Convolution Neural Network

Chapter 3– (Introduction to Deep Learning)

Neural network layer connectivity

- Forward feed neural networks are fully connected.
  - All linear units of a layer is connected to all linear units of the next layer.
  - Number of outputs of a layer is equal to the number of input of next layer.
- Neural network does not have requirement of being fully connected.
- Convolution neural networks are special case of partially connected neural networks.
  - Useful approach for calculating local light intensity differences.

Convolution filter

- Convolution filter or convolution kernel is a small array of numbers.
- Sample filter shows as below,

1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0

- To convolve a filter on an image, is to take the dot product of the filter and equal size piece of the image.
  - Dot product multiplies the corresponding elements of the arrays and sum all products.

Convolution filter function

- Convolution kernel is a function, also known as kernel function.
- This function provides a value V at a position x,y of image I.

$$V(x,y) = (I \cdot K)(x,y) = \sum_m \sum_n I(x+m, y+n) K(m,n)$$

- Convolution takes two functions I and K to return a third function which performs the operation shown in the rightmost side of the above equation.
- Convolution filter does not need to have absolute values. They can be designed based on the changes in light intensity.

Convolution function Example

- A convolution function with the below filter and image data.

1.0	1.0	1.0	1.0
1.0	1.0	1.0	1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0

Sample Filter

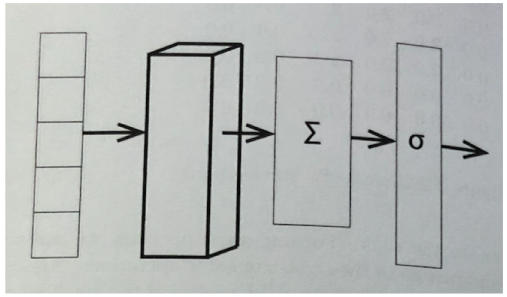
0.0	0.0	0.0	0.0	0.0	0.0
0.0	2.0	2.0	2.0	0.0	0.0
0.0	2.0	**2.0**	**2.0**	**0.0**	**0.0**
0.0	2.0	**2.0**	**2.0**	**0.0**	**0.0**
0.0	0.0	**0.0**	**0.0**	**0.0**	**0.0**
0.0	0.0	**0.0**	**0.0**	**0.0**	**0.0**

Image

- If we convolve the filter on the bottom rightmost part of the image, then the top leftmost part of the filter overlap with 2.0 of the image.
- The value of the filter from this convolution is 8.

Convolution Neural Network

- As illustrated in the previous slide, the filters are smaller than the image and convolution function works on a patch of the image.
- This process can be done multiple times to cover the full image. Different filters can be used with different patches of the image to bring out specific features of the image.



- Filter values from each convolution function is fed to softmax function and in turn to loss function.

Stride

- Stride is the distance between two applications of the filter.
- A stride of two, implies that the convolution filter be applied to every other pixel of the image.
- Horizontal stride is defined as S_h and vertical stride is defined as S_v.
  - A filter will be applied to every S_h pixel in a given line and will be descended to every S_v line at the end of the previous line.
- A stride decides where the filter is applied next.

Padding

- Padding decides the end of line for the filter application.
- Two types of padding:
  - Valid padding
  - Same padding
- After moving the filter S_h stride, there could be three possible scenarios:
  - We are nowhere near the edge of the image.
  - Next leftmost pixel of the patch is beyond the image edge.
    - Same padding stops in this case.
  - Filter is moving out of the image on the rightmost side.
    - Valid padding stops in this case.

Valid vs. Same Padding

- Same padding uses imaginary pixels by the time, filter reaches end of the line.
  - Tensorflow uses 0 for imaginary boundary pixels.
  - Padding is applied equally to all the edges of the image.
- Valid padding never uses imaginary pixels because filter stops at the edge of the image.
- Same padding provides same number of output as input image pixels when stride is set to 1, while valid padding always provide output smaller than input image pixels.
- Same padding is quite popular compared to valid padding.

Convolution NN with Tensorflow

```
image = tf.reshape (img, [100, 28, 28, 1])
```

- Image is the 4 dimensional tensor. The batch size is set to 100, and each image is represented by 28x28 pixels. Each image has 1 channel as each is a black and white image.
- img in the above line represents the one-dimensional vector of an image with 784 values in each.

```
flts = tf.Variable(tf.truncated_normal([4,4,1,4], stddev=0.1))
```

- This line creates a filter. Here 4x4 size filter is created with 1 channel. Total 4 such filters will be created. Number of channels are decided based on the image channel number.

Convolution NN with Tensorflow – cont'd

```
convOut = tf.nn.conv2d(image, flts, [1,2,2,1], "SAME")
```

- Above line contains the logic for convolution neural network. As the name suggest, conv2d works with images.
- Stride argument in the above line is a list of 4 arguments. These arguments relate to each dimension of the input.
  - First and last values of the list is set to 1. The first 1 signifies that each image will be processed in the batch and the last 1 signifies that each channel of the image will be processed.
  - 2,2 will convolve only every other image patch in both horizonal and vertical direction.
- The final argument is the padding type which is set to "SAME" padding.

Convolution NN with Tensorflow – cont'd

- The output of conv2d function is a 4 dimension tensor. The first dimension is the batch size.
- Next 2 dimensions are number of filter applications horizontally followed by vertically.
- The last dimension is the number of filters used.
- For the example here, the input was each image with 28x28 pixels and 1 channel. The output of conv2d will be 14x14 with 4 channels for 4 filters used.
- This output will be fed to the non-linear function and then used to generate logits.

Multilevel Convolution

- Accuracy can be improved by adding multiple layers of convolution.
- In a single level convolution, the input is the 3 dimensional vector which has 2 dimensional image and last dimension represents channels.
- The output of the single level convolution as well is 3 dimensional vector which has filter application values in 2 dimensional vector while the last dimension represents the number of filters.
- This makes it easy to use the output of tf.nn.conv2d as an input to the next level tf.nn.conv2d.

Multilevel Convolution NN flow

```
flts = tf.Variable(tf.normal([4,4,1,16], stddev=0.1))
convOut = tf.nn.conv2d(image, flts, [1,2,2,1], "SAME")
flts2 = tf.Variable(tf.normal([2,2,16,32], stddev=0.1))
convOut2 = tf.nn.conv2d(convOut, flts2, [1,2,2,1], "SAME")
```

- The first filter variable is 4x4 with 1 channel and there are total 16 filters applied to the image in tf.nn.conv2d.
- The second filter variable is 2x2 with 16 channels and there will be 32 such filters applied to the output of first convolution layer.
- To put it numerically, the 784 pixel image was converted to 7x7 which is 49 pixel with 32 filters values.

Convolution Details - Biases

- A bias is used when multiple filters are applied to a patch of the image.
- It provides a way to give more or less weight to a particular filter channel by adding values to the convolution output.

```
bias = tf.Variable(tf.zeros [16])
convOut += bias
```

- A bias in the program can be added by using two lines shown above.

Convolution Details – Layers with Convolution

- A convolution layer can be defined as below

```
tf.contrib.layers.conv2d(input, numFlts, fltDim, strides, pad)
```

- For a multilevel convolution,

```
convOut = layers.conv2d(image, 16, [4,4], 2, "Same")
```

- The above line creates 16 different filters each with dimension of 4x4. Horizontal and vertical strides are set to 2 with SAME padding.
- By default, layers.conv2d assumes that biases are included in the calculation. User can explicitly disable use of biases using use_bias = False argument.

Convolution Details - Pooling

- For pictures with larger pixel values, reduction in image size from original image to values fed to fully connected layer can be extereme.

```
convOut = tf.nn.conv2d(image, flts, [1,1,1,1], "SAME")
```

- Above statement has stride set to 1 which makes convOut of dimension [100, 28,28,1].

```
convOut = tf.nn.max_pool(convOut, [1,2,2,1], [1,2,2,1], "SAME")
```

- max_pool finds maximum value for a filter in a region in the image. The first, third and last arguments are same as tf.nn.conv2d.
- The second argument specifies the region of the image to look for the max value. The two 2s in the second argument asks to take maximum over 2x2 patch of convOut.