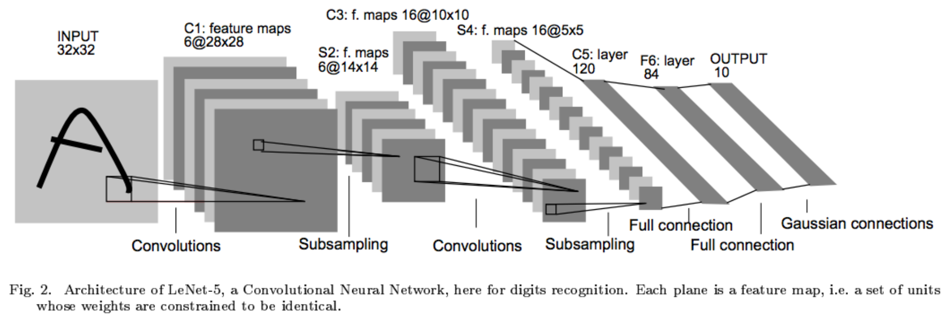
Lab: LeNet in TensorFlow



LeNet. Source: Yann Lecun.

You're now going to put together everything you've learned and implement the LeNet architecture using TensorFlow.

When you get to your next project, remember that LeNet can be a great starting point for your network architecture!

Instructions:

1. Set up your development environment with the CarND Starter Kit
2. `git clone https://github.com/udacity/CarND-LeNet-Lab.git`
3. `cd CarND-LeNet-Lab`
4. `jupyter notebook`
5. Finish off the architecture implementation in the `LeNet` function. That's the only piece that's missing.

Preprocessing

An MNIST image is initially 784 features (1D). If the data is not normalized from [0, 255] to [0, 1], normalize it. We reshape this to `(28, 28, 1)` (3D), and pad the image with 0s such that the height and width are 32 (centers digit further). Thus, the input shape going into the first convolutional layer is `32x32x1`.

Specs

Convolution layer 1. The output shape should be `28x28x6`.

Activation 1. Your choice of activation function.

Pooling layer 1. The output shape should be `14x14x6`.

Convolution layer 2. The output shape should be `10x10x16`.

Activation 2. Your choice of activation function.

Pooling layer 2. The output shape should be `5x5x16`.

Flatten layer. Flatten the output shape of the final pooling layer such that it's 1D instead of 3D. The easiest way to do is by using `tf.contrib.layers.flatten`, which is already imported for you.

Fully connected layer 1. This should have **120 outputs**.

Activation 3. Your choice of activation function.

Fully connected layer 2. This should have **84 outputs**.

Activation 4. Your choice of activation function.

Fully connected layer 3. This should have **10 outputs**.

You'll return the result of the final fully connected layer from the `LeNet` function.

If implemented correctly you should see output similar to the following:

```
EPOCH 1 ...
Validation loss = 52.809
Validation accuracy = 0.864

EPOCH 2 ...
Validation loss = 24.749
Validation accuracy = 0.915

EPOCH 3 ...
Validation loss = 17.719
Validation accuracy = 0.930

EPOCH 4 ...
Validation loss = 12.188
Validation accuracy = 0.943

EPOCH 5 ...
Validation loss = 8.935
Validation accuracy = 0.954

EPOCH 6 ...
Validation loss = 7.674
Validation accuracy = 0.956

EPOCH 7 ...
Validation loss = 6.822
Validation accuracy = 0.956

EPOCH 8 ...
Validation loss = 5.451
Validation accuracy = 0.961

EPOCH 9 ...
Validation loss = 4.881
Validation accuracy = 0.964

EPOCH 10 ...
Validation loss = 4.623
Validation accuracy = 0.964

Test loss = 4.726
Test accuracy = 0.962
```

Parameters Galore

As an additional fun exercise calculate the total number of parameters used by the network. Note, the convolutional layers use weight sharing!

Supporting Materials
lenet.py