ECE 763 Computer Vision

Project-2 Report

Priya Diwakar

Student ID 200205361

Face image classification

using a simple neural network to get familiar with steps of training neural networks

TABLE OF CONTENTS

Dataset: http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

The dataset contains more than 200K images with its respective bounding box face annotation.

Training Dataset:

Face: 10000 images from the dataset are used for training the model. After cropping out the bounding box such that only faces are kept, the images are resized to 60 x 60 x 3-pixel size.

Non-face: 10000 images for training are formed by cropping out 60 x 60 x 3 regions from the background of the original training face images.

Test Dataset:

Face: 1000 images from the dataset are used for testing the model. After cropping out the bounding box such that only faces are kept, the images are resized to 60 x 60 x 3-pixel size.

Non-face: 1000 images for testing are formed by cropping out 60 x 60 x 3 regions from the background of the original testing face images.

Validation Dataset:

Face: 1000 images from the dataset are used for validating the model. After cropping out the bounding box such that only faces are kept, the images are resized to 60 x 60 x 3-pixel size.

Non-face: 1000 images for validation are formed by cropping out 60 x 60 x 3 regions from the background of the original validation face images.

```
26          img=cv2.imread(l[0])
27          x_1=int(l[1])
28          y_1=int(l[2])
29          w=int(l[3])
30          h=int(l[4])
31          #print (l[0])
32          crop_img=img[y_1:y_1+h,x_1:x_1+w]
33          resized_image = cv2.resize(crop_img, (60, 60),interpolation = cv2.INTER_AREA)
34#      #Set path to the folder where you want to write the cropped face images
35          os.chdir(path/"face")
36          cv2.imwrite(l[0], resized_image)
37          os.chdir(path/"nonface")
38          newimg=img[0:60,0:60,:]
39          cv2.imwrite(l[0],newimg)
```

Fig 1. Preparing Data

Step 1: Preprocess Data

Perform preprocessing on the data to get zero-centered data and normalized data

```
113
114 """
115 Preprocess data to get zero centered and normalized data
116
117 """
118 scaler = StandardScaler()
119 scaler.fit(train_data)
120 StandardScaler(copy=True, with_mean=True, with_std=True)
121 train_data = scaler.transform(train_data)
122 test_data = scaler.transform(test_data)
123 valid_data = scaler.transform(valid_data)
124
```

Fig 2. Preprocess data

Training and Testing Accuracy was observed for both preprocessed and non-preprocessed data. There was an increase in accuracy for preprocessed data. The results were compared for a neural network of single hidden layer of 1024 nodes and training batch size of 128. Learning rate of 0.001 and regularization factor (lambda) of 0.01.

Minibatch loss at step 3000: 47315.41796875	Minibatch loss at step 3000: 6768.2822265625
Minibatch accuracy: 89.8	Minibatch accuracy: 96.9
Validation accuracy: 83.3	Validation accuracy: 85.0
Test accuracy: 83.5	Test accuracy: 85.5
(a) No preprocessing	(b) With preprocessing

Fig 3. Compare accuracy

Step 2: Choose the Architecture

Initially with a simple neural network of 1 Hidden layer only with 1000 nodes and training batch size of 100

```
    tf_train_dataset = tf.placeholder('float', shape=(batch_size, size))
    tf_train_labels = tf.placeholder('float', shape=(batch_size, num_labels))
    tf_valid_dataset = tf.constant(valid_data)
    tf_test_dataset = tf.constant(test_data)

    # Variables.
    weights_1 = tf.Variable(tf.truncated_normal([size, num_nodes]))
    biases_1 = tf.Variable(tf.zeros([num_nodes]))
    weights_2 = tf.Variable(tf.truncated_normal([num_nodes, num_labels]))
    biases_2 = tf.Variable(tf.zeros([num_labels]))
```

Fig 4. Define the network weights

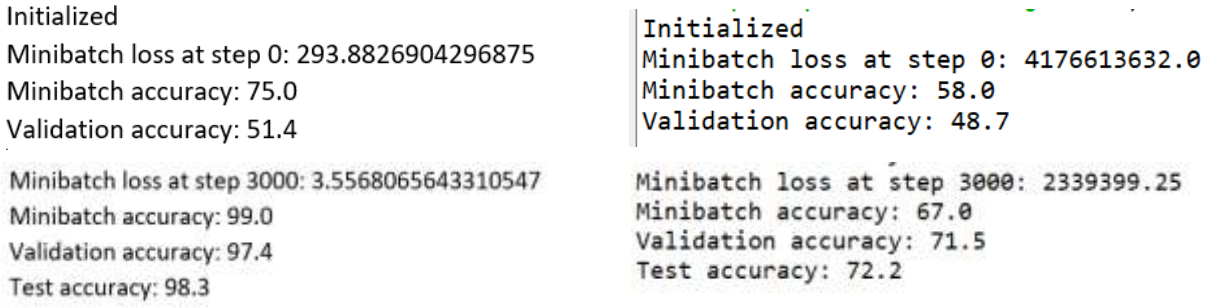
Babysitting process

```

# Training computation.
logits_1 = tf.matmul(tf_train_dataset, weights_1) + biases_1
relu_layer= tf.nn.relu(logits_1)
logits_2 = tf.matmul(relu_layer, weights_2) + biases_2
# Normal loss function
loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=logits_2, labels=tf_train_labels))
optimizer = tf.train.AdamOptimizer().minimize(loss)

# Predictions for the training
train_prediction = tf.nn.softmax(logits_2)

# Predictions for validation
logits_1 = tf.matmul(tf_valid_dataset, weights_1) + biases_1
relu_layer= tf.nn.relu(logits_1)
logits_2 = tf.matmul(relu_layer, weights_2) + biases_2

valid_prediction = tf.nn.softmax(logits_2)

# Predictions for test
logits_1 = tf.matmul(tf_test_dataset, weights_1) + biases_1
relu_layer= tf.nn.relu(logits_1)
logits_2 = tf.matmul(relu_layer, weights_2) + biases_2

test_prediction =  tf.nn.softmax(logits_2)
```

Throughout the project,
Activation function used was ReLU and Adam Optimizer and loss calculated using softmax cross entropy

Fig 5. Predictions computations

Compare regularization effect



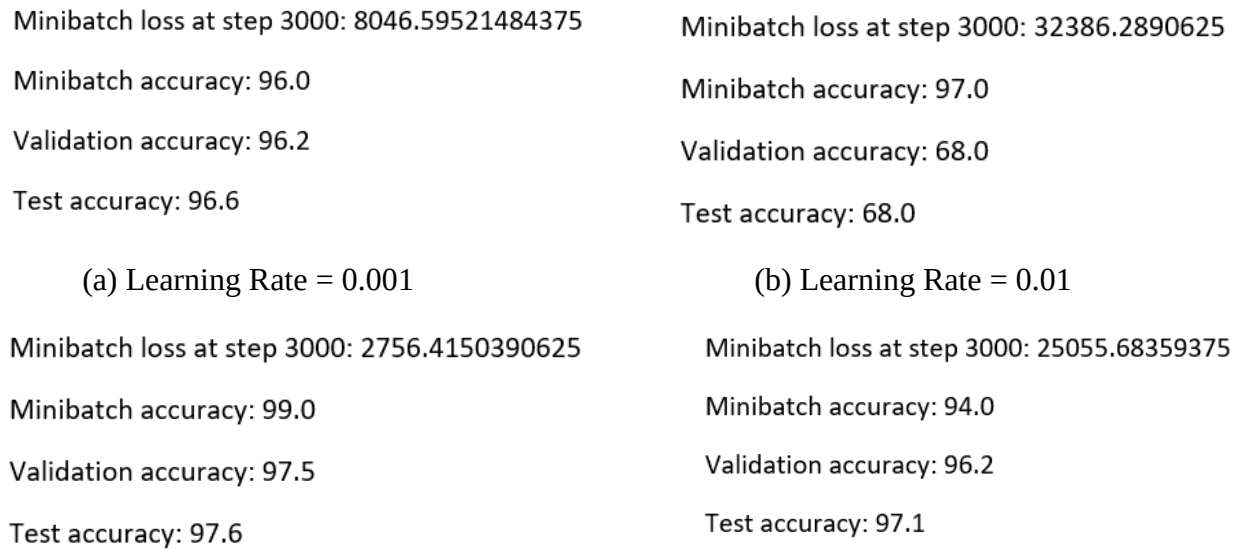
(a) No regularization (b) Lambda =1e3 for regularization factor

Fig 6. Regularization, Learning rate was 0.001

Loss went up as regularization factor was increased. Also there is quite a drop in accuracy.

Learning Rate

Learning rate was changed and accuracy was observed.



(a) Learning Rate = 0.001 (b) Learning Rate = 0.01

(c) Learning Rate = 0.001 (d) Learning Rate = 0.01

Fig 7. Regularization factor (lambda) = 0.01 for (a) and (b)
Regularization factor (lambda) = 0.001 for (c) and (d)

Learning rate starting at 0.5 with decay rate of 0.96 these observations were obtained

Learning rate: 0.500000
Minibatch loss at step 0: 43007.8984375
Minibatch accuracy: 17.0
Validation accuracy: 52.1
Learning rate: 0.480000
Minibatch loss at step 500: 13787915.0
Minibatch accuracy: 46.0
Validation accuracy: 64.7
Learning rate: 0.460800
Minibatch loss at step 1000: 30334266.0
Minibatch accuracy: 52.0
Validation accuracy: 84.8
Learning rate: 0.442368
Minibatch loss at step 1500: 2638541.0
Minibatch accuracy: 95.0
Validation accuracy: 78.4
Learning rate: 0.424673
Minibatch loss at step 2000: 2134269.25
Minibatch accuracy: 94.0
Validation accuracy: 79.3
Learning rate: 0.407686
Minibatch loss at step 2500: 844442.5
Minibatch accuracy: 98.0
Validation accuracy: 77.2
Learning rate: 0.391379
Minibatch loss at step 3000: 332023.3125
Minibatch accuracy: 99.0
Validation accuracy: 61.8
Test accuracy: 61.2

With a high learning rate the accuracy is low and loss is high
It is observed Learning Rate of 0.001 gives best accuracy among these values.

For this neural network architecture of 1 hidden layer highest test accuracy obtained was 98.3% for no regularization and learning rate =0.001.

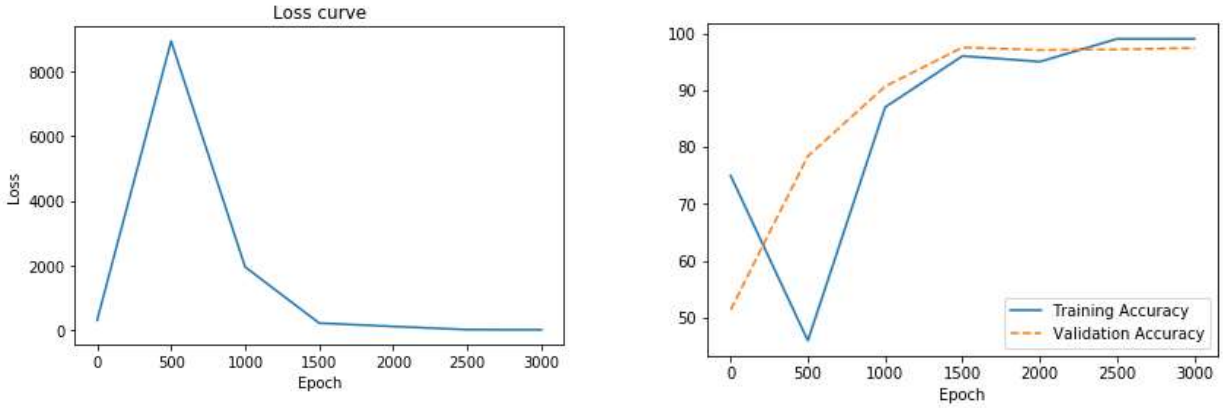


Fig 8. One Hidden Layer with 1000 nodes
Test accuracy = 98.3%

For two Hidden Layers Neural Network

```
159
160 # Variables.
161 weights_1 = tf.Variable(tf.truncated_normal([size,hidden_nodes_1 ]))
162 biases_1 = tf.Variable(tf.zeros([hidden_nodes_1]))
163 weights_2 = tf.Variable(tf.truncated_normal([hidden_nodes_1, hidden_nodes_2]))
164 biases_2 = tf.Variable(tf.zeros([hidden_nodes_2]))
165 # Output Layer
166 weights_3 = tf.Variable(tf.truncated_normal([hidden_nodes_2, num_labels]))
167 biases_3 = tf.Variable(tf.zeros([num_labels]))
168
169
170 # Hidden RELU Layer 1
171 logits_1 = tf.matmul(tf_train_dataset, weights_1) + biases_1
172 hidden_layer_1 = tf.nn.relu(logits_1)
173
174
175 # Hidden RELU Layer 2
176 logits_2 = tf.matmul(hidden_layer_1, weights_2) + biases_2
177 hidden_layer_2 = tf.nn.relu(logits_2)
178
179
180 # Output Layer
181 logits_3 = tf.matmul(hidden_layer_2, weights_3) + biases_3
182
183 # Normal Loss function
184 loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=logits_3, labels=tf_train_labels))
185 # Loss function with L2 Regularization
186 regularizers = tf.nn.l2_loss(weights_1) + tf.nn.l2_loss(weights_2) + \
187                tf.nn.l2_loss(weights_3)
188 loss = tf.reduce_mean(loss + beta * regularizers)
189
190 optimizer = tf.train.AdamOptimizer().minimize(loss)
```

Fig 9. Define network weight and architecture for 2 Hidden layers

```
191
192 # Predictions for the training
193 train_prediction = tf.nn.softmax(logits_3)
194
195 # Predictions for validation
196 valid_logits_1 = tf.matmul(tf_valid_dataset, weights_1) + biases_1
197 valid_relu_1 = tf.nn.relu(valid_logits_1)
198
199 valid_logits_2 = tf.matmul(valid_relu_1, weights_2) + biases_2
200 valid_relu_2 = tf.nn.relu(valid_logits_2)
201
202 valid_logits_3 = tf.matmul(valid_relu_2, weights_3) + biases_3
203
204
205 valid_prediction = tf.nn.softmax(valid_logits_3)
206
207 # Predictions for test
208 test_logits_1 = tf.matmul(tf_test_dataset, weights_1) + biases_1
209 test_relu_1 = tf.nn.relu(test_logits_1)
210
211 test_logits_2 = tf.matmul(test_relu_1, weights_2) + biases_2
212 test_relu_2 = tf.nn.relu(test_logits_2)
213
214 test_logits_3 = tf.matmul(test_relu_2, weights_3) + biases_3
215
216
217 test_prediction = tf.nn.softmax(test_logits_3)
218
```

Fig 10. Predictions computations

A second simple neural network of 2 Hidden layers with 50 and 25 nodes and training batch size of 100 was trained and results were observed for
Learning Rate = 0.001 and Regularization factor (lambda) = 0.01
Testing accuracy of 95 % was obtained.

Initialized
Minibatch loss at step 0: 2115.986328125
Minibatch accuracy: 97.0
Validation accuracy: 50.0
Minibatch loss at step 200: 2317.554931640625
Minibatch accuracy: 64.0
Validation accuracy: 80.7
Minibatch loss at step 400: 2222.820556640625
Minibatch accuracy: 62.0
Validation accuracy: 82.2
Minibatch loss at step 600: 2453.8408203125
Minibatch accuracy: 71.0
Validation accuracy: 81.0
Minibatch loss at step 800: 1992.5531005859375
Minibatch accuracy: 79.0
Validation accuracy: 85.6
Minibatch loss at step 1000: 1840.5462646484375
Minibatch accuracy: 74.0
Validation accuracy: 88.3
Minibatch loss at step 1200: 1644.425537109375
Minibatch accuracy: 85.0
Validation accuracy: 90.8
Minibatch loss at step 1400: 1592.3096923828125
Minibatch accuracy: 84.0
Validation accuracy: 91.8
Minibatch loss at step 1600: 1519.7130126953125
Minibatch accuracy: 90.0
Validation accuracy: 92.0
Minibatch loss at step 1800: 1452.201904296875
Minibatch accuracy: 84.0
Validation accuracy: 92.8
Minibatch loss at step 2000: 1368.598388671875
Minibatch accuracy: 90.0
Validation accuracy: 92.8
Minibatch loss at step 2200: 1288.3709716796875
Minibatch accuracy: 94.0
Validation accuracy: 92.1
Minibatch loss at step 2400: 1226.506591796875
Minibatch accuracy: 85.0
Validation accuracy: 93.7
Minibatch loss at step 2600: 1146.2833251953125
Minibatch accuracy: 93.0
Validation accuracy: 93.8
Minibatch loss at step 2800: 1076.744384765625
Minibatch accuracy: 90.0
Validation accuracy: 94.1
Minibatch loss at step 3000: 1003.8374633789062
Minibatch accuracy: 84.0
Validation accuracy: 94.0
Test accuracy: 95.0

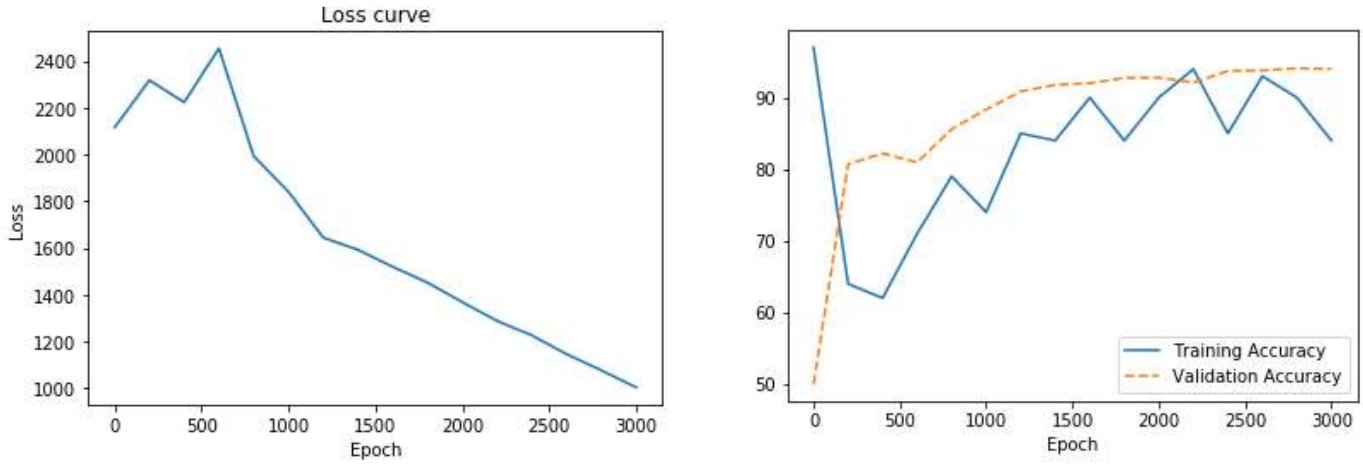


Fig 11. Two Hidden Layers with 50 and 25 nodes
Test accuracy = 95 %

A third simple neural network of 2 Hidden layers with 2000 and 1000 nodes and training batch size of 100 was trained and results were observed for
Learning Rate = 0.001 and Regularization factor (lambda) = 0.01
Testing accuracy of 98.1 % was obtained.

Initialized
Minibatch loss at step 0: 148128.9375
Minibatch accuracy: 4.0
Validation accuracy: 51.4
Minibatch loss at step 200: 421059.84375
Minibatch accuracy: 17.0
Validation accuracy: 63.9
Minibatch loss at step 400: 281575.96875
Minibatch accuracy: 52.0
Validation accuracy: 76.7
Minibatch loss at step 600: 322211.6875
Minibatch accuracy: 52.0
Validation accuracy: 75.2
Minibatch loss at step 800: 231896.703125
Minibatch accuracy: 63.0
Validation accuracy: 81.6
Minibatch loss at step 1000: 221773.609375
Minibatch accuracy: 74.0
Validation accuracy: 84.2
Minibatch loss at step 1200: 155260.5625
Minibatch accuracy: 71.0
Validation accuracy: 88.7
Minibatch loss at step 1400: 122696.7421875
Minibatch accuracy: 84.0
Validation accuracy: 91.7
Minibatch loss at step 1600: 95274.1796875
Minibatch accuracy: 91.0
Validation accuracy: 94.2
Minibatch loss at step 1800: 93604.8359375

Minibatch accuracy: 93.0
Validation accuracy: 95.8
Minibatch loss at step 2000: 89518.3515625
Minibatch accuracy: 96.0
Validation accuracy: 96.3
Minibatch loss at step 2200: 87143.0859375
Minibatch accuracy: 98.0
Validation accuracy: 97.0
Minibatch loss at step 2400: 85454.53125
Minibatch accuracy: 98.0
Validation accuracy: 97.3
Minibatch loss at step 2600: 86446.703125
Minibatch accuracy: 97.0
Validation accuracy: 97.3
Minibatch loss at step 2800: 84319.6015625
Minibatch accuracy: 97.0
Validation accuracy: 97.4
Minibatch loss at step 3000: 82522.2734375
Minibatch accuracy: 99.0
Validation accuracy: 97.5
Test accuracy: 98.1

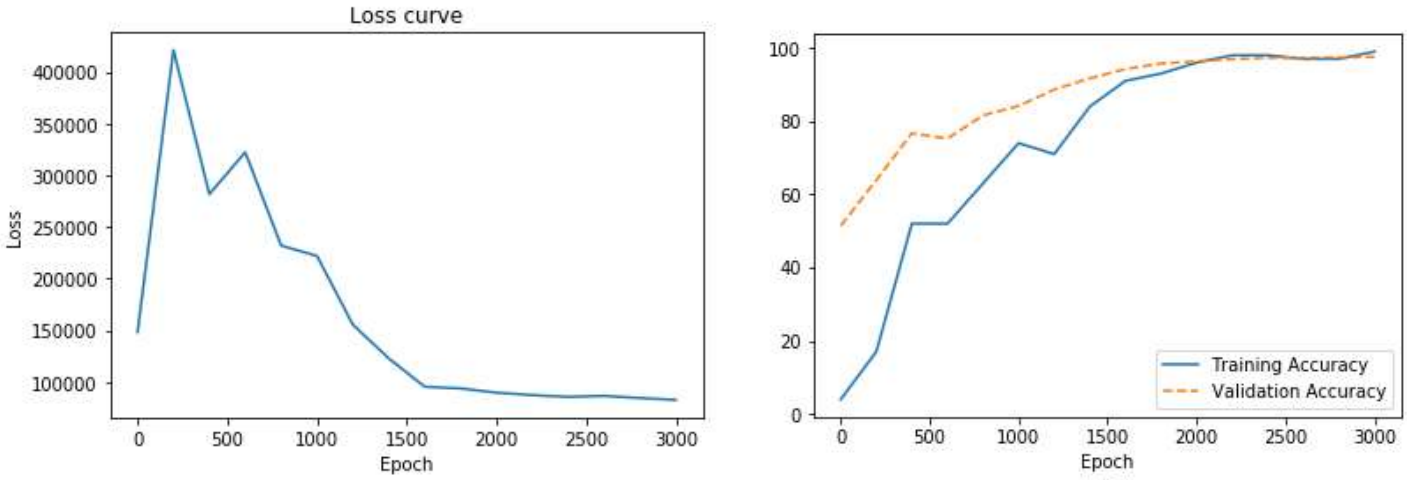


Fig 12. Two Hidden Layers with 2000 and 1000 nodes
Test accuracy = 98.1%