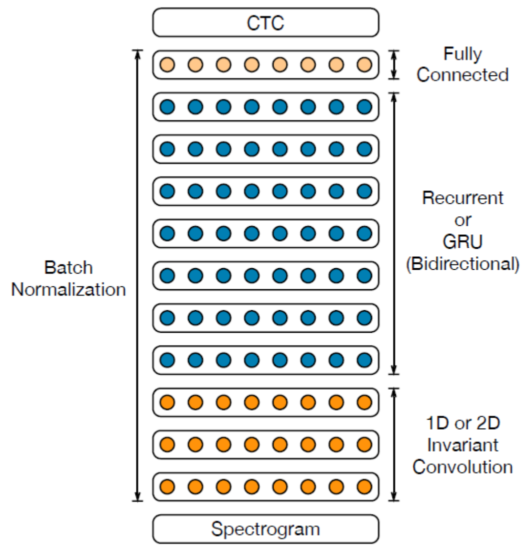
Recep Oğuz Araz
Prof. Engin Erzin
07.11.2020
ELEC 390

Report III: Lightweight DS2 Requirements



Input Layer

Path ➔ audio➔ Spectrogram ➔ Input processing ➔ Batch

$$\text{Input size} = N_{Batch} * N_{Features} * N_{Frames} \text{ or,}$$

$$N_{Batch} * 1 * N_{Features} * N_{Frames} \text{ if 2D Convolution}$$

Because N_{Frames} is different for each recording, we will have to zero pad to batch correctly.

For each Batch, $N_{Frames} = \max\{length(Sequence_i) : i = 1,2,3,\ldots,N_{Batch}\}$

Convolutional Layers

1 – 3 layers of 1D or 2D Invariant Convolutions

Same convolution: Number of input features are preserved in both frequency and time. (?)

Stride: sometimes specified a stride over either dimension to reduce input size.

DS2: using 2D conv is much better than 1D. Stride 2,3,4.

Lightweight Model: 1 layer of 1D Invariant Convolutional layer.

Recep Oğuz Araz
Prof. Engin Erzin
07.11.2020
ELEC 390

Recurrent Layers

Elman RNN, LSTM or GRU for recurrent layers.

Bidirectional RNN. (BiGRU, BiLSTM possible)

The input weights are shared for both directions.

The hidden states are defined as the sum of forward and backward hidden states.

No output calculation.

5-9 stacked RNN layers.

Activation function not specified.

Lightweight Model: 5 stacked Bidirectional Elman (Vanilla) RNNs

Fully connected Layers

1 or more fully connected layers.

Activation function not specified.

Lightweight Model: 1 fully connected layer.

Output Layer

Softmax, computing probability distribution over characters.

Loss Function

CTC loss (in some repositories CTC-Warp is used???)

Challenges

1. Finding which dimension 1D Convolution is over?
2. How to deal with the zero pads of the batches in the Recurrent Layer?
3. Defining the RNN cells for the recurrent layer with shared input matrices in PyTorch.
4. Defining the output of the Bidirectional RNN layer with PyTorch.
5. Dealing with variable output sizes of the Recurrent Layer in Fully Connected Layer.