Deep Learning Challenge: Speaker Identification

Kaya Alhan, Chiara Lovati, Ted Manders

March 2022

1 Introduction

This task aims at labeling each spoken utterance with the ID of one of the 183 unique speakers. Therefore we are solving a classification problem. The raw training data is in the form of WAVE files, which have to be processed to be usable as input to a deep learning model.

2 Deep learning architectures used

For this speaker identification task, two model architectures were implemented. A Convolutional Neural Network (CNN) based on a model by Doshi (2021) and an architecture with a combination of convolution and gated recurrent units (GRU) by Ye and Yang (2021). Convolutional layers are highly suitable for extracting abstract features from image data, which fits well with sound data as a common way to process sound data into a more usable format is creating a spectrogram or MFCC. These formats are ideal input for the aforementioned architectures as they can be processed like an image. The task can then be approached as an image classification problem.

The CNN model architecture has six convolution layers with Rectified Linear Unit (ReLU) activation and Kaiming Initialization. After passing through the convolution layers a max pooling layer is applied and finally input to the linear layer which outputs a prediction score for each of the 183 classes.

Fig. 1 shows the CNN architecture.

Layer	In	Out	Kernel	Stride	Padding
Conv1	2	8	5x5	2x2	2x2
Conv2	8	16	3x3	2x2	1x1
Conv3	16	32	3x3	2x2	1x1
Conv4	32	64	3x3	2x2	1x1
Conv5	64	128	3x3	2x2	1x1
Conv6	128	256	3x3	2x2	1x1
AvgPool	-	-	1xn	1x1	1x1
Linear	256	183	1x1	1x1	1x1

Figure 1. CNN architecture

The alternative model that was used consists of a convolutional layer combined with average pooling to extract features from the spectrograms which are subsequently fed into a stack of three gated recurrent units (GRU). The output of these units is passed into a fully connected layer with a final softmax layer providing the probabilities with respect to the prediction labels.

Fig. 2 shows a table of this architecture.

Layer	In	Out	Kernel	Stride	Hidden Units
Conv1	1	64	5x5	2x2	-
AvgPool	-	-	2x2	2x2	-
GRU1	64	256	-	-	256
GRU2	256	256	-	-	256
GRU3	256	256	-	-	256
Linear	256	256	-	-	-
Linear	256	183	-	-	-

Figure 2. Conv & GRU architecture

3 Training, hyperparameters and optimization

To input valid data to our models, pre-processing of the audio data was done according to the following steps: 1). Resample to 16kHz 2). Rechannel to 2 channels 3). Truncate or pad the signal to 4000 milliseconds 4). Generate a Mel spectrogram or MFCC 5). Convert to decibels (dB) (amplitude to dB). After pre-processing, augmentation of the data was experimented with to see if this would improve the model in terms of accuracy. Different techniques were implemented such as frequency- and time masking and time shifting. Fig 3. shows an example of a processed spectrogram. Fig. 4 shows an example of a processed MFCC.

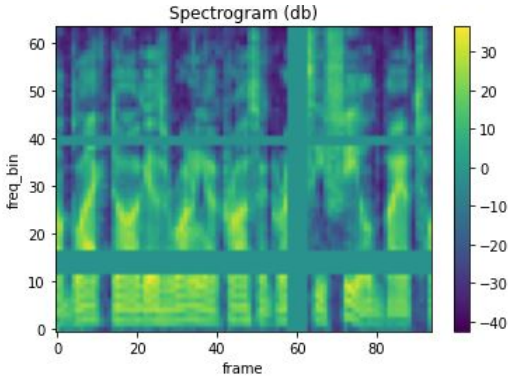

Figure 3. Example of a plotted processed spectrogram, showing frequency- and time masking.

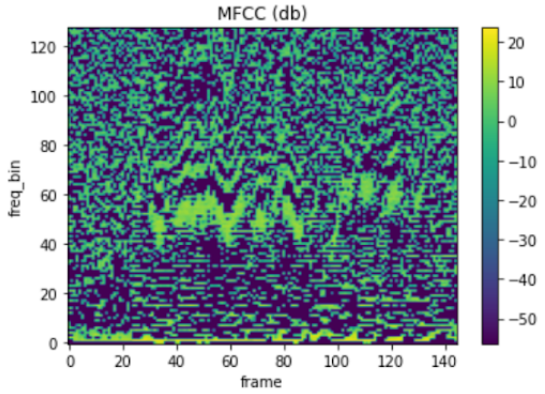

Figure 4. Example of MFCC used in the final model.

The most common sampling rate is 16kHz so all files are resampled the same. If a file only has one channel, the channel is copied as a second channel so all files have 2 channels. Every file is truncated or padded with zeros so they have the same length of 4000 milliseconds. Different lengths greater and smaller than 4000 were experimented with, where 4000 gave optimal results in terms of accuracy on validation. The spectrograms and MFCCs were generated with a cutoff of 80 dB.

During the exploration phase and hyper parameter tuning of the model the training data (30.000 files) was randomly split with 80% used for training and 20% used for validation. As the data is quite big in terms of total file size, a data loader is implemented to serve the model with smaller batches of data at a time.

The CNN model is fed batches of size 16 where preprocessing of data is done by the data loader during training. For this classification task cross-entropy loss is chosen as loss function, the Adam algorithm as optimizer (learning rate 0.001) and a learning rate scheduler was implemented to adjust the learning rate based on the number of epochs. Different model architectures were tested with 4, 5 and 6 convolutional layers, where 6 layers performed best. Different epoch settings were experimented with between 5 and 30 epochs. The optimal amount of epochs was found at 14 epochs, returning highest accuracy on validation data.

As for the Conv & GRU model, every layer, except for the output layer, is accompanied by Leaky ReLU activations. The model also includes several normalization layers, after each GRU. After the convolution, pooling and fully connected layer batch normalization is applied. A batch size of 128, and 30 epochs, were found to yield the best performance with this model.

4 The performance of our solution

Model performance was evaluated in terms of average loss and accuracy.

The best performing score of the CNN model returned an error rate of 0.0122. For a relatively simple CNN model, it performed well with an accuracy of 98.78%. The final model was trained with optimal parameters on the full training data to allow for an optimal training run on all available data. The best performing model was trained using MFCCs ($n\_mfcc = 128, n\_fft = 1024, n\_mels = 128, hop\_length = 512$), as they performed much better compared to the Mel spectrograms. For data augmentation, time shift was used as it improved accuracy. The other methods were discarded as they did not improve the results. The model architecture with 6 convolutional layers was used for the final model as it improved the accuracy the most. In addition to that, using more layers sped up the speed of learning significantly.

The alternative Conv-GRU model scored an accuracy of 95.55% with an adjusted batch size of 128. The lower performance of the model could be attributed to lower model size due to computational restraints or the nature of the dataset.

5 Codalab competition

The name of the account under which this group submitted their results to the competition on Codalab is *Haluqikite*.

6 Detailed specification of the work done by group members

- Ted researched and implemented the CNN model architecture. Including data pre-processing and augmentation methods for creating optimal input for the model. He also focused on parameter tuning the final CNN model and formatting the code for submission.
- Kaya investigated different model architectures. He further researched, developed and evaluated the final GRU model.
- Chiara explored data pre-processing and augmentation methods, and partially contributed to model comparison via further research and parameter tuning. She formatted the report.

7 References

Doshi, K. (2021) Audio Deep Learning Made Simple Sound Classification, Step-by-Step. Towards Data Science. *https://towardsdatascience.com/audio-deep-learning-made-simple-sound-classification-step-by-step-cebc936bbe5*

Ye, F., & Yang, J. (2021). A Deep Neural Network Model for Speaker Identification. Applied Sciences, 11(8), 3603.