Applying Image Recognition to Automatic Speech Recognition

Determining suitability of spectrograms for training a deep neural network for speech recognition.

Nikè Lambooy

Student number: 4090349

Supervisor: Dr. Frans Adriaans

Second judge: Dr Ben Rin

30 June 2017

Bachelor Kunstmatige Intelligentie, UU

ECTS: 7.5

TABLE OF CONTENTS

1 INTRODUCTION

Automatic Speech Recognition (ASR) is one of the many fields of Artificial Intelligence. From programs such as Siri and personal assistants such as Alexa to automatic answering machines, devices that try to convert speech utterances in natural language into the corresponding text are making their way into many parts of our daily lives. To get from soundbites to letters on screen, however, many steps have to be taken.

1.1 ASR ARCHITECTURE

In the mid twentieth century Claude Shannon devised the noisy channel model, where based on a sound and a word, the probability of the word corresponding to the sound $P(word|sound)$ is calculated. The word with the highest likelihood of corresponding to the sound is chosen as the 'correct' word. Most speech recognition systems also make the *Markov Assumption*: that words uttered in sequence depend on each other, making the problem solvable with a *Hidden Markov Model* (see the section on HMMs) (Russel and Norvig 2010). Before this model comes into play, however, we need some way to represent the sound, a digital representation an algorithm can perform computations on.

1.1.1 Representation and Division

One of the regularly used ways of representing the speech sound is called the *Mel Frequency Cepstral Coefficient* (MFCC). Essentially, this is a set of features, such as formants and other spectral peaks and valleys, calculated from the waveform of the utterance. This waveform is transformed into a vector of spectral information gleaned from 15 to 40 ms of the audio signal. This spectral information reflects how prevalent each sound frequency is at that point in time. The vector is then subjected to a series of calculations to form an MFCC. This is done for the entire signal (Jurafsky and Martin 2008).

The choice to 'slice' the data into pieces of 15-40 ms instead of individual phonemes stems from the difficulty of separating phonemes from each other. As no automated systems have approached the accuracy of a human processor on this issue, the valid solution is to slice the data into small enough pieces and recognise the same phoneme multiple times. This way we can be sure every phoneme is recognised in sequence without the risk of learning form data that is faultily separated.

1.1.2 Recognition

1.1.2.1 Hidden Markov Models

Many different ways of recognition have been tried, but the most prevalent make use of Hidden Markov Models (HMMs) such as the one in figure a. A HMM makes use of a set of states, organised in a graph. The likelihoods of traversing from the one state to another can be calculated in various ways. One of these approaches is the *Gaussian Mixture Model* (GMM), where these probabilities are computed by adding together a variety of Gaussian curves, mapping data vectors to observation likelihoods. Another way to calculate these probabilities is with an *Artificial Neural Network* (ANN) (see the section on Neural Network Approaches). The result of a HMM is a chain of most likely phonemes. These phonemes are then converted into actual text using a lexicon (Jurafsky and Martin 2008).

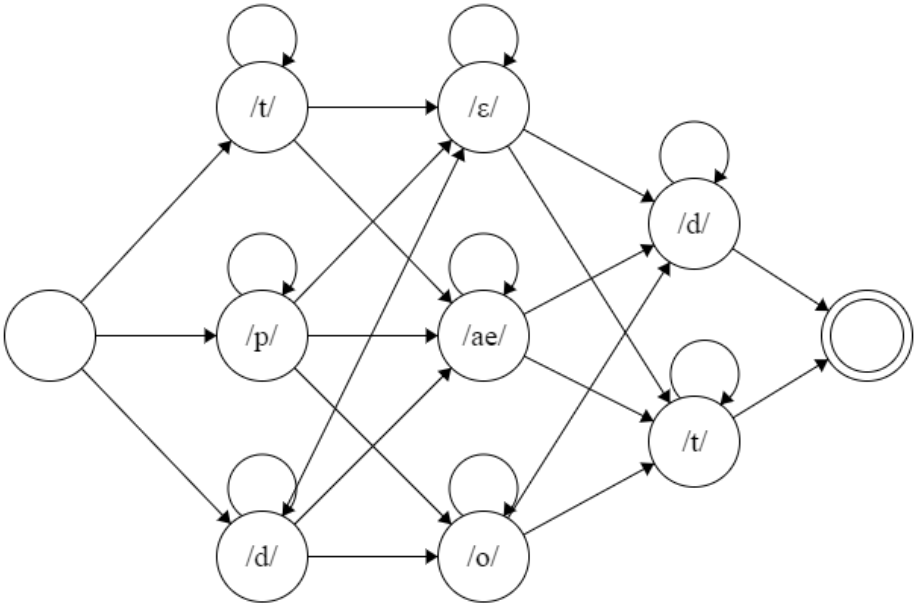

Figure a. A representation of a HMM. The empty states on the left and right are the entry- and exit nodes respectively. The arrows show to which states the model can traverse, and each arrow carries the likelihood of traversing, or rather, the likelihood of the one phoneme following another. It is possible to traverse to the same state as the model is currently in, because of the slicing of the data as mentioned.

1.1.3 Difficulties

ASR research currently concerns itself mostly with two subjects: finding better, faster and more efficient methods, and investigating methods that deal better with difficulties such as speaker variability and noise.

Speaker variability pertains to both differences in the vocal tract, such as oral- and nasal cavity size or larynx size, and differences in the speakers themselves, such as accent. Both 'to – mah – to' and 'to – may – to' must correspond to the same word 'tomato', and ASR methods should hold up just as well to Received Pronunciation as to American English.

Noise is the disruption of the speech signal, such as background speech, helicopters, cars or factory noise. The problem this creates is that data is lost. For example, if the noise occupies the frequency bands from 250 Hz to 400 Hz, it is impossible to be certain whether the speech signal does or doesn't contain those frequencies: they are masked by the noise. Another problem with noise is that a neural network might attempt to learn from noise and associate certain patterns in the noise with certain phonemes. The resulting network will detect these patterns, which results in much less accurate classification, as the patterns in the noise of course have nothing to do with the actual phonemes.

Attempts to solve the problem of noise have been made using two approaches: *marginalisation* and *state-based data imputation*. Marginalisation computes output probabilities based on the reliable evidence only. State-based data imputation tries to fill the gaps in the data the noise makes by estimating values for the unreliable regions. This is done by calculating the probabilities of possible correct values, and then substituting the unreliable values for the most likely of the possibly correct ones. Both methods were tested on differing

levels of noise and show improvement in classification from a standard MFCC baseline, with marginalisation giving the best results (Cooke, et al. 2001).

1.2 Neural Network Approaches

Artificial Neural Networks (ANNs) are inspired by and (loosely) based on brains. In a set of brains an enormous number of neurons are interconnected. When one such a neuron receives a signal, it will send it on towards the next neuron if a neuron dependent threshold is surpassed. This structure is emulated in ANNs. An ANN consists of at least two layers of 'neurons', called *nodes*: an input layer and an output layer. Generally, it has one or more 'hidden' layers in between the input- and output layers. The simplest ANN is the perceptron (figure b), where a set of input values (the first layer) is weighted (each input is multiplied by its own weight, a value between 0 and 1) and sent to a single output node. This node gives a binary output based on the sum of its inputs. The weights are then updated based on the correct output, determining the influence of each of the inputs to ensure correct future classifications.

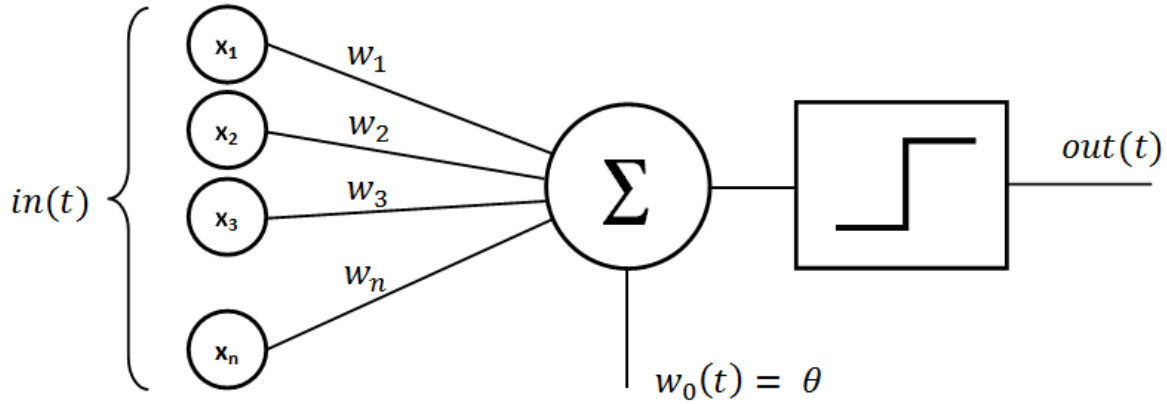

Figure b. A schematic representation of a perceptron

In more complicated ANNs there are multiple nodes in the hidden layers with possibly multiple output nodes, depending on the nature of the data that is being classified. For example, if the ANN is designed for classifying the numbers 0-9 from the pixel values of an image, it will have ten output nodes: one for each possible classification. The classification the network returns is the value corresponding to the node with the highest output.

Neural Networks have been used in phoneme recognition since the late 1980's. Waibel et al. use *Time Delay Neural Networks* (TDNNs), which "have the ability to relate and compare current input to the past history of events" by introducing delays in the input of the data. The data is 'cut up', and each delay has a separate set of weights. This way, patterns that occur over time can be recognised. These TDNNs were applied to the task of recognising the phonemes /g/, /b/ and /d/. The resulting network had a reduced error rate from 6.3 to 1.5 percent, achieving an average recognition score over testing data from three speakers of 98.5%. A variation of HMMs they applied to the same task had a recognition score of 'only' 93.7%, making the usage of TDNNs a very successful way for phoneme recognition tasks (Waibel, et al. 1989).

1.2.1 Deep Learning

A deep neural network distinguishes itself from a regular ANN by its multiple hidden layers. As in a regular ANN, consecutive layers are all interconnected. This results in a network that can recognise patterns within patterns – a meta level of patterns. Each layer adds a level of abstraction, allowing for complex data processing. Recent advances made with DNNs on the subject of ASR have prompted a renewed interest in using them in acoustic modelling. *Pre-training* especially, which adjusts the weights of the network to be more attuned to the data in advance, has been very successful in reducing *overfitting* (the network being too good at classifying the training data, and therefore less accurate when classifying new data) of neural networks (Hinton, et al. November 2012).

Recurrent Neural Networks (RNNs) are also used in speech recognition. These networks are not only fully interconnected in subsequent layers, but are also connected with previous layers, creating cycles of nodes that give the network a 'memory'. This makes RNNs very suitable for usage in ASR. The best results booked with such a network is a phoneme error rate of 17.7% on the TIMIT corpus: a dataset of continuous natural speech (Graves, Mohamed and Hinton 2013).

Context-Dependent Deep Neural Networks (CD-DNNs) are another category of DNN applied to the problem of speech recognition. In CD-DNNs the output layer of the network uses the likelihoods of the HMM that are estimated using a log-linear model. The resulting network in combination with a HMM led to 17% relative word error rate on an internal Xbox voice search dataset, suggesting that this model can be very effective when applied to speech recognition tasks (Yao, et al. 2012).

1.2.2 Convolutional Neural Networks

A *Convolutional Neural Network* (CNN) has for each hidden layer in the network two layers: a *convolutional layer* and a *pooling layer*. In the convolutional layer, a matrix operation is applied to the value sent by each node of the input layer. These matrices are also called 'filters', with each filter recognising their own pattern. The pooling layer then 'pools' several values together. The value sent on is generally the average or maximum of the pooled values. This significantly reduces the size of the network, speeding up the process while maintaining accuracy of results. CNNs are most often used in image recognition to recognise patterns that occur across an entire image. These patterns occur in colour configurations, light/dark nuances, saturation differences or a combination of the above (Lawrence, et al. 1997) (Zisserman and Simonyan 2014).

This technique has also successfully been applied to ASR. Abdel Hamid et al. use CNNs in speech recognition to accommodate for some types of speech variability. They present a modification of regular CNNs: *Limited Weight Sharing* (LWS). Regular CNNs use *Full Weight Sharing* (FWS), where the same filters are shared between parts of the data to recognise similar patterns in different timeframes or frequency ranges. In the LWS scheme only the frequency bands share weights. The experiments conducted evaluate the effectiveness of the different weight sharing schemes compared to each other and compared to regular DNNs. The resulting data shows that better performance is achieved by increasing the pooling sizes, which regulate the amount of downsizing the pooling layers do, up to 6. A larger number of filters also increases performance. LWS however needs fewer filters than FWS for a higher performance. The CNN with LWS gave over 8% improvement compared to

DNNs. With pretraining this percentage can be even higher, though the effect of pretraining is higher for DNNs than for CNNs (Abdel-Hamid, et al. October 2014).

1.3 IN THIS PAPER

The main objective of this paper is to investigate the possibility of using *spectrograms* as a representation of speech data for classification. A spectrogram is a visual representation of a speech signal over time and frequency. The level of blackness denoted at a specific time and frequency indicates how prevalent said frequency was at that point in time (see figure c). The advantage Spectrograms have, compared to, for example, MFCCs is that it represents the entire phoneme, giving a more holistic view as they change (a little) over time. This creates patterns a neural network can learn from. Another advantage is that we can use image recognition techniques to recognise patterns in these spectrograms, combining two fields of research. As ANNs are the main technique for recognising patterns in images, they are what we will use to recognise patterns in the spectrograms.

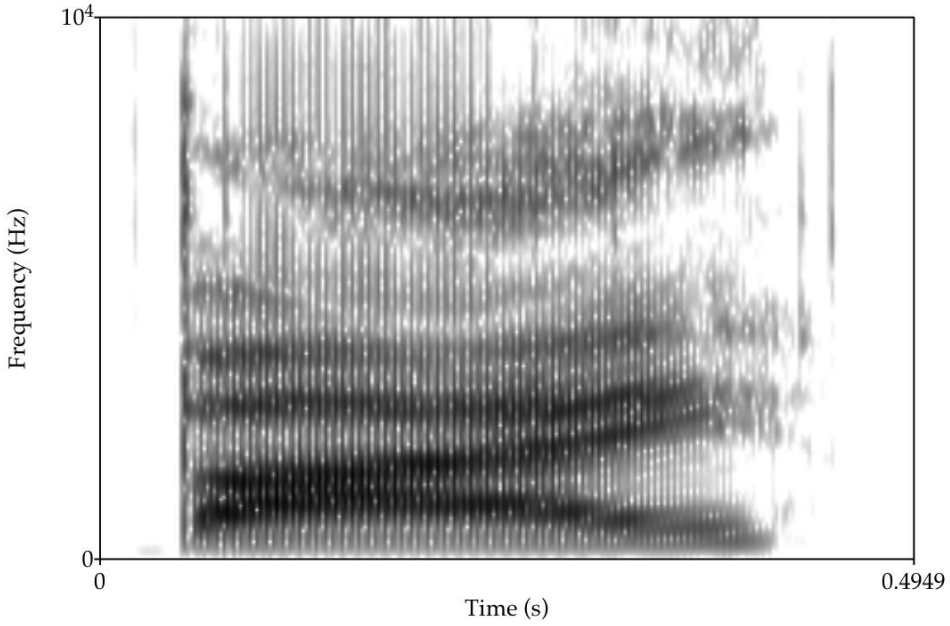

Figure c. Spectrogram of the utterance of the word 'buy'

First, we will assert what kind of pre-processing the data needs for a neural network to be able to learn from it. The parameters resolution and contrast will be tested. Secondly, we will test how deep a DNN is most suitable for the task, varying the number of hidden layers. Lastly, we will test how suitable for learning spectrograms are compared to a well-known image dataset.

The aim of this paper is to investigate a way of improving current classification techniques. We will focus on accuracy rather than time or memory as a measure of improvement, as these are independent of the hardware being used.

2 METHODS

2.1 DATA

2.1.1 Vowels

The data the neural network will be learning from is the Hillenbrand vowel data. This dataset consists of recordings of 46 men, 48 women and 46 children speaking twelve vowels between the consonants /h/ and /d/. The vowels in question are (with the actually pronounced word in brackets): ae (had), ah (hod), aw (hawed), eh (head), er (heard), ey (haid), ih (hid), iy (heed), oa (hoad), oo (hood), uh (hud), uw (who'd) (Hillenbrand 1995). The data from the children was not used in our research, as Hillenbrand shows a lower classification accuracy of human processors when classifying this set.

This dataset was chosen because vowels have very distinctive properties. The reverberations in the cavities of the vocal tract form distinctive frequencies, called *formants*. By the formants F1 and F2 all vowels can be distinguished from each other. This makes vowels very suitable for a neural network to learn from. We will test the network on the whole dataset, a set of the 'iy', 'ah' and 'uw' sounds because their formants lie furthest apart, and a set of the 'iy', 'ih' and 'eh' sounds, because their F1 and F2 lie very close together. It is visible how different the former three vowels are and how similar the latter, in figure d and e.

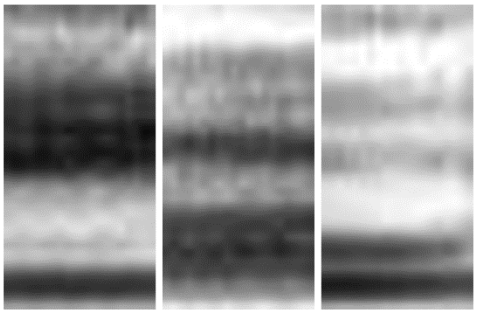

Figure d. Processed spectrograms of the vowels 'iy', 'ah' and 'uw' from left to right

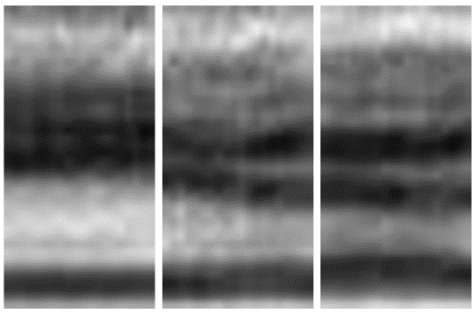

Figure e. Processed spectrograms of the vowels 'iy', 'ih' and 'eh' from left to right

2.1.2 Written digits

The second dataset, with which we will compare the results of the spectrograms to determine how hard or how easy it is for a neural network to learn from said spectrograms, is the MNIST dataset of written digits (see figure f). This is a well-known dataset used for image recognition that many different kinds of neural network have been trained and tested on (LeCun, et al. 1998).

To ensure an equal comparison, the network will be trained on a similar amount of data from both the Spectrograms and the MNIST data. The total Spectrogram dataset consists of 1116 spectrograms. In case of training and testing on all the digits 1100 examples have been used. When classifying only three different examples, the spectrograms numbered 279. A set of 300 digits has been used in comparison when classifying between 3 different digits. The images of the digits have a similar size as the spectrograms, i.e. 28 by 28 pixels, resulting in an input of 784 pixels compared to the 800 pixels of the spectrograms.

The digits ‘0’, ‘1’ and ‘4’ have been chosen as digits that lie far apart in looks, and because each has a feature distinguishing them from the other two. The ‘0’ has a hole in it, the ‘4’ is often written in two pen-strokes and the ‘1’ has none of the above, often being no more than a straight line. For the classification of very similar digits, the ‘3’, ‘8’ and ‘9’ were selected, as they can easily be mistaken for each other.

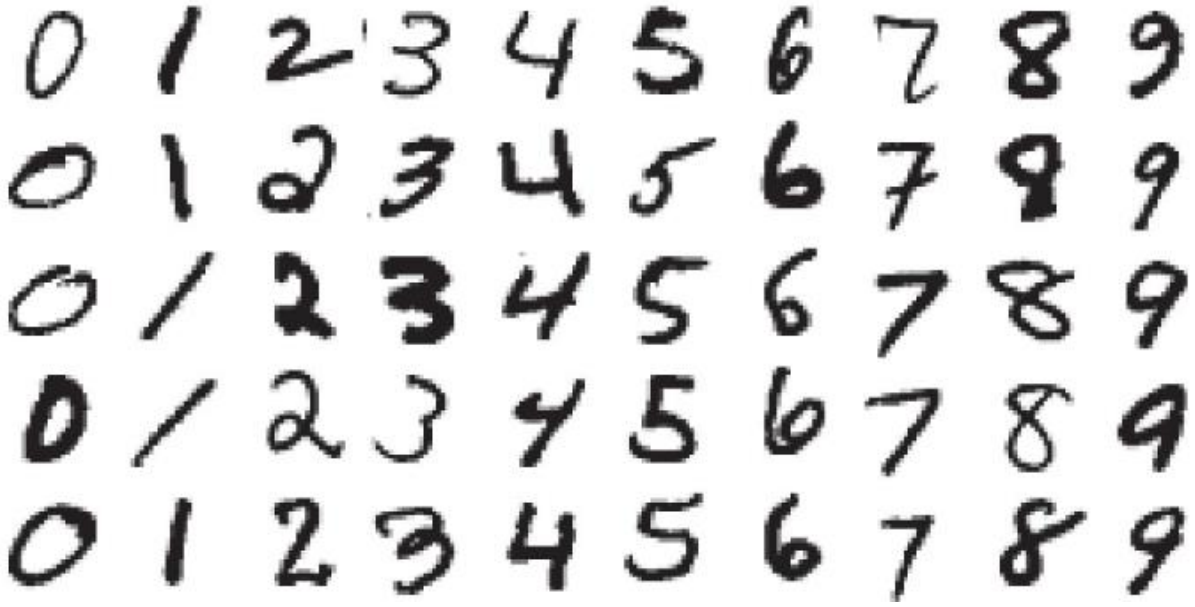

Figure f. Examples of the MNIST dataset

85% of both datasets was chosen randomly each run to be the training data. The leftover 15% was used as test data. Because of the size of the datasets, the accuracy of the network varied per run. Therefore, each test consisted of 20 runs, yielding an average, minimum and maximum percentage of classification.

2.1.3 Processing the data

Before the network was capable of learning from the data, the data had to be processed. First, with the program Praat[1], the audio files (see figure g) of the Hillenbrand vowel data were converted into spectrograms of 1800 by 1200 pixels (see figure h). The resulting pictures have subsequently been cut: 660 pixels from the sides each and 120 pixels from both the top and bottom. This is because Praat gives the spectrograms a white border meant for garnishing. More pixels were cut from the sides to cut away as much of the /h/ on the left and /d/ on the right; after all we want to learn purely from the vowel. Thereafter the pictures were scaled down from 480x960 pixels to 20x40 pixels. With these images six datasets were created, each with a different level of contrast[2] ranging from 0 to 125 in increments of 25 (see figure i).

[1] http://www.fon.hum.uva.nl/praat/

[2] The contrast is represented on a scale from -127 to 127, where -127 is entirely grey, 0 is the original picture and 127 is the picture fully black and white.

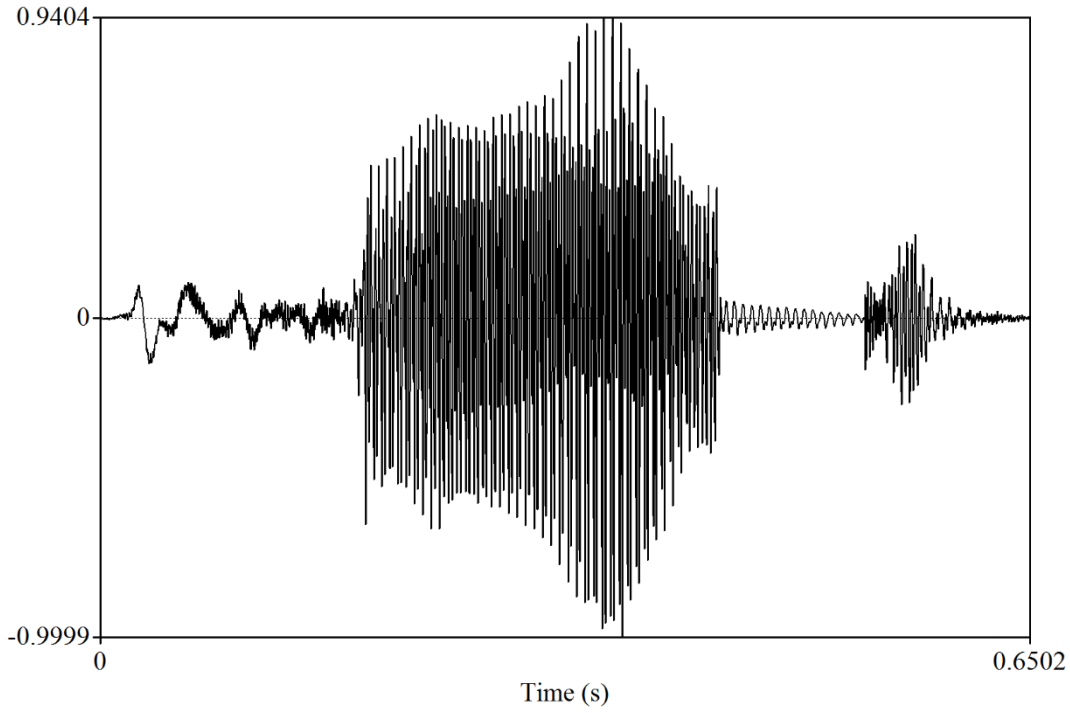

Figure g. Waveform of the utterance 'hood'

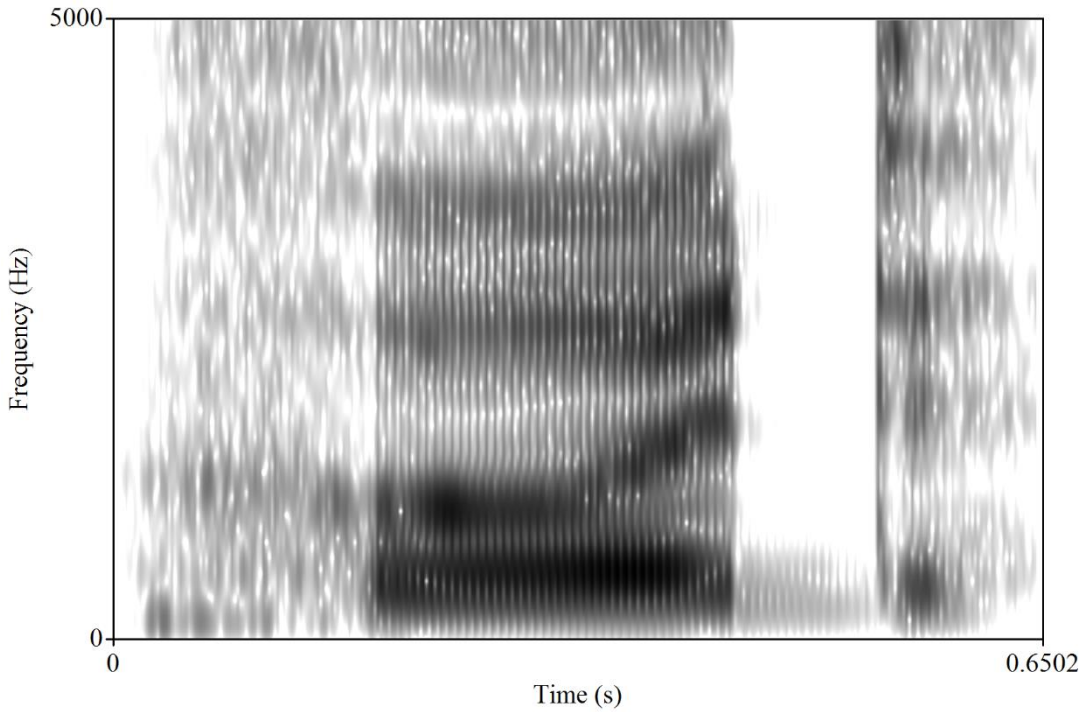

Figure h. Spectrogram of the utterance 'hood'

Figure i. Spectrograms of 'oo' with increasing amounts of contrast in the resolution the network was trained on

Before coming to this series of operations, the data had first been scaled down from the original spectrogram generated by Praat to 300 by 200 pixels. The images were then cut, shearing 60 pixels off each side and 20 off the top and bottom. When none of these operations yielded a better result than random classification, the data was cut once more, taking away 50 pixels on either side. The resulting 80x160 image was then scaled down to 20 by 40 pixels, which allowed the network to learn from the data.

2.2 MODEL

The neural network was created as suggested in the book *Artificial Intelligence for Games* with a learning rate of 0,25 (Millington and Funghe 2009). The network has a variable number of hidden layers, each of 50 nodes. The amount of input layers depends on the input data. In the case of the spectrograms this is the amount of pixels of each spectrogram. The amount of output layers depends on the categories into which the network must sort the input data, for example three output nodes when classifying between ‘iy’, ‘ah’ and ‘uw’.

When passing forward through the network, each node calculates its output from the input values using the following formula:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Where x is the sum of all the input values.

This yields an output of somewhere between 0 and 1, which is then weighted by the weights between nodes. After every forward pass the weights are updated. The update rules for the weights, using *backpropagation*, are as follows:

$$w'_{ij} = w_{ij} + \eta \cdot \delta_j \cdot o_i$$

Where w_{ij} is the weight between node i and node j, η is the learning rate, δ_j is the error of node j, and o_i is the output of node i.

The *error term*, or how ‘wrong’ the output of a node was, is calculated with the following rule for the output nodes:

$$\delta_j = o_j \cdot (1 - o_j) \cdot (t_j - o_j)$$

Where o_j is the network’s output of node j and t_j is the target output of node j.

The error term for the nodes of the hidden layers is calculated with a different rule because it depends on the error of the nodes it is connected to:

$$\delta_j = o_j \cdot (1 - o_j) \cdot \sum_K (w_{jk} \cdot \delta_k)$$

Where o_j is again the output of node j, K is the set of nodes in the next layer, w_{jk} is the weight between nodes j and k and δ_k is the error of node k.

3 RESULTS

3.1 RESULTS OF CLASSIFYING 'IY', 'AH' AND 'UW'

Of the spectrograms, this dataset was easiest for the neural net to classify, due to the significant differences in the spectrograms of the vowels. The highest accuracy of classification was gained when using a network with two hidden layers and no extra contrast added to the data. This yielded a mean of 99.73%, a minimum of 99.13% and a maximum of 100% correct classification during 20 runs of the network. Figure j shows the percentages graphed per amount of contrast for 1, 2 and 3 hidden layers.

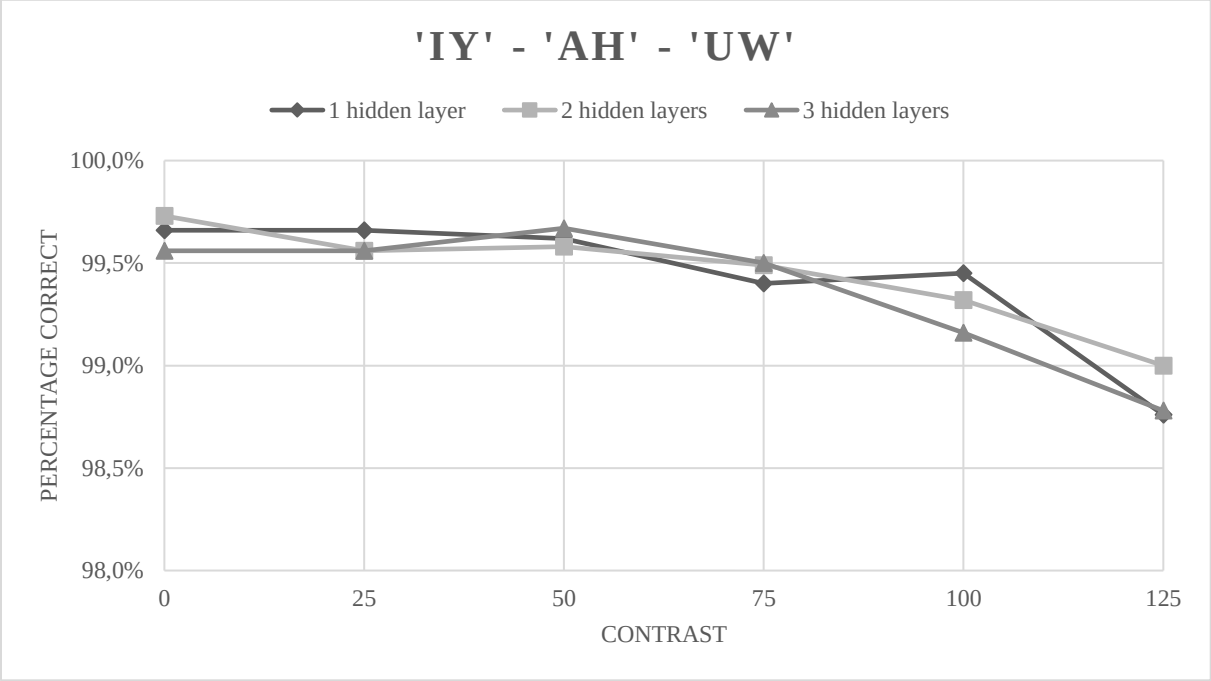

Figure j. Graphs of the data of the spectrograms of 'iy', 'ah' and 'uw'

3.1.1 Effect of hidden layers

As visible in the graphs of figure j, the number of hidden layers has not much of an effect. The data lie close together, often not differing more than a single percent, and the network with the highest accuracy varies.

3.1.2 Effect of contrast

For this particular dataset, the accuracy of the network shows a steady decline as the amount of contrast increases. While still attaining an accuracy of 98.76%, the difference is relatively large between this and the rest of the results.

3.2 RESULTS OF CLASSIFYING ‘IY’, ‘IH’ AND ‘EH’

As previously mentioned, ‘iy’, ‘ih’ and ‘eh’ lay very close in the formant space. This makes them harder to classify, and the accuracy of the network when doing so reflects that. The highest accuracy of classification was gained when using a network with three hidden layers and no extra contrast added to the data. This yielded a mean of 91.87%, a minimum of 84.94% and a maximum of 95.18% correct classification during 20 runs of the network. Figure k shows the percentages graphed per amount of contrast for 1, 2 and 3 hidden layers.

3.2.1 Effect of hidden layers

The number of hidden layers, similar to the ‘iy’, ‘ah’ and ‘uw’ classification task, does not significantly affect the accuracy of the network. Again, the network with the highest accuracy per amount of contrast varies, but in this case less, with the network with a single hidden layer performing slightly lower than the networks with two or three hidden layers. The exception to this is when the contrast is 125; there the network with one hidden layer performs best.

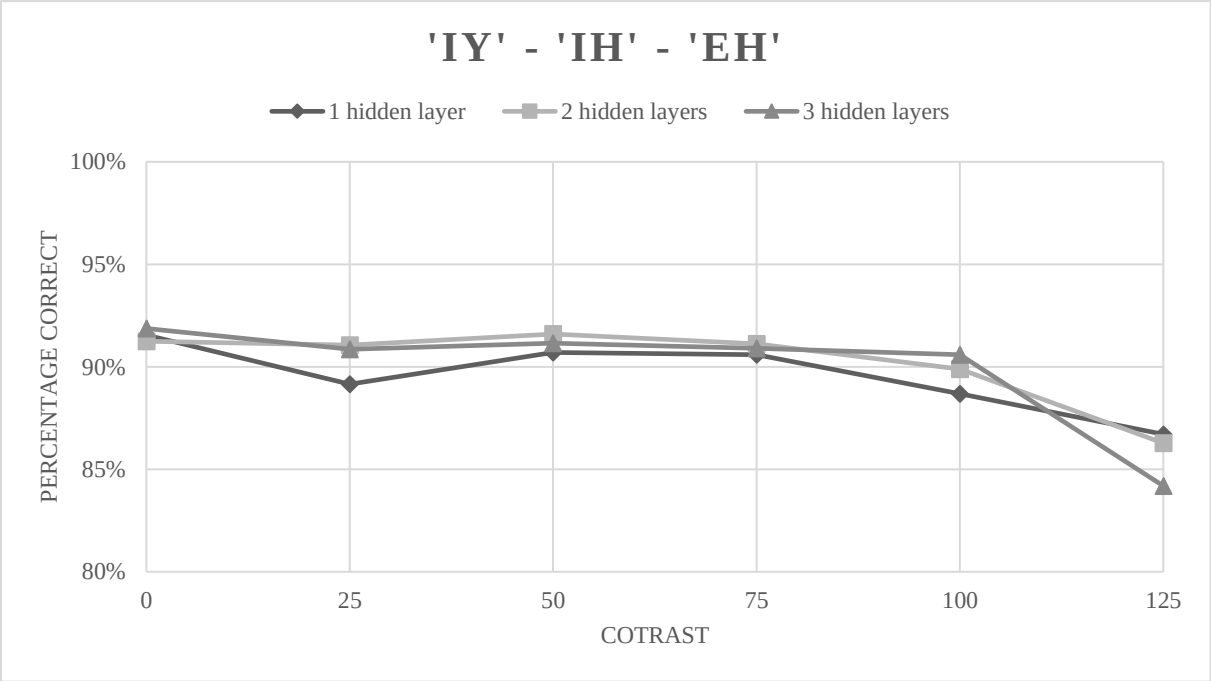

Figure k. Graphs of the data of the spectrograms of 'iy', 'ih' and 'eh'

3.2.2 Effect of contrast

Increasing the amount of contrast has a negative effect on this dataset too. Especially contrast as high as 125 is detrimental to the accuracy of classification. Noticeable in the ‘iy’, ‘ah’ and ‘uw’ dataset and more pronounced here is a dip in accuracy around a contrast of 25. The network with two hidden layers, however peaks at 91.60% accuracy with the higher contrast of 50.

3.3 RESULTS OF CLASSIFYING ALL DATA

Classifying the dataset of all data is the hardest task for the network. The best result was booked with a network with a single hidden layer and a contrast of 25. This yielded a mean of 75.97%, a minimum of 71.08% and a maximum of 81.73% correct classification during 20 runs of the network. Figure l shows the percentages graphed per amount of contrast for 1, 2 and 3 hidden layers.

3.3.1 Effect of hidden layers

As is visible in figure k, the number of layers makes a significant difference when classifying the spectrograms of all twelve vowels. The network with a single hidden layer clearly comes out on top, giving the most accurate classifications regardless of contrast.

3.3.2 Effect of contrast

Unlike with the other datasets, contrast can be beneficial to the classification of all data. Depending on the number of hidden layers, a contrast between 25 and 100 shows a clear improvement compared to adding no contrast at all. As with the other datasets, too much contrast, e.g. 125 is detrimental to the accuracy of classification.

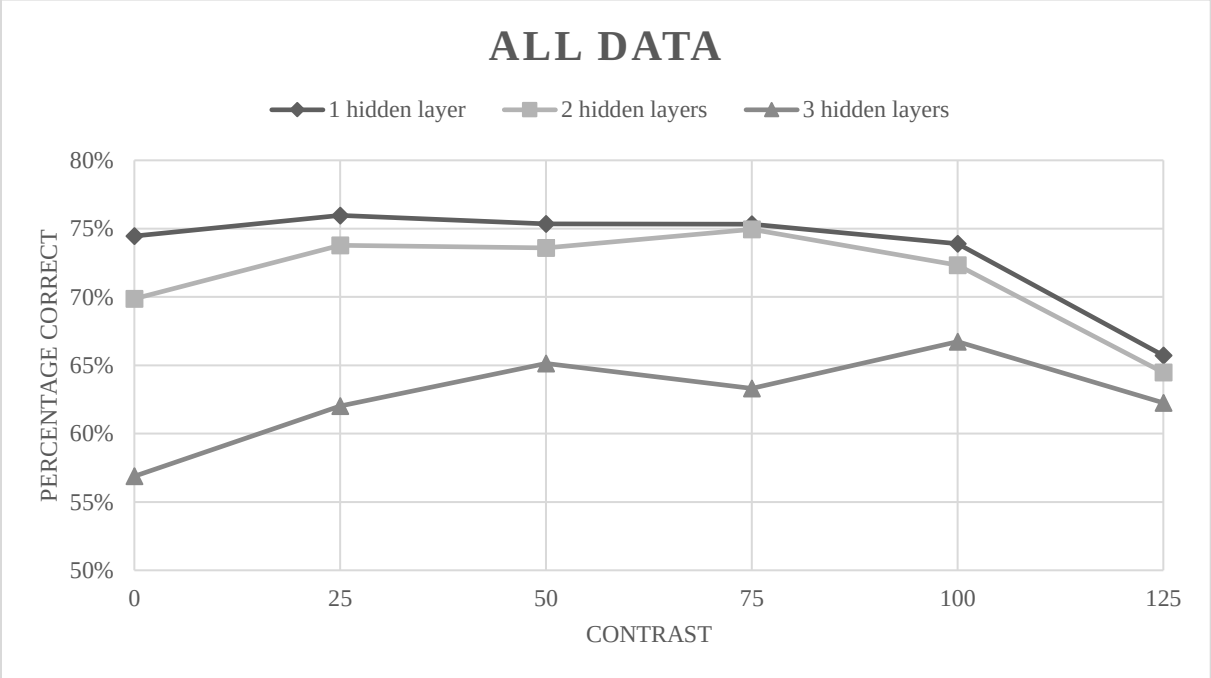

Figure l. Graphs of the classification of all spectrogram data

3.4 EFFECTS OF IMAGE RESOLUTION

A lower resolution gives less data to learn from, and it also downsizes the network considerably. This way the network learns from the most obvious patterns. The result was very apparent when testing the network at higher resolutions: classification was random. The network was unable to recognise any patterns in the data.

3.5 SPECTROGRAMS VS MNIST

3.5.1 MNIST results

The network performs best on the MNIST data with three hidden layers. As expected, the dataset with the three digits that lie furthest apart, ‘0’, ‘1’ and ‘4’, are easiest to classify. The network manages this with an accuracy of 91.65%. Hardest to classify is the entire dataset with a best result of 43.56% correctly classified digits (see figure m).

3.5.2 In Comparison

The network performs much better on the spectrograms than on the written digits. As visible in the table below, which shows the best results achieved by the network, this difference is as large as 32.32% in the case of all data (see table 1). This division is most likely due to how similar data with the same classification is. In the case of spectrograms, the pixel values of the phoneme ‘iy’ from one speaker will not differ wildly from those of another speaker, but with written digits the differences are much more prominent. A seven can be streaked through, for

Classification task	Spectrograms	MNIST
Three very different examples	99.73%	91.65%
Three very similar examples	91.87%	68.57%
All Data	75.97%	43.65%

Table 1. Data comparison of the best results achieved with spectrograms and MNIST datasets

example, or not, just as the top of a five may or may not be connected to its body. These differences are a big deal for a neural network, as they are patterns that occur in the data, and will be learned from. This is evident from the tests done with the network: even when classifying three very different examples of the data, the network is 8% more accurate when classifying spectrograms.

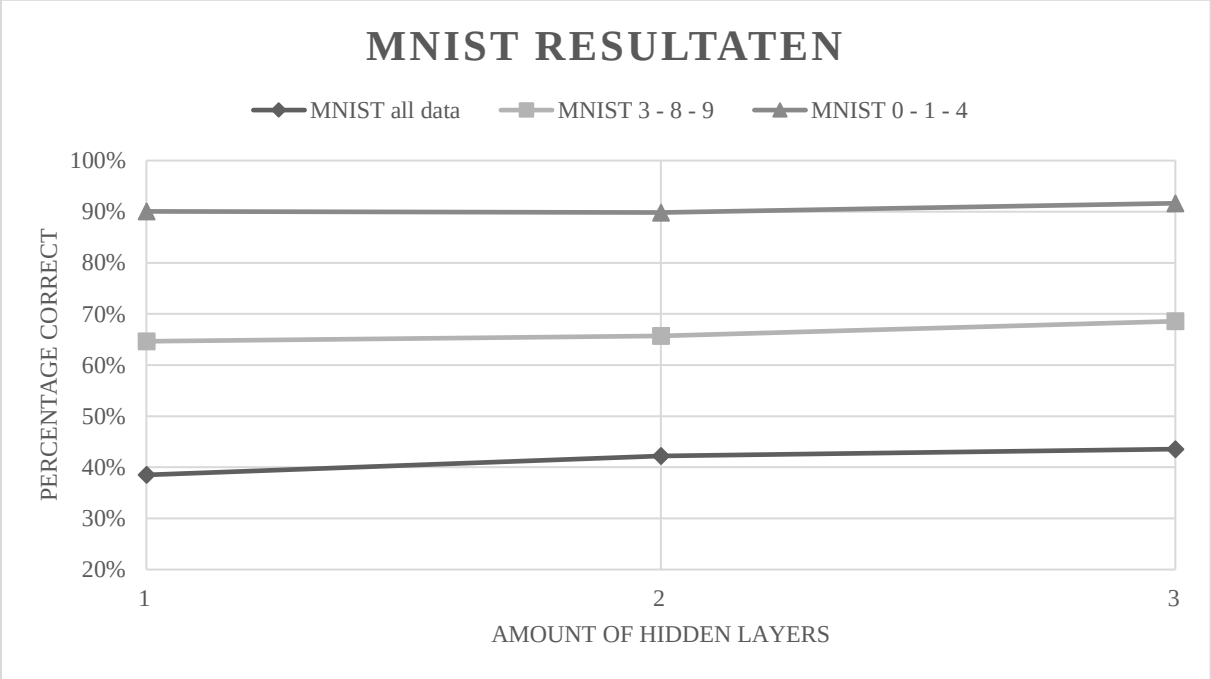

Figure m. Graphs of the classification results of the MNIST dataset

4 SUMMARY AND CONCLUSION

In this paper, we have presented spectrograms as a representation of speech data to train a neural network. To investigate this, we have applied a deep neural network with varying numbers of hidden layers to the task of classifying three sets of vowel data: the set of ‘iy’, ‘ah’ and ‘uw’, the set of ‘iy’, ‘ih’ and ‘eh’ and the set of all data. We have also experimented with different levels of pre-processing of the spectrograms by varying the resolution and level

of contrast to find the parameters optimal classification. Lastly, we compare the classification results gained from applying our network on our spectrogram dataset and on the MNIST dataset, to determine if spectrograms are a suitable training resource for neural networks.

From this we can conclude the following: when classifying spectrograms, the number of hidden layers is best kept to a low number, especially when dealing with many possible classifications. When pre-processing the data, the resolution of the images needs to be very low. An image size of 800 pixels has shown itself to be suitable. Heightening the contrast is a pre-processing step that can be beneficial to classification accuracy. When comparing to the MNIST dataset, it is clear that the spectrograms are even more suited for classification by ANN than the hand-written digits.

The overall best classification results can be found in the table below. When classifying all vowels an accuracy of 75.97 has been achieved. This is, however, a far cry from the classification accuracy achieved by human agents, which is 95.4% when classifying the full dataset (Hillenbrand 1995).

Classification task Spectrograms	Best result	Achieved with…
iy – ah – uw	99.73%	No added contrast; 2 hidden layers
iy – ih – eh	91.87%	No added contrast; 3 hidden layers
All Data	75.97%	Contrast of 25; 1 hidden layer

Table 2. The best results achieved by the network with spectrogram data

Overall, Spectrograms form a valuable way of representing speech data, and are worth investigating further. For future research, it would be interesting to see how well spectrograms hold up when classifying different categories of phonemes, such as plosives or fricatives, and what the effect of applying contrast to such datasets is. Another option is to use convolutional neural networks to the image recognition of spectrograms to further improve the accuracy of classification. The impact of this research is to increase the standards of the accuracy of classification.

This paper shows that there are still many options to explore to improve speech recognition. Near-perfect speech recognition will make the perfect personal assistant an attainable goal in the near future. While J.A.R.V.I.S. is still science fiction, his speech recognition architecture is slowly coming together.

5 REFERENCES

Abdel-Hamid, Ossama, Abdel-rahman Mohammed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. *Convolutional Neural Networks for Speech Recognition.* IEEE/ACM Transactions on Audio, Speech and Language Processing, Vol 22, No. 10, October 2014.

Cooke, Martin, Phil Green, Ljubomir Josifovski, and Ascension Visinho. *Robust automatic speech recognition with missing and unreliable acoustic data.* Speech communication, Vol. 34 No. 3: 267-285., 2001.

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton. *Speech Recognition with Deep Recurrent Neural Networks.* IEEE international conference on Acoustics, Speech and Signal Processing, 2013.

Hillenbrand, James. *Acoustic characteristics of American English vowels.* The Journal of the Acoustical society of America, 1995.

Hinton, Geoffrey, et al. *Deep Neural Networks for Acoustic Modeling in Speech Recognition (the shared views of four research groups.* IEEE Signal Processing Magazine Vol. 29, No. 6: 82-97., November 2012.

Jurafsky, Dan, and James H. Martin. *Speech and language processing.* Pearson, 2008.

Lawrence, Steve, C. Lee Giles, Ah Chung Tsoi, and Andrew D. Back. *Face recognition: A Convolutional Neural Network Approach.* IEEE Transactions on Neural Networks Vol. 8, No 1:98-113, 1997.

LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. *Gradient-based learning applied to document recognition.* Proceedings of the IEEE, 1998.

Millington, Ian, and John Funghe. *Artificial Intelligence for Games 2nd ed.* CRC Press, 2009.

Russel, Stuart, and Peter Norvig. *Artificial Intelligence: A Modern Approach.* 3rd. New Jersey: Pearson, 2010.

Waibel, Alexander, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. *Phoneme recognition using time-delay neural networks.* IEEE transactions on acoustics, speech, and signal processing, Vol. 37 No. 3: 328-339., 1989.

Yao, Kaisheng, Dong Yu, Frank Seide, Hang Su, Li Deng, and Yifan Gong. *Adaptation of Context-Dependent Deep Neural Networks for Automatic Speech Recognition.* IEEE Spoken Language Technology Workshop, 2012.

Zisserman, Andrew, and Karen Simonyan. *Very Deep Convolutional Networks for Large-Scale Image Recognition.* arXiv preprint, 2014.