AlexNet

March 21, 2019

1 AlexNet

1.1 TensorFlow model

Following is an AlexNet model implementation with some of the hyperparameter changes.

AlexNet model implementation is given in paper krizhevsky et al. 2012 ImageNet Classification with Deep Convolutional Neural Networks

```
In [23]: import math
         import numpy as np
         import h5py
         import matplotlib.pyplot as plt
         import scipy
         from PIL import Image
         from scipy import ndimage
         import tensorflow as tf
         from tensorflow.python.framework import ops
         from cnn_utils import *

         %matplotlib inline
         np.random.seed(1)
```

Run the next cell to load the "SIGNS" dataset you are going to use.

```
In [24]: # Loading the data (signs)
         X_train_orig, Y_train_orig, X_test_orig, Y_test_orig, classes = load_datas
```

```
In [25]: X_train = X_train_orig/255.
         X_test = X_test_orig/255.
         Y_train = convert_to_one_hot(Y_train_orig, 6).T
         Y_test = convert_to_one_hot(Y_test_orig, 6).T
         print ("number of training examples = " + str(X_train.shape[0]))
         print ("number of test examples = " + str(X_test.shape[0]))
         print ("X_train shape: " + str(X_train.shape))
         print ("Y_train shape: " + str(Y_train.shape))
         print ("X_test shape: " + str(X_test.shape))
         print ("Y_test shape: " + str(Y_test.shape))
         conv_layers = {}
```

```
number of training examples = 1080
number of test examples = 120
X_train shape: (1080, 64, 64, 3)
Y_train shape: (1080, 6)
X_test shape: (120, 64, 64, 3)
Y_test shape: (120, 6)
```

In [26]:
```
# GRADED FUNCTION: create_placeholders

def create_placeholders(n_H0, n_W0, n_C0, n_y):
    """
    Creates the placeholders for the tensorflow session.

    Arguments:
    n_H0 -- scalar, height of an input image
    n_W0 -- scalar, width of an input image
    n_C0 -- scalar, number of channels of the input
    n_y -- scalar, number of classes

    Returns:
    X -- placeholder for the data input, of shape [None, n_H0, n_W0, n_C0]
    Y -- placeholder for the input labels, of shape [None, n_y] and dtype
    """

    ### START CODE HERE ### (≈2 lines)
    X = tf.placeholder(tf.float32 , shape = (None, n_H0, n_W0, n_C0) )
    Y = tf.placeholder(tf.float32 , shape = (None, n_y) )
    ### END CODE HERE ###

    return X, Y
```

In [27]:
```
X, Y = create_placeholders(64, 64, 3, 6)
print ("X = " + str(X))
print ("Y = " + str(Y))
```

```
X = Tensor("Placeholder_2:0", shape=(?, 64, 64, 3), dtype=float32)
Y = Tensor("Placeholder_3:0", shape=(?, 6), dtype=float32)
```

Expected Output

X = Tensor("Placeholder:0", shape=(?, 64, 64, 3), dtype=float32)
Y = Tensor("Placeholder_1:0", shape=(?, 6), dtype=float32)

In [28]:
```
# GRADED FUNCTION: initialize_parameters

def initialize_parameters():
    """
    Initializes weight parameters to build a neural network with tensorflo
```

```
                                W1 : [4, 4, 3, 8]
                                W2 : [2, 2, 8, 16]
    Returns:
    parameters -- a dictionary of tensors containing W1, W2
    """

    tf.set_random_seed(1)                                   # so that your "ra

    ### START CODE HERE ### (approx. 2 lines of code)
    W1 = tf.get_variable("W1", [4,4,3,96], initializer = tf.contrib.layers
    W2 = tf.get_variable("W2", [5,5,96,256], initializer = tf.contrib.laye
    W3 = tf.get_variable("W3", [3,3,256,384], initializer = tf.contrib.lay
    W4 = tf.get_variable("W4", [3,3,384,384], initializer = tf.contrib.lay
    W5 = tf.get_variable("W5", [3,3,384,256], initializer = tf.contrib.lay
    ### END CODE HERE ###

    parameters = {"W1": W1,
                  "W2": W2,
                  "W3": W3,
                  "W4": W4,
                  "W5": W5}

    return parameters
```

```
In [29]: tf.reset_default_graph()
         with tf.Session() as sess_test:
             parameters = initialize_parameters()
             init = tf.global_variables_initializer()
             sess_test.run(init)
             print("W1 = " + str(parameters["W1"].eval()[1,1,1]))
             print("W2 = " + str(parameters["W2"].eval()[1,1,1]))
             print("W3 = " + str(parameters["W3"].eval()[1,1,1]))
             print("W4 = " + str(parameters["W4"].eval()[1,1,1]))
             print("W5 = " + str(parameters["W5"].eval()[1,1,1]))
```

```
W1 = [ 0.01671694  0.06110588 -0.03386452  0.05938012  0.0145212  -0.04832294
 -0.0015871  -0.05174213 -0.0278542  -0.05713523  0.04564724 -0.0372206
  0.05195162 -0.05525731  0.01492433  0.05662248  0.04479869 -0.03715155
 -0.05752828 -0.00128394 -0.03468554  0.0339203  -0.03956766  0.02586722
  0.060818    0.04539385 -0.04254989  0.02562458 -0.04011025 -0.03983246
 -0.02190537  0.00698699 -0.03841595 -0.01560952 -0.04289921  0.0344194
  0.00680743 -0.03082583 -0.02350465 -0.0006859  -0.04149082  0.04848495
 -0.01811537  0.06091134  0.04603212  0.00425005  0.0264094  -0.0155791
  0.01267537  0.03423977  0.04724856 -0.02301043 -0.06008575 -0.04107204
  0.05599377  0.01036631 -0.02794269 -0.01615478  0.0023118  -0.0564984
 -0.04331018  0.01856513 -0.01615087 -0.00593869 -0.05587354  0.00578145
  0.03594472  0.00681745  0.0383089  -0.01609302  0.04841047 -0.01697941
  0.04661693 -0.05066967  0.02465578  0.00763012  0.01949468 -0.03252451
```

```
  0.02658572  0.05321065  0.03188632 -0.0090435   0.02677608  0.02768232
  0.02984896  0.00816341 -0.04608572 -0.0448757   0.01875175 -0.01630981
  0.01755036 -0.05491311  0.05647813 -0.00041866  0.05029892  0.02728762]
W2 = [ -2.28116140e-02   1.56503953e-02  -2.09522806e-02  -1.05351806e-02
  -2.86834314e-03  -2.60384977e-02  -2.04855613e-02  -8.38680193e-04
  -2.48638541e-03  -1.70377977e-02   2.28094310e-02   1.25329942e-02
   1.93224326e-02   6.72557205e-03  -1.94250420e-02  -6.17854297e-03
  -4.96730953e-03   1.85426772e-02  -1.10777393e-02   4.52413410e-03
  -2.20224708e-02   2.78520212e-03  -1.19479336e-02   1.21424980e-02
   7.11625442e-04  -1.36418492e-02   1.93708539e-02   2.07001567e-02
   2.27590576e-02   1.32034197e-02  -1.09712146e-02  -1.31583456e-02
  -2.20609996e-02  -1.40878754e-02  -1.73405558e-02  -1.96183659e-03
  -1.73670147e-02   7.14662671e-03  -1.68680847e-02  -2.08857115e-02
   2.41937712e-02  -7.04086199e-03  -1.85305607e-02  -1.55747551e-02
  -2.23334022e-02  -2.12480240e-02  -7.87477382e-03   1.71285793e-02
   1.89388357e-02   8.92788544e-03  -9.75785218e-03   1.98293477e-03
   1.07878745e-02   1.58816054e-02   2.02306844e-02  -1.16106309e-03
  -1.27728060e-02   4.86495532e-03   1.25059485e-02   1.42796338e-02
  -1.88271068e-02   1.23864189e-02  -2.73988768e-03   2.11786665e-02
  -1.19657815e-03   4.15571034e-04  -8.56810249e-03  -1.58635937e-02
   3.33899260e-03   1.38684697e-02   2.09810026e-03  -8.75842758e-03
  -1.79842599e-02   2.01283991e-02  -7.38026388e-03  -6.75064698e-03
  -1.13674486e-02   8.60524178e-03  -3.83711234e-03  -4.21114080e-03
  -9.25161317e-03  -6.26409985e-03   1.49438307e-02  -1.48791363e-02
   2.16972493e-02  -9.37076285e-03   1.26143992e-02   1.74523219e-02
  -2.16881596e-02  -5.96321560e-03  -8.06864910e-03   2.40674727e-02
  -1.41511885e-02  -2.20954772e-02  -1.07581774e-02  -1.57033987e-02
   2.75510922e-03  -1.84299871e-02   2.17168033e-02   2.49045417e-02
  -9.32989269e-03   3.71674076e-04   1.67023726e-02   1.95019022e-02
   2.26771422e-02  -2.31597256e-02  -2.28918120e-02   1.42620578e-02
   1.53184123e-02  -2.53770817e-02  -1.14697032e-02   2.60040164e-02
  -1.37862926e-02   8.81697983e-03   2.50400938e-02  -2.37854384e-02
   2.08353847e-02   2.60140114e-02   1.88717730e-02   2.25756541e-02
   1.75004900e-02   2.56490931e-02   7.05140084e-03  -1.10510755e-02
  -1.61550343e-02  -2.59027630e-02   2.04093754e-03  -1.62372421e-02
   1.43824518e-02   1.53250061e-02   2.18021683e-02  -2.10200027e-02
   2.47574635e-02   4.44566272e-03  -4.96343151e-03  -8.17756355e-03
  -1.07720112e-02   5.96679375e-03  -2.42185593e-02   1.25457421e-02
   1.68355443e-02   1.49464346e-02  -1.97378285e-02   1.01043098e-03
  -1.70328468e-03  -1.21614309e-02   6.31091744e-03   1.33283585e-02
  -8.03607702e-03   1.70042999e-02  -1.28471637e-02   2.48464383e-02
   1.28031373e-02   1.71082467e-02   1.12551861e-02   4.56025451e-03
   3.12375091e-03   1.60355233e-02   2.50408873e-02   1.10869221e-02
  -1.58079304e-02  -1.46511588e-02  -1.56884901e-02  -2.24675238e-03
  -9.51839983e-03  -2.44935341e-02   4.31986898e-03   1.74547248e-02
   5.31339645e-03  -1.78105459e-02  -1.24370214e-02  -5.58655336e-03
   2.43464857e-02  -2.26440672e-02   2.15265639e-02   1.44334696e-02
   2.30001807e-02   1.32468604e-02  -1.18725486e-02  -2.20355187e-02
```

```
  -8.28113221e-03   2.86033750e-03  -2.37379763e-02  -1.69697218e-02
   1.15917027e-02   1.92906074e-02  -4.10260633e-04   2.02245899e-02
   8.64890963e-03   1.86929144e-02   9.22200456e-03  -1.81243606e-02
   6.22365996e-03   4.87270020e-03  -1.29485643e-02   2.06260085e-02
  -3.23323347e-03  -7.11390004e-03  -3.64359282e-03   2.25017220e-02
   9.93504748e-03  -2.47857533e-02   1.22857429e-02   1.23111717e-02
   4.86853532e-03   1.09994113e-02   1.29152276e-02  -1.08455587e-02
   2.18650140e-03   2.59937420e-02  -1.76716726e-02  -1.47688575e-02
  -3.64778377e-03   1.72021687e-02  -2.27755494e-02  -1.39360540e-02
   1.36064552e-03  -3.25546414e-03   2.79416703e-03  -2.06871256e-02
   1.67099275e-02  -9.58685577e-03  -2.28858851e-02  -9.30551253e-03
   1.64918229e-02  -2.30539367e-02   1.15740746e-02   1.55139714e-05
  -4.50919941e-03   2.07831524e-02  -2.08854321e-02   1.12305321e-02
  -8.55347142e-04   2.47392729e-02   8.80973414e-03   1.98072568e-02
   6.22695312e-03   7.62192532e-03   1.62524395e-02   2.53255852e-02
   1.14379339e-02   1.71308517e-02  -1.30487457e-02  -1.57330576e-02
  -3.31781246e-03   1.87824517e-02   9.50602815e-04  -1.97389740e-02
   2.04719156e-02  -2.17265654e-02  -2.56286785e-02  -7.98400119e-03
  -1.89660490e-03  -1.71508845e-02  -1.82185266e-02  -2.42804158e-02]
W3 = [ -2.52654366e-02   9.48805362e-03  -1.86284631e-02  -3.21126431e-02
   8.69024917e-03   6.01965934e-04  -1.96405984e-02   1.84573904e-02
   7.76708126e-03   3.16396765e-02  -2.72833239e-02  -2.28701457e-02
  -3.22012603e-03  -7.60075636e-03  -8.93948786e-03   2.40180120e-02
  -1.99917424e-02  -1.12908501e-02  -3.05692498e-02  -1.46081485e-02
   1.35720633e-02   9.17029753e-03   2.10062861e-02  -4.75275517e-03
   3.18480320e-02  -2.20453329e-02   7.89177045e-03   3.67438048e-03
   1.63373463e-02   7.88483024e-03  -1.14708208e-03  -1.53476689e-02
   1.04521103e-02   2.30432637e-02   2.09836662e-03  -9.64063406e-03
  -2.86115687e-02   3.09754573e-02  -7.88081251e-03   9.62344557e-03
  -1.63701177e-02  -1.33356843e-02  -2.22635008e-02  -3.05798072e-02
  -8.23901221e-03  -6.94086216e-03  -1.71542615e-02  -1.60118490e-02
   2.45709233e-02   1.24289393e-02  -2.90630981e-02   1.99617259e-02
   3.12704220e-03   2.77591310e-02   1.92498043e-02  -4.47371416e-03
   2.88650170e-02  -1.48755852e-02   3.13170962e-02   2.67066956e-02
   2.58136690e-02  -5.47687896e-03  -9.07146372e-03   2.64718570e-02
   1.43976621e-02   8.66010785e-04  -8.58764537e-03   1.82803459e-02
  -1.03779882e-03  -2.12974325e-02   7.14082271e-03   4.84588742e-03
   1.51312426e-02   1.37211010e-02  -1.79184452e-02   1.48426890e-02
  -3.53084691e-03  -1.13312714e-02  -9.14226472e-03  -1.12383794e-02
   1.31572783e-02   1.59223489e-02  -1.93343647e-02  -1.51769351e-02
  -2.77663339e-02  -1.77864991e-02   2.43163444e-02  -2.99473852e-02
   2.84502916e-02  -2.93214321e-02  -3.08156386e-03   5.17319143e-03
  -1.21186320e-02  -3.11755557e-02  -1.42795444e-02   9.06738639e-03
  -2.99577806e-02   1.26180798e-03  -2.53256503e-02   3.19252573e-02
  -1.58906858e-02  -1.60747245e-02   8.89611244e-03   1.41575336e-02
  -2.95532886e-02  -4.33970243e-04  -2.02652588e-02   2.15807594e-02
   7.97616690e-03  -5.87994792e-03  -2.89721601e-03   2.04581246e-02
  -7.87183642e-04   1.17731467e-03  -7.31881335e-03  -2.61217430e-02
```

```
-2.82618254e-02  -7.79837742e-03  -8.37631524e-04  -1.99464336e-02
 1.82352960e-05   2.96582542e-02   2.97308788e-02   2.36910582e-03
-8.32738727e-03  -1.35082211e-02  -3.18604819e-02  -2.61129625e-03
 8.76035914e-03  -5.81083074e-03  -3.15470211e-02  -2.07581967e-02
 7.81723112e-03   1.40576139e-02  -1.84134357e-02  -3.16274799e-02
 2.10222863e-02   2.88765579e-02  -1.55311618e-02   2.35224739e-02
-2.04435103e-02   1.73229724e-03  -5.04173338e-04   9.37767327e-05
-2.02058218e-02   2.04105377e-02  -8.73883627e-03  -1.16048642e-02
 1.38149932e-02   3.00846994e-02  -3.08130793e-02  -2.73738001e-02
-1.27821509e-02  -2.79430859e-02  -1.27585120e-02   2.65248567e-02
 1.03123002e-02  -3.19578387e-02  -6.58590347e-03  -4.78730537e-03
 1.35402158e-02  -1.84784736e-02   2.08281875e-02  -3.04682180e-03
-2.86676399e-02   2.99472213e-02  -1.12949982e-02  -1.77997202e-02
-3.20704691e-02   3.00224796e-02   2.71638669e-02   1.16583072e-02
-1.74432825e-02  -3.17719989e-02   5.58549166e-03  -2.17913538e-02
 2.95937024e-02  -2.81560831e-02  -5.01029752e-03   1.97515637e-03
 2.83542499e-02  -5.76616265e-03  -1.64092220e-02  -6.69310056e-03
 3.17269526e-02  -1.18017327e-02   1.42288022e-02   1.71669424e-02
 1.02547742e-02   1.04069114e-02  -1.09344590e-02  -3.02985497e-03
 3.67402658e-03  -2.78475769e-02   3.20602022e-02  -2.09702980e-02
 2.91639268e-02   1.36400796e-02  -2.96610482e-02   1.21603012e-02
 2.22789831e-02   1.77013092e-02  -2.07042620e-02   1.21951178e-02
 1.84795037e-02  -1.03297997e-02  -7.20819086e-03   1.16985105e-02
-1.01767015e-02  -1.63314119e-03   2.63395719e-02   2.38251686e-03
-2.88248174e-02   2.67854780e-02  -1.16053335e-02   2.31085718e-02
 8.10972229e-03   1.67791657e-02  -2.81229094e-02   9.21143591e-03
 2.20419541e-02   2.78143361e-02  -1.33951735e-02   4.52110916e-03
-3.35382670e-03   8.79030675e-03  -1.04315430e-02   9.38742608e-03
 2.73697078e-03   1.85816325e-02   1.20667852e-02   1.00556836e-02
-3.14987190e-02  -8.44261236e-03  -3.14451307e-02   3.01005840e-02
 1.44591294e-02   1.65085718e-02  -2.63085682e-02  -2.90829353e-02
-2.69476548e-02   1.89783424e-02  -8.19619000e-03   2.98139602e-02
-2.39140987e-02  -2.84877867e-02  -5.56484796e-03   2.26782262e-02
-2.18301676e-02  -2.83756498e-02   1.94959193e-02  -1.27526708e-02
-1.26247499e-02  -2.19299868e-02   1.23868920e-02  -1.47584993e-02
 9.02951136e-03   1.96096115e-02   1.15674287e-02   1.53777190e-02
-2.97047868e-02  -3.06967162e-02   2.02891827e-02  -2.19811499e-02
-1.04892161e-02   1.46604516e-02   2.36157589e-02  -2.34402772e-02
 2.21064463e-02  -1.96184218e-02  -3.90041992e-03  -1.53082106e-02
 1.85879432e-02   2.99208760e-02   3.11652943e-03  -2.35038362e-02
-2.50062644e-02  -2.82088928e-02   8.94382223e-03  -1.95155647e-02
 2.19933987e-02  -3.13628204e-02   3.02137360e-02  -1.74973160e-04
-2.75591947e-02  -1.22428276e-02  -1.78473033e-02  -1.36919990e-02
 2.02232450e-02   5.57336956e-04  -2.99807452e-03  -1.51637457e-02
-1.85195878e-02  -2.78642513e-02   5.31440973e-03   6.85732812e-04
 2.19102576e-02   2.47329399e-02   2.88912095e-02   6.72215596e-03
 2.32652947e-03  -2.47483384e-02   2.41762586e-02   7.58368149e-03
-1.12736207e-02   1.14830062e-02  -2.54772399e-02   6.33840263e-03
```

```
   4.50205430e-03  -1.28488429e-02  -2.93681175e-02  -2.16162950e-02
   2.43331641e-02   1.19332597e-02  -1.76074468e-02  -1.30328964e-02
   4.70712408e-03   4.93161753e-03  -2.81401686e-02   1.92249119e-02
  -2.50384677e-02   6.46509975e-03   2.75478065e-02  -1.62562560e-02
   2.65124477e-02  -2.89820321e-02   2.46308595e-02   2.62328275e-02
  -7.33321160e-03   4.69728187e-03  -6.25040382e-03  -1.54863708e-02
  -6.36629015e-03  -1.06993262e-02  -1.26027353e-02   1.77243948e-02
  -7.51685910e-03  -2.75672209e-02   2.11036839e-02   2.63386630e-02
  -3.74324247e-03  -2.22334433e-02  -1.58813279e-02  -1.63184237e-02
   2.50214152e-02  -2.38050707e-02  -2.30590552e-02  -6.44195825e-04
  -3.00810989e-02  -1.70257092e-02   2.72374153e-02   3.15164439e-02
  -1.44076720e-03  -2.72787604e-02  -3.20438892e-02  -1.56763960e-02
   8.75578821e-03  -6.98973238e-03   5.97841293e-03  -1.34601891e-02
  -1.28397923e-02   1.49974674e-02   2.77152546e-02   1.52868032e-03
  -2.68319398e-02   6.22936711e-03  -8.37869942e-04  -1.32319257e-02
   1.69289783e-02   1.13459006e-02  -9.98224318e-03  -1.42330285e-02
   1.99795067e-02  -1.52655412e-02   1.07876323e-02  -2.38478705e-02
  -8.78676027e-03  -6.86823763e-03   1.53603964e-02   1.52571462e-02
  -1.70840994e-02  -2.87272688e-02  -9.01207514e-03  -2.97464542e-02]
W4 = [ -4.89076972e-03  -2.91089248e-02   2.25343537e-02  -1.62415691e-02
   1.77900493e-03   1.58868898e-02   1.06018502e-02   3.23152356e-03
  -2.07652189e-02  -2.92120092e-02  -2.63499357e-02   1.73448753e-02
   2.31873635e-02  -2.16206349e-03   1.38823632e-02   7.75706954e-03
   6.52522407e-03  -2.57320628e-02   7.09596463e-03  -2.85090841e-02
   2.66359579e-02  -2.86701545e-02  -9.97641124e-03  -1.17212292e-02
   2.76828576e-02   1.05945934e-02   5.35250269e-03   2.18085777e-02
  -2.93732285e-02   7.24026933e-04   4.01734747e-03   1.67196337e-02
   1.24212354e-03   2.39806939e-02  -7.38351047e-03  -2.63811313e-02
  -4.57471795e-03   1.89058669e-03  -4.49907780e-03   3.65865417e-03
  -3.88938934e-04  -1.89946517e-02   1.25014577e-02  -2.05800254e-02
   9.33294930e-03   1.84220280e-02   2.90899780e-02  -9.39756632e-03
   2.82955337e-02  -5.90318628e-03   1.47771481e-02  -7.59123079e-03
   2.03307625e-02  -1.37071311e-02  -2.31697410e-03  -9.64168832e-03
   7.67331757e-03  -1.94885284e-02  -1.31968316e-02  -2.09946167e-02
  -8.99825431e-03  -1.17791519e-02   2.61277314e-02   2.89409067e-02
   6.62823953e-03  -3.38429399e-03  -7.40532763e-03   2.60269456e-03
  -2.67013144e-02  -1.94629226e-02  -9.18764807e-03   2.06070151e-02
   4.81856428e-03   2.88231988e-02  -1.98557172e-02   1.42271314e-02
  -1.17137972e-02  -1.66150331e-02   2.77652200e-02  -1.35200564e-02
   1.62001383e-02   1.16430596e-03  -8.62583704e-03  -1.99205615e-02
   2.13208552e-02  -5.03392890e-03  -1.75483394e-02   2.36127526e-04
  -2.02017780e-02  -2.77884789e-02   5.39316051e-03  -1.64439585e-02
   2.33250502e-02   2.02515032e-02   1.56586077e-02  -4.60171327e-03
   7.36508518e-05   1.05681624e-02   1.10908728e-02  -1.65428072e-02
  -1.09365396e-03  -1.26323234e-02  -6.16871752e-03  -2.37983093e-02
   2.77916472e-02  -1.19941924e-02   2.42158640e-02   2.36359965e-02
   8.10554251e-04   3.46785598e-03  -4.88059036e-03   6.12413324e-03
  -1.78131536e-02  -1.66686848e-02   1.69964600e-02  -2.52321176e-03
```

```
 6.57428242e-03   2.65861619e-02   2.08942425e-02  -1.73065569e-02
-2.00105235e-02   2.05476638e-02   2.39481907e-02   2.70683002e-02
 2.55674589e-02   1.27443727e-02   6.31717965e-04  -1.87002122e-03
-2.33135074e-02  -1.26183666e-02   1.41791068e-03  -1.59693211e-02
 5.62367402e-03  -1.39481556e-02   2.04696786e-02   2.84778606e-02
 1.65169146e-02  -1.23725086e-02   1.09592471e-02  -4.19420935e-03
-2.86516100e-02  -1.39074977e-02  -8.17440450e-05  -2.87534297e-02
 2.11314168e-02   8.10145400e-03  -4.52022813e-03   2.34010126e-02
-2.66654752e-02   2.50673238e-02   1.75902192e-02   2.11156625e-02
 5.66243567e-03   1.23136733e-02   1.75234675e-03   1.16527062e-02
-8.84109549e-03  -7.05795363e-03   1.76740754e-02  -3.47647630e-03
 2.73754653e-02  -1.48626603e-03  -1.46226399e-03  -5.35931438e-04
-2.40378436e-02  -1.56417973e-02  -1.39538310e-02   1.97659936e-02
-9.29832458e-03  -1.67754367e-02  -8.72808509e-03  -2.79385224e-03
-9.74237919e-04   2.17257533e-02   2.60607731e-02   2.22415868e-02
-1.10615175e-02   2.07750332e-02   1.07468646e-02   1.47852469e-02
-1.79185085e-02   1.66895259e-02   2.75237318e-02   2.01491509e-02
 1.59693863e-02   1.43354442e-02   2.26448905e-02  -1.39871761e-02
-2.28597969e-02  -8.67331587e-03  -1.87239274e-02  -1.00585632e-03
-4.94880602e-04   3.73067148e-03   1.90931279e-02  -1.43646291e-02
-1.71081163e-03  -2.13225894e-02   1.81576628e-02   2.25261990e-02
-1.73978191e-02   2.52765808e-02  -1.28773302e-02   1.74034890e-02
 1.29772257e-02   1.84867177e-02  -2.32727863e-02   1.30072236e-03
-2.61392649e-02   2.18022410e-02   2.32571494e-02   1.88937914e-02
 1.23115871e-02  -4.86035272e-03  -2.77061164e-02   1.31422151e-02
 6.42335601e-03  -1.43020274e-02  -1.52600389e-02   1.06902812e-02
-1.94338560e-02   2.28215065e-02  -2.11002007e-02   2.76649650e-02
-2.86598578e-02   1.20120291e-02  -1.44523010e-03   9.83663835e-03
 2.26780195e-02  -6.86174631e-03   2.17739735e-02  -1.48475617e-02
-1.70812048e-02   2.37910543e-02  -1.21577457e-03   1.06347594e-02
 1.29892807e-02  -1.73293017e-02   2.50640269e-02   2.86251996e-02
 2.54491158e-03  -2.22986899e-02   1.08260717e-02   1.81416888e-02
 8.55893455e-03  -2.05188282e-02   1.07323211e-02  -3.47303227e-04
 8.40176456e-03  -2.78166309e-03  -5.51898032e-04   1.17862243e-02
 2.69863028e-02  -1.72539931e-02   1.14465877e-03   2.07032617e-02
-6.38691150e-03  -2.66829655e-02  -5.51449135e-04  -1.34142041e-02
 2.03154050e-03  -2.51865778e-02   1.37392823e-02  -1.63140055e-02
 1.46699045e-02  -1.18065681e-02   2.83077639e-02   1.48068834e-02
-1.53904911e-02   6.91946037e-03   9.45172645e-03   1.91511773e-03
 1.33257676e-02  -2.30845660e-02   2.60180850e-02  -2.01955065e-03
-7.31946714e-03  -5.11095114e-03   1.79091450e-02  -1.91896930e-02
-1.37849841e-02   2.05508117e-02   1.97410379e-02   2.61525121e-02
 3.38738970e-03  -2.72920504e-02   2.47877296e-02  -1.86123513e-02
-3.31436470e-03  -8.09652172e-03   1.49807688e-02  -2.37813871e-02
 2.36467626e-02  -1.95772257e-02  -1.38027631e-02  -4.46417928e-03
-2.20034439e-02  -1.84362456e-02  -2.80286092e-02   4.67365608e-04
 2.51508448e-02  -1.24410260e-02  -8.39685276e-03  -2.38021947e-02
-1.23174302e-02   1.60309132e-02  -1.13261938e-02   6.98613003e-04
```

```
   6.20189495e-03  -5.02273068e-03  -1.53753394e-02  -2.53609177e-02
  -7.77665526e-03   1.41681489e-02   1.81314480e-02   2.75476780e-02
  -1.41110178e-02   2.80952659e-02  -1.55253522e-03  -1.41592696e-04
   2.94369478e-02   1.54754575e-02   1.76287200e-02   5.00087999e-03
   1.22136306e-02  -2.03773752e-02  -8.25195946e-03  -1.35234911e-02
   1.99332740e-02   1.25652272e-02  -2.21600477e-02  -1.34808589e-02
  -2.77890544e-02   1.59263182e-02  -1.22619085e-02  -2.65075509e-02
  -3.61553207e-03  -1.17459968e-02   6.68115355e-03   2.86727902e-02
  -1.38831083e-02   1.66508928e-03   3.46855260e-03   1.21306833e-02
   5.20572625e-03   1.40513014e-02   4.91523556e-03  -2.86502130e-02
   2.90413294e-02  -2.55340151e-02   1.40729528e-02   9.06254910e-03
  -1.53602930e-02  -1.55783882e-02   1.81161053e-03  -1.41366431e-02
  -1.53782470e-02   2.06340421e-02  -1.09641515e-02   2.49069761e-02
  -2.24996246e-02   1.40047725e-02  -2.00740378e-02  -2.00746283e-02
   3.61340307e-03   5.20570390e-03  -2.70388834e-03   2.57981140e-02
   2.73082796e-02   1.33823138e-02  -1.96659938e-03   2.02569179e-03
   2.59946678e-02   2.82173920e-02  -1.23807788e-03   2.64149737e-02
  -8.96634907e-03  -5.26448712e-04   2.78111380e-02   2.54497696e-02
   6.34213723e-03   3.02872993e-03  -1.55791538e-02  -2.32341383e-02]
W5 = [  1.92275271e-02   7.53147900e-03  -2.91527994e-02   9.72753391e-03
   6.84950873e-03  -7.38799945e-03  -5.81317768e-03   2.05970779e-02
   1.72854066e-02  -2.60632243e-02   2.45170146e-02  -3.22503150e-02
  -4.59666364e-03   1.96576528e-02  -2.90786419e-02  -1.55715067e-02
   1.24333017e-02   2.55253948e-02   1.64799541e-02   1.51696019e-02
  -4.53264266e-03   2.77751908e-02   1.08459853e-02  -1.84808895e-02
   3.84243205e-03  -3.11368126e-02   2.78578252e-02  -1.86226759e-02
   2.87191980e-02   2.71252692e-02   2.52398290e-02  -1.79977641e-02
  -2.09563170e-02  -2.88551580e-02  -2.94115786e-02  -2.77974643e-03
  -3.09810899e-02   9.76259261e-03  -3.21706794e-02  -2.47014910e-02
  -3.11596729e-02   2.32211426e-02   2.68629342e-02   6.60672411e-03
   2.27521136e-02  -5.23351133e-03  -4.20087576e-03   2.00313255e-02
  -1.50813479e-02  -2.79840473e-02   2.01260895e-02  -2.74006333e-02
   1.86673142e-02  -2.46787220e-02  -1.10475291e-02  -3.00861858e-02
  -1.95201896e-02   2.09020302e-02  -1.21839456e-02  -2.57611591e-02
   8.67732242e-03   4.43997979e-03  -8.41055065e-04   7.36048073e-03
  -1.19520724e-03   2.41987593e-02  -2.62665469e-02   1.72586367e-02
   1.11584961e-02  -1.17990840e-02  -1.64970830e-02   3.81037965e-03
   2.31272317e-02   2.33251154e-02  -2.08918750e-03  -1.10982768e-02
  -1.48104094e-02   2.51681879e-02  -2.51972210e-02   1.87493265e-02
   6.58901036e-03  -1.11191534e-02  -1.64932981e-02   8.57320055e-03
  -8.58688354e-03   3.17182280e-02   2.60371752e-02   6.93454593e-03
  -1.25867072e-02  -1.30001158e-02  -1.48793869e-02   2.75516883e-03
   2.19464377e-02  -9.71005112e-03   4.36833128e-03  -1.13633741e-02
   2.94772238e-02  -2.24150047e-02   2.43558660e-02   3.50125134e-03
  -2.19724160e-02   1.99813396e-04  -5.12155145e-03  -1.54592227e-02
  -1.94635466e-02  -1.96376815e-03   1.74782574e-02   5.60801476e-03
  -8.50153901e-03   5.83323091e-03   3.47994268e-04   2.82813162e-02
   7.41049275e-03   2.04787850e-02  -2.66970471e-02  -2.61700824e-02
```

```
  -3.02474555e-02   2.36873925e-02  -9.37646814e-03  -1.77363511e-02
  -1.90707296e-03   2.23765858e-02   6.59492984e-03  -1.64286364e-02
   1.83587335e-02  -7.91140087e-03  -1.57474596e-02  -1.89100821e-02
  -2.52654366e-02   9.48805362e-03  -1.86284631e-02  -3.21126431e-02
   8.69024917e-03   6.01965934e-04  -1.96405984e-02   1.84573904e-02
   7.76708126e-03   3.16396765e-02  -2.72833239e-02  -2.28701457e-02
  -3.22012603e-03  -7.60075636e-03  -8.93948786e-03   2.40180120e-02
  -1.99917424e-02  -1.12908501e-02  -3.05692498e-02  -1.46081485e-02
   1.35720633e-02   9.17029753e-03   2.10062861e-02  -4.75275517e-03
   3.18480320e-02  -2.20453329e-02   7.89177045e-03   3.67438048e-03
   1.63373463e-02   7.88483024e-03  -1.14708208e-03  -1.53476689e-02
   1.04521103e-02   2.30432637e-02   2.09836662e-03  -9.64063406e-03
  -2.86115687e-02   3.09754573e-02  -7.88081251e-03   9.62344557e-03
  -1.63701177e-02  -1.33356843e-02  -2.22635008e-02  -3.05798072e-02
  -8.23901221e-03  -6.94086216e-03  -1.71542615e-02  -1.60118490e-02
   2.45709233e-02   1.24289393e-02  -2.90630981e-02   1.99617259e-02
   3.12704220e-03   2.77591310e-02   1.92498043e-02  -4.47371416e-03
   2.88650170e-02  -1.48755852e-02   3.13170962e-02   2.67066956e-02
   2.58136690e-02  -5.47687896e-03  -9.07146372e-03   2.64718570e-02
   1.43976621e-02   8.66010785e-04  -8.58764537e-03   1.82803459e-02
  -1.03779882e-03  -2.12974325e-02   7.14082271e-03   4.84588742e-03
   1.51312426e-02   1.37211010e-02  -1.79184452e-02   1.48426890e-02
  -3.53084691e-03  -1.13312714e-02  -9.14226472e-03  -1.12383794e-02
   1.31572783e-02   1.59223489e-02  -1.93343647e-02  -1.51769351e-02
  -2.77663339e-02  -1.77864991e-02   2.43163444e-02  -2.99473852e-02
   2.84502916e-02  -2.93214321e-02  -3.08156386e-03   5.17319143e-03
  -1.21186320e-02  -3.11755557e-02  -1.42795444e-02   9.06738639e-03
  -2.99577806e-02   1.26180798e-03  -2.53256503e-02   3.19252573e-02
  -1.58906858e-02  -1.60747245e-02   8.89611244e-03   1.41575336e-02
  -2.95532886e-02  -4.33970243e-04  -2.02652588e-02   2.15807594e-02
   7.97616690e-03  -5.87994792e-03  -2.89721601e-03   2.04581246e-02
  -7.87183642e-04   1.17731467e-03  -7.31881335e-03  -2.61217430e-02
  -2.82618254e-02  -7.79837742e-03  -8.37631524e-04  -1.99464336e-02
   1.82352960e-05   2.96582542e-02   2.97308788e-02   2.36910582e-03
  -8.32738727e-03  -1.35082211e-02  -3.18604819e-02  -2.61129625e-03]
```

** Expected Output:**

```
<tr>
    <td>
    W1 =
    </td>
    <td>
```

[0.00131723 0.14176141 -0.04434952 0.09197326 0.14984085 -0.03514394 -0.06847463 0.05245192]

```
<tr>
    <td>
```

W2 =
</td>
<td>

[-0.08566415 0.17750949 0.11974221 0.16773748 -0.0830943 -0.08058 -0.00577033 -0.14643836
0.24162132 -0.05857408 -0.19055021 0.1345228 -0.22779644 -0.1601823 -0.16117483 -0.10286498]

In [30]:
```
# GRADED FUNCTION: forward_propagation

def forward_propagation(X, parameters):
    """
    Implements the forward propagation for the model:
    CONV2D -> RELU -> MAXPOOL -> CONV2D -> RELU -> MAXPOOL -> FLATTEN ->

    Arguments:
    X -- input dataset placeholder, of shape (input size, number of exampl
    parameters -- python dictionary containing your parameters "W1", "W2"
                  the shapes are given in initialize_parameters

    Returns:
    Z3 -- the output of the last LINEAR unit
    """

    # Retrieve the parameters from the dictionary "parameters"
    W1 = parameters['W1']
    W2 = parameters['W2']
    W3 = parameters['W3']
    W4 = parameters['W4']
    W5 = parameters['W5']

    ### START CODE HERE ###
    # CONV2D: stride of 1, padding 'VALID'
    Z1 = tf.nn.conv2d( X, W1, strides = [1,1,1,1], padding = 'VALID')
    # ?? have not taken b1 parameter
    # RELU
    A1 = tf.nn.relu(Z1)
    # MAXPOOL: window 3x3, sride 2, padding 'VALID'
    P1 = tf.nn.max_pool(A1, ksize = [1,3,3,1], strides = [1,2,2,1], paddir
    # CONV2D: filters W2, stride 1, padding 'SAME'
    Z2 = tf.nn.conv2d(P1,W2, strides = [1,1,1,1], padding = 'SAME')
    # RELU
    A2 = tf.nn.relu(Z2)
    # MAXPOOL: window 2x2, stride 1, padding 'VALID'
    P2 = tf.nn.max_pool(A2, ksize = [1,2,2,1], strides = [1,1,1,1], paddir
    # CONV2D: filters W3, stride 1, padding 'SAME'
    Z3 = tf.nn.conv2d(P2,W3, strides = [1,1,1,1], padding = 'SAME')
    # RELU
    A3 = tf.nn.relu(Z3)
```

```
    # CONV2D: filters W4, stride 1, padding 'SAME'
    Z4 = tf.nn.conv2d(A3,W4, strides = [1,1,1,1], padding = 'SAME')
    # RELU
    A4 = tf.nn.relu(Z4)
    # CONV2D: filters W5, stride 1, padding 'SAME'
    Z5 = tf.nn.conv2d(A4,W5, strides = [1,1,1,1], padding = 'SAME')
    # RELU
    A5 = tf.nn.relu(Z5)
    # MAXPOOL: window 3x3, stride 2, padding 'VALID'
    P3 = tf.nn.max_pool(A5, ksize = [1,3,3,1], strides = [1,2,2,1], paddin
    # FLATTEN
    P3 = tf.contrib.layers.flatten(P3)
    # FULLY-CONNECTED with non-linear activation function.
    A6 = tf.contrib.layers.fully_connected(P3, 4096)
    A7 = tf.contrib.layers.fully_connected(A6, 4096)
    # FULLY-CONNECTED without non-linear activation function (not not cal
    # 6 neurons in output layer. Hint: one of the arguments should be "act
    Z8 = tf.contrib.layers.fully_connected(A7, 6, activation_fn=None)
    ### END CODE HERE ###

    return Z8
```

```
In [31]: tf.reset_default_graph()

    with tf.Session() as sess:
        np.random.seed(1)
        X, Y = create_placeholders(64, 64, 3, 6)
        parameters = initialize_parameters()
        Z8 = forward_propagation(X, parameters)
        init = tf.global_variables_initializer()
        sess.run(init)
        a = sess.run(Z8, {X: np.random.randn(2,64,64,3), Y: np.random.randn(2,
        print("Z8 = " + str(a))
```

```
Z8 = [[-0.00572854 -0.10159744  0.19650701  0.0035533   0.02199078  0.00088809]
 [ 0.00432064 -0.09484988  0.18270491  0.02251715  0.09731591 -0.02702793]]
```

Expected Output:

Z3 =

[[-0.44670227 -1.57208765 -1.53049231 -2.31013036 -1.29104376 0.46852064] [-0.17601591 -1.57972014 -1.4737016 -2.61672091 -1.00810647 0.5747785]]

```
In [43]: # GRADED FUNCTION: compute_cost

    def compute_cost(Z8, Y):
        """
        Computes the cost
```

```
    Arguments:
    Z3 -- output of forward propagation (output of the last LINEAR unit),
    Y -- "true" labels vector placeholder, same shape as Z3

    Returns:
    cost - Tensor of the cost function
    """

    ### START CODE HERE ### (1 line of code)
    cost = tf.reduce_mean( tf.nn.softmax_cross_entropy_with_logits(logits
    ### END CODE HERE ###

    return cost
```

```
In [44]: tf.reset_default_graph()

    with tf.Session() as sess:
        np.random.seed(1)
        X, Y = create_placeholders(64, 64, 3, 6)
        parameters = initialize_parameters()
        Z8 = forward_propagation(X, parameters)
        cost = compute_cost(Z8, Y)
        init = tf.global_variables_initializer()
        sess.run(init)
        a = sess.run(cost, {X: np.random.randn(4,64,64,3), Y: np.random.randn
        print("cost = " + str(a))
```

```
cost = 1.02641
```

Expected Output:
cost =

```
<td>
2.91034
</td>
```

```
In [45]: # GRADED FUNCTION: model

    def model(X_train, Y_train, X_test, Y_test, learning_rate = 0.009,
              num_epochs = 100, minibatch_size = 64, print_cost = True):
        """
        Implements a three-layer ConvNet in Tensorflow:
        CONV2D -> RELU -> MAXPOOL -> CONV2D -> RELU -> MAXPOOL -> FLATTEN -> F

        Arguments:
        X_train -- training set, of shape (None, 64, 64, 3)
        Y_train -- test set, of shape (None, n_y = 6)
        X_test -- training set, of shape (None, 64, 64, 3)
```

```
    Y_test -- test set, of shape (None, n_y = 6)
    learning_rate -- learning rate of the optimization
    num_epochs -- number of epochs of the optimization loop
    minibatch_size -- size of a minibatch
    print_cost -- True to print the cost every 100 epochs

    Returns:
    train_accuracy -- real number, accuracy on the train set (X_train)
    test_accuracy -- real number, testing accuracy on the test set (X_test
    parameters -- parameters learnt by the model. They can then be used to
    """

    ops.reset_default_graph()                         # to be able to reru
    tf.set_random_seed(1)                             # to keep results co
    seed = 3                                          # to keep results co
    (m, n_H0, n_W0, n_C0) = X_train.shape
    n_y = Y_train.shape[1]
    costs = []                                        # To keep track of t

    # Create Placeholders of the correct shape
    ### START CODE HERE ### (1 line)
    X, Y = create_placeholders(n_H0, n_W0, n_C0, n_y)
    ### END CODE HERE ###

    # Initialize parameters
    ### START CODE HERE ### (1 line)
    parameters = initialize_parameters()
    ### END CODE HERE ###

    # Forward propagation: Build the forward propagation in the tensorflow
    ### START CODE HERE ### (1 line)
    Z8 = forward_propagation(X, parameters)
    ### END CODE HERE ###

    # Cost function: Add cost function to tensorflow graph
    ### START CODE HERE ### (1 line)
    cost = compute_cost(Z8, Y)
    ### END CODE HERE ###

    # Backpropagation: Define the tensorflow optimizer. Use an AdamOptimiz
    ### START CODE HERE ### (1 line)
    optimizer = tf.train.AdamOptimizer(learning_rate = learning_rate).mini
    ### END CODE HERE ###

    # Initialize all the variables globally
    init = tf.global_variables_initializer()

    # Start the session to compute the tensorflow graph
```

```
with tf.Session() as sess:

    # Run the initialization
    sess.run(init)

    # Do the training loop
    for epoch in range(num_epochs):

        minibatch_cost = 0.
        num_minibatches = int(m / minibatch_size) # number of minibat
        seed = seed + 1
        minibatches = random_mini_batches(X_train, Y_train, minibatch_

        for minibatch in minibatches:

            # Select a minibatch
            (minibatch_X, minibatch_Y) = minibatch
            # IMPORTANT: The line that runs the graph on a minibatch.
            # Run the session to execute the optimizer and the cost, t
            ### START CODE HERE ### (1 line)
            _ , temp_cost = sess.run([optimizer, cost], feed_dict = {X
            ### END CODE HERE ###

            minibatch_cost += temp_cost / num_minibatches


        # Print the cost every epoch
        if print_cost == True and epoch % 5 == 0:
            print ("Cost after epoch %i: %f" % (epoch, minibatch_cost))
        if print_cost == True and epoch % 1 == 0:
            costs.append(minibatch_cost)


    # plot the cost
    plt.plot(np.squeeze(costs))
    plt.ylabel('cost')
    plt.xlabel('iterations (per tens)')
    plt.title("Learning rate =" + str(learning_rate))
    plt.show()

    # Calculate the correct predictions
    predict_op = tf.argmax(Z3, 1)
    correct_prediction = tf.equal(predict_op, tf.argmax(Y, 1))

    # Calculate accuracy on the test set
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
    print(accuracy)
    train_accuracy = accuracy.eval({X: X_train, Y: Y_train})
```

```
    test_accuracy = accuracy.eval({X: X_test, Y: Y_test})
    print("Train Accuracy:", train_accuracy)
    print("Test Accuracy:", test_accuracy)

    return train_accuracy, test_accuracy, parameters
```

Run the following cell to train your model for 100 epochs. Check if your cost after epoch 0 and 5 matches our output. If not, stop the cell and go back to your code!

```
In [ ]: _, _, parameters = model(X_train, Y_train, X_test, Y_test)
```

Expected output: although it may not match perfectly, your expected output should be close to ours and your cost value should decrease.

Cost after epoch 0 =

```
<td>
  1.917929
</td>
```

Cost after epoch 5 =

```
<td>
  1.506757
</td>
```

Train Accuracy =

```
<td>
  0.940741
</td>
```

Test Accuracy =

```
<td>
  0.783333
</td>
```