Explanation of CNN Code

The BuildModel action is specified, and an empty deep learning convolutional model is created.

**proc cas;**

**BuildModel / modeltable={name='ConVNN', replace=1} type = 'CNN'**

1. The first layer added is an input layer. The number of channels is set to 3 in the NCHANNELS= option. The width and height of the image is also specified in the input layer. The offsets are also applied.

**AddLayer / model='ConVNN' name='data' layer={type='input' nchannels=3 width=32 height=32 offsets={113.852228, 123.021097, 125.294747}};**

1. Next, a convolutional layer is added to the data. The convolutional layer contains six different size filters (1x1, 3x3, 5x5, 7x7, 4x4, and 6x6). The odd-value-sized filters use a stride of 1, the 4x4-sized filters use a stride of 2, and the 6x6-sized filters use a stride of 4. The 4x4 and 6x6 sets include four extra filters. In addition, dropout is applied to both the 4x4 and 6x6 sets of filters. The exponential linear (ELU) activation function is used is each filter.

**AddLayer / model='ConVNN' name='ConVLayer1a' layer={type='CONVO' nFilters=12 width=1 height=1 stride=1 act='ELU'} srcLayers={'data'};**

**AddLayer / model='ConVNN' name='ConVLayer1b' layer={type='CONVO' nFilters=12 width=3 height=3 stride=1 act='ELU'} srcLayers={'data'};**

**AddLayer / model='ConVNN' name='ConVLayer1c' layer={type='CONVO' nFilters=12 width=5 height=5 stride=1 act='ELU'} srcLayers={'data'};**

**AddLayer / model='ConVNN' name='ConVLayer1d' layer={type='CONVO' nFilters=12 width=7 height=7 stride=1 act='ELU'} srcLayers={'data'};**

**AddLayer / model='ConVNN' name='ConVLayer1e' layer={type='CONVO' nFilters=16 width=4 height=4 stride=2 dropout=.2**

**act='ELU'} srcLayers={'data'};**

**AddLayer / model='ConVNN' name='ConVLayer1f' layer={type='CONVO' nFilters=16 width=6 height=6 stride=4 dropout=.25**

**act='ELU'} srcLayers={'data'};**

1. A concatenation layer is used to combine the four paths into a single path in preparation for the next layer. Specifically, the convolutions that use a stride of 1 are combined and connected to the next layer. The other convolutions will be connected to later layers.

**AddLayer / model='ConVNN' name='concatlayer1a' layer={type='concat'} srcLayers={'ConVLayer1a','ConVLayer1b','ConVLayer1c', 'ConVLayer1d'};**

d. The concatenation layer is connected to a pooling layer that is set to extract the maximum value from each neighborhood. The pooling layer creates a 2 \* 2 neighborhood that moves across columns with a stride of 2.

**AddLayer / model='ConVNN' name='PoolLayer1max' layer={type='POOL' width=2 height=2 stride=2 pool='max'} srcLayers={'concatlayer1a'};**

1. A concatenation layer is used to combine the pooling layer with the convolutional layer that is connected to the input layer and uses a 4x4 filter.

**AddLayer / model='ConVNN' name='concatlayer2' layer={type='concat'} srcLayers={'PoolLayer1max','ConVLayer1e'};**

f. The concatenation layer is then connected to another 2x2 pooling layer with a stride of 2, max pooling layer.

**AddLayer / model='ConVNN' name='PoolLayer2max' layer={type='POOL' width=2 height=2 stride=2 pool='max'} srcLayers={'concatlayer2'};**

g. A concatenation layer is used to combine the pooling layer with the convolutional layer that is connected to the input layer and uses a 6x6 filter.

**AddLayer / model='ConVNN' name='concatlayer3' layer={type='concat'} srcLayers={'PoolLayer2max','ConVLayer1f'};**

h. The concatenation layer then branches and is connected to several other layers. The first layer the concatenation layer is connected to is another 2x2 max pooling layer with a stride of 2.

**AddLayer / model='ConVNN' name='PoolLayer3max' layer={type='POOL' width=2 height=2 stride=2 pool='max'} srcLayers={'concatlayer3'};**

i. **The concatenation layer is also connected to a convolution layer with 64, 3x3 filters that use a stride value of 1, and incorporates batch normalization. Typically, the hidden bias would be removed before applying batch normalization. However, in an attempt to show the flexibility of SAS, the instructor decided to not follow conventional methods.**

**AddLayer / model='ConVNN' name='ConVLayer1g' layer={type='CONVO'**

**nFilters=64 width=3 height=3 stride=2 init='msra2' dropout=.2} srcLayers={'concatlayer3'};**

j. The concatenation layer is connected to a second convolution layer (third branch) that includes 3x3 filters that move across the input columns using a stride of 2. Batch normalization is again applied and there are

**AddLayer / model='ConVNN'name='BatchLayer1' layer={**

**type='BATCHNORM' act='ELU'} srcLayers={' ConVLayer1g'};**

**AddLayer / model='ConVNN' name='ConVLayer1h' layer={type='CONVO'**

**nFilters=128 width=3 height=3 stride=2 init='msra2' dropout=.2 }**

**srcLayers={'BatchLayer1'};**

128 filters used in the layer. ELU activation function is applied to the feature map values.

**Note:** Concatenation layers require that the incoming feature maps be the same size. It is a common practice to down-sample larger feature maps when combining feature maps of varying sizes. To down-sample, simply increase the stride value to a value greater than one.

**AddLayer / model='ConVNN' name='BatchLayer2'layer={**

**type='BATCHNORM' act='ELU'} srcLayers={'ConVLayer1h'};**

k. . A concatenation layer is used to combine the pooling layer with the batch layer 2.

**AddLayer / model='ConVNN' name='concatlayer4' layer={type='concat'} srcLayers={'PoolLayer3max','BatchLayer2'};**

L. One fully connected layer is added and has 240 neurons. Batch normalization is applied to each of the fully connected layers. The fully-connected layer has an aggressive dropout rate of 65% applied. Batch normalization is added to fully connected layer.

**AddLayer / model='ConVNN' name='FCLayer1' layer={type='FULLCONNECT' n=240 act='Identity' init='msra2' dropout=.65 includeBias=False} srcLayers={'concatlayer4'};**

**AddLayer / model='ConVNN' name='BatchLayer3' layer={type='BATCHNORM' act='ELU'} srcLayers={'FCLayer1'};**

M. Output layer is connected to fully connected layer with SoftMax as activation function.

**AddLayer / model='ConVNN' name='outlayer' layer={type='output' act='SOFTMAX'} srcLayers={'BatchLayer3'};**