Project 4 Report

Deep Learning

Submitted by
Hima Sujani Adike, 50246828
Anjali Sujatha Nair ,50248735
Soumya Venkatesan, 50246599

Problem Statement

The purpose of the project was to train the system on celebrity dataset A and make it recognize the pictures with eye glasses

Approaches

Used convoluted neural network available from tensor flow package to trained celebrity dataset with back propagation. Adam optimizer was used for optimizing. Achieved an accuracy of 92.842 on a dataset of 50000. The dataset is tested against the higher resolution images yields an accuracy of 89.99

Implementation

All code resides in Main.py, the main file to be executed. Loads the celebrity dataset and runs the algorithm.

Main.py

Class to be called for the execution of the algorithm. Main methods are getImageData, getLabels, create_convoluted_multilayer_perceptron1, train_and_test_convNN. The methods getImageData and getLabels are used for loading the images and the labels respectively. In the method create_convoluted_multilayer_perceptron1 we initialize the values with some biases and use a small convolution neural network whichconsists of colvolution and pooling which will help to reduce the dimensionality of feature map and use adropout to avoid overfitting the data and we update the weight vectors using adamoptimizer.

Images are preprocessed before training as the dataset consists of images of 176x218 pixels it is reduced to 28x28 and flattened to [-1,28,28,3] as the images are colored ones with RGB values we flatten accordingly.

The images are read in batches to avoid disk space and memory space issues and trained the data in batches, in which the images were flattened weights are calculated for every 5x5 patch features and validated the model against validation data after every 100 epochs to verify whether the data is overfitting and the parameters are checked for all different values of learning rates 1e-4 and 1e-2..

The tf.nn.dropout function is used to regularize the the model to avoid overfitting the data which is available with tensor flow library and created convoluted layers and flatten the data and applied the softmax function to decide the output.

Though there is perpetual loss in data by reducing the resolution for images the accuracy for the lower resolution images is higher than the images with higher resolution. As to identify the object in higher resolution is difficult, the accuracy is lower for these images.

There are 2 classes for the dataset the person not keeping the glasses with value -1 and keeping the glasses with a value of 1.

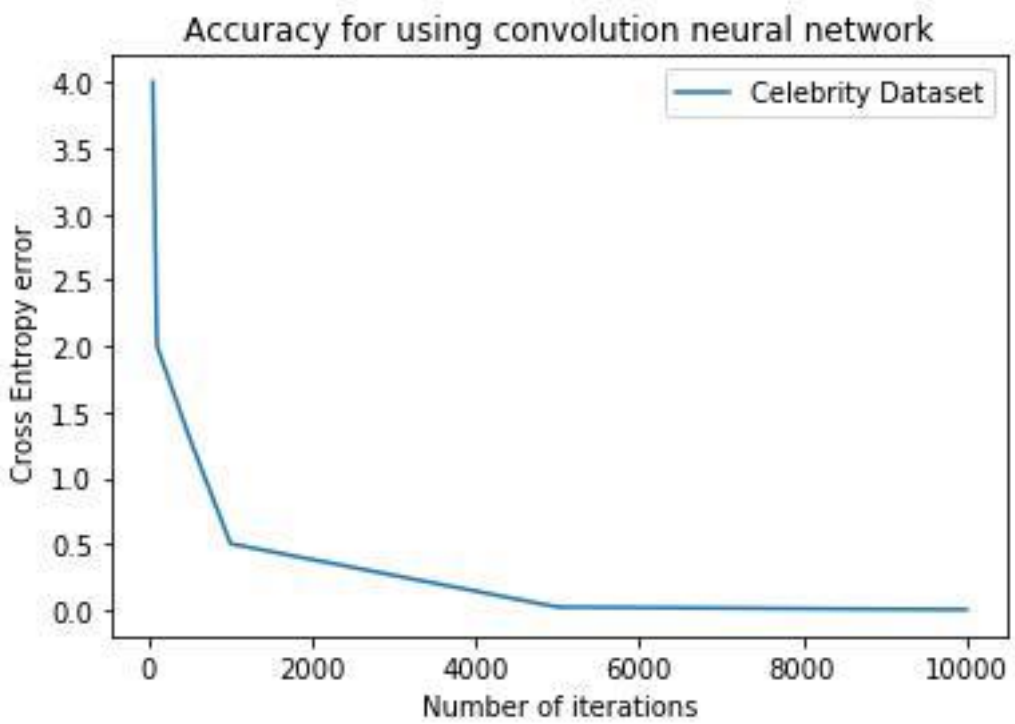
```
def train_and_test_convNN1(labels):
   learning_rate = 0.5
   training_epochs = 1000
   batch_size = 50
   pred,x,y,keep_prob = create_convoluted_multilayer_perceptron1()
   cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=pred, labels=y))
   optimizer = tf.train.AdamOptimizer(1e-4).minimize(cost)
   init = tf.global_variables_initializer()
   testing_labels=labels[162599:172599]
   testing_data=getImageData(162599,10000)
   batch_start=0
   with tf.Session() as sess:
      sess.run(init)
      for epoch in range(training_epochs):
         batch_x=getImageData(batch_start,batch_size)
         batch_y=labels[batch_start:batch_start+batch_size]
         #print(len(batch_x),len(batch_y),batch_start,labels[batch_start])
          if(epoch%100!=0):
                  c = optimizer.run(feed_dict={x: batch_x, y: batch_y, keep_prob: 0.5})
          else:
                  c=tf.equal(tf.argmax(pred,1),tf.argmax(y,1))
                  a=tf.reduce_mean(tf.cast(c,"float"))
                  print("accuracy against validation
data",accuracy.eval({x:batch_x,y:batch_y,keep_prob:1.0})*100 )
         batch_start+=batch_size
      #print('done')
      correct_prediction = tf.equal(tf.argmax(pred, 1), tf.argmax(y, 1))
      accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
      print("Accuracy for Celebrity test data for convoluted Neural network trained data:", accuracy.eval({x:
testing_data, y: testing_labels, keep_prob: 1.0})*100)
```

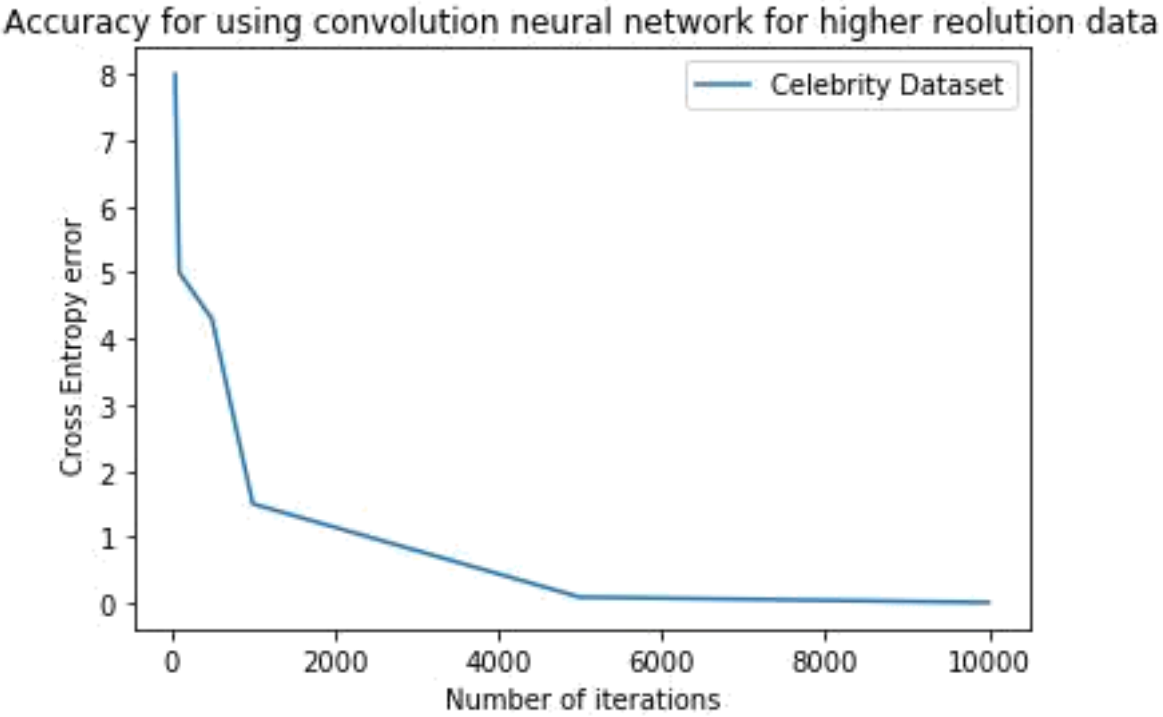
For images with higher resolution , used this method.

```
def train_and_test_convNN2(labels):
   training_epochs = 50
   batch_size = 50
   pred,x,y,keep_prob = create_convoluted_multilayer_perceptron2()
   cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=pred, labels=y))
   optimizer = tf.train.AdamOptimizer(1e-4).minimize(cost)
   init = tf.global_variables_initializer()
   testing_labels=labels[162599:172599]
   testing_data=getImageData(162599,10000)
   batch_start=0
   with tf.Session() as sess:
      sess.run(init)
      for epoch in range(training_epochs):
         batch_x=getImageData(batch_start,batch_size)
         batch_y=labels[batch_start:batch_start+batch_size]
         #print(len(batch_x),len(batch_y),batch_start,labels[batch_start])
```

```
            if epoch!=100:
                c = optimizer.run(feed_dict={x: batch_x, y: batch_y, keep_prob: 0.5})
            else:
                c=tf.equal(tf.argmax(pred,1),tf.argmax(y,1))
                a=tf.reduce_mean(tf.cast(correct_prediction, "float"))
                        print("Accuracy for Celebrity test data for convoluted Neural network trained data:",
accuracy.eval({x: testing_data, y: testing_labels, keep_prob: 1.0})*100)
            batch_start+=batch_size
        #print('done')
        correct_prediction = tf.equal(tf.argmax(pred, 1), tf.argmax(y, 1))
        accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
        print("Accuracy for Celebrity test data for convoluted Neural network trained data:", accuracy.eval({x:
testing_data, y: testing_labels, keep_prob: 1.0})*100)
```

Plots




Accuracy for using convolution neural network for higher reolution data
Celebrity Dataset
Cross Entropy error
Number of iterations
0
1
2
3
4
5
6
7
8
0
2000
4000
6000
8000
10000