# Project 3: Building Networks with Tensorflow and Keras

COSC 525 Su-Ann Chong 03-26-2021

## 1. Introduction

In this project, multiple face attribute classifers with different network architectures for images are built. A version of the FairFace dataset was used to train the networks. The images are converted to grayscale and resized to 32 × 32 pixels. Each face has 3 different attributes that can be used for classification, namely race, gender and age. In this project, models are trained to classify gender and race of a face.

Five different architectures were experimented:
(1) Fully Connected Neural Network
(2) Small Convolutional Neural Network
(3) Convolutional Neural Network
(4) Convolutional Neural Network on Two Tasks Simultaneously
(5) Variational Auto Encoder

## 2. Introduction to Each Network Architectures

(1) Task 1: Fully Connected Neural Network

This architecture consists of all fully connected layers with the following specifications:

Table 1. Fully Connected Neural Network Architecture

| Layer | Type | Number of neurons | Activation Function |
|---|---|---|---|
| 1 | Flatten | - | - |
| 2 | Dense | 1024 | "tanh" |
| 3 | Dense | 512 | "sigmoid" |
| 4 | Dense | 100 | "relu" |
| 5 | Dense | n - number of classes | "softmax" |

(2) Task 2: Small Convolutional Neural Network

This architecture consists of layers with the following specifications:

Table 2. Small Convolutional Neural Network Architecture

| Layer | Type | Number of neurons | Stride | Padding | Activation Function | Pool size |
|---|---|---|---|---|---|---|
| 1 | Conv2D | 5 × 5 × 40 | 1 | "valid" | "relu" | - |
| 2 | MaxPooling2D | - | - | - | - | 2 |
| 3 | Flatten | - | - | - | - | - |
| 4 | Dense | 100 | - | - | "relu" | - |
| 5 | Dense | n - number of classes | - | - | "softmax" | - |

(3) Task 3: Convolutional Neural Network
This architecture consists of layers with the following specifications:

Table 3. Convolutional Neural Network Architecture

| Layer | Type | Number of neurons | Stride | Padding | Activation function | Pool size |
|---|---|---|---|---|---|---|
| 1 | Conv2D | $3 \times 3 \times 32$ | 1 | "same" | "relu" | - |
| 2 | Conv2D | $3 \times 3 \times 32$ | 1 | "same" | "relu" | - |
| 3 | MaxPooling2D | - | - | - | - | 2 |
| | Dropout 0.25 | - | - | - | - | - |
| 4 | Conv2D | $3 \times 3 \times 64$ | 1 | "same" | "relu" | - |
| 5 | Conv2D | $3 \times 3 \times 64$ | 1 | "same" | "relu" | - |
| 6 | MaxPooling2D | - | - | - | - | 2 |
| | Dropout 0.25 | - | - | - | - | - |
| 7 | Flatten | - | - | - | - | - |
| 8 | Dense | 512 | - | - | "relu" | - |
| | Dropout 0.5 | - | - | - | - | - |
| 9 | Dense | n - number of classes | - | - | "softmax" | - |

(4) Task 4: Multi-task Convolutional Neural Network

This architecture consists of layers with the following specifications:

Table 4. Multi-task Convolutional Neural Network Architecture

| Layer | Type | Number of neurons | Stride | Padding | Activation function | Pool size |
|---|---|---|---|---|---|---|
| 1 | Conv2D | $7 \times 7 \times 32$ | 1 | "same" | "relu" | - |
| 2 | MaxPooling2D | - | 2 | "same" | - | 3 |
| 3 | Lambda | - | - | - | - | - |
| 4 | Conv2D | $1 \times 1 \times 64$ | 1 | "same" | "relu" | - |
| 5 | Conv2D | $3 \times 3 \times 192$ | 1 | "same" | "relu" | - |
| 6 | MaxPooling2D | - | - | - | - | 2 |
| 7 | Lambda† | - | - | - | - | - |
| 8 | Flatten | - | - | - | - | - |
| 9 (1) | Dense | 100 | - | - | "relu" | - |
| 9 (2) | Dense | 100 | - | - | "relu" | - |
| 10 (1) | Dense | n - number of classes | - | - | "softmax" | - |
| 10 (2) | Dense | n - number of classes | - | - | "softmax" | - |

† This Lambda layer is the local response normalization function.

(5) Task 5: Variational Auto Encoder

This architecture consists of layers with the following specifications:

Table 5. Encoder Network Architecture

| Layer | Type | Number of neurons | Stride | Padding | Activation function | Pool size |
|---|---|---|---|---|---|---|
| 1 | Conv2D | $7 \times 7 \times 16$ | 1 | "same" | "relu" | - |
| 2 | Conv2D | $3 \times 3 \times 32$ | 1 | "same" | "relu" | - |
| 3 | Flatten | - | - | - | - | - |
| 4 | Dense | 32 | - | - | "relu" | - |
| 5 (1) | Dense | latent dimension | - | - | - | - |
| 5 (2) | Dense | latent dimension | - | - | - | - |
| 6 | Lambda | - | - | - | - | - |

† This Lambda layer is the local response normalization function.

Table 6. Decoder Network Architecture

| Layer | Type | Number of neurons | Stride | Padding | Activation function | Pool size |
|---|---|---|---|---|---|---|
| 1 | Dense | $32 \times 32 \times 32$ | - | - | "relu" | - |
| 2 | Reshape | - | - | - | - | - |
| 3 | Conv2DTranspose | $3 \times 3 \times 32$ | 1 | "same" | "relu" | - |
| 4 | Conv2DTranspose | $7 \times 7 \times 16$ | 1 | "same" | "relu" | - |
| 5 | Conv2DTranspose | $3 \times 3 \times 1$ | 1 | "same" | "relu" | - |

† This Lambda layer is the local response normalization function.

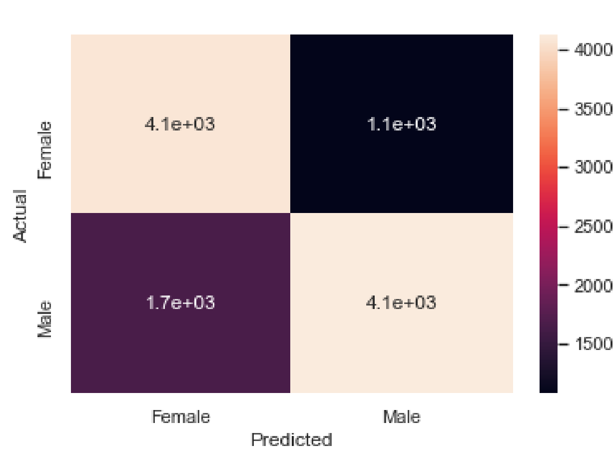
## 3. Results of Each Network Architecture

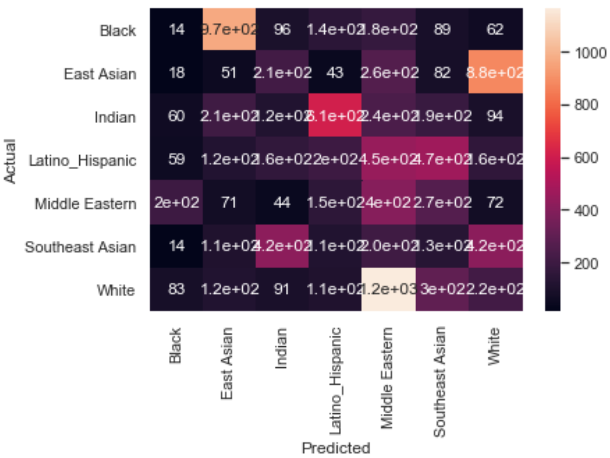
Here is a summary of parameters used for each network and the accuracy obtained:

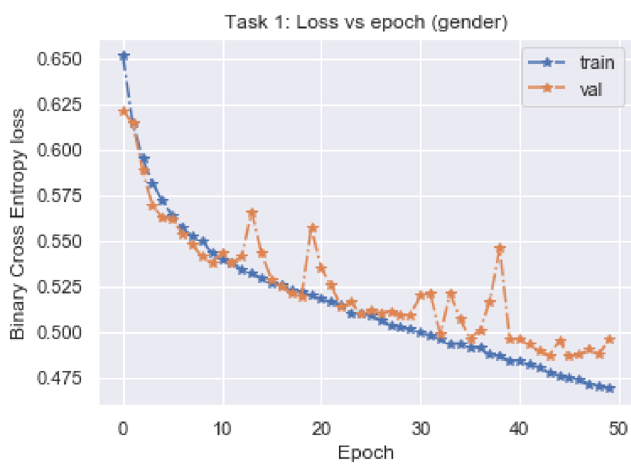
| Task | type | learning rate | momentum | batch size | epoch | loss function | accuracy(%) |
|---|---|---|---|---|---|---|---|
| 1 | gender | 0.001 | 0.9 | 32 | 50 | categorical cross entropy | 75.04 |
| 2 | gender | 0.001 | 0.9 | 32 | 50 | categorical cross entropy | 79.66 |
| 3 | gender | 0.001 | 0.9 | 32 | 25 | categorical cross entropy | 81.96 |
| 4 | gender | 0.001 | 0.9 | 32 | 30 | categorical cross entropy | 81.81 |

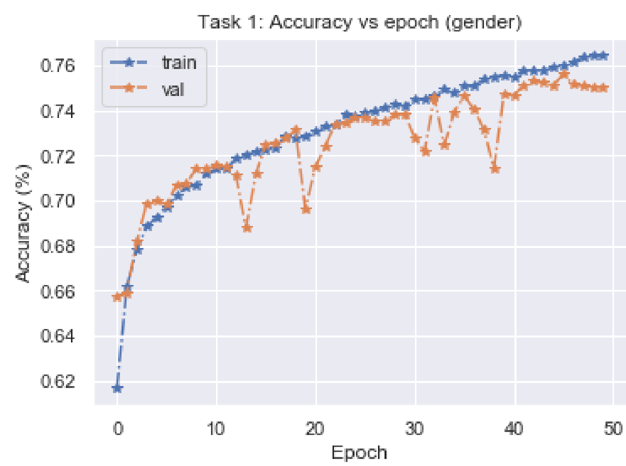
| Task | type | learning rate | momentum | batch size | epoch | loss function | accuracy(%) |
|---|---|---|---|---|---|---|---|
| 1 | race | 0.001 | 0.9 | 32 | 50 | categorical cross entropy | 43.10 |
| 2 | race | 0.001 | 0.9 | 32 | 50 | categorical cross entropy | 46.61 |
| 3 | race | 0.001 | 0.9 | 32 | 25 | categorical cross entropy | 52.39 |
| 4 | race | 0.001 | 0.9 | 32 | 30 | categorical cross entropy | 51.24 |
| 5 | - | 0.001 | 0.9 | 256 | 50 | mean squared error | - |

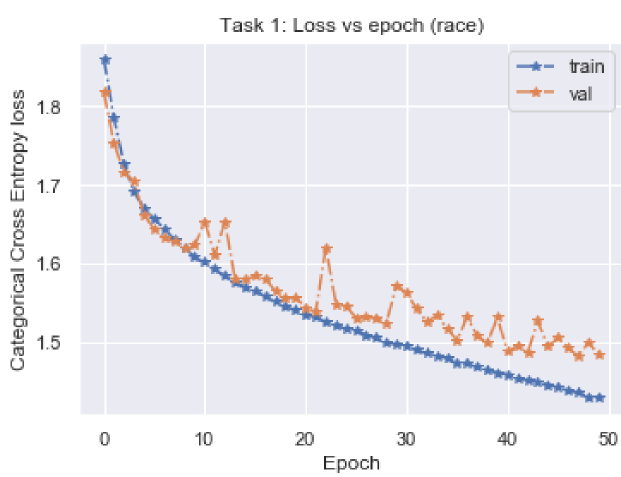
(1) Fully Connected Neural Network

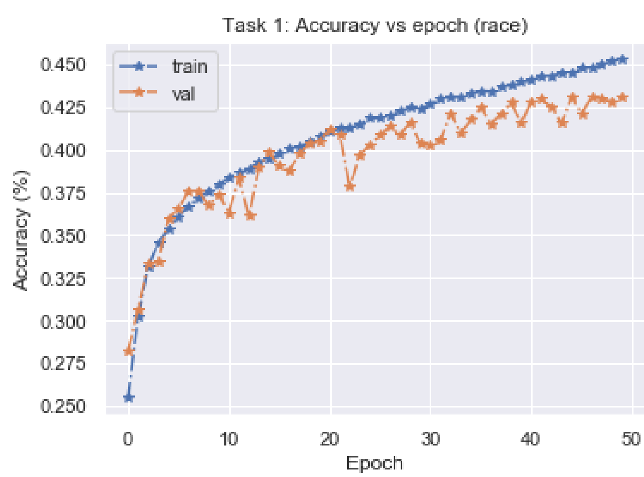


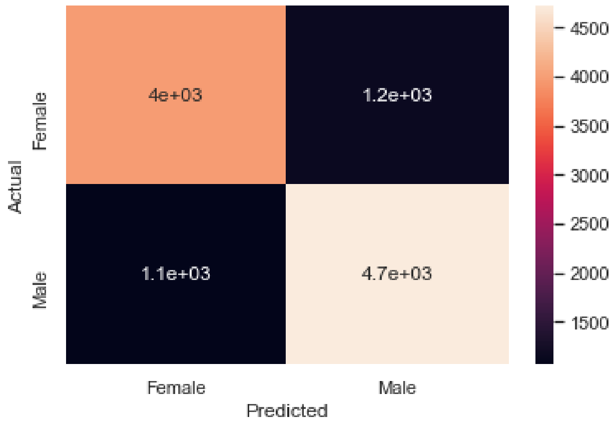



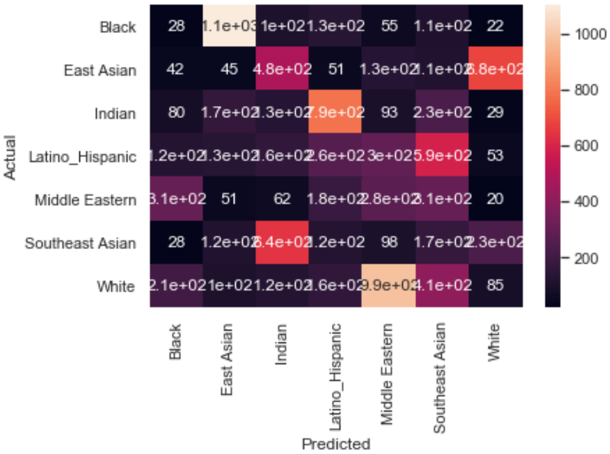



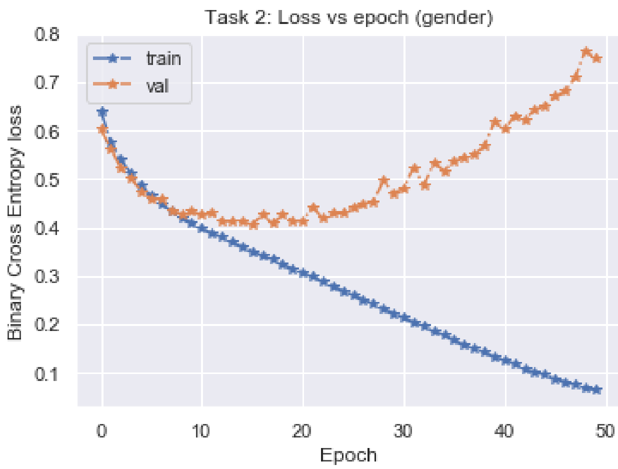



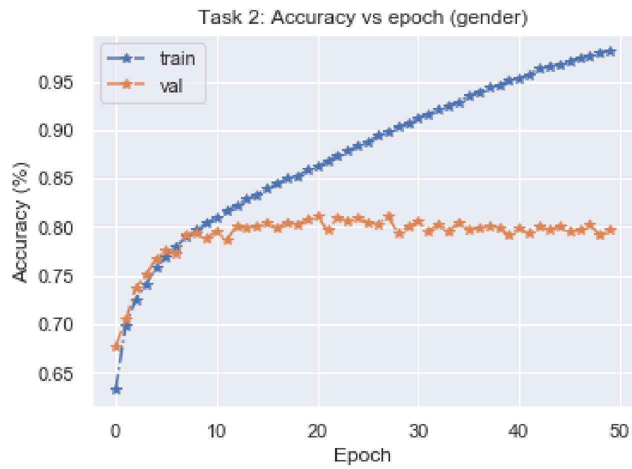



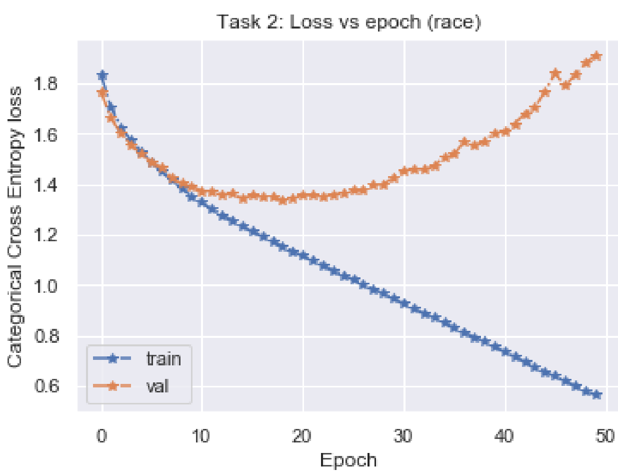
(2) Small Convolutional Neural Network

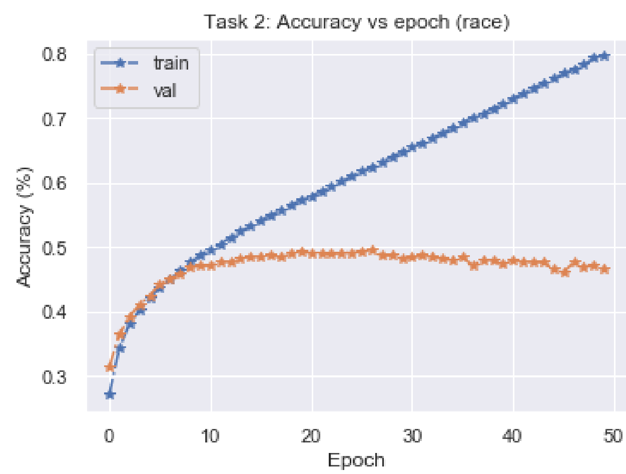


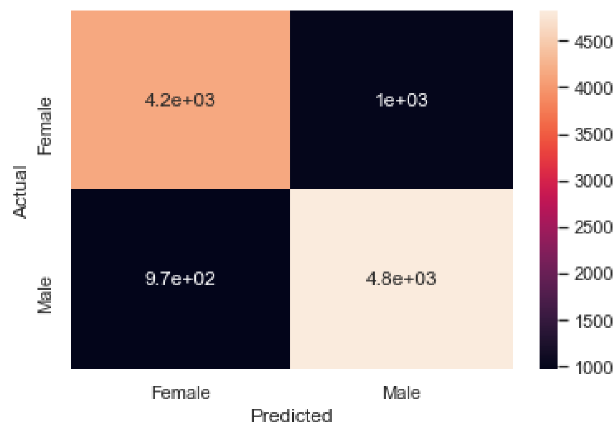



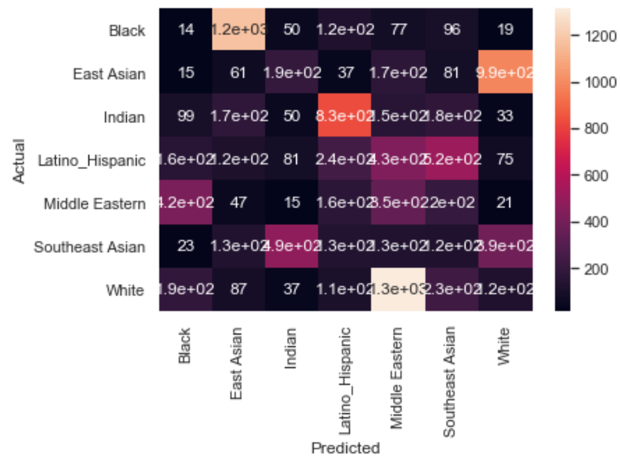



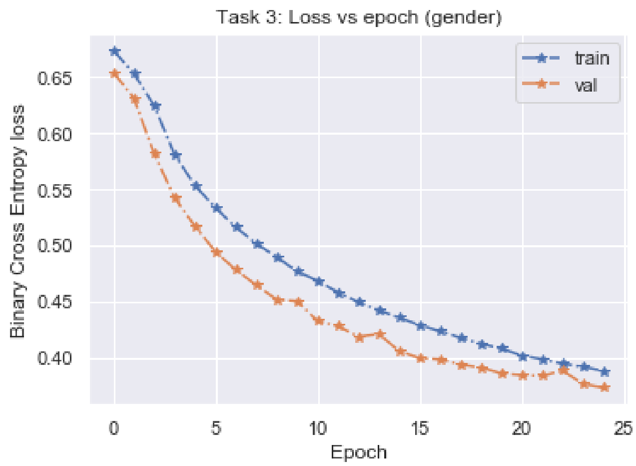



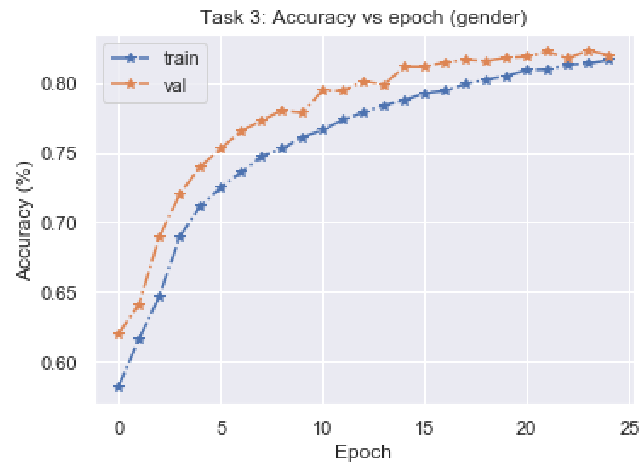



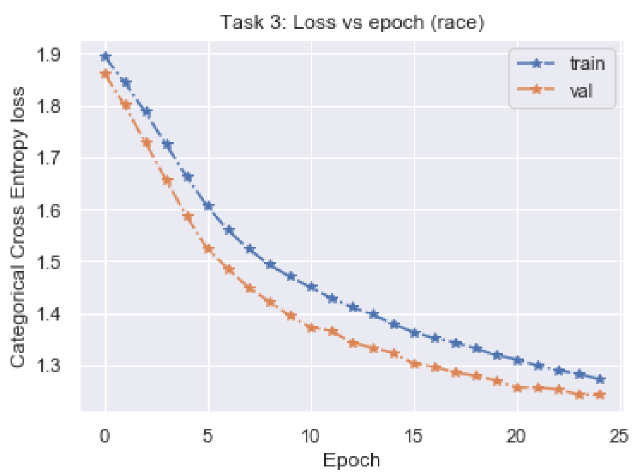
(3) Convolutional Neural Network

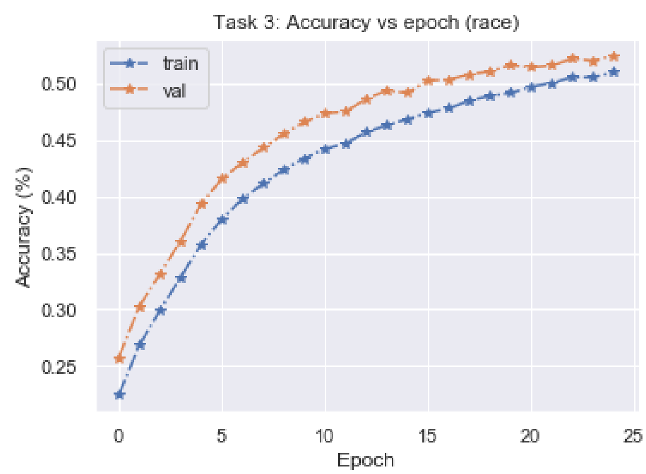


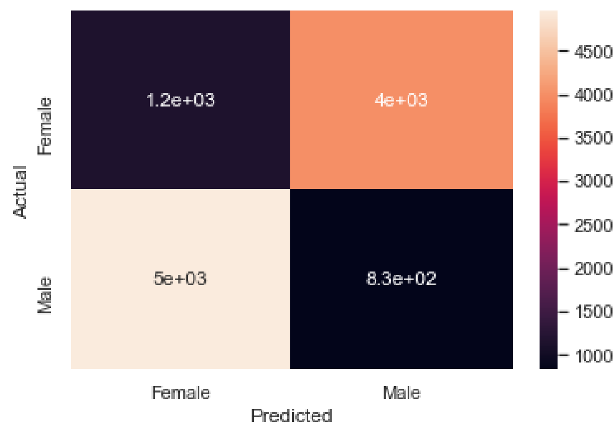



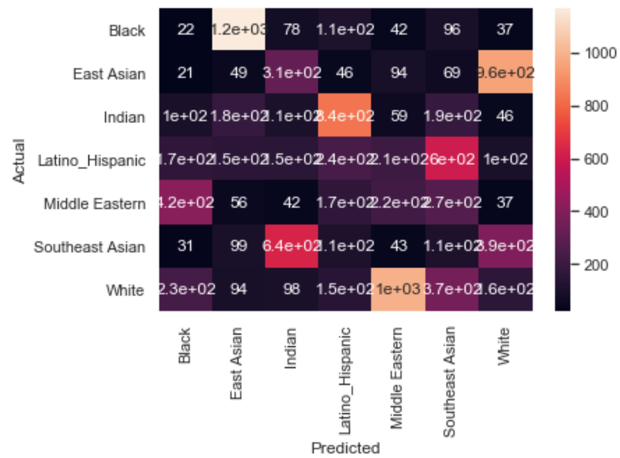



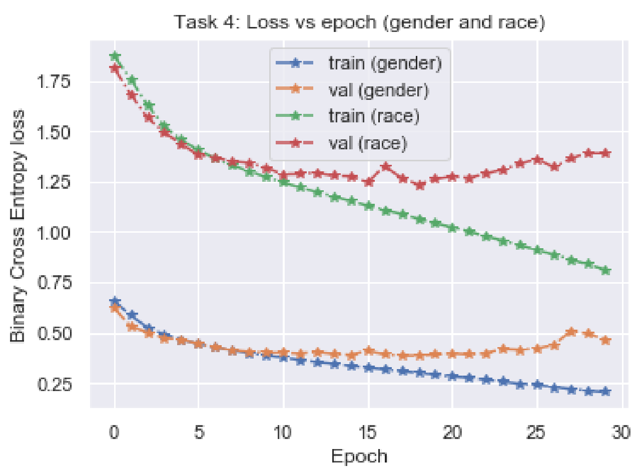



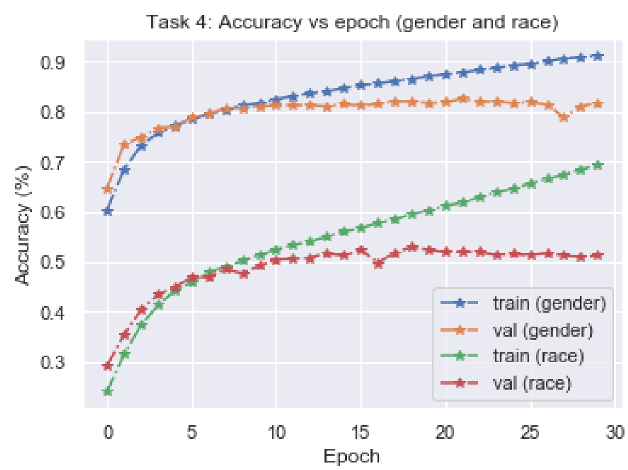



(4) Convolutional Neural Network on Two Tasks Simultaneously

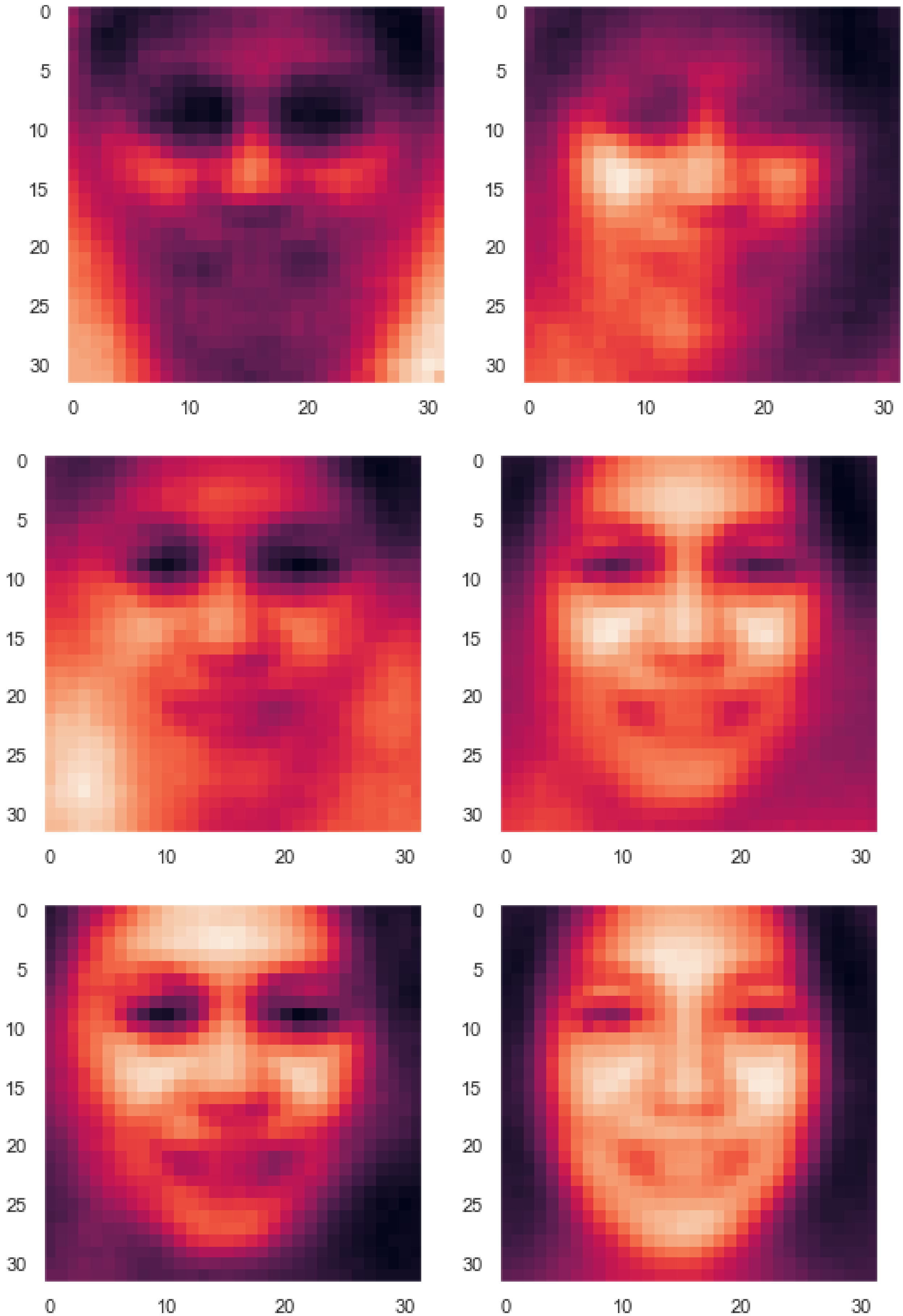


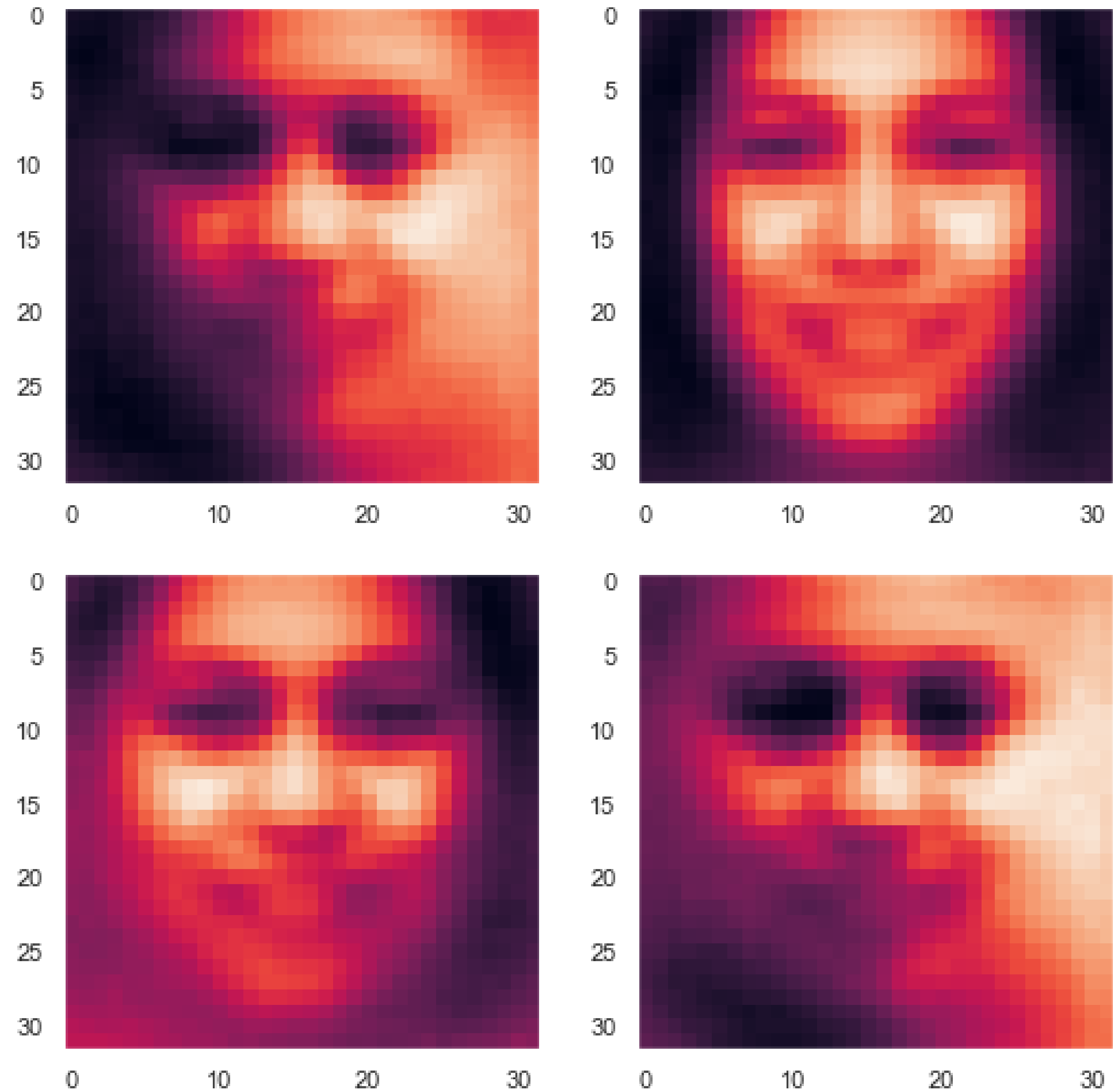



(5) Variational Auto Encoder






## 4. Conclusion

- Convolutional neural networks (Task 2) performs better in classifying image dataset than fully connected networks (Task 1) in both tasks.
- Deeper convolutional neural networks (Task 3) give better accuracy than shallower convolutional neural network (Task 2) and are less proned to overfitting.
- Multi-task convolutional neural networks (Task 4) have comparable performance when compared to single-task convolutional neural networks. Overfitting of the training data is observed.
- The variational autoencoder model (Task 5) is capable of picking up important features of the faces but the produced images are a lot blurrier than the original images. Training for longer may produce better result if more computing resources is available.

## 5. How to run code

You can run the project code with examples using the following command:

```
$ python proj3.py task[1-5]
```