CSC321 Project 1: Face Recognition and Gender Classification with K-Nearest Neighbours

Due on Wednesday, February 3, 2016

Wei Zhen Teoh
1000960597

February 3, 2016

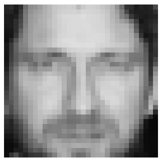
Part 1

Dataset description

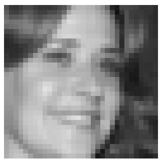
The initial dataset consists of 720 images of the actors and actresses obtained through the dataset on `http://vintage.winklerbros.net/facescrub.html`, among which 120 are collected for each of the following 6 persons: Gerald Butler, Daniel Radcliffe, Michael Vartan, Lorraine Bracco, Peri Gilpin and Angie Harmon. This set of images are to be used for face recognition and gender classification (Part 3, 4 and 5). Another 120 images (60 male and 60 female) from other actors and actresses are later on added to this dataset for gender classification in Part 6. The faces are cropped out from these images based on the bounding boxes dimensions given, grayscaled and resized to 32×32 pixels, for use in the implementation of K-nearest neighbour (KNN) algorithm. A few examples of the images we obtained and their cropped out faces are shown on the next page.

Examining the dataset we obtained, it appears that most of the bounding boxes accurately cropped out the full faces from these images. However for a few reasons the cropped out faces do not align very well with one another in general. One of those is that the head postures of the actors and actresses on the images are slightly different from one to another. Some faces appear to be in full frontal view but some others are slightly tilted as shown in Figure 1d. Besides, some of the bounding boxes are not tight (Figure 1f). The same actor or actress also has different facial expressions on different images(Figure 1h), which may deter the implementation of KNN algorithm. Some also wear spectacles on some of the images, which alter their appearances(Figure 1j). Anyway the above circumstances to KNN performance can be overcome by collecting more images into the dataset to expand the training set.

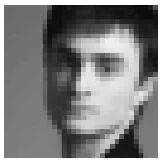
(a) Original image 1


(b) Extracted face 1

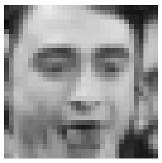
(c) Original image 2

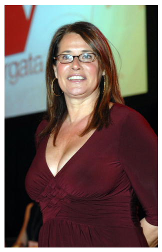

(d) Extracted face 2

(e) Original image 3


(f) Extracted face 3

(g) Original image 4


(h) Extracted face 4


(i) Original image 5

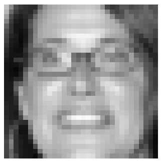

(j) Extracted face 5

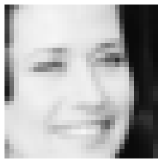
Figure 1: Sample images from the dataset and extracted faces

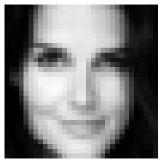
Part 2

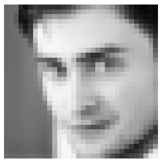
Setting up training set, validation set and test set
A class `face` (line 162 in `faces.py`) is first created to carry information associated with each face including name and gender. These information are recorded for each face for labelling purpose (training) and KNN guess verification (validation and test). The class also carries as an attribute the vector obtained by flattening the image array of the extracted face, which will be used for KNN algorithm implementation.
An instance is created for each of the 720 faces extracted from the images of the 6 actors and actresses in the initial dataset. These face instances are split into 3 non-overlapping sets: the test set (60 faces), training set(600 faces) and validation set(60 faces). The separation is done as follow:
test set: faces of each actor/actress with index 1 to 10
training set: faces of each actor/actress with index 11 to 110
validation set: faces of each actor/actress with index 111 to 120

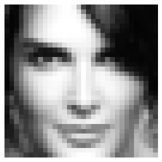
Part 3

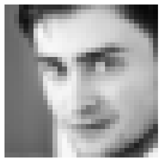
KNN algorithm implementation for face recognition
For each of the three sets established in the last part in turn, we apply KNN algorithm to guess the actor/actress to which each face in the set belongs to. For each face, The L2-distance between the vector of the face (obtained previously) and those of the others were computed and the k nearest neighbour faces in the training set for the face are selected. This is done using the function `kNN` (line 278 in `faces.py`). The function `namevote` (line 289 in `faces.py`) then carries out a majority voting among those k nearest neighbours, where the name with the highest count among them would be chosen. The name becomes our KNN guess for the face in question. The procedure is performed for each odd K from 1 to 39 on each of those 3 sets.The performance (success rate) of the KNN guesses at every K is measured for all 3 sets.
For the dataset we obtained when we ran the code, the performance on the validation set was optimized at K= 11. We used this K to perform KNN guesses on the test set and obtained a success rate of 51.7%. 5 of the failed cases (KNN guesses with K=11) along with their 5 nearest neighbours from the training set in order are displayed in Figures 2 to 6.


(a) Face 1: Lorraine Bracco


(b) Face 1 nearest neighbour 1


(c) Face 1 nearest neighbour 2


(d) Face 1 nearest neighbour 3


(e) Face 1 nearest neighbour 4

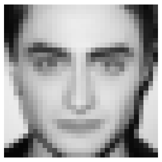
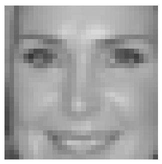
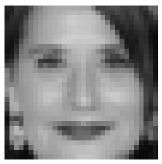
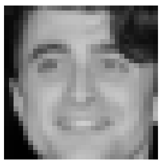
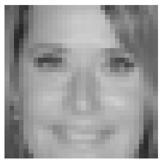

(f) Face 1 nearest neighbour 5

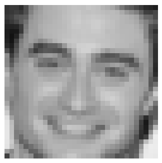
Figure 2: Failed Case 1 (guess = Angie Harmon)


(a) Face 2: Peri Gilpin


(b) Face 2 nearest neighbour 1


(c) Face 2 nearest neighbour 2


(d) Face 2 nearest neighbour 3


(e) Face 2 nearest neighbour 4

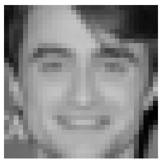
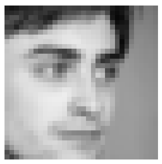
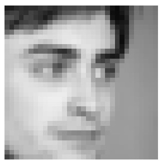
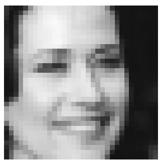
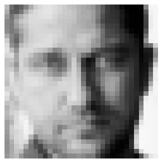

(f) Face 2 nearest neighbour 5

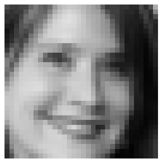
Figure 3: Failed Case 2 (guess = Daniel Radcliffe)


(a) Face 3: Daniel Radcliffe


(b) Face 3 nearest neighbour 1


(c) Face 3 nearest neighbour 2


(d) Face 3 nearest neighbour 3


(e) Face 3 nearest neighbour 4

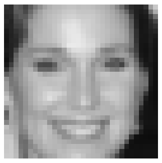
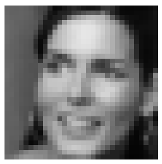
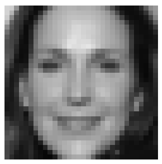
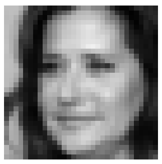
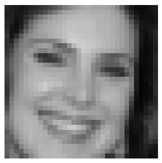

(f) Face 3 nearest neighbour 5

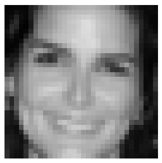
Figure 4: Failed Case 3 (guess = Angie Harmon)


(a) Face 4: Angie Harmon


(b) Face 4 nearest neighbour 1


(c) Face 4 nearest neighbour 2


(d) Face 4 nearest neighbour 3


(e) Face 4 nearest neighbour 4

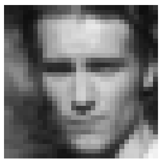
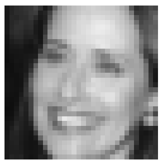
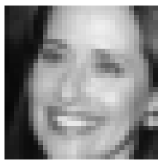
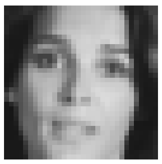
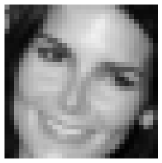

(f) Face 4 nearest neighbour 5

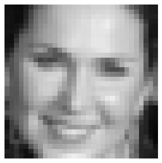
Figure 5: Failed Case 4 (guess = Peri Gilpin)


(a) Face 5: Lorraine Bracco


(b) Face 5 nearest neighbour 1


(c) Face 5 nearest neighbour 2


(d) Face 5 nearest neighbour 3


(e) Face 5 nearest neighbour 4

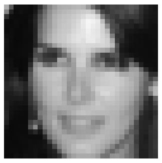

(f) Face 5 nearest neighbour 5

Figure 6: Failed Case 5 (guess = Angie Harmon)

Part 4

KNN face recognition performance on training set, validation set and test set

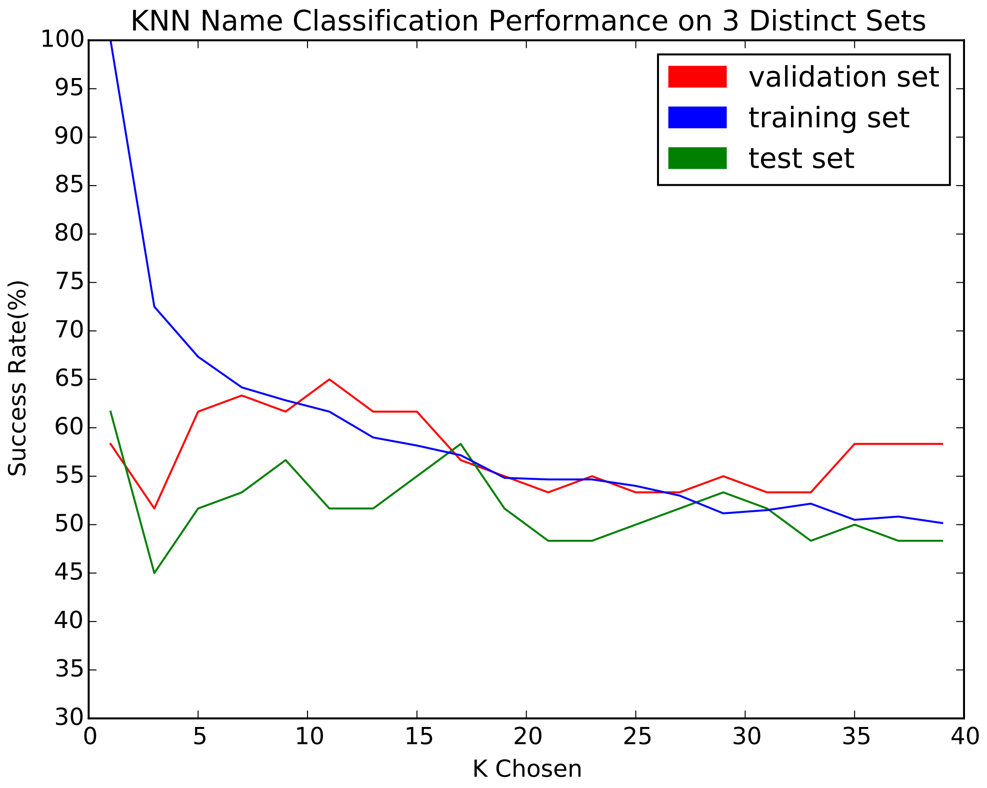


Figure 7: Face recognition performance on 3 different sets
(Note: Downloading or network issues may vary the performances obtained)

For the dataset we used, the performance of the KNN face recognition algorithm exhibited similar pattern on both validation and test sets for the range of K's we chose. KNN performed better on both sets for K = 1 and for K between 7 and 19. The performance peaked at K=11 for the validation set (65.0% success rate) and K=17 for the test set (58.3% success rate). Since our faces in each set are decided arbitrarily, the validation and test sets, each carrying same number of faces are expected to yield similar results.
On the other hand the training set had its peak when K=1, with a success rate of 100%. The performance decayed rapidly when K increases, and slowly leveled off. This can be explained by the nature of the training set. When a face in the training set looks for its neighbours, its closest neighbour will be exactly the same face. As a result, when K is set to 1, KNN generates the correct guess every time. When K increases, the correct matching face gradually loses some of its voting power in the name vote among K neighbours, thus the performance decays.

Part 5

Implementation of KNN algorithm for gender classification
The KNN algorithm is implemented in exactly the same way as in part 3. The only difference is that we use the function `gendervote` (line 416 in `faces.py`) instead of `namevote` this time to generate the guess for the gender. The two votings work exactly the same way except that one is for gender and the other is for name. the performance on the validation set is measured at various K. The best K is then chosen for the implementation on the test set and the performance is measured again.
The performance we previously obtained on the validation set using our dataset is recorded below.

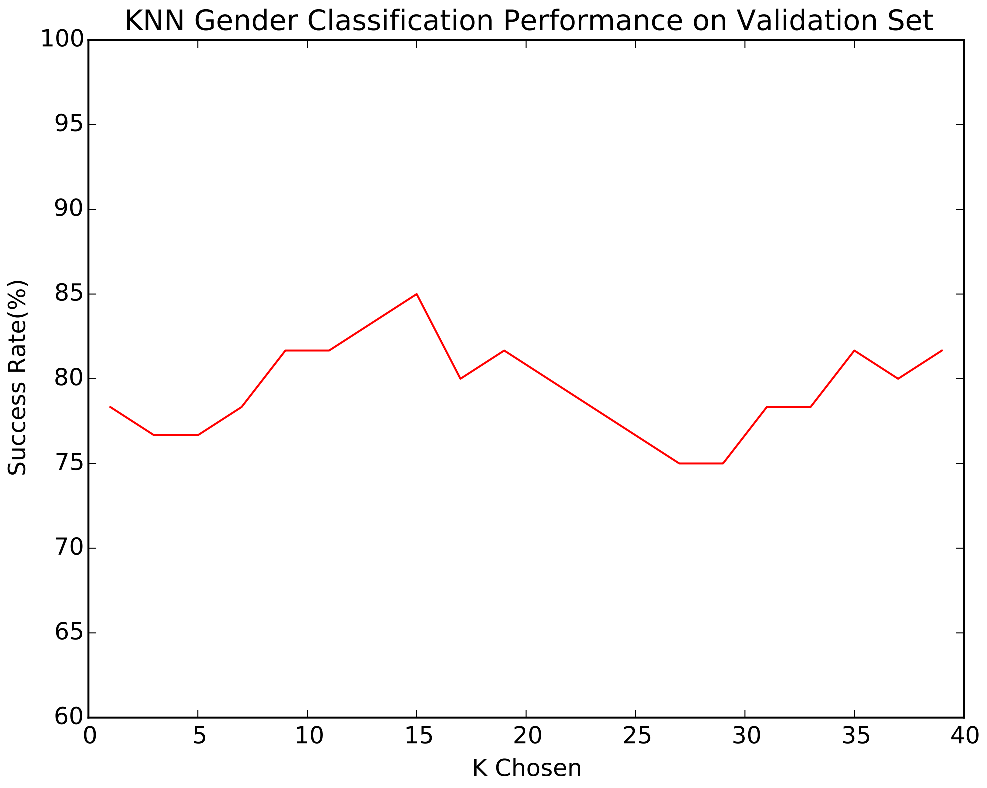


Figure 8: Gender classification performance on validation set
(Note: Downloading or network issues may vary the performances obtained)

For the dataset we used, the KNN gender classification performance on the validation set peaked at K=15, with a success rate of 85.0%. This K was then chosen for implementing the KNN gender classification on the test set, which achieved a success rate of 86.7%.

Part 6

KNN gender classification on new faces
A new set of images (120 of them) , chosen randomly, are downloaded and processed as stated in part 1. The processed dataset consists of 60 male faces and 60 female faces. A `face` instance is created for each of them with their gender recorded. Again, we split the dataset into 2 non-overlapping sets: validation set (first 30 male and female faces) and test set (all the rest).

KNN gender classification is performed on the new validation set. The performance is measured at each odd K for 1 to 39. The best K is chosen for the implementation of KNN gender classification on the test set and the performance is measured again. Our previous results are recorded below.

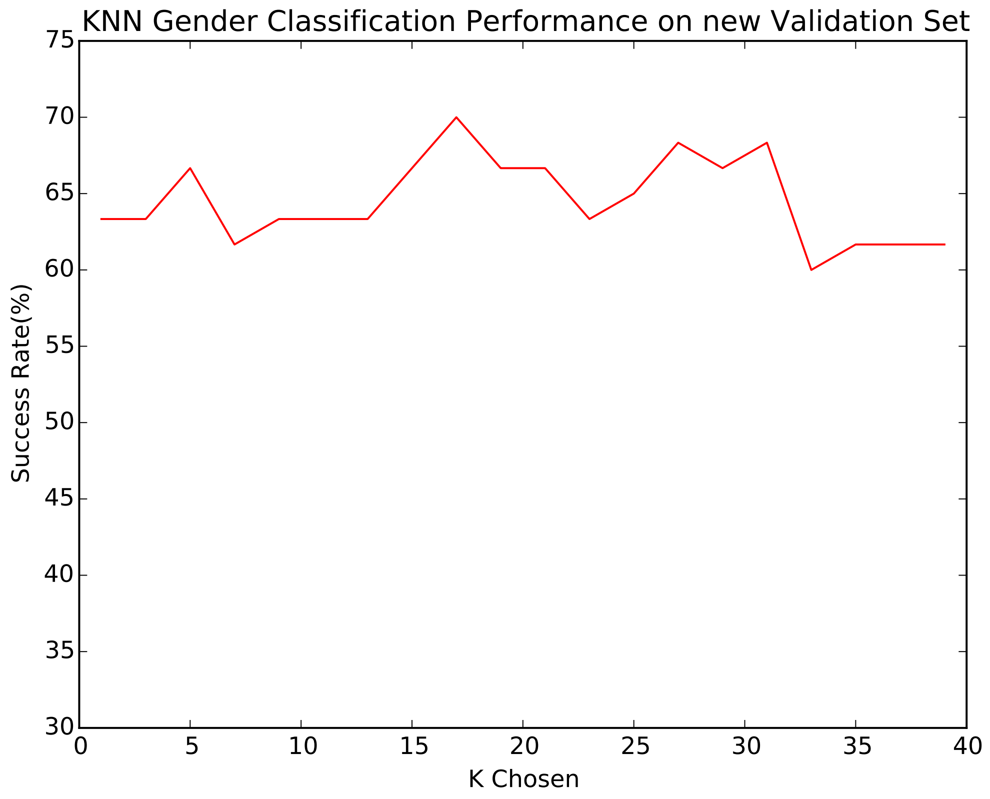


Figure 9: Gender classification performance on new validation set
(Note: Downloading or network issues may vary the performances obtained)

For our dataset, the KNN gender classification performance on the new validation set peaked at K=17, with a success rate of 70.0%. This K was chosen for the implementation of KNN gender classification on the test set, which achieved a success rate of 70.0% as well.
In general, the performance of KNN gender classification on the new sets are poorer than on the old sets. One obvious reason is that the training set contains faces of the same actors/actresses as in the old sets, which are more likely to be chosen as their nearest neighbours and vote for the correct gender.