KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Building a state of the art speech recogniser

Moritz Wolter

Thesis submitted for the degree of Master of Science in Mathematical Engineering

Thesis supervisor:
Prof. dr. ir. Patrick Wambacq

Assessor:
Prof. dr. ir. Johan Suykens

Mentor:
Ir. Vincent Renkens

Academic year 2015 – 2016

Preface

I would like to thank everybody who kept me busy the last year, especially my promotor and my assistants. I would also like to thank the jury for reading the text. My sincere gratitude also goes to my family for supporting me trough my studies.

Moritz Wolter

Contents

Abstract

The `abstract` environment contains a more extensive overview of the work. But it should be limited to one page.

List of Figures and Tables

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations

ConvNet	Convolutional neural network
MSE	Mean Square error
PSNR	Peak Signal-to-Noise ratio

Symbols

42	"The Answer to the Ultimate Question of Life, the Universe, and Everything" according to [**?**]
c	Speed of light
E	Energy
m	Mass
π	The number pi

Chapter 1

Literature Study

The first contains a general introduction to the work. The goals are defined and the modus operandi is explained. TODO: describe the problem.

1.1 The classical approach to speech recognition

1.1.1 Preprocessing and feature extraction

f-bank features

f-bank (filter banks) features are one option or raw data. time domian or frequency domain?

1.1.2 Gaussian mixture models

1.1.3 Hidden Markov Models

1.1.4 DNN

Stochastic gradient descent

When training networks on very large training sets, working with the full data set to compute the current gradient becomes very inefficient. As a remedy its is good practice in machine learning to work with so called mini-batches. A mini-batch includes a random subset of the training data set. This procedure is known as randomized gradient descent. With an added momentum term it can be formalized as [6, page 4]:

$$\triangle w_{ij}(t) = \alpha \triangle w_{ij}(t-1) - \epsilon \frac{\partial C}{\partial w_{ij}(t)} \tag{1.1}$$

C is the cost, which is computed by comparing the current network output to the desired output. $\alpha \in (0, 1)$ is a momentum coefficient. The weight of a connection from unit i in the layer under consideration to unit j in the layer below is given by the expression w_{ij}.

Figure 1.1: TODO: basic RNN

1.1.5 Classical layer architecture

1.1.6 Convnets

When looking for a signal in different parts of a recoding it is not always advisable to relearn recognition of the signal in different locations. With a conventional fully connected structure every sample on the time axis gets its own input weight. For shifted version of this signal the network will not be able to reuse weights it used to recognize the same signal at another time point in the recording. Convolutional neural nets aim to solve this problem [4, page 6], while preserving essential ordering information.

1.1.7 Recurrent neural Networks

Feedforward neural nets do not possess memory. These networks make decisions, starting from zero every time. When processing speech its is important however to take context into account. That is why recurrent loops are introduced into the net, which give it access to it's past outputs. Introducing recurrent connections creates the problem of exploding and vanishing gradients. Very large gradients can be treated by clipping, if their norm becomes too large. To tackle the vanishing gradient problem structural changes are required.

Long short-term memory

Long short-term memory (LSTM) cells are one solution to the vanishing gradient problem. These cells use the equation system [5, page 5]:

$$\mathbf{i_t} = \sigma(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ih}\mathbf{h_{t-1}} + \mathbf{W}_{ic}\mathbf{c_{t-1}} + \mathbf{b}_i) \tag{1.2}$$
$$\mathbf{f_t} = \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fh}\mathbf{h_{t-1}} + \mathbf{W}_{fc}\mathbf{c_{t-1}} + \mathbf{b}_f) \tag{1.3}$$
$$\mathbf{c_t} = \mathbf{f_t}\mathbf{c_{t-1}} + \mathbf{i_t}\tanh(\mathbf{W}_{cx}\mathbf{x}_t + \mathbf{W}_{ch}\mathbf{h_{t-1}} + \mathbf{b}_c) \tag{1.4}$$
$$\mathbf{o_t} = \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{oh}\mathbf{h_{t-1}} + \mathbf{W}_{oc}\mathbf{c_t} + \mathbf{b}_o) \tag{1.5}$$
$$\mathbf{h_t} = \mathbf{o}_t\tanh(\mathbf{c}_t) \tag{1.6}$$
$$\tag{1.7}$$

From the definition of the matrix product follows that

$$\mathbf{A}\mathbf{x}_1 + \mathbf{B}\mathbf{x}_2 = \begin{bmatrix}\mathbf{A} & \mathbf{B}\end{bmatrix} \cdot \begin{bmatrix}\mathbf{x}_1 \\ \mathbf{x}_2\end{bmatrix}. \tag{1.8}$$

Which this relation in mind the equations above can be rewritten, by creating column wise concatenated weight matrices for every neuron gate W_i, W_f, W_o, as well as for the state W_c. These matrices can then be multiplied by a row wise concatenated

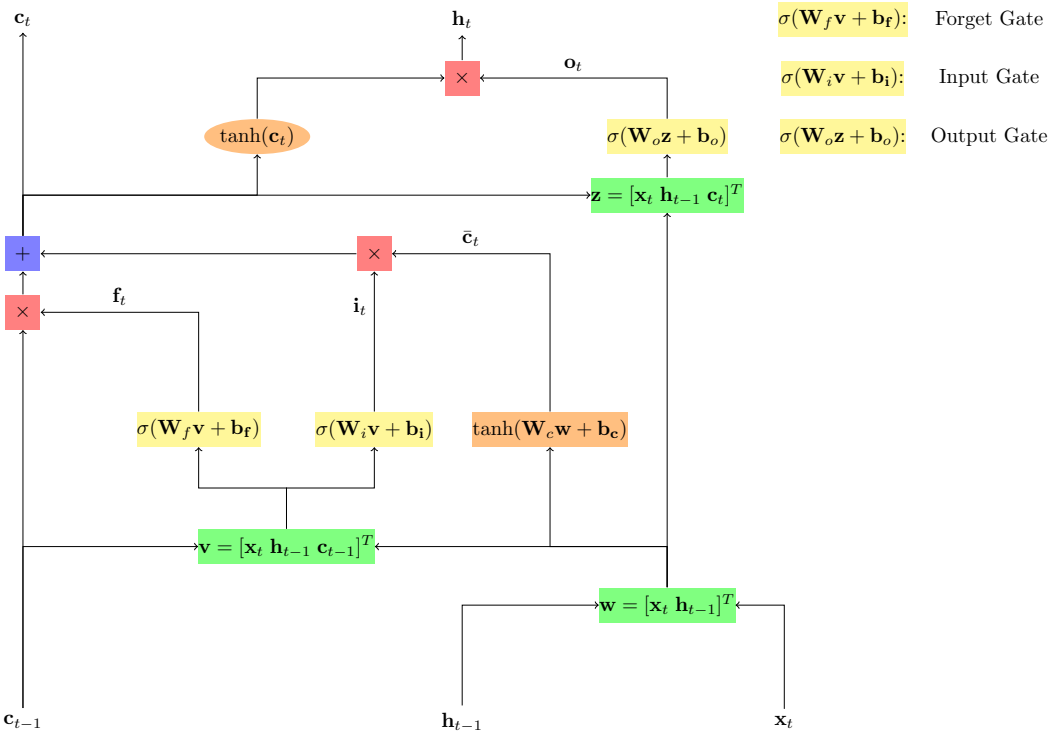

Figure 1.2: Visualization of the LSTM architecture following.

vector $[\mathbf{x}_t\ \mathbf{h_{t-1}}\ \mathbf{c}]^T$, which leads to the slightly simplified system of equations below:

$$\mathbf{i_t} = \sigma(\mathbf{W}_i[\mathbf{x}_t\ \mathbf{h_{t-1}}\ \mathbf{c_{t-1}}]^T + \mathbf{b}_i) \tag{1.9}$$
$$\mathbf{f_t} = \sigma(\mathbf{W}_f[\mathbf{x}_t\ \mathbf{h_{t-1}}\ \mathbf{c_{t-1}}]^T + \mathbf{b}_f) \tag{1.10}$$
$$\mathbf{c_t} = \mathbf{f}_t\mathbf{c_{t-1}} + \mathbf{i}_t \tanh(\mathbf{W}_c[\mathbf{x}_t\ \mathbf{h_{t-1}}]^T + \mathbf{b}_c) \tag{1.11}$$
$$\mathbf{o_t} = \sigma(\mathbf{W}_o[\mathbf{x}_t\ \mathbf{h_{t-1}}\ \mathbf{c_t}]^T + \mathbf{b}_o) \tag{1.12}$$
$$\mathbf{h_t} = \mathbf{o_t} \tanh(\mathbf{c_t}) \tag{1.13}$$

This system of equations is visualized in figure 1.2. The diagram is read from bottom to top. The most important part is the line from $\mathbf{c}_{t-1}$ to $\mathbf{c}_t$ [3]. It records operations on the cell state $\mathbf{c_t}$. The cell state contains information from the past which helps the block make decisions regarding the current output $\mathbf{h}_t$. The sigmoid functions $\sigma()$ are applied element wise on the input vectors and produce outputs between zero and one. In the case of the forget gate output $\mathbf{f}_t$ these values $\in (0, 1)$ well serve as a measure of how much of the past state the cell would like to remember. One means keep this variable and zero throw it away [3]. The following task is to determine what should be added to the memory. This information can be found in the input gate result $\mathbf{i}_t$. $\mathbf{i}_t$ is multiplied element wise with the candidate values $\bar{\mathbf{c}}_t$. These are computed by a hyperbolic tangent neuron. The tanh() function makes sure all vector elements are between -1 and 1. The neuron looks at input data and the past outputs. Both are labeled $\mathbf{w}$ in figure 1.2, $\mathbf{w}$ contains all information that could

possibly be included in the new state. Finally the weighted candidate values are added to what was previously stored. This operation leads to the updated memory state $\mathbf{c}_t$. Last but not least the new output value has to be computed, which will be a filtered version of the cell state. The decision of which and how much of each state variable will be send outside is made by output gate. It's output $\mathbf{o}_t$ is multiplied with a rescaled version of the cell state. The rescaling is done using another hyperbolic tangent, which again sets all values between minus one and one. The product of this rescaled state and the weights found in $\mathbf{o}_t$ then yields the new output $\mathbf{h}_t$.

1.2 Tensor-flow

1.3 Listen, Attend and Spell

The Listen Attend and Spell architecture (LAS) is the main Idea around which this thesis revolves. This section is based on [2]. The las-network consists of two mayor parts, the listener and the speller. The listener is a pyramidal recurrent neural net. It accepts filter bank spectra $\mathbf{x}_n$ as inputs and produces high level output features $\mathbf{h}_m$. The speller in turn accepts the features as input and outputs distributions over Latin character sequences $\mathbf{y}_p$. An overview of the las-achrcitecture is given in figure 1.3.

1.3.1 The listener

The listener shown in figure 1.3 on the bottom, consists of Bidirectional Long Short Term Memory RNN (BLSTM) blocks. These blocks are arranged in a pyramidal structure, such that the time resolution is cut in half in every layer. This operation reduces the length U of the high level features $\mathbf{H}$. Without this compression the following attend and spell operation has a hard time extracting the relevant information. Additionally the compression reduces the problem complexity, which speeds up the training process significantly [2, page 4].

1.3.2 Attend and spell

The speller takes the features and produces a distribution over Latin character sequences as output. The computation of this output involves the context vector $\mathbf{c}_i$, the decoder state $\mathbf{s}_i$, the features $\mathbf{H}$ and the previous output $\mathbf{y}_i$. The index i denotes time, $i-1$ is used to refer to results from the last time step.
These values are computed using [2, page 4]:

$$s_i = \mathrm{RNN}(\mathbf{s}_{i-1}, \mathbf{y}_{i-1}, \mathbf{c}_{i-1}) \tag{1.14}$$

$$\mathbf{c}_i = \mathrm{AttentionContext}(\mathbf{s}_i, \mathbf{H}) \tag{1.15}$$

$$P(\mathbf{y}_i|\mathbf{x}, \mathbf{y}_{<i}) = \mathrm{CharacterDistribution}(s_i, \mathbf{c}_i) \tag{1.16}$$

The state follows from a recurrent neural net (RNN) made of a two layer LSTM. The attention mechanism, called AttentionContext above, computes a new context vector once every time step. This computation starts with the determination of the scalar

energy $e_{i,u}$, which will be used as weight for its corresponding feature vector h_u. The computation starts with two feedforward neural networks or multilayer perceptrons (MLP), ϕ and ψ [2, page 5]:

$$e_{i,u} = \phi(\mathbf{s}_i)^T \psi(\mathbf{h_u}) \tag{1.17}$$

$$\alpha_{i,u} = \frac{\exp(e_{i,u})}{\sum_u \exp(e_{i,u})} \tag{1.18}$$

$$\mathbf{c}_i = \sum_u \alpha_{i,u} \mathbf{h}_u \tag{1.19}$$

α is produced by running $\mathbf{e}$ trough a softmax function, which scales $\mathbf{e}$ such that all elements are within $(0, 1)$ and add up to one. These scaled weights, can then be used to form the context vector $\mathbf{c}_i$. When the training process converges the α_is typically follow a distribution with sharp edges[2, page 5]. Thus it is justified to think of the alphas as a sliding window. This window contains only those parts of the condensed input data set, which are currently relevant.

1.3.3 Training

For end-to-end speech recognition the all networks must be trained jointly. The objective is to maximize the logarithmic probability:

$$\max_\theta \sum_i \log P(y_i|\mathbf{x}, y_{<i}; \theta). \tag{1.20}$$

Here y_i denotes the current output distribution, x the input, θ the various network parameters and finally $y_{<i}$ the ground truth, which is the known true desired output. Using the known output during training creates a situation, where the past outputs are always right. In practice however the situation will be different, as the network is going to make mistakes. As it is desired to create a robust model it is necessary to sometimes include the character distribution generated by the networks being trained. Which leads to the objective [2, page 5]:

$$\hat{y}_i = \text{CharacterDistribution}(s_i, \mathbf{c}_i) \tag{1.21}$$

$$\max_\theta \sum_i \log R(y_i|\mathbf{x}, \hat{y}_{<i}; \theta) \tag{1.22}$$

The novelty in comparison to the previous expression is that $\hat{y}_{<i}$ is sometimes taken from the past network outputs instead of the ground truth. An idea which Chan et al. found in [1].

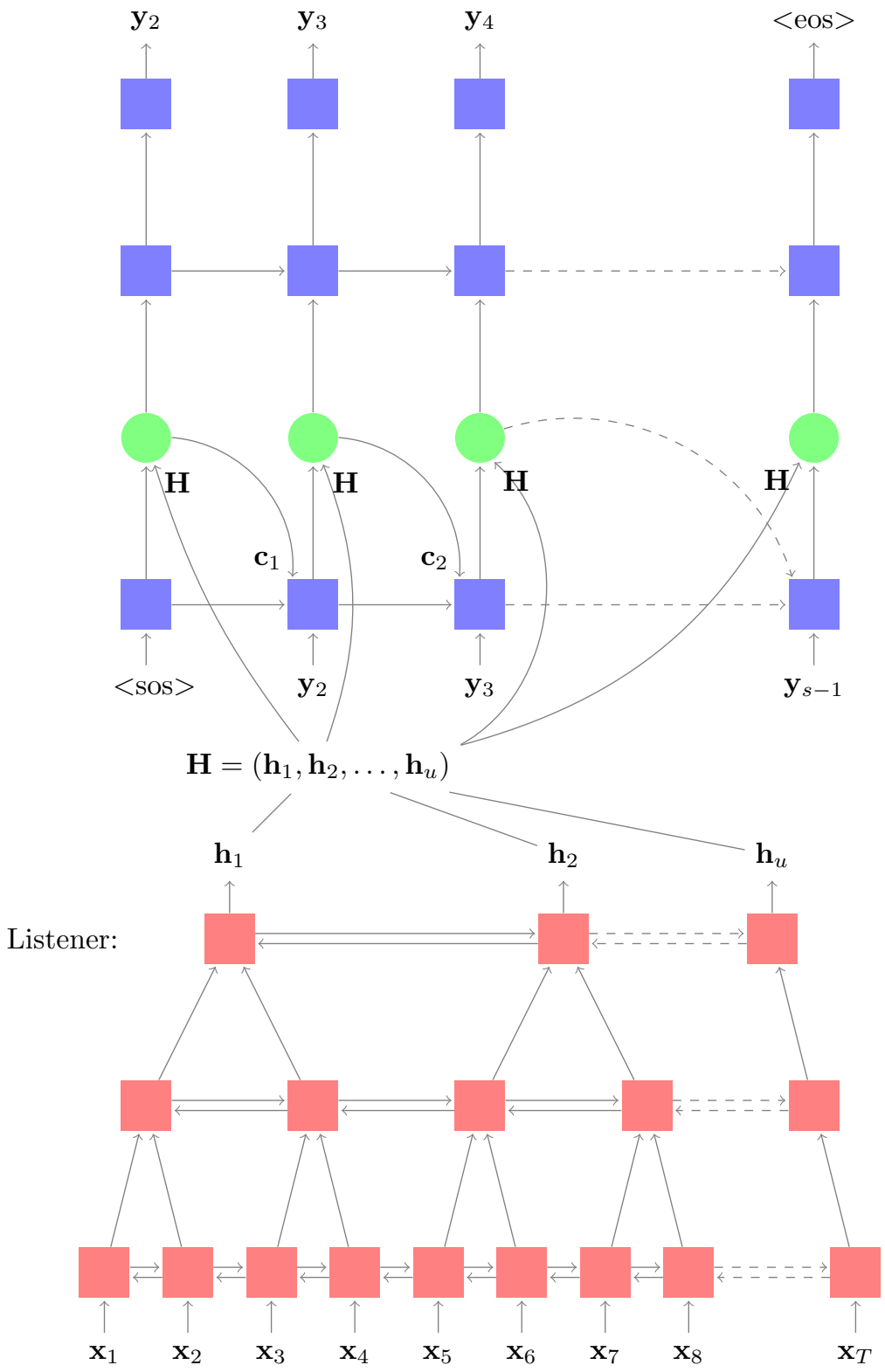

Figure 1.3: The LAS architecture [2, page 3]. BLSTM blocks are shown in red. LSTM blocks in blue and attention nets in green.

Bibliography

[1] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. *arXiv*, pages 1–9, 2015.

[2] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals. Listen, attend and spell. *arXiv preprint*, pages 1–16, 2015.

[3] Colah. Understanding LSTM Networks, 2015.

[4] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning. pages 1–28, 2016.

[5] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, pages 1–43, 2013.

[6] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury. Deep Neural Networks for Acoustic Modeling in Speech Recognition. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

Master thesis filing card

Student: Moritz Wolter

Title: Building a state of the art speech recogniser

UDC: 621.3

Abstract:

In the past machine learning relied heavily on algorithms designed by experts to solve a specific task. Which lead to highly sophisticated algorithms, which could be grasped only by small groups of people. The human brain however does not work this way, although specialized areas exist, these areas consist of similar building blocks. Artificial neural networks attempt to mimic this layout. Similar algorithmic structures are used for a wide variety of tasks. This thesis deals with the application of neural networks in speech recognition. Replacing the various subsystems by one integrated network based approach.

Thesis submitted for the degree of Master of Science in Mathematical Engineering

Thesis supervisor: Prof. dr. ir. Patrick Wambacq

Assessor: Prof. dr. ir. Johan Suykens

Mentor: Ir. Vincent Renkens