Modular Construction of Complex Deep Learning Architectures in HTK

Florian Kreyssig, Chao Zhang and Phil Woodland
Cambridge University Engineering Department

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

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
- Noticeable improvement came from using ~5x larger L2-regularization for the two pre-training epochs than for the succeeding epochs

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

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Width</th>
<th>PER</th>
</tr>
</thead>
<tbody>
<tr>
<td>7L-RELU-MLP</td>
<td>500</td>
<td>21.43</td>
</tr>
<tr>
<td>9L-SELU-MLP</td>
<td>250</td>
<td>20.80</td>
</tr>
<tr>
<td>21L-(FC)ResNet</td>
<td>250</td>
<td>20.37</td>
</tr>
<tr>
<td>CNN</td>
<td>2048</td>
<td>20.12</td>
</tr>
<tr>
<td>3L-RELU-RNN</td>
<td>1024</td>
<td>18.54</td>
</tr>
<tr>
<td>3L-RELU-BDRNN</td>
<td>750</td>
<td>18.15</td>
</tr>
<tr>
<td>5L-RELU-BDRNN</td>
<td>750</td>
<td>17.48</td>
</tr>
</tbody>
</table>

References