



הפקולטה להנדסה
המעבדה לעיבוד אותות

Multi-label audio classification

סיווג שמע מרובה תוויות

ינאי לוי

אופיר גונן

פרויקט שנה ד' לקראת תואר ראשון בהנדסה

מנחים:

גב' דניאל לוי

אחראי אקדמי:

פרופסור שרון גנות

אוקטובר 2023

Final Project – Multi-label audio classification

Introduction:

In recent years, the field of audio processing and classification has witnessed remarkable advancements, enabling machines to understand and interpret audio content in ways previously thought unimaginable. One of the key challenges in this domain is the problem of multi-label classification for audio data. Unlike traditional single-label classification where each instance is assigned to a single category, multi-label classification involves assigning multiple labels to an instance, reflecting the complex and diverse nature of audio content. This problem has gained significant attention due to its wide range of applications, from music genre classification and environmental sound recognition to speech emotion analysis and audio event detection.

Motivation:

The motivation behind delving into multi-label classification of audio stems from the inherently intricate nature of real-world audio data. Unlike visual or textual data, audio content often comprises a mixture of various sources, making it difficult to assign a single label that encapsulates all its characteristics. Take, for example, a segment of audio from an urban environment - it could contain a combination of sounds such as car horns, footsteps, and sirens. A single-label approach would fail to capture the richness and complexity of this audio snippet.

Moreover, there exists a demand for more holistic and informative audio content analysis. Consider a piece of music that is both energetic and melancholic. Capturing these multiple emotional aspects is crucial for delivering a more nuanced user experience. In the context of healthcare, understanding both the spoken words and the emotional tone in a patient's voice can lead to more accurate diagnostic tools.

Furthermore, the rise of multimedia platforms and the increasing availability of vast audio datasets have fueled the need for advanced audio classification techniques. Efficiently tagging audio with multiple labels not only enhances content recommendation systems but also aids in organizing and retrieving audio files in a more granular and intuitive manner.

As we delve deeper into the complexities of multi-label classification for audio, this project aims to explore and develop methodologies that can accurately decipher the multifaceted nature of audio data. By addressing this challenge, we not only contribute to the field of audio processing but also

open up opportunities to revolutionize various industries that rely on accurate audio content analysis.

Preprocessing:

Before delving into the details of our data preprocessing, it's important to provide a brief overview of STFT, Spectrograms and MelSpectrograms:

- STFT:

- STFT stands for Short-Time Fourier Transform, and it is a widely used mathematical tool in signal processing and audio analysis. STFT is used to analyze how the frequencies of a signal change over time.
- The equation for STFT in the continuous domain:

$$STFT \{x(t)\} = X(\tau, \omega) = \int_{-\infty}^{\infty} x(t) * w(t - \tau) * e^{-j\omega t} dt$$

- The equation for STFT in the discrete domain:

$$X(n, \omega) = \sum_{m=-\infty}^{\infty} x(m) * w(n - m) * e^{-j\omega m}$$

- Here's a breakdown of what STFT is and how it works:
- Fourier Transform:
Before diving into the short-time aspect, let's briefly touch on the Fourier Transform. It's a mathematical technique that takes a time-domain signal (a signal varying with time) and decomposes it into its constituent frequency components. In essence, it converts a signal from the time domain to the frequency domain, revealing the different frequencies that make up the signal.
- Short-Time Aspect:
While the Fourier Transform gives us a complete frequency representation of a signal, it doesn't tell us how the frequencies change over time. This limitation is where the Short-Time Fourier Transform (STFT) comes in.
- Windowing:
In the STFT, the signal is divided into smaller segments or frames, typically overlapping in time. Each frame represents a short duration of the signal. To analyze the frequency content of these frames, a window function is applied to each one. Common window functions include the Hamming window and the Hanning window.

This windowing reduces the spectral leakage that can occur when abruptly cutting a signal into frames.

- Fourier Transform of Frames:

After windowing, the Fourier Transform is applied to each of these frames. This operation transforms each frame from the time domain into the frequency domain, providing a snapshot of the frequency content within that specific time window.

- Time Evolution:

By analyzing the frequency content frame by frame, you can see how the frequencies change over time. This time-varying frequency information can be used for tasks like speech analysis, music analysis, and many other applications in audio and signal processing.

-

- Spectrogram:

- A spectrogram is a visual representation of how the frequencies in a signal change over time. It is a two-dimensional graph that displays how the intensity or amplitude of different frequencies in a signal vary as a function of time. Spectrograms are commonly used in fields such as audio processing, speech analysis, music analysis, and even in some scientific disciplines like physics and geophysics. Here's a breakdown of the key elements of a spectrogram:

- Time vs. Frequency:

The horizontal axis of a spectrogram represents time, typically in seconds. It shows how the signal's characteristics change over time, allowing you to see events like the onset of a sound, its duration, and how it evolves.

The vertical axis represents frequency, typically in Hertz (Hz). It displays the range of frequencies present in the signal, from low-frequency components (e.g., bass) at the bottom to high-frequency components (e.g., treble) at the top.

- Intensity/Amplitude vs. Color:

The intensity or amplitude of each frequency at a given point in time is represented by the color or brightness of that point. Usually, warmer colors (such as red or yellow) indicate higher intensity or amplitude, while cooler colors (like blue) represent lower intensity.

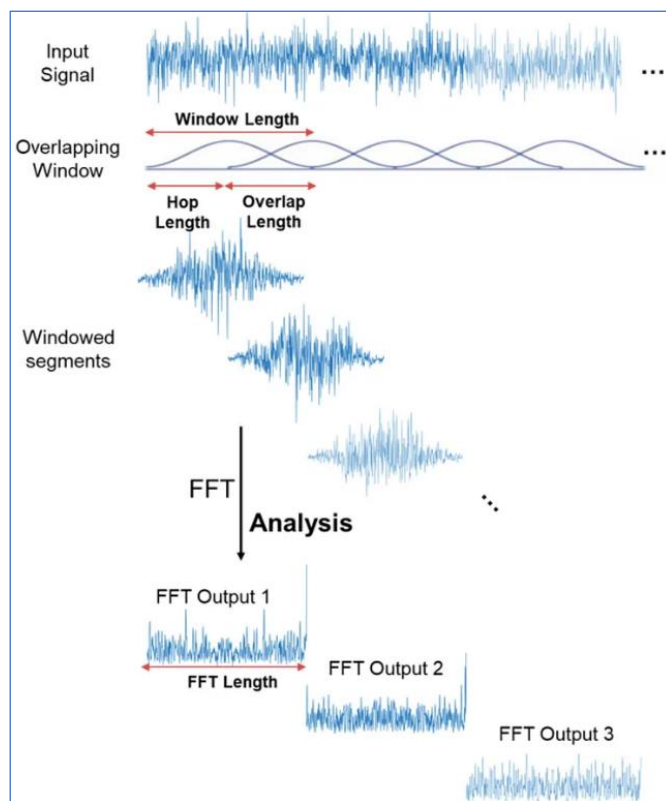
- Frequency Resolution:

The horizontal resolution of a spectrogram depends on the length of the analysis window used to compute the spectrum for each time slice. Shorter windows provide

better time resolution but lower frequency resolution, while longer windows offer better frequency resolution but lower time resolution.

- Windowing and Overlapping:

To create a spectrogram, the input signal is divided into short overlapping segments or "windows". The Fourier Transform is applied to each window to calculate the intensity of different frequencies within that window. The overlapping of windows helps maintain continuity in the spectrogram and captures information about how frequencies change over time. Here is a visual explanation:



- Applications:

Spectrograms are used in various applications. In audio analysis, they can be used to identify musical notes, detect speech phonemes, locate sound events in environmental audio, and more. In scientific fields, they are used to analyze signals from machinery, study seismic activity, and even visualize data from radio telescopes observing celestial bodies

- MelSpectrogram:

- MelSpectrogram, short for Mel-frequency Spectrogram, is a visual representation of the spectrum of frequencies of a signal as they vary with time. It is a widely used tool

in audio processing and analysis. Let's break down the key elements of a MelSpectrogram:

- Spectrogram:
A spectrogram is a way to represent how the frequencies in a signal change over time. It's a two-dimensional graph with time on the x-axis and frequency on the y-axis. The intensity or color at each point in the graph represents the strength or amplitude of a particular frequency component at a given time.
- Mel-Frequency Scaling:
Human perception of sound is not linear but rather logarithmic. The Mel scale is a scale that approximates how humans perceive the difference in pitch or frequency. It's designed to be more aligned with human auditory perception. So, instead of representing frequencies in Hertz (Hz), which is linear, the Mel scale uses Mel (a unit of perceived pitch).
- MelSpectrogram Conversion:
To create a MelSpectrogram, you take an audio signal (which is a time-domain signal) and break it into short overlapping frames. For each frame, you calculate the Fourier Transform to get the frequency components. Then, you apply a filterbank of triangular filters spaced in Mel-scale intervals to these components. The result is a set of Mel-frequency bins, each representing the energy or magnitude of the signal in a particular Mel-frequency range for each time frame.
- Visualization:
The MelSpectrogram is often displayed as an image, with time on the horizontal axis, Mel-frequency bins on the vertical axis, and the intensity of each bin represented by color or grayscale. Typically, warmer colors (like red and yellow) represent higher intensity or energy, while cooler colors (like blue) represent lower intensity.

Our Data for CNN:

The dataset employed for our Convolutional Neural Network (CNN) consists of a combination of two records extracted from the well-established UrbanSound8K Dataset.

This particular dataset encompasses a collection of 8,732 pre-labelled sound excerpts, each lasting no longer than four seconds. These sound excerpts belong to ten distinct urban sound classes, namely: air_conditioner, car_horn, children_playing, dog_bark, drilling, engine_idling, gun_shot,

Ofir Gonen
Yanai Levi

jackhammer, siren, and street music. These categories are drawn from the urban sound taxonomy, reflecting the diverse sounds commonly encountered in urban environments.

Supplementing the sound excerpts, we also utilize a CSV (Comma-Separated Values) file named "UrbanSound8k.csv," which contains essential metadata corresponding to each sound excerpt within the dataset.

UrbanSound8k.csv Metadata:

This file includes comprehensive meta-information for each audio file in the dataset, encompassing the following attributes:

1. **slice_file_name:**

- This denotes the name of the audio file, adhering to the format [fsID]-[classID]-[occurrenceID]-[sliceID].wav, where:
 - [fsID] represents the Freesound ID of the recording from which the excerpt (slice) originates.
 - [classID] denotes a numeric identifier for the sound class (additional details on classID are provided below).
 - [occurrenceID] is a numeric identifier that distinguishes different instances of the sound within the original recording.
 - [sliceID] is a numeric identifier that distinguishes various slices extracted from the same occurrence.

2. **fsID:**

- This attribute signifies the Freesound ID of the recording from which the audio excerpt (slice) is derived.

3. **start:**

- Indicates the start time of the slice within the original Freesound recording.

4. **end:**

- Represents the end time of the slice within the original Freesound recording.

5. **salience:**

- This is a subjective rating of the sound's salience, where 1 denotes foreground sounds, and 2 denotes background sounds.

6. **fold:**

- Denotes the fold number, ranging from 1 to 10, to which the file has been assigned.

7. **classID:**

- Serves as a numeric identifier for the sound class, with the following mapping:
 - 0 = air_conditioner
 - 1 = car_horn
 - 2 = children_playing
 - 3 = dog_bark
 - 4 = drilling
 - 5 = engine_idling
 - 6 = gun_shot
 - 7 = jackhammer
 - 8 = siren
 - 9 = street_music

8. **class:**

- Refers to the class name, including: air_conditioner, car_horn, children_playing, dog_bark, drilling, engine_idling, gun_shot, jackhammer, siren, and street_music.

From this initial dataset, we created our own dataset through the following steps:

1. **Duplication and Shuffling:**

- We replicated each audio file three times to expand the dataset.
- These duplicated files were then randomly shuffled to ensure they were not presented in the original order.

2. **Concatenation:**

- Pairs of audio files were concatenated to form new audio files.

3. **MelSpectrogram Calculation:**

- We computed the MelSpectrogram for each of the newly created audio files.

4. **Addition of Noise:**

- Noise was introduced to the MelSpectrogram.

5. **New Class ID Assignment:**

- The MelSpectrogram, along with its associated class ID, was saved in a new file.
- The new class ID was determined based on the combination of the two original class IDs.

In total, this process yielded 55 distinct classes. This can be calculated as follows:

$$\binom{10}{2} + 10 = 45 + 10 = 55$$

Where $\binom{10}{2}$ signifies the number of ways to select two different classes from a set of ten, without considering the order of selection. The addition of 10 accounts for the scenario where the same class is chosen twice, also without considering the order of selection.

Our Data for CRNN:

In contrast to the previous network, the CRNN model processes raw audio files as input rather than MelSpectrograms. To facilitate this transition, we adopted a modified pre-processing approach:

1. **Audio File Duplication:**

- Initially, all audio files were replicated threefold, thus increasing the dataset size.

2. **Concatenation of Audio Files:**

- Subsequently, every two consecutive audio files were concatenated to create a novel composite audio file.

3. **Metadata and New Class ID Preservation:**

- For each of these newly formed audio files, we diligently preserved the associated new class ID and pertinent metadata.

Ofir Gonen
Yanai Levi

- This information was systematically catalogued in a CSV (Comma-Separated Values) file.

Much like the previous dataset, this approach resulted in a dataset encompassing 55 distinct classes, following the same calculation: $\binom{10}{2} + 10 = 45 + 10 = 55$

Solution:

CNN:

CNN is a type of artificial neural network that's particularly well-suited for tasks involving images and spatial data. CNNs have been revolutionary in the field of computer vision and have also found applications in various other domains. Here's a brief explanation of key concepts within CNNs:

1. **Convolutional Layer:** A convolutional layer is the fundamental building block of a CNN. It applies convolution operations to the input data and produces feature maps. Convolutional layers learn to recognize different features such as edges, textures, or shapes in an image.
2. **Pooling (or Subsampling) Layer:** After one or more convolutional layers, pooling layers are often added. Pooling reduces the spatial dimensions (width and height) of the feature maps while retaining important information. Max-pooling and average-pooling are common pooling techniques.
3. **Activation Function:** An activation function is applied to the output of each convolutional layer to introduce non-linearity into the model. Rectified Linear Unit (ReLU) is a commonly used activation function in CNNs.
4. **Fully Connected Layer:** After one or more convolutional and pooling layers, a CNN typically has one or more fully connected layers (also known as dense layers). These layers are similar to those in traditional neural networks and help in making final predictions.
5. **Flattening:** Before passing the data to fully connected layers, the feature maps are often flattened into a one-dimensional vector.
6. **Output Layer:** The output layer produces the final predictions or classifications based on the features learned by the previous layers. The choice of activation function in the output layer depends on the task (e.g., sigmoid for binary classification, softmax for multi-class classification).

Ofir Gonen
Yanai Levi

CRNN:

What is CRNN?

CRNN stands for "Convolutional Recurrent Neural Network." It's a type of neural network architecture that combines features of both convolutional neural networks (CNNs) and recurrent neural networks (RNNs). CRNNs are designed to handle sequential data, such as time-series data or sequences of images, where both spatial and temporal information are important.

Here's a brief overview of the two components within CRNN:

1. Convolutional Neural Network (CNN): CNNs are widely used for image analysis tasks. They excel at capturing spatial patterns and hierarchical features from input data, such as images or spectrograms. In a CNN, convolutional layers apply filters to input data to learn local patterns and extract increasingly abstract features. Pooling layers downsample the spatial dimensions of the feature maps, reducing computational complexity and preserving important information.
2. Recurrent Neural Network (RNN): RNNs are designed to handle sequential data by introducing the concept of memory or hidden states. They maintain an internal state that evolves with each new input, allowing them to capture temporal dependencies and patterns within sequences.

The CRNN architecture aims to combine the strengths of CNNs and RNNs to handle sequences efficiently, while also capturing both spatial and temporal information. It's particularly useful for tasks like speech recognition, music generation, and various audio and time-series data analysis. The convolutional layers in a CRNN handle spatial features and pattern recognition within local windows of the sequence, while the recurrent layers capture long-range temporal dependencies.

In applications involving audio, such as speech recognition or sound classification, CRNNs can process audio spectrograms (which represent the frequency content over time) to capture both the frequency patterns (spatial information) and how they evolve over time (temporal information).

Our CRNN building blocks:

Our network is designed for audio data classification using a combination of convolutional and recurrent neural network layers.

Let's break down each component of the network and explain their functions:

1. **MelspectrogramStretch:**

This is the first layer of the network.

This is a preprocessing technique for audio data used in Convolutional Recurrent Neural

Networks (CRNNs). It combines creating a Mel spectrogram, which shows audio frequency changes over time, with controlled time-stretching, adjusting audio duration. This prepares audio for neural network input by transforming it into a stretched Mel spectrogram, preserving both frequency content and timing patterns. This enhances the network's ability to learn from audio features effectively.

This layer transforms the raw audio data into a spectrogram representation.

2. **Conv2d** layer:

Short for 2D convolutional layer, is a core component of convolutional neural networks (CNNs). It performs a mathematical operation called convolution on 2D input data, typically images or feature maps. In the context of audio analysis, the input data can also be the spectrogram of the audio, where the spectrogram is a 2D representation of the audio signal's frequency content over time. During the convolution operation, a set of learnable filters, also known as kernels, slide over the input spectrogram in a systematic manner. At each position, the filter computes element-wise products with the local region it covers, and the resulting products are summed to generate a single value in the output feature map. This process allows the **Conv2d** layer to capture local patterns and features within the spectrogram, such as spectral shapes, changes, and patterns over time. By learning the optimal filter weights through the training process, the layer becomes proficient at detecting specific acoustic features relevant to audio analysis tasks, making it a crucial tool for tasks like music genre classification, sound event detection, and speech emotion analysis.

3. **Batch normalization** layer:

Batch normalization layers play a pivotal role in the Convolutional Recurrent Neural Network (CRNN) architecture. In the CRNN's context, batch normalization is employed to enhance training stability and accelerate convergence. During the training process, as each mini-batch of data passes through the network, batch normalization standardizes the activations of each layer. This ensures that the input to subsequent layers remains consistent, reducing internal covariate shift and allowing for more effective gradient flow during backpropagation. By normalizing activations, batch normalization mitigates the vanishing and exploding gradient problems, which can hamper training deep networks.

Additionally, batch normalization introduces learnable parameters that enable the network to scale and shift the normalized activations. This capacity gives the network greater flexibility in adapting to data characteristics. Overall, batch normalization contributes to improved training efficiency, faster convergence, and increased stability in CRNN models,

making them better suited for handling complex audio data and extracting meaningful features.

4. **ELU Activation Functions:**

The Exponential Linear Unit (ELU) activation function is a non-linear function applied element-wise to the output of a layer in a neural network. In the CRNN, ELU activation functions are used after convolutional and fully connected layers. ELU addresses the vanishing gradient problem by avoiding saturation of negative values while providing a smooth gradient for positive values. This helps accelerate training and improve model convergence.

ELU's unique characteristic is that it allows negative activations to have non-zero gradients, which can help prevent dead neurons and contribute to more expressive feature learning. By introducing non-linearity, ELU activations enable the CRNN to capture complex relationships in audio data, making them an essential element for effective information extraction.

5. **Max-Pooling Layers:**

Max-pooling layers are employed in the CRNN to down-sample the spatial dimensions of feature maps produced by convolutional layers. Max-pooling involves dividing the feature map into non-overlapping regions and retaining the maximum value within each region. This operation reduces the spatial resolution while retaining the most salient features.

In the context of audio analysis, max-pooling helps abstract away fine-grained temporal details while preserving crucial spectral information. This down-sampling also contributes to model efficiency by reducing the number of parameters and computational complexity. Max-pooling's ability to retain significant features while discarding less relevant information makes it a valuable tool in the CRNN's feature extraction process.

6. **Dropout Layers:**

Dropout layers are integrated into the CRNN to prevent overfitting, a common challenge in deep learning. During training, dropout randomly deactivates a fraction of neurons in the layer by setting their outputs to zero. This forces the network to learn robust representations that do not rely on specific neurons. Dropout effectively acts as a regularization technique, preventing the network from becoming overly specialized to the training data and promoting better generalization to unseen data.

In the CRNN, dropout layers are strategically positioned after convolutional and fully connected layers. By applying dropout, the CRNN gains resilience to noise and variations in

audio data, leading to improved model performance and the ability to handle diverse audio sources effectively.

7. **LSTM:**

This sub-module represents a Long Short-Term Memory (LSTM) recurrent layer. The LSTM is a type of recurrent neural network (RNN) designed to capture temporal dependencies in sequential data. It takes in the features extracted by the convolutional layers and processes them across time steps.

The **num_layers** parameter specifies the number of LSTM layers stacked on top of each other. The LSTM's output represents the temporal features learned from the input spectrogram.

8. **Linear** layer:

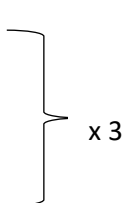
A Linear layer, also known as a Fully Connected layer, takes input values and transforms them using weights and biases to produce output values.

It's like a mathematical equation ($y=Wx+b$) that learns how input features contribute to desired outputs. Activation functions are often applied afterward to introduce non-linearity.

Linear layers play a central role in neural networks by enabling them to model complex relationships between inputs and outputs.

The architecture of the network:

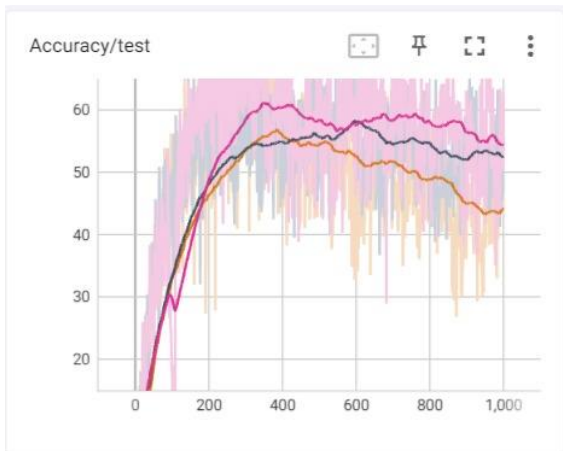
The order of the layers is as follows:

- MelspectrogramStretch
 - Conv2d
 - BatchNorm2d
 - ELU
 - MaxPool2d
 - Dropout
 - LSTM
 - Dropout
 - BatchNorm1d
 - Linear
- 

Ofir Gonen
Yanai Levi

Results:

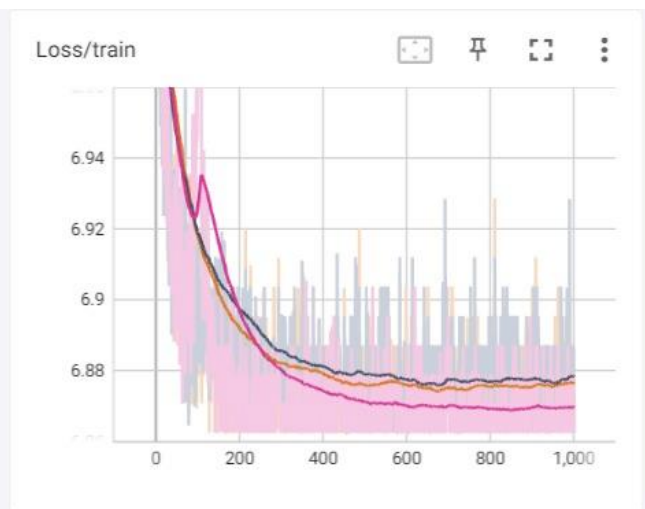
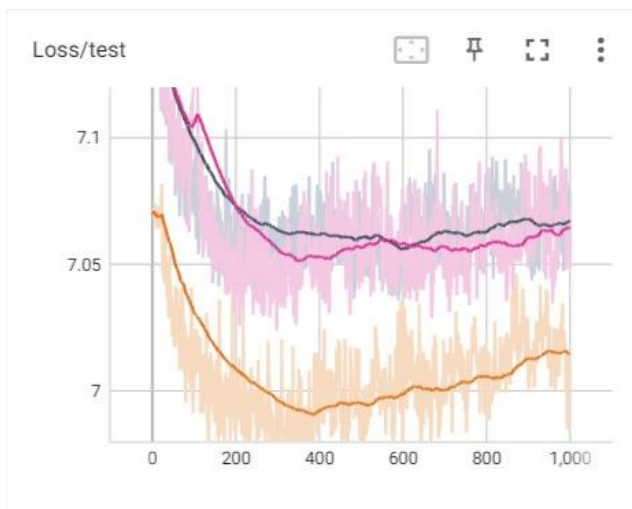
We will first show the results of the first network: **CNN10**



Orange: batch_size: 32
overlap: 3/4
momentum: 0.85

Blue: batch_size: 64
overlap: 3/4
momentum: 0.85

Pink: batch_size: 64
overlap: 0.5
momentum: 0.9

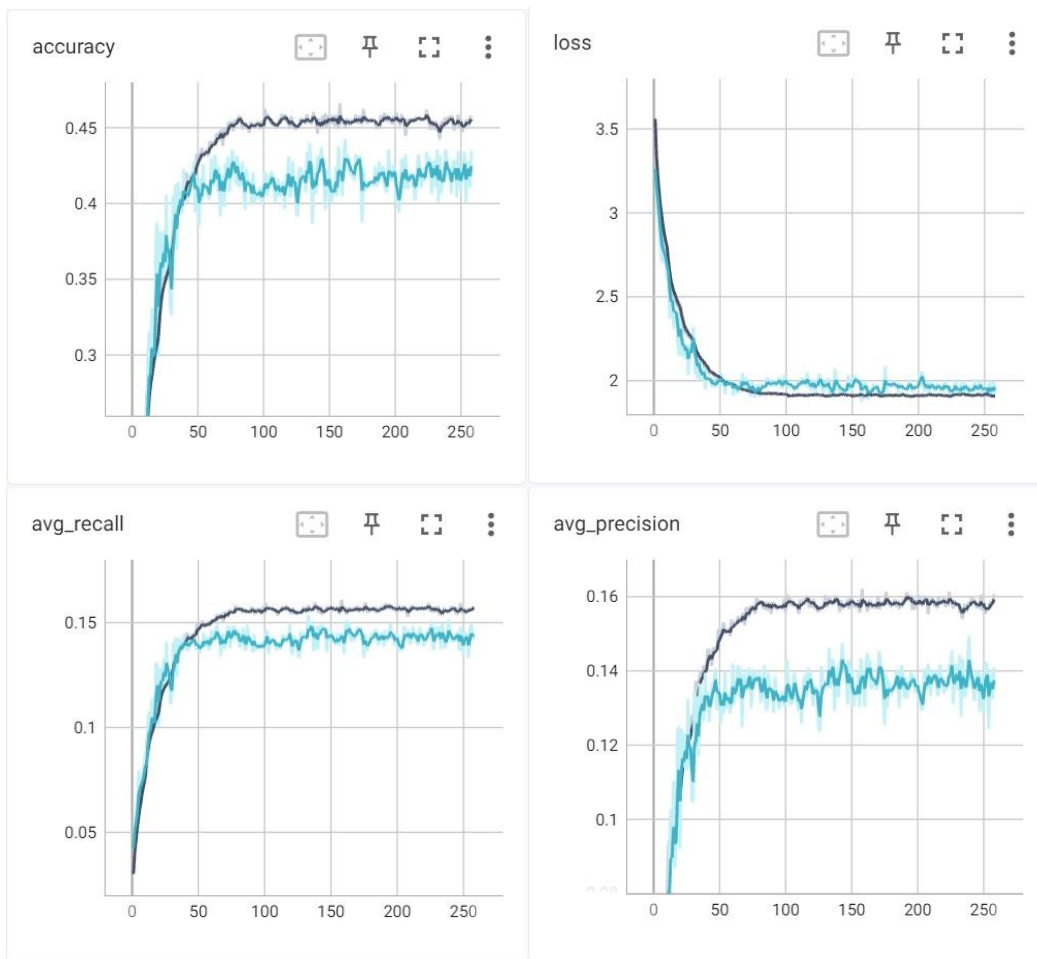


We can see that about after 400 epochs, our network is optimal in terms of accuracy (63%) and also in terms of the validation loss which starts increasing afterwards.

- Loss_new_dataSet_mel_BCE_loss_without
_same_energy_epochs:1000_batchsize:32
_fixed:True_optimizer:adam_lr:0.001_mom
entum:0.85
- Loss_new_dataSet_mel_BCE_loss_without
_same_energy_epochs:1000_batchsize:64
_fixed:True_optimizer:adam_lr:0.001_mom
entum:0.85
- Loss_new_dataSet_mel_BCE_loss_without
_same_energy_overlap0.5_epochs:1000_b
atchsize:32_fixed:True_optimizer:adam_lr:
0.001_momentum:0.9
- Loss_new_dataSet_mel_BCE_loss_without
_same_energy_overlap0.5_epochs:1000_b
atchsize:64_fixed:True_optimizer:adam_lr:
0.001_momentum:0.9

Ofir Gonen
Yanai Levi

CRNN results:



Light Blue – test set

Dark Blue – training set

We observe that our results indicate an average precision score of 16% and an average recall score of 0.15. Let's first grasp the meanings of the terms Precision, Recall, as well as Macro and Micro Averaging.

Precision:

A measure of how many of the positive predictions made by your network are actually correct. In other words, it quantifies the accuracy of the positive predictions. The formula for precision is:

$$Precision = \frac{TP}{TP + FP}$$

Where:

True Positives (TP) are the number of correct positive predictions. These are cases where the network correctly predicted a positive class, and it was indeed a positive class in reality.

False Positives (FP) are the number of incorrect positive predictions. These are cases where the network predicted a positive class, but it was actually a negative class in reality.

Recall:

Also known as sensitivity or true positive rate, is another important metric for evaluating the performance of a classification network. While precision focuses on the accuracy of positive predictions, recall focuses on how effectively a model can identify all the actual positive instances in a dataset.

Recall is calculated using the following formula:

$$Recall = \frac{TP}{TP + FN}$$

Where:

False Negatives (FN) are the number of missed positive predictions. These are cases where the network predicted a negative class, but it was actually a positive class in reality.

In the context of classification evaluation metrics like precision and recall, "macro" and "micro" are ways to aggregate these metrics when you have multiple classes. They provide different approaches to computing these metrics across all classes in your dataset.

Macro-Averaging:

- Macro Precision: To calculate macro precision, you compute the precision for each class individually and then take the average (mean) of these precisions. It treats each class equally, regardless of its size in the dataset.

$$Macro\ Precision = \frac{\sum_{i=1}^n Precision_{class_i}}{n}$$

- Macro Recall: Similarly, for macro recall, you calculate the recall for each class and then take the average.

$$Macro\ Recall = \frac{\sum_{i=1}^n Recall_{class_i}}{n}$$

Ofir Gonen
Yanai Levi

In the macro-averaging approach, each class contributes equally to the final metric, regardless of its prevalence in the dataset. This means that small classes have the same weight as large classes.

Micro-Averaging:

- Micro Precision: To calculate micro precision, you sum up the true positives, false positives, and false negatives across all classes and then compute precision using these aggregated values.

$$\text{Micro Precision} = \frac{\sum_{i=1}^n TP_{class_i}}{\sum_{i=1}^n TP_{class_i} + FP_{class_i}}$$

- Micro Recall: Similarly, for micro recall, you sum up the true positives, false negatives, and false positives across all classes and then compute recall using these aggregated values.

$$\text{Micro Recall} = \frac{\sum_{i=1}^n TP_{class_i}}{\sum_{i=1}^n TP_{class_i} + FN_{class_i}}$$

In the micro-averaging approach, all individual predictions are treated as a single large binary classification problem. It doesn't distinguish between different classes and computes precision and recall based on the global true positives, false positives, and false negatives.

Which to Use?

Macro-Averaging: This approach is useful when you want to give equal importance to each class, regardless of class size. It's a good choice when you want to assess the model's performance across different classes independently.

Micro-Averaging: This approach is useful when you want to assess the overall performance of the classifier as if it's a binary classification problem, without distinguishing between classes. It can be particularly helpful when you have class imbalance issues because it gives more weight to the larger classes.

Since we need to treat all classes equally to evaluate the overall performance of the classifier against the most common class labels, we calculated the Macro recall and Macro precision.

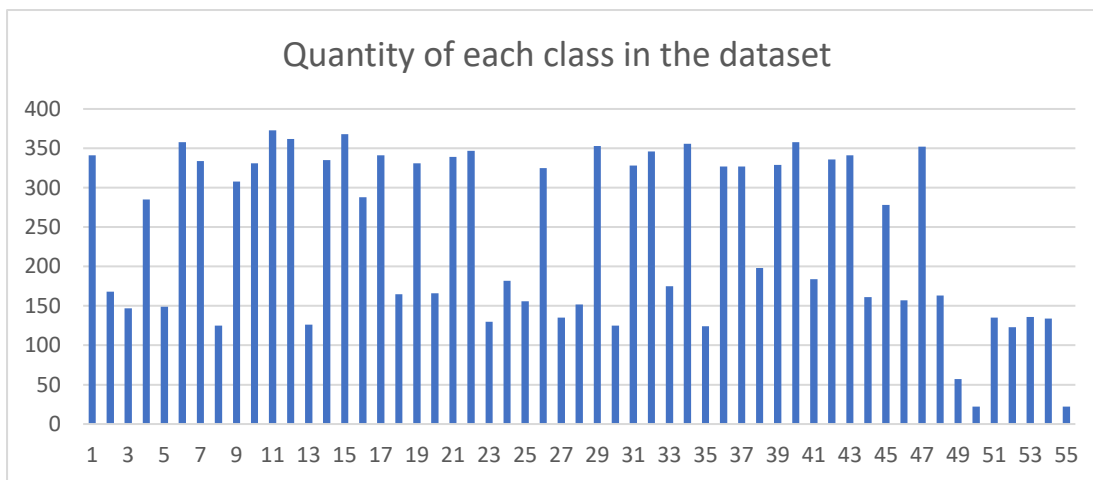
Upon conducting a comprehensive analysis comparing recall to accuracy, it becomes evident that our dataset exhibits certain imbalances. These imbalances are primarily attributable to the methodology employed for calculating recall, which, in our case, aligns more closely with the principles of Macro recall rather than micro recall. This distinction has significant implications.

The essence of this imbalance is that particular classes within our dataset demonstrate significantly lower recall values. Essentially, what this means is that these specific classes are not adequately represented in our dataset, and this underrepresentation is exerting a noticeable impact on the overall average recall metric.

In an ideal scenario with a perfectly balanced dataset, the values of accuracy and recall would tend to converge. In other words, when the class distribution is uniform, the model's ability to make accurate positive predictions (recall) aligns with its overall accuracy.

However, when we delve into the distribution of our classes, it becomes quite evident that certain classes have a relatively low number of instances or samples.

Our class distribution:



Ofir Gonen
Yanai Levi

Summary:

To conclude, our findings suggest that the CNN demonstrates superior performance in terms of accuracy when compared to the CRNN. Several factors may contribute to this outcome, primarily the inherent complexity of the CNN architecture, which allows it to capture intricate patterns and features in the data more effectively. However, this comparison should be viewed within the context of our current dataset and experimental setup.

To further investigate the performance of the CRNN, it is recommended for future research to conduct evaluations with a more balanced dataset. Imbalanced datasets can often skew results, and addressing this issue could provide a more accurate assessment of the CRNN's capabilities.

Moreover, a significant avenue for potential improvement lies in conducting an extensive hyperparameter sweep for both models. This process involves systematically varying and fine-tuning the model's hyperparameters, such as learning rates, batch sizes, and layer configurations, to identify the optimal combination. By doing so, we can potentially unlock hidden potential within the CRNN and further enhance the networks' performance.

Additionally, the availability of a larger dataset could prove invaluable in refining and fine-tuning both neural network architectures. A larger dataset provides more diverse and representative examples for the models to learn from, potentially reducing overfitting and increasing generalization. This could lead to improved performance and greater robustness in real-world applications.

In conclusion, while our initial results favor the CNN over the CRNN in terms of accuracy, future work should involve a more balanced dataset, a comprehensive exploration of hyperparameters, and the consideration of larger datasets to ensure a more thorough evaluation and potential enhancements to both neural network models.