In [1]:

```
import os
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime
```

In [2]:

```
import tensorflow as tf
from tensorflow.contrib.layers import fully_connected,dropout,l2_regularizer
```

In [3]:

```
x = tf.placeholder('float', [None, 4096])
y = tf.placeholder('float')
hl = tf.placeholder('float') #hinge_loss
```

In [4]:

```
def get_video_list(path):
    videos=[]
    with open(path, 'r') as f:
        for line in f:
            videos.append(line.strip())
    return videos
```

In [5]:

```
def get_training_set(path):

    abnormal_path =path

    normal_path = path

    batchsize=60
    batch_Ab_size = 30

    num_abnormal = 170

    num_normal =160

    abnorm_list = np.random.permutation(num_abnormal)
    abnorm_list=abnorm_list[:batch_Ab_size]

    norm_list = np.random.permutation(num_normal)
    norm_list = norm_list[:batch_Ab_size]

    #print("Loading abnornmal Features...")
    videos = get_video_list(abnormal_path+"/anomaly.txt")

    abNormal_features =[]

    abnormal_features=[]
    for i in abnorm_list:
        vid_path=os.path.join(abnormal_path,videos[i] )
        with open(vid_path,'r') as f:
            lines = f.read().splitlines()
        for line in lines:
            abNormal_features.append(np.float32(line.split()))

    abNormal_features=np.array(abNormal_features)

    #print("Abnoraml features videos loded suceesfully.")

    #print("Loading normal Features...")

    videos = get_video_list(abnormal_path+"/normal.txt")
    normal_features=[]

    for i in norm_list:
        vid_path=os.path.join(normal_path,videos[i] )
        if (os.path.isfile(vid_path) ):
            with open(vid_path,'r') as f:
                lines = f.read().splitlines()
            for line in lines:
                  normal_features.append(np.float32(line.split()))

    normal_features=np.array(normal_features)

    #print("Normal Features Loaded successfully")
    return abNormal_features, normal_features
```

In [6]:

```
Anomaly,Normal=get_training_set("./out")
```

In [7]:

```
n_hidden1 = 512
n_hidden2 = 32
n_output = 1
batch_size = 32
```

In [8]:

```
def dnn_network(X):
    tf_h1 = {'weights':tf.Variable(tf.random_normal([4096, n_hidden1])), #4096 f
eature vector
                      'biases':tf.Variable(tf.random_normal([n_hidden1]))}

    tf_h2 = {'weights':tf.Variable(tf.random_normal([n_hidden1, n_hidden2])),
                      'biases':tf.Variable(tf.random_normal([n_hidden2]))}

    tf_op = {'weights':tf.Variable(tf.random_normal([n_hidden2, n_output])),
                    'biases':tf.Variable(tf.random_normal([n_output])),}


    l1 = tf.add(tf.matmul(X,tf_h1['weights']), tf_h1['biases'])
    l1 = tf.nn.relu(l1)
    l1 = tf.layers.dropout(l1, 0.6)


    l2 = tf.add(tf.matmul(l1,tf_h2['weights']), tf_h2['biases'])
    l2  = tf.layers.dropout(l2, 0.6)


    output = tf.matmul(l2,tf_op['weights']) + tf_op['biases']
    output = tf.nn.sigmoid(output)


    return output , tf_h1['weights'] , tf_h2['weights'] , tf_op['weights']
```

In [9]:

```
output , weights_1 , weights_2, weights_3 = dnn_network(x)
cost = 0.001 * (tf.nn.l2_loss(weights_1) + tf.nn.l2_loss(weights_2) + tf.nn.l2_l
oss(weights_3)) + hl
optimizer = tf.train.AdagradOptimizer(0.001).minimize(cost)
```

In [10]:

```
saver = tf.train.Saver()  # to save model
```

In [11]:

```
# Stoering details for tensorboard
tf.summary.scalar('cost', cost)
tf.summary.histogram('h_w1', weights_1)
tf.summary.histogram('h_w2',weights_2)
tf.summary.histogram('h_w3',weights_3)
merged_summary_op = tf.summary.merge_all()
```

In [12]:

```
# Initialize and run
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)
```

In [25]:

```
epochs=501
print_epoch =20
save_model_epoch=500
count=0
```

In [26]:

```
#los =[] #loss array

train_writer = tf.summary.FileWriter("./logs/Dnn_session",tf.get_default_graph
())

for epoch in range(epochs):

    i=0
    anomaly,normal = get_training_set("./out") # getting data

    while i < len(anomaly):

        start = i

        end =i+batch_size

        batch_x =np.array(anomaly[start:end])
        batch_y =np.array(normal[start:end])  # Single Video of normal and anomo
lus


        anomaly_score = sess.run(output,feed_dict={x: batch_x})
        normal_score,W1= sess.run([output,weights_1],feed_dict={x: batch_y})  #
 calaculated score of anomolus and normal video

        # calculation of hinge loss----------------------------------------------
----
        anomaly_score = anomaly_score.flatten()
        normal_score = normal_score.flatten()

        l = max(0.0,(1-anomaly_score.max()+normal_score.max())) #loss implementa
tion

        add = 0.0
        for index in range(len(anomaly_score) - 1):
            add += (anomaly_score[index] - anomaly_score[index+1]) ** 2

        final_cost = l + (add*1.0 + anomaly_score.sum()) * 0.00008

        # loss completd----------------------------------------------------------
-


        o,_,cst,summ = sess.run([optimizer,output,cost,merged_summary_op], feed_
dict={hl : final_cost,x: batch_x})


        i =i + batch_size

        train_writer.add_summary(summ,count) # summaries for tensorBoard

        count += 1
    #los.append(cst)

    if epoch %print_epoch ==1:
        print('Epoch', epoch, 'completed out of',epochs,'loss:',cst)

    if epoch%save_model_epoch == 1:
        saved_path = saver.save(sess, './model_logs/'+str(epoch)+'/model', globa
l_step=epoch)
```

```
```

```
Epoch 1 completed out of 2 loss: 293.29483
```

In [15]:

```
import matplotlib.pyplot as plt
```

In [21]:

```
plt.plot(los)
```

Out[21]:

```
[<matplotlib.lines.Line2D at 0x7f8bc78f97b8>]
```