Modular Construction of Complex Deep Learning Architectures in HTK

Florian Kreyssig, Chao Zhang and Phil Woodland

Cambridge University Engineering Department

UNIVERSITY OF CAMBRIDGE

Abstract

- Extensions of HTK's Artificial Neural Network (ANN) acoustic model capabilities are presented
- New Layer-Types such as CNN, GRU and LSTM layers are introduced to HTK
- Input-Features can now be combined in a range of different ways and are not necessarily concatenated
- `PyHTK` provides user-friendly python interface to set up an architecture

Building Blocks: Layer Types

- Fully-Connected Layers are supported as in HTK 3.5 [1]
- 2D-Convolutional Layers defined by:
  - Number of Input/Output Feature Maps
  - Stride, Padding, Kernel-Size
- Max- and Average-Pooling Layers
- Recurrent Layers:
  - Simple RNN-Layers, GRU and LSTM layers are supported
  - LSTM-definition is very flexible in terms of bias-vectors and peep-holes
  - are trained using Truncated Backpropagation through time and Frame-Level shuffling, leading to less biased gradients
- Activation-Only Layer
  - Combines Input-Features, adds bias-vector if required and applies Activation-Function
  - can be used to build ResNet-Blocks (see Figure 2)
- Bias-Only Layer
  - Applies Activation-Function to a trainable vector
  - Can be used for scaling outputs of layers

Building Blocks: Activation Functions

- ReLU, Soft-ReLU, Sigmoid, TANH and parameterised versions
- Scaled Exponential-Linear Unit (SELU) and Softmax

Building Blocks: Input-Feature Combinations

- Element-wise Addition as used in ResNets
- Element-wise Multiplication for Gating
- Element-wise MAX Operations
- Concatenation as in HTK 3.5
- Each Input-Feature can be scaled by the value of an output node of another layer as used in Attention-Models

`PyHTK`

- Simple config-file is used to define the neural network
- The network can easily be assembled from the previously described building blocks
- For recurrent layers, the number of unfolded time-steps For training can be defined separately for each layer and network is then automatically unfolded (see Figure 1)
- Recurrent layers can also operate at lower frame-rate as in [2]
- Recurrent layers can be bi-directional

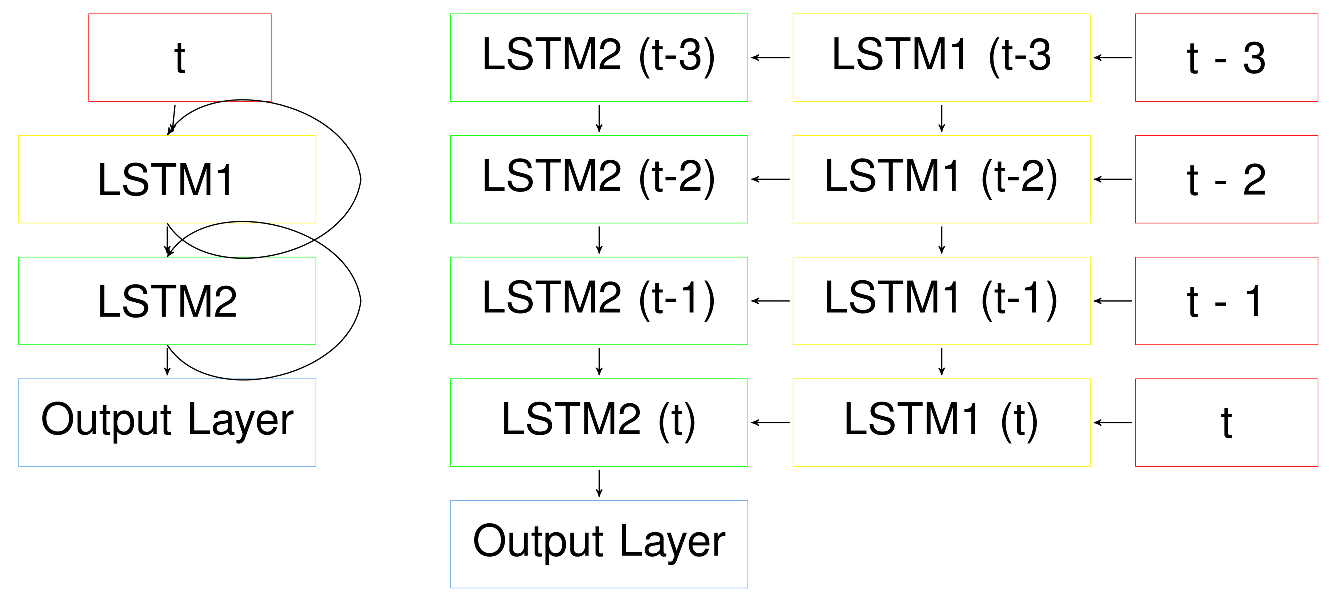

Figure 1: Unfolding a deep recurrent neural network

Experimental Setup and Results

- A range of Models, explained in the next section, are evaluated on TIMIT dataset
- Labels over 854 tri-phone states are derived from 48 phone labels which are mapped to the standard set of 39 phones for testing (after decoding)
- Models were decoded on the full test set using a bigram phone language model
- All models used 24 log-Mel filter bank coefficients with their Δ and $\Delta\Delta$ values as input features, except the CNN which used 40 without any Δ.

Architecture	Width	PER
7L-RELU-MLP	500	21.43
9L-SELU-MLP	250	20.80
21L-(FC)ResNet	250	20.37
CNN	2048 for FC	20.12
3L-RELU-RNN	1024	18.54
3L-RELU-BDRNN	750	18.15
5L-RELU-RNN	750	17.48

Table 1: Phoneme error rates (PER) for the full TIMIT testset

Models

- 21L-(FC)ResNet
  - ResNet Block was created by appending two fully-connected(FC) layers with an ActivationOnly-Layer that performs the addition operation on the inputs see Figure 2
  - 9 blocks, preceded were used to create 21-Layer ResNet
- 9L-SELU-MLP
  - 9-Layer MLP could be trained without pre-training or skip connection
- CNN
  - Conv(9x9)+MaxPool+Conv(4x3)+DNN[3]
- 3L-RELU-RNN
  - PER of 18.54%
- 3L-RELU-BDRNN
  - two Bi-Directional Layers followed by one uni-directional layer
- 5L-RELU-RNN
  - Pre-trained by training three and then four layers for one epoch each, with copying of the parameters of the recurrent layers
  - Noticeable improvement came from using ~5x larger L2-regularization for the two pre-training epochs than for the succeeding epochs

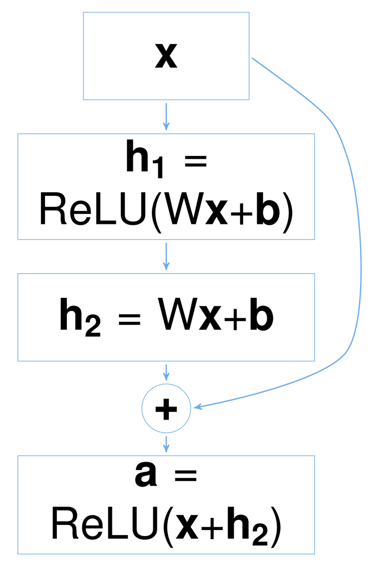

Figure 2: ResNet-Block

References

- C. Zhang, P. C. Woodland, "A General Artificial Neural Network Extension for HTK.", in *Proc. Interspeech'15*.
- V. Peddinti and Y. Wang and D. Povey and S. Khudanpur, "Low latency acoustic modeling using temporal convolution and LSTMs", in *IEEE Signal Processing Letters*, 2017.
- G. Klambauer and T. Unterthiner, and A. Mayr and S. Hochreiter, "Self-Normalizing Neural Networks", *arXiv preprint arXiv:1706.02515*, 2017.

E-Mail: `flk24@cam.ac.uk`