Cemhan Kaan Özaltan
21902695

Homework 2 Report

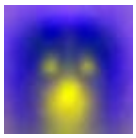
Question 1.1

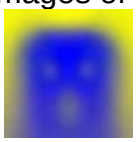
```
10 best PVEs
Red channel's PVE #1: 0.2150928295705464
Green channel's PVE #1: 0.2004593939345657
Blue channel's PVE #1: 0.2299724795720598

Red channel's PVE #2: 0.13543590648775305
Green channel's PVE #2: 0.13767809111140247
Blue channel's PVE #2: 0.13678519667322256

Red channel's PVE #3: 0.07504975573273252
Green channel's PVE #3: 0.07695367428026625
Blue channel's PVE #3: 0.07034022694974615

Red channel's PVE #4: 0.051732173422999836
Green channel's PVE #4: 0.05397123455088882
Blue channel's PVE #4: 0.05356417895492482

Red channel's PVE #5: 0.04228943097350487
Green channel's PVE #5: 0.042918749347138276
Blue channel's PVE #5: 0.039821729668577326

Red channel's PVE #6: 0.024583277160715355
Green channel's PVE #6: 0.026022591594315067
Blue channel's PVE #6: 0.02373237907459678

Red channel's PVE #7: 0.021772692745900574
Green channel's PVE #7: 0.02142652916670169
Blue channel's PVE #7: 0.02099218512082034

Red channel's PVE #8: 0.019898578692023182
Green channel's PVE #8: 0.02081297174873689
Blue channel's PVE #8: 0.02075866937751995

Red channel's PVE #9: 0.017070945760386032
Green channel's PVE #9: 0.017393519667203525
Blue channel's PVE #9: 0.01668136288830131

Red channel's PVE #10: 0.01655960329228331
Green channel's PVE #10: 0.016811649418805547
Blue channel's PVE #10: 0.016293072799311355

PVE sums
Red: 0.6194851938388453
Green: 0.6144484048200242
Blue: 0.6289414810790804

Number of PVEs needed for 0.7 for the red channel: 18
Number of PVEs needed for 0.7 for the green channel: 19
Number of PVEs needed for 0.7 for the blue channel: 17
```

Here, we see that higher index PCs have lower values compared to lower index PCs. This is also evident due to the fact that PVE #1 has the highest value for all color channels.

Question 1.2

The combined RGB images of the 10 best PCs are as follows:


PC #1


PC #2


PC #3


PC #4


PC #5


PC #6


PC #7


PC #8


PC #9

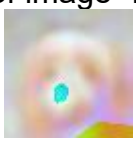

PC #10

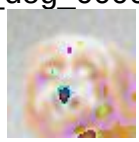
We see that all of these PC images somewhat look like dogs while also being significantly distinct considering that they are PCs of different dimensions. These PCs contain the most important features for model training while eliminating the redundant ones. Therefore, even though the images are not very understandable to the human eye, they serve a dimensionality reduction purpose for less computational cost.

Question 1.3

Reconstruction of image “flickr_dog_000002.jpg” with increasing k values is as follows:


k = 1


k = 50


k = 250


k = 500


k = 1000

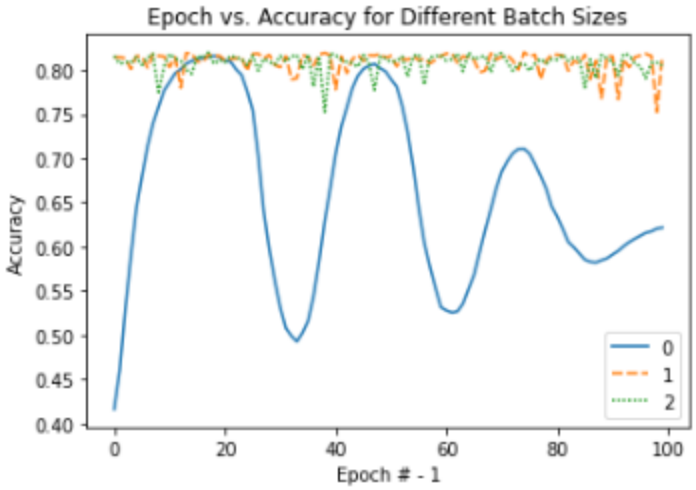

k = 4096

Here, we first calculate the dot products of the first k PCs and the image for each color channel. We then project the obtained data back onto the image space using the first k eigenvectors, creating the resulting image which will be saved by the function. We see that as we use more PCs, the image becomes clearer as opposed to its initial state where a lot of blur and noise exists.

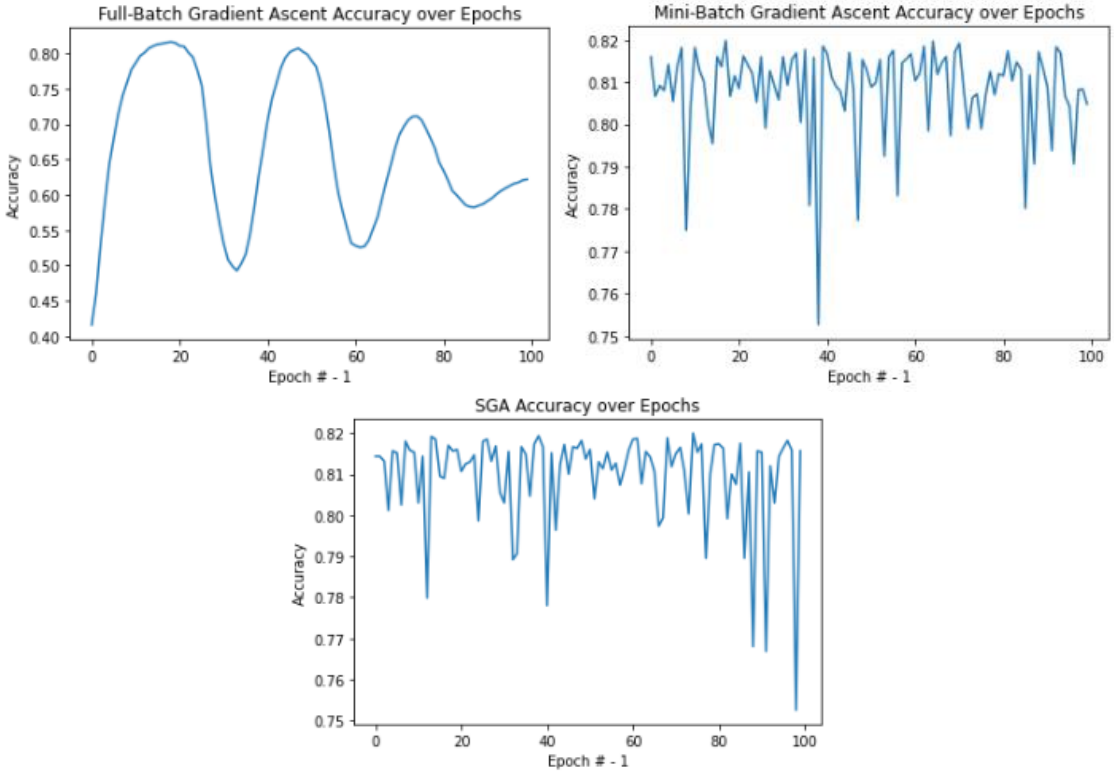
Question 2.1

The accuracies and the plot is as follows for different batch sizes (In the legend, 0 is FBGA, 1 is SGA, 2 is MBGA. The x-axis is epoch number minus 1 which starts from 0.):

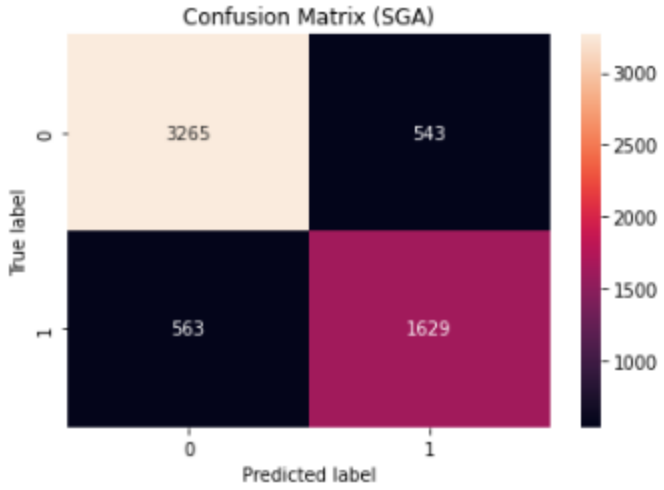
```
Accuracy for full-batch gradient ascent: 0.6215
Accuracy for stochastic gradient ascent: 0.8156666666666667
Accuracy for mini-batch gradient ascent: 0.8048333333333333
```



Separate plots are as follows for clarity:



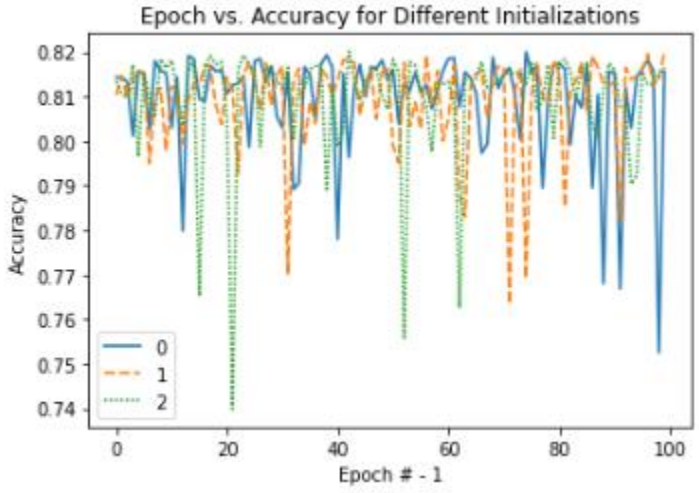
Here, we see that with an equal learning rate of 0.1, the batch size that performs the best is obtained with SGD, followed by mini-batch and full-batch. This is expected since the batch sizes are 42000, 64, and 1 respectively for full-batch, mini-batch and SGA, making SGA the most computationally expensive due to its large amount of weight update operations. I also observed that SGD took slightly longer to compute due to these frequent updates, showing that batch size affects training time. Finally, we see that the fluctuations are much more frequent and sharper as the batch size decreases, again due to the frequent updates. The confusion matrix is as follows for SGA:



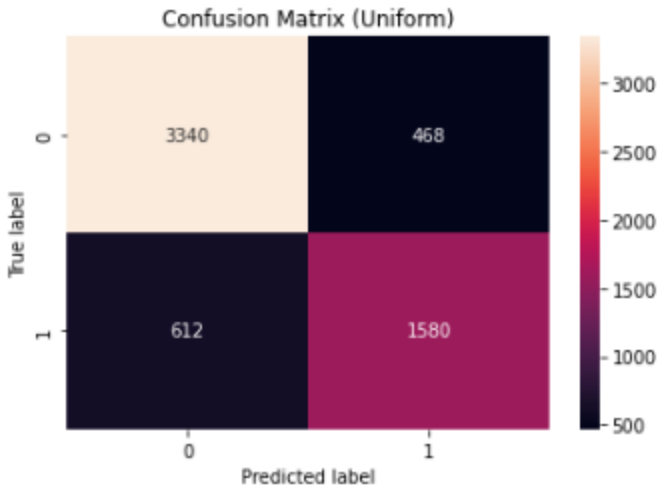
Question 2.2

I additionally initialized my weights using uniform distribution and zeros, leaving other hyperparameters the same. Different initializations may yield different final accuracies since all models are trained over 100 epochs and a bad initialization may make it hard for the weights to get updated close to their correct values in the given number of iterations. However, it is not really possible to know which distribution is the best initialization before knowing the best weights for the model. The following outputs and plot show my results:

```
Accuracy with gaussian distibution: 0.8156666666666667
Accuracy with uniform distibution: 0.82
Accuracy with zero distibution: 0.8165
```



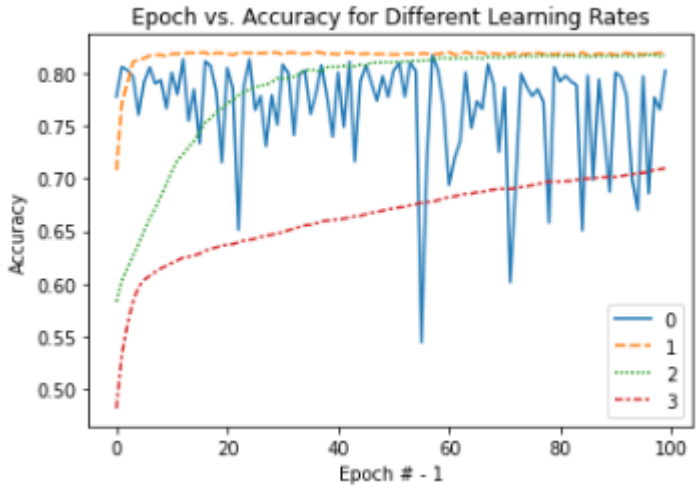
In the plot, 0 = Gaussian, 1 = Uniform, 2 = Zero. The confusion matrix for the model initialized with uniform distribution (best results obtained with this distribution) is as follows:



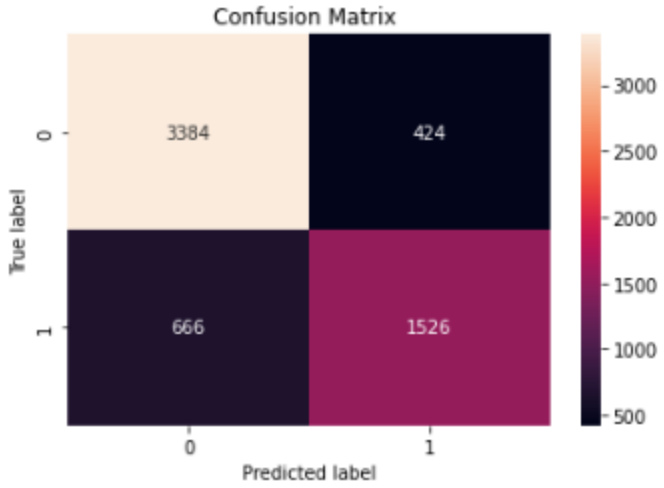
Question 2.3

In gradient ascent, the learning rate determines how much effect each update will have on the weights of the model, i.e. how fast it is trained. If it is too large, we may not reach the maxima and oscillate, if it is too small, our model may not reach the maxima due to being slow. Here, I tried the learning rates {1, 10^3, 10^4, 10^5} and obtained the best accuracy with the learning rate 10^-3 as the plot shows (Labels: 0 = 1, 1 = 10^-3, 2 = 10^-4, 3 = 10^-5) along with the accuracy outputs:

```
Accuracy for LR with label 0: 0.8021666666666667
Accuracy for LR with label 1: 0.8183333333333334
Accuracy for LR with label 2: 0.817
Accuracy for LR with label 3: 0.7095
```

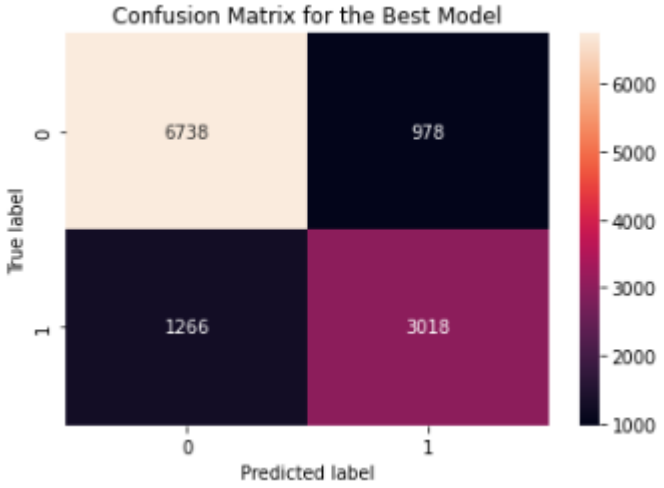


The confusion matrix of this learning rate is as follows:



Question 2.4

Confusion matrix with the best parameters (batch size = SGD, weight initialization = uniform, learning rate = 10^-3):



In our case, recall is more important since it is the true positive rate and we want to catch the positives more than we want to eliminate negatives. Therefore, we should consider positives and negatives differently. The metrics are as follows:

```
Accuracy: 0.813
Precision 0.8418290854572713
Recall 0.8732503888024883
F1 0.8572519083969464
F2 0.8667798703303489
F0.5 0.8479311386288131
```