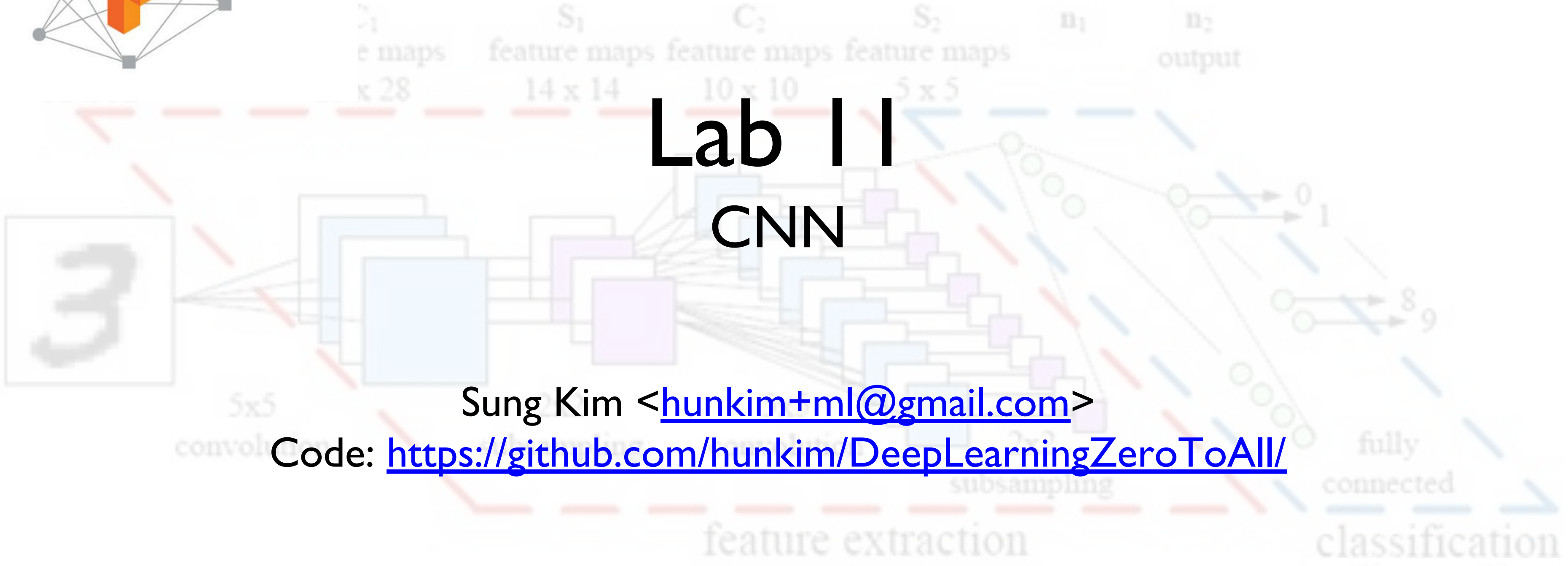
With TF 1.0!



Lab 11

CNN

Sung Kim <hunkim+ml@gmail.com>
Code: https://github.com/hunkim/DeepLearningZeroToAll/

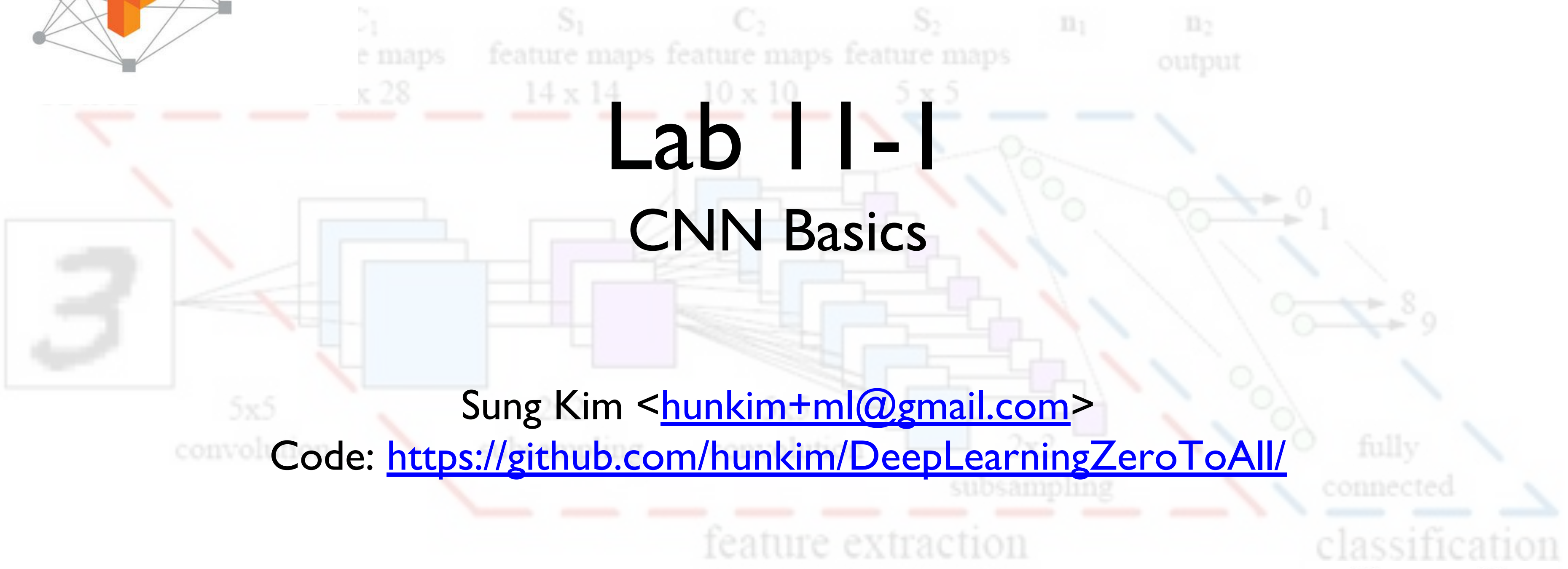
Call for comments

Please feel free to add comments directly on these slides

Other slides: https://goo.gl/jPtWNt

Picture from http://www.tssablog.org/archives/3280

With TF 1.0!



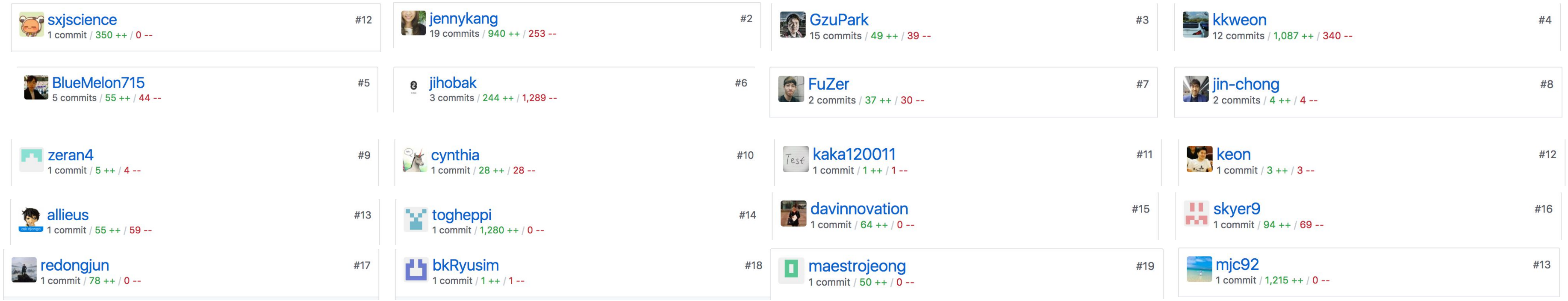
Lab 11-1

CNN Basics

Sung Kim <hunkim+ml@gmail.com>

Code: https://github.com/hunkim/DeepLearningZeroToAll/

https://github.com/hunkim/DeepLearningZeroToAll/



sxjscience #12
1 commit / 350 ++ / 0 --

jennykang #2
19 commits / 940 ++ / 253 --

GzuPark #3
15 commits / 49 ++ / 39 --

kkweon #4
12 commits / 1,087 ++ / 340 --

BlueMelon715 #5
5 commits / 55 ++ / 44 --

jihobak #6
3 commits / 244 ++ / 1,289 --

FuZer #7
2 commits / 37 ++ / 30 --

jin-chong #8
2 commits / 4 ++ / 4 --

zeran4 #9
1 commit / 5 ++ / 4 --

cynthia #10
1 commit / 28 ++ / 28 --

kaka120011 #11
1 commit / 1 ++ / 1 --

keon #12
1 commit / 3 ++ / 3 --

allieus #13
1 commit / 55 ++ / 59 --

togheppi #14
1 commit / 1,280 ++ / 0 --

davinnovation #15
1 commit / 64 ++ / 0 --

skyer9 #16
1 commit / 94 ++ / 69 --

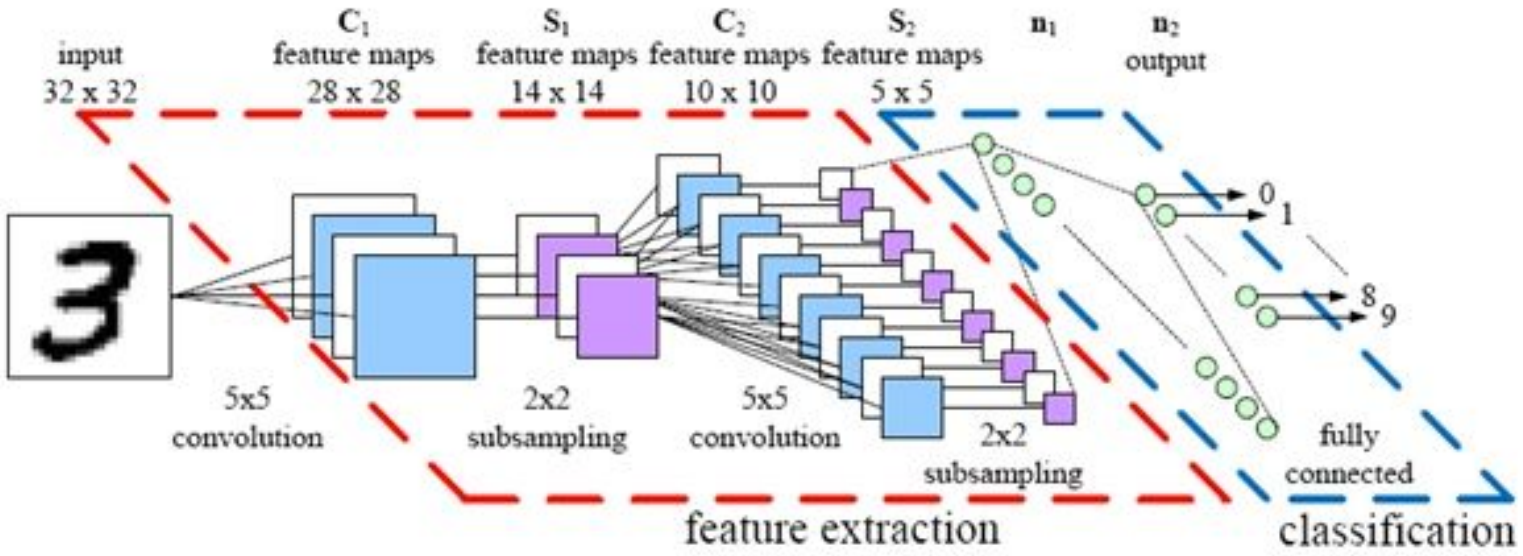
redongjun #17
1 commit / 78 ++ / 0 --

bkRyusim #18
1 commit / 1 ++ / 1 --

maestrojeong #19
1 commit / 50 ++ / 0 --

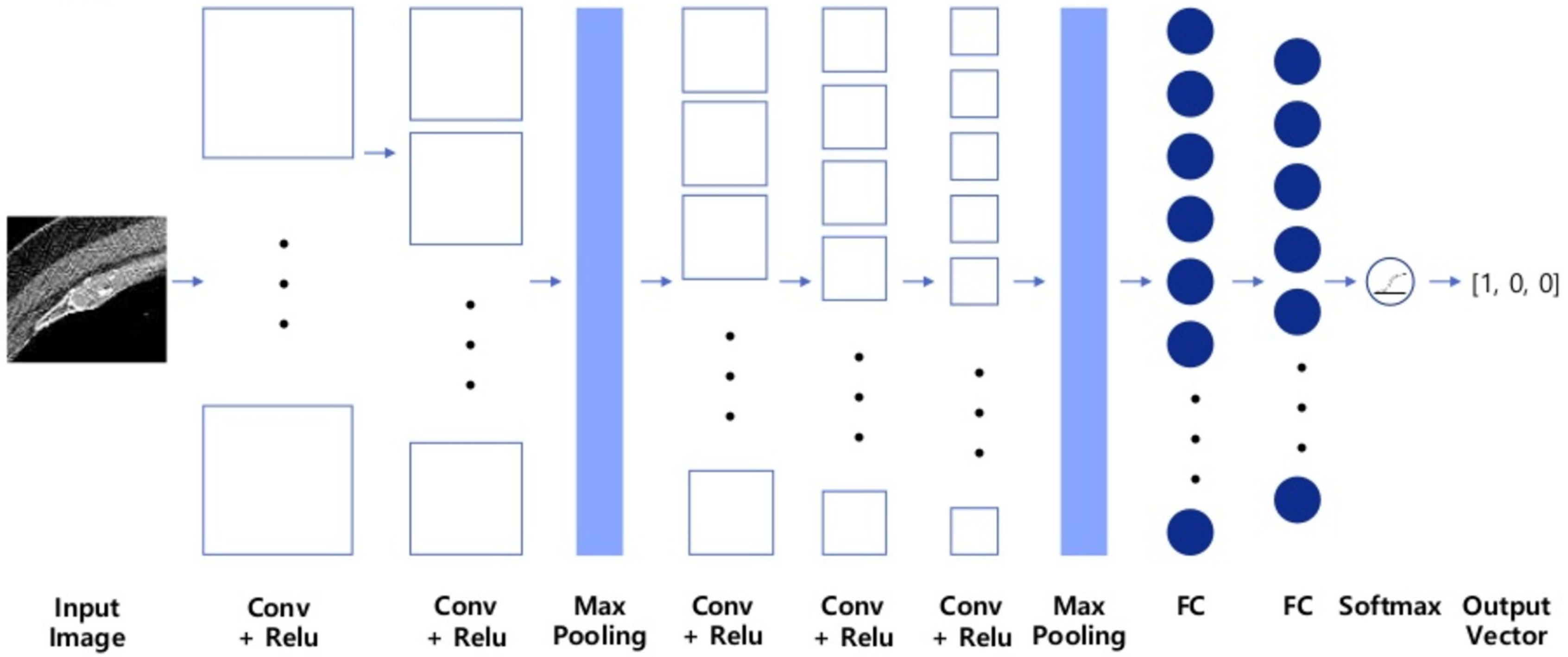
mjc92 #13
1 commit / 1,215 ++ / 0 --

CNN



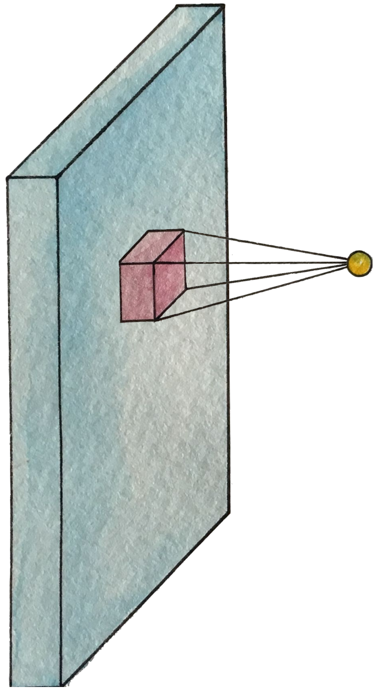
http://parse.ele.tue.nl/cluster/2/CNNArchitecture.jpg

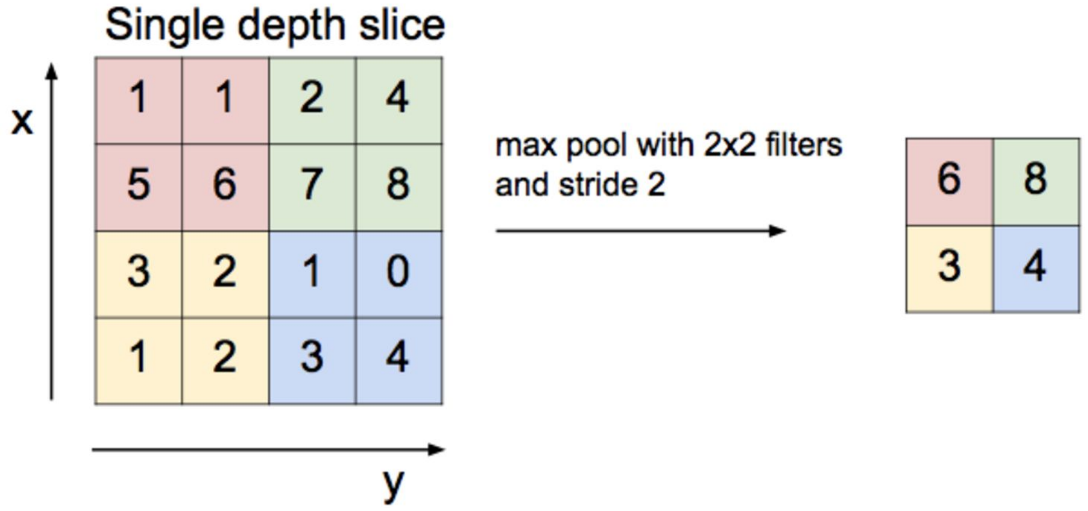
CNN for CT images



Asan Medical Center & Microsoft Medical Bigdata Contest Winner by GeunYoung Lee and Alex Kim
https://www.slideshare.net/GYLee3/ss-72966495

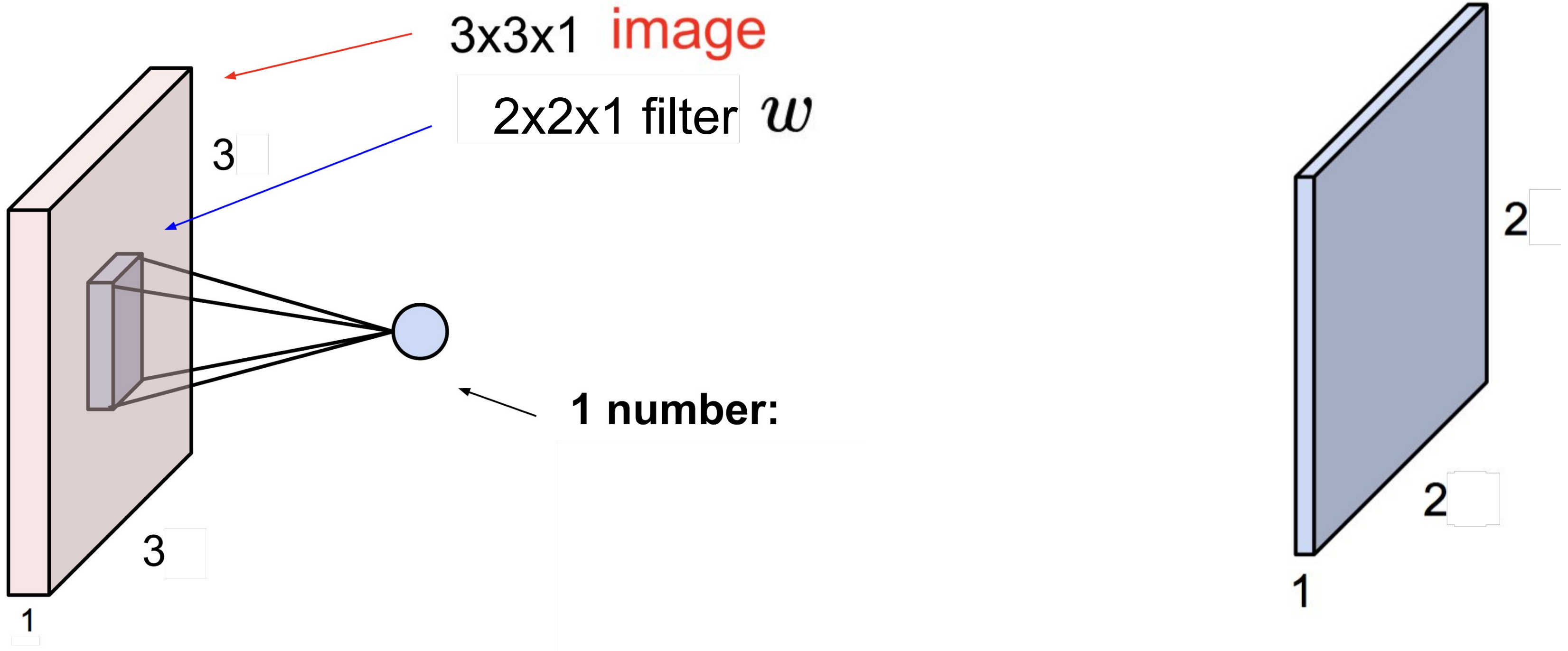
Convolution layer and max pooling

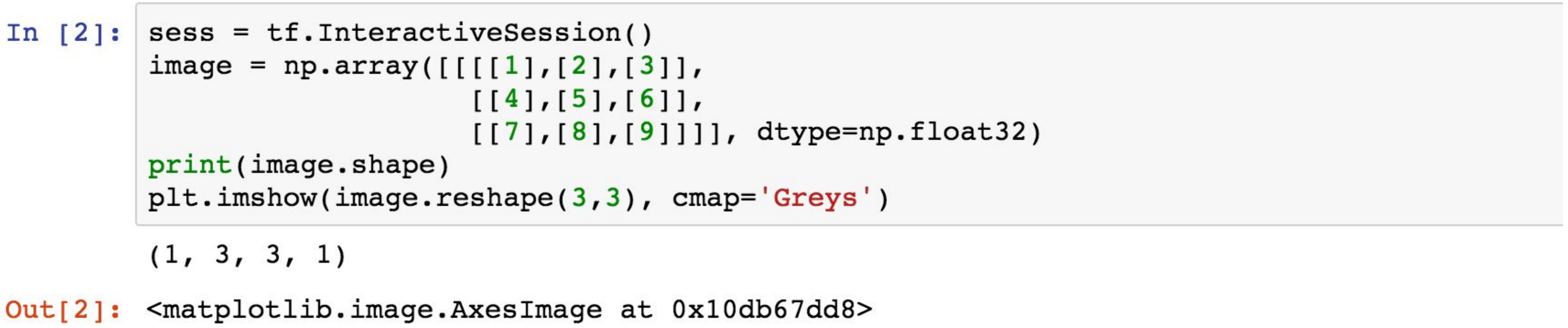




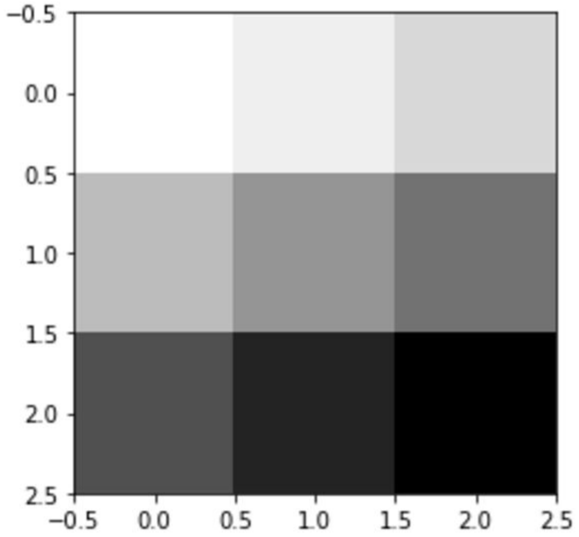
Simple convolution layer

`Stride: 1x1`





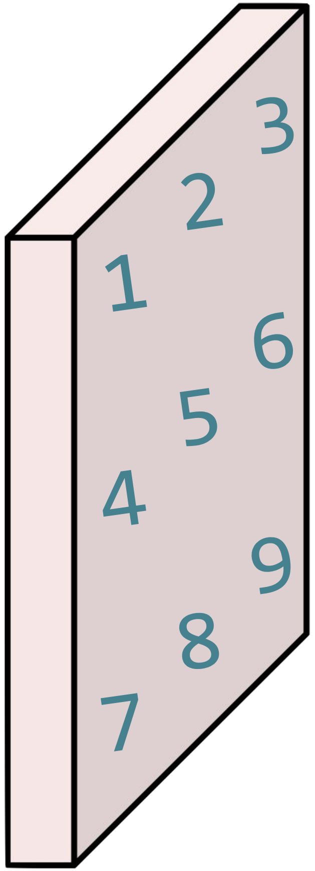
```
In [2]: sess = tf.InteractiveSession()
        image = np.array([[[[1],[2],[3]],
                           [[4],[5],[6]],
                           [[7],[8],[9]]]], dtype=np.float32)
        print(image.shape)
        plt.imshow(image.reshape(3,3), cmap='Greys')

(1, 3, 3, 1)

Out[2]: <matplotlib.image.AxesImage at 0x10db67dd8>
```

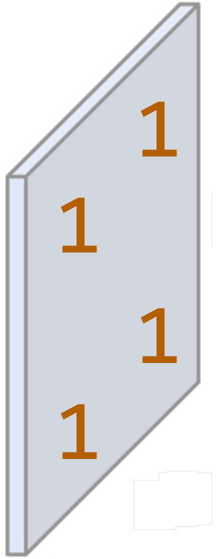


Toy image

Simple convolution layer

`Image: 1,3,3,1 image, Filter: 2,2,1,1, Stride: 1x1, Padding: VALID`





```
[[[[1.]],[[1.]]],
 [[[1.]],[[1.]]]]
shape=(2,2,1,1)
```

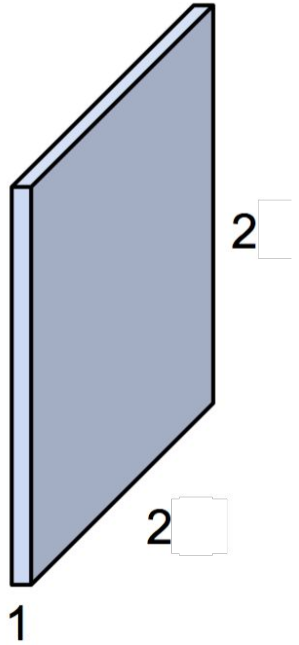
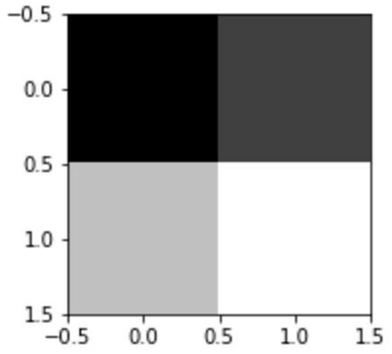
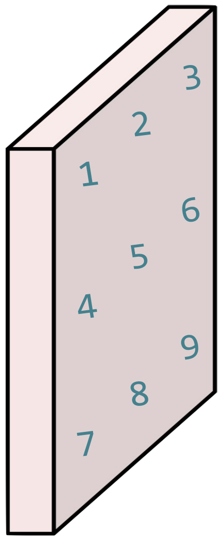


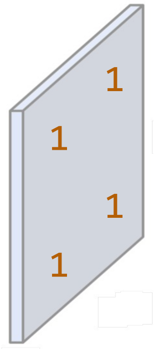
Image: 1,3,3,1 image, Filter: 2,2,1,1, Stride: 1x1, Padding: VALID

```
# print("imag:\n", image)
print("image.shape", image.shape)
weight = tf.constant([[[[1.]],[[1.]]],
                      [[[1.]],[[1.]]]])
print("weight.shape", weight.shape)
conv2d = tf.nn.conv2d(image, weight, strides=[1, 1, 1, 1], padding='VALID')
conv2d_img = conv2d.eval()
print("conv2d_img.shape", conv2d_img.shape)
conv2d_img = np.swapaxes(conv2d_img, 0, 3)
for i, one_img in enumerate(conv2d_img):
    print(one_img.reshape(2,2))
    plt.subplot(1,2,i+1), plt.imshow(one_img.reshape(2,2), cmap='gray')
```

```
image.shape (1, 3, 3, 1)
weight.shape (2, 2, 1, 1)
conv2d_img.shape (1, 2, 2, 1)
[[ 12.  16.]
 [ 24.  28.]]
```







Simple convolution layer

`Image: 1,3,3,1 image, Filter: 2,2,1,1, Stride: 1x1, Padding:` **SAME**

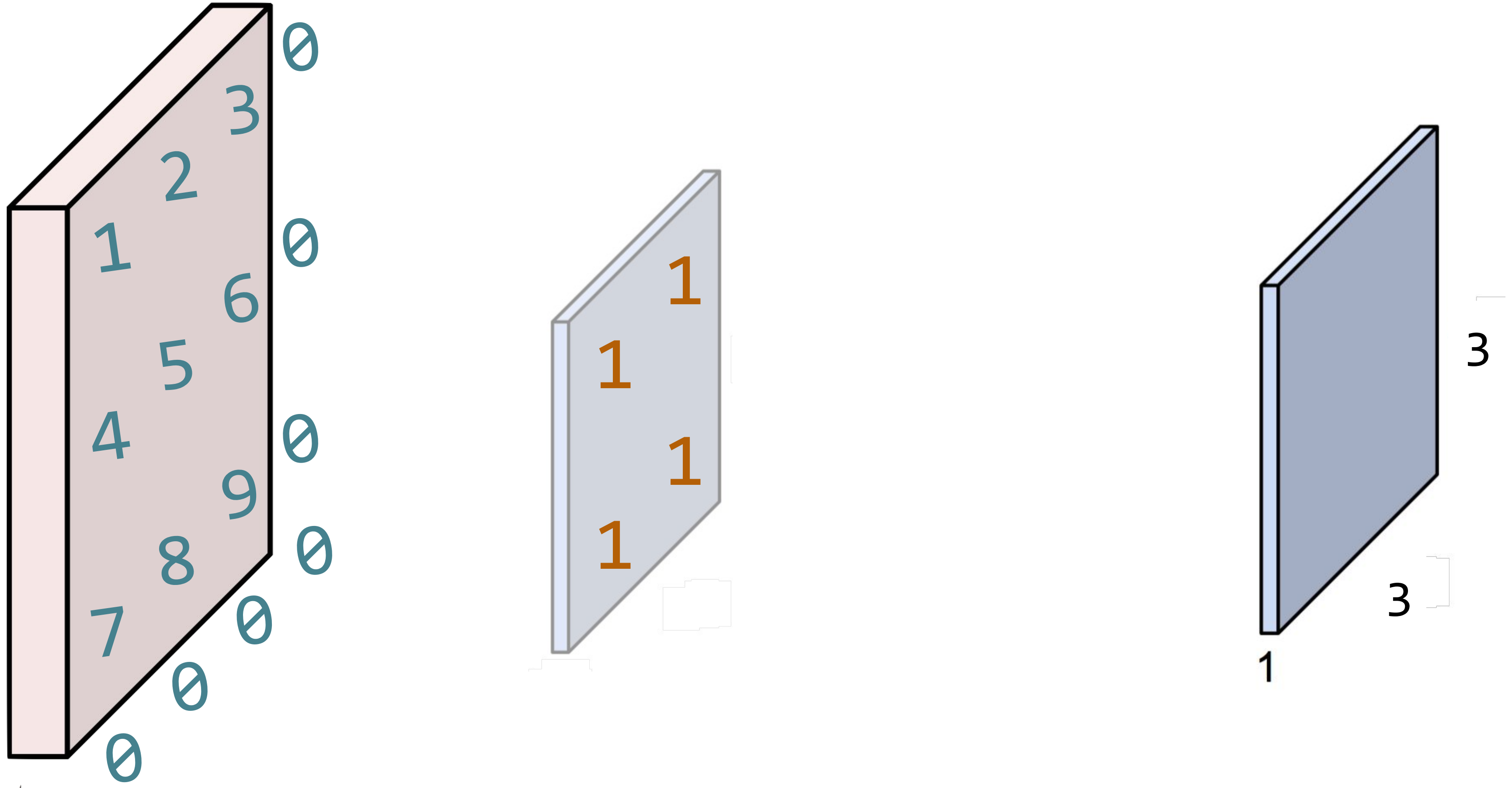
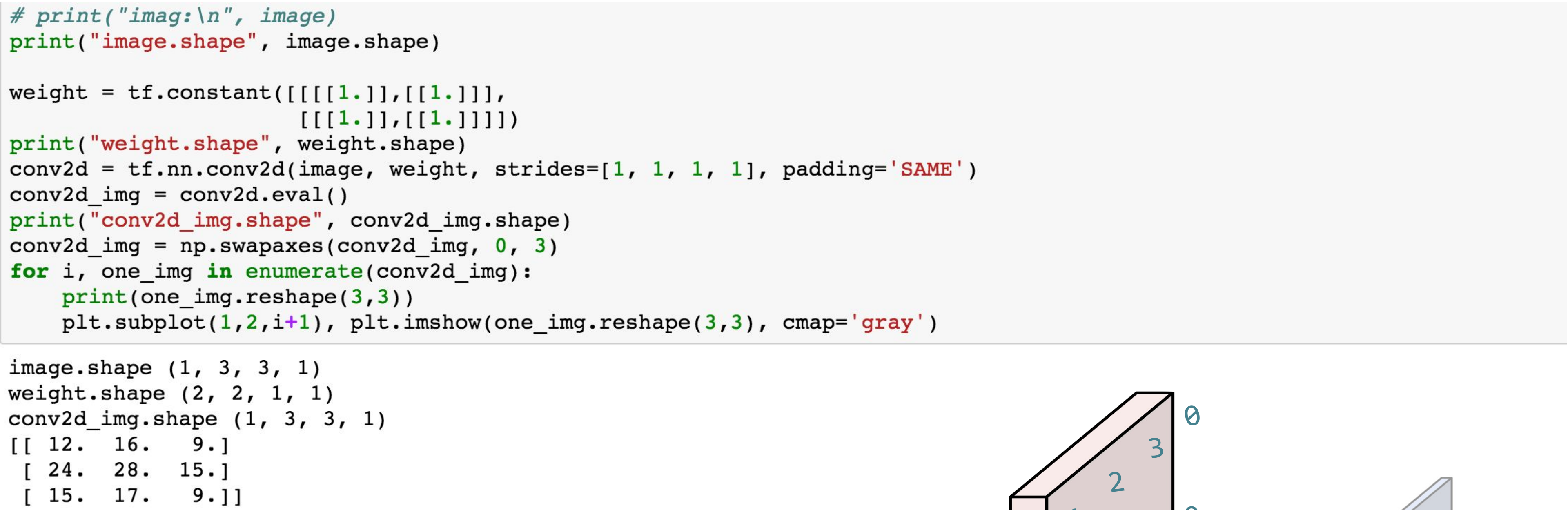
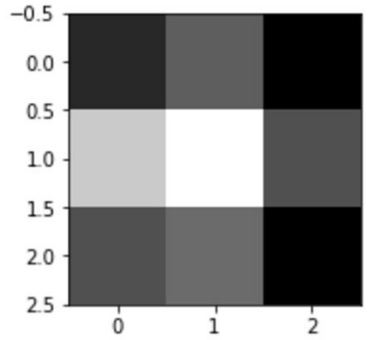


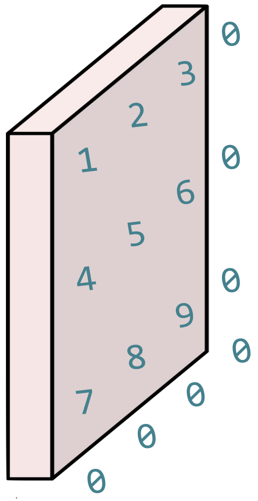
Image: 1,3,3,1 image, Filter: 2,2,1,1, Stride: 1x1, Padding: SAME

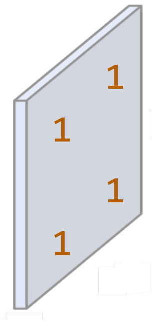


```
# print("imag:\n", image)
print("image.shape", image.shape)

weight = tf.constant([[[[1.]],[[1.]]],
                      [[[1.]],[[1.]]]])
print("weight.shape", weight.shape)
conv2d = tf.nn.conv2d(image, weight, strides=[1, 1, 1, 1], padding='SAME')
conv2d_img = conv2d.eval()
print("conv2d_img.shape", conv2d_img.shape)
conv2d_img = np.swapaxes(conv2d_img, 0, 3)
for i, one_img in enumerate(conv2d_img):
    print(one_img.reshape(3,3))
    plt.subplot(1,2,i+1), plt.imshow(one_img.reshape(3,3), cmap='gray')
```

```
image.shape (1, 3, 3, 1)
weight.shape (2, 2, 1, 1)
conv2d_img.shape (1, 3, 3, 1)
[[ 12.  16.   9.]
 [ 24.  28.  15.]
 [ 15.  17.   9.]]
```



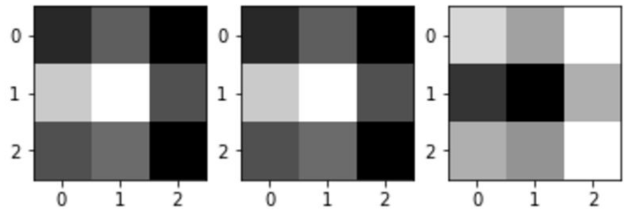


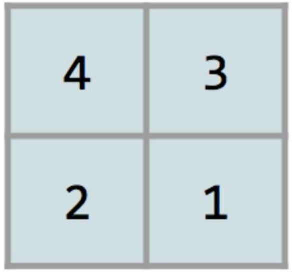


3 filters (2,2,1,3)

```python
# print("imag:\n", image)
print("image.shape", image.shape)

weight = tf.constant([[[[1.,10.,-1.]],[[1.,10.,-1.]]],
                      [[[1.,10.,-1.]],[[1.,10.,-1.]]]])
print("weight.shape", weight.shape)
conv2d = tf.nn.conv2d(image, weight, strides=[1, 1, 1, 1], padding='SAME')
conv2d_img = conv2d.eval()
print("conv2d_img.shape", conv2d_img.shape)
conv2d_img = np.swapaxes(conv2d_img, 0, 3)
for i, one_img in enumerate(conv2d_img):
    print(one_img.reshape(3,3))
    plt.subplot(1,3,i+1), plt.imshow(one_img.reshape(3,3), cmap='gray')
```

```
image.shape (1, 3, 3, 1)
weight.shape (2, 2, 1, 3)
conv2d_img.shape (1, 3, 3, 3)
[[ 12.  16.   9.]
 [ 24.  28.  15.]
 [ 15.  17.   9.]]
[[ 120.  160.   90.]
 [ 240.  280.  150.]
 [ 150.  170.   90.]]
[[-12. -16.  -9.]
 [-24. -28. -15.]
 [-15. -17.  -9.]]
```



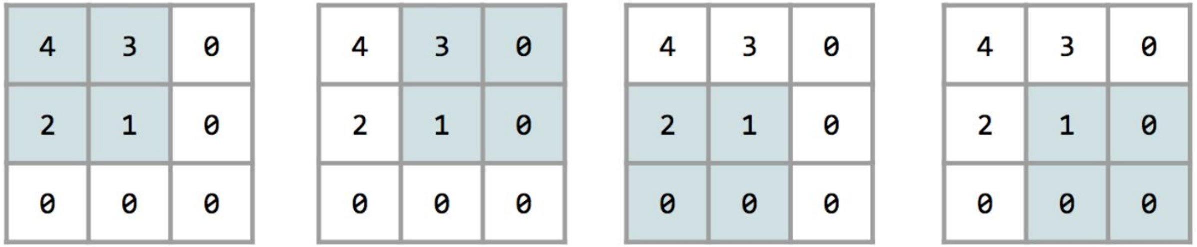


Max Pooling

```
In [19]: image = np.array([[[[4],[3]],
                             [[2],[1]]]], dtype=np.float32)
         pool = tf.nn.max_pool(image, ksize=[1, 2, 2, 1],
                             strides=[1, 1, 1, 1], padding='SAME')
         print(pool.shape)
         print(pool.eval())
```

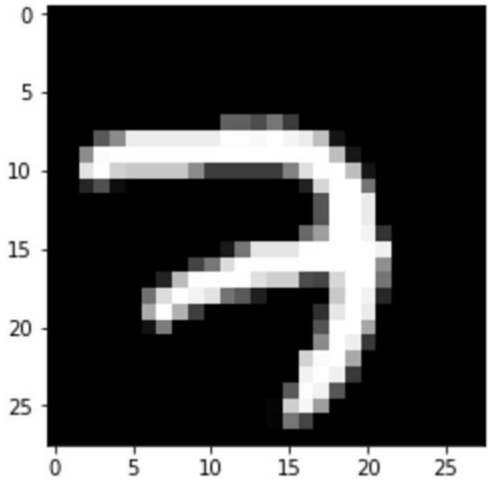
```
(1, 2, 2, 1)
[[[[ 4.]
   [ 3.]]

  [[ 2.]
   [ 1.]]]]
```

SAME: Zero paddings



https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-11-0-cnn_basics.ipynb

```
In [6]: from tensorflow.examples.tutorials.mnist import input_data
        mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
        # Check out https://www.tensorflow.org/get_started/mnist/beginners for
        # more information about the mnist dataset

        Extracting MNIST_data/train-images-idx3-ubyte.gz
        Extracting MNIST_data/train-labels-idx1-ubyte.gz
        Extracting MNIST_data/t10k-images-idx3-ubyte.gz
        Extracting MNIST_data/t10k-labels-idx1-ubyte.gz

In [7]: img = mnist.train.images[0].reshape(28,28)
        plt.imshow(img, cmap='gray')

Out[7]: <matplotlib.image.AxesImage at 0x115029ac8>
```



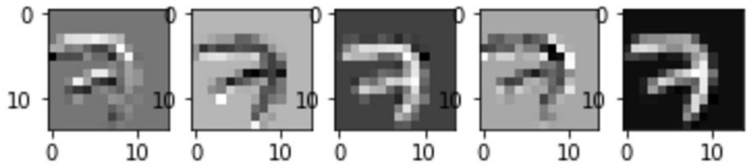
MNIST image loading

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-11-0-cnn_basics.ipynb

MNIST Convolution layer

```
In [8]: sess = tf.InteractiveSession()

        img = img.reshape(-1,28,28,1)
        W1 = tf.Variable(tf.random_normal([3, 3, 1, 5], stddev=0.01))
        conv2d = tf.nn.conv2d(img, W1, strides=[1, 2, 2, 1], padding='SAME')
        print(conv2d)
        sess.run(tf.global_variables_initializer())
        conv2d_img = conv2d.eval()
        conv2d_img = np.swapaxes(conv2d_img, 0, 3)
        for i, one_img in enumerate(conv2d_img):
            plt.subplot(1,5,i+1), plt.imshow(one_img.reshape(14,14), cmap='gray')
```

```
Tensor("Conv2D_1:0", shape=(1, 14, 14, 5), dtype=float32)
```

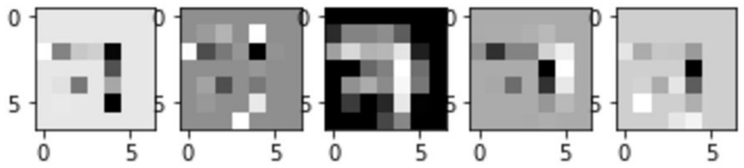


https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-11-0-cnn_basics.ipynb

MNIST Max pooling

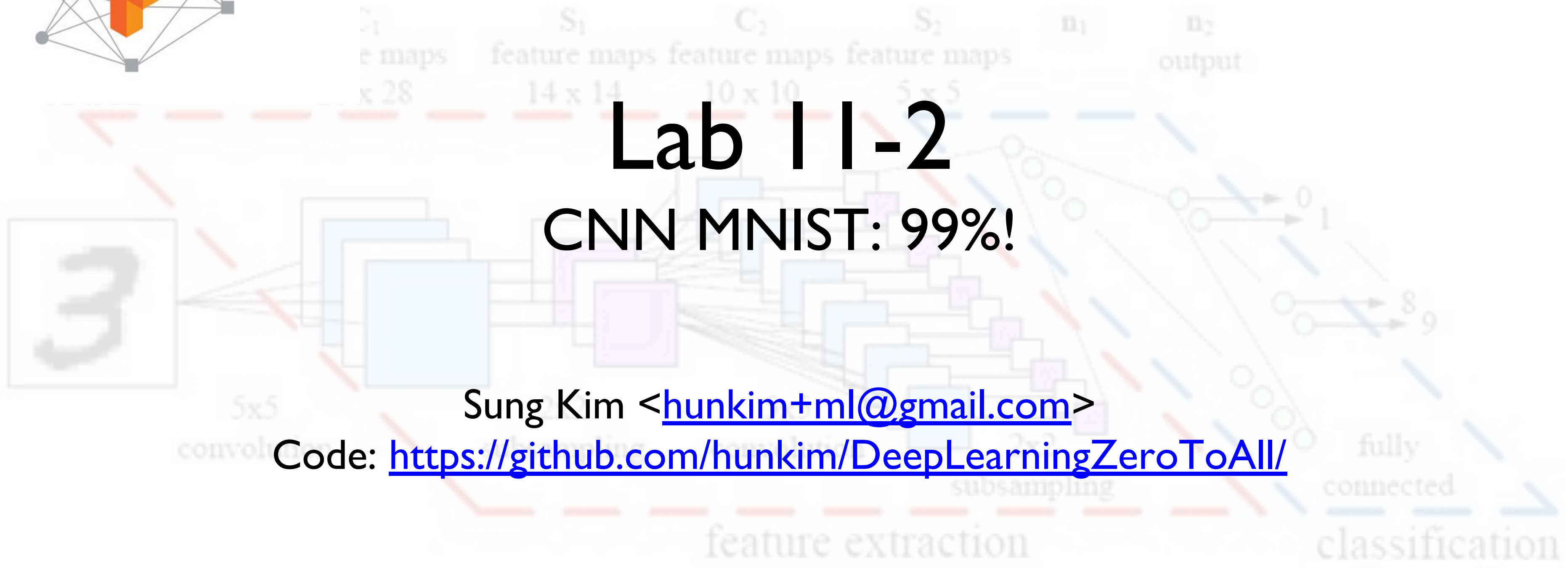
```
In [9]: pool = tf.nn.max_pool(conv2d, ksize=[1, 2, 2, 1], strides=[
                            1, 2, 2, 1], padding='SAME')
        print(pool)
        sess.run(tf.global_variables_initializer())
        pool_img = pool.eval()
        pool_img = np.swapaxes(pool_img, 0, 3)
        for i, one_img in enumerate(pool_img):
            plt.subplot(1,5,i+1), plt.imshow(one_img.reshape(7, 7), cmap='gray')
```

```
Tensor("MaxPool_2:0", shape=(1, 7, 7, 5), dtype=float32)
```



https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-11-0-cnn_basics.ipynb

With TF 1.0!

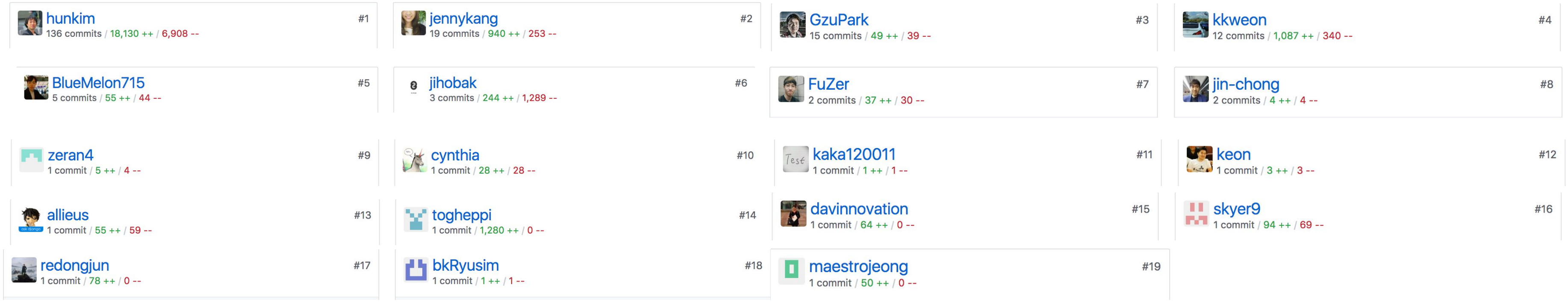


Lab 11-2

CNN MNIST: 99%!

Sung Kim <hunkim+ml@gmail.com>
Code: https://github.com/hunkim/DeepLearningZeroToAll/

https://github.com/hunkim/DeepLearningZeroToAll/



hunkim #1
136 commits / 18,130 ++ / 6,908 --

jennykang #2
19 commits / 940 ++ / 253 --

GzuPark #3
15 commits / 49 ++ / 39 --

kkweon #4
12 commits / 1,087 ++ / 340 --

BlueMelon715 #5
5 commits / 55 ++ / 44 --

jihobak #6
3 commits / 244 ++ / 1,289 --

FuZer #7
2 commits / 37 ++ / 30 --

jin-chong #8
2 commits / 4 ++ / 4 --

zeran4 #9
1 commit / 5 ++ / 4 --

cynthia #10
1 commit / 28 ++ / 28 --

kaka120011 #11
1 commit / 1 ++ / 1 --

keon #12
1 commit / 3 ++ / 3 --

allieus #13
1 commit / 55 ++ / 59 --

togheppi #14
1 commit / 1,280 ++ / 0 --

davinnovation #15
1 commit / 64 ++ / 0 --

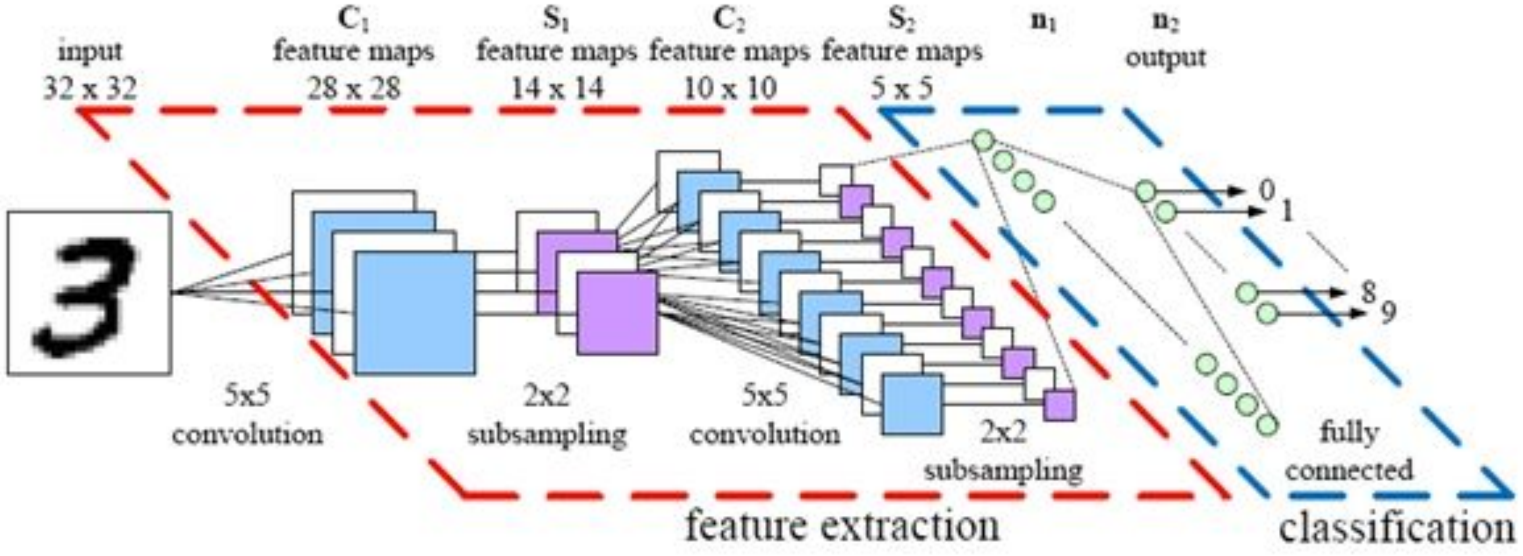
skyer9 #16
1 commit / 94 ++ / 69 --

redongjun #17
1 commit / 78 ++ / 0 --

bkRyusim #18
1 commit / 1 ++ / 1 --

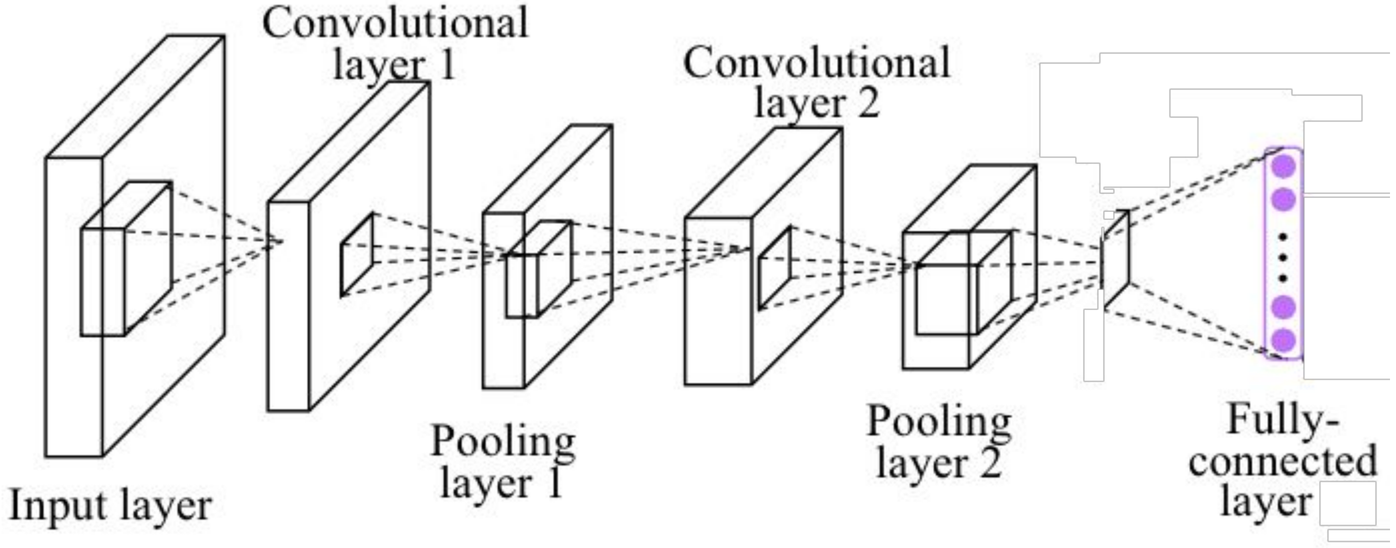
maestrojeong #19
1 commit / 50 ++ / 0 --

CNN

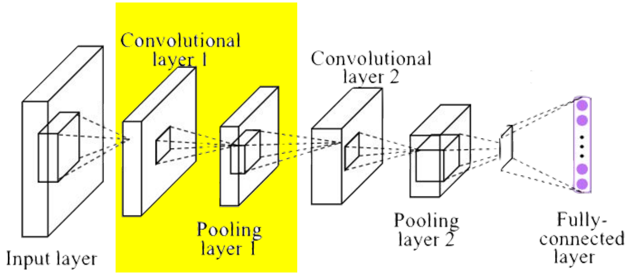


http://parse.ele.tue.nl/cluster/2/CNNArchitecture.jpg

Simple CNN


Convolutional layer 1
Convolutional layer 2
Input layer
Pooling layer 1
Pooling layer 2
Fully-connected layer

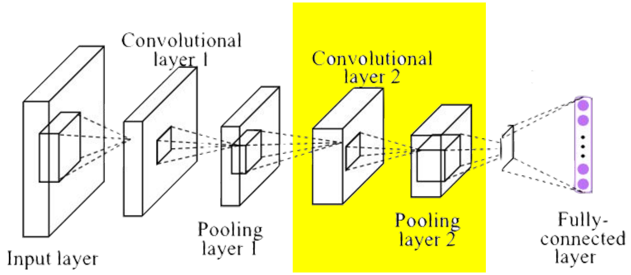
Conv layer 1



```
# input placeholders
X = tf.placeholder(tf.float32, [None, 784])
X_img = tf.reshape(X, [-1, 28, 28, 1])   # img 28x28x1 (black/white)
Y = tf.placeholder(tf.float32, [None, 10])

# L1 ImgIn shape=(?, 28, 28, 1)
W1 = tf.Variable(tf.random_normal([3, 3, 1, 32], stddev=0.01))
#    Conv     -> (?, 28, 28, 32)
#    Pool     -> (?, 14, 14, 32)
L1 = tf.nn.conv2d(X_img, W1, strides=[1, 1, 1, 1], padding='SAME')
L1 = tf.nn.relu(L1)
L1 = tf.nn.max_pool(L1, ksize=[1, 2, 2, 1],
          strides=[1, 2, 2, 1], padding='SAME')
'''
Tensor("Conv2D:0", shape=(?, 28, 28, 32), dtype=float32)
Tensor("Relu:0", shape=(?, 28, 28, 32), dtype=float32)
Tensor("MaxPool:0", shape=(?, 14, 14, 32), dtype=float32)
'''
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-11-1-mnist_cnn.py

Conv layer 2

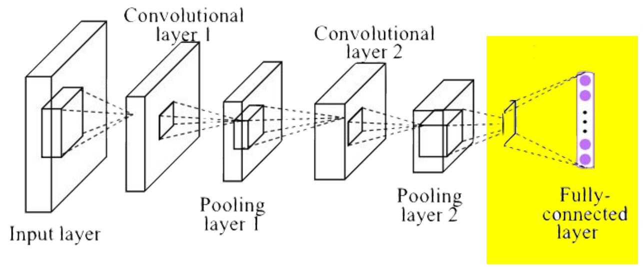


```
'''
Tensor("Conv2D:0", shape=(?, 28, 28, 32), dtype=float32)
Tensor("Relu:0", shape=(?, 28, 28, 32), dtype=float32)
Tensor("MaxPool:0", shape=(?, 14, 14, 32), dtype=float32)
'''

# L2 ImgIn shape=(?, 14, 14, 32)
W2 = tf.Variable(tf.random_normal([3, 3, 32, 64], stddev=0.01))
#    Conv      ->(?, 14, 14, 64)
#    Pool      ->(?, 7, 7, 64)
L2 = tf.nn.conv2d(L1, W2, strides=[1, 1, 1, 1], padding='SAME')
L2 = tf.nn.relu(L2)
L2 = tf.nn.max_pool(L2, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
L2 = tf.reshape(L2, [-1, 7 * 7 * 64])
'''
Tensor("Conv2D_1:0", shape=(?, 14, 14, 64), dtype=float32)
Tensor("Relu_1:0", shape=(?, 14, 14, 64), dtype=float32)
Tensor("MaxPool_1:0", shape=(?, 7, 7, 64), dtype=float32)
Tensor("Reshape_1:0", shape=(?, 3136), dtype=float32)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-11-1-mnist_cnn.py

Fully Connected (FC, Dense) layer

```
'''
Tensor("Conv2D_1:0", shape=(?, 14, 14, 64), dtype=float32)
Tensor("Relu_1:0", shape=(?, 14, 14, 64), dtype=float32)
Tensor("MaxPool_1:0", shape=(?, 7, 7, 64), dtype=float32)
Tensor("Reshape_1:0", shape=(?, 3136), dtype=float32)
'''
L2 = tf.reshape(L2, [-1, 7 * 7 * 64])

# Final FC 7x7x64 inputs -> 10 outputs
W3 = tf.get_variable("W3", shape=[7 * 7 * 64, 10],
initializer=tf.contrib.layers.xavier_initializer())
b = tf.Variable(tf.random_normal([10]))
hypothesis = tf.matmul(L2, W3) + b

# define cost/loss & optimizer
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=hypothesis, labels=Y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)
```



https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-11-1-mnist_cnn.py

Training and Evaluation

```
# initialize
sess = tf.Session()
sess.run(tf.global_variables_initializer())

# train my model
print('Learning stared. It takes sometime.')
for epoch in range(training_epochs):
    avg_cost = 0
    total_batch = int(mnist.train.num_examples / batch_size)
    for i in range(total_batch):
        batch_xs, batch_ys = mnist.train.next_batch(batch_size)
        feed_dict = {X: batch_xs, Y: batch_ys}
        c, _, = sess.run([cost, optimizer], feed_dict=feed_dict)
        avg_cost += c / total_batch
    print('Epoch:', '%04d' % (epoch + 1), 'cost =', '{:.9f}'.format(avg_cost))

print('Learning Finished!')

# Test model and check accuracy
correct_prediction = tf.equal(tf.argmax(hypothesis, 1), tf.argmax(Y, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
print('Accuracy:', sess.run(accuracy, feed_dict={X: mnist.test.images, Y: mnist.test.labels}))
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-11-1-mnist_cnn.py

Training and Evaluation

```python
# initialize
sess = tf.Session()
sess.run(tf.global_variables_initializer())

# train my model
print('Learning stared. It takes sometime.')
for epoch in range(training_epochs):
   avg_cost = 0
   total_batch = int(mnist.train.num_examples / batch_size)
   for i in range(total_batch):
       batch_xs, batch_ys = mnist.train.next_batch(batch_size)
       feed_dict = {X: batch_xs, Y: batch_ys}
       c, _, = sess.run([cost, optimizer], feed_dict=feed_dict)
       avg_cost += c / total_batch
   print('Epoch:', '%04d' % (epoch + 1), 'cost =', '{:.9f}'.format(avg_cost))

print('Learning Finished!')

# Test model and check accuracy
correct_prediction = tf.equal(tf.argmax(hypothesis, 1), tf.argmax(Y, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
print('Accuracy:', sess.run(accuracy, feed_dict={X: mnist.test.images, Y: mnist.test.labels}))
```

```
Epoch: 0001 cost = 0.340291267
Epoch: 0002 cost = 0.090731326
Epoch: 0003 cost = 0.064477619
Epoch: 0004 cost = 0.050683064
...
Epoch: 0011 cost = 0.017758641
Epoch: 0012 cost = 0.014156652
Epoch: 0013 cost = 0.012397016
Epoch: 0014 cost = 0.010693789
Epoch: 0015 cost = 0.009469977
Learning Finished!
Accuracy: 0.9885
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-11-1-mnist_cnn.py

Deep CNN

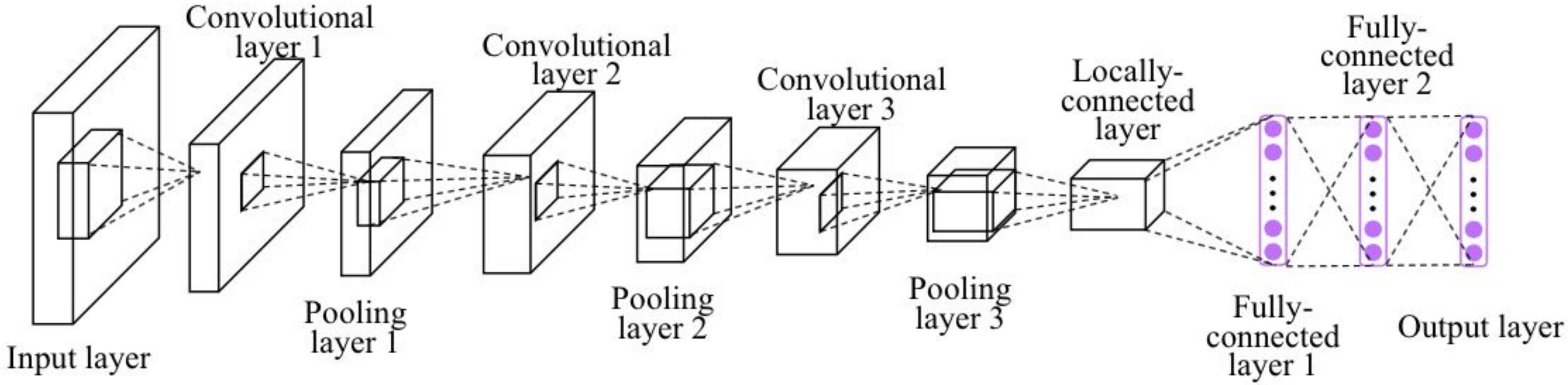


Image credit: http://personal.ie.cuhk.edu.hk/~ccloy/project_target_code/index.html

Deep CNN

```
# L1 ImgIn shape=(?, 28, 28, 1)
W1 = tf.Variable(tf.random_normal([3, 3, 1, 32], stddev=0.01))
#    Conv     -> (?, 28, 28, 32)
#    Pool     -> (?, 14, 14, 32)
L1 = tf.nn.conv2d(X_img, W1, strides=[1, 1, 1, 1], padding='SAME')
L1 = tf.nn.relu(L1)
L1 = tf.nn.max_pool(L1, ksize=[1, 2, 2, 1],
                    strides=[1, 2, 2, 1], padding='SAME')
L1 = tf.nn.dropout(L1, keep_prob=keep_prob)
'''Tensor("Conv2D:0", shape=(?, 28, 28, 32), dtype=float32)
   Tensor("Relu:0", shape=(?, 28, 28, 32), dtype=float32)
   Tensor("MaxPool:0", shape=(?, 14, 14, 32), dtype=float32)
   Tensor("dropout/mul:0", shape=(?, 14, 14, 32), dtype=float32)'''

# L2 ImgIn shape=(?, 14, 14, 32)
W2 = tf.Variable(tf.random_normal([3, 3, 32, 64], stddev=0.01))
#    Conv      ->(?, 14, 14, 64)
#    Pool      ->(?, 7, 7, 64)
L2 = tf.nn.conv2d(L1, W2, strides=[1, 1, 1, 1], padding='SAME')
L2 = tf.nn.relu(L2)
L2 = tf.nn.max_pool(L2, ksize=[1, 2, 2, 1],
                    strides=[1, 2, 2, 1], padding='SAME')
L2 = tf.nn.dropout(L2, keep_prob=keep_prob)
'''Tensor("Conv2D_1:0", shape=(?, 14, 14, 64), dtype=float32)
   Tensor("Relu_1:0", shape=(?, 14, 14, 64), dtype=float32)
   Tensor("MaxPool_1:0", shape=(?, 7, 7, 64), dtype=float32)
   Tensor("dropout_1/mul:0", shape=(?, 7, 7, 64), dtype=float32)'''
```

```
# L3 ImgIn shape=(?, 7, 7, 64)
W3 = tf.Variable(tf.random_normal([3, 3, 64, 128], stddev=0.01))
#    Conv      ->(?, 7, 7, 128)
#    Pool      ->(?, 4, 4, 128)
#    Reshape   ->(?, 4 * 4 * 128) # Flatten them for FC
L3 = tf.nn.conv2d(L2, W3, strides=[1, 1, 1, 1], padding='SAME')
L3 = tf.nn.relu(L3)
L3 = tf.nn.max_pool(L3, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1],
padding='SAME')
L3 = tf.nn.dropout(L3, keep_prob=keep_prob)
L3 = tf.reshape(L3, [-1, 128 * 4 * 4])
'''Tensor("Conv2D_2:0", shape=(?, 7, 7, 128), dtype=float32)
   Tensor("Relu_2:0", shape=(?, 7, 7, 128), dtype=float32)
   Tensor("MaxPool_2:0", shape=(?, 4, 4, 128), dtype=float32)
   Tensor("dropout_2/mul:0", shape=(?, 4, 4, 128), dtype=float32)
   Tensor("Reshape_1:0", shape=(?, 2048), dtype=float32)'''
# L4 FC 4x4x128 inputs -> 625 outputs
W4 = tf.get_variable("W4", shape=[128 * 4 * 4, 625],
initializer=tf.contrib.layers.xavier_initializer())
b4 = tf.Variable(tf.random_normal([625]))
L4 = tf.nn.relu(tf.matmul(L3, W4) + b4)
L4 = tf.nn.dropout(L4, keep_prob=keep_prob)
'''Tensor("Relu_3:0", shape=(?, 625), dtype=float32)
   Tensor("dropout_3/mul:0", shape=(?, 625), dtype=float32)'''
# L5 Final FC 625 inputs -> 10 outputs
W5 = tf.get_variable("W5", shape=[625, 10],
initializer=tf.contrib.layers.xavier_initializer())
b5 = tf.Variable(tf.random_normal([10]))
hypothesis = tf.matmul(L4, W5) + b5
'''Tensor("add_1:0", shape=(?, 10), dtype=float32)'''
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-11-2-mnist_deep_cnn.py

Deep CNN

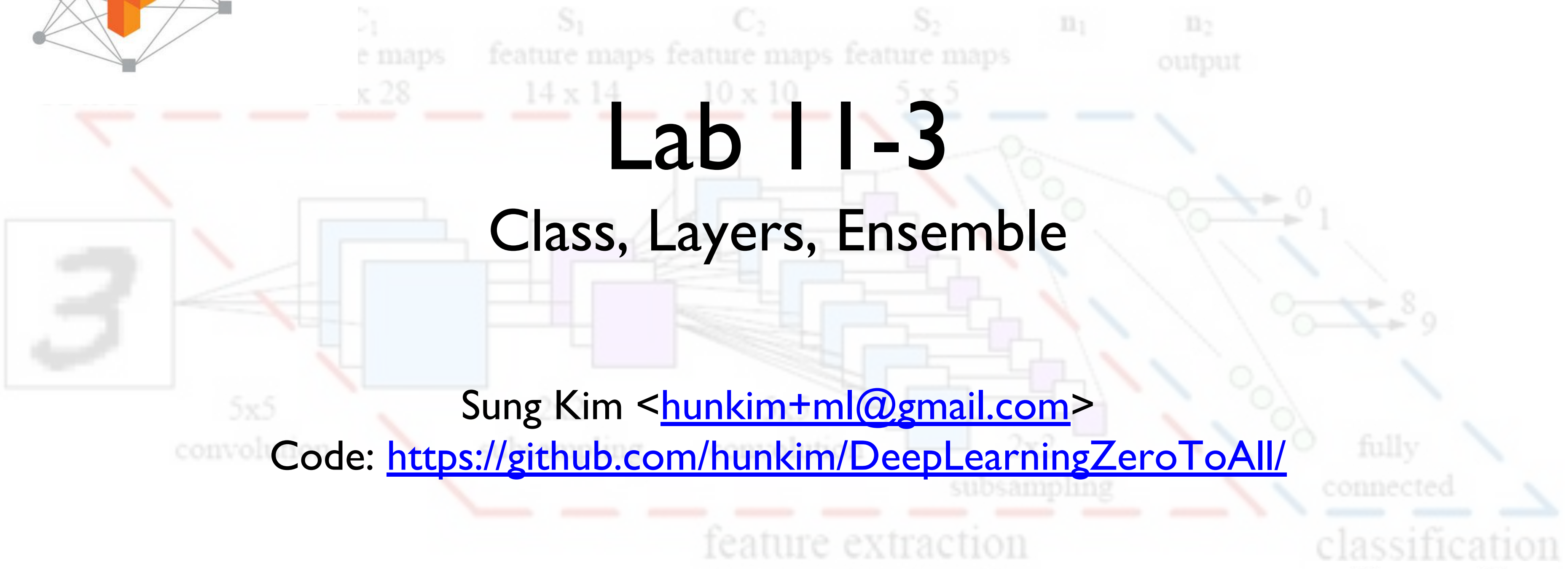
```
# L1 ImgIn shape=(?, 28, 28, 1)
W1 = tf.Variable(tf.random_normal([3, 3, 1, 32], stddev=0.01))
#    Conv     -> (?, 28, 28, 32)
#    Pool     -> (?, 14, 14, 32)
L1 = tf.nn.conv2d(X_img, W1, strides=[1, 1, 1, 1], padding='SAME')
L1 = tf.nn.relu(L1)
L1 = tf.nn.max_pool(L1, ksize=[1, 2, 2, 1],
                    strides=[1, 2, 2, 1], padding='SAME')
L1 = tf.nn.dropout(L1, keep_prob=keep_prob)
'''Tensor("Conv2D:0", shape=(?, 28, 28, 32), dtype=float32)
   Tensor("Relu:0", shape=(?, 28, 28, 32), dtype=float32)
   Tensor("MaxPool:0", shape=(?, 14, 14, 32), dtype=float32)
   Tensor("dropout/mul:0", shape=(?, 14, 14, 32), dtype=float32)'''
...
...
# L4 FC 4x4x128 inputs -> 625 outputs
W4 = tf.get_variable("W4", shape=[128 * 4 * 4, 625],
initializer=tf.contrib.layers.xavier_initializer())
b4 = tf.Variable(tf.random_normal([625]))
L4 = tf.nn.relu(tf.matmul(L3, W4) + b4)
L4 = tf.nn.dropout(L4, keep_prob=keep_prob)
'''Tensor("Relu_3:0", shape=(?, 625), dtype=float32)
   Tensor("dropout_3/mul:0", shape=(?, 625), dtype=float32)'''
# L5 Final FC 625 inputs -> 10 outputs
W5 = tf.get_variable("W5", shape=[625, 10],
initializer=tf.contrib.layers.xavier_initializer())
b5 = tf.Variable(tf.random_normal([10]))
hypothesis = tf.matmul(L4, W5) + b5
'''Tensor("add_1:0", shape=(?, 10), dtype=float32)'''
```

```
# Test model and check accuracy
correct_prediction = tf.equal(tf.argmax(hypothesis, 1),
          tf.argmax(Y, 1))
accuracy =
tf.reduce_mean(tf.cast(correct_prediction,tf.float32))
print('Accuracy:', sess.run(accuracy,
feed_dict={X: mnist.test.images,
           Y: mnist.test.labels, keep_prob: 1}))
```

```
Epoch: 0013 cost = 0.027188021
Epoch: 0014 cost = 0.023604777
Epoch: 0015 cost = 0.024607201
Learning Finished!
Accuracy: 0.9938
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-11-2-mnist_deep_cnn.py

With TF 1.0!

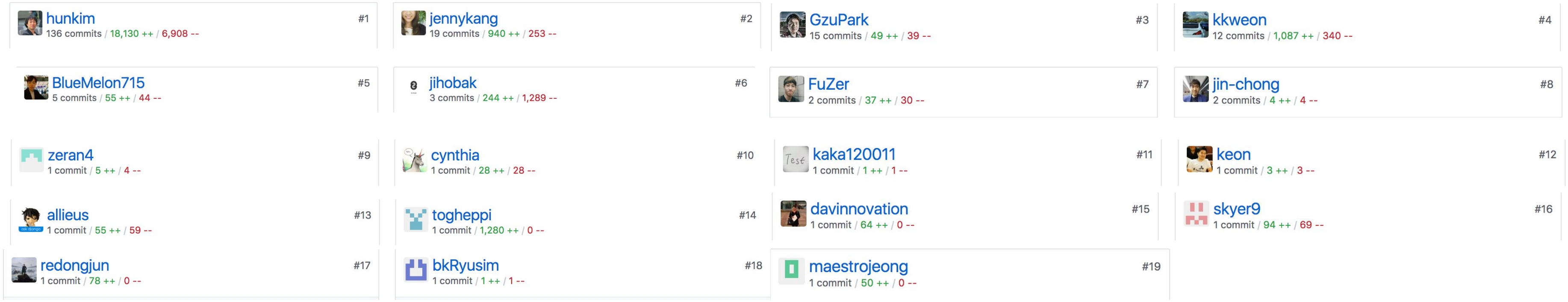


Lab 11-3

Class, Layers, Ensemble

Sung Kim <hunkim+ml@gmail.com>
Code: https://github.com/hunkim/DeepLearningZeroToAll/

https://github.com/hunkim/DeepLearningZeroToAll/



hunkim #1
136 commits / 18,130 ++ / 6,908 --

jennykang #2
19 commits / 940 ++ / 253 --

GzuPark #3
15 commits / 49 ++ / 39 --

kkweon #4
12 commits / 1,087 ++ / 340 --

BlueMelon715 #5
5 commits / 55 ++ / 44 --

jihobak #6
3 commits / 244 ++ / 1,289 --

FuZer #7
2 commits / 37 ++ / 30 --

jin-chong #8
2 commits / 4 ++ / 4 --

zeran4 #9
1 commit / 5 ++ / 4 --

cynthia #10
1 commit / 28 ++ / 28 --

kaka120011 #11
1 commit / 1 ++ / 1 --

keon #12
1 commit / 3 ++ / 3 --

allieus #13
1 commit / 55 ++ / 59 --

togheppi #14
1 commit / 1,280 ++ / 0 --

davinnovation #15
1 commit / 64 ++ / 0 --

skyer9 #16
1 commit / 94 ++ / 69 --

redongjun #17
1 commit / 78 ++ / 0 --

bkRyusim #18
1 commit / 1 ++ / 1 --

maestrojeong #19
1 commit / 50 ++ / 0 --

CNN

```python
# L1 ImgIn shape=(?, 28, 28, 1)
W1 = tf.Variable(tf.random_normal([3, 3, 1, 32], stddev=0.01))
#    Conv     -> (?, 28, 28, 32)
#    Pool     -> (?, 14, 14, 32)
L1 = tf.nn.conv2d(X_img, W1, strides=[1, 1, 1, 1], padding='SAME')
L1 = tf.nn.relu(L1)
L1 = tf.nn.max_pool(L1, ksize=[1, 2, 2, 1],
                    strides=[1, 2, 2, 1], padding='SAME')
L1 = tf.nn.dropout(L1, keep_prob=keep_prob)
'''Tensor("Conv2D:0", shape=(?, 28, 28, 32), dtype=float32)
   Tensor("Relu:0", shape=(?, 28, 28, 32), dtype=float32)
   Tensor("MaxPool:0", shape=(?, 14, 14, 32), dtype=float32)
   Tensor("dropout/mul:0", shape=(?, 14, 14, 32), dtype=float32)'''
...
...
# L4 FC 4x4x128 inputs -> 625 outputs
W4 = tf.get_variable("W4", shape=[128 * 4 * 4, 625],
initializer=tf.contrib.layers.xavier_initializer())
b4 = tf.Variable(tf.random_normal([625]))
L4 = tf.nn.relu(tf.matmul(L3, W4) + b4)
L4 = tf.nn.dropout(L4, keep_prob=keep_prob)
'''Tensor("Relu_3:0", shape=(?, 625), dtype=float32)
   Tensor("dropout_3/mul:0", shape=(?, 625), dtype=float32)'''
# L5 Final FC 625 inputs -> 10 outputs
W5 = tf.get_variable("W5", shape=[625, 10],
initializer=tf.contrib.layers.xavier_initializer())
b5 = tf.Variable(tf.random_normal([10]))
hypothesis = tf.matmul(L4, W5) + b5
'''Tensor("add_1:0", shape=(?, 10), dtype=float32)'''
```

```python
# Test model and check accuracy
correct_prediction = tf.equal(tf.argmax(hypothesis, 1),
          tf.argmax(Y, 1))
accuracy =
tf.reduce_mean(tf.cast(correct_prediction,tf.float32))
print('Accuracy:', sess.run(accuracy,
feed_dict={X: mnist.test.images,
           Y: mnist.test.labels, keep_prob: 1}))
```

```
Epoch: 0013 cost = 0.027188021
Epoch: 0014 cost = 0.023604777
Epoch: 0015 cost = 0.024607201
Learning Finished!
Accuracy: 0.9938
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-11-2-mnist_deep_cnn.py

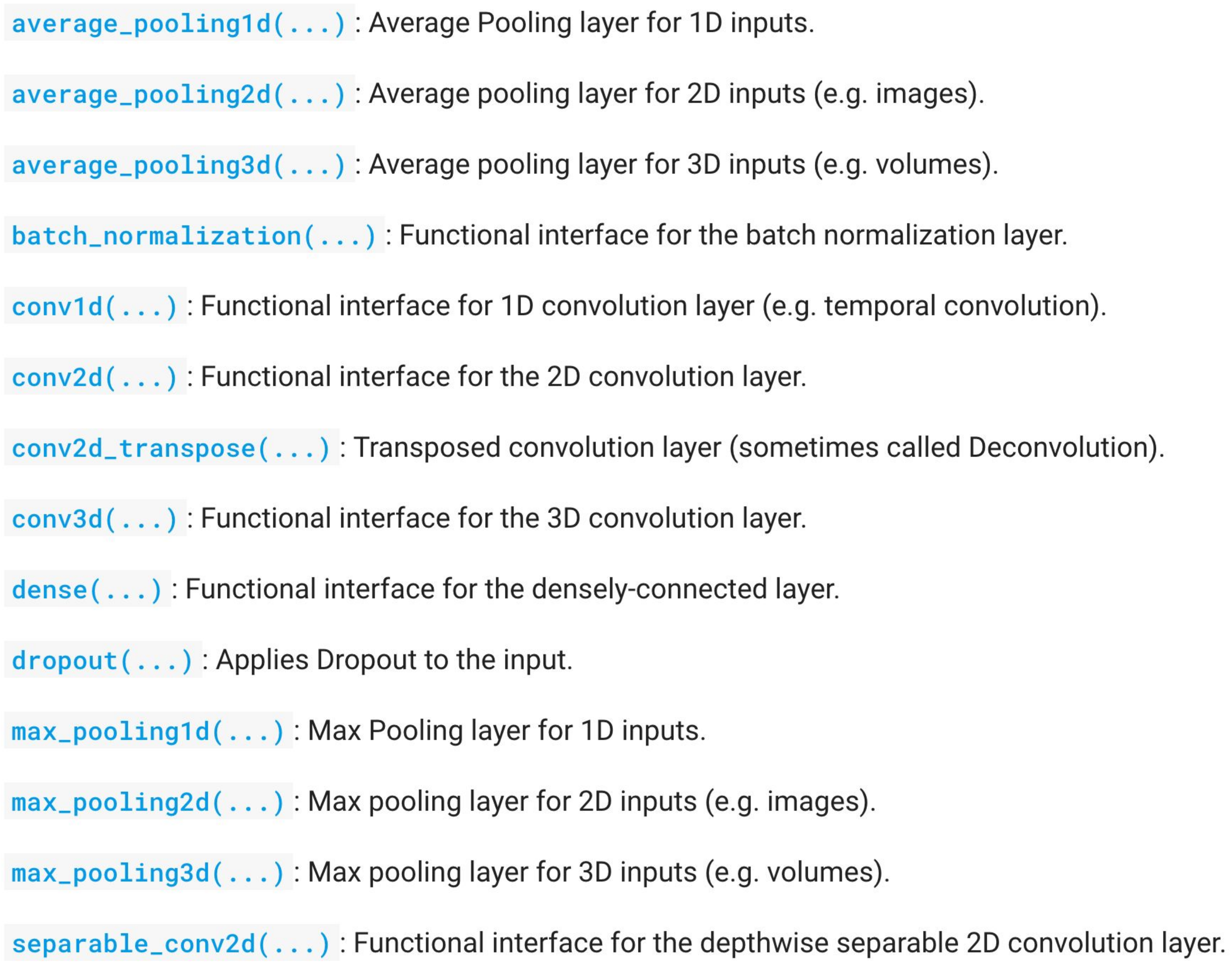
Python Class

```python
class Model:

   def __init__(self, sess, name):
       self.sess = sess
       self.name = name
       self._build_net()

   def _build_net(self):
       with tf.variable_scope(self.name):
           # input place holders
           self.X = tf.placeholder(tf.float32, [None, 784])
           # img 28x28x1 (black/white)
           X_img = tf.reshape(self.X, [-1, 28, 28, 1])
           self.Y = tf.placeholder(tf.float32, [None, 10])

           # L1 ImgIn shape=(?, 28, 28, 1)
           W1 = tf.Variable(tf.random_normal([3, 3, 1, 32],
                                                       stddev=0.01))

           ...
def predict(self, x_test, keep_prop=1.0):
   return self.sess.run(self.logits,
        feed_dict={self.X: x_test, self.keep_prob: keep_prop})

def get_accuracy(self, x_test, y_test, keep_prop=1.0):
   return self.sess.run(self.accuracy,
    feed_dict={self.X: x_test, self.Y: y_test, self.keep_prob: keep_prop})

def train(self, x_data, y_data, keep_prop=0.7):
   return self.sess.run([self.cost, self.optimizer], feed_dict={
       self.X: x_data, self.Y: y_data, self.keep_prob: keep_prop})
```

```python
# initialize
sess = tf.Session()
m1 = Model(sess, "m1")

sess.run(tf.global_variables_initializer())

print('Learning Started!')

# train my model
for epoch in range(training_epochs):
    avg_cost = 0
    total_batch = int(mnist.train.num_examples / batch_size)

    for i in range(total_batch):
        batch_xs, batch_ys = mnist.train.next_batch(batch_size)
        c, _ = m1.train(batch_xs, batch_ys)
        avg_cost += c / total_batc
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-11-3-mnist_cnn_class.py

tf.layers



`average_pooling1d(...)` : Average Pooling layer for 1D inputs.

`average_pooling2d(...)` : Average pooling layer for 2D inputs (e.g. images).

`average_pooling3d(...)` : Average pooling layer for 3D inputs (e.g. volumes).

`batch_normalization(...)` : Functional interface for the batch normalization layer.

`conv1d(...)` : Functional interface for 1D convolution layer (e.g. temporal convolution).

`conv2d(...)` : Functional interface for the 2D convolution layer.

`conv2d_transpose(...)` : Transposed convolution layer (sometimes called Deconvolution).

`conv3d(...)` : Functional interface for the 3D convolution layer.

`dense(...)` : Functional interface for the densely-connected layer.

`dropout(...)` : Applies Dropout to the input.

`max_pooling1d(...)` : Max Pooling layer for 1D inputs.

`max_pooling2d(...)` : Max pooling layer for 2D inputs (e.g. images).

`max_pooling3d(...)` : Max pooling layer for 3D inputs (e.g. volumes).

`separable_conv2d(...)` : Functional interface for the depthwise separable 2D convolution layer.

https://www.tensorflow.org/api_docs/python/tf/layers

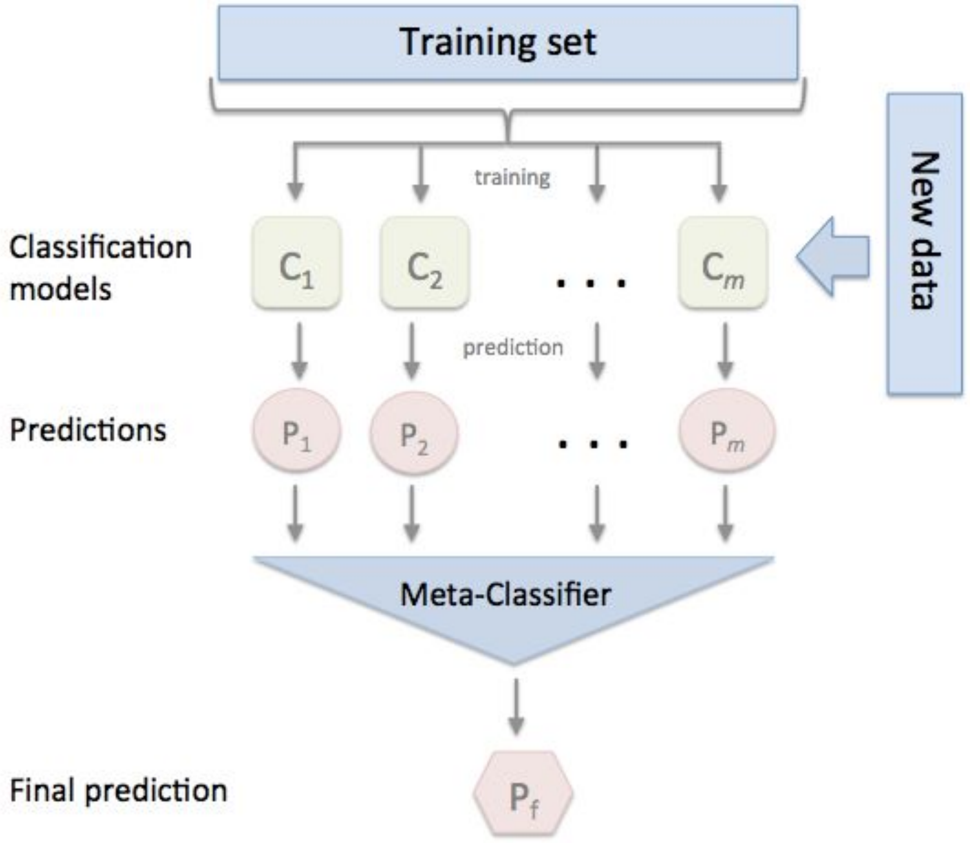
tf.layers

```
# L1 ImgIn shape=(?, 28, 28, 1)
W1 = tf.Variable(tf.random_normal([3, 3, 1, 32], stddev=0.01))
#    Conv     -> (?, 28, 28, 32)
#    Pool     -> (?, 14, 14, 32)
L1 = tf.nn.conv2d(X_img, W1, strides=[1, 1, 1, 1], padding='SAME')
L1 = tf.nn.relu(L1)
L1 = tf.nn.max_pool(L1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')
L1 = tf.nn.dropout(L1, keep_prob=self.keep_prob)
…
# L2 ImgIn shape=(?, 14, 14, 32)
W2 = tf.Variable(tf.random_normal([3, 3, 32, 64], stddev=0.01))
```

```
# Convolutional Layer #1
conv1 = tf.layers.conv2d(inputs=X_img,filters=32,kernel_size=[3,3],padding="SAME",activation=tf.nn.relu)
pool1 = tf.layers.max_pooling2d(inputs=conv1, pool_size=[2, 2], padding="SAME", strides=2)
dropout1 = tf.layers.dropout(inputs=pool1,rate=0.7, training=self.training)

# Convolutional Layer #2
conv2 = tf.layers.conv2d(inputs=dropout1,filters=64,kernel_size=[3,3],padding="SAME",activation=tf.nn.relu)
…
flat = tf.reshape(dropout3, [-1, 128 * 4 * 4])
dense4 = tf.layers.dense(inputs=flat, units=625, activation=tf.nn.relu)
dropout4 = tf.layers.dropout(inputs=dense4, rate=0.5, training=self.training)
...
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-11-4-mnist_cnn_layers.py

Ensemble



http://rasbt.github.io/mlxtend/user_guide/classifier/StackingClassifier/

Ensemble training

```
class Model:

  def __init__(self, sess, name):
     self.sess = sess
     self.name = name
     self._build_net()

  def _build_net(self):
     with tf.variable_scope(self.name):

 ...
```

```
models = []
num_models = 7
for m in range(num_models):
   models.append(Model(sess, "model" + str(m)))

sess.run(tf.global_variables_initializer())
print('Learning Started!')

# train my model
for epoch in range(training_epochs):
   avg_cost_list = np.zeros(len(models))
   total_batch = int(mnist.train.num_examples / batch_size)
   for i in range(total_batch):
       batch_xs, batch_ys =mnist.train.next_batch(batch_size)

       # train each model
       for m_idx, m in enumerate(models):
           c, _ = m.train(batch_xs, batch_ys)
           avg_cost_list[m_idx] += c / total_batch

   print('Epoch:','%04d'%(epoch + 1),'cost =', avg_cost_list)

print('Learning Finished!')
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-11-5-mnist_cnn_ensemble_layers.py

Ensemble prediction

Ensemble prediction

	0	1	2	3	4	5	6	7	8	9
C_1	0.1	0.01	0.02	0.8	...	...	...	...	...	...
C_2	0.01	0.5	0.02	0.4	...	...	...	...	...	...
. . .										
C_m	0.01	0.01	0.1	0.7	...	...	...	...	...	...
Sum	0.12	0.52	0.14	**1.9**	...	...	...	...	...	...

argmax

Ensemble prediction

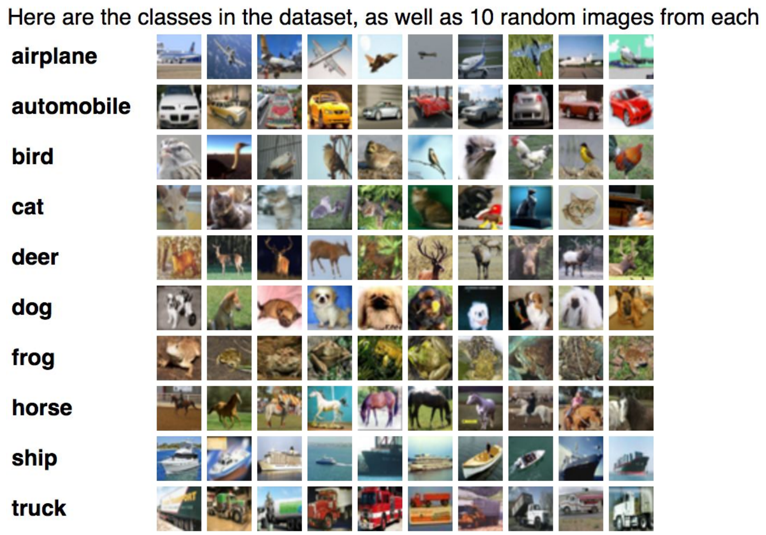
```python
# Test model and check accuracy
test_size = len(mnist.test.labels)
predictions = np.zeros(test_size * 10).reshape(test_size, 10)


for m_idx, m in enumerate(models):
   print(m_idx, 'Accuracy:', m.get_accuracy(mnist.test.images, mnist.test.labels))
   p = m.predict(mnist.test.images)
   predictions += p

ensemble_correct_prediction = tf.equal(
    tf.argmax(predictions, 1), tf.argmax(mnist.test.labels, 1))
ensemble_accuracy = tf.reduce_mean(
   tf.cast(ensemble_correct_prediction, tf.float32))
print('Ensemble accuracy:', sess.run(ensemble_accuracy))
```

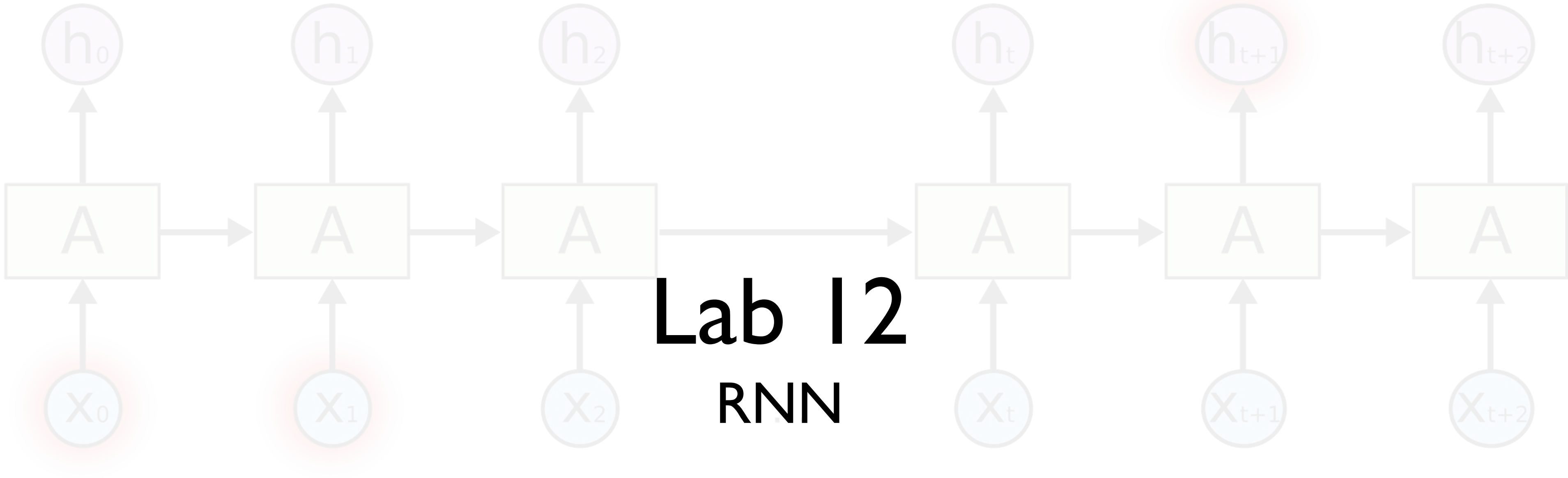
```
0 Accuracy: 0.9933
1 Accuracy: 0.9946
2 Accuracy: 0.9934
3 Accuracy: 0.9935
4 Accuracy: 0.9935
5 Accuracy: 0.9949
6 Accuracy: 0.9941

Ensemble accuracy: 0.9952
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-11-5-mnist_cnn_ensemble_layers.py

Exercise

- Deep & Wide?
- CIFAR 10
- ImageNet


Here are the classes in the dataset, as well as 10 random images from each
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck



Lab 12

RNN

Sung Kim <hunkim+ml@gmail.com>
http://hunkim.github.io/ml/