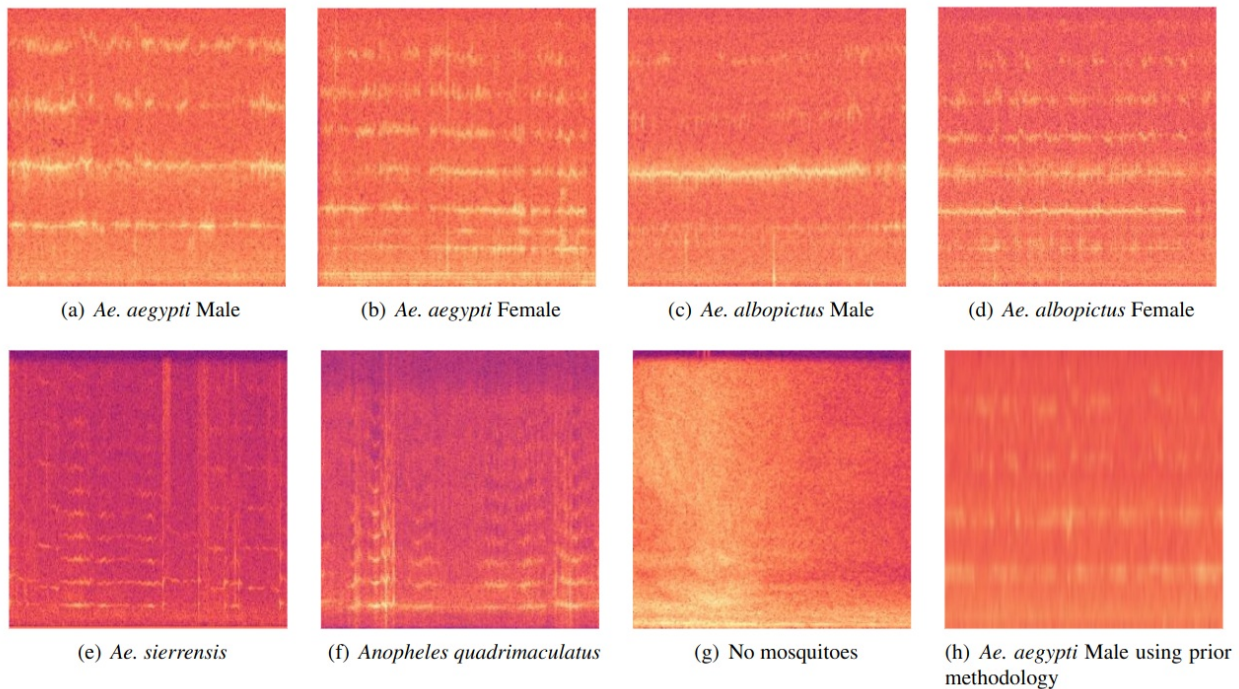


Sound classification - Neural Network Models

This repository contains the implemented code for the classification of mosquito audios using deep neural networks. It includes state-of-the-art algorithms and advanced techniques employed in the study, providing a robust basis for the analysis and categorization of complex acoustic patterns.



[View original publication](#)

The code made available aims to facilitate the replication of the experiments and the application of state-of-the-art methodologies in audio processing and bioacoustics. The implementation contains the definitions of the models, layers, blocks and loss functions necessary for the correct functioning of the models, as well as an evaluation framework that allows the analysis of the models' performance.

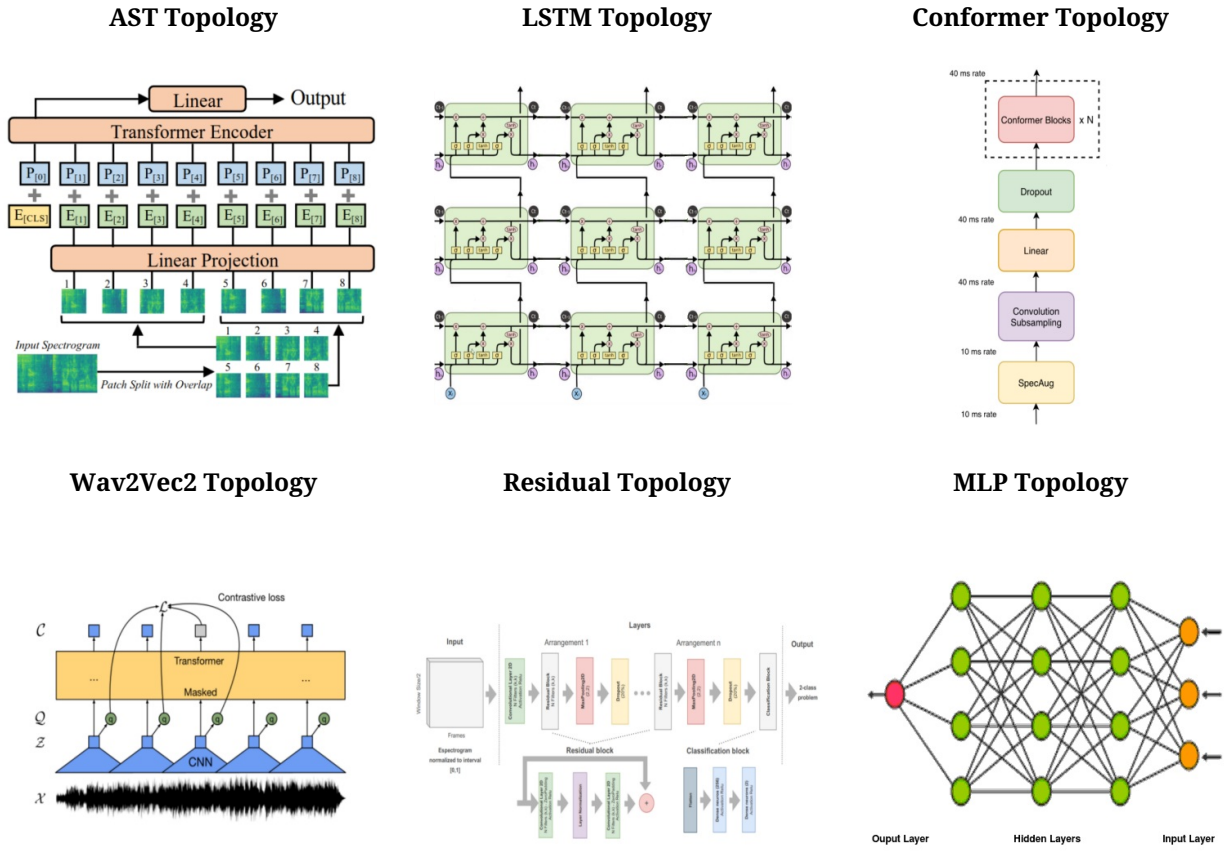
Neural Network Topologies

This repository contains the implementation and evaluation of six distinct deep neural network topologies for audio recognition. Each topology was developed for the analysis and identification of specific acoustic patterns in the audio emitted by mosquito wings, providing a robust technical basis for the comparative evaluation of the proposed solutions. Below are listed each of the topologies present in this repository, as well as their structure and original work.

Original Papers:

1. Audio Spectrogram Transformer [<https://arxiv.org/abs/2104.01778>]
 2. Long Short Term Memory [<https://www.bioinf.jku.at/publications/older/2604.pdf>]
 3. Conformer [<https://arxiv.org/abs/2005.08100>]
 4. Wav2Vec2 [<https://arxiv.org/abs/2006.11477>]
 5. Residual [<https://doi.org/10.1016/j.bspc.2024.106342>]
 6. MLP [<https://ieeexplore.ieee.org/document/8942209>]
-

Models:



Experimental Evaluation

Dataset for Experiments RAW

Description of the datasets used to train and validate the models, as well as the link to obtain them. The table below details the raw dataset obtained.

Dataset RAW

Subset	ID	Species	Gender	Number of Mosquitoes	Number of Recordings	Recording Total Length (s)
No	RD1	No mosquitoes	NA	0	2,000	10,000
Classic dataset (Mukundarajan et al., 2017)	RD2	<i>Ae. aegypti</i>	F/M	1	22	1,736
	RD3	<i>Ae. albopictus</i>	F/M	1	7	966
	RD4	Non-Aedes	NA	1	871	13,237
Novel dataset	RD5	<i>Ae. aegypti</i>	F	1	192	4,607
	RD6	<i>Ae. aegypti</i>	M	1	345	8,279
	RD7	<i>Ae. albopictus</i>	F	1	19	570
	RD8	<i>Ae. albopictus</i>	M	1	17	510
	RD9	Non-Aedes	NA	0	158	4,740

Table 1

Raw audio datasets. NA stands for *Not Available*, F stands for *Female*, and M stands for *Male*.

Dataset for Experiments Processed

Description of the datasets used to train and validate the models, as well as the link to obtain them. The table below details the processed dataset obtained.

Dataset Processed

ID	Description	Raw Datasets	Number of positive <i>Ae. aegypti</i>	Number of negative <i>Ae. aegypti</i>
D1	M <i>Ae. aegypti</i> plus No mosquitoes	{RD1,RD6}	345	2,000
D2	F <i>Ae. aegypti</i> plus No mosquitoes	{RD1,RD5}	192	2,000
D3	F/M <i>Ae. aegypti</i> plus No mosquitoes	{RD1,RD5,RD6}	537	2,000
D4	F/M <i>Ae. aegypti</i> plus Non- <i>Aedes</i> mosquitoes	{RD5,RD6,RD9}	537	158
D5	F/M <i>Ae. aegypti</i> plus F/M <i>Ae. albopictus</i>	{RD3, RD5, RD6, RD7, RD8}	537	43
D6	F/M <i>Ae. aegypti</i> plus No mosquitoes plus Non- <i>Aedes</i> mosquitoes	{RD1-RD5-RD6-RD9}	537	2,158
D7	Original dataset (Mukundarajan et al., 2017)	{RD2-RD3-RD4}	22	878

Table 2
Segmented audio datasets. F stands for *Female* and M stands for *Male*.

Training Parameters

Definition of general parameters used for the evaluation. The parameters were chosen to obtain the fairest possible configuration with all models. The selection process considered various factors to ensure that the evaluation metrics are unbiased and provide an accurate representation of each model's performance under similar conditions. This approach ensures that comparisons between models are valid and meaningful.

Parameters evaluated

Parameter	Description	Evaluated Value
Epochs	Total number of training epochs	[10, 20, 30]
Learning Rate	Learning rate used	[0.1, 0.01, 0.001]
Loss Function	Loss function employed	[Categorical Cross-Entropy]
Optimization Algorithm	Optimization algorithm used	[Adam]
Number of Folds	Number of folds for cross-validation	[10]
Batch Size	Batch size for training	[32]
Sample Rate	Sample rate of sounds	[8000]
Segment Length	Length of sound segment	[40, 60]

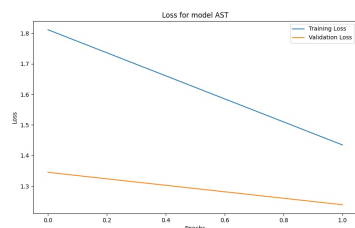
Fitting Analysis

This section is dedicated to the evaluation of models, providing a comprehensive analysis of training curves, confusion matrices, and performance metrics. Through this approach, we ensure a deep understanding of each model's strengths and weaknesses, allowing for continuous adjustments and improvements.

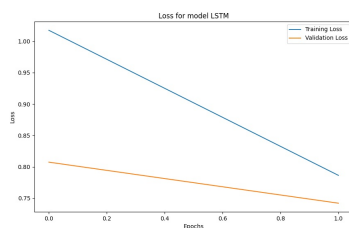
Training Curve

Visualization of the training curves for each of the six model topologies, showing both the training curve and the validation curve. Using cross entropy as a metric, these curves allow a detailed evaluation of the performance of two models and are used to identify possible problems during training, such as overfitting.

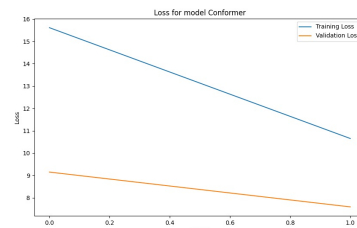
AST Topology



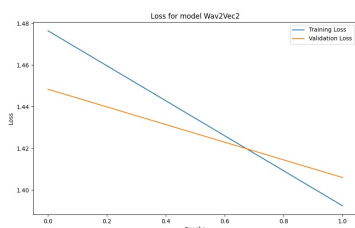
LSTM Topology



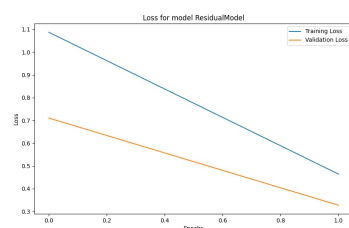
Conformer Topology



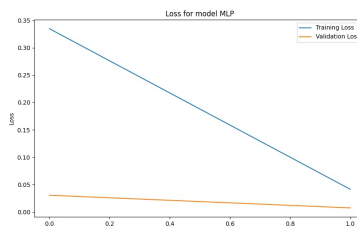
Wav2Vec2



Residual



MLP

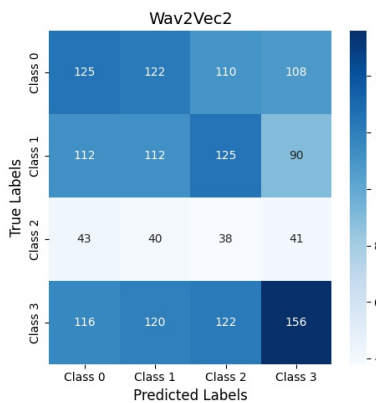


Evaluation Analysis

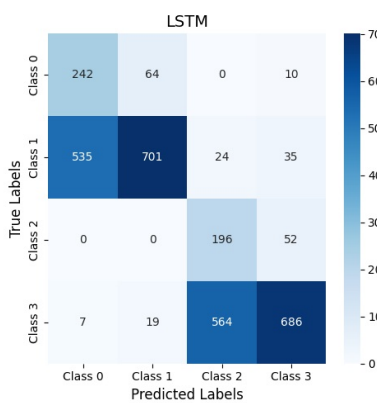
Confusion Matrices

Multiclass confusion matrices for each of the evaluated models. The configurations were defined based on the best configuration found among those evaluated.

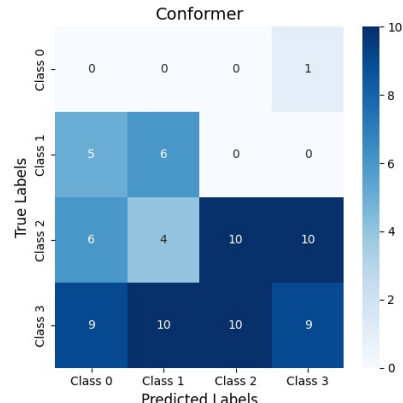
AST Topology



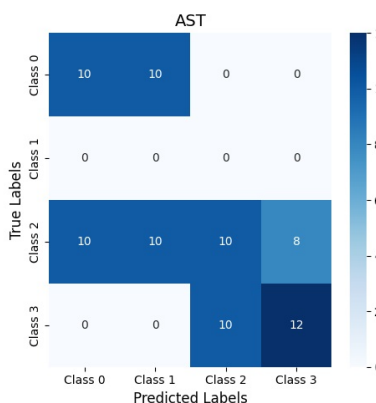
LSTM Topology



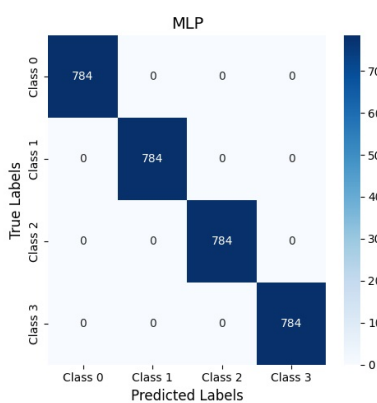
Conformer Topology



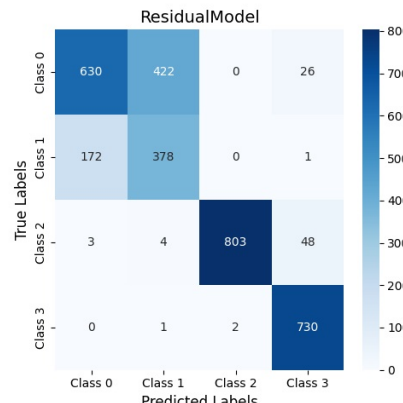
Wav2Vec2 Topology



MLP Topology

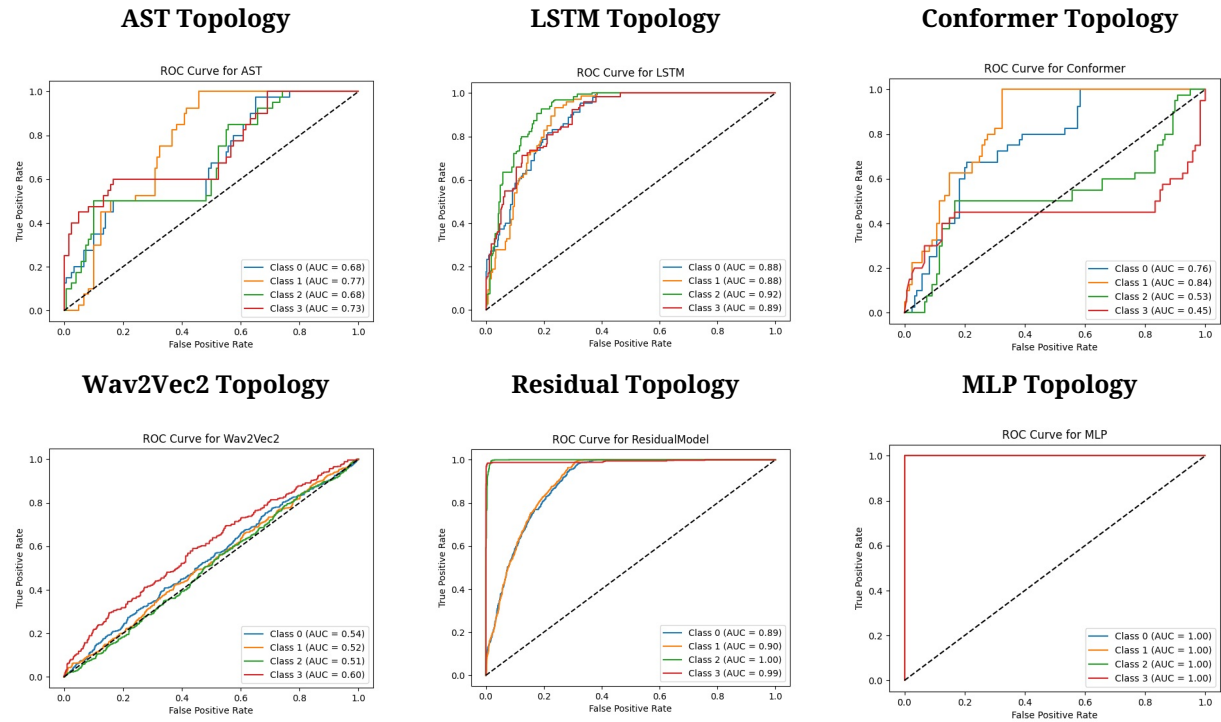


Residual Topology



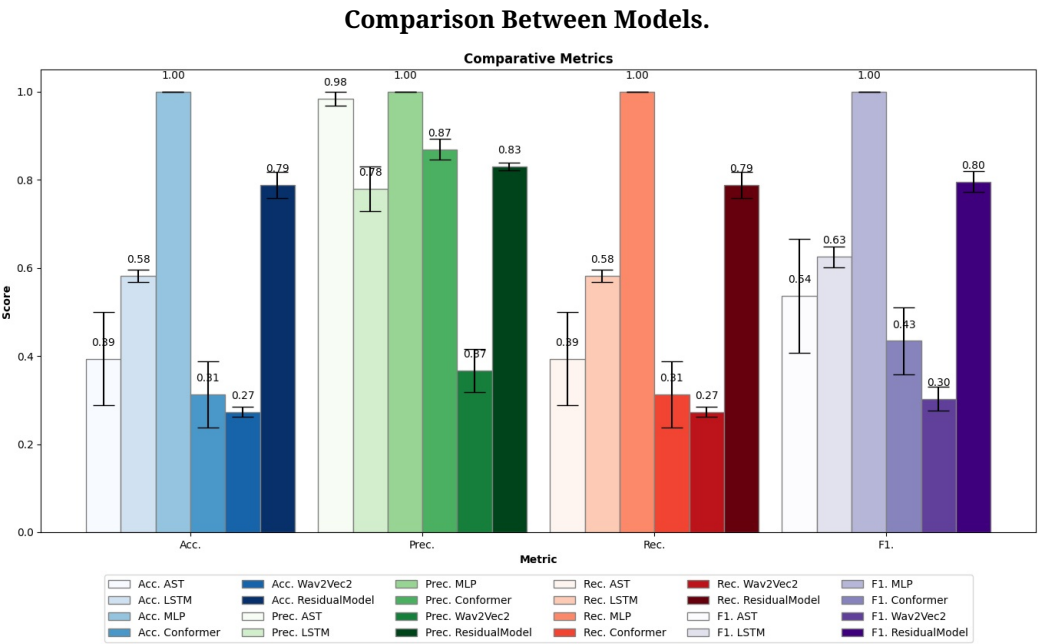
ROC Curve

Visualization of the ROC curves for each of the six model topologies, showing both the training and validation ROC curves. Using the area under the curve (AUC) metric, these curves provide a detailed evaluation of model performance and help identify potential issues during training, such as model generalization capacity.



Comparing our Neural Networks

This comprehensive analysis evaluates the performance of several models by comparing key metrics, including accuracy, precision, recall, and F1-score. These metrics provide insights into each model's ability to correctly classify data, balance false positives and false negatives, and overall performance. The comparison aims to identify the most effective model for the given task.



Steps to Install:

1. Upgrade and update

- `sudo apt-get update`
- `sudo apt-get upgrade`

2. Installation of application and internal dependencies

- `git clone [https://github.com/kayua/ModelsAudioClassification]`
- `pip install -r requirements.txt`

Run experiments:

Run (EvaluationModels.py)

`python3 EvaluationModels.py`

Input parameters:

Arguments:

<code>--dataset_directory</code>	Directory containing the dataset.
<code>--number_epochs</code>	Number of training epochs.
<code>--batch_size</code>	Size of the batches for training.
<code>--number_splits</code>	Number of splits for cross-validation.
<code>--loss</code>	Loss function to use during training.
<code>--sample_rate</code>	Sample rate of the audio files.
<code>--overlap</code>	Overlap for the audio segments.
<code>--number_classes</code>	Number of classes in the dataset.
<code>--output_directory</code>	Directory to save output files.
<code>--plot_width</code>	Width of the plots.
<code>--plot_height</code>	Height of the plots.
<code>--plot_bar_width</code>	Width of the bars in the bar plots.
<code>--plot_cap_size</code>	Capsize of the error bars in the bar plots.

Parameters Audio Spectrogram Transformers:

Arguments:

<code>--ast_projection_dimension</code>	Dimension for projection layer
<code>--ast_head_size</code>	Size of each head in multi-head attention
<code>--ast_number_heads</code>	Number of heads in multi-head attention
<code>--ast_number_blocks</code>	Number of transformer blocks
<code>--ast_hop_length</code>	Hop length for STFT
<code>--ast_size_fft</code>	Size of FFT window
<code>--ast_patch_size</code>	Size of the patches in the spectrogram
<code>--ast_overlap</code>	Overlap between patches in the spectrogram
<code>--ast_dropout</code>	Dropout rate in the network
<code>--ast_intermediary_activation</code>	Activation function for intermediary layers
<code>--ast_loss_function</code>	Loss function to use during training
<code>--ast_last_activation_layer</code>	Activation function for the last layer
<code>--ast_optimizer_function</code>	Optimizer function to use
<code>--ast_normalization_epsilon</code>	Epsilon value for normalization layers
<code>--ast_audio_duration</code>	Duration of each audio clip
<code>--ast_decibel_scale_factor</code>	Scale factor for converting to decibels
<code>--ast_window_size_fft</code>	Size of the FFT window for spectral analysis
<code>--ast_window_size_factor</code>	Factor applied to FFT window size
<code>--ast_number_filters_spectrogram</code>	Number of filters in the spectrogram

Parameters Conformer:

Arguments:

--conformer_input_dimension	Input dimension of the model
--conformer_number_conformer_blocks	Number of conformer blocks
--conformer_embedding_dimension	Dimension of embedding layer
--conformer_number_heads	Number of heads in multi-head attention
--conformer_max_length	Maximum length for positional encoding
--conformer_kernel_size	Kernel size for convolution layers
--conformer_dropout_decay	Dropout decay rate
--conformer_size_kernel	Size of convolution kernel
--conformer_hop_length	Hop length for STFT
--conformer_overlap	Overlap between patches in the spectrogram
--conformer_dropout_rate	Dropout rate in the network
--conformer_window_size	Size of the FFT window
--conformer_decibel_scale_factor	Scale factor for converting to decibels
--conformer_window_size_factor	Factor applied to FFT window size
--conformer_number_filters_spectrogram	Number of filters in the spectrogram
--conformer_last_layer_activation	Activation function for the last layer
--conformer_optimizer_function	Optimizer function to use
--conformer_loss_function	Loss function to use during training

Parameters LSTM:

Arguments:

--lstm_input_dimension	Input dimension of the model
--lstm_list_lstm_cells	List of LSTM cell sizes for each layer
--lstm_hop_length	Hop length for STFT
--lstm_overlap	Overlap between patches in the spectrogram
--lstm_dropout_rate	Dropout rate in the network
--lstm_window_size	Size of the FFT window
--lstm_decibel_scale_factor	Scale factor for converting to decibels
--lstm_window_size_factor	Factor applied to FFT window size
--lstm_last_layer_activation	Activation function for the last layer
--lstm_optimizer_function	Optimizer function to use
--lstm_recurrent_activation	Activation function for LSTM recurrent step
--lstm_intermediary_layer_activation	Activation function for intermediary layers
--lstm_loss_function	Loss function to use during training

Parameters Multilayer Perceptron:

Arguments:

--mlp_input_dimension	Input dimension of the model
--mlp_list_lstm_cells	List of LSTM cell sizes for each layer
--mlp_hop_length	Hop length for STFT
--mlp_overlap	Overlap between patches in the spectrogram
--mlp_dropout_rate	Dropout rate in the network
--mlp_window_size	Size of the FFT window
--mlp_decibel_scale_factor	Scale factor for converting to decibels
--mlp_window_size_factor	Factor applied to FFT window size
--mlp_last_layer_activation	Activation function for the last layer
--mlp_file_extension	File extension for audio files
--mlp_optimizer_function	Optimizer function to use
--mlp_intermediary_layer_activation	Activation function for intermediary layers
--mlp_loss_function	Loss function to use during training

Parameters Residual Model:

Arguments:

--residual_hop_length	Hop length for STFT
--residual_window_size_factor	Factor applied to FFT window size
--residual_number_filters_spectrogram	Number of filters for spectrogram generation
--residual_filters_per_block	Number of filters in each convolutional block
--residual_file_extension	File extension for audio files
--residual_dropout_rate	Dropout rate in the network
--residual_number_layers	Number of convolutional layers
--residual_optimizer_function	Optimizer function to use
--residual_overlap	Overlap between patches in the spectrogram
--residual_loss_function	Loss function to use during training
--residual_decibel_scale_factor	Scale factor for converting to decibels
--residual_convolutional_padding	Padding type for convolutional layers
--residual_input_dimension	Input dimension of the model

--residual_intermediary_activation	Activation function for intermediary layers
--residual_last_layer_activation	Activation function for the last layer
--residual_size_pooling	Size of the pooling layers
--residual_window_size	Size of the FFT window
--residual_size_convolutional_filters	Size of the convolutional filters

Parameters Wav2Vec 2:

Arguments:

--wav_to_vec_input_dimension	Input dimension of the model
--wav_to_vec_number_classes	Number of output classes
--wav_to_vec_number_heads	Number of heads in multi-head attention
--wav_to_vec_key_dimension	Dimensionality of attention key vectors
--wav_to_vec_hop_length	Hop length for STFT
--wav_to_vec_overlap	Overlap between patches in the spectrogram
--wav_to_vec_dropout_rate	Dropout rate in the network
--wav_to_vec_window_size	Size of the FFT window
--wav_to_vec_kernel_size	Size of the convolutional kernel
--wav_to_vec_decibel_scale_factor	Scale factor for converting to decibels
--wav_to_vec_context_dimension	Context dimension for attention mechanisms
--wav_to_vec_projection_mlp_dimension	Dimension of the MLP projection layer
--wav_to_vec_window_size_factor	Factor applied to FFT window size
--wav_to_vec_list_filters_encoder	List of filters for each encoder block
--wav_to_vec_last_layer_activation	Activation function for the last layer
--wav_to_vec_optimizer_function	Optimizer function to use
--wav_to_vec_quantization_bits	Number of quantization bits for the model
--wav_to_vec_intermediary_layer_activation	Activation function for intermediary layers
--wav_to_vec_loss_function	Loss function to use during training

Requirements:

matplotlib 3.4.1 tensorflow 2.4.1 tqdm 4.60.0 numpy 1.18.5

keras 2.4.3 setuptools 45.2.0 h5py 2.10.0