Verb recognition using a convolutional neural network and extreme learning machine

Course Coordinator
Josephine Sullivan

Neural Speech group
Carl Ridnert
Michel Postigo Smura
Óttar Guðmundsson

Abstract: This project aims to explore classifying audio files of simple verbs using Convolutional Neural Networks (CNN) and Extreme Learning Machines (ELM). The CNN layers are pre-trained with fully connected layers using ten classes, the convolutional layers are then extracted and an ELM is fitted at the end of the network. This new network is thereafter trained on twelve other classes from the same dataset. The network is benchmarked against an untrained Multilayer Perceptron (MLP) fitted on the same pre-trained convolutional layers. The results shows that the CNN + ELM architecture works well and trains very fast on new datasets. We define a new measure of the performance, named *Accuracy per Seconds* (APS) which is suited to get an overview and compare the performance of similar models. Lastly we proceed to conclude that the CNN+ELM outperforms the CNN+MLP network in in terms of this measure.

Table of contents

1. Introduction

Convolutional Neural Networks (CNN) have made huge improvements on image recognition in the last years, scoring over 90% accuracy on correctly classifying datasets of pictures [1]. The main reason behind the success of CNN's in comparison to standard Multi Layer Perceptrons (MLP) lies in the formers capability of extracting relevant features in the images by performing transformations of the data using filters and pooling operations.

With the background of the advances in speech-recognition applications over the last few years, our team was intrigued about the idea of researching whether or not they are suitable for classifying audio. Audio files can be drawn as simple spectrograms images which means that the task can be cast into the framework of image classification. One application that sparked an initial interest was the use of the network as an aid in changing slides during presentations. The network could be trained on certain keywords and each word would be mapped with a slide. When the network would listen to an audio stream, it could pick up a certain word while being on a specific slide, triggering the next one. This would require the network to be able to adapt and learn new words for each specific presentation, having the user retraining it with his own specific keywords.

One major issue with CNN's and deep networks in general is the computational effort of finding local minima for the defined loss function, especially as the network becomes deeper. In order to use a network for different classes or a new data-set the network has to be re-trained, this can be a very time-consuming process. In this paper we aim to investigate more flexible approaches to this problem. We look in to the method of pre-training the network on a dataset, then extracting the convolutional layers and fitting so called Extreme Learning Machines (ELM) which takes inputs from the convolutional layers and regular MLP's. The final fully connected layers are then trained on new data with new classes while keeping the convolutional layers fixed. Our problem can be summarized and formulated as the following question:

Can a pre-trained network together with an ELM or MLP be generalizable and perform well in audio classification?

2. Background

There are many different papers on using neural networks for image classification. With the differences mainly laying in the datasets and architecture of the networks. There are various tutorials on how to implement a MLP for classifying images. The accuracy on the CIFAR-10 dataset for these networks can reach as high as about 60% with pre-training the layers in an unsupervised manner [2]. Another approach is to use support-vector machines (SVM) for image classification combined with some feature extractor, a previously common approach was an SVM combined with an Local Binary Pattern (LBP). However this network became cast into shades after the introduction of CNN's [3].
CNN's has proven to outperform classical deep neural networks with far fewer parameters. In 2012, a deep convolutional neural network achieved a huge improvement in the ImageNet challenge and the winner of 2017's ImageNet challenge BDAT also used a CNN [4]. The current state-of the art image classification networks are based on convolutional neural networks.

As previously mentioned, the most significant problem for deep CNN's are the computational costs for training. Even if the number of parameters is drastically smaller for convolutional networks compared to similar MLP's the training times can in some cases be infeasible long, especially for adaptive learning where the network learns along the way as users provide new inputs.

In 2004 The Extreme Learning Machine (ELM) was introduced as a faster to train and better generalizable alternative to backpropagation in a one-layer neural network [5]. It works by using a least squares approach to the classification problem, we give an overview of the method in chapter 3.3. The learning speed to the ELM is extremely fast compared to a single layer networked trained by backpropagation, in some cases several thousands of times faster [5]. Thus, this kind of network could be attractive to use as the last fully connected layer of a CNN.

It turns out that this idea has been thought and tested. In the paper by Kölsch et al [6], an approach (for document classification) was proposed where a CNN is pre-trained on a large dataset whereafter the weights of the convolutional layers are extracted and an ELM is fitted at the end. This network was then trained on different datasets with promising results. By using a GPU they were able to train the CNN+ELM structure on a completely different dataset consisting of 800 training points in little over one second achieving a test accuracy of 74% compared to re-training the entire convolutional network which took 10 minutes achieving 76% accuracy.

This paper served as the main inspiration for this report, since preparation for slide presentation should be quick and changing keywords for slides should be relatively easy. There was one issue with the report however, we are of the opinion that the comparison that the authors had made was unfair (to the advantage of the ELM). They had compared the CNN+ELM architecture to the whole CNN network trained again on the new dataset. This means that the number of parameters to adjust in the latter case would be many times more. We propose that a more fair comparison is that with an MLP fitted to the extracted convolutional network and train only the fully-connected layers.

Now the question is if this can be used in audio classification, CNN's work on images but sound is typically not represented on this form so there is reason for doubt. Fortunately studies have shown that CNN's can indeed be used successfully for audio recognition. Salamon and Bello managed to achieve a mean class accuracy of 79% for classifying environmental sounds using spectral methods and data-augmentation combined with CNN networks [7].

3. Approach

In this part we outline the methods we used to answer our question if an pre-trained CNN+ELM can be used effectively in audio classification.

3.1 Data

Our dataset is in the wav (Waveform Audio file) format retrieved from [8] which stores audio bitstreams, I.e. sampled versions of some soundwave. Feeding these arrays of sampled amplitudes directly into a network is not the best approach. All information that characterize words such as frequency content and amplitudes of different frequencies are combined into something hard to learn from. Therefore, in order to make it easier for a network to learn from the data, a fourier transform is used to extract frequency information about the sound files. The output of such a transform is the frequency content (amplitudes) of the signal.
The fourier transform of the signal is commonly represented in a so called spectrogram which is an image depicting the signal strength as a function of the frequencies.

It is on this data that we will train our networks on, because finding patterns in this data is easier compared to raw soundwaves.

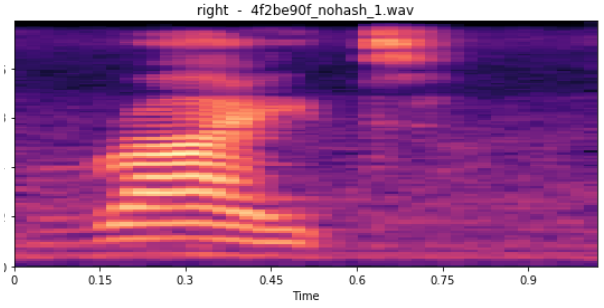


Figure 1: Example of spectrogram for the word “bed”

The spectrogram is essentially a 2D image or pixel map where high values (in Figure 1 the bright spots) represent a strong presence of frequencies in that area. Thus the problem of classifying audio files can be cast into the framework of image classification.

3.1.1 Noise reduction in data

Because individual data points (spectrograms) had considerably more noise than others, preprocessing the data to handle this was made. To remove some of the noise each data point was subtracted by one percent of it’s mean value, resulting in a cleaner soundwaves as can be seen in figure 2.

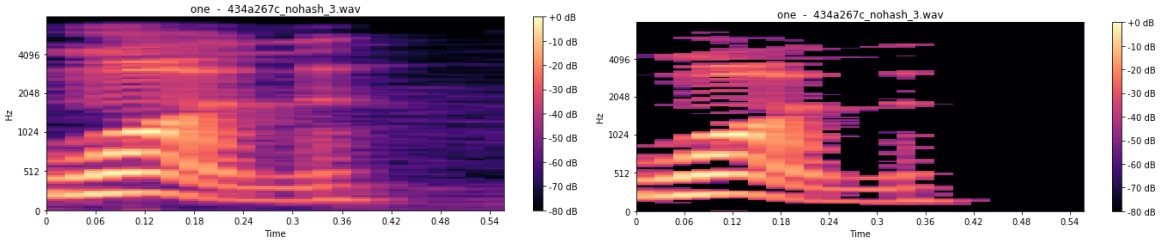


Figure 2: Audio file of *One* before threshold and after threshold editing

3.1.2 Add Silence to set

Some audio files were nearly 0.5 seconds long while others were closer to 1 second. A requirement for our network is dimension consistency of data. To make sure that every data

point would have the same dimension the longest data point was found and its difference in length was added as silence to all other points.

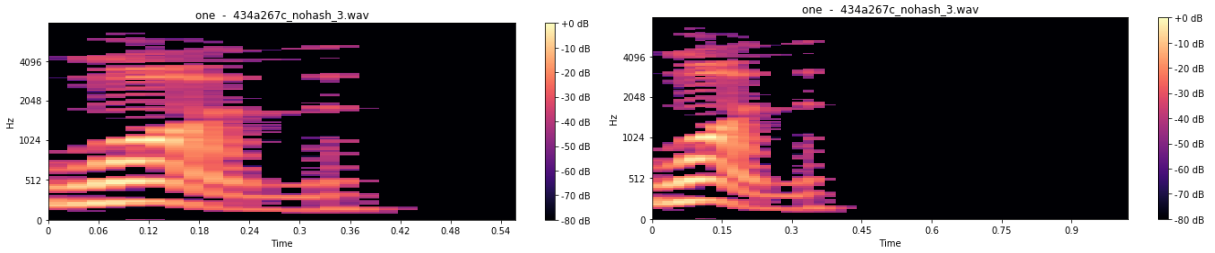


Figure 3: A 0.54 second audio of *One* before (left) and after (right) adding silence.

3.1.3 Normalize the data

Since the data has a considerable spread some kind of normalization is needed, each data point (i.e. each spectrogram) was then normalized by finding the max and min values (searching through every pixel) of the training set and then setting the new datapoint according to (resulting in every datapoint to be in the interval $[0, 1]$):

$$Datapoint_{New} = \frac{Datapoint_{Old} - min}{max - min}$$

Data Augmentation is a technique where one extends (augments) the training data by transforming the old data. For instance, it is possible to augment the dataset of images by shifting, rotating, scaling, or including random noise. This is a technique that should reduce the risk of overfitting and increase the performance of the model on real time data. However in some cases, data augmentation can deteriorate the performance [7]. For answering the research question posed, data augmentation was not necessary but it could most likely be used effectively in order to obtain better performing models in terms of accuracy .

3.2 Architecture

The approach will be similar to the one proposed in the paper by Kölch et.al [6]. Firstly we train a multilayer convolutional network with the following setup on the data. This architecture is however a bit simplified for computational reasons. The following table gives an overview of the different layers and how they are structured. The width, height and channels specify the shape of the output after each operation. Note that the width and height

only changes after max-pooling, this is because tensorflow pads the output from the convolutional layers with zeros to match the input.

# Layer	Type	Channels	Width	Height	Kernel size/stride
0	Input	1	128	44	**
1	Conv	32	128	44	5x5/1
2	Max Pool	32	64	22	3x3/2
3	Conv	64	64	22	4x4/1
4	Max Pool	64	32	11	2x2/2
5	Conv	64	32	11	3x3/1
6	Max Pool	64	16	6	2x2/2
7	Conv	128	16	6	3x3/1
8	Max Pool	128	8	3	2x2/1
9	Conv	256	8	3	3x3/1
10	Max Pool	256	4	2	2x2/2

Table 1: The architecture of the CNN model

The output from the last maxpool layer uses dropout of 0.5 and then flattened into a 1D array of size 2048 extracted features (256*4*2), used as an input for the fully connected layers. The fully connected layer is constructed of two hidden layer networks with 1024 nodes in each hidden layer using a dropout of 0.5.

After this initial training phase we can construct the ELM network by simply removing the MLP from the model and replacing it with a one-hidden layer network with 750 nodes with which we can train with the ELM algorithm described below.

3.3 ELM training

We train the ELM using a very straightforward and simple approach. First, one regards the output from the final convolutional layer as the input for the ELM, the input is denoted by x_j .

Then the weights from the input to the hidden layer W1 are randomly initialized. After applying the ReLu activation on these weighted input one obtains a response vector $\varphi(x_j)$ for each input data j. The ELM minimizes the following sum of squared losses:

$$\frac{C}{2}\sum_{j=1}^{N}\left\|e_j\right\|_2^2 + \frac{1}{2}||B||_F^2$$

Where e_j is the prediction error and B is the weight matrix connecting the hidden and the output layer. By putting all the response vectors for the training data together, denoted by H, and letting the target vectors be denoted as T, we can write this as the optimization problem:

$$min_{B \in R^{Nxq}} \frac{1}{2}||B||_F^2 + \frac{C}{2}||T - HB||_2^2$$

One can show that the optimal solution to this problem is given by the following equation:

$$B^* = (H^T H + \frac{I_N}{C})^+ HT$$

Where the $*^+$ denotes the Moore-Penrose inverse of the matrix *. We structured our ELM from a Github project (link to code found in notebooks provided) and modified it to match the requirements of this research.

3.4 Training Details

The training and validation of the networks was done using Python and Tensorflow. Furthermore, NumPy, Librosa and Pickle were used for array multiplication, managing audio and saving/loading data respectively. After preprocessing the data the weights in the convolutional net were pre-trained on 10 classes in the audio dataset called *AUDIOMNIST*, namely zero, one, two, three, four, five, six, seven, eight, nine. It added up to a total of 23.666 data points, with 20.116 (85 %) training samples and the rest counting 3.549 (15 %) data points were used for the validation samples.

After this, we extracted the convolutional network weights and added the ELM at the end. The ELM with the pre-trained model was then trained on twelve new classes from the same data set called *AUDIOCIFAR*, these classes had the following labels: bed, bird, cat, dog, tree, wow, house, go, right, left, stop and happy. Here, the number of datapoints was 23.365 split into 19.860 (85 %) for training samples and 3.504 (15 %) for validation samples.

Finally in order to make a fair comparison with a MLP, we fitted a three-layer MLP once again on top of the convolutional network and trained it on these new samples, note here that we do not fine-tune the entire network as was done in the ELM paper [6], because we argue that this is not a fair comparison between the two methods since that would require far more parameters to be adjusted. By doing this we are able to better assess the ELM's performance.

As for the learning parameters, we tried a coarse and fine search of parameter combinations with insignificant results. Instead we decided to use the Adam optimizer which is an algorithm for first-order gradient-based optimization of stochastic objective functions. It is based on adaptive estimates since it computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. The *Adam* method is designed to combine the advantages from two recently popular methods AdaGrad and RMSProp. Therefore its advantages are that the magnitudes of parameter updates are invariant to re-scaling the gradient, its stepsizes are approximately bounded by the step-size hyperparameter, it does not require a stationary objective, it works well with sparse gradients and it naturally performs a form of step size annealing [9].

In order to make the results more intuitive we propose the Accuracy Per Second (APS) as a measure of performance of networks. It is in this setting defined as the quotient of the accuracy achieved on validation set and the training time as follows:

Definition 1. Accuracy per Second (APS)

$$APS = \frac{Accuracy\ [\%]}{Training\ Time\ [s]} .$$

In general high accuracy combined with low training time is desirable we could in a naive way consider a model with higher APS to be better in that sense compared to a model with lower APS. However, APS alone is not enough to draw conclusions about what model is best (see discussion).

4. Experiments & Results

In this section we outline the results from the experiments that was made in figures and tables.

4.1 Pre-training the CNN network

The training of the CNN network on *AUDIOMNIST* took approximately 12,5 hours. Figure 4 shows how the accuracy increases with each epoch both for training and validation set which is indicates that the network manages to learn classifying the sound waves. The loss is also constantly decreasing, as seen plotted against epochs in Figure 5.

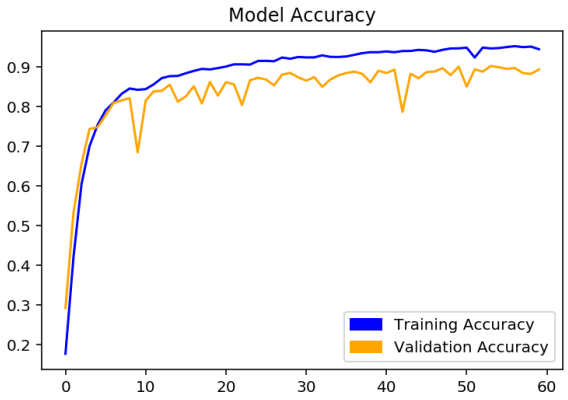


Figure 4: Accuracy plotted against epoch for CNN model.

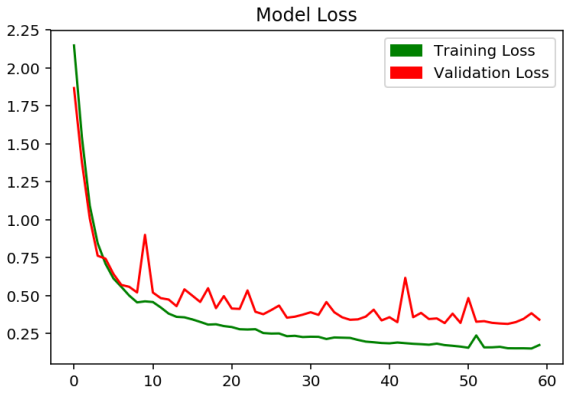


Figure 5: Loss plotted against epoch for CNN model.

4.2 CNN+ELM Training

When training the ELM on the whole dataset our computational memory is not enough, crashing the kernel's execution. To find a number of datapoints our resources could handle, the dataset was cut to smaller sets and size increased stepwise.

The results for this procedure can be seen in Figure 6. It shows how the training and test accuracy increases when we use a larger part of the dataset, indicated that using more data the better.

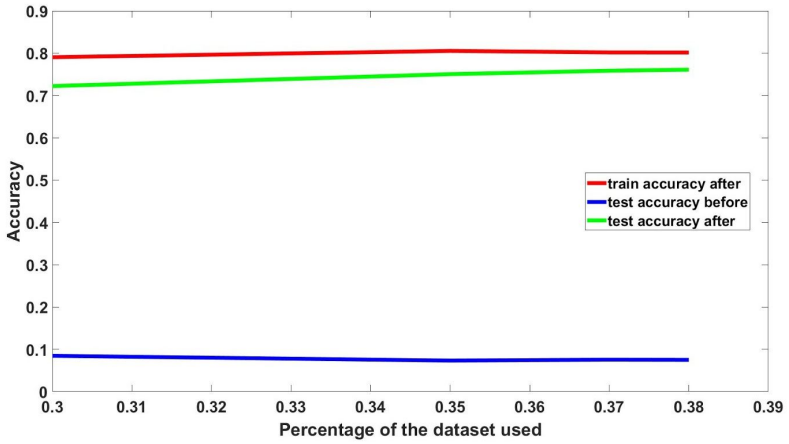


Figure 6: Accuracy plotted against amount of data used for ELM model.

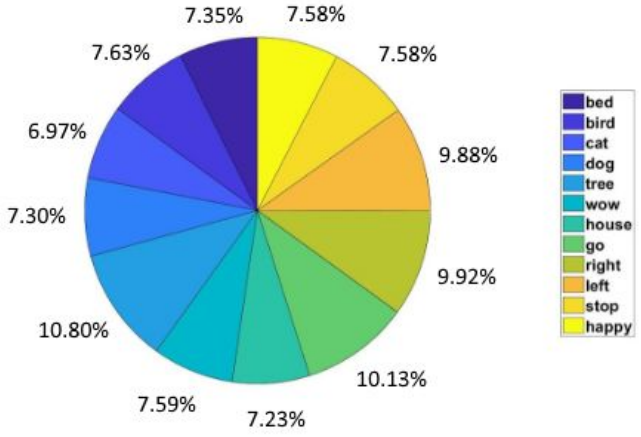
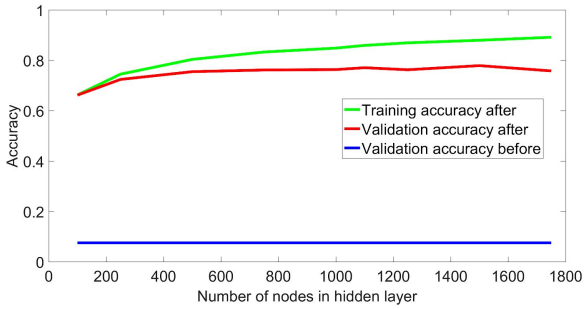


Table 2: Data distribution of classes when training the ELM model.

The pie chart in Table 2 shows how the data for the different classes are distributed after cutting all the dataset to 38%. In order to find a good value for optimal number of nodes in hidden layer, a coarse search of nodes was performed as seen in Figure 7 and 8.

The number of nodes that gives the highest accuracy on the test dataset is 1500 nodes and that is how many nodes we proceed with in our model. Notable is that adding more nodes than that does not give higher accuracy for the test data. However the model with 1750 hidden nodes is very close in performance on the test data and has higher accuracy on training set but takes longer time to train. The raw data used to create Figure 7 and 8 can be seen in Table 3.



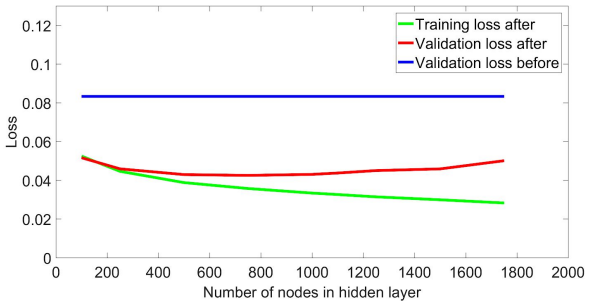


Figure 7 (Left): Accuracy plotted against number of nodes in hidden layer for the ELM model.
Figure 8 (Right) : Loss plotted against number of nodes in hidden layer for the ELM model.

Number of hidden nodes	**Training accuracy after**	**Validation accuracy after**	**Training loss after**	**Validation loss after**	**Training duration (s)**	**Validation APS**
100	66.26 %	66.19 %	5.25 %	5.16 %	0.1349	490,65
250	74.52 %	72.43 %	4.46 %	4.59 %	0.3958	182,99
500	80.39 %	75.51 %	3.88 %	4.30 %	0.7316	103,21
750	83.32 %	76.18 %	3.58 %	4.25 %	1.2443	61,22
1000	84.87 %	76.33 %	3.34 %	4.31 %	1.9049	40,07
1100	85.95 %	77.08 %	3.27 %	4.38 %	2.2097	34,88
1250	86.96 %	76.26 %	3.15 %	4.50 %	2.7195	28,04
1500	87.98 %	77.91 %	2.99 %	4.59 %	3.6843	21,14
1750	89.17 %	75.81 %	2.83 %	5.01 %	5.4669	13,86

Table 3: The data produced when trying different amount of nodes in hidden layer.

The APS as a function of the number of hidden nodes ranging from 1 to 1750 for the CNN+ELM network is depicted in Figure 11. It peaks at 800 for 20 nodes in the hidden layer and thereafter decreases as the number of hidden nodes increase. For the model with 1500 nodes having the highest validation accuracy as seen in Table 3 the APS score was slightly above 21.

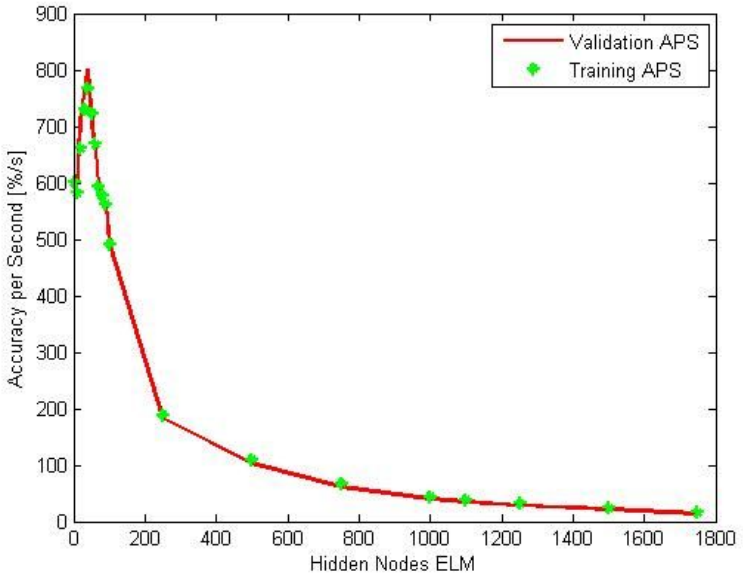


Figure 11: APS plotted against number of hidden nodes in the ELM.

4.3 CNN+MLP Training

Validation accuracy and validation loss before training was constant for all number of nodes in hidden layer and was 7.53 % respectively 8.33 %. Training the MLP for 5 epochs took approximately 48.35 seconds. We see that the validation and training accuracy have a similar behaviour which is good and confirms we are not overfitting and that it is increasing when we train for more epochs. The loss function (seen in Table 4) is decreasing which also is a good sign.

Epoch	Training Loss	Validation Loss	Training Accuracy	Validation Accuracy	Validation APS
1	1.1173	0.7285	67.25%	75.64%	7.49
2	0.6250	0.6623	80.51%	77.80%	3.96
3	0.4985	0.6439	83.74%	79.55%	2.72
4	0.4097	0.6109	86.72%	81.11%	2.09
5	0.3336	0.6205	89.25%	80.37%	1.66

Table 4: The data produced when training the MLP model.

We see that the APS scores are several factors lower for the CNN+MLP network when it achieves around the same accuracies as the 1500 node CNN+ELM (3.96 compared to 21). This should give an indication that the ELM is outperforming the MLP when looking at the tradeoff between accuracies and training times.

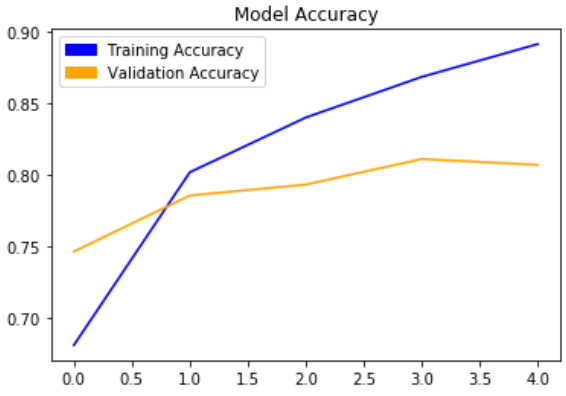


Figure 9: Accuracy plotted against epochs for the MLP model.

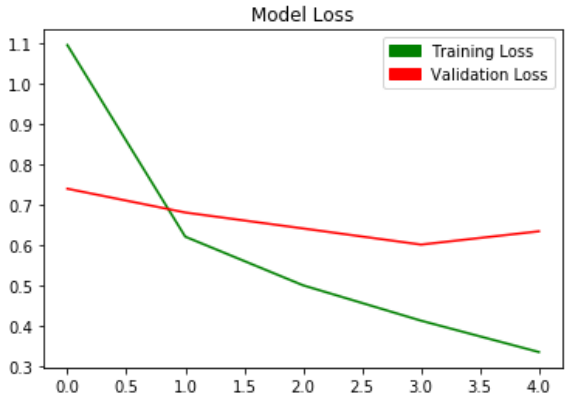


Figure 10: Loss plotted against epochs for the MLP model.

We see that the training accuracy increases with each epoch and also that the validation accuracy increases but perhaps not as much as. However, an accuracy of approximately 80% is not too bad. The loss function is decreasing strictly for the training dataset but has a small

increase on the validation loss from epoch 3 to 4 which might be due to overfitting since the training loss keeps going down.

5. Discussion

5.1 Pre-Processing data

The foundation of any machine learning practice is good data to base models on. In this project we encountered some issues with our data that might have had a negative impact on the results. Firstly, some of the audio files are recorded with good equipment and others are not. By looking at Figure 12, which is noisy and corrupt, even after the pre-processing steps and comparing it with Figure 13 and 14 one can notice the difference in noise levels. The dissimilarity between Figure 13 and 14 are also noticeable but not as substantial. Having augmented data is not necessarily negative. In fact up to a certain degree it is necessary in order to obtain an generalizable model. But there is of course a limit on how much augmentation is good.

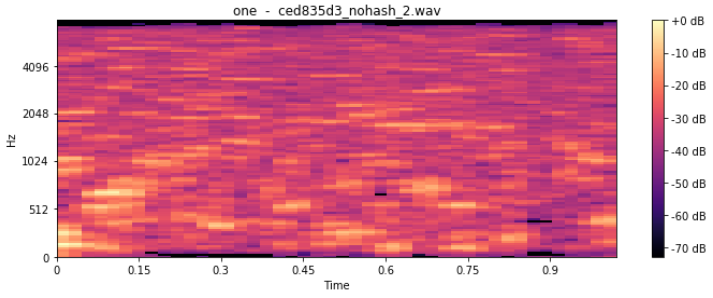


Figure 12: First “one” spectrogram.

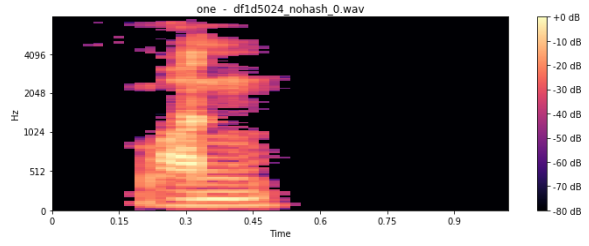


Figure 13: Spectrogram of a datapoint from the class “one”.

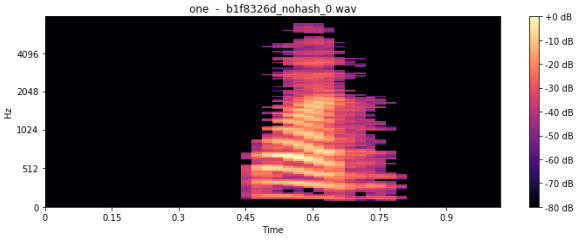


Figure 14: Spectrogram of a datapoint from the class “one”.

It would be interesting in further application to develop a way to completely remove "outliers", i.e. data points (like Figure 12 depicts) that are highly dissimilar to other data points of the same class and see how that affects model performance.

Another pre-processing factor which might affect performance is the way the arrays were formatted into the same shape. We chose a straightforward approach of simply extending "shorter" images with zeros. This was because it does not distort the sounds unlike cropping images would have. Pitfalls of doing this could be if the sizes of images are class-dependent the network could learn predictions by the feature of the zeros at the end. Also, adding a lot of unnecessary information to short audio files will affect the training time and needs a larger space to store all the data. We did not investigate if there are outliers in length also perhaps there were just a few recordings that are really long but now have a large impact on the rest of the data, we decided though for this report to go with a simple way and make everything the same size.

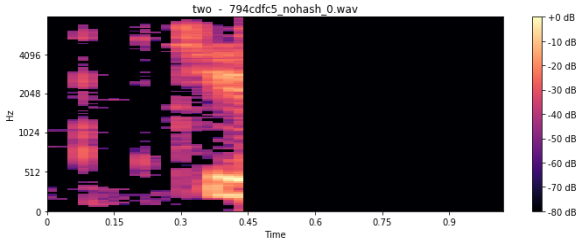


Figure 15: Spectrogram of a datapoint from the class "two".

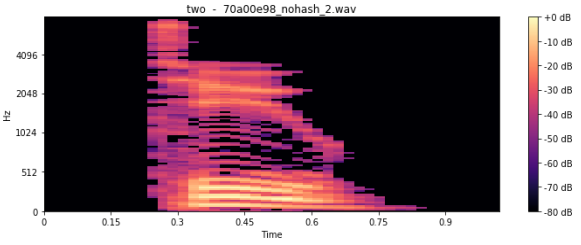


Figure 16: Spectrogram of a datapoint from the class "two".

5.2 On the APS measure

Comparing machine learning models in terms of performance is not a trivial task. Some performance measures are quite clear, such as the accuracy and training time where higher accuracy and lower training time in general is better. However, if one is comparing two models where the accuracy and training times differ then it becomes harder to compare them directly. One would have to look at the measures separately and decide on what is most important. This could be a tedious task if the number of different models is large.

We view the APS measure as an additional tool for this kind of comparison but not claiming it to be a replacement for looking at the accuracy/time separately. It has some drawbacks, in

some cases comparing models in this way would be the same as deciding who is the best runner between a marathon runner and a sprinter.

One would imagine that accuracy is the most important factor at least to some extent since one would in general not want a model that can only achieve 10 % accuracy even if it reaches that point in the blink of an eye. The problem with accuracy caps for different models becomes apparent and thus we recommend only using this as an comparison between models that can reach the same levels of accuracy. Further research could look into possible ways of improving the measure to identify use cases where it would be more suitable..

A final remark is that we could regard the negative-accuracy and the training time as a function of some variables (for instance number of hidden nodes in the network). Then it is not an unreasonable idea that these functions are both integer convex and strictly decreasing respectively increasing (integer convexity of accuracy could be seen in Figure 7 where the variable is the number of hidden nodes). In this case one could model it as an optimization problem and using marginal allocation for finding efficient solutions to the APS function.

6. Conclusion

Pre-training a CNN and combining the extracted network with an ELM works well on audio classification problems where the data is transformed into the frequency domain and treated as images. We show that our best pre trained CNN + ELM model performs better than an pre trained CNN + MLP on new datasets in terms of the APS measure. It is very fast to train on new dataset which shows that the ELM could be a potent tool in on-line audio classification tasks where fast training is crucial. Improvements on the training phase can be made by finding more advanced methods to augment the data, using all of the data provided, train the ELM sequentially and improving the feature extractor. However, further research is also needed on how augmenting the dataset affects loss and accuracy of the ELM.

Finally, the authors suggest a further research with the newfound APS metric and where it would be suitable. This measure could be a good decision base in some applications where one could potentially require only “good enough” answers in a shorter amount of time e.g. a network that learns about its ever changing environment to actuate in real time.

We wanted to show our gratitude for the free dataset provided by Google so we contributed 15 minutes of extra audio to the dataset. The reader can also contribute to it recording his voice in the browser [10].

7. References

[1] Das,Siddharth (2017,Nov 16) “CNNs Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more …”. Retrieved from : https://medium.com/@siddharthdas_32104/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

[2] Lin,Membevic,Konda(2016) “How Far Can We Go Without Convolution: Improving Fully Connected Networks” Retrieved from: https://openreview.net/pdf?id=1WvovwjA7UMnPB1oinBL

[3] Chahat Deep Singh “Image Classification: CIFAR-10 Neural Networks vs Support Vector Machines” Retrieved from: https://chahatdeep.github.io/docs/NNvsSVM.pdf

[4] ImageNetChallenge Results: http://image-net.org/challenges/LSVRC/2017/results, http://image-net.org/challenges/LSVRC/2017/results#team

[5] Guang-Bin Huang, Qin-Yu Zhu, Chee-Kheong Siew (25-29 Jul, 2004.) “*Extreme Learning Machine: A New Learning Scheme of Feedforward Neural Networks*” Retrieved from: https://pdfs.semanticscholar.org/2b9c/0e4d1d473aadbe1c2a76f75bc02bfa6416b0.pdf

[6] Kölsch,Afzal,Ebbecke, Liwicki (3 Nov 2017) “Real-Time Document Image Classification using Deep CNN and Extreme Learning Machines” Retrieved from: https://arxiv.org/pdf/1711.05862

[7] Justin Salomon, Juan Pablo Bello (28 Nov 2016) “Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification” Retrieved from: https://arxiv.org/abs/1608.04363

[8] Tensorflow, Google(2018, January 13) “Simple Audio Recognition”. Retrieved from: https://www.tensorflow.org/versions/master/tutorials/audio_recognition

[9] P. Kingma, Lei Ba (30 Jan 2017) “Adam: A Method for Stochastic Optimization” Retrieved from:
https://arxiv.org/pdf/1412.6980.pdf

[10] AIY Team,(No date), “Open Speech Recording“ Retrieved from:
https://aiyprojects.withgoogle.com/open_speech_recording