Phoneme Classification Using Transfer Learning from Image Classification

Axel Joigneau
SCR Student
KTH
joigneau@kth.se

Sylvain Potuaud
SCR Student
KTH
potuaud@kth.se

Abstract

With the development of the Deep Learning applied to the Automatic Speech Recognition, many methods and techniques are explored in order to get the best performance possible. Since the sound is characterized by its frequencies, what about analyzing the speech as an image, using its spectrum representation ? In this project we try to classify different phonemes using Transfer Learning. A relatively high performance is achieved since our classifier reaches 82% accuracy with different phonemes.
Andrew Ng who is chief scientist at Baidu and professor at Stanford, said during his NIPS 2016 tutorial that transfer learning will be the next driver of ML commercial success[1]. This made us curious to know how far transfer learning could work.

1 Introduction

Our first project idea was to use Speech Recognition as a learner for foreign languages. Here is the paper that interested us first [2]. The goal was to create a kind of game whose target are young non-native speakers that will learn with a network trained from adult native speakers speeches. For this idea we wanted to focus on phonemes pronunciation, but the topic was already well covered by other projects, so we decided to try something new.

Numerous research projects have been carried out on Deep Learning, for both Speech and Image Recognitions. But what about mixing these two ? Since we can represent speeches as spectral images, it seems possible to do image recognition on the spectral representation in order to classify sounds. In this paper [3] Daniel Nouri worked for the Kaggle Whale Detection Challenge, which asked competitors to classify two-second audio recordings, some of which had a certain call of a specific whale on them, and others didn't. His work is actually similar to ours since he decided to frame the problem of finding the whale sound patterns as an image recognition problem, by turning each two-second sound clips into spectrograms. A similar method has also been used for Emotion Prediction in this paper [4]. They also use a network based on convolutional layers and trained with spectrum pictures, and they reached 93% accuracy for anger, which shows that using FFT spectrums can be efficient. Their dataset and goal are different though : since they want to classify emotions, they might pay more attention to speech tones than to speech pronunciations.

2 Method

Transfer learning is a technique that takes a deep fully-trained model, and retrains it for new classes. This network, based the Inception-v3 model[6], contains mainly convolutional layers, but also pooling layers and a fully-connected one at the end.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

We have used TensorFlow because it's the most adapted and documented library to work with the Inception model.

2.1 The Network

It was revealed that a lot of applications make use of convolutional networks, especially for visual recognition [5]. Actually, when dealing with high-dimensional inputs such as images, it is impractical to connect neurons to all neurons of the previous layer. The high number of parameters would be synonymous with overfitting. Instead, it is better to connect each neuron to only a local region of the input volume, which is the general difference between a convolutional layer and a fully connected layer. A convolutional layer operates on a part of the data, instead of all of it, and generally uses multiple filters, which are matrix operator used to calculate the output of a convolutional node. The filter moves across the image and multiplies the input values by the filter's value to produce the output. The mechanism is similar to a Sobel operator in an edge detection process. Typical sizes of the filter is 3x3 or 5x5.

Generally, a convolutional layer consists of many filters, which means that every filter operation is done multiple times at each position in the image, using filters of the same size, but with different values. This means that the output of each filter will differ, depending on the input features, which enables a convolutional layer to store specific information about image features in each filter. In theory this enables a filter to "learn" specific features that later layers can use do determine the image class.

In addition to a convolutional layer, a pooling layer is used in order to affect the output of a convolutional layer. The pooling layer is a layer that reduces the size of the convolutional layer, by using an area max search, which is a fixed size area moving across the input, outputting the largest value in that area to the output. If the area has size 2x2, and moves with a stride of 2 over the input data, the output data will be divided in height and width by 2.

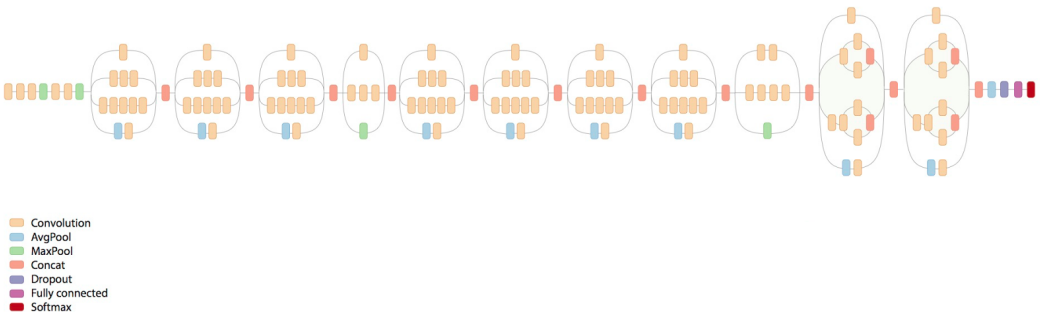


Figure 1: Schematic diagram of Inception-v3

2.2 Training the last layer

In this paper we will focus on a very limited retraining consisting in retraining only the last layer from scratch. This corresponds to use the values of the last layer of the deep network as features to train a simple one layer network. This has been proven to be efficient, especially in image classification. Features obtained thanks to this process perform better than hand-engineered features on image classification[4].

Of course the results of this method cannot be better than the results of a fully trained on the specific task deep neural network, because there are much less parameters that we can adjust. But it is much faster and requires fewer labeled data, which makes it still interesting.

Trying to use a network trained to recognize images of object in order to recognize images of speech is very challenging. That is why our goal is not to outperform current state-of-the-art phoneme recognition methods, but rather to see if this method is usable or totally ridiculous.

As previously said, the Inception-v3 model is used in this paper as a starting point for the transfer learning. This model is one of the best models in image classification, it matches or exceeds human performance in some domains. It has a 21.2% top-1 and 5.6% top-5 error rate[6]. It has been trained over 1000 classes using data from ImageNet[7].

For the transfer learning we take the 2048 features just before the final softmax layer of Inception-v3. We use them as input to train one layer to classify 21 new classes (21 phonemes).

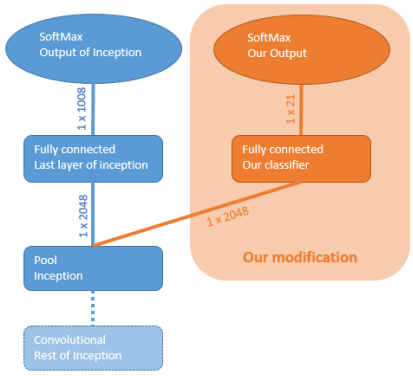


Figure 2: Last layer added to Inception-v3 for the transfer learning to 21 new classes

Another important aspect of this project was to generate images from the speech. On this aspect there are many possibilities. Our goal was to test several ones to see which ones are the most efficient.
The idea is to use the same process used to create the usual hand-engineered features of speech recognition, the MFCC, but to stop earlier. In fact the interest of having a very deep network is that it doesn't need such features.
Even when what should be represented as a picture is fixed, there still need to choose some parameters such as the size of the picture and how to use the colors.

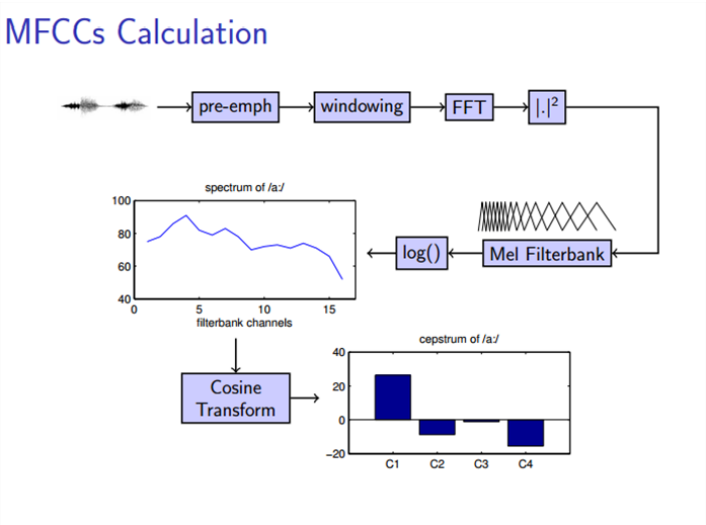


Figure 3: MFCC Steps[8]

3 Experiments

For the experiments we have used the TIDIGITS database [9] because we are familiar with it and it is adapted to our needs. This database contains speech that was collected by Texas Instruments. There are 111 men speakers and 114 women speaker each pronouncing 77 digit sequences. They are divided approximately half and half into test and training subsets.

When doing digit recognition we have used 22 speech files for each speaker : 2 repetitions of 11 isolated digits ("oh", "zero", "one", "two", "three", "four", "five", "six", "seven", "eight" and "nine").

For the purpose of phoneme recognition, we need a division of these speech files into phonemes. For that we have used the time aligned transcriptions of the phonemes create in lab 3 using a GMM-HMM model trained with the same database.

Some phoneme are a lot more present in this database than some other. We have tried to get around 500-600 pictures for each phonemes to train our model. This reduction is done by fixing a proportion of each phoneme that should be processed, the phonemes we have used are equally distributed among the dataset to keep its diversity. The phoneme 'ey' is only present 210 times in the training data so we cannot use more images.

Table 1: Training phonemes files

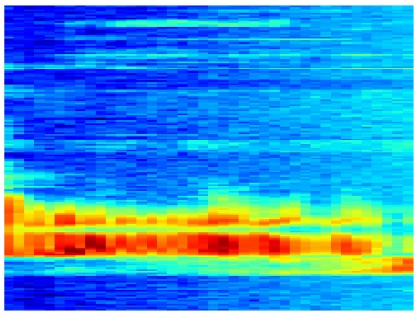
Phoneme	'ah'	'ao'	'ay'	'eh'	'ey'	'f'	'hh'	'ih'	'iy'	'k'	'n'
Proportion used	1/8	1/2	1/2	1/4	1	1/8	1/4	1/3	1/6	1/4	1/11
Number	521	553	740	631	210	505	524	638	614	602	595

Phoneme	'ow'	'r'	's'	'sil'	't'	'th'	'uw'	'v'	'w'	'z'
Proportion used	1/2	1/10	1/10	1	1/6	1/2	1	1/7	1/4	1/3
Number	753	536	534	630	577	772	892	559	513	563

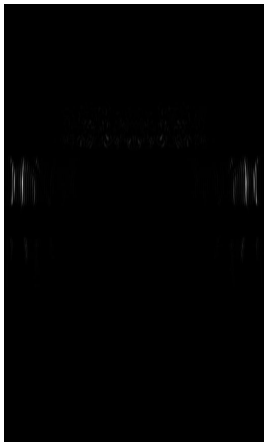
4 Results

4.1 Digits

The first results we got were on digits recognition and not on phoneme classification. We started with that because it was easier and faster to have a first result to see if the idea of transfer learning from image to speech can work.
We try with two different kind of pictures, the output of the MelFilterBank or directly the FFT.



(a) FBANK



(b) FFT

Figure 4: Different image representation of one digit.

As it can be seen on Table 2 the results are better directly with FFT, that is why we have decided to use FFT in our other tests.
This is not very surprising since a very deep network tends to prefer working with data closer to the

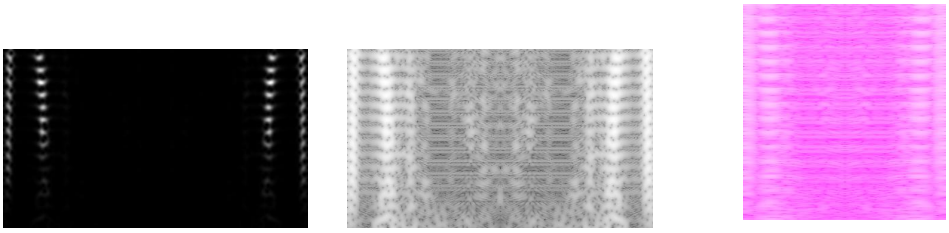
Table 2: Results on digits recognition

Input Image	Success Rate
MelFilterBank	77%
FFT	87%

'raw' data. But this result is interesting because it's different than the one observed with networks of a few layer like in Lab 3.

4.2 Phonemes

Then we apply the same method to phonemes, we also have a lot of parameters to choose to generate the images. We have try several ones : the FFT, the logarithm of the FFT, and both the FFT and the logarithm of the FFT on the same image using colors.



(a) FFT

(b) log(FFT)

(c) RGB with red values = FFT, green values = log(FFT) and blue values = constant

Figure 5: Different image representation of one phoneme.

Table 3: Results on phoneme recognition

Input Image	Success Rate
FFT	82%
log(FFT)	79%
FFT and log(FFT)	74%

The images of the FFT are the ones performing the best. We can try to explain that by the fact that the logarithm emphasizes the noise and reduces the contrast of the strong components of the FFT. The color picture having the FFT on its red canal and the log(FFT) on its green canal is not performing well either. This surprised us at the beginning because this image contains all the informations of the FFT. But the fact is that the first layers of the network dealing with the 3 color canals are not retrained during our training, so they know how to interpret relation between colors in objects pictures but not in our speech images.

4.3 Final Results

Here are more detailed result of our best model using the FFT of the phonemes.

4.3.1 Learning

We have trained during 10 000 epochs, we can see on the curve on Figure 6 that adding more epochs will not significantly improve the results.

The difference between the training error and the validation error is not too big, that shows that we don't have too much overfitting.

The learning takes about 1h20 on a classical laptop running on CPU (Intel i5) with 12.500 images and 10.000 epochs. It takes about 1h to compute the images features using Inception-v3 and 20min

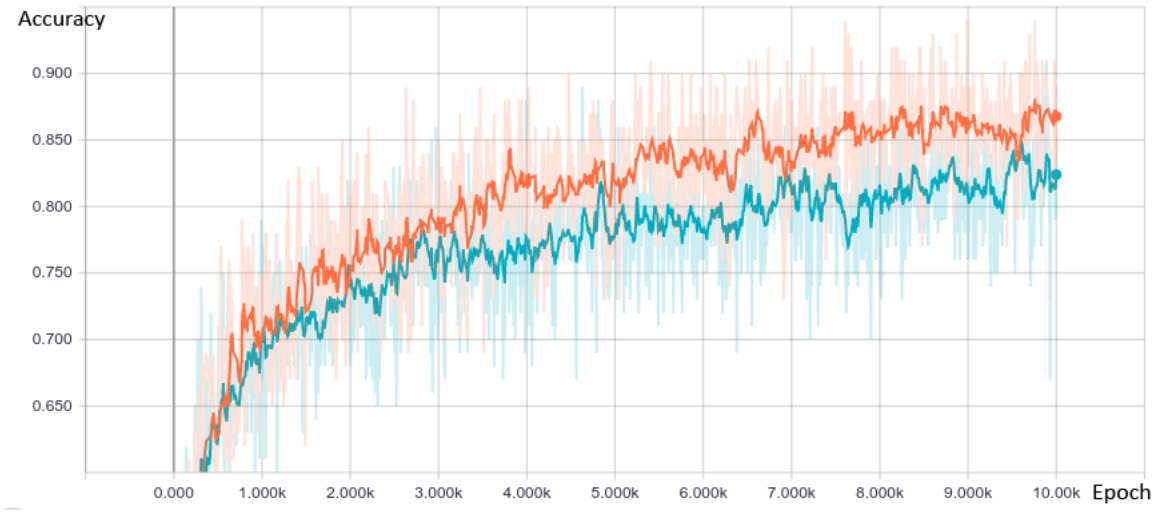


Figure 6: Accuracy depending on the epoch, orange : training, blue : validation

to train the one layer classifier. This time can of course been highly reduced by using the GPU. The speed is an interesting aspect of our model when we compare it with the Lab3 model.

4.3.2 Test

	output class																						
		ah	ao	ay	eh	ey	f	hh	ih	iy	k	n	ow	r	s	sil	t	th	uw	v	w	z		total	success rate
true class	ah	356	0	7	3	10	0	9	1	11	1	13	21	7	0	0	0	0	15	36	6	1	ah	497	0.72
	ao	0	509	1	0	2	0	18	0	2	0	0	10	13	0	0	0	0	4	0	26	0	ao	585	0.87
	ay	13	8	616	0	51	0	0	0	23	0	0	12	21	0	0	0	0	0	6	44	0	ay	794	0.78
	eh	0	0	0	559	0	2	2	29	0	1	1	0	0	3	0	5	8	1	0	0	5	eh	616	0.91
	ey	6	0	20	1	158	0	0	3	33	0	1	4	2	0	0	1	1	14	5	4	1	ey	254	0.62
	f	0	0	0	2	0	422	2	1	0	11	1	0	1	14	26	8	20	0	0	0	4	f	512	0.82
	hh	4	12	11	2	3	9	409	3	6	3	16	7	18	1	0	0	5	3	1	7	1	hh	521	0.79
	ih	7	0	0	52	16	0	1	530	2	2	5	0	1	2	1	3	3	33	8	0	7	ih	673	0.79
	iy	24	2	17	0	68	0	3	2	411	0	7	7	11	0	0	0	0	24	12	11	0	iy	599	0.69
	k	0	0	0	0	0	6	0	0	0	575	5	0	1	0	0	16	1	0	1	0	3	k	608	0.95
	n	45	2	0	4	10	1	15	4	28	5	428	14	14	0	0	6	1	10	14	3	1	n	605	0.71
	ow	59	24	20	2	6	0	15	1	24	0	26	416	71	0	0	3	2	5	49	19	1	ow	743	0.56
	r	8	8	20	1	5	1	8	0	10	0	15	58	370	0	1	1	1	4	7	13	0	r	531	0.70
	s	0	0	0	0	0	9	0	0	0	0	0	0	0	483	1	5	2	0	0	0	24	s	524	0.92
	sil	0	0	0	0	0	26	0	0	1	5	2	0	0	7	593	11	7	0	0	0	4	sil	656	0.90
	t	0	0	0	2	0	8	0	0	0	45	11	0	0	7	20	481	15	0	1	0	1	t	591	0.81
	th	0	0	0	5	0	41	4	7	0	2	2	0	0	6	5	18	632	1	1	0	19	th	743	0.85
	uw	17	6	2	2	55	0	11	19	24	0	13	1	3	0	0	0	0	793	13	7	0	uw	966	0.82
	v	41	0	5	1	5	1	1	1	8	1	10	33	3	2	1	1	3	2	460	1	2	v	582	0.79
	w	8	25	16	0	8	0	3	0	24	0	2	6	14	0	1	0	0	1	0	413	0	w	521	0.79
	z	2	0	0	7	1	17	1	2	1	0	1	0	0	23	3	7	12	0	0	0	460	z	537	0.86
																					total correct :			10074	
	total																							12658	0.80

Figure 7: Confusion matrix.

Of course the confusion matrix is almost diagonal. But sometimes we have relatively bad results for some special phonemes. As an example, the phoneme "ow" has only 58% accuracy while the phoneme k reaches 95%. This can be explained because the phoneme "ow" is sometimes confused with the phoneme "r" (in the word "four" for example), which is in accordance with the similar FFTs we get for the phonemes "ow" and "r", in the Figure 8.

(a) FFT of the phoneme "ow"

(b) FFT of the phoneme "r"

(c) FFT of the phoneme "k")

Figure 8

5 Discussion and Conclusions

Our results are weak compared to state-of-the-art phoneme recognition techniques, but our method seems at least quite fast and does not require a lot of data. Our method might then be adapted to some particular situations, and can surely be improved. Actually, as we previously said, it seems to work better on emotion recognition [4], which can possibly be explained by the FFT that contains mainly information about amplitudes, which is adapted for tones recognition.

One of the limits of our current method is that it is classifying the entire phoneme. This supposes that we know the limit of each phoneme which is usually not the case. Using a fixed length for generating the images can be a solution. Another limit comes from the dataset, due to the limited number of words the phonemes are present in a very limited number of contexts. Using a more various dataset will be better.

Our method can be improved by finding a way to discriminate the phonemes on which we have bad results. Many changes on the images are possible, color can be used to convey additional information even if we have seen that it does not alway improve the results.

References

[1] Sebastian Ruder (2016), Why Transfer Learning Now? : http://sebastianruder.com/transfer-learning/index.html#whytransferlearningnow

[2] Adapting Automatic Speech Recognition for Foreign Language Learners in a Serious Game : https://www.aaai.org/ocs/index.php/AIIDE/AIIDE14/paper/download/9080/9037

[3] Using deep learning to listen for whales : http://danielnouri.org/notes/2014/01/10/using-deep-learning-to-listen-for-whales/

[4] Speech Emotion Recognition from Spectrograms with Deep Convolutional Neural Network, Abdul Malik Badshah, Jamil Ahmad, Nasir Rahim, Sung Wook Baik, 2017 International Conference on Platform Technology and Service (PlatCon) : http://ieeexplore.ieee.org/document/7883728/

[5] Convolutional Neural Network for Visual Recognition http://cs231n.github.io/convolutional-networks/

[6] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna (2015) Rethinking the Inception Architecture for Computer Vision

[7] ImageNet : http://image-net.org/

[8] Giampiero Salvi (2017) DT2119 Speech and Speaker Recognition, KTH Course

[9] TIDIGITS, Linguistic Data Consortium : https://catalog.ldc.upenn.edu/LDC93S10