ASR Project: Deep Learning Speech Classification and LMFCC improvements

Axel MAZA
Department of Computer Science
Kungliga Tekniska Högskolan
Stockholm, 114 28
axelmaza@kth.se

Victor SAILLANT
Department of Computer Science
Kungliga Tekniska Högskolan
Stockholm, 114 28
saillant@kth.se

Abstract

In this project, the use of LMFCC, Liftered-Mel-Frequency-Cepstrum-Coefficient, is explored for both classification purposes and synthesis. We first compare the influence of LMFCCs as features instead of raw signals. We shed light on the importance of preprocessing and how different processing such as padding or resizing of features have an influence on the performance of an architecture? The results on the classification task are promising even with raw signal and shows the value of deep learning in the field. We don't have any results regarding speech synthesis yet.

1 Introduction

Speech recognition is a technology that enable to capture words and information spoken by humans and makes its automatically understood by a computer. Historically, techniques such as dynamic-time-warped word template matching, linguistically inspired approaches using phonemes units, and hidden Markov models have been developed. In this latter method, the model is based on phonemes of target language, divided in states to make up the model. Using a decision tree and a maximum likelihood estimates to choose the right parameters of the HMM-model. Those models yield to satisfactory performances [3]. A common practice is to extract features like Mel-Spectrogram Cepstrum Coefficients (MFCC) on which the training is performed [12].

In this work, the use of recent deep learning approaches is explored. The study aims at first comparing influence of preprocessing for a classification task over a dataset of spoken words. Raw inputs as well as liftered Mel-Frequency Cepstrum Coefficient are fed in an adapted CNN and performances are compared. The second part of this works aims at using a more advance architecture developed lately: Variational-Auto-Encoder. The property of generation and speech synthesis is also considered in the scope of the project.

2 Methods

2.1 Convolutional Neural Network

Convolutional networks have seen a recent explosion in the field of machine learning with images. It adds up one or more convolutional layers to basic fully connected networks. A convolutional layer consist of various filters, masks that perform local operation on the image and leads to a newer modified version of this one with other information, such as gradients, borders, pattern. It thus allow sparse interaction, parameter sharing, and equivariant representation of the data [1].

2.2 Variational Auto Encoder

Among methods for classification and synthesis of signal such as images, Variational Auto-Encoder have shown their relevance [14]. They are also vastly used over 1D-signal cutting-edge technology such the Google Project Music-VAE [5]. They are also widely used in the field of Automatic Speech Recognition [8] [10]. It consist of a two pieces network with first an encoder reducing the dimensionality of the data and second a decoder to re-compose the input thanks to the generated latent space. Each of the pieces are multi-layered neural networks, with some convolutional layers. Now what makes a variational AE different than a regular one is the encoding of the latent space: whereas it is just fixed weight in the case of a regular Auto-Encoder, a Variational Encoder represent the latent space thanks to gaussian distribution of coefficient. Instead of optimizing the weights, the optimization of the means and variances of gaussian distributions are optimized. See figure 1.

Training a VAE consists then in the optimization of two losses, each one regarding a specific part of the network. The first one aims at tuning the parameters of the distribution in the latent space, making it the closest possible to the one given by the training set. It is measured using the Kullback-Leibler divergence. The second loss relates to the reconstruction (and than more generally to the synthesis) property of the network, it is the reconstruction loss. The resulting loss is a balance of the two previous losses, defined as following [8]:

$$||\Phi(\theta(x)) - x|| + KL(\mathcal{N}(\theta_\mu(x), \theta_\Sigma(x), \mathcal{N}(0, I)$$

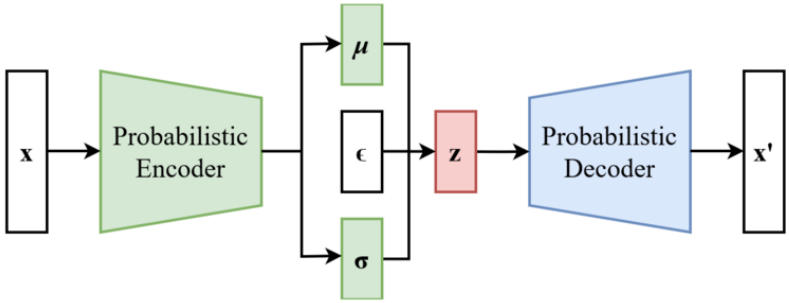

Figure 1: General Architecture of a VAE

2.3 The case of classification

The project falls in the case of classification. To perform classification over multiple classes such as, in this case, words like either "backward", "down" or another one, the networks needs to return a score over each class as an output, so that the most probable class is used as a prediction. Hence a loss function negative-log-likelihood can be used and the network has to have a softmax function in its output layer to predict a score over all of the different classes.

2.4 Feature Extraction and preprocessing

The data used is extracted from an open-source dataset described in section 3.1. This one consists of raw audio file in the WAV format. It could actually be used this way as in PyTorch official tutorial for the dedicated dataset [13]. In this work, the choice is made to compare this first approach with another one, using preprocessed featured called LMFCC (liftered Mel Frequency Cepstrum Coefficient). Several adaptation of the data have to be made for both cases [6].

2.4.1 Padding and resizing

All the signals don't have the exact same length or duracy. Any deep learning must be able to handle this kind of variation. In the case of raw inputs, shorter samples are padded with 0 values to make sure every signal is as long as the longest sample. This makes perfect sense as it allows not to deform the signal and keep the information without added noise: the 1D information is contained in the shape

of the waveform were it is deformed and adding 0 is just like adding silences. For computing LMFCC a different choice is made as padding could bring some border effects with the 2D-convolutional model described in section 2.4.2. In this situation, padding is tested as well as resizing, deforming the waveform to match the longest waveform in the dataset 3.

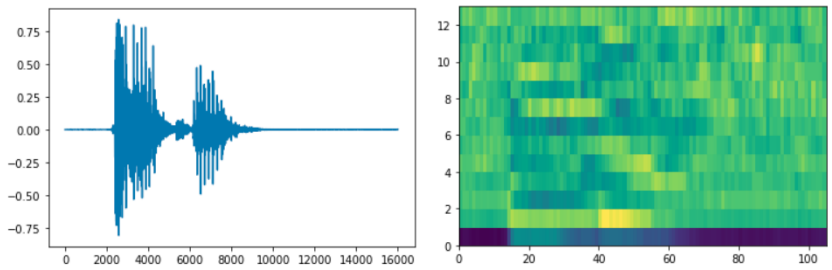

Figure 2: Raw signal and computed LMFFC for a recording of word "backward".

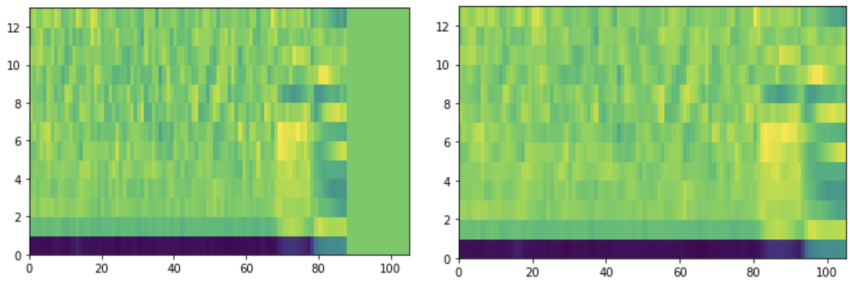

Figure 3: Example of result obtained after padding (left) and resizing (right) on the LMFCC.

2.4.2 LMFCC processing

The computation of LMFCC is made with basic Python code for better control of the data. Libraries such as Librosa [9] could also have been used but the choice has been made to re-implement it from scratch. Signals are first enframed into pieces of stacked next to each other, followed by a pre-emphasis filter. Hamming windowing is then used to avoid border effects and compute a power spectrogram based on the Fourier transform of each sample piece. The spectrogram is converted in log-scale using Mel-filterbanK log outputs. Finally, cepstrum coefficients are computed and lifter for a better scaling of the features. Parameters of the preprocessing steps have evolved during the process, final ones are given in section 3.2 in table 3. We end up with LMFCC features of shape 13*80.

Name	Description	Size
sampling rate		16 000
framing	window size	400 samples
	intersection	200 samples
pre-emphasis		0.97
windowing	window size	400 samples
nfft	Fourier transform size	512
nceps	Cepstrum coefficient	13
liftercoeffiient		22

Table 1: Preprocessing of LMFCCs

2.4.3 Normalization and downsampling

Both raw signals and LMFCCs have been normalized with mean zero and variance one to make the learning convenient for any kind of architecture. It is crucial in the case of neural networks so that optimization is well led. Down sampling has also been implemented to speed up the learning process.

This latter method was very useful over raw signal, however it turned out to compress to much the data for LMFCCs and it has been decided not to keep it.

3 Experiments

3.1 Dataset

The chosen dataset for the project is the Speech Commands dataset. It has about 100 000 sounds sample of recorded utterances of 35 English words spoken by several speakers. [15]. It is a rather lightweight dataset adapted for the scope of this course project. Examples of words are the following: "Marvin", "backward", "down", "left". The distribution of samples over classes is fairly distributed, the rarest class having 1557 records and the highest about 4052.

It is relevant to note that recording are noisy. They sound that they have been recorded in any regular situation thanks to a phone for example. It makes the application made out of this dataset dedicated to this kind of devices.

3.2 Parameter Settings

The choice has been made to use Python along with PyTorch for this project for implementation of Neural Networks and training [11].

Layer	Parameters	
Conv1	(ninput=1, nchannels=32, kernelsize=80)	(1D)
BatchNorm1	(nchannels=32)	
ReLu		
MaxPool1	(size=4)	
Conv2	(ninput=32, nchannels=32, kernelsize=3)	
BatchNorm2	(nchannels=32)	
ReLu		
MaxPool2	(size=4)	
Conv3	(ninput=32, nchannels=64, kernelsize=3)	
BatchNorm3	(nchannels=64)	
ReLu		
MaxPool3	(size=4)	
Conv4	(ninput=64, nchannels=64, kernelsize=3)	
BatchNorm4	(nchannels=64)	
ReLu		
MaxPool4	(size=4)	
FullyConnected	(ninput=64, noutput=35)	
batch size	256	
learning rate	0.01	
optimizer	Adam	

Table 2: Learning parameters for signal based CNN

3.3 Evaluation

In order to evaluate our different models, we will use both the recall (equation number 1), the precision (equation number 2) and the F1-score (equation number 3).

Recall is the number of retrieved relevant items as a proportion of all relevant items. Recall is, therefore, a measure of effectiveness in retrieving (or selecting) performance and can be viewed as a measure of effectiveness in including relevant items in the retrieved set [2].

$$Recall = \frac{TP}{TP + FN} \tag{1}$$

Layer	Parameters
Conv1	(ninput=1, nchannels=32, kernelsize=(3,5))
BatchNorm1	(nchannels=32)
ReLu	
Conv2	(ninput=32, nchannels=32, kernelsize=(3,5))
BatchNorm2	(nchannels=32)
ReLu	
MaxPool1	(size=(2,2)
Conv3	(ninput=32, nchannels=64, kernelsize=(3,5))
BatchNorm3	(nchannels=64)
ReLu	
Conv4	(ninput=64, nchannels=64, kernelsize=(3,5))
BatchNorm4	(nchannels=64)
ReLu	
MaxPool2	(size=(2,2)
Conv5	(ninput=64, nchannels=64, kernelsize=(3,5))
BatchNorm5	(nchannels=64)
ReLu	
Conv6	(ninput=64, nchannels=64, kernelsize=(3,5))
BatchNorm6	(nchannels=64)
ReLu	
Dropout	(rate=0.25)
FullyConnected	(ninput=3648, noutput=128)
FullyConnected	(ninput=128, noutput=35)
batch size	256
learning rate	0.001
optimizer	Adam

Table 3: Learning parameters for LMFCC based CNN

It answers the question "how effective is the results?". Another way of understanding it is: the higher the recall of a model, the more confident it is not to miss any target sample. It can however have a lot of false positive.

Precision is the number of retrieved relevant items as a proportion of the number of retrieved. Precision is, therefore, a measure of purity in retrieval performance, a measure of effectiveness in excluding non-relevant items from the retrieved set [2].

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

It answers the question "how pure is the result?". Another way of understanding it is: the higher the precision of a model, the more confident it is when detecting a positive sample not to make a mistake. It can however miss a lot of the object to be detected.

A high precision implies a good purity but fewer detections, whereas a high recall corresponds to a high detection rate but more false positives. Hence, there is a trade-off between the precision and recall for having a good model. This trade-off can be describe by the harmonic mean of those two values, also known as the **F1-Score** [4]. It is thus by definition given by the following formula:

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3}$$

4 Results

Results of signal-based CNN as well as two version of LMFFC-based CNN are presented in this section. The development of a Variational Auto-Encoder was too much time consuming for the scope of the project and is kept as a further and promising work, that could as well be applied for command speech synthesis.

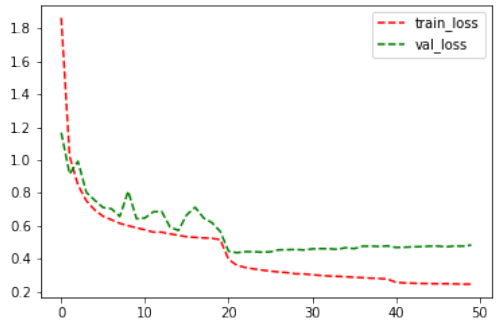

Figure 4: Signal based architecture, training and validation loss during training using resizing.

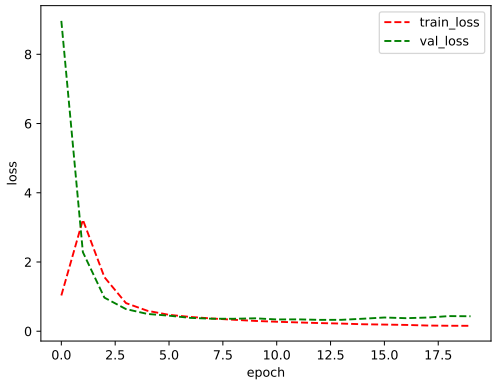

Figure 5: LMFFC architecture, training and validation loss during training using padding.

4.1 Loss curves

4.2 Final metrics

Model	Precision	Recall	F1-score
Signal based	93.25 %	93.17%	93.18%
LMFCC based using padding	94.39%	93.45%	93.92%
LMFCC based using resize	94.84%	93.87%	94.35%

Table 4: Metrics computed over the test set

5 Discussion

5.1 Loss curves

Thanks to the loss curves in section 4.1, one can see that the two learning curves of models built with Liftered Mel Frequency Cepstrum Coefficients, figures 5 and 4, are similar regardless of the preprocessing method used (feature resizing or zero padding). The validation curve stop decreasing and both models starts over-fitting from the 8th epoch. The choice is made for the final model to have only 8 epochs of training.

We can however see that the validation curve for the version with resized samples goes up with a steeper slope compared to the padding-based version. This can be explain by the fact that padding

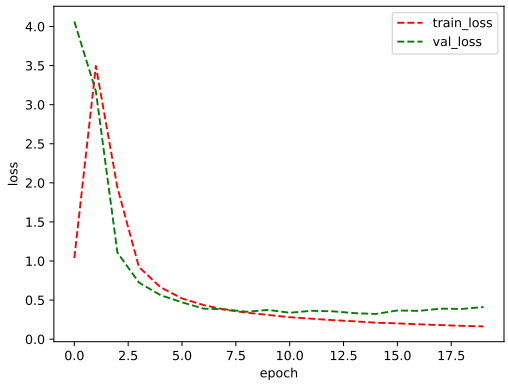

Figure 6: LMFFC architecture, training and validation loss during training using resizing.

adds external information to the data and make the dataset less homogeneous. It is thus slightly harder to overfit it due to this preprocessing step.

On the other side, the loss curve of the signal-based method, in figure 6, acts differently. The overfitting point appears much later around the 20th epoch. The choice is made to train the final signal-based architecture to this point. This phenomenon shows that computing LMFCC is somehow adding information to the signal and thus the architecture needs less training, which justify it's massive use in Automatic Speech Recognition technologies [7].

5.2 Final metrics

Final results are presented in table 4. We can see that all models presents balanced performances in precision and recall. A minor difference between the signal-based architecture and the LMFCC ones is that the first one has a higher precision whereas LMFCC-CNNs have higher recall values.

LMFCC architectures performs better than the signal based one. This confirms what has been introduced in section 5.1: LMFCC as a widely used preprocessing step in the field of A.S.R. [7] and bringing additional information to the feature space.

Comparing the two LMFCC-based Convolutional Neutral Networks, using resizing in the preprocessing seems to be better than using padding. This can be explained as padding is an external added information that somehow add noise to the training as the information to recognize in each sample does not depend on the length but only on the shape of the signal when the command is spoken.

6 Conclusion

To sum up the study, building two comparable CNN based either on raw signal or Liftered Mel Frequency Cepstrum Coefficients, it turns out that LMFCC are indeed an improvement when it comes to automatic speech commands classification. It adds up information before any step of optimization. Several preprocessing methods have been compared.

Padding is highly relevant in the case of a 1D signal and ensure that there is no deformation of the signal It could however bring some border effects to a 2D convolutions-based architecture. It is faster to implement but it adds up external noise to the feature and lower the performances. Resizing the signal so all the input can be feed to a Convolutional Neural Network with the same size seems to be a better solution for LMFCC.

For further works, it would be relevant to better tune the CNN by trying more settings in terms of size and number of layers in the architecture. It would also be interesting to finish the implementation of a Variational Auto-Encoder which shows promising results and would allow us to go one step further with speech generation.

References

[1] Laith Alzubaidi et al. “Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions”. In: *Journal of big Data* 8.1 (2021), pp. 1–74.

[2] Michael Buckland and Fredric Gey. “The relationship between recall and precision”. In: *Journal of the American society for information science* 45.1 (1994), pp. 12–19.

[3] Rupali S Chavan and Ganesh S Sable. “An overview of speech recognition using HMM”. In: *International Journal of Computer Science and Mobile Computing* 2.6 (2013), pp. 233–238.

[4] Davide Chicco and Giuseppe Jurman. “The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation”. In: *BMC genomics* 21.1 (2020), pp. 1–13.

[5] Junyan Jiang et al. “Transformer vae: A hierarchical model for structure-aware and interpretable music representation learning”. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 516–520.

[6] Lauri Juvela et al. “Speech waveform synthesis from MFCC sequences with generative adversarial networks”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 5679–5683.

[7] A Lawson et al. “Survey and evaluation of acoustic features for speaker recognition”. In: *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2011, pp. 5444–5447.

[8] Matteo Lionello and Hendrik Purwins. “End-To-End Dilated Variational Autoencoder with Bottleneck Discriminative Loss for Sound Morphing–A Preliminary Study”. In: *arXiv preprint arXiv:2011.09744* (2020).

[9] Brian McFee et al. “librosa: Audio and music signal analysis in python”. In: *Proceedings of the 14th python in science conference*. Vol. 8. Citeseer. 2015, pp. 18–25.

[10] Aaron van den Oord et al. “Wavenet: A generative model for raw audio”. In: *arXiv preprint arXiv:1609.03499* (2016).

[11] Adam Paszke et al. “Automatic differentiation in PyTorch”. In: (2017).

[12] Mahdi Shaneh and Azizollah Taheri. “Voice command recognition system based on MFCC and VQ algorithms”. In: *World Academy of Science, Engineering and Technology* 57 (2009), pp. 534–538.

[13] *SPEECH COMMAND CLASSIFICATION WITH TORCHAUDIO*. `https://pytorch.org/tutorials/intermediate/speech_command_classification_with_torchaudio_tutorial.html`. Accessed: 2022-05-20.

[14] Jakub Tomczak and Max Welling. “VAE with a VampPrior”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2018, pp. 1214–1223.

[15] P. Warden. “Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition”. In: *ArXiv e-prints* (Apr. 2018). arXiv: `1804.03209 [cs.CL]`. URL: `https://arxiv.org/abs/1804.03209`.

Peer review feedback

Changes made after the peer-review:

- Added additional graphs for clarity,
- Added several citations for explanation,
- Better labelling of graphs and especially learning curves,
- Gave more explanation on CNN,
- Tried a deeper architecture on LMFCC.

Changes not made after the peer-review:

- We did not add as much background as recommended to keep the report tidy and clear,
- We have not expanded the study to VAE, due to time limitations.