Karlsruhe Institute of Technology

Online Neural Network-based Language Identification

Master's Thesis of

Daniel H. Draper

at the Department of Informatics
Institute for Anthropomatics and Robotics

Reviewer:	Prof. Dr. Alexander Waibel
Second reviewer:	
Advisor:	M.Sc. Markus Müller
Second advisor:	Dr. Sebastian Stüker

12. December 2016 – 11. May 2017

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, 12th of May, 2017

. . . . . .

(Daniel H. Draper)

Abstract

Zusammenfassung

Contents

Abstract i

Zusammenfassung iii

1. Introduction 1
- 1.1. Motivation . . . 1
- 1.2. Overview . . . 2

2. Fundamentals 3
- 2.1. Janus Recognition Toolkit (jrtk) . . . 3
- 2.2. Neural Networks . . . 3
  - 2.2.1. General Setup . . . 3
  - 2.2.2. Artificial Neuron . . . 4
  - 2.2.3. Network Types . . . 4
  - 2.2.4. Learning . . . 5
- 2.3. Related Work . . . 6

3. Experimental Setup 7
- 3.1. Euronews 2014 . . . 7
- 3.2. Lecture Data . . . 7
- 3.3. European Parliament . . . 9
- 3.4. Feature Preprocessing . . . 9
  - 3.4.1. Feature Access . . . 9
  - 3.4.2. Feature Description . . . 9
  - 3.4.3. ASR BNF network . . . 9

4. LID Network Structure and Results 11
- 4.1. Basic Setup . . . 11
- 4.2. Improving Network Layout . . . 12
- 4.3. Results . . . 12

5. Smoothing and Evaluation 13
- 5.1. Basic Test Filter . . . 13
- 5.2. Advanced Test Filter . . . 14
- 5.3. Variance Test Filter . . . 15
- 5.4. Two-Language setup . . . 15

6. Conclusion 17

A. Appendix **21**

List of Figures

List of Tables

3.1. The Euronews corpus speaker breakdown with total utterances length . 8
3.2. The Euronews corpus breakdown into the three data sets. 8
3.3. The Lecture Data corpus speaker breakdown with total utterances length. 8

1. Introduction

Language Identification (LID) describes the classification task of differentiating between spoken speech in different languages and being able to correctly classify which speech-segments consists of which language. Neural Networks refer to Artificial Neural Network's, a Machine Learning approach to classification tasks employed greatly throughout all sciences and especially in computer science and tasks concerned with the processing of spoken speech. This thesis tries to find a low-latency, fast, or "online", approach to Language Identification using the classification method of neural networks.

The work done in this thesis uses the Janus Recognition Toolkit (jrtk)[1], an Automatic Speech Recognition toolkit developed in joint cooperation by the KIT and CMU. The jrtk offers a tcl/tk[2] script-based environment for the development of Automatic Speech Recognition systems, therefore source code in this thesis will consist of tcl/tk scripts with (some) janus-specific commands. The jrtk and tcl/tk are further explained in sec. 2.1.

1.1. Motivation

Automatic Speech Recognition (ASR) is used in many applications and devices today, especially in the rise of handheld mobile devices like smart-phones and tablets. It has progressed quickly in the last five years and has found commercial success. Famous examples include Google[3]'s "Ok, Google" and Apple[4]'s Siri. Which both include voice search[FHBM08] and a form of voice control, that even is extensible in the case of Google and Android e.g[bAO14]. Many other applications have emerged, including spoken language translation[5], especially relevant for this thesis in the realm of Lecture Translation[MNN+16] .

The task of Language Identification can be applied in all of those fields, as Automatic Speech Recognition is mostly trained on one language and therefore requires a totally different setup of classifiers per language, making a manual change of language previous to recognition necessary. Robust and low-latency language identification would eliminate the need for this.

LID would be especially applicable in the realm Spoken language translation, as used for example in the European Parliament where already components of ASR and Machine Translation are employed and are being actively developed in the TC-STAR inititative[6], e.g as in[VMH+05], and LID would further be able to automate these translation tasks.

[1] Janus Recognition Toolkit: `http://isl.anthropomatik.kit.edu/cmu-kit/english/1406.php`
[2] Tcl/tk: `https://www.tcl.tk/`
[3] Google: `www.google.com`
[4] Apple: `www.apple.com`
[5] IWSLT: `iwslt.org`
[6] TC-STAR: `tcstar.org`

This thesis will focus mostly on the KIT's lecture Translator[7] as the system trained was implemented for it. We believe our results are general enough to be transferable to other applications with small implementation-specific changes.

1.2. Overview

The following chapter gives an introductory view of the applications of Language Identification, and introduces the tasks this thesis tries to solve. Afterwards we give preliminary theoretical explanations and definitions, including an introduction to Neural Networks in Sec. 2.2. The next chapter describes the experimental setup used in this thesis, including the data corpusses and feature preprocessing done on raw audio files.

Chapters 4 and 5 describe our results that were accomplished by trying out different Network Structures and geometries in chapter 4 as well as different smoothing mechanisms on top of the direct neuronal output layer of the network in chapter 5. This is followed by the final summary of our work and an outlook onto future work improving upon these results.

[7] Lecture Translator: `https://lecture-translator.kit.edu`

2. Fundamentals

In the following chapter we want to define and explain terms and concepts used throughout this thesis as well as give an outlook to related work and the general language identification approaches.

2.1. Janus Recognition Toolkit (jrtk)

The Janus Recognition Toolkit (jrtk) also known as just "Janus" is a general-purpose speech recognition toolkit developed in joint cooperation by both the Carnegie Mellon University Interactive Systems Lab and the Karlsruhe Institute of Technology Interactive Systems Lab [LWL+97]. Part of janus and the jrtk are a speech-to-speech translation system which includes Janus-SR the speech recognition component mainly used in this thesis.

Developed to be flexible and extensible the jrtk can be seen as a programmable shell with janus functionality being accessible through objects in the tcl/tk scripting language. It features the IBIS decoder, that uses Hidden Markov Models for acoustic modeling in general, although in this thesis we used a neural network as our speech recognizer to generate the input features required by our Language ID network.

This thesis makes extensive use of the jrtk's and tcl/tk's scripting capabilities to be able to pre-process speech audio files for further use by our experimental setup. It also uses tcl/tk scripts and it's janus API functionality in the development of our smoothing and evaluation scripts as can be seen in Ch. 5.

2.2. Neural Networks

Artificial Neural Networks today are used in many different fields: from image recognition/face recognition in [LGTB97] to Natural Language Processing in [CW08] and, as relevant to this thesis, to Speech Recognition and very successfully as in [HDY+12]. It has also been used in the realm of Language Identification, which will be described in Sec. 2.3. This section will provide fundamental knowledge of how neural networks work and how to train them, to make the understanding of later chapters easier for the reader. The information in this section is based mostly on [HN04] and [GBC16].

2.2.1. General Setup

Neural Networks are based on collections of small "neural units" working together in tandem. The neuron's behavior can be loosely linked to the brain's axons. Each neuron is connected with others and a neuron is "stimulated" by input on these connections and then

decides on its own activation, or stimulation, by using a summation, or threshold, function with a certain limit to decide if the neuron ”fires“ and its own activation is propagated through the network to adjacent units. By changing weigths and activation thresholds in the network its output changes, therefore the training of artificial networks is done by adjusting the connection weights between neurons as well as the thresholds for its activation functions.

2.2.2. Artificial Neuron

An artificial neuron is a mathematical function that consists of four parameters that can be adjusted independently from each other:

- w_i the input weights for all inputs
- Σ the transfer function for summation of the weighted inputs
- φ the activation function that calculates the output value y_k basend on the transfer input and the threshold
- θ the threshold which defines when the neuron activates.

This means an artificial neuron with output y_k is the function 2.1. Many of these neurons coupled together (via the output of a neuron on a previous layer becoming the input for one on the current layer), make an Artificial Neural Network as used in this thesis. A schematic drawing of this can be seen in Fig. **??**.

$$y_k = \varphi(\sum_{j=0}^{m} w_{kj} x_j) \tag{2.1}$$

2.2.3. Network Types

2.2.3.1. Feed-Forward Neural Networks

A basic (non-deep) *Feed-Forward Neural Network* consists of three layers: the input, a hidden layer of neurons and the output layer. Feed-Forward refers to the fact, in opposition to *Recurrent Neural Networks*, that connections between the neural units are not cyclic.

In such a basic network, the output layer consists of as many neurons as classes that the network is trying to classify against and the neuron with the highest activation after entering input, is the classification output of the net.

2.2.3.2. Deep Feed-Forward Neural Networks

Deep Feed-Forward Neural Networks, DNNs, the net-type most used in this thesis, refer to Networks that have more than one hidden layer between input and output, but still feature non-cyclic connections between neurons. DNNs have a better performance than single-hidden-layer-networks in general, but require different techniques for training.

A common description of this phenomen is, that each hidden layer increases the level of abstraction the network can manage. E.g, in image processing, if the first layer recognizes

a color in a certain pixel, then the next layer can infer more abstract characteristics from the output of the first layer. For example, after knowing a certain pixel is dark the next layer can derive that area might be the eye in a picture of a face, etc. This obviously makes more complicated classification tasks possible but also makes learning algorithms more difficult.

2.2.3.3. Deep Recurrent Neural Networks

Deep Recurrent Neural Networks, RNNs, refer to neural networks that are DNN's but cyclic connections are allowed. This means the network can have temporal behavior, so its performance changes dynamically over time.

2.2.4. Learning

The interesting part about Neural Networks is their ability to learn from data and improve their own performance. Improving performance in this case means that by adjusting the available parameters of the neurons part of the network, we minimize a cost function that describes the difference between an optimal output and the actual output.

A Network can be seen to be an approximation of a function f^*. E.g. a network trying to classify an input x into a class y approximates:

$$y = f^*(x) \tag{2.2}$$

Then one run of the Network with parameter set Θ gives us the mapping $y = f(x; \Theta)$ and we are trying to minimize our cost function of $C = f^* - f$ by adjusting the set of parameters in Θ each run.

Three basic approaches exist for training a network:

- *Supervised Training*, where the optimal output for input train data is known. This means, the train data has been pre-classified by a "teacher". The method we use in this thesis and further explained below.
- *Reinforcement Training*, where the optimal input for train data is not known prior to training, but the environment gives the net feedback about its own output and good output is "reinforced" while bad output is discouraged.
- *Unsupervised Training*, where nothing is known about the environment and the net (often) just tries to learn the probabilistic distribution of the data

2.2.4.1. Supervised Training

Supervised Training refers to training where the optimal output for the training data is known, so a classification of the train data exists prior to training. This makes

Stochastic Gradient Descent

2.2.4.2. Sampling

2.3. Related Work

3. Experimental Setup

This chapter lays out the experimental setup used in this thesis. While Language Identification is applicable in many different scenarios, here the focus lies on trying to establish a low-latency online approach for recognizing the spoken language in a university-lecture environment. Because finding a suitable test setup for online data retrieval is hard, the data used was cut to short lengths to make an evaluation as to correctness of the recognition possible in an ”online-like” scenario.

This means that the output of the net is evaluated after short samples of speech and therefore can be seen as indicative of online performance of the neural net.

3.1. Euronews 2014

Our first data set we retrieved from Euronews [1] 2014. Euronews is a TV channel that is broadcast in 13 different languages simultaneously both on TV and over the Web and is semi-automatically transcribed. The data corpus includes 10 languages (Arabian, German, Spanish, French, Italian, Polish, Portuguese, Russian, Turkish and English) with around 18 hours of data per language provided overall. Data was taken both from online video and recordings of the transmissions as described in [Gre14].

It was then broken down to a per-speaker-basis based on its automatic transcriptions. From this data we took a random sample of 10.000 speakers, while making sure the total length of samples of that language were still comparable to the other languages. Details of this breakdown can be seen in table 3.1.

The speaker list was then split into three smaller datasets: the train set, development set and test set using the common Simple Random Sampling. The sizes were 80% train set, and 10% for both the development and test set. Table 3.1 shows the split data for the three sets.

We also had a second Euronews Corpus available that was much larger. This was used to conform intuition that a larger data corpus leads to better results which can be seen in Sec. 4.3. The breakdown of the large corpus’ data is listed in Table **??**

[1] Euronews: http://www.euronews.com/

Language	Number of Speakers	Combined Length
Arabian	1055	16.76 h
German	928	18.80 h
Spanish	932	18.78 h
French	1016	18.67 h
Italian	935	19.00 h
Polish	1229	18.30 h
Portuguese	1062	16.19 h
Russian	958	18.66 h
Turkish	957	18.61 h
English	928	18.54 h
Overall	10000	182.31 h

Table 3.1.: The Euronews corpus speaker breakdown with total utterances length

Set	Number of Speakers	Combined Length
Train	8000	149.76 h
Development	1000	19.46 h
Test	1000	19.29 h

Table 3.2.: The Euronews corpus breakdown into the three data sets.

3.2. Lecture Data

As part of the development of the KIT's Lecture Translator, German lectures at the KIT were recorded and annotated. This is described in [SKM+12]. This thesis then uses parts of this German corpus as well as newer recordings done at the KIT of English lectures with the same setup, English academic talks given at the conference Interspeech, as well as French talks done at the DGA's [2] yearly academic conference on speech recognition.

This lecture data was then used in two different ways: Firstly, to evaluate the 10-Language trained Euronews-Net(s) to see how it would fare in a lecture-environment as part of the KIT's lecture translator. Secondly to train a second net and further try out the findings about the net setup and net evaluation, as found with the Euronews corpus.

The breakdown of speakers and length can be seen in Table 3.2. As the main work was done on the Euronews corpus this data corpus is considerarbly smaller and was mostly just used as a proof-of-concept for a possible integration of a LID-Net into the Lecture Translator.

3.3. European Parliament

As another layer of evaluation we used recordings of the Europen Parliament speeches that are freely available online [3]. The video recordings come with the simultaneous translations

[2] DGA: `http://www.defense.gouv.fr/dga`

[3] EU-Parliament plenary speeches: `http://www.europarl.europa.eu/ep-live/en/plenary/`

Language	Number of Speakers	Combined Length
German	8	16.22 h
French	30	8.25 h
English	27	10.78 h
Overall	65	35.25 h

Table 3.3.: The Lecture Data corpus speaker breakdown with total utterances length.

into all the official languages of the EU-countries. This includes seven of the languages also available on Euronews, namely German, English, French, Spanish, Italian, Polish and Portuguese. We extracted the seven audio tracks embedded in the recordings with ffmpeg [4] and evaluated performance of our Lecture Data as well as Euronews trained nets on this data.

3.4. Feature Preprocessing

This chapter deals with the feature preprocessing used to form normal speech into feature vectors to be understood by neural networks. The setup we used is based on the standard capabilities of the Janus Recognition Toolkit. It is then run through a six layer Automatic Speech Recognition network that was pre-trained on 10 languages. The 2nd last layer of the ASR net is a Bottleneck feature layer, where the feature vectors are extracted and then used as input for the trained Language Identification Network. The following sections describe this Feature Preprocessing for data as well as the first ASR network used to create the BNF features the LID net requires.

3.4.1. Feature Access

3.4.2. Feature Description

The extracted ADC features from the audio files are then used for further preprocessing. We first use a standard Mel filter bank to extract only the necessary coefficients from the ADC0 feature.

First a spectrum is applied to the ADC0 Feature, therefore calculating the Fast Fourier Transformation of the Digitalized Signal.

3.4.3. ASR BNF network

[4] ffmpeg:`https://ffmpeg.org/`

4. LID Network Structure and Results

This chapter describes the actual Language Identification Neural Network trained as well as the results of different network/data setups used. Most network experiments in the Network Geometry, meaning the number of hidden layers as well as the layout of the neurons in these layers were tried using the Euronews corpus. Results from this corpus were then transferred over to the other corpusses, meaning that the network layout that worked best for Euronews was then adjusted for the lecture data but otherwise the geometry was kept intact.

4.1. Basic Setup

Many different tools exist for deep learning, the most acclaimed being Tensorflow [AAB+16], DL4J/ND4j[1] and Theano [BBB+11]. Pre-existing work on Language Identification using ASR BNFs used a python wrapper around Theano for training, that was developed by Jonas Gehring [Geh12]. This thesis continues the use of this wrapper. The basic layout of the network used and improved upon by this thesis were input vectors from the ASR net 3.4.1 with 966 coefficients. The net setup were 5 layers of denoising auto-encoders with each 1000 nodes and a tanh activation function using the mean squared error as the loss function.

The neural net was then trained using mini batches of size 2m and a learning Rate of 0.01. The pretrained net was then retrained with a 1000 neuron to 10 coefficient output to get to our 10 Languages as classes to classify against. In the basic setup this was trained using a learning rate of 1 and the exit condition of a minimum change of 0.005 / 0.0001 for the training/validation data respectively.

The beginning benchmark to improve upon was then set to the frame-based validation/-train error of 0.23 / 0.27 respectively, while of course understanding that non-training per-sample data would most likely have worse results at first than the frame-based validation error.

The following sections describe different network layouts and changes we made to the training of the neural network and the improvements we managed to make upon our initial result.

[1] DL4J: `https://deeplearning4j.org/index.html`

4.2. Improving Network Layout

Different experiments were undertaken with the network layout. This includes a 6 hidden layer pre-training as well as a change in the geometry. The differences in the frame-based errors can be seen in Table ??.

4.3. Results

5. Smoothing and Evaluation

While frame-based error rates on the training/validation sets were already sufficiently good from the LID networks, this of course is not a reliable indicator of real-world online performance, so the development setup consisted of (for each data corpus) a development set of speakers whose samples were run through the LID setup (Feature Preprocessing Sec. 3.4 → ASR BNF extraction Sec. 3.4.3 → LID network Sec. 4). The following smoothing approaches were applied in an ”online” fashion, meaning it was made sure they can be calculated “on-the-fly” while new data is still coming in.

5.1. Basic Test Filter

The first filter tried was a basic 5-Frame smoothing: It saves the value of the last direct outputs and only outputs a language if the last 5 direct outputs would have been the same. It also includes a filtering based on the actual output of the language ID neuron, only counting outputs higher than that. This of course means that the approach requires a 5 frame ($\approx 50ms$) “warm-up” time, which still would make it usable in an online environment. It did however provide no improvement over the bare network output.

See Lst. 5.1 for the corresponding tcl/tk code for this basic filtering approach. The test setup used then goes through the entire development set of samples and counts the correctly/wrongly classified samples. This means that an extra amount of smoothing is included, but results should still be sufficiently general to be able to infer properties of employed filters, as they all include this extra smoothing. Table **??** shows a comparison of the basic filter with a bare setup only outputting the direct output of the LID net. Fig. **??** shows a comparison of the length of samples and the amount of correctly classified samples of this length for the LID net with the best evaluation results, the 6-layered geometry-adjusted lower learning rate net (See Sec. **??**)

```
1  proc filter{} {
2      #setting up variables
3       for {set i 0} {$i < 10} {incr i} {
4          set totalM($i) 0
5      }
6      set lastFrameID -1
7      set counter 0
8      set currentOutput -1
9      #Going through whole sample frame by frame in output layer of nn
           called nnBNF-> can be changed to work on continuously incoming
           data easily
10     for {set i 0} { $i < [featureSetLID frameN nnBNF]} {incr i} {
11         #we find the current output of the net
```

```
12          set maxFrame [lindex [lsort -decreasing -real [featureSetLID
                frame nnBNF $i]] 0]
13          set maxFrameID [lsearch -real [featureSetLID frame nnBNF $i]
                $maxFrame]
14          #Actual Filtering: Only count if last 4 frames were also
                classified to be this language
15          if {$maxFrameID != $lastFrameID} {
16              set lastFrameID $maxFrameID
17              set counter 0
18          #Also filter for the actual output of the neuron
19          } elseif {$maxFrame >= 0.61} {
20              incr counter
21              if {$counter >= 5} {
22                  set currentOutput $maxFrameID
23              }
24          }
25          #setting total classification amounts for current sample
26          if {$currentOutput != -1} {
27                set totalM($currentOutput) [expr {[set totalM(
                      $currentOutput)] + 1}]
28          }
29      }
30      set maxOverall -1
31      set maxID -1
32      #Now get the output for the whole sample (smoothing
33      for {set i 0} {$i < 10} {incr i} {
34          if {[set totalM($i)] > $maxOverall} {
35              set maxOverall [set totalM($i)]
36              set maxID $i
37          }
38      }
39      #print out the total classification for the entire sample
40      puts -nonewline "Overall we have classified as: "
41      #help function to print language name not id.
42      puts [getName $maxID]
43      return $maxID
44  }
```

Listing 5.1: Most basic filter employed to smooth output

5.2. Advanced Test Filter

The advanced test filter is based on the jrtk's *FILTER* capability. It automatically takes a defined amount of frames and calculates the weighted arithmetic mean with predefined weights for incoming audio. First tries were done using a basic filter setup of:

```
filter          nnFILTER   nnBNF  {-2 {1 2 3 2 1}}
```

Herein the context is 2 frames on each side of the current frame (the first parameter) with weights 1, 2 , 3 , 2 , 1 for the 5 frames respectively. It improved the evaluation results, but only by a negligible amount.

It was adjusted to use a 10-frame based approach with descending weights for frames further out:

```
filter          nnFILTERREAL nnBNF {-5 {1 2 3 4 5 6 5 4 3 2 1}}
```

which however did not provide a furhter increase in correctness. The full tcl code for this can be found in the Appendix.

5.3. Variance Test Filter

As a next approach, the variance between the two most likely outputs was taken into account. This however, did not lead to an improvement in the robustness of the classification on the development set. The reason possibly being that the output of the 2^{nd} most likely neuron is not going to differ from the maximum output on many different language pairs: E.g. French/Italian, Russian/Polish, Italian/Portuguese. The full tcl code of this filter can be found in the appendix . The evaluation results can be seen in Tab. **??**

5.4. Two-Language setup

Table **??** shows the result produced by using the LID Euronews net for 10 languages on different combinations of 2 languages (french/italian and english/german). In this case we ignore the output of the net if it doesn't equal one of the two languages and instead keep the previous output intact in this case. This, as intuition predicted, gave a big boost in the recognition rate, bringing the rate up to 85 % for the two languages combined, an improvement of more than 10 % in correctness compared to the basic approach in Sec. 5.1.

6. Conclusion

Bibliography

[AAB+16] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016.

[bAO14] N. bt Aripin and M. B. Othman. Voice control of home appliances using android. In *2014 Electrical Power, Electronics, Communicatons, Control and Informatics Seminar (EECCIS)*, pages 142–146, Aug 2014.

[BBB+11] James Bergstra, Frédéric Bastien, Olivier Breuleux, Pascal Lamblin, Razvan Pascanu, Olivier Delalleau, Guillaume Desjardins, David Warde-Farley, Ian Goodfellow, Arnaud Bergeron, et al. Theano: Deep learning on gpus with python. In *NIPS 2011, BigLearning Workshop, Granada, Spain*, volume 3. Citeseer, 2011.

[CW08] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[FHBM08] A.M. Franz, M.H. Henzinger, S. Brin, and B.C. Milch. Voice interface for a search engine, April 29 2008. US Patent 7,366,668.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[Geh12] Jonas Gehring. *Training deep neural networks for bottleneck feature extraction*. 2012. Karlsruhe, KIT, Pittsburgh, Carnegie Mellon Univ., Interactive Systems Laboratories, Masterarbeit, 2012.

[Gre14] Roberto Gretter. Euronews: a multilingual benchmark for asr and lid. In *INTERSPEECH*, pages 1603–1607, 2014.

[HDY+12] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[HN04] Simon Haykin and Neural Network. A comprehensive foundation. *Neural Networks*, 2(2004):41, 2004.

[LGTB97] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

[LWL+97] Alon Lavie, Alex Waibel, Lori Levin, Michael Finke, Donna Gates, Marsal Gavalda, Torsten Zeppenfeld, and Puming Zhan. Janus-iii: Speech-to-speech translation in multiple languages. In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 1, pages 99–102. IEEE, 1997.

[MNN+16] Markus Müller, Thai Son Nguyen, Jan Niehues, Eunah Cho, Bastian Krüger, Thanh-Le Ha, Kevin Kilgour, Matthias Sperber, Mohammed Mediani, Sebastian Stüker, and Alex Waibel. Lecture translator - speech translation framework for simultaneous lecture translation. In *Proceedings of the Demonstrations Session, NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 82–86, 2016.

[SKM+12] Sebastian Stüker, Florian Kraft, Christian Mohr, Teresa Herrmann, Eunah Cho, and Alex Waibel. The kit lecture corpus for speech translation. In *LREC*, pages 3409–3414, 2012.

[VMH+05] David Vilar, Evgeny Matusov, Saša Hasan, Richard Zens, and Hermann Ney. Statistical machine translation of european parliamentary speeches. In *Proceedings of MT Summit X*, pages 259–266, 2005.

A. Appendix

In this Appendix we present images, complete source code listings as well as a Glossary at the end.