Key Word Spotting and Speech Enhancement using Convolutional Architectures

s1034065
Radboud University, Nijmegen, The Netherlands

ABSTRACT

Speech agents have become increasingly popular in recent times due to their ability to process speech and respond accurately. In this paper, we study and compare different architectures for two automatic speech recognition sub tasks namely: Key word spotting and Speech enhancement. We first discuss how we can transform speech signals into images using two commonly used feature extraction methods: MFCCs and Spectograms. Further, after conducting an extensive study, we discovered that the convolutional architectures outperform all the other methods used in our study for both tasks.

1 INTRODUCTION

In recent years, due to the growth of mobile devices, other sensory devices and the accessibility to high speed internet, the communication and interaction with machines using voice technology has rapidly increased. This has also led to large amount of audio data being made available. Major multinational companies have leveraged this data to build powerful and flexible speech agents such as Iphone Siri, Microsoft Cortana and Amazon Alexa. These systems do a very good job at recognising speech commands and processing them in real time. Surprisingly, these systems also perform equally well in noisy environments.

Neural networks have become extremely popular in solving complex machine learning tasks due to their high predictive power, availability of data and the easy accessibility to computational power. Hence, this field has gained a lot of research interest and researchers have been experimenting with different neural structures to solve complex problems in several domains. One of the most widely used network architecture is the convolutional architecture. This network was designed and is employed mainly for computer vision tasks. However, recent research has shown that these networks can also be generalised to work with other data formats such as audio.

In this paper, we employ two types of convolutional architectures to solve two crucial automatic speech recognition tasks:1) Key word spotting and 2) speech enhancement. We leverage three widely popular audio databases to solve these tasks. Lastly, perform a comparison study with other network architectures. This leads us to our research question which is three fold. 1) What is the best network topology to perform key word spotting? 2) What network topology is most suited to speech enhancement tasks? and 3) To investigate the performance of the best networks for both the tasks as a function of the topology of the respective networks.

Copyright held by the owner/author(s).
REMA ASR '19-'20, Radboud University, Nijmegen, Netherlands

2 RELATED WORK

Machine learning models have been quite popular and yield very good performance on classification tasks like key word spotting. Previously, Keyword/Filler Hidden Markov Model(HMM) has been commonly employed for key word spotting, for instance in Rohlicek et. al, [1]. In 2014, Chen et al., [2] proposed a Deep neural network to solve the keyword spotting task which performed significantly better than the HMM based approach. Further, Sainath et. al., [3] showed how convolutional nueral networks can be applied to the key word spotting task and how this method performs better than the deep neural networks in terms of false rejection rate.

Previously, speech enhancement tasks were performed by the means of Spectral subtraction[4] and Minimum mean-square error log-spectral amplitude estimator [5] which yielded good speech enhancement performance. More recently, denoising autoencoders have been employed to enhance speech using clean-noisy pairs as proposed by [6] and are the current state of the art in this task.

3 KEY WORD SPOTTING

Key word spotting is the identification of keywords in a stream of audio. This can be specifically useful for voice agents that perform some task for example playing music, to listen to keywords like "start", "stop","yes" and "ok". In this section, we discuss the pipeline for performing this task.

3.1 Dataset

To perform the key word spotting task we use the version 2 of Google Speech Commands dataset [7]. This dataset contains 105,829 audio recordings of people uttering a single word in WAV format. There are a total of 35 key words uttered that can be put into 3 categories: Numbers (zero to nine), Commands : "forward", "backward", "up", "down", "yes", "no","on", "off", "right", "left", "go" and "stop" and Auxiliary words: "bed", "bird", "cat", "dog","happy", "house", "marvin", "sheila", "tree", "follow", "learn", "visual" and "wow". The dataset also contains background noise files which are excluded for the scope of this study.

3.2 Feature Extraction

Before we begin building the key word spotting system, we first need to pre-process the audio files and extract relevant features from the audio for training. For this task, we employ Mel-Frequency Cepstral Coefficients (MFCC) to calculate spectral features. The procedure to obtain an MFCC is shown figure 1

Figure 1: The procedure to obtain MFCCs

As can be seen in figure 1,we first define a window and divide the signal into different time frames by sliding over this window. We then compute the Fast Fourier Transform(FFT) for each time frame to obtain features in the frequency domain. Further, we apply logarithmic Mel filter bank to the transformed frames. However, the filter bank coefficient calculated are highly correlated. Hence, we compute the Discrete Cosine Transformation(DCT) to decorrelate the filter bank coefficients which gives us the MFCC feature. Figure 2 depicts the conversion of a signal to a MFCC.

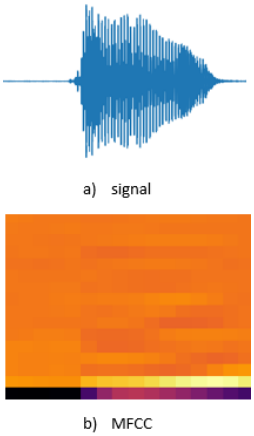

Figure 2: Visual depiction of an audio signal containing the word "go" converted to a MFCC

3.3 Method

In this section we discuss the architectures used for key word spotting. For this task, we start with employing a simple single layer perceptron, a dense fully connected network and a convolutional nueral network based architecture.

3.3.1 ***Single Layer Perceptron:*** In order, to achieve a baseline score, using neural networks we first implement a single layer perceptron network. As the name suggests, this neural network contains only one layer with a softmax activation function that predicts the class using the softmax probablity distribution.

3.3.2 ***Dense Neural Network:*** We further built a dense fully connected neural network. This network consists of three fully connected layers. The first two layers are activated using the Rectified linear units(RELU) activation and the final layer uses a softmax activation which does the classification.

3.3.3 ***Convolutional Neural Network:*** Lastly, we experiment further by building a simple convolutional neural network. We use a basic network because, the idea here is to compare the performance of different network topologies rather than achieving state of the art results. The architecture of the convolutional nueral network can be seen in figure 3

Layer	Activation	Output Shape	Parameters
Convolutional	RELU	15x15x32	160
Max pooling	None	7x7x32	0
Convolutional	RELU	6x6x64	8256
Max pooling	None	3x3x64	0
Flatten	None	576	0
Dense	RELU	64	36928
Dropout	None	64	0
Dense	Softmax	13	845
Total parameters: 46,189			

Figure 3: The architecture of the convolutional neural network

As seen in figure 3, the network consists of two convolutional layers, each activated using the RELU activation function and followed by a max pooling layer. These layers perform the task of learning the complex feature representation. In the end, we attach 2 dense layers that perform the classification. We also add in a Dropout layer in the end to prevent the model from overfitting the training set.

3.4 Experiments

As stated before,our research question is to compare network topologies as a function of performance for the key word spotting task. In this experiment, we only consider the command keywords: "forward", "backward", "up", "down", "yes", "no","on", "off", "right", "left", "go" and "stop" and bucket the other words into the "other" category. The goal of this task is to identify and distinguish the command keywords from a set other words.

The dataset was split with a 80%-10%-10% ratio into a training set , validation set and a test set. We first trained the three networks: single layer perceptron, dense neural network and the convolutional neural network. All the three networks were trained under similar conditions and with similar hyper parameters to have a fair comparison between them. We set the batch size to 256 samples to fit our GPU memory and used the Adam optimizer as our optimization algorithm with a learning rate of 0.01. We used the categorical cross entropy as our loss function since the task here is to classify the word into one of the key word categories or the "other word" category. We set up early stopping to monitor the validation loss with a patience of 10, and model checkpointing to save the best model based on the lowest validation loss. Finally, we tested the networks on the held out test set.

We used the accuracy metric and the validation loss as the measure of the performance of the model quality and for comparison. We trained each of the networks 10 times by setting different random seeds and leveraged the t-test to check for significance.

Further, we decided to conduct a more thorough study using convolutional neural network. We first built a simple network with one convolutional layer followed by one max pooling layer. We ran the same experiment 5 times but added a set of convolutional and pooling layers at every run. In the end, we analyse this addition of convolutional and pooling layers as a function of performance.

4 SPEECH ENHANCEMENT

In this section, we describe how we go about building a speech enhancement system that attenuates or cancels environmental noise.

4.1 Dataset

Speech enhancement research has been picking up pace in recent times. However, there are very few open source audio databases available to build these systems and for effective benchmarking. Speech enhancement generally requires clean and noisy audio pairs to train complex algorithms. To overcome this, we create our own dataset by merging 2 pre-existing open source datasets: Librispeech [8] and Environmental Sound Classification (ESC-50) [9].

The Librispeech dataset contains 100 hours clean speech that can be used for training. The authors of the dataset also provide separate validation and test sets for analysis and comparisons. Additionally, we use the ESC-50 dataset that contains environment noises of 50 types. For simplicity, we considered only 10 noise types for our analysis. These noise types are: vacuum cleaner,dog,helicopter,rooster,crying baby,clock alarm,coughing,washing machine,car horn,train.

4.2 Feature Extraction

The basic idea behind speech enhancement systems is that they are trained on noisy speech audio and can learn to reconstruct the audio without noise. To produce noisy speech, we perform data augmentation by blending clean speech with one of the 10 environmental noises. This blending was performed randomly with a noise level between 20% to 80%.

Now that we have created the noisy speech dataset we need to extract relevant features that we can use for training our algorithm. In the key word spotting task, we used MFCCs as features. However we use magnitude spectograms for this task to demonstrate how different features can be used to model audio data. Just as in the case with MFCCs , we first divide the signal into multiple windows and then compute the Fast fourier transform by sliding over each window to obtain the spectogram. The spectogram of the clean speech, noise and blended noisy speech can be seen in figure 4

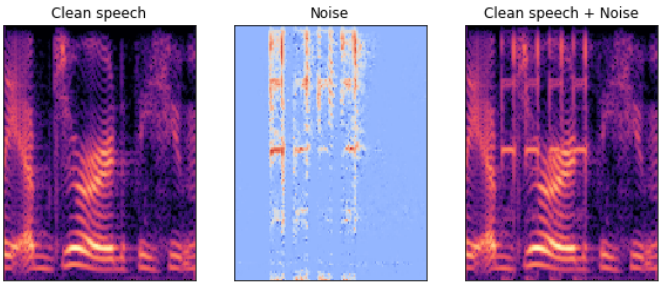

Figure 4: Spectogram of the clean speech , environmental noise and the blending of both

4.3 Method

Currently, the state of the art for speech enhancement has been obtained by using bottleneck architectures like the autoencoder. Autoencoders are types of neural networks that learn the representation of the data typically by ignoring noise in an unsupervised way.

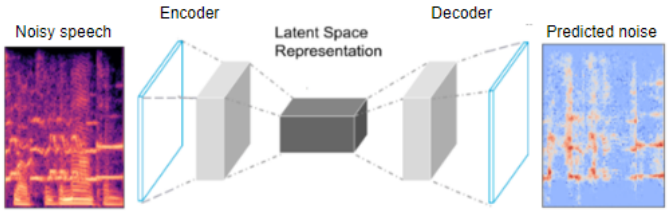

Figure 5: The denoising autoencoder process

Figure 5 depicts the process of employing an autoencoder for speech enhancement. The input noisy speech spectogram is first passed through an encoder which compresses the input into an encoded representation in the latent space. The layer that contains the compressed representation is called the bottleneck. The decoder then learns how to reconstruct the noise from the encoded representation. The predicted noise is then subtracted from the noisy speech to get enhanced speech without the noise.In this section, we discuss three types of denoising autoencoder architectures namely: simple autoencoder, dense autoencoder and convolutional autoencoder.

4.3.1 ***Simple autoencoder:*** We decided to start from the basics by building a simple autoencoder that can be used as the baseline for this task. Our simple autoencoder consists of an input layer that takes the flattened magnitude spectogram of size 128*128 as an input. This input layer is followed by two dense layers each that act as an encoder and a decoder. The encoder takes the input and reduces its dimension to a size of 64. The decoder takes the bottleneck representation and attempts to reconstruct the noise. The encoder is activated using the RELU activation function while the decoder output is activated using the sigmoid function.

4.3.2 ***Dense autoencoder:*** Further, we decided to extend the simple autoencoder by building a deeper fully connected encoder and decoder architecture. The dense autoencoder consists of three dense layers each for the encoder and decoder. The encoder takes the input spectogram of size 128*128 and reduces its dimension to a size of 64 as done in the case of the simple autoencoder as well. All the encoder layers and the top 2 layers of the decoder are activated using the RELU activation function while the last layer of the decoder is activated using the sigmoid function.

4.3.3 ***Convolution autoencoder:*** Lastly, we build a convolutional autoencoder. The encoder consists of four convolutional layers each activated by RELU and followed by a max pooling layer to reduce the dimension from input size 128x128x1 to 8x8x128. The decoder consist of the same set up but with up sampling layers instead of the max pooling operation to reconstruct the image to its original shape. The decoder convolutional layers are also activated using the RELU function apart from the last convolutional layer that uses the sigmoid function and reconstructs the image to its original shape 128x128x1. The detailed network architecture can be seen in figure 6

Architecture	Layer	Activation	Output Shape	Parameters
Encoder	Input	None	128x128x1	0
	Convolutional	RELU	128x128x16	160
	Max pooling	None	64x64x16	0
	Convolutional	RELU	64x64x32	4640
	Max pooling	None	32X32X32	0
	Convolutional	RELU	32X32X64	18496
	Max pooling	None	16X16X64	0
	Convolutional	RELU	16x16x128	73856
	Max pooling	None	8x8x128	0
Decoder	Convolutional	RELU	8x8x128	147584
	Up sampling	None	16x16x128	0
	Convolutional	RELU	16x16x64	73792
	Up sampling	None	32x32x64	0
	Convolutional	RELU	32x32x32	18464
	Up sampling	None	64x64x32	0
	Convolutional	RELU	64x64x16	4624
	Up sampling	None	128x128x16	0
	Convolutional	Sigmoid	128x128x1	145
Total parameters: 341,761				

Figure 6: The architecture of the convolutional autoencoder network

4.4 Experiments

As stated earlier, the research goal for the speech enhancement task is to find which network topology is best suited for this task. We divided the noisy speech input and the output noise spectogram pairs into train, validation and test sets with a ratio of 80%-10%-10%.

We then trained the three methods: simple autoencoder, dense autoencoder and the convolutional autoencoder using the same conditions so as to have a fair comparison as also done in the key word recognition task. We set the batch size to 256 samples and used the Adam optimiser as the optimization algorithm with a learning rate of 0.0001. We used the Huber loss as our loss function which calculated the loss between the actual noise and the noise predicted by the autoencoder. We set up early stopping to monitor the validation loss with a patience of 10, and model checkpointing to save the best model for each autoencoder architecture.

As done in the key word recognition task, we trained each model 10 times with different random seeds and used the t-test to determine if the difference in performance was statistically significant. Also, we further conducted a more detailed study on the convolutional autoencoder. We trained the architecture 5 times starting with the encoder and decoder having one convolutional and pooling or upsampling layers each. We further added a convolutional and pooling or upsampling layer for every run. We then analyse this addition of layers as a function of performance.

5 RESULTS

In this section we discuss the results from running the experiments for both tasks.

5.1 Key word spotting

Table 1 lists the accuracy and loss of all three networks. As can be seen in the table, the Convolutional neural network outperformed both the single layer perceptron as well as the dense neural network in terms of both having an higher accuracy and a lower loss. The performance increase of the convolutional neural network was found to be statistically significant compared to both the other networks. Though, the dense neural network outperformed the single layer perceptron which gives us an indication that the single layer perceptron network was not complex enough to handle the task.

Table 1: Performance comparison

Architecture	Validation Accuracy	Test Accuracy	Test Loss
Single Layer Perceptron	63.61%	63.45%	1.2171
Dense Neural Network	84.88%	84.81%	0.5229
Convolutional Neural Network	87.59%	88.10%	0.4002

Additionally, figure 7 depicts the trend of the validation accuracy for all the three networks. It is interesting to note that dense neural network accuracy peaks faster than the convolutional network and starts flattening at epoch 20, while the convolutional network performance keeps increasing steadily. Note: the lengths of the curves are different because all methods were trained using early stopping to prevent overfitting.

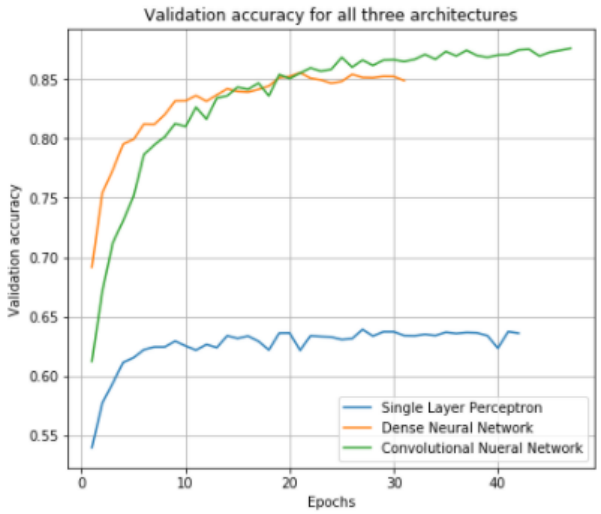

Figure 7: Accuracy trend over the validation set

Further, figure 8 shows the confusion matrix for the convolutional neural network. It can be clearly seen that the diagonal of the matrix has higher values which means that the network indeed performs very well. The only point to note is that the model sometimes classifies the words "go" as "no" and "off" as "up" and vice versa. This might be because of the different dialects and pronunciations of the speakers.

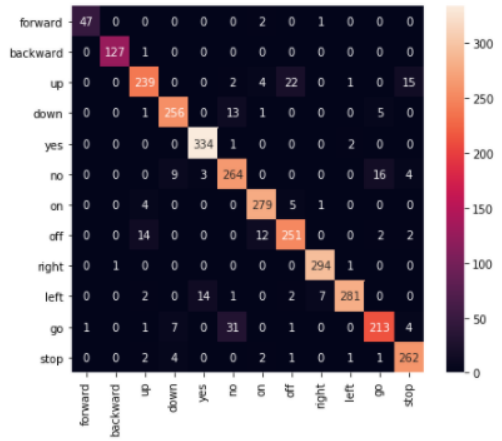

Figure 8: Test confusion matrix for the convolutional neural network

Lastly, we run an additional experiment to analyse the performance of the convolutional neural network by training the network for 5 iterations and adding a combination of convolutional and pooling layers at every iteration.

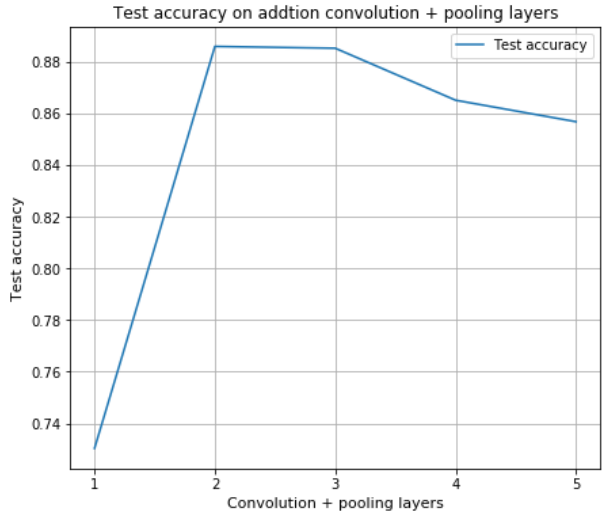

Figure 9: Test accuracy on addition of layers for every iteration

Figure 9 shows the test accuracy of addition of layers for all the 5 iterations. From the figure, it is clear that the model with a single convolutional and pooling layer performs far worse than the dense neural network. However, adding 2 such layers gave the best results and further adding more layers does not improve the performance or performs worse which means it might be overfitting the task.

5.2 Speech enhancement

The validation and test loss for the speech enhancement task can be seen in table 2. The convolutional autoencoder outperformed both the simple autoencoder and the dense autoencoder and the improvement was found to be significant using the t-test. However, we were surprised to see the simple autoencoder perform slightly better than the dense autoencoder , though the difference is not significant. This means that adding more fully connected layers to the encoder and decoder is equivalent or worse than having a one layer encoder and decoder network.

Table 2: Performance comparison

Architecture	Validation Loss	Test Loss
Simple Autoencoder	0.011	0.0115
Dense Autoencoder	0.0112	0.0115
Convolutional Autoencoder	0.0093	0.0094

Figure 10 shows the trend of the loss decreasing over the validation set. It can be seen that, though the simple autoencoder and the dense autoecoder have comparable performance, the dense autoencoder learns the representation faster than the simple autoencoder. Additionally, we can see the loss of the convolutional autoencoder decreasing steadily and gradually to perform much better than the other networks.

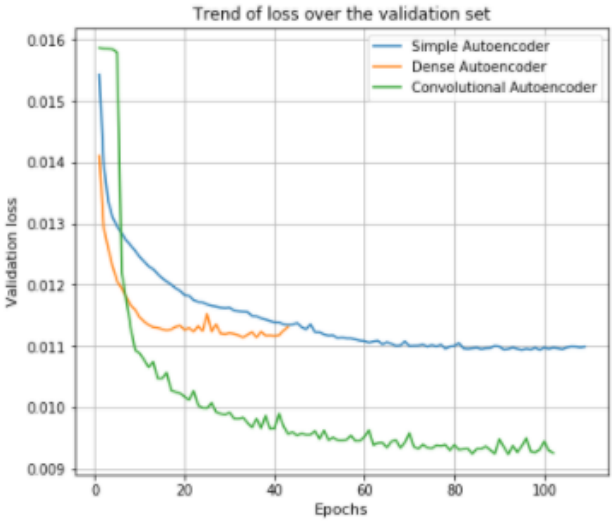

Figure 10: Loss trend over the validation set

Additionally, figure 11 depicts the output of the speech enhancement task. As stated before, the autoencoder takes the noisy speech as an input and learns to reconstruct the noise , which is then subtracted from the noisy speech to get the enhanced speech. In the figure we can see that the convolutional autoencoder does a decent job at enhancing the speech by reconstructing and decreasing the noise.

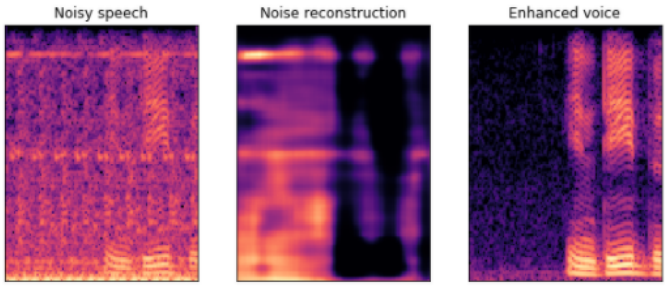

Figure 11: Speech enhancement by the convolutional autoencoder

Lastly, as in the case with key word spotting we decided to train the convolutional autoencoder for 5 iterations and add convolutional and pooling layers to the encoder and convolutional and upsampling layers to the decoder per iteration. The results can be seen in figure 12. In the figure we can see that with one convolutional and pooling or upsampling layer the model still performs better than the dense or the simple autoencoder. Additionally, the loss is the least when we add 4 convolutional and pooling or upsampling layers and adding more layers from there on yields comparative or worse results.

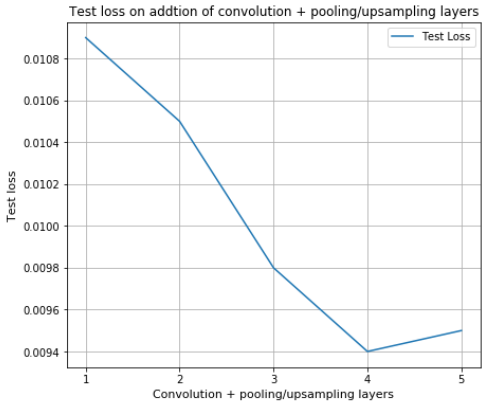

Figure 12: Test loss on addition of layers for every iteration

6 DISCUSSION AND CONCLUSION

In this paper, we discussed two sub tasks of automatic speech recognition namely: key word spotting and speech enhancement. After performing an extensive experiment, we report that 1) The best architecture to perform key word spotting is the convolutional neural network with two convolutional and pooling layer combinations. 2) The architecture most suited for speech enhancement is the convolutional autoencoder with four convolutional and pooling or upsampling layer combinations each for the encoder and decoder. A possible reason for convolutional networks performing better on both tasks is that the dense network ignores the input topology and reshapes the image into column vectors. However, audio signals show very strong correlations in time and frequency and this can be best modelled by convolutional architectures.

We also demonstrated how different input features namely: MFCCs and Spectograms, can be used to build efficient speech solutions. However, in future we would like to conduct a thorough comparison between which input feature set out of these two performs better for both tasks. Additionally, we found that the dense network outperforms the simple layer perceptron and converges faster than the convolutional neural network, though performing 4% worse for the key word spotting task. To the contrary, in the speech enhancement task, the dense autoencoder performs worse than the simple autoencoder. This implies that dense autoencoders might not be a good choice to employ in speech enhancement tasks

Additionally, it is important to remember that the goal of this experiment was not to obtain state of the art results, but to find which architectures suit these problems the most. Also, our comparison is limited to using architectures that take an image as an input rather than the signal itself. Since we are now aware that convolutional networks perform very well on these tasks, in future we would like to expand our research and employ state of the art pre trained networks for this task. We would also like to employ the Variational autoencoder, which learns the representation as a distribution and employs the reparameterization trick to update the weights in the backward pass. Also, we would like to expand our research to study the bottleneck representation more thoroughly.

Lastly, in this paper we discuss both the tasks as two separate tasks. However, in future we would like to study and propose a unified convolutional architecture that first performs speech enhancement in an unsupervised way and then performs key word spotting on the enhanced audio set.

REFERENCES

[1] J Robin Rohlicek, William Russell, Salim Roukos, and Herbert Gish. Continuous hidden markov modeling for speaker-independent word spotting. In *International Conference on Acoustics, Speech, and Signal Processing,*, pages 627–630. IEEE, 1989.

[2] G. Chen, C. Parada, and G. Heigold. Small-footprint keyword spotting using deep neural networks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4087–4091, 2014.

[3] Tara N Sainath and Carolina Parada. Convolutional neural networks for small-footprint keyword spotting. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[4] Rainer Martin. Spectral subtraction based on minimum statistics. *power*, 6(8), 1994.

[5] Yariv Ephraim and David Malah. Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator. *IEEE Transactions on acoustics, speech, and signal processing*, 32(6):1109–1121, 1984.

[6] Xugang Lu, Yu Tsao, Shigeki Matsuda, and Chiori Hori. Speech enhancement based on deep denoising autoencoder. In *Interspeech*, pages 436–440, 2013.

[7] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

[8] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[9] Karol J Piczak. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1015–1018, 2015.