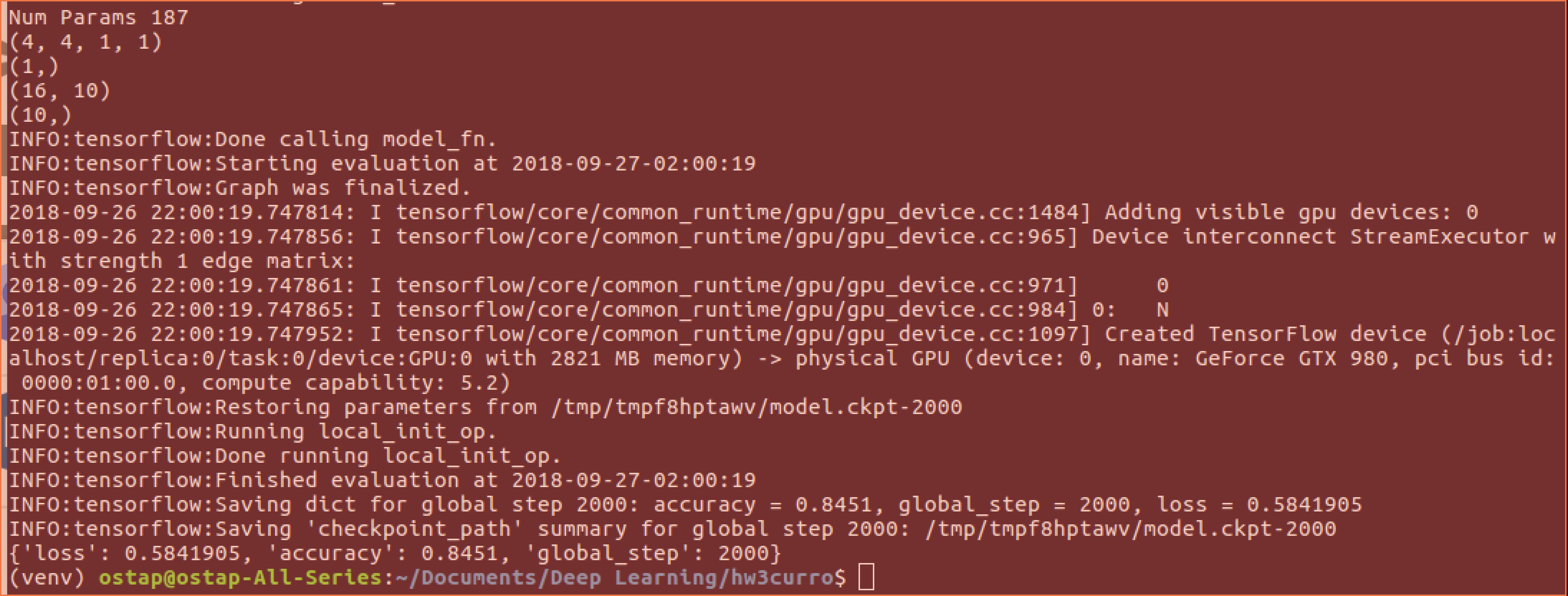
```
#!/bin/python3.6
#Ostap Voynarovskiy
#CGML HW3
#Sept 26 2018
#Professor Curro

import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import logging
from tqdm import tqdm
import os

#os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

# Define globals and import dataset
mnist = tf.keras.datasets.mnist
(x_train,y_train),(x_test,y_test) = mnist.load_data()
x_train,x_test = x_train/255.0 ,x_test/255.0            # go from 0-1 instead of 0-255 8 bit greyscale
y_train = y_train.astype('int32')                                # fix due to tensorflow complaining
y_test = y_test.astype('int32')                                  # fix due to tensorflow complaining

#import pdb; pdb.set_trace() # debug mode
'''def cutData(f, cut):
        l,w,h = f.shape
        s,e = cut,w-cut #assumes square
        fo = f[:,s:e,s:e]
        print (fo.shape)
        return fo
'''
def cutData(f, cutx,cuty):
        l,w,h = f.shape
        sx,ex = cutx,w-cutx #assumes square
        sy,ey = cuty,w-cuty #assumes square
        fo = f[:,sy:ey,sx:ex]
        print (fo.shape)
        return fo


def genTrainAndVal(f,l): #split the features and labels of the training data 80:20 train and validation
        valPercent=20
        lx,_,_=f.shape

        z = f.shape[0]          # 60000 hopefully
        s = np.arange(z)
        np.random.shuffle(s)
        fs = f[s]                       # features shuffled
        ls = l[s]                       # labels shuffled
        lx = f.shape[0]         # len of the features
        nv = int( lx *.2)       # num validation samp
        print (fs[nv:].shape, ls[nv:].shape, fs[:nv].shape, ls[:nv].shape)
        return fs[nv:], ls[nv:], fs[:nv], ls[:nv]

#send this the output of only the first 2 returned val of genTrainAndVal
def getBatch(feat,lab):
        l,_,_= feat.shape
        choices = np.random.choice(l, size=BATCH_SIZE)
        return feat[choices],lab[choices]

def cnn(features,labels,mode):
        #layer 0

        h,w = 20,20
        L0 = tf.reshape(features["x"], [-1, h,w, 1]) #the -1 makes it guess how big its supposed to be
```

```
                                                                        #[numExaples,height,width,channels]
        # hyperParams
        k = 4
        n = 5
        l = 0.002
        fil = 1
        dr  = .1
        # conv layer 1
        conv1 = tf.layers.conv2d(inputs=L0,filters=fil,kernel_size=[k, k], padding="same", activation=tf.nn.elu)
        # pool layer 1
        pool1 = tf.layers.max_pooling2d(inputs=conv1, pool_size=[n, n], strides=n)
        # flatten the layer for fully connected layer
        poolInt = int(w/n*h/n) * fil
        pFlat = tf.reshape(pool1, [-1, poolInt])
        # fully connected layer
        #dense = tf.layers.dense(inputs=pFlat, units=32, activation=tf.nn.elu)
        # dropout layer
        dropout = tf.layers.dropout(inputs=pFlat, rate=dr, training=mode == tf.estimator.ModeKeys.TRAIN)
        # logits layer layer that has the 10 outputs
        logits = tf.layers.dense(inputs=dropout, units=10)

        predictions = {
                "classes": tf.argmax(input=logits, axis=1),
                "probabilities": tf.nn.softmax(logits, name="softmax_tensor")}

        if mode == tf.estimator.ModeKeys.PREDICT:
                return tf.estimator.EstimatorSpec(mode=mode, predictions=predictions)

        num_params = np.sum([np.prod(v.get_shape().as_list()) for v in tf.trainable_variables()])
        print("Num Params",num_params)

        for v in tf.trainable_variables():
                print(v.get_shape())

        loss = tf.losses.sparse_softmax_cross_entropy(labels=labels, logits=logits) + l*tf.reduce_sum([tf.nn.l2_loss(tV) for tV in tf.trainable_variables()])

        if mode == tf.estimator.ModeKeys.TRAIN:
                optimizer = tf.train.AdamOptimizer(learning_rate=0.001)
                train_op = optimizer.minimize(loss=loss,global_step=tf.train.get_global_step())
                return tf.estimator.EstimatorSpec(mode=mode, loss=loss, train_op=train_op)

        eval_metric_ops = {      "accuracy": tf.metrics.accuracy(labels=labels, predictions=predictions["classes"])}
        return tf.estimator.EstimatorSpec(mode=mode, loss=loss, eval_metric_ops=eval_metric_ops)


def main():
        mnist_classifier = tf.estimator.Estimator(model_fn=cnn)#,model_dir="./tmp/modelCheckpoint"
        #tensors_to_log = {"probabilities": "softmax_tensor"}
        #logging_hook = tf.train.LoggingTensorHook(tensors=tensors_to_log, every_n_iter=50)
        tf.logging.set_verbosity(tf.logging.INFO)

        train_input_fn = tf.estimator.inputs.numpy_input_fn(x={"x": tx},y=ty,batch_size=100,
                num_epochs=None,shuffle=True)

        mnist_classifier.train(input_fn=train_input_fn,steps=2000)#,hooks=[loggi
ng_hook])

        eval_input_fn = tf.estimator.inputs.numpy_input_fn(x={"x": vx},y=vy,
                num_epochs=1,shuffle=False)

        eval_results = mnist_classifier.evaluate(input_fn=eval_input_fn)
        print(eval_results)

        #I did not have this here when I was tuning hyper Parameters
        eval_input_fn = tf.estimator.inputs.numpy_input_fn(x={"x": x_test},y=y_t
est,
                num_epochs=1,shuffle=False)

        eval_results = mnist_classifier.evaluate(input_fn=eval_input_fn)
        print(eval_results)




cutx = 4
cuty = 4
tx,ty,vx,vy = genTrainAndVal(x_train,y_train)
tx=cutData(tx,cutx,cuty)
vx=cutData(vx,cutx,cuty)
x_test = cutData(x_test,cutx,cuty)
print(tx.shape,’hi’)
main()

#end of code

# just for shits
def plotVal():
        len,_,_ = x_train.shape

        rn = np.random.randint(0,len-1)
        #rn = 26563
        print(len,rn)
        test_val=x_train[rn]

        fig1= plt.figure(1)
        dr=4 #pixels top bottom left and right to drop
        xc,yc = np.linspace(0+dr,27-dr,28-2*dr),np.linspace(27-dr,0+dr,28-2*dr)
        xv,yv = np.meshgrid(xc,yc)

        print(type(y_train))

        #reduce dimentions of the test data
        f,l = 0+dr, 28-dr
        z = test_val[f:l,f:l]

        CS = plt.contourf(xv,yv,z,cmap=’gray’)
        w= plt.xlabel(’w’)
        h= plt.ylabel(’h’)
        h.set_rotation(0)
        plt.title("MNIST")
        plt.axis(’equal’)
        #plt.clabel(CS, fontsize=9, inline=1)
        plt.show()
```



```
Num Params 187
(4, 4, 1, 1)
(1,)
(16, 10)
(10,)
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Starting evaluation at 2018-09-27-02:00:19
INFO:tensorflow:Graph was finalized.
2018-09-26 22:00:19.747814: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1484] Adding visible gpu devices: 0
2018-09-26 22:00:19.747856: I tensorflow/core/common_runtime/gpu/gpu_device.cc:965] Device interconnect StreamExecutor w
ith strength 1 edge matrix:
2018-09-26 22:00:19.747861: I tensorflow/core/common_runtime/gpu/gpu_device.cc:971]      0
2018-09-26 22:00:19.747865: I tensorflow/core/common_runtime/gpu/gpu_device.cc:984] 0:   N
2018-09-26 22:00:19.747952: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1097] Created TensorFlow device (/job:loc
alhost/replica:0/task:0/device:GPU:0 with 2821 MB memory) -> physical GPU (device: 0, name: GeForce GTX 980, pci bus id:
 0000:01:00.0, compute capability: 5.2)
INFO:tensorflow:Restoring parameters from /tmp/tmpf8hptawv/model.ckpt-2000
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Finished evaluation at 2018-09-27-02:00:19
INFO:tensorflow:Saving dict for global step 2000: accuracy = 0.8451, global_step = 2000, loss = 0.5841905
INFO:tensorflow:Saving 'checkpoint_path' summary for global step 2000: /tmp/tmpf8hptawv/model.ckpt-2000
{'loss': 0.5841905, 'accuracy': 0.8451, 'global_step': 2000}
(venv) ostap@ostap-All-Series:~/Documents/Deep Learning/hw3curro$
```