Speech Recognition
ELEC-E5510

Language recognition

Álvaro Orgaz Expósito
Student number: 802101
alvaro.orgazexposito@aalto.fi

Rachhek Shrestha
Student number: 802130
rachhek.shrestha@aalto.fi

December, 2019

Contents

1 Introduction

Language recognition is the automatic process to quantitatively determine which language is being spoken in a speech sample. In the last decades, several technological advancements in signal processing, pattern recognition, cognitive science and machine learning have allowed for incredible progress in this topic enabling applications such as spoken language translation or spoken document retrieval.

It is estimated that there exist several thousands of spoken languages in the world but the total number of phones required to represent all the sounds of these languages ranges only from 200 to 300 as explained by Haizhou et al. in [1]. Then, although there is a wide diversity in languages, many share similar characteristics like intonation, rhythm, stress pattern, phonetic system and phoneme sequences. Moreover, we know that humans recognize languages through a perceptual or psychoacoustic process that is inherent in the auditory system and, just like any other artificial intelligence technologies, spoken language recognition aims to replicate such human ability through computational means. Thus, the principal problem is how to scientifically measure the individuality of the diverse spoken languages in the world to classify correctly their speech samples.

Human listening experiments such as [1] suggest that there are different classes of language information or cues that contribute to the human perceptual process for spoken language recognition. Firstly, prelexical information including acoustic phonetics (physical sound patterns of the speech sounds as entities or phonemes that the human speech apparatus is capable of producing), phonotactics (constraints that determine permissible phones sequences or syllable structures in a language), and prosodic (features in running speech such as stress, duration, rhythm, and intonation). Secondly, lexical information of languages about their phonological system that governs how symbols are used to form words or morphemes, and their syntactic system that governs how words and morphemes are combined to form phrases. The lexical knowledge can be gained by experience and studies revealed that given only a little previous exposure humans listeners can effectively identify a language without much lexical information. Therefore, mainstream research about automatic language recognition systems uses acoustic phonetics and phonotactics information.

From a mathematical and engineering perspective the problem of language recognition starts from a speech waveform in an unknown language which is split into a sequence of frames with their respective acoustic feature vectors. Then, the problem is to assign the most likely language to this acoustic observation. But to deal with phones and phonotactic knowledge it is assumed that the speech waveform can be segmented into a sequence of phonemes which in practice is unknown and has to be decoded from the observation waveform with phone models.

In this project we are approaching the task by using state of the art artificial intelligence models, such as CNN and LSTM, as well as Fourier transforms to extract the Mel Spectrogram and MFCC features from labelled audio samples from 6 different languages. By using our classification approach, which is deeply explained in sections 3 and 4, we achieve an accuracy of 83% in a test set. **Keywords:** automatic language recognition, signal processing, acoustic features extraction, machine learning, classifiers, CNN, LSTM.

2 Literature study

Language recognition has been well studied in the literature and there are several different approaches to build a language recognition system. During the whole project, we have read a large number of papers to acquire a better understanding of the past and current state-of-the-art approaches and solutions for this task. Here we provide an overview of the main previous approaches which we have used as reference and inspiration.

The paper [1] by Haizjou et al. has listed down a couple of recent advances. They report that the models used to build contemporary language recognition systems from labelled training data could be: 1) stochastic e.g. Gaussian Mixture Models (GMM) and Hidden Markov Model (HMM); 2) deterministic e.g. Vector Quantization (VQ), Support Vector Machine (SVM) and Neural Networks; 3) discrete stochastic e.g. N-gram Models. The paper also claims that acoustic-phonetic and phonotactic approaches are the most effective ones for modelling the characteristics of languages. There is also an introduction to advanced techniques that have explored the combination of different features and approaches to build the most state-of-the-art language recognition system which is composed of several subsystems that complement each other and are combined using an effective fusion technique.

There is another novel way of language identification developed in [2] and [3] which uses Long Short-Term Memory (LSTM) Recurrent Neural Networks (RNN). This technique effectively exploits temporal dependencies in acoustic data to learn relevant features for language discrimination. It is reported to achieve better performance than previously used Deep Neural Networks (DNN) based systems with a reduced number of parameters.

Finally, another important reference used to select a suitable model architecture and features maps extraction is [4]. Indeed, for understanding and implementing well the concept of convolutional 1D layers we have used the article [5].

3 Data

The dataset is a subset of approximately 50 hours of audio files from the Mozilla Common Voice speech dataset, freely available at `https://voice.mozilla.org`. This source provides a lot of hours of speech for many languages, but we have selected 6 (Estonian, German, Mandarin, English, Spanish, Farsi, and Kabyle) and a subset of samples for each one not shorter than 3 seconds. It is a very balanced dataset for the language recognition task of these 6 different languages. Moreover, the audio files have been converted from MP3 format to waveform WAV files at a frequency of 16 kHz with a normalized volume of -3 dBFS.

The dataset is split into speaker-disjoint train and test sets, which means that none of the test speakers has been used for training, in the following way:

- German: In the training set 7053 files, total duration of 9 hours, and average file duration: 4.59 seconds. In the test set 494 files, total duration of 24.7 minutes, and fixed file duration 3 seconds.

- Spanish: In the training set 6527 files, total duration of 8.13 hours, and average file duration: 4.47 seconds. In the test set 452 files, total duration of 22.6 minutes, and fixed file duration 3 seconds.
- Estonian: In the training set 4818 files, total duration of 8.91 hours, and average file duration: 6.66 seconds. In the test set 487 files, total duration of 24.35 minutes, and fixed file duration 3 seconds.
- Farsi: In the training set 6337 files, total duration of 8.74 hours, and average file duration: 4.97 seconds. In the test set 516 files, total duration of 25.8 minutes, and fixed file duration 3 seconds.
- Kabyle: In training set 6676 files, total duration of 8.36 hours, and average file duration: 4.51 seconds. In the test set 486 files, total duration of 24.3 minutes, and fixed file duration 3 seconds.
- Mandarin: In the training set 4298 files, total duration of 8.03 hours, and average file duration: 6.71 seconds. In the test set 695 files, total duration of 34.75 minutes, and fixed file duration 3 seconds.

Note that all test audio files have exactly 3 seconds duration, which at a frequency of 16kHz corresponds to 48.000 audio samples. This fact determines the features preprocessing of the training audio files which have different durations.

4 Methods

4.1 Audio features extraction

Firstly, we are going to introduce the acoustic features extraction used to prepare the inputs for the modelling method explained in section 4.3. For each entire audio file (train and test sets with different durations) we have extracted:

a) Log-Mel-Spectrogram keeping 40 dimensions or filterbanks.

b) Mel-Frequency Cepstral Coefficients (MFCC) keeping 40 dimensions.

This feature extraction has been implemented in Python using the module *librosa*. The parameters configuration used is:

- Length of the FFT window (*n_fft*): 512 samples equivalent to 32ms.
- Samples between successive frames (*hop_length*): 160 equivalent to 10 ms.
- Sampling rate or frequency (*fr*): 16 kHz.
- Lowest/highest frequency (*f_min/f_max*): 300/8000 Hz.

Note that for test audio files with a fixed duration of 3 seconds, the shape of the feature is exactly (297,40) where 297 is the number of frames equal to $1 + (48000 - 512)//160$ and 40 the kept dimensions.

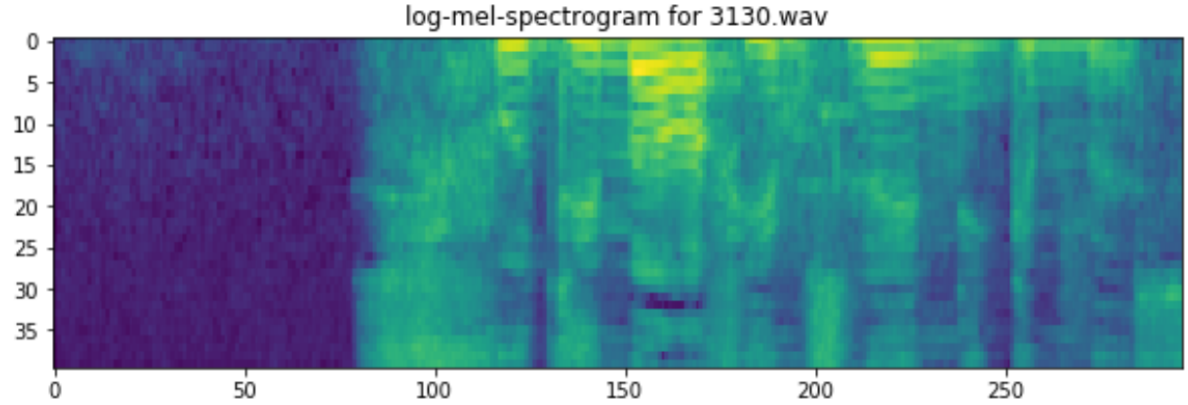

Figure 1: Feature extraction Log-Mel-Spectrogram for a test audio example. Statistics: shape (297,40), min -13.3, max 4.1, mean -4.1, std 2.5.

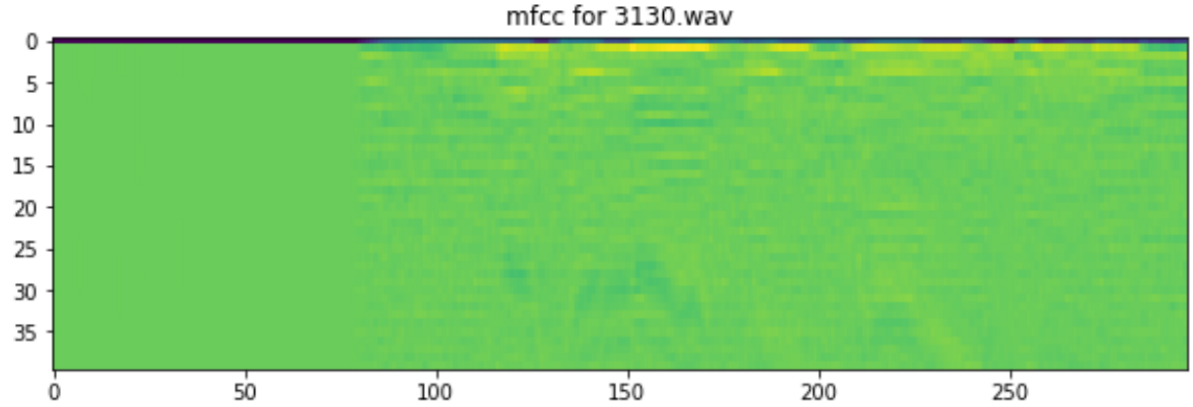

Figure 2: Feature extraction MFCC for a test audio example. Statistics: shape (297,40), min -648.6, max 191, mean -8.9, std 78.6.

4.2 Features preprocessing

Secondly, we are going to introduce the preprocessing applied to the extracted features before feeding the model. Since we are testing audio files of precisely 3 seconds, it is fair that each input feature sample has the same shape as the test ones. Then, we have split the features extracted from the training data audio files into equal chunks of shape (297,40) by using moving windows of hop length 100 frames. It is equivalent to $100 * 160 = 16.000$ samples or 1 second. Note that we delete the remaining seconds after the last possible complete chunk. The lower the number of frames between chunks, the lower the duration dropped, and more new chunks or training samples generated.

The reasoning behind this is:

- consider the time context of each original audio file without splitting the files exactly every 3 seconds,
- data augmentation since we are creating from 35k training audio files around 95k training samples versus 47k samples by splitting exactly every 3 seconds,
- and do not discard many remaining seconds or frames after the last possible split.

4.3 Modelling

Thirdly, we will introduce the 2 modelling techniques that we have used for this classification task.

CNN

The first one is a Convolutional Neural Network (CNN). Since CNNs are very popular models we are going to go deeper only about the type of convolutional filters used which is convolutional 1D filters. As explained in [5], CNNs work the same way whether they have 1, 2, or 3 input dimensions, however, the difference is the structure of the input data and how the filter, also known as a convolution kernel or feature detector, moves across the data. In our case, a features sample is made up of 297 frames and each one is a 40-dimensional vector that represents it. The filter covers at least one frame, and kernel size parameter specifies how many frames the filter should consider at once. In Figure 3 we can see how the filter scans the data. On the other hand, 2D convolutional filters move over the input features both horizontally and vertically.

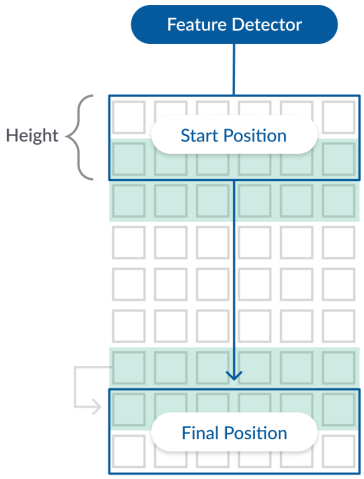

Figure 3: Convolutional 1D filters.

About the CNN model architecture, in Figure 4 you can see all the convolutional 1D layers used with $ReLU$ activation function, respective strides (1,2,1,1) and kernel size (6,7,1,1), as well as the global average pooling layer followed by fully connected layers and a final *softmax* activation function for the 6 output nodes.

LSTM

Secondly, we have implemented a LSTM model. LSTM Neural Networks have an edge over the conventional feed-forward network and Recurrent Neural Networks (RNN). This model can selectively "remember" and "forget" patterns for a longer duration of time than basic RNN. Their basic architecture is composed of *forget gates* responsible for removing information from cell states, *input gates* that add information to cell state, and *output gates* that select useful information as output. About the LSTM model architecture, in Figure 5 you can see the LSTM layer with 32 units followed by a fully connected layer and a final *softmax* activation function for the 6 output nodes.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_5 (Conv1D)            (None, 292, 500)          120500
_________________________________________________________________
conv1d_6 (Conv1D)            (None, 143, 500)          1750500
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 143, 500)          250500
_________________________________________________________________
conv1d_8 (Conv1D)            (None, 143, 3000)         1503000
_________________________________________________________________
global_average_pooling1d_2 ( (None, 3000)              0
_________________________________________________________________
dense_4 (Dense)              (None, 1500)              4501500
_________________________________________________________________
dense_5 (Dense)              (None, 600)               900600
_________________________________________________________________
dense_6 (Dense)              (None, 6)                 3606
=================================================================
Total params: 9,030,206
Trainable params: 9,030,206
Non-trainable params: 0
```

Figure 4: CNN model architecture.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_11 (LSTM)               (None, 32)                5888
_________________________________________________________________
dense_10 (Dense)             (None, 6)                 198
=================================================================
Total params: 6,086
Trainable params: 6,086
Non-trainable params: 0
```

Figure 5: LSTM model architecture.

4.4 Training procedure

According to the training method used, it is important to explain how we tackle the problem of working with 52 hours of audio files to train the models. We have implemented this project using Python and the module *Keras* of *Tensorflow*. Since the preprocessed features occupied up to 15GB depending on the experiment, we used the *fit_generator Keras* function to iteratively load a batch of size 40 samples. Each batch contained samples from all languages to balance the training and for completing each epoch including all training samples. About the loss function, we have used *Categorical Cross Entropy* because this is a multiclass classification problem. We have optimized it by using *Adam Amsgrad* because it is an adaptive learning rate optimization algorithm that's been designed specifically for training deep neural networks [6]. And the learning rate is 0.001. Moreover, early stopping in train and test sets for accuracy and loss metrics have been used in all the experiments.

5 Experiments

The main objective has been to build an end-to-end acoustic classifier (using only language labels and audio files) and for doing that, we have developed different modelling experiments to find the optimal performance.

CNN

Experiment 1: The first experiment compares the performance of the designed CNN model with different input features: Log-Mel-Spectrogram with 40 dimensions kept, Log-Mel-Spectrogram with 80 dimensions kept, Log-Mel-Spectrogram with 40 dimensions kept without moving window when creating multiple chunks of 3 seconds, and MFCC with 40 dimensions kept.

Experiment 2: The second experiment compares the best result of experiment 1 with the same case but normalizing the input features by frames.

Experiment 3: The third experiment compares the best result of experiment 2 with the same case but adding a *Dropout* layer and *Ridge* regularization.

Experiment 4: The fourth experiment compares the best result of experiment 3 with the same case but modifying the speed of the input audios to 0.9x. It is a method of slowing down the audio files which at the same frequency level will give as more frames in the input features so it is working as data augmentation too.

LSTM

Experiment 5: The fifth experiment compares the performance of the designed LSTM trained with the same features normalized MFCC with 40 dimensions. It is compared with a different number of units: 32, 100 and 200.

6 Results

In Figure6 we can find the result of experiment 1. We can observe that MFCC and Log-Mel-Spectrogram with 80 dimensions are not providing good results. Probably we would need to normalize those features since their variance is higher. Moreover, using moving windows when splitting the training features as explained before does not seem to increase performance but it helps to converge faster (remember that we are using early stoppings), so we will continue with this model.

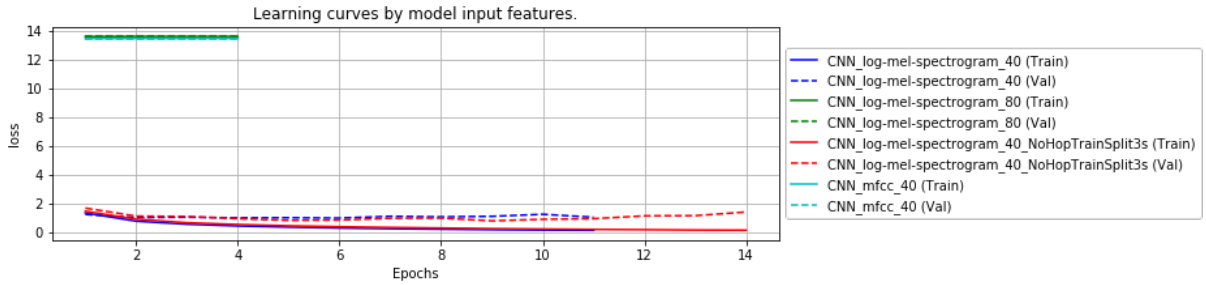

Figure 6: Loss curves of experiment 1. Respective accuracy curves in appendix.

In Figure 7 we can find the result of experiment 2. Clearly, the features normalization is boosting the performance so we will continue with this model.

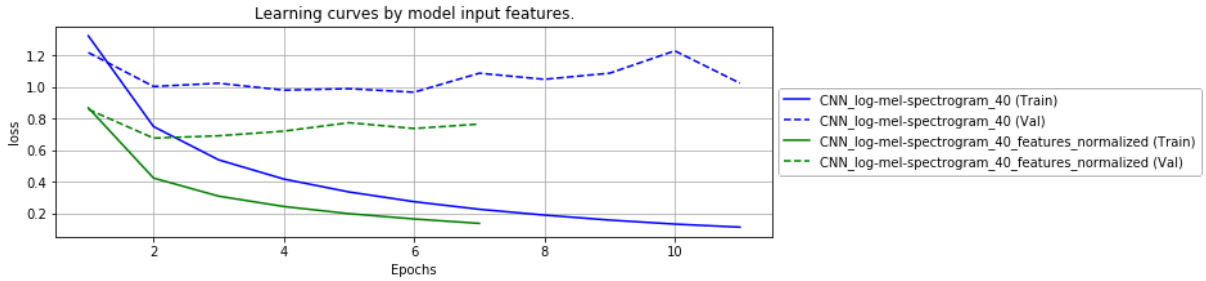

Figure 7: Loss curves of experiment 2. Respective accuracy curves in appendix.

In Figure 8 we can find the result of experiment 3. Clearly, the regularization components are increasing the training loss but reducing the validation loss so we will continue with this model.

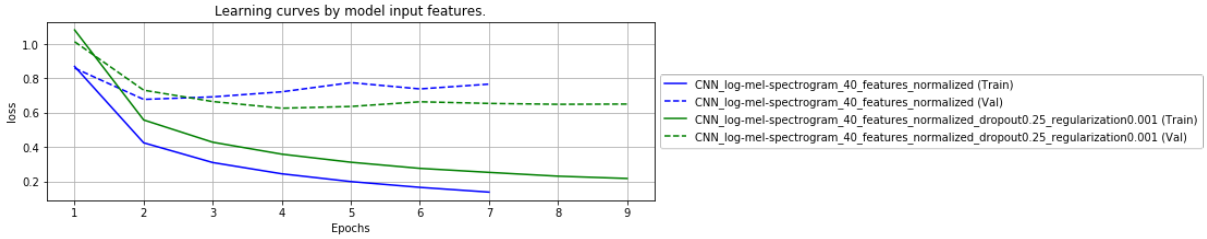

Figure 8: Loss curves of experiment 3. Respective accuracy curves in appendix.

In Figure 9 we can find the result of experiment 4. As we can observe, we do not find improvements by slowing down the speed.

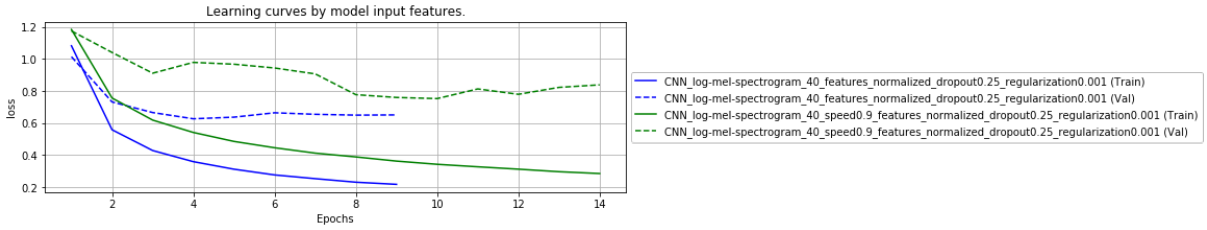

Figure 9: Loss curves of experiment 4. Respective accuracy curves in appendix.

In Figure 10 we can find the result of experiment 5. Clearly, increasing the number of LSTM units from 32 to 100 boosts the performance but increasing up to 200 not. LSTM with 100 and 200 follow a similar training curve. So we select 100 units as the best LSTM model. Increasing the LSTM units also increased the time required to train the model.

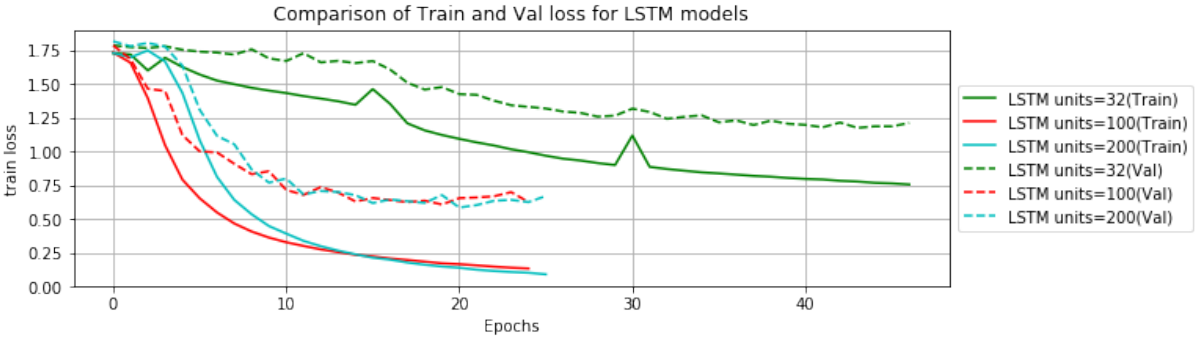

Figure 10: Loss curves of experiment 5. Respective accuracy curves in appendix.

7 Conclusions / Discussion

Firstly, according to the CNN experiments, we have seen that increasing the number of kept dimensions in the Log-Mel-Spectrogram does not improve the results as well as using MFCC features. On the other hand, the design of the preprocessing technique using hop splits like moving windows to get the same test features shape boosts the performance a bit. Moreover, adding a *Dropout* layer plus *Ridge* regularization reduce the overfitting of the model enabling higher test accuracy up to 81%. Finally, speeding down the input audio files does not improve our results.

Secondly, according to the LSTM experiments, a comparatively simpler model architecture than CNN gives decent accuracy results. A larger model learns faster but might overfit quickly. In short, LSTM has proved to be a very effective model in our experiments, although it takes a long time to train.

Overall, both CNN and LSTM were able to give similar accuracy but LSTM had simpler model architecture, hence less number of total trainable parameters. About the most difficult language to predict, we can see from Figures 10 and 12, which contain the confusion matrix for the best CNN and LSTM models, that Farsi is relatively the hardest language to recognize with an accuracy around 65% and Mandarin is way easier achieving an accuracy higher than 90%.

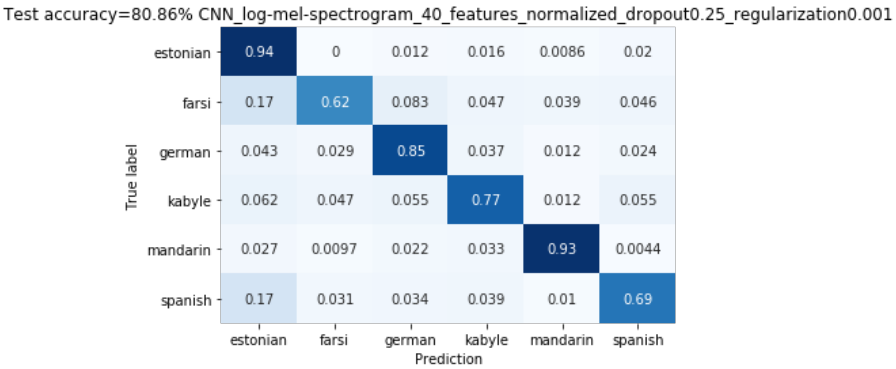

Figure 11: Confusion matrix of best CNN model in test set.

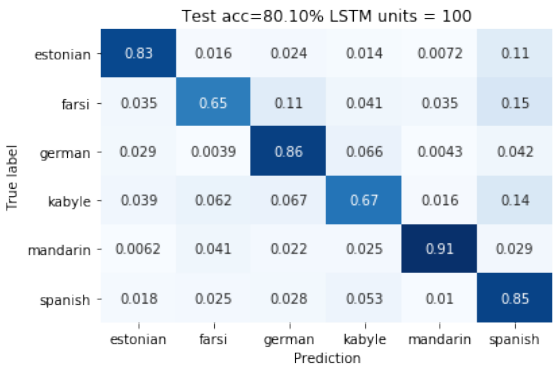

Figure 12: Confusion matrix of best LSTM model in test set.

8 Division of labor

Álvaro contributed to the literature study, development of project plan, report documentation, most part of the data preprocessing, and modelling all the experiments related to CNN. Similarly, Rachhek contributed to the literature study, report, partly with the preprocessing needed for LSTM experiments, and modelling all LSTM experiments. Both members jointly presented the findings of the project in the class on December 11 2019.

9 Acknowledgements

We would like to thank our lecturer and professor Mikko Kurimo for giving us the opportunity to work on this project. We are also really grateful to Matias Lindgren who guided us and patiently answered our questions every step of the way.

References

[1] H. Li, B. Ma, and K.-A. Lee, "Spoken language recognition: From fundamentals to practice," *Proceedings of the IEEE*, vol. 101, pp. 1136–1159, 2013.

[2] J. Gonzalez-Dominguez, I. Lopez-Moreno, and H. Sak, "Automatic language identification using long short-term memory recurrent neural networks," in *Interspeech*, 2014.

[3] Z. Tang, D. Wang, Y. Chen, L. Li, A. Abel, Z. Tang, D. Wang, Y. Chen, L. Li, and A. Abel, "Phonetic temporal neural model for language identification," *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, vol. 26, pp. 134–144, Jan. 2018.

[4] S. Shon, A. Ali, and J. Glass, "Convolutional neural network and language embeddings for end-to-end dialect recognition," pp. 98–104, 06 2018.

[5] N. Ackermann, "Introduction to 1d convolutional neural networks in keras for time sequences." https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf, 2018. [Accessed 15-Nov-2019].

[6] V. Bushaev, "Adam - latest trends in deep learning optimization." https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c, 2018. [Accessed 16-Dec-2019].

Appendix

Code

Please, find the code in the GitHub repository `www.github.com/alvarorgaz/Language-Speech-Recognition-with-CNN`.

Rest of figures

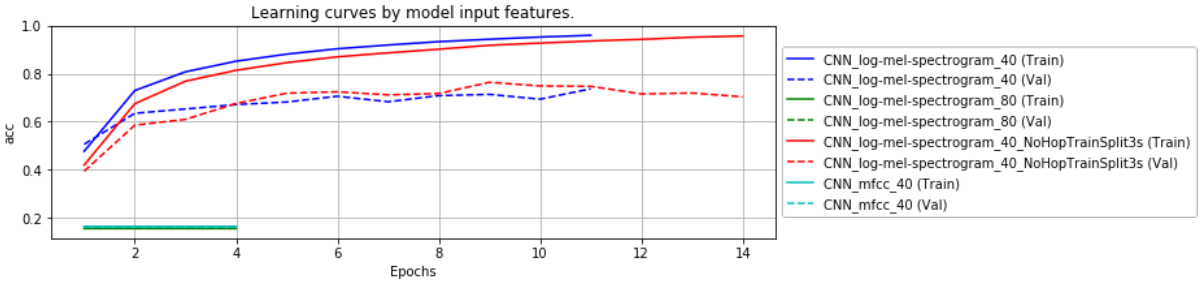

Figure 13: Accuracy curves of experiment 1.

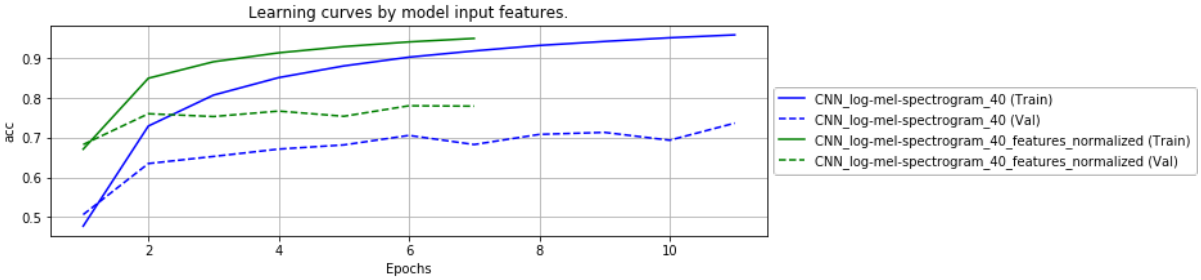

Figure 14: Accuracy curves of experiment 2.

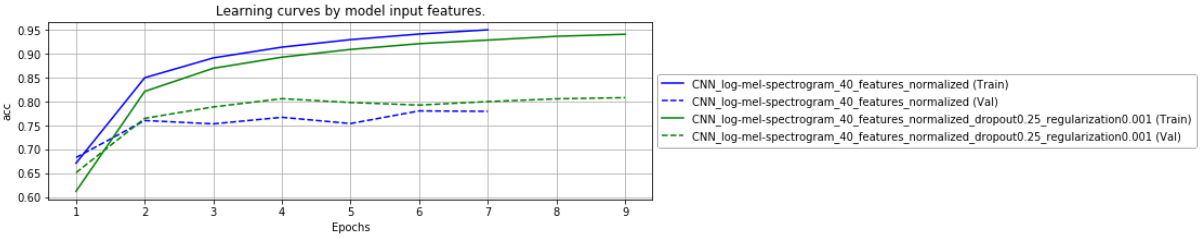

Figure 15: Accuracy curves of experiment 3.

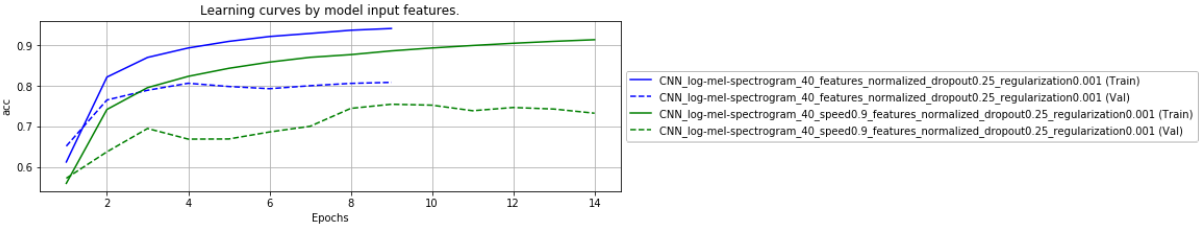

Figure 16: Accuracy curves of experiment 4.

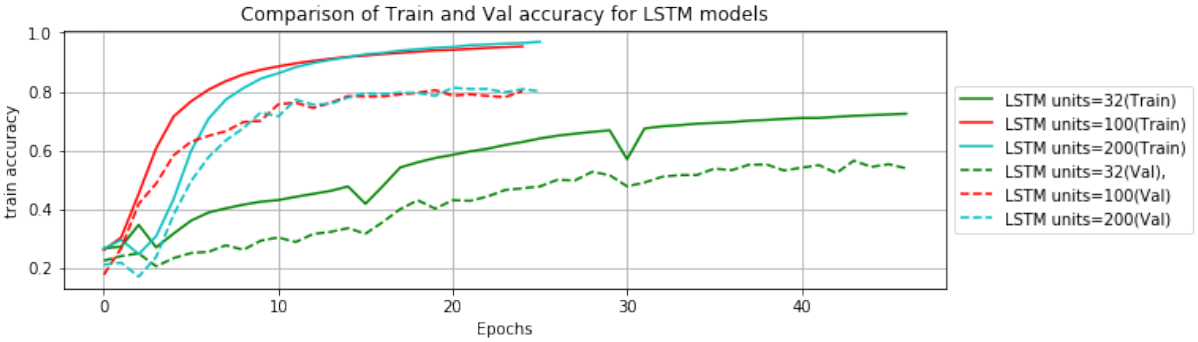

Figure 17: Accuracy curves of experiment 5.