university of groningen / faculty of science and engineering

University of Groningen,
the Netherlands

Master's Thesis
Artificial Intelligence

Deep Learning for Audiovisual Speech Recognition using the Correspondence Task

Marc Groefsema
s2055236

supervised by:
dr. M.A. Wiering (first supervisor)
prof. dr. L.R.B. Schomaker (second supervisor)
both supervisors are with the Artificial Intelligence department

October 28, 2019

Abstract

We humans can observe the world around us in many ways. We can see things, hear sounds, smell a scent. In multimodal learning a model needs to understand such different types of input, it has to be capable of handling different types of data, e.g. audio samples, video frames or text. Considering robotics also sensor data streams might be usable, e.g. LIDAR data and joint states. This work is in the context of audiovisual speech recognition, using the datasets *Lip reading in the wild* and *Lip reading sentences in the wild*. It explores the usage of the correspondence task as a pretraining and transfer learning technique for word classification and sentence recognition for audiovisual speech recognition. In this correspondence task, the feature extraction modules of the network architectures are pretrained to classify whether an audio and video stream pairing matches the same video, or whether they originate from different videos. The usage of two different modality fusion techniques are considered for correspondence classification, namely classification-based on the distance between modality features or based on concatenated modality features. Here the question is asked whether using the correspondence task results in useful features in a pretraining or transfer learning setting. Apart from this the performances using the different fusion methods are compared. The results suggest that using the correspondence task does indeed lead to useful feature extraction modules for a later classification task in audiovisual speech recognition.

Acknowledgements

I would like to thank dr. M.A. Wiering for the many interesting discussions over the years and supervision during this project. Also I would like to thank prof. dr. L.R.B. Schomaker for his supervision and useful insights.

A special thanks to the BBC, especially Rob Cooper, for granting access to the data used throughout this research.

Also I would like to thank my colleagues at the robotics laboratory, R.P. Timmers, Y. Chong, C.G. Kuiper and R. Niel for the joyful discussions and useful comments.

Finally, I would like to thank my family for their interest and support throughout this research and study in general.

Contents

1 Introduction **9**
- 1.1 Introduction . . . 9
- 1.2 Correspondence task protocol . . . 10
  - 1.2.1 Dataset generation . . . 10
  - 1.2.2 Combining feature learning modules for the different modalities for the correspondence task . . . 10
  - 1.2.3 Combining trained feature learning modules for a different classification task . . . 11
- 1.3 Related work . . . 11
  - 1.3.1 Objects that sound . . . 11
  - 1.3.2 End-to-end audiovisual speech recognition . . . 13
  - 1.3.3 Lipreading sentences in the wild . . . 14
- 1.4 Research questions . . . 14
- 1.5 Thesis layout . . . 14

2 Theoretical background **16**
- 2.1 Deep learning . . . 16
  - 2.1.1 Types of models . . . 16
  - 2.1.2 The multi-layer perceptron . . . 17
  - 2.1.3 Activation functions . . . 17
  - 2.1.4 Loss functions . . . 18
  - 2.1.5 Optimisation . . . 19
  - 2.1.6 Convolutional models . . . 19
  - 2.1.7 Batch normalisation . . . 19
  - 2.1.8 Temporal models . . . 20
  - 2.1.9 Pretraining . . . 21
  - 2.1.10 Transfer learning . . . 21
- 2.2 Audio processing . . . 21
- 2.3 Visual processing . . . 22
  - 2.3.1 Object recognition architectures . . . 22
  - 2.3.2 Learning temporal aspects of video data . . . 22
- 2.4 Sequence to sequence learning . . . 23
  - 2.4.1 Encoding stage . . . 23
  - 2.4.2 Decoding stage . . . 24

2.4.3 Training vs. inference decoding . . . 24
2.4.4 Multimodal sequence learning . . . 24
2.5 Correspondence learning . . . 25
2.5.1 Correspondence classification . . . 26
2.6 Audiovisual speech recognition . . . 26

3 Methodology **27**
3.1 Single word classification . . . 27
3.1.1 Dataset: lipreading in the wild . . . 27
3.1.2 Preprocessing . . . 28
3.1.3 Generating data for the correspondence task . . . 29
3.1.4 Generating data for the classification task . . . 29
3.1.5 Network architecture . . . 29
3.1.6 Optimisation . . . 33
3.1.7 Evaluation . . . 33
3.2 Sentence recognition . . . 33
3.2.1 Dataset: lipreading sentences in the wild . . . 33
3.2.2 Preprocessing . . . 34
3.2.3 Network architecture . . . 34
3.2.4 Correspondence training . . . 35
3.2.5 Optimisation . . . 36
3.2.6 Evaluation . . . 36
3.3 Hardware and software . . . 36

4 Experiments **37**
4.1 Single word recognition . . . 37
4.1.1 Baseline experiments . . . 37
4.1.2 Analysis of the correspondence task using the L2 method 37
4.1.3 The influence of feature normalisation . . . 38
4.1.4 Analysis of the correspondence task using the concatenation method . . . 38
4.2 Sentence recognition . . . 39
4.2.1 Baseline experiments . . . 39

5 Results **40**
5.1 Single word recognition . . . 40
5.1.1 End-to-end training results, without pretraining . . . 40
5.1.2 Analysis of the correspondence task using the L2 method without feature normalisation . . . 42
5.1.3 Analysis of the correspondence task using the L2 method with L2 feature normalisation . . . 45
5.1.4 Analysis of the correspondence task using concatenation . 45
5.1.5 Comparison of the models . . . 47
5.2 Sentence recognition . . . 48
5.2.1 Output inspection . . . 48

6 Discussion **50**
6.1 Answers to the research questions 50
6.2 The sentence-based model 51
6.3 Future work 51
6.3.1 Using multiple languages 51
6.3.2 Using different modalities 52
6.3.3 Learning correspondence from variable length sequence data 52

Appendices **58**

A Words in the LRW dataset **59**

List of Figures

1.3.1 An overview of the network architectures considered in [1]. Here a) shows the visual submodule. b) depicts the autitory submodule. c) illustrates the network used in [1], using the L2 distance for feature combination. d) shows the earlier used concatenation-based architecture used in [2]. This figure was taken from [1]. . 12
1.3.2 The architecture used in *end-to-end audiovisual speech recognition* [3]. This figures was taken from [3]. 13

2.1.1 A diagram of the workings of an LSTM cell. This figure was taken from [4]. 20
2.3.1 The ResNet block as proposed in [5]. This figure was taken from [5]. 22
2.4.1 A general overview of the seq2seq architecture. Here an input "ABC" is being mapped to the output "WXYZ". This figure was taken from [6] 23
2.4.2 The *watch*, *listen*, *attend* and *spell* model used in [7] is an example of a multimodal sequence to sequence architecture. Here video data of lip movements are used as input, together with an audio signal, to output a sequence of characters. This figure was taken from [7]. 25

3.1.1 Some illustrative video frames from the LRW dataset. From left to right the speakers are saying *Weekend*, *Questions*, *Significant* and *About*. 27
3.1.2 An example of cropping, colour conversion and downsampling of a speaker saying "About". 28
3.1.3 The auditory module. The first convolution layer uses a large kernel with a large stride (s), to reduce the large temporal dimension of the audio signal. Later ResNet blocks are used with average pooling. Finally three more convolution layers are used, after which the output is flattened. 30

3.1.4 The visual module, encoding the video frames. After the input, each operation is specified on top of a block, where the resulting output is the block itself. All operations used zero padding, unless no padding was used, specified by "no pad". The arrows in the image depict the residual connections. The dimensions are shown below each block, indicating time × height × width × channels. 31
3.1.5 A diagram of the distance-based combination of modality features and the classifier for the correspondence task. 31
3.1.6 A diagram of the concatenation-based combination of modality features and the classifier for the correspondence task. 32
3.1.7 A diagram showing the feature combination and the classifier of the word classifier. 32
3.2.1 Some illustrative video frames from the LRS dataset. 33

5.1.1 Training performance using audio and video without using any earlier training. This data was gathered using the validation set. 40
5.1.2 Training performance, using the validation set. Figure (a) shows the training performance for solely speech recognition. Figure (b) shows the training performance of only lip reading. 41
5.1.3 Figure (a) shows the training performance on the validation set of the distance-based correspondence task, without feature normalisation. Figure (b) shows histograms of the L2 distance between the modality features for corresponding and noncorresponding data from a single model. 42
5.1.4 The training performance on the validation set. Figure (a) shows the performance using transfer learning. Figure (b) shows the performance using pretraining. 44
5.1.5 Training performance of the correspondence task using concatenation. This data was gathered using the validation set. 45
5.1.6 Training performances on the validation set. Figure (a) shows the performance using the pretraining setup. Figure (b) shows the performances using the transfer learning setup. 46

List of Tables

5.1 Test results of the basic setup without any pretraining, showing the results of AVSR, speech recognition and lipreading. 42
5.2 A table indicating the normal distributions of L2 distances from the distance-based correspondence task. Here μ indicates the mean, where σ indicates the standard deviation. $+$ indicates corresponding data, where $-$ indicates non corresponding data. . 43
5.3 The test set results of the base model, together with the results of the pretraining and transfer learning setup, using the distance-based correspondence task. 44
5.4 The test set results of the base model, together with the results of the pretraining and transfer learning setup, using the concatenation-based correspondence task. 47
5.5 Summary of the test set performances for the different setups. Here DCT denotes the usage of the distance-based correspondence task. CCT denotes the usage of the concatenation-based correspondence task. 47

Chapter 1

Introduction

1.1 Introduction

We humans can observe the world around us in many ways. We can see things, hear sounds, smell a scent. For an AI system to understand such different types of input, it has to be capable of handling different types of data, e.g. audio samples, video frames, and considering robotics also different types of other sensor data streams, e.g. LIDAR data or joint states. In multimodal learning a multiple of such modalities are used as input for a learning model. When learning such models different feature extraction techniques are required for handling the different representations. Also once feature representations are obtained, they have to be combined in a manner, such that a classifier can be trained on the different representations. Often real world events provide multimodal information. For example, when a light cracker goes off, one can see and hear the bang. Many things we see and hear are corresponding to one event in a single point in time. One example of using this correspondence is *soundnet* [8]. Here an audio module is trained to output similar features as a pretrained video module, hence learning useful features for the audio domain.

This work will explore a different pretraining technique, utilising these correspondences in the domain of audiovisual speech recognition (AVSR). Here models will first be trained to distinguish between examples where the audio and visual data originates from the same original video, or where the visual data originates from a different video.

In AVSR speech is perceived from both video, i.e. lip reading, as well as from audio, i.e. listening. For this task the model has to learn useful features, both in the audio and video domain, to be potentially usable for a classifier or language model to map the input video to text annotations.

Multiple well explored techniques could be considered for this task, such as a split brain auto encoder [9], where high level features are learned from one domain, e.g. video frames, such that the resulting latent space representations are useful to reconstruct the other domain, e.g. audio. This work will explore a

less explored technique for potential feature extraction, namely the correspondence task (CT) as introduced in [2]. Here a multimodal classifier is built, with separate feature extraction modules for each modality, which are combined using a combination method. Later in this work different combination techniques for this will be described. This combined feature representation is next given to a simple classifier to classify whether a given sample from one modality corresponds with the sample from the other modality, or not. In [2] and [1] the authors illustrated that this technique can lead to useful feature learning in both domains, hence could be used as a proxy objective for feature learning for a later different task, e.g. classification of a word being said in the video.

1.2 Correspondence task protocol

1.2.1 Dataset generation

As described earlier, the correspondence task (CT) is a framework where a classifier is trained to distinguish between data from separate modalities that correspond, and data where the modalities don't correspond. For AVSR, data can easily be generated for this task, given videos of people that are speaking, and a technique for isolating lip movement from the speaker. Given many videos, one can mix up the audio from a given video with another video, given equal sample sizes with respect to time, i.e. an equal number of audio samples and video frames. Given a 50% chance of making a non-corresponding example, a balanced dataset can be created for this task.

1.2.2 Combining feature learning modules for the different modalities for the correspondence task

Since features have to be extracted from each modality separately, separate stacks of layers are used to extract features from the raw data. The final output of these separate modules have to be combined, such that a single vector can be provided to a classifier.

A trivial combination method is concatenation [2]. Here both modality vectors are concatenated to each other where the resulting vector is provided to a multi-layer perceptron (MLP) [10]. A different method is proposed in [1]. Here a L2 distance is taken from both vectors, reducing the provided data to a single value. This value is given to a small MLP, with a single hidden layer of 3 units, followed by two output units after which a softmax layer [11][12] is used for final binary classification.

Before combining both modality vectors, considerations have to be made. If one concatenates two vectors, where one is much larger than the other, the classifier output could be influenced more by the output of the larger modality vector than by the smaller one. Also, taking an L2 distance between the two modality vectors requires both modality vectors to be of equal size. Because of these reasons only modality vectors of equal size will be considered. This

latter technique will be referred to as the L2-method. The L2-method might have interesting properties. If a high accuracy is achieved on the CT using this technique it is possible that the L2 distance between the modality vectors are minimised during training for cases where the modalities do correspond. If this distance indeed reaches near zero, it could be argued that both modality inputs are being mapped into the same latent space.

1.2.3 Combining trained feature learning modules for a different classification task

Once the modality modules are trained with the CT, this knowledge has to be usable for different classification tasks, as the CT is merely a learning proxy. For this, again a combination method is required. For this task concatenation is again trivial in the case if the CT learning was done with a concatenation method, as well for the case if the CT learning was done with the L2-method. If the CT learning was done with the L2-method, also different combination methods might be possible here. If it is the case that the L2-method learns to map corresponding data to the same latent space, it might be considered taking the average of both modality vectors for the final classification task.

1.3 Related work

1.3.1 Objects that sound

A paper strongly related to this work is *objects that sound* [1]. Here the authors expand on their earlier work in [2], where the correspondence task was originally introduced. The authors considered the network architectures as shown in figure 1.3.1, where the network was trained for source localisation for a given audio signal, as well as for crossmodal retrieval.

L^3-net

Look listen and learn net (L^3-Net) was introduced in [2]. Here the network as shown in figure 1.3.1, was trained using the Flickr-Soundnet dataset [8], containing a large collection of unlabelled videos from Flickr, of which 500k videos were used. The training procedure was set up by constructing examples where the audio and video stream do correspond, and by constructing examples where the audio and video do not correspond. For these respective cases the one hot encoding label for corresponding or not corresponding was used. While the videos contain many frames, only one frame was used for the task. Based on this information the network was trained. Using this technique, here referred to as the correspondence task, the network was able to learn correlations between the audio and video modalities. To demonstrate the effectiveness of the learned feature representations from the sound and video, both submodules were used in a transfer learning setting. Here a trained module was followed by an MLP using a single hidden layer for classification. The visual module was evaluated on the

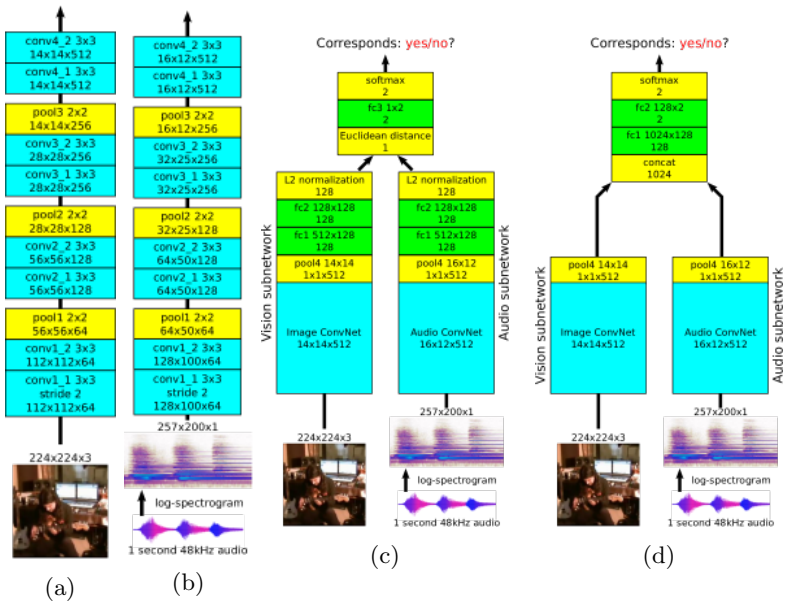


Figure 1.3.1: An overview of the network architectures considered in [1]. Here a) shows the visual submodule. b) depicts the autitory submodule. c) illustrates the network used in [1], using the L2 distance for feature combination. d) shows the earlier used concatenation-based architecture used in [2]. This figure was taken from [1].

ImageNet dataset [13], containing 1000 classes. Here the network obtained a result of 32.3% accuracy, compared to 12.9% using a randomly initialised visual module. The result with their transfer learning setup performed comparable to other unsupervised feature training techniques. The audio module was used in a transfer learning setup for two different datasets. On the ESC-50 [14] dataset, containing 50 classes, the network obtained an accuracy of 79.3% compared to 62.5% using a randomly initialised audio network. On the DCASE dataset [15], containing 10 classes, the network obtained an accuracy of 93% compared to 85% using a randomly initialised network. These results indicate that the usage of the CT results in usable modules for other tasks such as classification.

AVE-net

As the previously described approach was introduced in [2], the authors expanded on this work in [1] using an audiovisual embedding network (AVE-Net) as shown in figure 1.3.1. Instead of concatenating the modality features, the features were first normalised using L2-normalisation. After this the L2-distance was taken from these normalised features, on which a small MLP with one hidden layer was used for correspondence classification. This setup forced the architecture to learn similar embeddings for both the audio and visual modali-

ties. This architecture was trained using data from the Audio Set dataset [16].

While the network was trained using the CT, the performance was evaluated based on crossmodal retrieval. Here a key-value database of audio and images was constructed, where the key is the learned modality embedding, and the value the actual image or audio sample. Given this database, together with a collection of images and audio samples not included in this database, crossmodal retrieval can be evaluated. For example, using the embedding of an audio sample as a querry, which images in the database could have produced this audio sample? For video on the other hand, using the embedding of an image, which audio could have been produced together with this image? The authors also compare the results on the CT itself between L^3-Net and AVE-Net. Here the AVE-Net slightly outperforms L^3-Net with an accuracy of 81.9%, compared to 80.8% for L^3-Net.

1.3.2 End-to-end audiovisual speech recognition

A fully neural multimodal AVSR system for single word classification was proposed in [3]. Here the audio stream was encoded using a one dimensional variant of ResNet-18 [5]. The video stream was first encoded using a single 3D convolutional layer, followed by a ResNet-34 architecture. The ResNet architecture was followed by two bidirectional GRU layers [17], after which the states are concatenated, being followed by another two bidirectional GRU layers, after which a softmax layer was used for the final classification. This architecture is illustrated in figure 1.3.2.

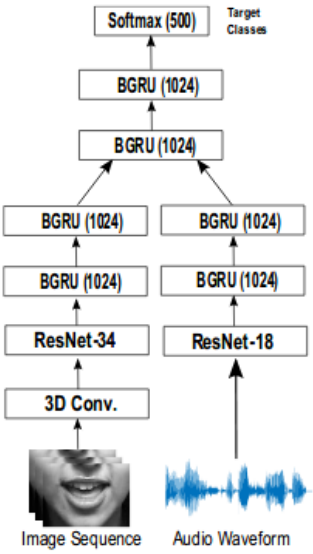


Figure 1.3.2: The architecture used in *end-to-end audiovisual speech recognition* [3]. This figures was taken from [3].

Using the lipreading in the wild (LRW) dataset [18], the model achieved an accuracy of 98.0% using both modalities. Comparing the performance with only audio between the earlier described audio module with the usage of MFCC encoding of the audio, the authors found no significant differences in performance, achieving both an accuracy of 97%.

1.3.3 Lipreading sentences in the wild

A sequence to sequence (seq2seq) [6] model for audiovisual speech recognition was proposed in [7], where the authors introduce a multimodal speech recognition model for variable length text sequences, as well as a dataset for audiovisual speech recognition for variable length sequences. The proposed model extends on the seq2seq model as proposed by [6], by using two separate encoders, one for each modality. Together with the seq2seq architecture, the model uses Bahdanau attention [19] for each modality, together with beam search [20][6]. As a training procedure, the model was first trained on single word examples, whereafter the model was trained on longer sentences, as it significantly helped for convergence of the model. The proposed model achieved a character error rate (CER) of 7.5% and a word error rate (WER) of 13.9% on the proposed dataset.

1.4 Research questions

This thesis will address the following research questions:

- Does the correspondence task yield feature extraction modules that are useful for AVSR? (1.1)

- Do the performances differ between using concatenated modality features or using the distance between modality features in the correspondence task? (1.2)

- Do the performances differ between using normalised or raw modality features in the distance-based correspondence task? (1.3)

- Does the distance-based correspondence task learn to map both modality features to the same latent space? (1.4)

The questions above will be addressed in the context of single word classification and sentence recognition.

1.5 Thesis layout

The rest of this thesis will be structured as follows. The next chapter will discuss the theoretical background. Here some explanations are provided on the

techniques used throughout this work. Next the methodologies are discussed, describing the used datasets, together with an overview of the used model. This chapter will be followed by the experimental setup, describing the experiments used to address the research questions. Hereafter the results of these experiments are discussed. Finally the research questions are addressed in the discussion, together with potential future work.

Chapter 2

Theoretical background

This chapter will briefly describe the different techniques used throughout this thesis. First a brief description of deep learning (DL) will be given, together with the different DL techniques used throughout this study. This is followed by a small overview of the correspondence task (CT). Finally the chapter is concluded with some earlier approaches regarding audiovisual speech recognition (AVSR).

2.1 Deep learning

Deep learning [21] refers to artificial neural network models containing many layers, each one representing the data in a more abstract manner as the layers stack. The general concepts used in DL are relatively old, going back to the perceptron [22] in 1958. As computers started to become more powerful in terms of floating point operations per second, with more available working memory, combined with large datasets, deep neural network architectures started to perform better than more classical machine learning techniques. Here often feature detectors were handcrafted for a classification task at hand, whereafter representations using these features were provided to a classifier such as a support vector machine [23]. This section will cover techniques used in DL, as they form the basic building blocks for the DL architectures used in this work.

2.1.1 Types of models

Exploring deep learning models, one can differentiate between different types. The simplest form of a DL model would be classification. Given some input data, the model has to map the input to a certain class. A simple example of this would be a model that receives images of animals as input, where the model has to output which animal is seen in the image. This specific task is referred to as object recognition. Also generative models exist [24][25], where e.g. given a class label, say 'cat', the model has to generate an image of a cat, after having seen many examples of the animals it has to be able to generate.

Apart from working with single target labels, also more complex targets exist, such as localising an object in an image, i.e. object detection. Also other models exist which do pixelwise classification of images of a scene, i.e. semantic segmentation [26].

The models described so far are mainly concerned with single image inputs. When data points, e.g. images or sound samples, are distributed over time, we speak of temporal models. These models are able to perform different kinds of tasks regarding temporal data. An example of this is speech recognition, where sequences of audio intensity are mapped to a sequence of characters or words.

Apart from these examples, many more usages of DL exist, e.g. deep reinforcement learning, although many relevant types have been mentioned before. The rest of this section will provide a more in depth overview on the working of the building blocks in DL as being used in this work.

2.1.2 The multi-layer perceptron

One of the simplest forms of an artificial neural network used is the multi-layer perceptron (MLP) [10]. An MLP consist of multiple layers, starting with an input layer, containing a 1D representation of the input values. This layer is followed by multiple layers of multiple neurons, where each neuron in a layer has weights connected to all neurons from the previous layer. The mathematical representation of a single layer is defined as shown in equation 2.1.

$$f(\vec{x}) = act(\mathbf{W} \cdot \vec{x} + \vec{b}) \tag{2.1}$$

Here $\vec{x}$ represents the input vector or the output of the previous layer. $\mathbf{W}$ represents the weight matrix, containing weights for each neuron to each neuron from the previous layer. $\vec{b}$ represents a vector of bias values. In equation 2.1 act represents the activation function of the neurons.

2.1.3 Activation functions

A common vanilla activation function in the MLP is the sigmoid function, as defined in equation 2.2.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

As a final layer for classification, most often, as well as throughout this work, a softmax function is used [11][12], as defined in equation 2.3.

$$Softmax(y)_i = \frac{e^{y_i}}{\sum_{j=1}^{K} e^{y_i}} \text{ for } i = 1, ..., K \tag{2.3}$$

Here y denotes the output values, and K the total number of classes.

Apart from the sigmoid function, many other activation functions are considered in artificial neural networks. A subset of these activation functions are

used throughout this work. One very popular function is the rectified linear unit (ReLU), which is defined as shown in equation 2.4.

$$f(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$

The derivative of the ReLU function is defined as shown in equation 2.5.

$$\frac{\partial f}{\partial x} = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.5}$$

A favourable property of the ReLU function compared to the sigmoid function is that partial derivatives of many layers of ReLU function, with arguments larger than zero will always be one. Using many layers with sigmoid functions will results in diminishing gradients, as its derivative will be maximally 0.25, but most often smaller. Apart from sigmoid and ReLU, the tanh function will be used as an activation function, as described in equation 2.6

$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2.6}$$

2.1.4 Loss functions

Machine learning (ML) architectures are designed to adapt their internal parameter set in such a manner that a loss function is minimised. This loss function directly corresponds to the task at hand for the architecture. For example, if a model is constructed to perform a classification problem, the loss function indicates how 'wrong' the output of the model currently is. By minimising this loss, the model gradually performs less 'wrong', hence performing better on the task at hand. Here the loss function used throughout this thesis will be briefly discussed.

Cross entropy loss

A widely used loss function for classification problems is the cross entropy loss function, which is defined in equation 2.7.

$$CE(p, q) = -\sum_{x \in \chi} p_x \cdot log(q_x) \tag{2.7}$$

Here p_x denotes the true probability distribution of the given example, which is indicated by x. Next q_x is the estimated probability distribution provided by the model and χ denotes the set of examples. By minimising the cross entropy loss, the predicted class distributions are forced to become more similar to the true class distributions.

2.1.5 Optimisation

Given a model, example data and a loss function, an optimisation problem can be defined. A popular technique for optimisation is stochastic gradient descent [27][28], where for each tunable parameter the gradient with respect to the loss is calculated. Next, these parameters are updated in the negative direction of the loss gradient, such that the loss slightly decreases. The update is performed with a fraction of the gradient magnitude, referred to as the learning rate. Next to stochastic gradient descent, more optimisation methods have been proposed, e.g. using momentum. A popular technique and also used throughout this work is the Adam optimiser [29]. An overview of popular optimisation methods is provided in [30].

2.1.6 Convolutional models

Convolutional neural networks (CNN) were introduced in [31]. While MLPs are very powerful function approximators, the number of weights will become very large, using large inputs, e.g. large images. The most popular approach against this problem is the usage of a sliding window together with feature detector kernels. The moving window approach is specified by a window size and stride, and a decision of whether the window should always remain within the boundaries of the image, or whether a padding method should be applied to have as many window positions as the input image has in 2D coordinates. Together with the moving window, feature detector kernels are used. Each kernel can be seen as a neuron, with weights fully connected to all values within the window, followed by a bias parameter and an activation function. Using the moving window approach together with this kernel, the output will be a 2D map, referred to as a feature map. Using multiple kernels, the output of this operation given a multi channel 2D input will be a multi channel 2D output, where each channel refers to a feature map of a different kernel. As it is possible to repeat this operation with different sets of kernels, CNNs with multiple layers can be constructed, where the early layers are able to learn fine detail patterns in the input, while higher layer kernels are able to learn to recognise large scale patterns in the image.

2.1.7 Batch normalisation

A popular method to help deep neural networks to converge quicker and also help against overfitting is batch normalisation [32]. This is a normalisation technique for a neuron, where the usage of this neuron over the different examples in the mini batch is considered. In this work batch normalisation will be used after the activation function, in contrast to [32]. As shown in equation 2.8, the outputs of a neuron are first normalised, given the data from a mini batch.

$$\hat{a}(x) = \frac{a(x) - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{2.8}$$

Here $a(x)$ denotes the output of the neuron for a given example, μ is the mean and σ represents the variance. ϵ is a small value to prevent division by zero. Finally, $\hat{a}(x)$ is the normalised activation. This operation causes the activation distribution to be centered at zero, with a unit variance. Next two learnable parameters are introduced to scale and shift this distribution, as shown in equation 2.9.

$$y(x) = \gamma \hat{a}(x) + \beta \tag{2.9}$$

Here γ denotes a parameter to scale the normalised activation distribution, and β is a parameter to shift the distribution.

The shift of this activation distribution is generally referred to as a covariance shift. Because of this shift the outputs of a hidden layer will generally vary much less during training, requiring smaller weight updates in the next layer, making the general model easier to train. As the shift of the activation distribution is parameterised by β, no biases are used within the activation function of the neurons when they are followed by batch normalisation.

2.1.8 Temporal models

Long short term memory

The long short term memory (LSTM) was originally introduced in [33] The LSTM consists of a memory cell, an input gate, an output gate and a forget gate. A depiction of the LSTM mechanism is shown in figure 2.1.1. At every

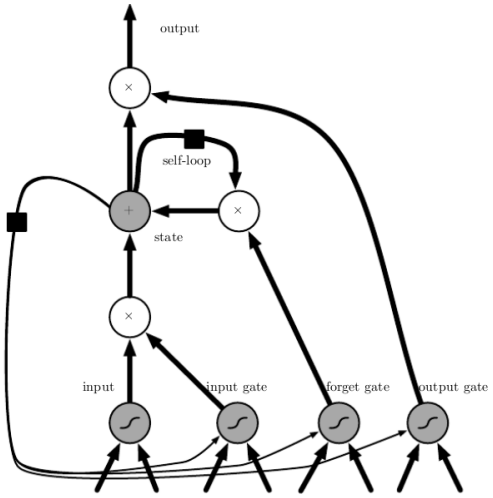


Figure 2.1.1: A diagram of the workings of an LSTM cell. This figure was taken from [4].

time the cell receives new input, this input is given to a neuron with a tanh activation, depicted as *input* in the diagram. The input is also provided to the

input, forget and output gate, which use a sigmoid activation. Simply put, the input gate controls how much of the input part should propagate to the new state. The forget gate controls how much the previous state should contribute to the new state. The output gate controls what fraction of the new state should be the output of the cell.

2.1.9 Pretraining

Pretraining is a technique where the lower layers, considered to learn useful representations from the data, from a deep architecture are already trained on a different training task. In [34], the authors pretrained an MLP with multiple hidden layers, by training them as restricted Boltzmann machines on the data. After pretraining the network the network is trained on the dataset at hand. This time the new top layers still have to be trained, but should converge faster than with randomly initialised weights for feature extraction. Also, during this training phase, the pretrained layers can be fine tuned for the new task at hand, to obtain a potentially higher performance.

2.1.10 Transfer learning

Transfer learning was introduced in [35], and more recently discussed in depth in a survey in [36]. Here a model is trained on a particular dataset, for which the model learns to produce useful feature representations for the input. The concept of transfer learning is using these learned feature representations for a new problem. This approach was shown to work remarkably well. A large benefit of this approach is that it is relatively cheap to train on a new dataset. As the feature extraction parts are already trained, one can map the dataset of the problem at hand to the respective latent space representation once, after which only a relatively small classifier has to be trained, which is a lot cheaper than training the entire network from scratch or using the pretrained parameters for fine-tuning.

2.2 Audio processing

Temporal convolutions

As a contract to the usage of log spectograms, as used in [1] and [2], the models used in this work solely use neural techniques on the raw data, similar to [3]. Here a 1D convolution is performed over an audio sample, often with a relatively large kernel, to encode the nuances in the signal. After this operation more layers with 1D convolutions are used. Often pooling layers are introduced between convolution layers to reduce the temporal dimension of the data, in benefit of saving computational memory.

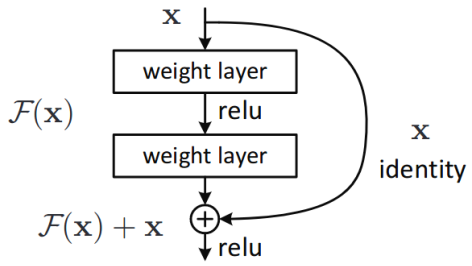


Figure 2.3.1: The ResNet block as proposed in [5]. This figure was taken from [5].

2.3 Visual processing

Visual processing is one of the most popular applications in DL. Here a wide scale of problems is defined regarding extracting useful information from images. One example of these problems is object recognition, for which the application of DL has become very popular since the publication of [37], where DL achieved state of the art performance on the ImageNet dataset [13]. Here a deep CNN was used to classify between 10,184 classes using 8.9 million images from the fall 2009 dataset in ImageNet. Here an error rate of 67.4% was achieved on the test set using the top-1 outputs, and 40.9% on the top-5 outputs, outperforming earlier state of the art, having errors rates of 78.1% and 60.9% on the top-1 and top-5 respectively. Since then many CNN architectures have been proposed for object recognition.

2.3.1 Object recognition architectures

ResNet

In [5], the authors introduce a residual learning framework for deep CNN architectures. Here 'blocks' of layers are used, as shown in figure 2.3.1. The output of the first layer in the block is added to the output of the last layer of the block, after which another ReLU activation is used. These connections throughout an architecture with many layers cause the network to be optimised more easily, while also resulting in higher performances than without the residual connection [5].

2.3.2 Learning temporal aspects of video data

As described earlier, 2D convolutions with 3D kernels can learn to encode local spatial structures in image-like data. When the data is video, i.e. a stack of video frames, it is possible to generalise to a 3D convolution with 4D kernels. These kernels are able to encode local spatial structures, as well as the local change of it over time. The usage of this technique was introduced in [38] and more recently discussed in [39].

2.4 Sequence to sequence learning

The sequence to sequence (seq2seq) architecture was introduced in [6]. Here a variable length input signal is mapped to another variable length encoding. The authors used this neural architecture for neural machine translation, although it can be used for more problems regarding sequences, with variable input and output length. Examples are speech recognition [20], text summarising [40] and AVSR [7].

To achieve these objectives, the problem is broken down in two parts, encoding and decoding. At the encoding state the input signal is reduced to a single "thought vector". Next this thought vector is given to a decoder stage, which composes a new sequence, which should converge to the target sequences An overview of this architecture is shown in figure 2.4.1.

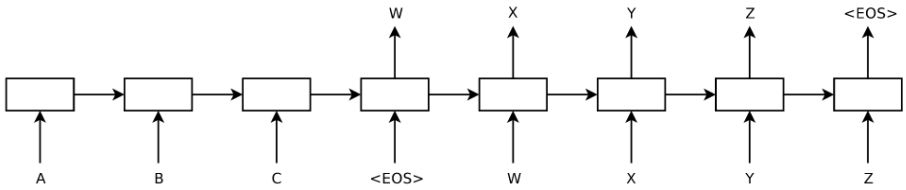


Figure 2.4.1: A general overview of the seq2seq architecture. Here an input "ABC" is being mapped to the output "WXYZ". This figure was taken from [6]

2.4.1 Encoding stage

The encoder stage receives the input sequence, where the elements in the sequence are first represented by some sort of an encoding.

Character encoding

For sequences of tokens, which could be representing e.g characters or words, the tokens first have to be encoded. This is done by encoding each token by a fixed number of values, which are randomly initialised. The values for these encodings are later updated during the training phase of the network.

Feature extraction

If the input sequence of the network is not based on tokens, but on others forms of data such as audio intensity values or images, a feature extraction module is used first. This module could be a CNN architecture for sound or images. The resulting feature representation from this module is then provided to the seq2seq encoding stage.

Temporal encoding

The temporal encoding, depicted as the "ABC" part of the network as shown in figure 2.4.1, is constructed using one or more LSTM layers. The output states at the end of the sequence are considered the output of the temporal encoding. Apart from using LSTM cells in this module, also other variations exists using other memory cells, e.g. gated recurrent units (GRU) [17].

2.4.2 Decoding stage

The decoding stage is also designed using one or more layers of memory cells, e.g. using LSTM or GRU cells. This stage typically decodes the vector from the encoding stage to a token sequence, e.g. characters or words. The final states of the encoder model are given to the decoder stage as initial states of its temporal memory cells. The output of these layers at each time step are followed by an MLP, which performs classification to decide which token to output. The representation of this token is constructed in the same manner as in the encoding stage, but here with its own encoding matrix. After the classification at this time step, the encoded representation of the classified token is fed as input to the decoding stage at the next time step, as shown in figure 2.4.1. Apart from the last encoder state, the classifier of the decoder also receives a weighted output of the outputs of the encoder of each time step, using the Bahdanau attention mechanism [19]. This ensures that the decoder is able to handle larger input sizes with respect to time, which is hard to do for the decoder without receiving the outputs over time from the encoder stage. In [7] the authors note that they were unable to train their AVSR sentence-based system without using attention.

In a setting where the decoder generates text, the decoder stage can be seen as a language model. E.g, if a decoder outputs single characters, the knowledge of words, spelling and grammar is learned by the decoding stage. The content of what the text should say is provided as initial states of the temporal cells in the decoder during the first time step.

2.4.3 Training vs. inference decoding

An important detail to note on the seq2seq architecture is the difference between training and inference in the decoder stage. As shown in figure 2.4.1, at each time step the output of the previous decoding step is used as the input for the next decoding step, except for the first step, where a *start of sentence* (SOS) token is used. This is true during inference. During training, also the SOS token is used at the first step. For later steps the output of the decoder from the previous time step is omitted. Instead the ground truth token from the previous step is used as input for each step.

2.4.4 Multimodal sequence learning

A multimodal example of a seq2seq architecture is the model used in [7], named the *watch*, *listen*, *attend* and *spell* architecture, which is depicted in figure 2.4.2.

Here two separate encoder stages are used for video and audio encoding, referred to *watch* and *listen* stage. The output states of these modules are concatenated, after which they are given to the decoder stage, referred to as the *spell* stage. The temporal outputs of the *listen* and *watch* stages are provided to the attention mechanism of the decoder, referred to as the *attend* stage.

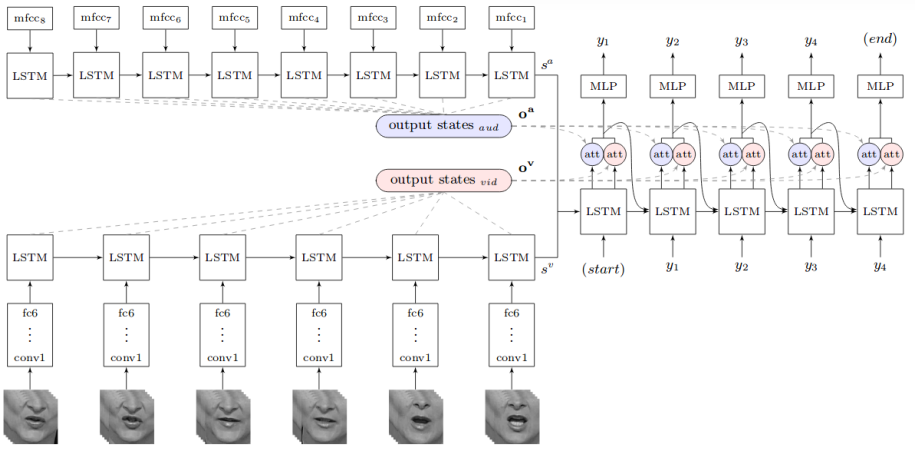


Figure 2.4.2: The *watch*, *listen*, *attend* and *spell* model used in [7] is an example of a multimodal sequence to sequence architecture. Here video data of lip movements are used as input, together with an audio signal, to output a sequence of characters. This figure was taken from [7].

2.5 Correspondence learning

As discussed in the introduction, the correspondence task (CT) was introduced in [2]. The CT is a technique that enables learning representations from multiple modalities, by using a classification problem as whether both modalities correspond to the same event or not. For a model to achieve a high accuracy on such a task, it has to be able to obtain an understanding of the signal for a modality to do so. A major advantage of this technique is that it does not require detailed annotation of the signal for each modality. For example, given the task of audio visual speech recognition, no annotations of the data would be required to learn useful representations of the modalities, as one can simple mix up the visual and auditory signal between of the examples to construct negative examples of correspondence. This technique was used for audio localisation in the video signal, as in [2] and [1] and tasks in audiovisual scene analysis as in [41]. Not much work could be found regarding usage of this technique. No earlier usage of this technique was found in the context of audiovisual speech recognition.

2.5.1 Correspondence classification

To perform classification on whether the modality feature should correspond or not, two techniques are required. First one has to decide on how to combine the feature representations from the different feature extraction modules. Second a classifier is needed to classify this combined feature representation to whether the modalities correspond to the same event. In [2], the modality features were concatenated to each other to form one large representation of the combined modalities. A depiction of this architecture was shown in figure 1.3.1. This combined representation is given to an MLP to perform the classification. In later work [1], the authors used the L2-distance between the modality features, resulting in a single distance value for a given example. This distance value was given to a small MLP, with a single hidden layer which performed the classification part of this task.

2.6 Audiovisual speech recognition

AVSR is an interesting field, as apart from purely auditory-based speech recognition systems, it also incorporates the visual modality. For humans this is very usable, as it allows for using vision to enhance a potentially very noisy audio signal [42][43]. An example of this is the famous cocktail party effect [44]. Deep learning approaches related to AVSR are speech recognition models [45] and lip reading models [46]. The first paper introducing AVSR was published in [47]. Here the authors demonstrated that using visual information of lips next to audio yielded a higher performance than only using the audio signal.

Regarding the visual element of AVSR, regions of interest of the lips have to be encoded to be useful for recognising the words. Also the audio element in AVSR needs a method for encoding the audio signal. After these encodings have been acquired, a fusion method is used to obtain a single representation for both modalities. In [48] the authors list multiple different techniques for feature fusing, often being used for systems using hidden Markov models (HMM). Examples are plain feature concatenation [49], feature weighting [50][51] or hierarchical discriminant feature extraction [52]. For the final classification often HMMs are used [53][54] or otherwise support vector machines [55][56].

Chapter 3

Methodology

This chapter will introduce the methodology used to perform the experiments described in the following chapter. First used methods such as the dataset and the respective network architecture are described for experiments regarding single word classification. This is followed by the dataset description and network architecture for sentence recognition.

3.1 Single word classification

This section will describe the methodology used regarding single word detection. For this task video data has to be mapped to a single word. First the dataset with its specifications will briefly be discussed, followed by the used preprocessing techniques. After this the generation procedure of the data on which the models are trained is discussed. Finally the model architecture is described.

3.1.1 Dataset: lipreading in the wild

Figure 3.1.1: Some illustrative video frames from the LRW dataset. From left to right the speakers are saying *Weekend*, *Questions*, *Significant* and *About*.

The dataset *lipreading in the wild* (LRW) was introduced in [18] for audio-visual speech recognition (AVSR), containing around 1000 short videos for 500 different words each in the trainset. The validation and test set contain 50 examples for each class. As an illustration some video frames are shown in figure

3.1.1. The different target words are shown in appendix A. Every video contains 29 video frames, together with 56448 audio samples, which is convenient with respect to the dataset preparation for generating non-corresponding examples of audio and video mixes.

3.1.2 Preprocessing

Video preprocessing

The video frames are processed with the following steps. First crops of the lips from the speakers are extracted. This is done by utilising the alignment provided by the dataset itself. The dataset is generated such that the middle of the lips from the speakers are always located at coordinates [163, 127]. Given this point, the surrounding environment with the shape of 101×101 is extracted to be considered as a lip patch. Next this patch is downsized using bicubic interpolation to a shape of 32×32. An illustration of this cropping is shown in figure 3.1.2.

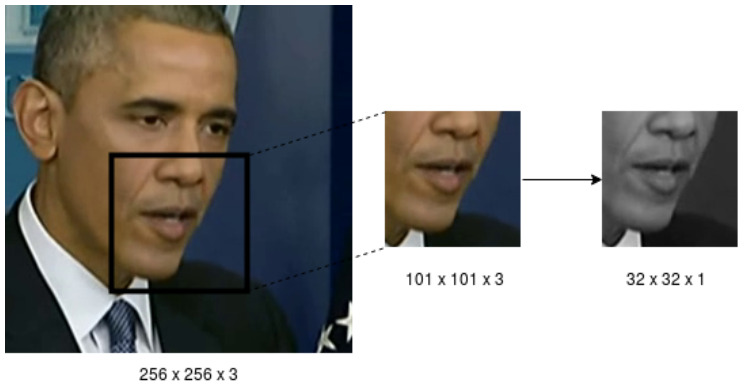


Figure 3.1.2: An example of cropping, colour conversion and downsampling of a speaker saying "About".

This procedure is performed for all frames in the video. Once collected all patches are mapped to greyscale format using the OpenCV [57] colour mapping method. Finally the value of all pixels in the patches are divided by 255 to result in pixels with values between 1.0 and 0.0.

Audio preprocessing

Videos from the LRW dataset are provided with stereo audio. The first step of preprocessing is utilising only the audio from the left channel. Further preprocessing is performed by simply dividing the sample values by the standard deviation within the collection of audio samples from a single video. This is followed by subtracting the average value of all sample points.

3.1.3 Generating data for the correspondence task

The dataset for the correspondence task (CT) was built in the following manner. This procedure was performed for the training, validation and test set separately. First, a list was constructed of all available data in the respective dataset, containing the name of the video, together with the label, i.e. the word said in the video. Next this list of data was shuffled to ensure a random order of the different videos and labels. Given this list of data, for each example a coin flip was performed to decide whether this video should be used for a corresponding modality example, or a noncorresponding example. This coin flip ensures that a balanced dataset is created, with an approximate equal amount of corresponding and noncorresponding data.

If the modalities should correspond, the lip extractions were collected together with audio representations, collected as described above, and put in the dataset being constructed, together with a one-hot label encoding for corresponding. If the modalities should not correspond, a different video was collected, containing a different label from the first video. This was done to ensure that no noncorresponding examples are provided where the speaker is saying the same word, but with a different voice. Given these videos, the lip extractions are taken from the first video, whereafter the audio collected from the second video, which are also put in the dataset being constructed, together with the one-hot label encoding for noncorresponding. The one-hot label encodings used here consist of two values, where one is zero, and the other is one, depending on the target label.

3.1.4 Generating data for the classification task

The dataset generation for the classification task was done similar as the dataset creation for the CT. First a list of all available videos together with the respective label was made, after which this list was shuffled to ensure a random order of the examples. For each video the lip extractions were taken, together with the audio data. Next a one-hot encoding was made to encode which word was spoken in the video. This results in a vector of 500 separate numbers, where all numbers are zero, except the one representing the respective word. These lip extraction, audio and one-hot encoding were being placed in the dataset being constructed. This procedure was performed separately for the training, validation and test set.

3.1.5 Network architecture

The network architecture consists of 4 different modules. First, a deep architecture is made for encoding the audio. The second module is made for encoding the video frames. The third module combines the modality features and performs the classification for the CT. The last module also combines the different modality features and performs the classification between the different words. The general architecture of the network was largely inspired by [3].

Audio module

The audio module used was inspired by the audio module used in [3]. A diagram of the audio module as shown in figure 3.1.3. First a convolution is performed using a large 1D kernel of 220 neurons. This layer is meant to learn small nuances in the audio. After this convolution, average pooling was performed to reduce the dimensions resulting from the previous convolution. After this operation two ResNet [5] blocks are used, using 1D kernels instead of 2D. Finally the output of the last ResNet is followed by two more convolutions, after which the output is stretched out to a 1D array of values. All convolution layers in this modules use a ReLU activation function, without the use of a bias. Instead of a bias, batch normalisation is performed after each convolution layer, after the activation function [58].

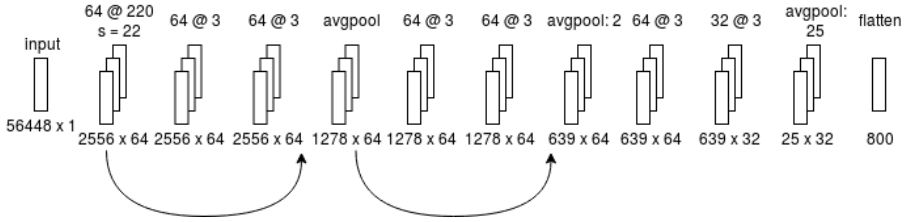


Figure 3.1.3: The auditory module. The first convolution layer uses a large kernel with a large stride (s), to reduce the large temporal dimension of the audio signal. Later ResNet blocks are used with average pooling. Finally three more convolution layers are used, after which the output is flattened.

Visual module

The video module is inspired by the video module as used in [3]. A diagram of the video module as used here is shown in figure 3.1.4. First relatively large spatial structures are encoded here, together with local temporal structures using a $5 \times 7 \times 7$ kernel, reflecting *time*, *height* and *width*. After this layer max pooling is used to reduce the image size. This is followed by three ResNet blocks, using 3x3 kernels and average pooling, followed by two more convolutions, after which the output is streched to a single vector. As in the audio module all convolutions are performed without a bias, and batch normalisation after each activation function. This encoder is built such that the spatial dimensions of the final convolution layer is 1×1. This means the output dimensions reflect only time and the number of kernels used in the last layer. These output dimensions are equivalent to the output dimensions of the audio encoder.

Combining the modality features

Throughout the models of this task two different methods of combining the modality features are being used. The first method is referred to as the L2-

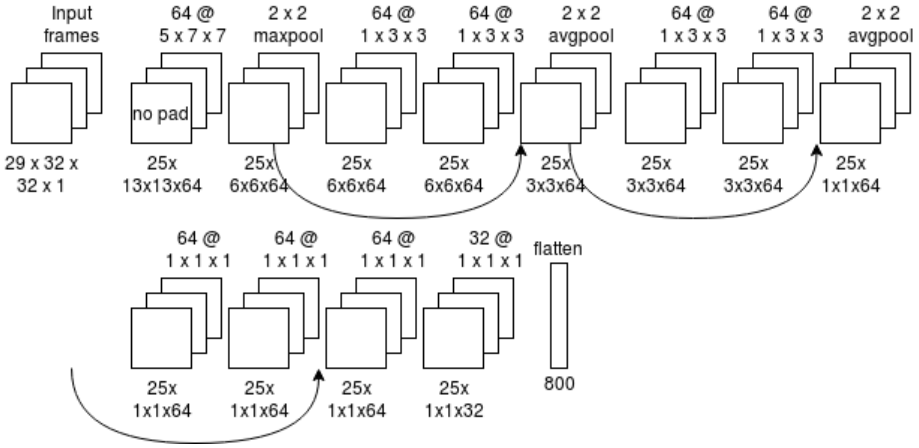


Figure 3.1.4: The visual module, encoding the video frames. After the input, each operation is specified on top of a block, where the resulting output is the block itself. All operations used zero padding, unless no padding was used, specified by "no pad". The arrows in the image depict the residual connections. The dimensions are shown below each block, indicating time × height × width × channels.

method, as shown in figure 3.1.5. Here the Euclidean distance is taken between the two different modality features. This makes use of the property of the two modality modules that they both output the same size modality vector. The output of this combination method is a single value representing the distance between the features. When modality feature normalisation is used in this setup, the shown video and audio input are first normalised using the L2 norm.

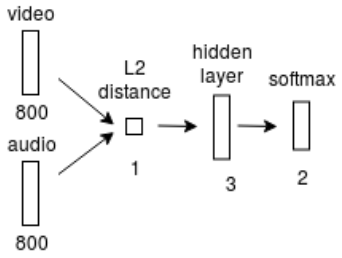


Figure 3.1.5: A diagram of the distance-based combination of modality features and the classifier for the correspondence task.

The other method used is referred to as the concatenation method, as shown in figure 3.1.6. Here the two different modality features are concatenated to each other.

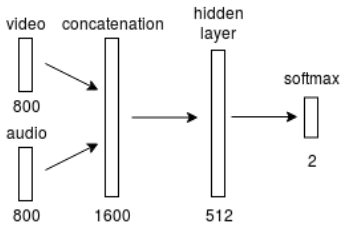


Figure 3.1.6: A diagram of the concatenation-based combination of modality features and the classifier for the correspondence task.

Correspondence task classifiers

Two different types of classifiers for the CT are being used for the models for this task. The first classifier receives the output of the L2 combination method. This single value is given to an MLP as shown in figure 3.1.5. This MLP uses one hidden layer of three neurons, using a ReLU activation function, followed by a softmax layer using two units.

The second type of classifier makes use of the concatenation method for combination of the modality features, as shown in figure 3.1.6. The resulting vector from this method is also given to an MLP, using a single hidden layer of 512 units, followed by a softmax layer using two units.

Word classifier

The classifier for word classification makes use of the concatenation method for combining the modality features. A diagram of this setup is shown in figure 3.1.7. The concatenated vector is given to an MLP using one hidden layer with 1024 neurons, using a ReLU activation function. This layer is followed by a softmax layer using 500 units.

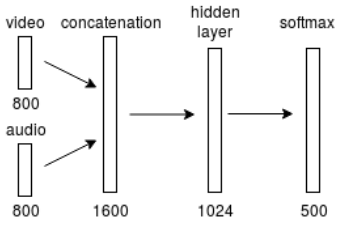


Figure 3.1.7: A diagram showing the feature combination and the classifier of the word classifier.

3.1.6 Optimisation

For the later mentioned experiments both correspondence training as well as classification training is done using the Adam optimisation technique [29]. For correspondence training a learning rate of 0.002 is used. For the training for word classification a learning rate of 0.0002 is used. The training is done using a batch size of 60 examples.

3.1.7 Evaluation

Evaluation of the models is done by assessing the fraction of answers from the models which are classified correctly. Each experimental setup is executed three times. The epoch at which a model performs best is determined by inspecting the performance on the validation set. The model from this epoch is evaluated by using the test set. Different experimental setups will be compared by comparing the distribution on the test set performances of the different runs of a model.

3.2 Sentence recognition

3.2.1 Dataset: lipreading sentences in the wild

This dataset *lipreading sentences in the wild* (LRS) was introduced in [59]. In contrast to the LRW dataset, the speakers here speak entire sentences, which are to be recognised. These sentences are of variable length, in terms of input, as well as output. The dataset consists out of three subsets. The first, called *pretrain* with 96318 videos. The second *train* with 45839 videos. Finally *test* and *val* with 1243 and 1082 videos respectively. Here only the *pretrain* has videos which are centered at the speaker's lip. Because of this only the *pretrain* set is used here. For the experiments in this thesis, the *pretrain* set is split up in a training set of 80% of the data, and a validation and test set of 10% each.

Figure 3.2.1: Some illustrative video frames from the LRS dataset.

3.2.2 Preprocessing

Preprocessing for sentence recognition

Preprocessing for sentence recognition is done similarly as with single word recognition. First crops of the lips of the speaker are extracted. The LRS dataset has different centre points of the lips as the LRW dataset, namely at $[79, 79]$ in the $160, 160$ frames. Here the surrounding patch with a height and width of $79, 79$ is taken, which is again downsized using bicubic interpolation to be 32×32. Next this patch is mapped to grey scale, to have only a channel depth of 1 value. The audio data is preprocessed in the same manner as with the LRW dataset.

Preprocessing for the correspondence task

For the correspondence task, examples are generated of equal length with respect to time. The examples start at the fourth video frame, and in total encodes 50 video frames. As a few examples are smaller than 50 video frames, these examples are not used for the correspondence task. Where the preprocessing for the correspondence task for the LRW dataset made sure not to make negative examples using the same said word, this is not enforced for the LRS dataset, as the LRS dataset is sentence-based, making it hard to enforce this for these examples.

3.2.3 Network architecture

Modality feature extraction

For feature extraction of the audio and visual data, the same modules are used as with single word classification, as shown in figures 3.1.3 and 3.1.4. Only here the output of the networks are not flattened, resulting in a $[T \times D]$ shaped output, where T denotes the temporal dimension, and D the feature depth. The other difference is that here no batch normalisation is used. Instead all convolution layers use a bias. Given memory limitations regarding long input videos, training was done on single examples.

Modality encoding

For the audio and visual modalities separate encoders are used, similar to the model from [7]. The temporal encoding in these stages were done by a single layer of 200 LSTM cells. The outputs over time of these cells are provided to the separate Bahdanau attention mechanisms. The final state of the LSTM cells are concatenated as a final encoding representation.

Decoding

The decoding state of the model consists of 400 LSTM cells, being initialised with concatenated final states of the encoding cells. During the training stage

the decoder receives the previous target token as input. During the inference stage the decoder is provided with the previously outputted token. As initial input of the first iteration the decoder receives a *start-of-sentence* token, both during training and classification. During character classification, the classifier also receives the weighted outputs of the encoder stage from the Bahdanau attention mechanisms for audio and video, as described in [7].

Character encoding

The decoder of the model was able to use the following characters:

1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ ' <>

Apart from only numbers and letters, the model was able to use a *space* and the apostrophe. The list of tokens finishes with <, the *start of sentence* token and >, the *end of sentence* token. Each token was represented by a 50 dimensional embedding, which was randomly initialised, after which they are updated throughout learning.

3.2.4 Correspondence training

Regarding the sequence to sequence (seq2seq) architecture, two different forms of correspondence learning are considered. As the seq2seq encoding stage is composed of feature extraction from the modalities, after which the temporal aspects of the sequences are encoded, correspondence learning could be done in either the feature extraction stage or the temporal encoding stage.

Feature extraction level correspondence learning

When correspondence learning is performed at the feature extraction stage, it is done in a similar way as for single word classification. The output from the visual and auditory modules are given to a single classifier for correspondence classification by which the feature extraction modules are trained. In this case, as with single word classification, both the distance-based as the concatenation-based correspondence task are used to pretrain the feature extractors.

Encoder level correspondence learning

Apart from using the output of the feature extraction modules directly to train a correspondence classifier, it is also possible to add a temporal encoding stage before classifying the modality embeddings for correspondence. When a setup is used where the modality features are first temporally encoded by one or more LSTM layers, the LSTM states can be used as a representation of the entire sequence of data. A major benefit of this technique is that it would be possible to use the CT for variable length inputs. I.e. in the case where the audio and video do not correspond, it is not a requirement that both signals have to be of equal length in time, as the LSTM layers encode variable length sequences to a

fixed size output. This output of the LSTM layers are then given to an MLP for correspondence classification. If this approach is used, this trained encoding stage is used solely for correspondence training. When training the full seq2seq architecture new encoders are trained for sentence recognition.

3.2.5 Optimisation

For the experiments described later regarding sentence recognition, the Adam optimisation technique was used, using a learning rate of 0.0002. Training was done on single examples due to memory limitations given long input videos.

3.2.6 Evaluation

The models described above for sentence recognition are evaluated based on the character error rate (CER), which is defined as shown in equation 3.1.

$$CER = (i + s + d)/n \tag{3.1}$$

Here i denoted the number of character insertions required to transform the output to the target text. Similarly d denotes the number of deletions required, s the number of substitutions required, and n the number of characters in the target text.

3.3 Hardware and software

The software built throughout this project was mainly based on tensorflow 1.13.1 [60], numpy [61] and opencv [57]. The experiments were performed on a machine using a gtx 1070 with 8 GB of memory. The datasets were stored on a samsung 970 EVO M2 SSD, to allow for a high bandwidth data stream during training. The machine used an intel i7-6700k as CPU, with 32 GB available memory.

Chapter 4

Experiments

4.1 Single word recognition

The first set of experiments will be using the LRW dataset for single word classification. Here the behaviour of the features resulting from training on the correspondence task (CT) will be assessed, as well as their usability for being used in a word classification setting.

4.1.1 Baseline experiments

To allow for comparison, baseline experiments will be performed where the model is trained directly to recognise the spoken words, where the audio and video modules are randomly initialised. Also the model will be trained using only the video module or the audio module, to verify that both modules are able to learn useful representations for speech recognition and lipreading. These setups will be trained each for 40 epochs.

4.1.2 Analysis of the correspondence task using the L2 method

Analysis of the L2 distance of modality features

To analyse the distances between the modality features, resulting from the distance-based CT, the network is first trained on the full LRW dataset, prepared for the CT as described earlier. After being trained for 20 epochs, the validation set is given to the trained network, where the L2 distances between the modality features are extracted. To explore the behaviour of this distance between the modality features, the distances for corresponding data are compared to the distances between features for non-corresponding data. It is hypothesised that the features for corresponding data are implicitly forced to express both modalities in the same latent space, given the results in [1]. It is expected that features for non-corresponding data result in a larger distance.

Classification performance in a transfer setting

Once the network is trained using the CT, the classification performance using the trained auditory and visual modules are analysed. The modules are kept constant, after being trained on the CT, though this time the resulting modality features are concatenated after which they are given to an MLP for word classification for the LRW dataset.

Classification performance in a pretraining setting

As a different technique, the network is trained end-to-end while using the pretrained modality modules. This time also the modality modules are being updated while training.

Comparison

The classification performance of the setups described will be compared with each other and the baseline performance based on their classification accuracy.

4.1.3 The influence of feature normalisation

In [1], before calculating the L2 distance of the modality features, they were first normalised. An experiment is performed to assess the effects of feature normalisation for the L2 combination method for correspondence learning. In principle the correspondence training method should work regardless of this normalisation operation, for which no detailed motivation was provided by the authors. The above mentioned experiments using the distance-based CT for transfer learning and pretraining are also performed here, although this time the features are first normalised before being given to the CT classifier. For word classification the features will be concatenated without normalisation.

4.1.4 Analysis of the correspondence task using the concatenation method

Classification performance in a transfer learning setting

Once the network is trained on the CT, the network is trained on the classification task, where the network has to differentiate between the 500 different target words in the dataset. Also this time the modality modules are kept constant while training for this task, using only the feature representations learned during the CT.

Classification performance in a pretraining setting

As a different approach, the network is trained end-to-end, with the modality modules being pretrained by the CT. In this setting also the modality modules are being updated during training.

Comparison

Finally the classification performance of these two approaches are being compared with each other, as well to the baseline performance of the network being trained end-to-end without any form of pretraining. This will be done using a unpaired Welch two sample t-test, using a decision threshold of $p = 0.05$.

4.2 Sentence recognition

4.2.1 Baseline experiments

First experiments will be performed to inspect the baseline performance of the sentence recognition setup and to validate the network architecture. During preliminary training no proper convergence of the models was achieved, including variations of the described network, hence no further experiments were performed regarding sentence-based AVSR.

Chapter 5

Results

This chapter will show the results obtained with the earlier mentioned experiments. First the training performances are discussed for each setup, together with the final performance on the test set. The training performances are shown as graphs using metrics based on the validation set, together with standard error bars. Finally a short overview is provided, together with comparisons of the test set performance of different setups also together with standard errors.

5.1 Single word recognition

5.1.1 End-to-end training results, without pretraining

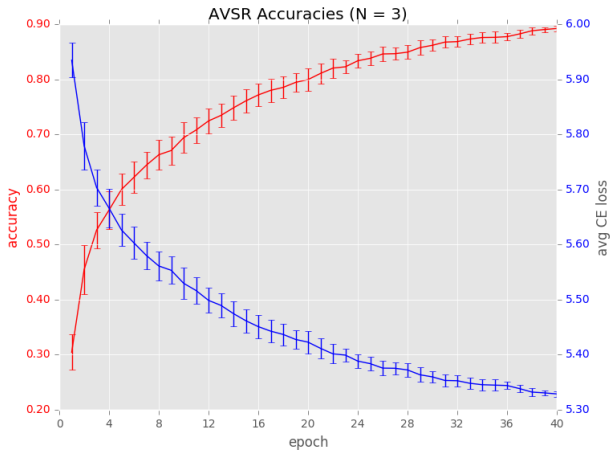


Figure 5.1.1: Training performance using audio and video without using any earlier training. This data was gathered using the validation set.

Audiovisual baseline results

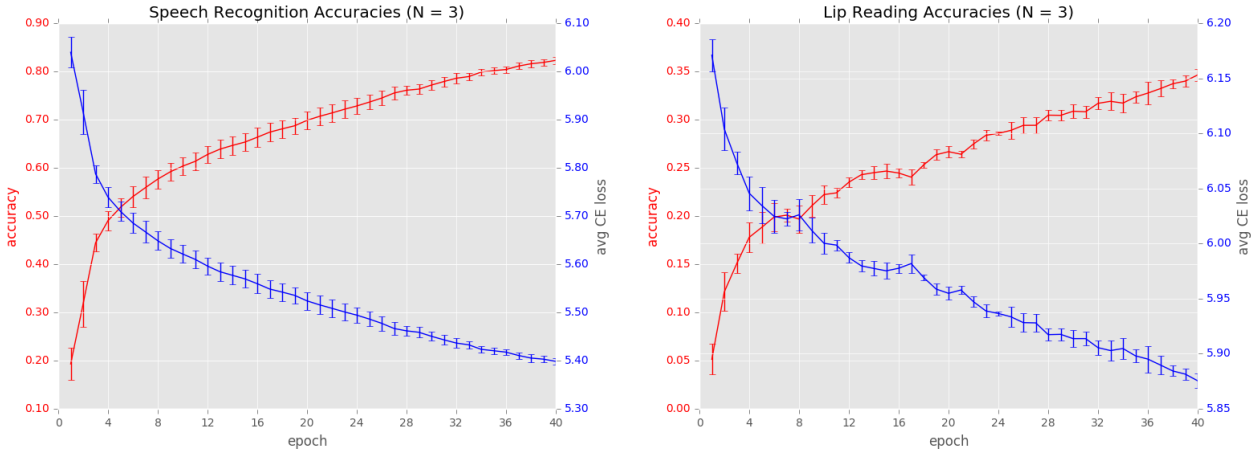
Considering the results on the validation set as shown in figure 5.1.1, the model's accuracy climbs steadily. The maximum performances were achieved after epoch 40, 40, 39 for the three different runs respectively. Here a validation performance of $89.24 \pm 0.47\%$ was achieved. When using these models on the test set, a test accuracy of $88.90 \pm 0.49\%$ was achieved. Experiments using this setup took roughly 12 hours to run.

Speech recognition results

The training performance of the model using only audio as input is shown in figure 5.1.2a. For the three models the maximum performance was obtained at epoch 40, 40 and 39 respectively having an average validation score of $82.29 \pm 0.74\%$, resulting in a test set performance of $82.01 \pm 0.72\%$. Experiments using this setup took roughly 7 hours to run.

Lip reading results

The training performance of the model using only video of the lips as input is shown in figure 5.1.2b. For the three models the maximum performance was obtained at epoch 40 for all models having an average validation score of $34.65 \pm 0.60\%$, resulting in a test set performance of $34.21 \pm 0.37\%$. Experiments using this setup took roughly 7 hours to run.



(a) (b)

Figure 5.1.2: Training performance, using the validation set. Figure (a) shows the training performance for solely speech recognition. Figure (b) shows the training performance of only lip reading.

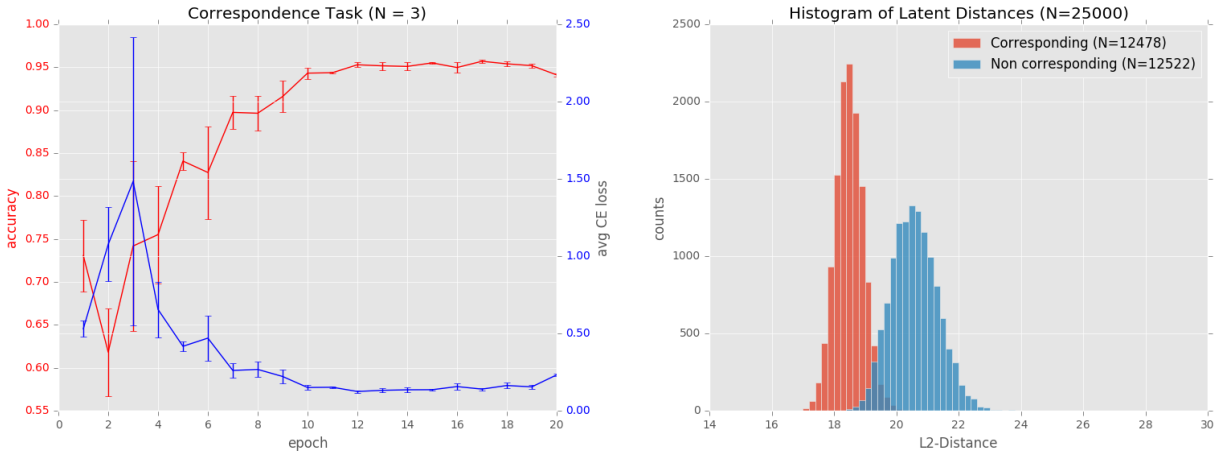
Table 5.1: Test results of the basic setup without any pretraining, showing the results of AVSR, speech recognition and lipreading.

Model	Test accuracy
AVSR	$88.90 \pm 0.49\%$
Audio only	$82.01 \pm 0.72\%$
Lips only	$34.21 \pm 0.37\%$

Comparing results of the basic setup

As can bee seen in table 5.1, the AVSR model outperformed both the speech recognition setup ($p(t = 6.42) = 0.0048$) as well as the lip reading setup ($p(t = 72.65) << 0.05$). Comparing the model with only audio with the model using only lips, the model using only audio outperforms the other ($p(t = 47.66) << 0.05$).

5.1.2 Analysis of the correspondence task using the L2 method without feature normalisation



(a) (b)

Figure 5.1.3: Figure (a) shows the training performance on the validation set of the distance-based correspondence task, without feature normalisation. Figure (b) shows histograms of the L2 distance between the modality features for corresponding and noncorresponding data from a single model.

Correspondence learning

The performance for the L2 distance-based correspondence task (DCT), using the L2 method, without feature normalisation is shown in figure 5.1.3a. Here the graph shows a rapid increase in accuracy, with a slightly high variance. After this, the accuracy seems to flatten out after epoch 12. The highest validation scores were achieved after epoch 16, 13 and 17 for the three models respectively. Here the models achieved an average validation score of $96.16 \pm 0.10\%$ and a test set score of $96.14 \pm 0.12\%$. Experiments using this setup took roughly 6 hours to run.

Analysis of the feature distances

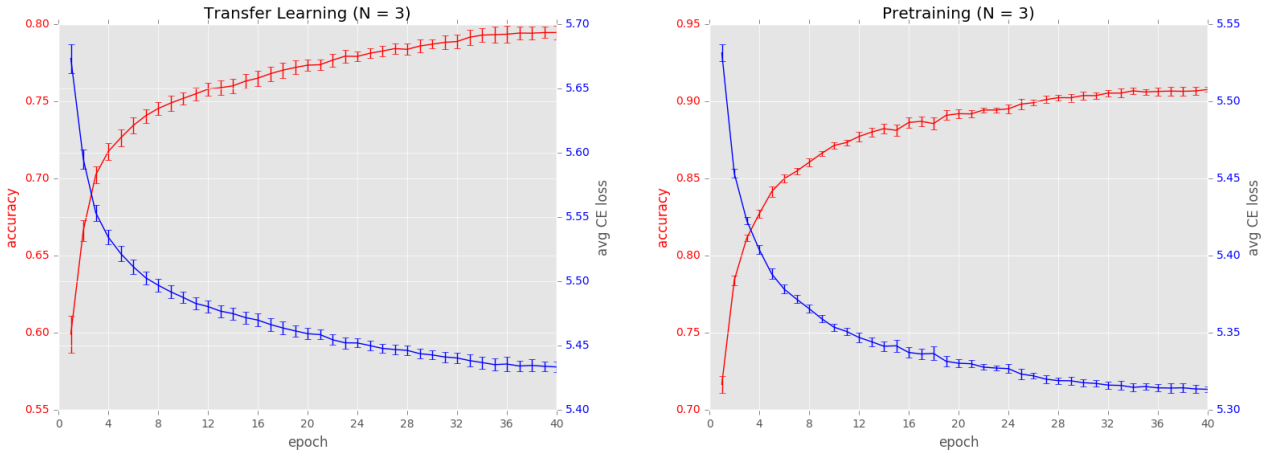
Considering the network, after being trained, the L2 distances of the latent space features from both modalities are compared. A histogram of L2 distances for corresponding and non corresponding validation examples are shown in figure 5.1.3b. This is the a depiction given the L2 distances from a single model. If the distributions from the three separate models are considered results are obtained as shown in table 5.2. These results indicate that corresponding modality features are not being mapped to the same latent space. In that case the means for corresponding data would be near zero, and non corresponding data should result in much larger distances. Here it can be seen that this is not the case, since in two from the three models the distances for noncorresponding examples are smaller than those of corresponding ones.

Table 5.2: A table indicating the normal distributions of L2 distances from the distance-based correspondence task. Here μ indicates the mean, where σ indicates the standard deviation. $+$ indicates corresponding data, where $-$ indicates non corresponding data.

Model	μ_+	σ_+	μ_-	σ_-
1	18.50	0.45	20.59	0.75
2	27.23	0.27	25.98	0.46
3	23.98	0.28	22.85	0.43

Transfer learning

When using the modality modules, being trained by the correspondence task, training only the top layer for word classification, results are obtained as shown in figure 5.1.4a. These results show a steady increase of performance during the entire runs. The highest validation score for transfer learning were achieved after epoch 38, 36 and 39 for the three models respectively. Here an average validation score of $79.55 \pm 0.40\%$ was achieved. Using these models on the test set, a test performance of $79.04 \pm 0.41\%$ is achieved. Experiments using this setup took roughly 4 hours to run.



(a) (b)

Figure 5.1.4: The training performance on the validation set. Figure (a) shows the performance using transfer learning. Figure (b) shows the performance using pretraining.

Pretraining

When using the modality modules, as trained by the distance-based correspondence task, followed by end-to-end training, the results as shown in figure 5.1.4b are obtained. For the pretraining setup a rapid increase of accuracy can be seen at the beginning, whereafter the performance gradually increases. The highest validation scores were achieved after epoch 34, 37 and 40 for the three models respectively. Here a validation performance of $90.88 \pm 0.20\%$ was achieved, with a performance of $90.60 \pm 0.15\%$ on the test set. Experiments using this setup took roughly 12 hours to run.

Model comparison

Table 5.3: The test set results of the base model, together with the results of the pretraining and transfer learning setup, using the distance-based correspondence task.

Model	Test accuracy
Baseline	$88.90 \pm 0.49\%$
Pretraining	$90.60 \pm 0.15\%$
Transfer	$79.04 \pm 0.41\%$

The final performances of pretraining and transfer learning are shown in table 5.3, together with the earlier mentioned performance of the model without

earlier training. Comparing the results of pretraining with the baseline performance, pretraining seems to achieve a higher score on average, although a t-test finds no significant difference ($p(t = 2.71) = 0.094$). Comparing the transfer learning setting to the baseline, it can be seen that transfer learning performs worse than the baseline ($p(t = 12.56) << 0.05$), although the performance is still high compared to random chance.

5.1.3 Analysis of the correspondence task using the L2 method with L2 feature normalisation

When training the network with the L2 method, while the modality features are normalised first by the L2 norm, the model did not converge unfortunately. The highest validation accuracy achieved was $49.91 \pm 0.00\%$, being no different than answering based on random chance. The pretraining and transfer learning setups based on this technique were dropped from the comparison.

5.1.4 Analysis of the correspondence task using concatenation

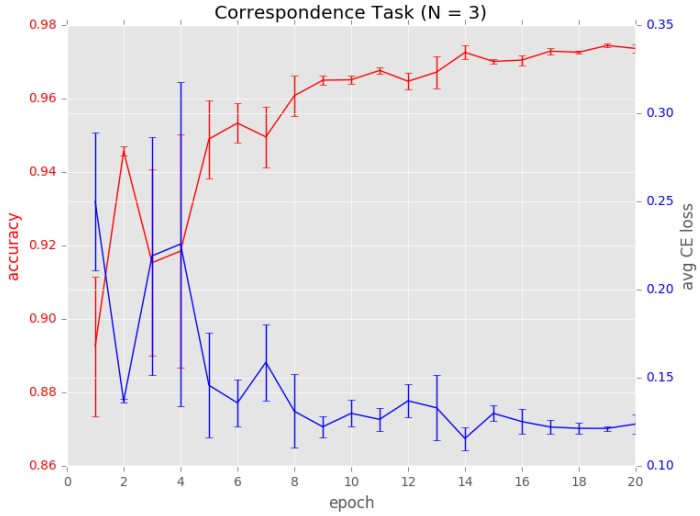
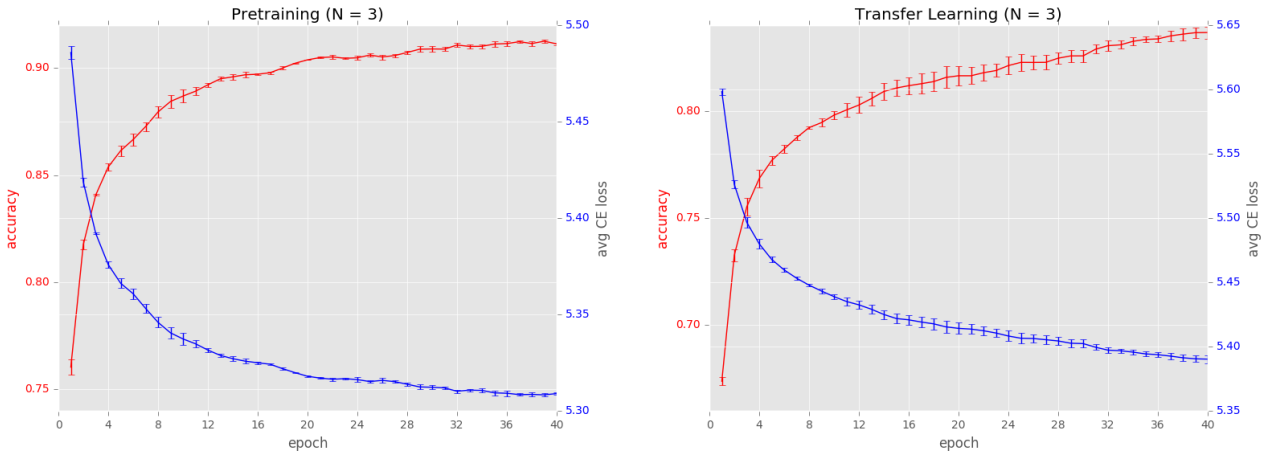


Figure 5.1.5: Training performance of the correspondence task using concatenation. This data was gathered using the validation set.

The training performance of the correspondence task using concatenation is shown in figure 5.1.5. Also here the graph shows a rapid performance increase in the beginning with a high variance between the models. After epoch 9 the variance seems to decrease, while the average performance still slightly increases. The best performances of the models were after epoch 14, 19 and 16 respectively,

with an average validation accuracy of $97.52 \pm 0.07\%$, and a test set accuracy of $97.49 \pm 0.04\%$. Experiments using this setup took roughly 6 hours to run.



(a) (b)

Figure 5.1.6: Training performances on the validation set. Figure (a) shows the performance using the pretraining setup. Figure (b) shows the performances using the transfer learning setup.

Pretraining

The training performance of the model in a pretraining setting using the concatenation-based correspondence task is shown in figure 5.1.6a. Also here a rapid increase in the beginning can be observed, followed by a steady graduate increase. The best performance for the three respective models were achieved after epoch 39, 39 and 37, with an average validation accuracy of $91.30 \pm 0.08\%$. Using the test set, the three models achieved an average accuracy of $91.05 \pm 0.04\%$. Experiments using this setup took roughly 12 hours to run.

Transfer learning

The training performance of the model in a transfer learning setting using the concatenation-based correspondence task is shown in figure 5.1.6b. Here the graph shows a steady increase of performance on the validation set. It is noted that the model still seems to be learning after epoch 40, although comparison will be done on the best achieved results. The best performances on validation data were achieved after epoch 40, 39 and 40 for the three respective models, with an average validation score of $83.73 \pm 0.25\%$. Using these models, a test accuracy of $83.25 \pm 0.28\%$ was achieved. Experiments using this setup took roughly 4 hours to run.

Model comparison

Table 5.4: The test set results of the base model, together with the results of the pretraining and transfer learning setup, using the concatenation-based correspondence task.

Model	Test accuracy
Baseline	$88.90 \pm 0.49\%$
Pretraining	$91.05 \pm 0.04\%$
Transfer	$83.25 \pm 0.28\%$

A summary of the test results using the concatenation-based correspondence task is shown in table 5.4. When comparing the transfer learning setup, this model also performs worse than the baseline setup ($p(t = 8.2) = 0.003$). When comparing the pretraining setup to the baseline performance, pretraining seems to perform slightly better than baseline, though a t-test finds no significant difference ($p(t = 3.58) = 0.069$).

5.1.5 Comparison of the models

Table 5.5: Summary of the test set performances for the different setups. Here DCT denotes the usage of the distance-based correspondence task. CCT denotes the usage of the concatenation-based correspondence task.

Model	Accuracy
Baseline	$88.90 \pm 0.49\%$
Audio only	$82.01 \pm 0.72\%$
Lips only	$34.21 \pm 0.37\%$
DCT pretrain	$90.60 \pm 0.15\%$
DCT transfer	$79.04 \pm 0.41\%$
CCT pretrain	$91.05 \pm 0.04\%$
CCT transfer	$83.25 \pm 0.28\%$

Finally the performances of the different variations of the correspondence task are compared to each other. In comparing the results between pretraining using the concatenation and distance-based correspondence task, no significant difference is found ($p(t = 2.34) = 0.13$). Next the results between the concatenation and distance-based correspondence task, when applied in a transfer learning setting, are compared. Here the concatenation-based correspondence task yields a higher performance than the distance-based method ($p(t = 6.88) = 0.004$).

5.2 Sentence recognition

Unfortunately, the basic architecture designed for sentence recognition did not converge well enough to be applicable for speech recognition. The model did seem to learn words that occur in the dataset, although when comparing the target sentences with the output sentences, no words were recognised correctly. Some example output of the trained model is provided below. Also the model failed to produce the required *end of sentence* token, resulting in output continuing until the predetermined cutoff point of 1000 characters was achieved. Including the redundant characters an average character error rate of 10.90 was achieved. When only considering the first N outputs, where N denotes the number of characters in the target sentence, a character error rate of 0.9068 is measured, indicating that at least 90% of the character outputs are classified incorrectly. Because of these results no further experiments were performed regarding sentence recognition. Regarding time, a single epoch over the data using this setup took approximately 12 hours to run.

5.2.1 Output inspection

To illustrate the output of the model, some examples are provided regarding the target output, and the output of the model. For the target output > denotes the *end of sentence* token. For the output of the model the final part surrounded by brackets is repeated indefinitely at the end. This indicates that the model failed to output an *end of sentence* token.

Example 1:

Target:
these days when you're cooking chips at home the traditional chip pan often stays on the shelf in favour of a baking tray and a bag of frozen oven>
Output:
they were the station of the contact to the state (that they can see the contact to the state)

Example 2:

Target:
the factory they use a slicing machine called a hydro cutter which involves firing a potato down a pipe through what they call a knife block that slices the potato into chip>
Output:
they can see the control (and they can see the control)

Example 3:

Target:
that's that done now what could be better to fire my potatoes through the slicer

than a spud>
Output:
they can see the company (that they can see the company)

Example 4:

Target:
and hopefully chip shaped potatoes come through>
Output:
they can see the company (that is the company)

Example 5:

Target:
so if you have it in a restaurant and it's delicious you go back next week you want it to be the same don't you that's right the>
Output:
they can see the company (that is the country)

Chapter 6

Discussion

Given the results as shown in the previous chapter, first the research questions will be addressed. Given the fact that the sentence recognition models did not converge, the discussion will be limited to only single word recognition. Hereafter the results of the sentence-based architecture are discussed. Finally potential future work regarding the usage of the correspondence task in deep learning is discussed.

6.1 Answers to the research questions

Does the correspondence task yield feature extraction modules that are useful for AVSR?

Given the results that using the correspondence task, with the concatenation method, in a transfer learning setting was able to achieve a test set accuracy of $83.25 \pm 0.28\%$, it is concluded that the features obtained by using the correspondence task are useful for AVSR in a single word recognition context. Also, when using the correspondence task within a pretraining setting resulted in a much faster convergence of the classification model.

Do the performances differ between using concatenated modality features or using the distance between the features in the correspondence task?

When comparing the performances of the two setups in a transfer learning setting, the results of the word classifiers do differ. Here the distance-based correspondence task performs worse than the concatenation-based correspondence task. When comparing the performances in a pretraining setting, no significant difference between the two setups was found. Although, when comparing the quality of the modality features for classification it can be concluded that there is a difference, given the results from the transfer learning setting. Concatenated modality features result in higher performances in a transfer learning setting.

Do the performances differ between using normalised or raw modality features in the distance-based correspondence task?

Using normalised modality features the model was unable to learn the correspondence task. Using no feature normalisation did yield convergence. Using the resulting features from this setup resulted in relatively high performances. It is merely concluded that both setups behaved differently during training.

Does the distance-based correspondence task learn to map both modality features to the same latent space?

Based on the results described comparing the distances between corresponding and noncorresponding features, it is concluded that the distance-based correspondence task does not learn to map both modality features to the same latent space. The models even seem to learn a higher distance for corresponding examples than for noncorresponding examples, which conflicts with the results obtained by [1], where high performances were obtained by a cross modal retrieval task. This model performs retrieval by searching an example from a different modality with the lowest distance in latent space to the search query. If the model learns to assign lower distances to noncorresponding examples, the mentioned cross modal retrieval model would return noncorresponding examples.

6.2 The sentence-based model

Given the results from the sentence models, it can be seen that the model does seem to learn basic word usage for a limited set of words. During inference the model fails to output an *end of sentence* token, which resulted in indefinite repeating sequences. Although the model did output text, the output did not seem to be related to what was actually said in the examples. This suggests that the main difficulty seems to be the interpretation of the encoder state. In [7] the authors note this difficulty, which they approached by using *curriculum learning*. Here the model is first trained on short sentences. Once the model seems to perform well, longer sentences are introduced. This technique was not used in this work, due to memory constraints. Given the long training time of the model, which was around 12 hours to show all training data once, the number of exploratory experiments were limited.

6.3 Future work

6.3.1 Using multiple languages

The correspondence task as used here makes no assumptions about the domain. Given enough data, the correspondence task should be able to learn quite general representations. For example, if an AVSR system would be trained to recognise English spoken speech, the features learned for this task by the auditory

and visual feature extraction modules are probably to be specialised to extract features for recognising English speech. If the same feature extraction modules were to be used in a transfer learning setting for an AVSR system used to recognise Chinese, it is quite likely that the feature extraction modules would return less useful representations, as when they were to be used to recognise English speech. The correspondence task would merely utilise the information whether the modality signals were caused by the same event or not. So if trained with large collections of audio visual data of speech from a large variety of languages in this case, it is possible that the correspondence task would result in broader applicable feature extraction modules, usable to recognise speech in multiple languages.

6.3.2 Using different modalities

Apart from using the correspondence task for AVSR, other applications can be imagined where other modalities are used. Examples might be where one modality is an image of a robotic arm, where the other modality represents the joint states. The correspondence task might be usable here to pretrain a CNN, processing a video data stream of the arm, to be used later for e.g. visual servoing. In general when data from multiple modalities are available of events corresponding in time, the correspondence task might be a usable pretraining technique.

6.3.3 Learning correspondence from variable length sequence data

While the sequence to sequence model failed to converge properly during the experiments in this work, it still remains well possible to use the correspondence task in such a sequence-based setup. Such sequence-based modelling could be possible using multiple different modalities as discussed above. For future work on the sequence to sequence model, it is suggested to use curriculum learning, as proposed in [7].

Bibliography

[1] R. Arandjelović and A. Zisserman, "Objects that sound," in *European Conference on Computer Vision*, 2018.

[2] R. Arandjelovic and A. Zisserman, "Look, listen and learn.," in *ICCV*, pp. 609–617, IEEE Computer Society, 2017.

[3] S. Petridis, T. Stafylakis, P. Ma, F. Cai, G. Tzimiropoulos, and M. Pantic, "End-to-end audiovisual speech recognition," pp. 6548–6552, 2018.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

[6] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, (Cambridge, MA, USA), pp. 3104–3112, MIT Press, 2014.

[7] J. S. Chung and A. Zisserman, "Lip reading in the wild," in *Asian Conference on Computer Vision*, 2016.

[8] Y. Aytar, C. Vondrick, and A. Torralba, "Soundnet: Learning sound representations from unlabeled video," in *Advances in Neural Information Processing Systems*, 2016.

[9] R. Zhang, P. Isola, and A. A. Efros, "Split-Brain Autoencoders: Unsupervised Learning by Cross-Channel Prediction," *arXiv e-prints*, p. arXiv:1611.09842, Nov 2016.

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1," ch. Learning Internal Representations by Error Propagation, pp. 318–362, Cambridge, MA, USA: MIT Press, 1986.

[11] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neurocomputing*, pp. 227–236, Springer, 1990.

[12] J. S. Bridle, "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters," in *Advances in Neural Information Processing Systems 2* (D. S. Touretzky, ed.), pp. 211–217, Morgan-Kaufmann, 1990.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[14] K. J. Piczak, "Esc: Dataset for environmental sound classification," in *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, (New York, NY, USA), pp. 1015–1018, ACM, 2015.

[15] A. Mesaros, T. Heittola, and T. Virtanen, "A multi-device dataset for urban acoustic scene classification," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, pp. 9–13, November 2018.

[16] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Proc. IEEE ICASSP 2017*, (New Orleans, LA), 2017.

[17] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, (Doha, Qatar), pp. 103–111, Association for Computational Linguistics, Oct. 2014.

[18] J. S. Chung and A. Zisserman, "Lip reading in the wild," in *Asian Conference on Computer Vision*, 2016.

[19] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014.

[20] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *ICASSP*, 2016.

[21] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[22] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, pp. 65–386, 1958.

[23] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, pp. 273–297, Sept. 1995.

[24] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 2672–2680, Curran Associates, Inc., 2014.

[25] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *ArXiv*, vol. abs/1411.1784, 2014.

[26] B. Zhao, J. Feng, X. Wu, and S. Yan, "A survey on deep learning-based fine-grained object classification and semantic segmentation," *International Journal of Automation and Computing*, vol. 14, pp. 119–135, Apr 2017.

[27] J. Kiefer and J. Wolfowitz, "Stochastic estimation of the maximum of a regression function," *Ann. Math. Statist.*, vol. 23, pp. 462–466, 09 1952.

[28] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, pp. 400–407, 09 1951.

[29] "Adam: A method for stochastic optimization," 2014. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[30] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016.

[31] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, pp. 2278–2324, 1998.

[32] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," pp. 448–456, 2015.

[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, Nov. 1997.

[34] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, pp. 1527–1554, July 2006.

[35] L. Y. Pratt, "Discriminability-based transfer between neural networks," in *Advances in Neural Information Processing Systems 5* (S. J. Hanson, J. D. Cowan, and C. L. Giles, eds.), pp. 204–211, Morgan-Kaufmann, 1993.

[36] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, *A Survey on Deep Transfer Learning: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4–7, 2018, Proceedings, Part III*, pp. 270–279. 10 2018.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[38] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning Spatiotemporal Features with 3D Convolutional Networks," *arXiv e-prints*, p. arXiv:1412.0767, Dec 2014.

[39] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 221–231, Jan 2013.

[40] T. Shi, Y. Keneshloo, N. Ramakrishnan, and C. K. Reddy, "Neural abstractive text summarization with sequence-to-sequence models," *CoRR*, vol. abs/1812.02303, 2018.

[41] A. Owens and A. A. Efros, "Audio-Visual Scene Analysis with Self-Supervised Multisensory Features," *arXiv e-prints*, p. arXiv:1804.03641, Apr 2018.

[42] J. Jiang, A. Alwan, P. A. Keating, E. T. Auer, and L. E. Bernstein, "On the relationship between face movements, tongue movements, and speech acoustics," *EURASIP Journal on Advances in Signal Processing*, vol. 2002, p. 506945, Nov 2002.

[43] Q. Summerfield, "Lipreading and audio-visual speech perception," *Philosophical Transactions: Biological Sciences*, vol. 335, no. 1273, pp. 71–78, 1992.

[44] E. C. Cherry, "Some experiments on the recognition of speech, with one and with two ears," *Journal of the Acoustical Society of America*, vol. 25, pp. 975–979, 1953.

[45] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan, "Speech recognition using deep neural networks: A systematic review," *IEEE Access*, vol. 7, pp. 19143–19165, 2019.

[46] Y. M. Assael, B. Shillingford, S. Whiteson, and N. de Freitas, "LipNet: End-to-End Sentence-level Lipreading," *arXiv e-prints*, p. arXiv:1611.01599, Nov 2016.

[47] E. D. Petajan, *Automatic Lipreading to Enhance Speech Recognition (Speech Reading)*. PhD thesis, Champaign, IL, USA, 1984. AAI8502266.

[48] G. Potamianos, C. Neti, G. Gravier, A. Garg, and A. W. Senior, "Recent advances in the automatic recognition of audiovisual speech," *Proceedings of the IEEE*, vol. 91, pp. 1306–1326, Sep. 2003.

[49] A. Adjoudani and C. Benoît, *On the Integration of Auditory and Visual Parameters in an HMM-based ASR*, pp. 461–471. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996.

[50] Tsuhan Chen, "Audiovisual speech processing," *IEEE Signal Processing Magazine*, vol. 18, pp. 9–21, Jan 2001.

[51] P. Teissier, J. Robert-Ribes, J. . Schwartz, and A. Guerin-Dugue, "Comparing models for audiovisual fusion in a noisy-vowel recognition task," *IEEE Transactions on Speech and Audio Processing*, vol. 7, pp. 629–642, Nov 1999.

[52] C. Neti, G. Potamianos, J. Luettin, I. Matthews, H. Glotin, D. Vergyri, J. Sison, and A. Mashari, "Audio visual speech recognition," 01 2000.

[53] S. Dupont and J. Luettin, "Audio-visual speech modeling for continuous speech recognition," *Multimedia, IEEE Transactions on*, vol. 2, pp. 141 – 151, 10 2000.

[54] S. M. Chu and T. S. Huang, "Bimodal speech recognition using coupled hidden markov models," in *INTERSPEECH*, 2000.

[55] M. Heckmann, F. Berthommier, and K. Kroschel, "Noise adaptive stream weighting in audio-visual speech recognition," *EURASIP journal on advances in signal processing*, 07 2002.

[56] M. Gordan, C. Kotropoulos, and I. Pitas, "A support vector machine-based dynamic network for visual speech recognition applications," *EURASIP Journal on Advances in Signal Processing*, vol. 2002, p. 427615, Nov 2002.

[57] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[58] G. Chen, P. Chen, Y. Shi, C.-Y. Hsieh, B. Liao, and S. Zhang, "Rethinking the Usage of Batch Normalization and Dropout in the Training of Deep Neural Networks," *arXiv e-prints*, p. arXiv:1905.05928, May 2019.

[59] T. Afouras, J. Son Chung, A. Senior, O. Vinyals, and A. Zisserman, "Deep Audio-Visual Speech Recognition," p. arXiv:1809.02108, Sep 2018.

[60] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[61] T. E. Oliphant, *A guide to NumPy*, vol. 1. Trelgol Publishing USA, 2006.

Appendices

Appendix A

Words in the LRW dataset

About; Absolutely; Abuse; Access; According; Accused; Across; Action; Actually; Affairs; Affected; Africa; After; Afternoon; Again; Against; Agree; Agreement; Ahead; Allegations; Allow; Allowed; Almost; Already; Always; America; American; Among; Amount; Announced; Another; Answer; Anything; Areas; Around; Arrested; Asked; Asking; Attack; Attacks; Authorities; Banks; Because; Become; Before; Behind; Being; Believe; Benefit; Benefits; Better; Between; Biggest; Billion; Black; Border; Bring; Britain; British; Brought; Budget; Build; Building; Business; Businesses; Called; Cameron; Campaign; Cancer; Cannot; Capital; Cases; Central; Certainly; Challenge; Chance; Change; Changes; Charge; Charges; Chief; Child; Children; China; Claims; Clear; Close; Cloud; Comes; Coming; Community; Companies; Company; Concerns; Conference; Conflict; Conservative; Continue; Control; Could; Council; Countries; Country; Couple; Course; Court; Crime; Crisis; Current; Customers; David; Death; Debate; Decided; Decision; Deficit; Degrees; Described; Despite; Details; Difference; Different; Difficult; Doing; During; Early; Eastern; Economic; Economy; Editor; Education; Election; Emergency; Energy; England; Enough; Europe; European; Evening; Events; Every; Everybody; Everyone; Everything; Evidence; Exactly; Example; Expect; Expected; Extra; Facing; Families; Family; Fight; Fighting; Figures; Final; Financial; First; Focus; Following; Football; Force; Forces; Foreign; Former; Forward; Found; France; French; Friday; Front; Further; Future; Games; General; George; Germany; Getting; Given; Giving; Global; Going; Government; Great; Greece; Ground; Group; Growing; Growth; Guilty; Happen; Happened; Happening; Having; Health; Heard; Heart; Heavy; Higher; History; Homes; Hospital; Hours; House; Housing; Human; Hundreds; Immigration; Impact; Important; Increase; Independent; Industry; Inflation; Information; Inquiry; Inside; Interest; Investment; Involved; Ireland; Islamic; Issue; Issues; Itself; James; Judge; Justice; Killed; Known; Labour; Large; Later; Latest; Leader; Leaders; Leadership; Least; Leave; Legal; Level; Levels; Likely; Little; Lives; Living; Local; London; Longer; Looking; Major; Majority; Makes; Making; Manchester; Market; Massive; Matter; Maybe; Means; Measures; Media; Medical; Meeting; Member; Members; Message; Middle; Might;

Migrants; Military; Million; Millions; Minister; Ministers; Minutes; Missing; Moment; Money; Month; Months; Morning; Moving; Murder; National; Needs; Never; Night; North; Northern; Nothing; Number; Numbers; Obama; Office; Officers; Officials; Often; Operation; Opposition; Order; Other; Others; Outside; Parents; Parliament; Parties; Parts; Party; Patients; Paying; People; Perhaps; Period; Person; Personal; Phone; Place; Places; Plans; Point; Police; Policy; Political; Politicians; Politics; Position; Possible; Potential; Power; Powers; President; Press; Pressure; Pretty; Price; Prices; Prime; Prison; Private; Probably; Problem; Problems; Process; Protect; Provide; Public; Question; Questions; Quite; Rates; Rather; Really; Reason; Recent; Record; Referendum; Remember; Report; Reports; Response; Result; Return; Right; Rights; Rules; Running; Russia; Russian; Saying; School; Schools; Scotland; Scottish; Second; Secretary; Sector; Security; Seems; Senior; Sense; Series; Serious; Service; Services; Seven; Several; Short; Should; Sides; Significant; Simply; Since; Single; Situation; Small; Social; Society; Someone; Something; South; Southern; Speaking; Special; Speech; Spend; Spending; Spent; Staff; Stage; Stand; Start; Started; State; Statement; States; Still; Story; Street; Strong; Sunday; Sunshine; Support; Syria; Syrian; System; Taken; Taking; Talking; Talks; Temperatures; Terms; Their; Themselves; There; These; Thing; Things; Think; Third; Those; Thought; Thousands; Threat; Three; Through; Times; Today; Together; Tomorrow; Tonight; Towards; Trade; Trial; Trust; Trying; Under; Understand; Union; United; Until; Using; Victims; Violence; Voters; Waiting; Wales; Wanted; Wants; Warning; Watching; Water; Weapons; Weather; Weekend; Weeks; Welcome; Welfare; Western; Westminster; Where; Whether; Which; While; Whole; Winds; Within; Without; Women; Words; Workers; Working; World; Worst; Would; Wrong; Years; Yesterday; Young;