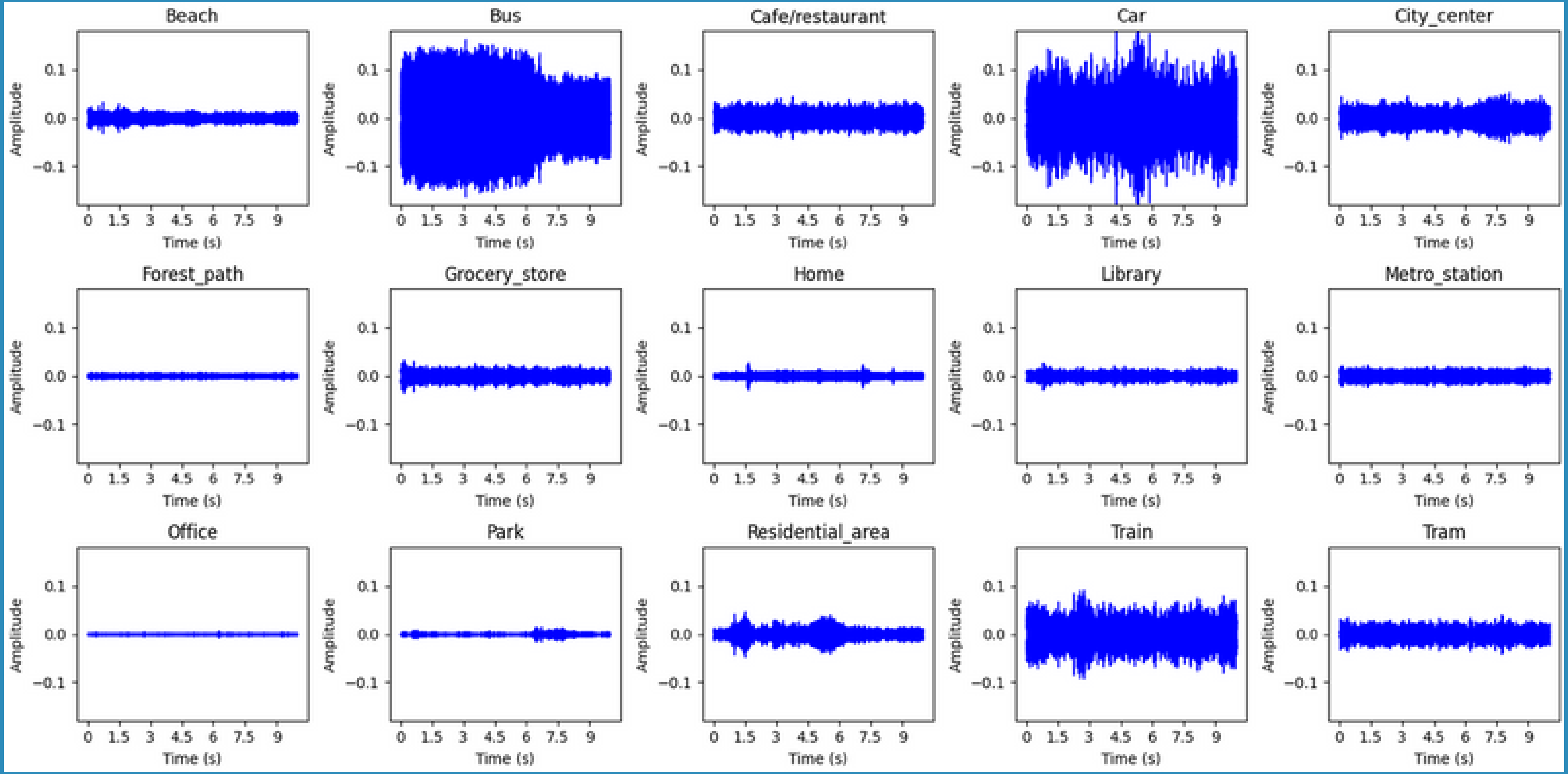
Analysis of the available data

Between June 2015 and January 2016, Tampere University of Technology collected data in Finland. The dataset contains 4680 audio tracks, each lasting 10 seconds, from 15 different scenes like trains, buses, beaches, libraries, and homes. The dataset is **perfectly-balanced**, with each scene having 312 segments, resulting in a total of 52 minutes of audio. All the recordings were taken from different locations. The dataset's size is 10.7 GB.

Instances	4680
Classes	15
Instances per class	312

Class	Count
beach	312
bus	312
cafe/restaurant	312
car	312
city_center	312
forest_path	312
grocery_store	312
home	312
library	312
metro_station	312
office	312
park	312
residential_area	312
train	312
tram	312

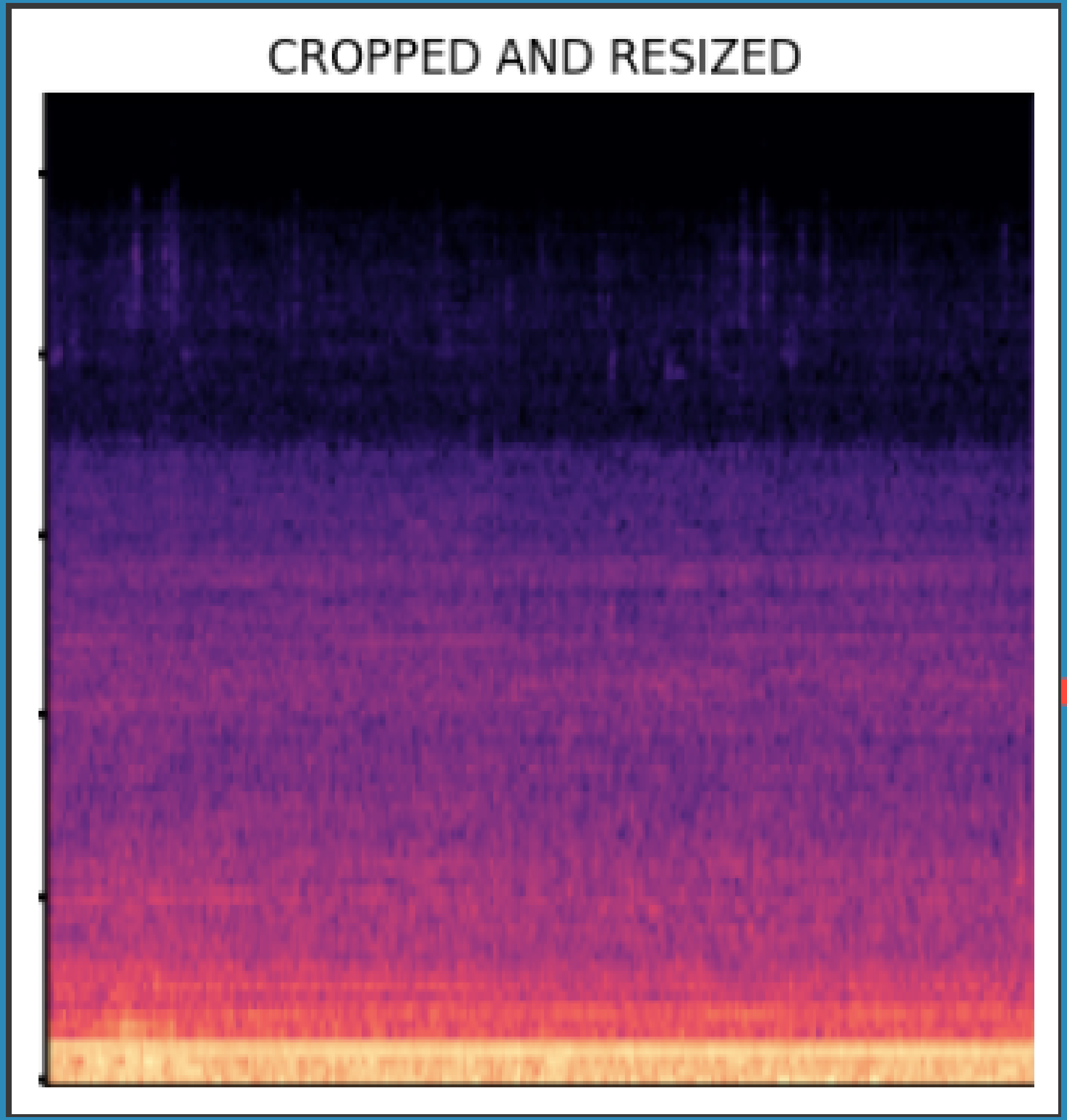
EXPLORING VISUAL REPRESENTATIONS OF AUDIO



AUDIO PREPROCESSING

Spectrograms: each audio has been transformed in spectograms. Two kinds of spectrograms have been generated using librosa library, STFTs (Short-Time Fourier Transforms) and MEL.

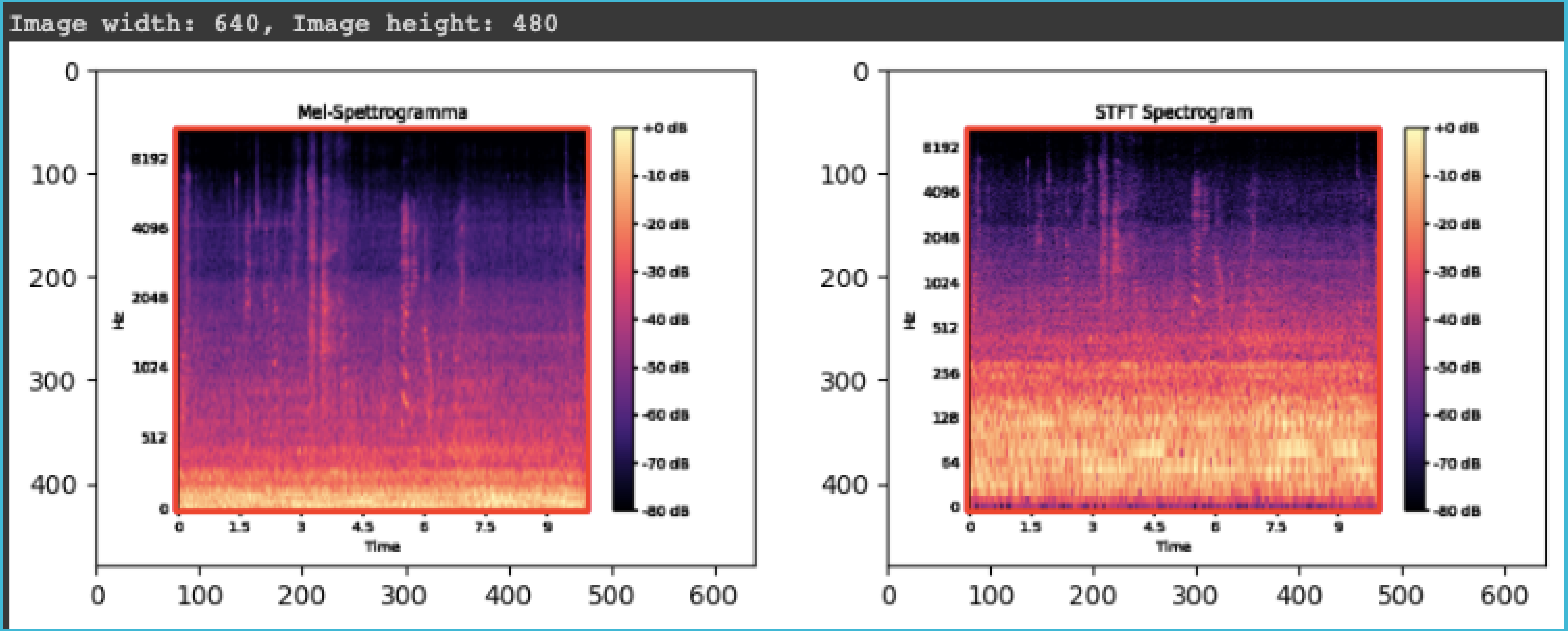
Cropping: cropping the image allows the model to focus on the most relevant features or regions,   improving the model's ability to learn and make accurate predictions. Cropping can also help reduce computational requirements by reducing the input image size without losing critical information.



STFTS AND MEL

STFTs: The process involves utilizing a Fourier-based method to analyze local segments of a signal and determine their sinusoidal frequency and phase components as they vary over time. This is accomplished by dividing the signal into small windows and applying the Fourier Transform (FT) to each window. As a result, a spectrum is obtained for each segment. Finally, the spectra are plotted as a function of time to visualize the temporal changes.

MEL: Audio signals are split into short frames and Fast Fourier Transform (FFT) is applied to each, converting time-domain signals to frequency-domain. These are mapped onto the Mel scale, more aligned with human auditory perception. A filter bank applies overlapping triangular filters, transforming frequency content into Mel frequency. The results are logarithmically scaled, compressing the dynamic range. The final Mel spectrogram is a 2D representation of time, Mel-scaled frequency, and frequency magnitude.

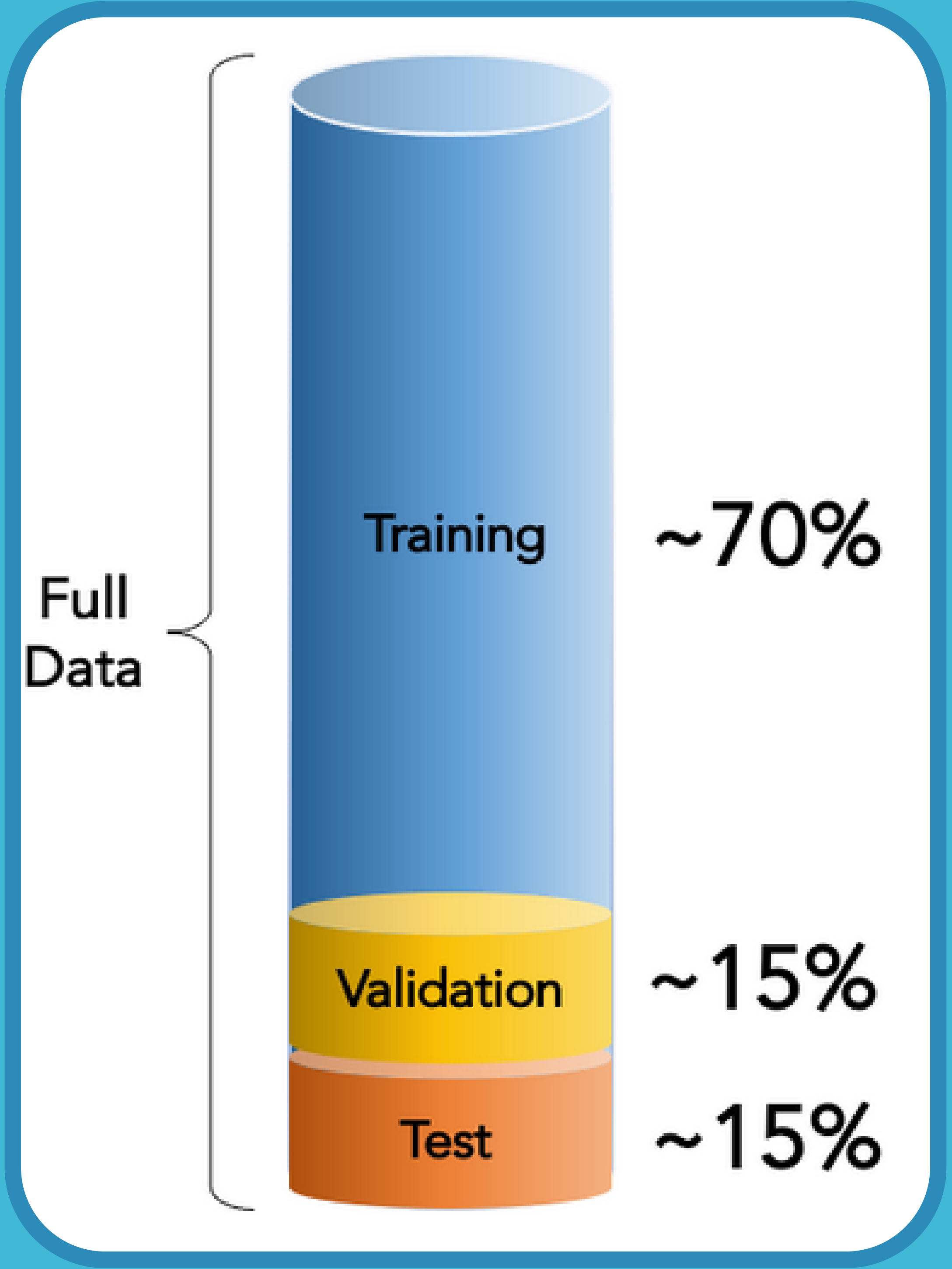


EXAMPLE OF MEL SPECTOGRAM FOR THE CLASS CAR

EXAMPLE OF STFT SPECTOGRAM FOR THE CLASS CAR

TRAINING SET & TEST SET

- **Training set**: 70% of the total images(3276 immages). This is used for training the model.
- **Validation set**: 15% of the total images(702 immages). This is used for tuning model parameters and for early stopping to prevent overfitting.
- **Test set**: 15% of the total images(702 immages). This set is used for evaluating the final model.



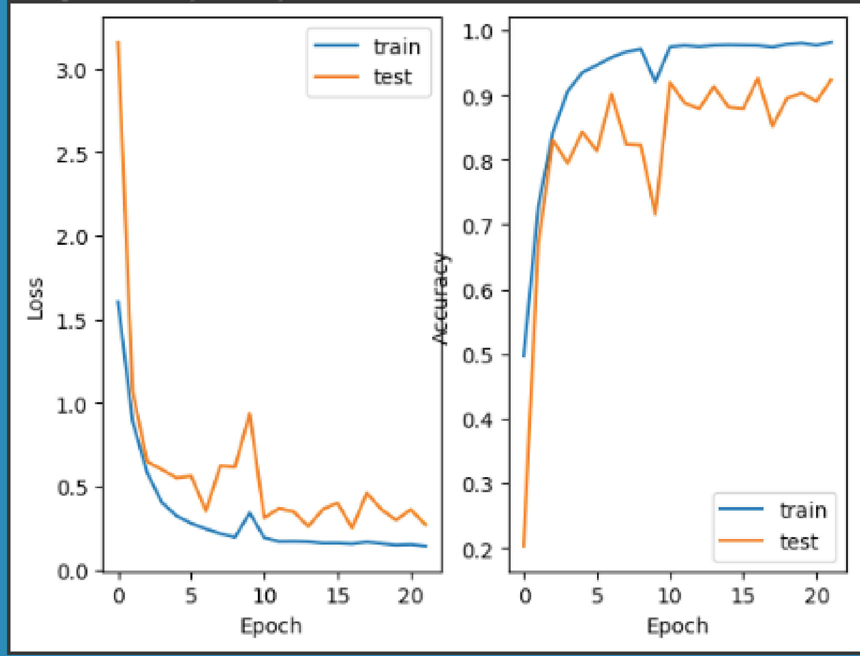
MODELS

To classify the spectrograms, we employed a range of models with increasing complexity, including convolutional neural networks (CNNs), convolutional recurrent neural networks (CNN+RNN), and 2 pretrained models(restnet50 and InceptionV3). For all these models, we utilized the Adam optimizer, set the number of epochs to 50, and implemented early stopping with a patience of 5 for the loss. The loss function employed throughout the classification process was categorical cross-entropy.

FIRST MODEL PROPOSED: SIMPLE CNN

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 256, 256, 16)      448

 batch_normalization (BatchN  (None, 256, 256, 16)     64
 ormalization)

 activation (Activation)     (None, 256, 256, 16)      0

 max_pooling2d (MaxPooling2D  (None, 128, 128, 16)     0
 )

 conv2d_1 (Conv2D)           (None, 128, 128, 32)      4640

 batch_normalization_1 (Batc  (None, 128, 128, 32)     128
 hNormalization)

 activation_1 (Activation)   (None, 128, 128, 32)      0

 max_pooling2d_1 (MaxPooling  (None, 64, 64, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 64, 64, 64)        18496

 batch_normalization_2 (Batc  (None, 64, 64, 64)       256
 hNormalization)

 activation_2 (Activation)   (None, 64, 64, 64)        0

 max_pooling2d_2 (MaxPooling  (None, 32, 32, 64)       0
 2D)

 conv2d_3 (Conv2D)           (None, 32, 32, 128)       73856

 batch_normalization_3 (Batc  (None, 32, 32, 128)      512
 hNormalization)

 activation_3 (Activation)   (None, 32, 32, 128)       0

 max_pooling2d_3 (MaxPooling  (None, 16, 16, 128)      0
 2D)

 conv2d_4 (Conv2D)           (None, 16, 16, 256)       295168

 batch_normalization_4 (Batc  (None, 16, 16, 256)      1024
 hNormalization)

 activation_4 (Activation)   (None, 16, 16, 256)       0

 flatten (Flatten)           (None, 65536)             0

 dense (Dense)               (None, 512)               33554944

 batch_normalization_5 (Batc  (None, 512)              2048
 hNormalization)

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 15)                7695

=================================================================
Total params: 33,959,279
Trainable params: 33,957,263
Non-trainable params: 2,016
_________________________________________________________________
```

```
Test accuracy: 89.32%
<matplotlib.legend.Legend at 0x790c9c3c8fd0>
```



SECOND MODEL PROPOSED: A MORE COMPLEX CNN

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_27 (Conv2D)          (None, 256, 256, 32)      896

 batch_normalization_22 (Bat  (None, 256, 256, 32)     128
 chNormalization)

 activation_12 (Activation)  (None, 256, 256, 32)      0

 max_pooling2d_26 (MaxPoolin  (None, 128, 128, 32)     0
 g2D)

 conv2d_28 (Conv2D)          (None, 128, 128, 64)      18496

 batch_normalization_23 (Bat  (None, 128, 128, 64)     256
 chNormalization)

 activation_13 (Activation)  (None, 128, 128, 64)      0

 max_pooling2d_27 (MaxPoolin  (None, 64, 64, 64)       0
 g2D)

 conv2d_29 (Conv2D)          (None, 64, 64, 128)       73856

 batch_normalization_24 (Bat  (None, 64, 64, 128)      512
 chNormalization)

 activation_14 (Activation)  (None, 64, 64, 128)       0

 max_pooling2d_28 (MaxPoolin  (None, 32, 32, 128)      0
 g2D)

 conv2d_30 (Conv2D)          (None, 32, 32, 256)       295168

 batch_normalization_25 (Bat  (None, 32, 32, 256)      1024
 chNormalization)

 activation_15 (Activation)  (None, 32, 32, 256)       0

 max_pooling2d_29 (MaxPoolin  (None, 16, 16, 256)      0
 g2D)

 conv2d_31 (Conv2D)          (None, 16, 16, 512)       1180160

 batch_normalization_26 (Bat  (None, 16, 16, 512)      2048
 chNormalization)

 activation_16 (Activation)  (None, 16, 16, 512)       0

 max_pooling2d_30 (MaxPoolin  (None, 8, 8, 512)        0
 g2D)

 conv2d_32 (Conv2D)          (None, 8, 8, 512)         2359808

 batch_normalization_27 (Bat  (None, 8, 8, 512)        2048
 chNormalization)

 activation_17 (Activation)  (None, 8, 8, 512)         0

 max_pooling2d_31 (MaxPoolin  (None, 4, 4, 512)        0
 g2D)

 conv2d_33 (Conv2D)          (None, 4, 4, 1024)        4719616

 batch_normalization_28 (Bat  (None, 4, 4, 1024)       4096
 chNormalization)

 activation_18 (Activation)  (None, 4, 4, 1024)        0

 max_pooling2d_32 (MaxPoolin  (None, 2, 2, 1024)       0
 g2D)

 conv2d_34 (Conv2D)          (None, 2, 2, 1024)        9438208

 batch_normalization_29 (Bat  (None, 2, 2, 1024)       4096
 chNormalization)

 activation_19 (Activation)  (None, 2, 2, 1024)        0

 max_pooling2d_33 (MaxPoolin  (None, 1, 1, 1024)       0
 g2D)

 flatten_2 (Flatten)         (None, 1024)              0

 dense_18 (Dense)            (None, 2048)              2099200

 batch_normalization_30 (Bat  (None, 2048)             8192
 chNormalization)

 activation_20 (Activation)  (None, 2048)              0

 dropout_13 (Dropout)        (None, 2048)              0

 dense_19 (Dense)            (None, 2048)              4196352

 batch_normalization_31 (Bat  (None, 2048)             8192
 chNormalization)

 activation_21 (Activation)  (None, 2048)              0

 dropout_14 (Dropout)        (None, 2048)              0

 dense_20 (Dense)            (None, 15)                30735

=================================================================
Total params: 24,443,087
Trainable params: 24,427,791
Non-trainable params: 15,296
```

THIRD MODEL PROPOSED: CONVOLUTIONAL RECURRENT NEURAL NETWORKS (CNN+RNN)

```
Model: "sequential_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_46 (Conv2D)          (None, 256, 256, 64)      1792

 max_pooling2d_45 (MaxPoolin  (None, 128, 128, 64)     0
 g2D)

 conv2d_47 (Conv2D)          (None, 128, 128, 128)     73856

 max_pooling2d_46 (MaxPoolin  (None, 64, 64, 128)      0
 g2D)

 conv2d_48 (Conv2D)          (None, 64, 64, 256)       295168

 max_pooling2d_47 (MaxPoolin  (None, 32, 32, 256)      0
 g2D)

 reshape_7 (Reshape)         (None, 1024, 256)         0

 lstm_9 (LSTM)               (None, 128)               197120

 dense_26 (Dense)            (None, 512)               66048

 dropout_18 (Dropout)        (None, 512)               0

 dense_27 (Dense)            (None, 15)                7695

=================================================================
Total params: 641,679
Trainable params: 641,679
Non-trainable params: 0
_________________________________________________________________
```

PRE-TRAINED MODELS: INCEPTIONV3 AND RESTNET50

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_v3 (Functional)   (None, 6, 6, 2048)        21802784

 global_average_pooling2d_5   (None, 2048)             0
 (GlobalAveragePooling2D)

 dense_10 (Dense)            (None, 512)               1049088

 batch_normalization_569 (Ba  (None, 512)              2048
 tchNormalization)

 dropout_5 (Dropout)         (None, 512)               0

 dense_11 (Dense)            (None, 15)                7695

=================================================================
Total params: 22,861,615
Trainable params: 22,826,159
Non-trainable params: 35,456
_________________________________________________________________
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 8, 8, 2048)        23587712

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 512)               1049088

 batch_normalization (BatchN  (None, 512)              2048
 ormalization)

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 15)                7695

=================================================================
Total params: 24,646,543
Trainable params: 24,592,399
Non-trainable params: 54,144
_________________________________________________________________
```

DATA AUGMENTATION

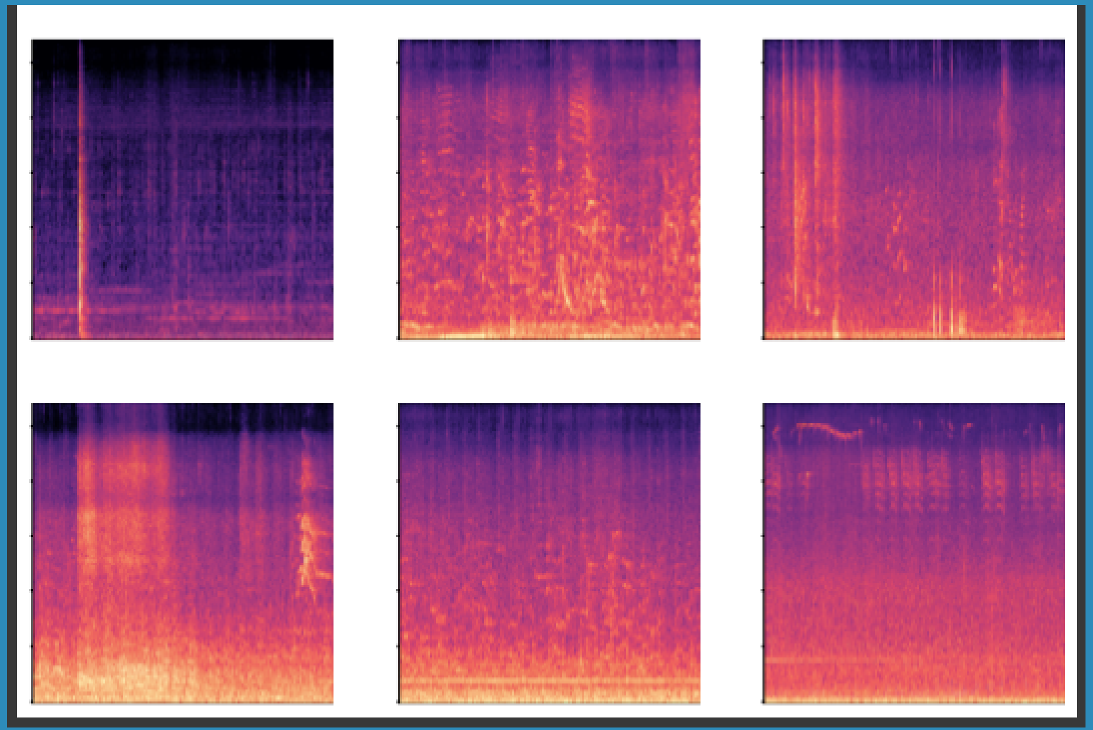
Data augmentation refers to the process of artificially increasing the diversity and quantity of training data by applying various transformations or modifications to existing samples. The main advantages of this process are :

- **increased robustness:** augmentation simulates different recording conditions, improving the model's accuracy in real-world audio classification scenarios.
- **Improved generalization**: augmentation diversifies training data, helping the model perform well on unseen or varied audio samples.
- **Mitigation of overfitting**: augmentation reduces overfitting by providing the model with a wider range of augmented spectrograms, avoiding reliance on specific patterns in the original data.

IN THE PROJECT 2 TECHNIQUES HAVE BEEN APPLIED:

Mixup:

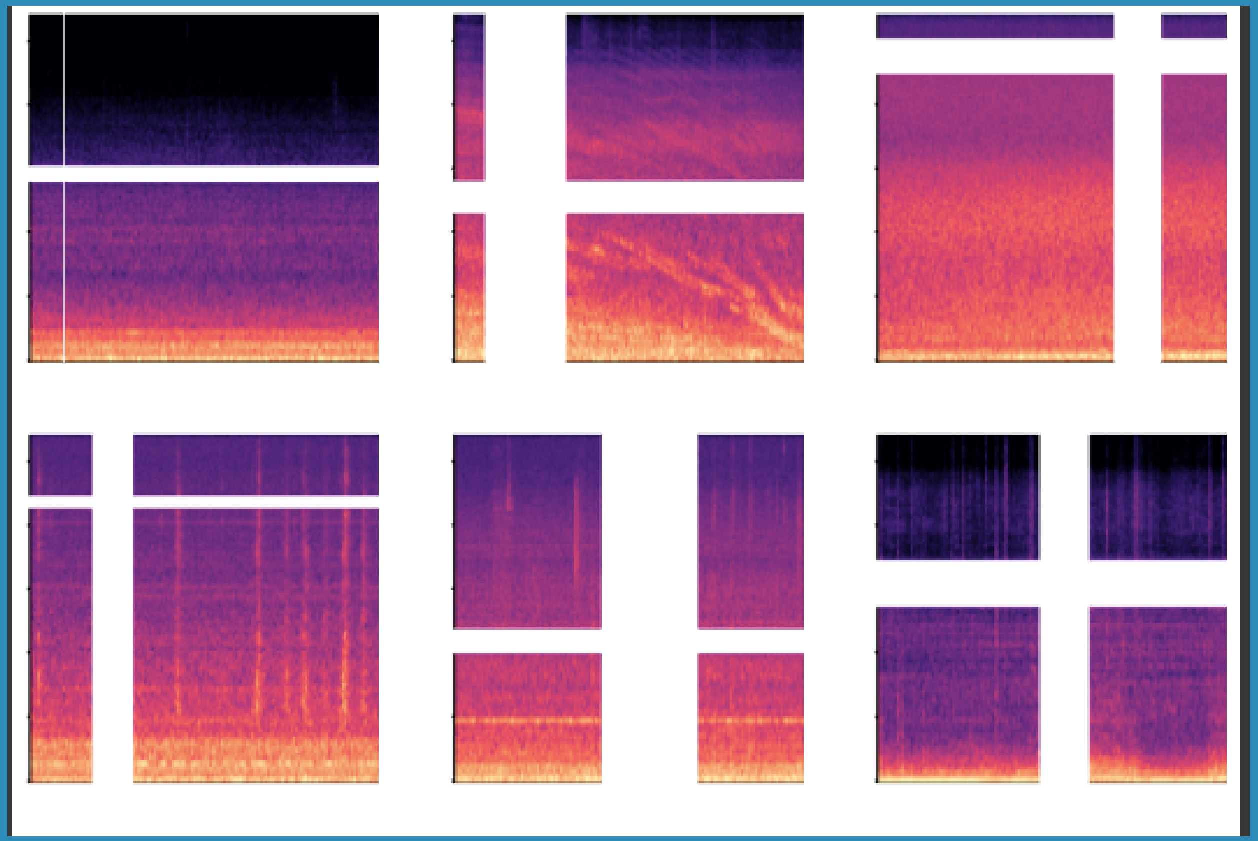
- Combines pairs of samples and labels.
- Generates synthetic samples by linearly interpolating between the samples and labels. Interpolation controlled by a parameter (lambda/λ) sampled from a gamma distribution.
- Encourages the model to learn from convex combinations of samples.



```
[0.0, 0.37129393219947815, 0.0, 0.0, 0.0, 0.0, 0.0, 0.6287060976028442, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.03998817503452301, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.9600118398666382, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.9995344877243042, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0004654930380638689, 0.0]
[0.0, 0.0, 0.0, 0.17841988801956177, 0.0, 0.0, 0.0, 0.8215801119804382, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.9572190046310425, 0.0, 0.0, 0.0, 0.0, 0.0, 0.04278099536895752, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.9999344944953918, 6.55055046081543e-05, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.9360079169273376, 0.0, 0.0, 0.0, 0.0, 0.06399208307266235]
[0.0, 0.0, 0.0, 0.0006569027900695801, 0.0, 0.0, 0.0, 0.0, 0.0, 0.9993430972099304, 0.0, 0.0, 0.0, 0.0, 0.0]
```

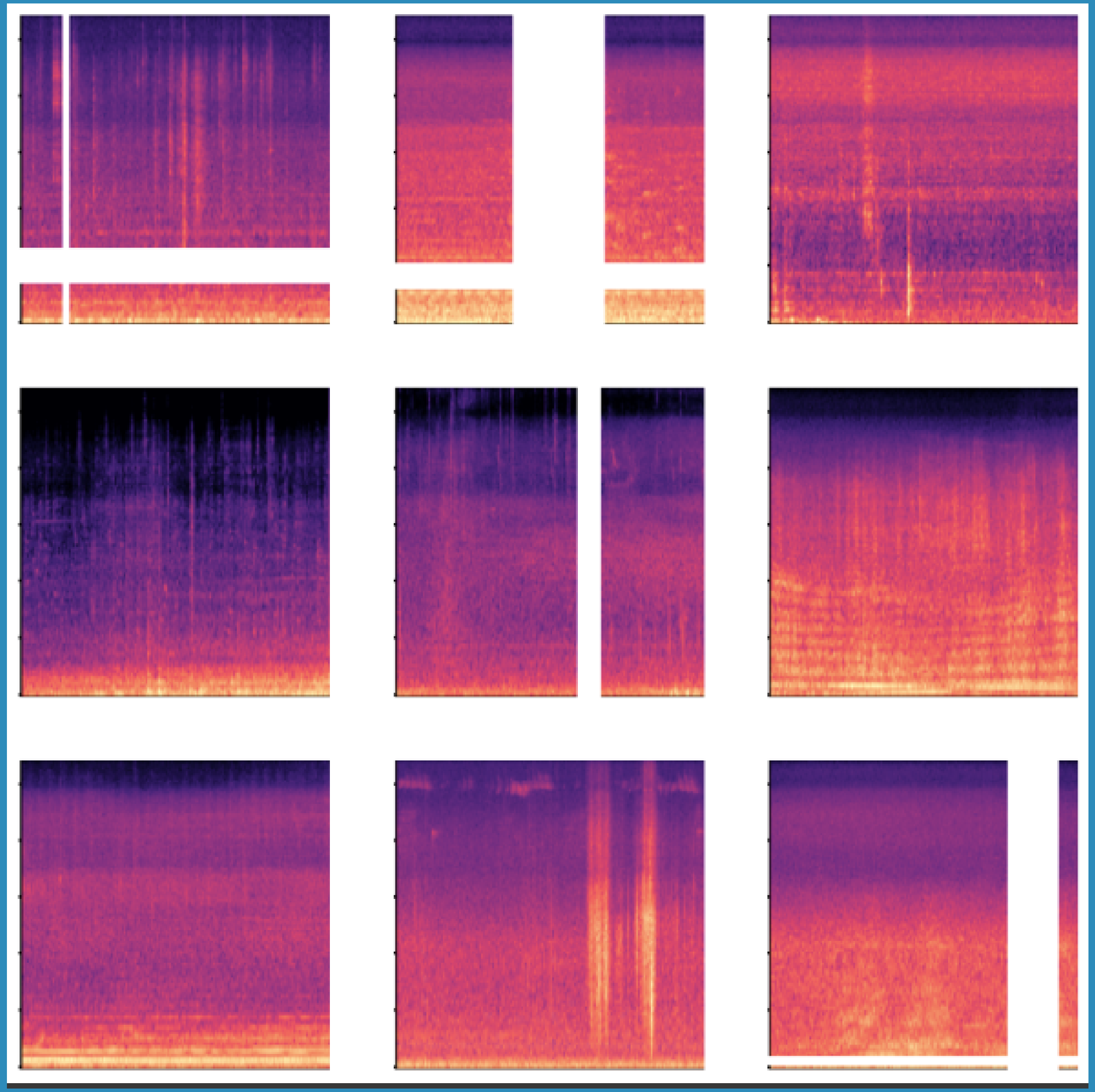
SpecAugment:

- Creates frequency and time maksing randomly on spectrograms
- Enhances model's robustness to signal variations and background noise



```
[0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0]
```

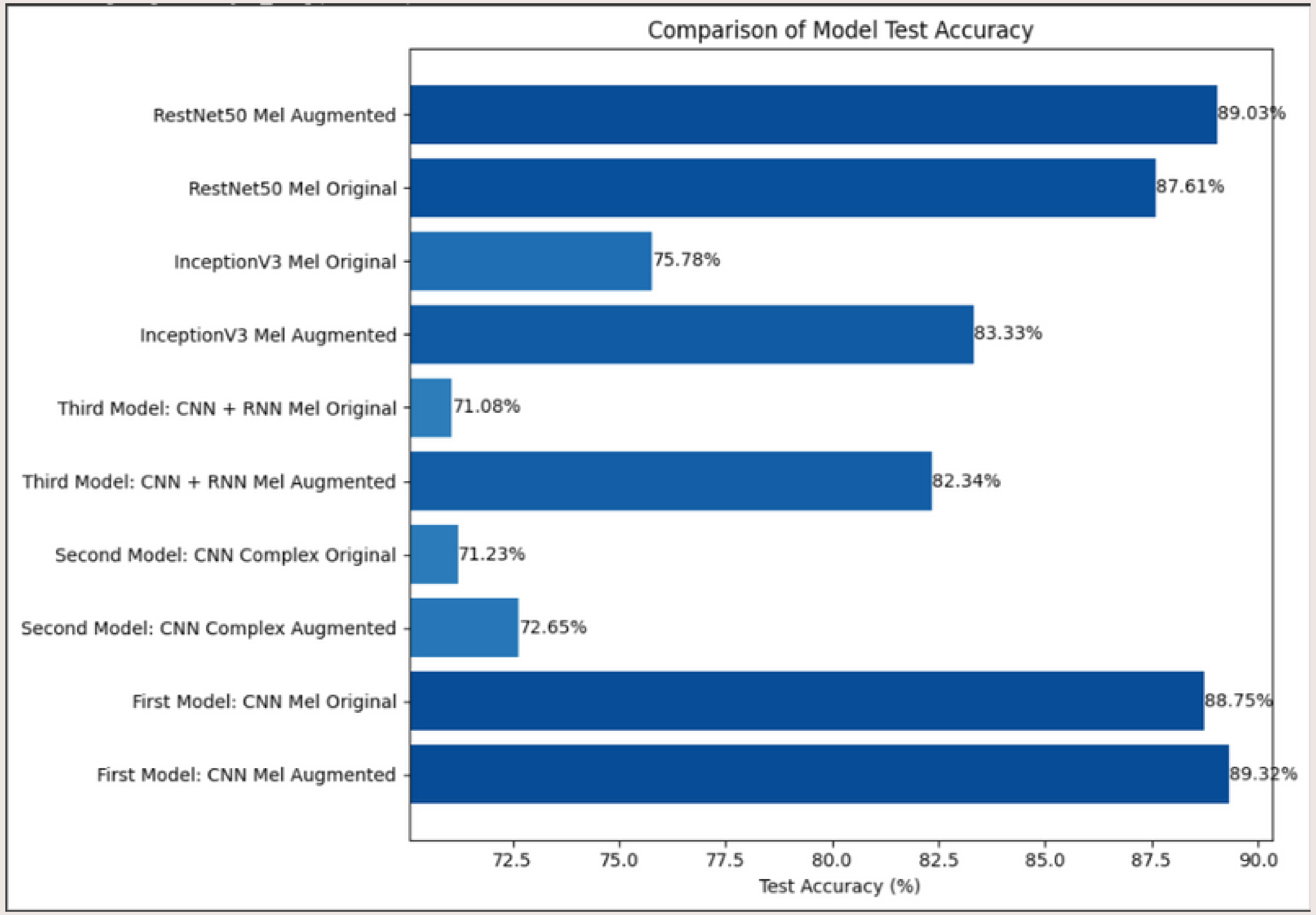
OUR STRATEGY



One-third of the 50% of the images from the original training set were mixed up and added back to the training set. The same procedure was applied to the specAugmented images, as well as to the mix of the two methodologies. As a result, the training set consisted of 300 batches, with each batch containing 16 images.

```
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.9999998807907104, 0.0, 9.822683466609305e-08, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.8964688777923584, 0.10353109985589981, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
[0.0, 0.04235987737774849, 0.0, 0.0, 0.0, 0.9576401114463806, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
```

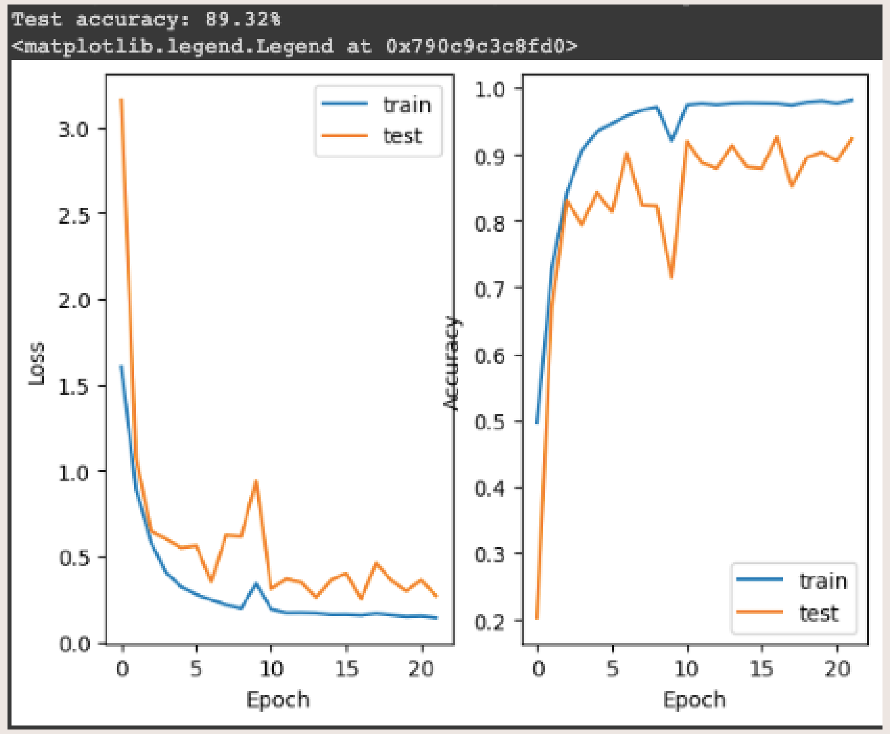
RESULTS



Comparison of performance of every model trained on augmented data and not augmented data

	STFTs	MEL
First model(cnn)	Accuracy: 86.18%	Accuracy:88.75%
Second model(second cnn)	Accuracy: 63.39%	Accuracy: 71.23%

We are just showing two examples, but in most cases, MEL spectrograms have performed better than STFT



accuracy and log graph of the first CNN on mel augmented

CONCLUSION

We employed various models along with diverse data augmentation techniques to analyze the dataset. Eventually, we determined that the most efficient 'from scratch' model was the one with an accuracy of 89%, Considering the results achieved, we can confidently state that the model exhibits good capabilities in classifying environmental audio types.

Future developments

Trying a more advanced data augmentation:
experiment with more sophisticated data augmentation techniques to further enhance the diversity and quality of the training data

Trying different model architecture and approaches :
It's beneficial to explore alternative model architectures and approaches to tackle the problem.

Trying different types of Advanced Optimization Algorithms:
Investigate advanced optimization algorithms