Deep Learning Project 2 Report

Jedrzej Ruciński, Adam Czerwoński

April 2024

Contents

1 Introduction

The goal of this project was to create classification models for TensorFlow Speech Recognition Challenge hosted by Kaggle, but without using CNN architectures, which were used in all of the top scoring solutions. We attempted to leverage RNNs and Transformer architecture [Ashish Vaswani, 2017] which is now a state of the art technique, but wasn't popularised at the time the competition took place (2017).

1.1 Data

We were provided with 1-second audio recordings from the Speech Commands Dataset. They were split into train (64k+ files) and test (150k+ files) parts. The files from the train part were labeled with one of the following classes: "yes", "no", "up", "down", "left", "right", "on", "off", "stop", "go", "zero", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine", "bed", "bird", "cat", "dog", "happy", "house", "marvin", "sheila", "tree", "wow", whereas the test part had no labels. Our goal was to train a model that would predict labels for the test part, however, it should only output one of the following classes: "yes", "no", "up", "down", "left", "right", "on", "off", "stop", "go", "silence" or "unknown" for all of the other words.

2 Data Analysis

To be able to model any data we of course need to analyze it. The raw audio data is given to us in *wav* files from which we read the signal and sampling rate. Let us take a look at how these signals pan out for each of our target classes.

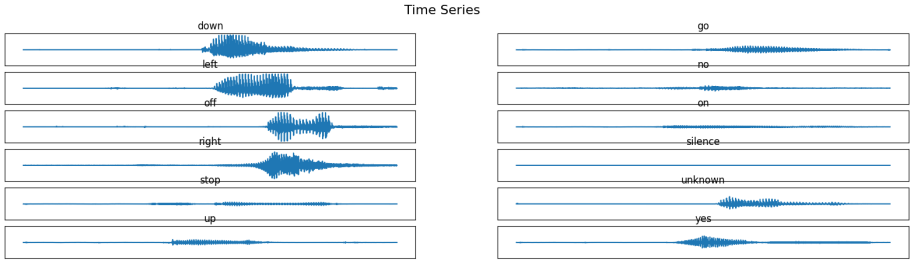

Figure 1: Signal as time series for each class.

These are values for example sample audio clips for each word, but they give us a good understanding that within each clip there is a lot of blank space that needs to be cleared out. We do this by applying an **envelope mask** on our audio clips that calculates the mean of values over a rolling window on the time

series and if that value does not meet a certain threshold it is masked off. This results in the following for the example sound bites:

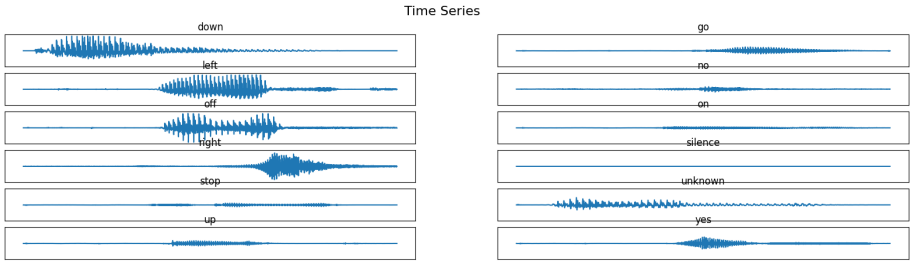

Figure 2: Filtered signal as time series for each class.

Now that we have gotten rid of some of the useless silence we may proceed to calculate the Fourier Transform [Bracewell and Bracewell, 1986] on these signals using the Fast Fourier Transform (FFT) method for computational optimization. This will allow us to look at the frequencies of particular wavelengths in the signal and may allow for a better basis of distinction between these words.

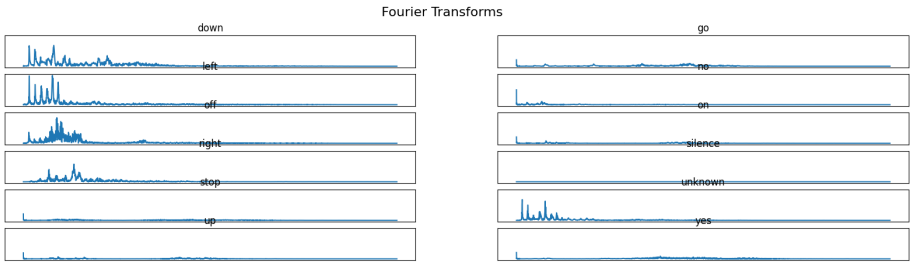

Figure 3: Fourier frequencies plot

As we may see, although applying Fourier's transform on our audio files did provide some distinction between groups of words, we still may see words that are very similar in terms of their plots, which may be a problem when trying to model this data. We next try out a filter bank, which, in audio processing, is done to decompose an audio signal into multiple frequency bands, each containing a different range of frequencies. This decomposition allows for more focused analysis and manipulation of the signal in various frequency regions. By dividing the audio signal into frequency bands, a filter bank enables the analysis of the spectral content of the signal. This analysis can reveal information about the distribution of energy across different frequency ranges, aiding in tasks such as the one at hand.

Now that we have this we may try to reduce the correlation and similarity

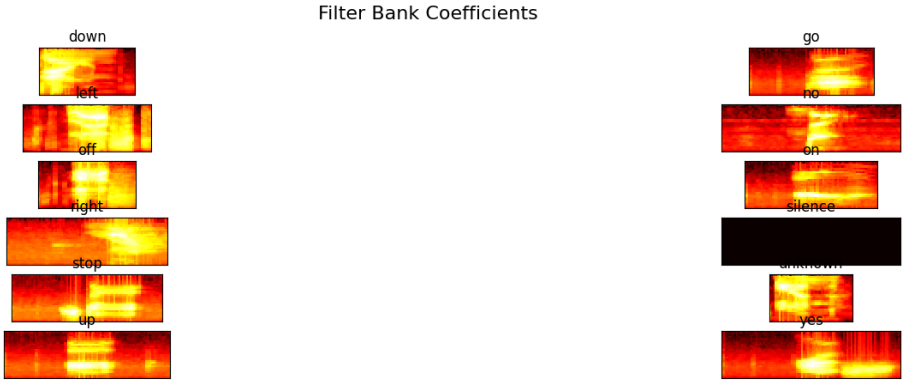

Figure 4: Filter bank coefficients heat plot

between the classes by taking our preprocessing a step further, and applying what is called a *Discrete Cosine Transform* on the filter banks which will lead us to obtain something called Mel-Frequency Cepstral Coefficients or MFCCs [Abdul and Al-Talabani, 2022]. Data in this format utilizes all of the methods we wrote about, so masking, window filter, Fourier Transforms, and filter banks, all together to achieve audio data in an optimal form for modeling. In general, to obtain MFCCs we must follow somewhat of a similar pattern to the below, which is all excellently descirbed in detail in the article [Fayek, 2016]:

1. Divide the audio signal into short, overlapping frames.
2. Apply a windowing function (like Hamming window) to each frame.
3. Use Fast Fourier Transform (FFT) to convert each frame into the frequency domain.
4. Apply a Mel filter bank to the power spectrum to map it onto the mel scale.
5. Take the logarithm of the energy in each filterbank output.
6. Use Discrete Cosine Transform (DCT) to transform the log-power spectrum.
7. Retain the lower-order coefficients as MFCCs, which represent the spectral characteristics.

Here is what these coefficients look like once applied to our data.

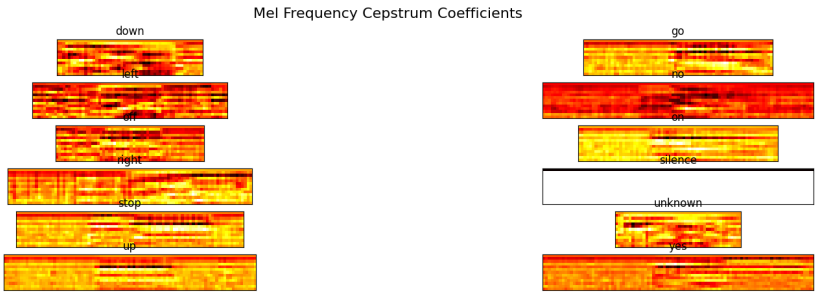

Figure 5: Mel-Frequency Cepstral Coefficients heat plot

3 Audio Spectrogram Transformer

In this section, we will discuss the approach that gave us 92.8% accuracy which is 1.7 percentage points more than the winning submission on Kaggle. And that is the Audio Spectrogram Transformer (AST) introduced in [Yuan Gong, 2021]. AST leverages the standard approach of transforming audio into an image (spectrogram) and then uses Visual Transformer [Alexey Dosovitskiy, 2020] for classification.

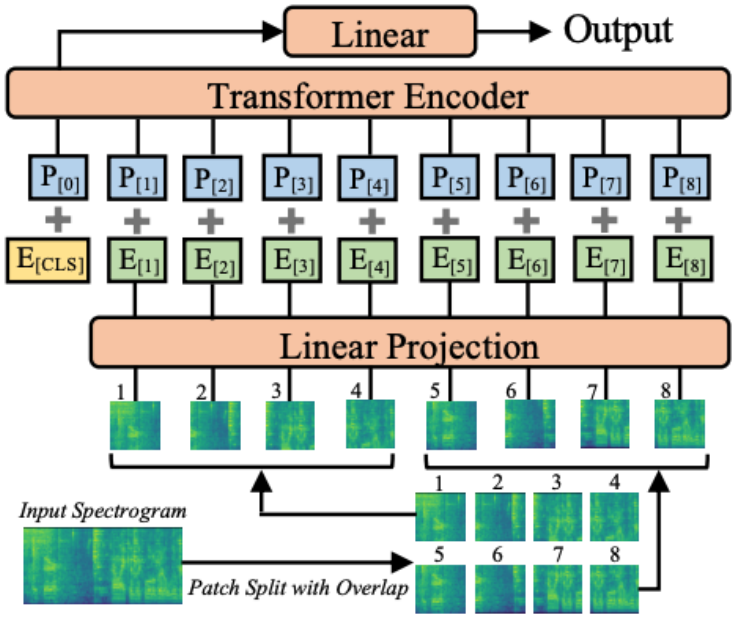

Figure 6: Audio Spectrogram Transformer architecture.

3.1 Model architecture

Let's investigate the AST architecture by tracking the forward pass of one sound wave step by step.

1. Convert the wave into a spectrogram. This is more of a data preprocessing step than an actual part of the model as this is done independently to the model parameters.
2. Split the spectrogram into overlapping patches.
3. Flatten the patches into one dimensional vectors using linear projection.
4. Append CLS token to the sequence of embedded patches.
5. Add trainable positional encoding to each of the embedded patches.
6. Pass this sequence through the Transformer encoder fixed number of times.
    a. Normalise the sequence.
    b. Pass it through a multi-head attention block.
    c. Add the residual connection.
    d. Normalise.
    e. Pass it through a multilayer perceptron.
    f. Add the residual connection.
7. Extract only the value of CLS token.
8. Pass it through the final multilayer perceptron that is customised for the given classification task.

3.2 Results

We have used the model from Hugging Face transformers library that was firstly pretrained on the ImageNet dataset and then finetuned on the Speech Commands v2 dataset. After passing our test data through the model we have changed all of the predicted labels that were not one of the 10 classes mentioned in 1.1 to "unknown". Then if the confidence of the model for the predicted label was less then 40% we relabeled it to "silence". This approach gave us 92.77% accuracy on the Kaggle private leaderboard.

4 RNN and CNN approaches

Apart from the excellent result mentioned in the previous section which was achieved using the Transformer architecture, we also tested out other, simpler network architectures on the data, some known to us before (CNNs) and some which we learned for the purpose of this task (RNNs). We train these models using both Early Stopping and Reduce Learning Rate on Plateau callbacks in tensorflow which allow us to control the overfitting and learning tempo of all the models.

4.1 Reccurent Neural Networks

Recurrent Neural Networks (RNNs) are a class of neural networks designed to process sequential data by maintaining an internal memory. Unlike traditional feedforward neural networks, which process data input by input, RNNs can capture information from previous inputs to make decisions about the current input.

However, standard RNNs can suffer from the vanishing gradient problem, where gradients diminish as they propagate back in time, leading to difficulties in learning long-term dependencies.

To address this issue, various advanced RNN architectures have been developed, such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs).

LSTM has a more complex architecture compared to GRU. It consists of three gates: the input gate (i), the forget gate (f), and the output gate (o). These gates control the flow of information through the cell state, allowing the LSTM to remember or forget information over time. On the other hand, GRU has a simplified architecture with two gates: the update gate (z) and the reset gate (r). The update gate controls how much of the previous hidden state should be retained, and the reset gate determines how much of the past information to forget.

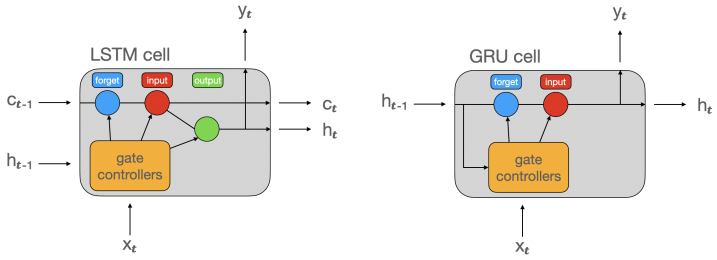

Figure 7: LSTM and GRU layer comparison, diagram taken from Datacamp online campus.

For our task we trained two small Recurrent Neural Networks, using 60% of

our train data for the training and the rest for testing and validation as we did not submit these results to the Kaggle competition as they heavily stood apart from the the results achieved by the Transformer model. These networks were built using TensorFlow in the following way:

1. GRU and LSTM layers at the beginning, depending on the network.
2. These were followed by a Dropout layer to reduce overfitting.
3. We then added multiple TimeDistributed Dense layers.
4. The networks were rounded out by a Flatten layer along with a Dense layer with softmax activation to return the class probabilities.

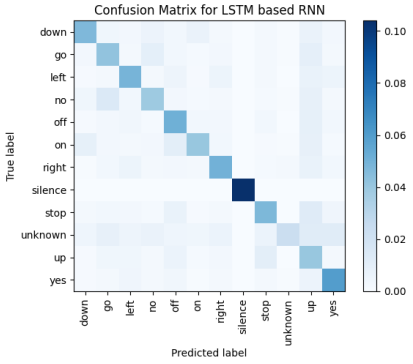

Figure 8: Confusion matrix for predictions made by LSTM model which achieved a local accuracy score of 59.28% on all 12 classes, on our local test data.

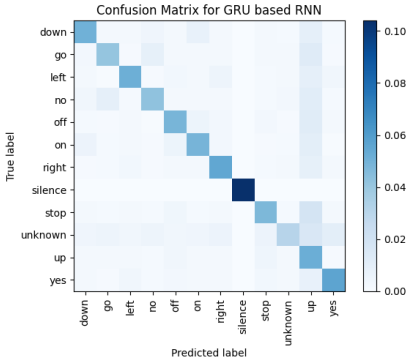

Figure 9: Confusion matrix for predictions made by GRU model which achieved a local accuracy score of 62.88% on all 12 classes, on our local test data.

As we may see these models performed fairly similarly with the GRU model having a better accuracy score by around 3.5 percentage points. This may be due to the fact that, as previously stated this model has less parameters and we did not due much tuning for either of the models. In both cases, we may see that *silence* is the easiest class to predict.

4.2 The convolutional approach

Additionally, we also tried out two simpler models, based on the idea of convolution which gave us much success during the previous image recognition project. Since looking at audio in terms of the MFCC spectrum does sort of resemble a form of image recognition we may seek success with these methods here.

We try out a simple CNN with multiple Convolutional Layers that iteratively increase in number of filters and then are finished off by a couple of Dense layers along with max pooling.

We also try out a ResNet model with three residual blocks that utilize max pooling and global average pooling applied on their outputs. We also use Elastic Net Regularization in the output layer to control overfitting, which is just a mixture of l_1 and l_2 losses.

So let us once again see the results of this training process looking at confusion matrices that are produced as the outcome of analyzing results on our local testing data.

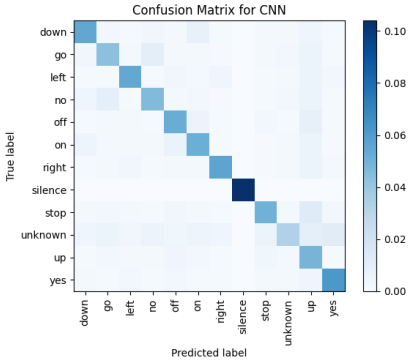

Figure 10: Confusion matrix for predictions made by CNN model which achieved a local accuracy score of 65.76% on all 12 classes, on our local test data.

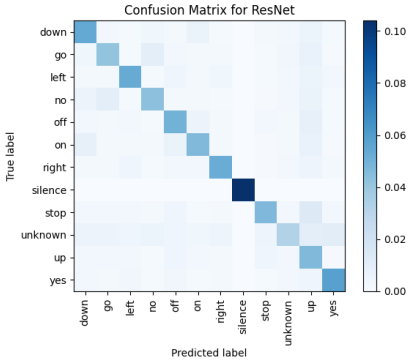

Figure 11: Confusion matrix for predictions made by ResNet model which achieved a local accuracy score of 62.86% on all 12 classes, on our local test data.

We may see that convolutional-based networks perform better on local tests than our recurrent networks. This behavior might be due to the fact that choosing the perfect parameters for an LSTM or GRU layer is a more difficult task than taking CNNs we tested out previously to work on images. Given more time and resources we could strive towards optimizing our RNNs but for the local results, it looks like CNNs prevail.

Additionally, we take a look at an ensemble of all 4 of the models presented in this section, which improves the local accuracy to 66.84% and results in the following confusion matrix:

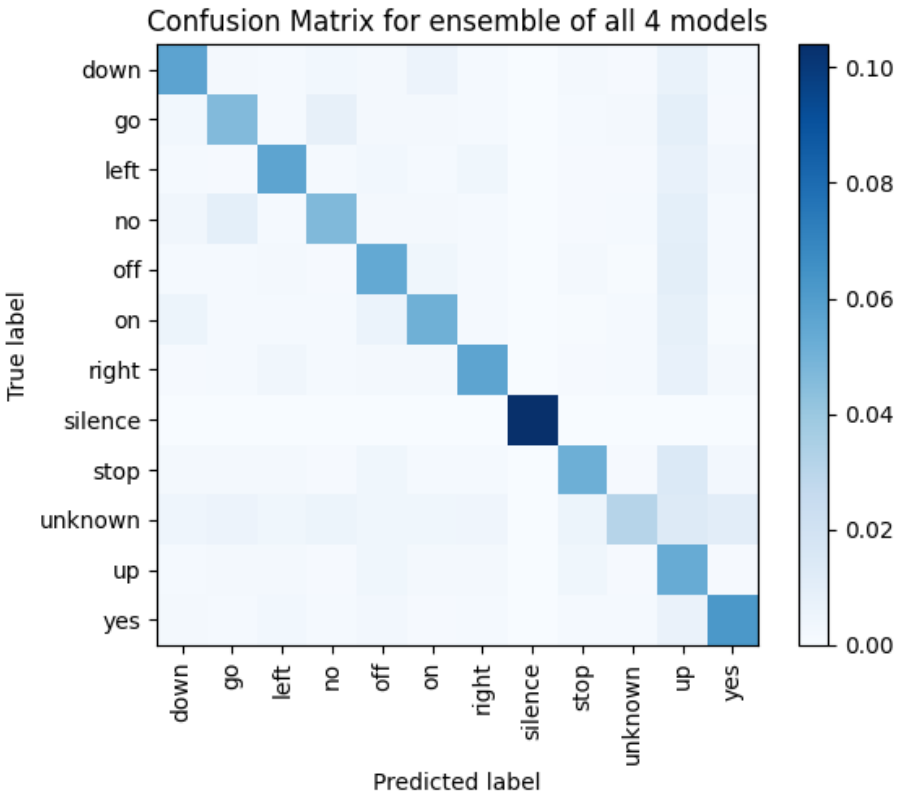

Figure 12: Ensemble confusion matrix

5 Binary task

Due to the fact that all of the above calculations and modeling are very complex in terms of time resources and computing resources, we also took a look at a simplified version of this audio classification task. In this sub task we only aim to differentiate between the *yes* and *no* phrases. Since this will be easier for us to model in terms of computing, this task is the one we will use to check parameter impact of on the model accuracy.

We again focus on our Recurrent Neural Networks, so the ones powered by LSTM and GRU architectures. Our parameter will also be correlated to these architectures as it will be the number of these layers in their corresponding networks. The values we shall test will be in the range of (1, 3, 5). The obtained accuracies are as follows:

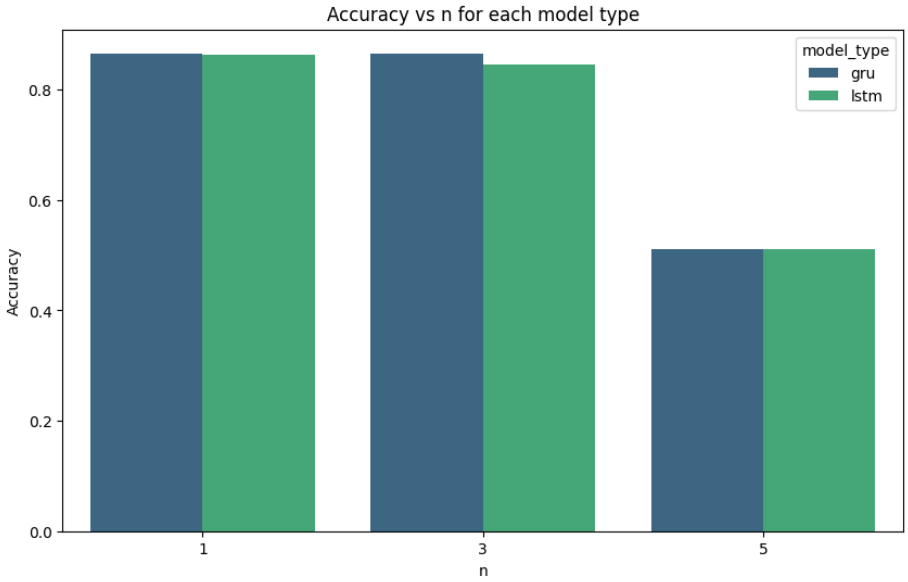

Figure 13: Binary task accuracy for LSTM and GRU models

As expected, the accuracies of our models are much higher than on the multi-classification tasks. We also see a very clear impact of our parameter as 5 of these recurrent layers are already too much and lead to a useless model.

References

[Abdul and Al-Talabani, 2022] Abdul, Z. K. and Al-Talabani, A. K. (2022). Mel frequency cepstral coefficient and its applications: A review. *IEEE Access*, 10:122136–122158.

[Alexey Dosovitskiy, 2020] Alexey Dosovitskiy, Lucas Beyer, A. K. D. W. X. Z. T. U. M. D. M. M. G. H. S. G. J. U. N. H. (2020). An image is worth 16x16 words: Transformers for image recognition at scale.

[Ashish Vaswani, 2017] Ashish Vaswani, Noam Shazeer, N. P. J. U. L. J. A. N. G. L. K. I. P. (2017). Attention is all you need.

[Bracewell and Bracewell, 1986] Bracewell, R. N. and Bracewell, R. N. (1986). *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York.

[Fayek, 2016] Fayek, H. M. (2016). Speech processing for machine learning: Filter banks, mel-frequency cepstral coefficients (mfccs) and what's in-between.

[Yuan Gong, 2021] Yuan Gong, Yu-An Chung, J. G. (2021). Ast: Audio spectrogram transformer.