# Name: Mian Usman Naeem Kakakhel
# Id: 21701015
# Section: 02
# HW 2
# 02/05/2020

**Q1.1) Logistic Regression Mini-Batch and Stochastic for HOG and inception**

HOG mini batch (size = 25):

Learning rate: 0.0001
Accuracy: 68.0
Precision: 0.6978021978021978
Recall: 0.635
NPV: 0.6651376146788991
FPR: 0.3021978021978022
FDR: 0.25229357798165136
F1: 0.6649214659685864
F2: 0.6466395112016293
----Confusion Matrix----
TP: 127 FP: 55
FN: 73 TN: 145
Time: 4.0706939697265625 s

HOG Stochastic:

Learning rate: 0.0001
Accuracy: 67.5
Precision: 0.6923076923076923
Recall: 0.63
NPV: 0.6605504587155964
FPR: 0.3076923076923077
FDR: 0.25688073394495414
F1: 0.6596858638743456
F2: 0.6415478615071284
----Confusion Matrix----
TP: 126 FP: 56
FN: 74 TN: 144
Time: 20.07291531562805 s

Inception mini batch (size = 25):

Learning rate: 0.0001
Accuracy: 87.25
Precision: 0.8743718592964824
Recall: 0.87
NPV: 0.8706467661691543
FPR: 0.12562814070351758
FDR: 0.12437810945273632
F1: 0.8721804511278195
F2: 0.8708708708708709
----Confusion Matrix----
TP: 174 FP: 25
FN: 26 TN: 175
Time: 37.200258016586304 s

Inception stochastic:

Learning rate: 0.0001
Accuracy: 87.0
Precision: 0.87
Recall: 0.87
NPV: 0.87
FPR: 0.13
FDR: 0.13
F1: 0.87
F2: 0.87
----Confusion Matrix----
TP: 174 FP: 26
FN: 26 TN: 174
Time: 58.4375205039978 s

Mini Batch vs Stochastic:

The time taken in both HOG and inception for Mini Batch is less than stochastic because in mini batch, the number of times gradient ascent that has to be applied is way greater than for stochastic. Also, since in the case of logistic regression there is no local minima/maxima, there is one global maxima so the model converges to the global minima/maxima in either case. Therefore, the accuracy in both Mini Batch and Stochastic is almost the same. Thus we can conclude that here, Mini Batch is the better gradient ascent technique to use.

HOG vs Inception:

In HOG we have considerably less number of features than Inception thus the time taken in Inception is way greater than HOG. Even though the time taken for Inception is greater, the accuracy given by Inception (87.25) is greater than HOG (67.5) thus Inception proves to be a

better extraction method for use by logistic regression. As far as the accuracy and recall are concerned, they are 0.87, 0.87 and 0.69, 0.63 for Inception and HOG respectively. Since the dataset is not skewed towards any specific class, these are approximately the same as our accuracy. The NPV dictates the probability of a negative prediction being actually negative. FPR dictates the probability of a positive prediction actually being negative. FDR dictates the probability of both predictions actually not being what they actually are.

**Q1.2) Logistic Regression Full Batch for HOG and inception**

Inception full batch:

Learning rate: 0.0001
Accuracy: 87.75
Precision: 0.8793969849246231
Recall: 0.875
NPV: 0.8756218905472637
FPR: 0.12060301507537688
FDR: 0.11940298507462686
F1: 0.8771929824561404
F2: 0.8758758758758759
----Confusion Matrix----
TP: 175 FP: 24
FN: 25 TN: 176
Time 6.462533473968506 s
Indices of 10 most important features: [1057 300 695 288 1109 169 75 1784 12 641]

HOG full batch:

Learning rate: 0.0001
Accuracy: 67.5
Precision: 0.6902173913043478
Recall: 0.635
NPV: 0.6620370370370371
FPR: 0.30978260869565216
FDR: 0.2638888888888889
F1: 0.6614583333333333
F2: 0.6453252032520325
----Confusion Matrix----
TP: 127 FP: 57
FN: 73 TN: 143
Time 0.47311997413635254 s
Indices of 10 most important features: [154 289 61 30 291 155 290 196 165 156]

As in Mini Batch Gradient Ascent, the Accuracy of Full Batch is also 85.70 and 68.0 for Inception and HOG respectively. Again the time taken by HOG is less than Inception due to the number of features in Inception being greater. The recall and precision is the same as the accuracy as the classes are not imbalanced. The most important weights highlight the fact that the features at these 10 locations are the most important for the classification. The greater the value of the weight, the greater importance that feature has. We can again conclude that Inception features are better than HOG for logistic regression.

NOTE: All Validation is being performed by using Accuracy as there is no class imbalance.

**Q1.4) Soft Margin SVM with linear kernel for HOG and Inception**

HOG:

C: 0.01 Accuracy: 66.5
C: 0.1 Accuracy: 67.9
C: 1 Accuracy: 67.15
C: 10 Accuracy: 66.3
C: 100 Accuracy: 64.85
fin_C: 0.1
----Confusion Matrix----
[[126. 54.]
[ 74. 146.]]
Accuracy: 68.0
Precision: 0.7
Recall: 0.63
Time: 35.32589077949524 s

Inception:

C: 0.01 Accuracy: 85.1
C: 0.1 Accuracy: 84.95
C: 1 Accuracy: 84.95
C: 10 Accuracy: 84.95
C: 100 Accuracy: 84.95
fin_C: 0.01
----Confusion Matrix----
[[167. 24.]
[ 33. 176.]]
Accuracy: 85.75
Precision: 0.8743455497382199
Recall: 0.835
Time: 87.5162410736084 s

The C selected is 0.1 and 0.01 for HOG and Inception respectively. The Accuracy given for final trained HOG is 68 and for Inception is 85.75. As expected the time taken for Inception is greater than that for HOG. Thus we can say that Soft Margin SVM with linear kernel gives pretty decent results with Inception.

**Q1.5) Hard Margin SVM with RBF kernel for HOG and Inception**

HOG:

C: 10000000000.0 G: 0.0625 Accuracy: 68.55
C: 10000000000.0 G: 0.125 Accuracy: 68.6
C: 10000000000.0 G: 0.25 Accuracy: 69.7
C: 10000000000.0 G: 0.5 Accuracy: 70.35
C: 10000000000.0 G: 1 Accuracy: 71.8
C: 10000000000.0 G: 2 Accuracy: 70.95
C: 10000000000.0 G: 64 Accuracy: 50.25
fin_C: 10000000000.0 fin_gamma: 1
----Confusion Matrix----
[[146. 57.]
[ 54. 143.]]
Accuracy: 72.25
Precision: 0.7192118226600985
Recall: 0.73
Time: 50.79240131378174 s

Inception:

C: 10000000000.0 G: 0.0625 Accuracy: 50.3
C: 10000000000.0 G: 0.125 Accuracy: 50.1
C: 10000000000.0 G: 0.25 Accuracy: 50.1
C: 10000000000.0 G: 0.5 Accuracy: 60.2
C: 10000000000.0 G: 1 Accuracy: 51.85
C: 10000000000.0 G: 2 Accuracy: 50.5
C: 10000000000.0 G: 64 Accuracy: 50.0
fin_C: 10000000000.0 fin_gamma: 0.5
----Confusion Matrix----
[[197. 150.]
[ 3. 50.]]
Accuracy: 61.75000000000001
Precision: 0.5677233429394812
Recall: 0.985
Time: 341.061824798584 s

The C =10000000000.0 and gamma = 1 for HOG and gamma = 0.5 for Inception. The Accuracy given for final trained HOG is 72.25 and for Inception is 61.75. As expected the time taken for Inception is greater than that for HOG. Here, the precision for Inception is 0.56 but the recall is almost 1. We can see clearly from the confusion matrix that this model was very conservative towards predicting negative class. Thus ending up getting almost all positive and getting the given recall and accuracy. These results in Inception are due to the fact that the SVM is hard margined and since no mistakes are allowed, almost all points are pushed to one side of the decision boundary. Thus resulting in HOG being the better extraction method for this specific model.

**Q1.6) Soft Margin SVM with RBF kernel for HOG and Inception**

HOG:

C: 0.01 G: 0.25 Accuracy: 65.75
C: 0.01 G: 2 Accuracy: 68.1
C: 0.01 G: 64 Accuracy: 50.2
C: 1 G: 0.25 Accuracy: 68.2
C: 1 G: 2 Accuracy: 69.6
C: 1 G: 64 Accuracy: 50.2
C: 100 G: 0.25 Accuracy: 67.15
C: 100 G: 2 Accuracy: 69.95
C: 100 G: 64 Accuracy: 50.3
fin_C: 100 fin_gamma: 2
----Confusion Matrix----
[[147. 64.]
[ 53. 136.]]
Accuracy: 70.75
Precision: 0.6966824644549763
Recall: 0.735
Time: 59.472586154937744 s

Inception:

C: 0.01 G: 0.25 Accuracy: 80.95
C: 0.01 G: 2 Accuracy: 50.4
C: 0.01 G: 64 Accuracy: 50.0
C: 1 G: 0.25 Accuracy: 50.05
C: 1 G: 2 Accuracy: 50.4
C: 1 G: 64 Accuracy: 50.0
C: 100 G: 0.25 Accuracy: 50.15
C: 100 G: 2 Accuracy: 50.4
C: 100 G: 64 Accuracy: 50.0

fin_C: 0.01 fin_gamma: 0.25
----Confusion Matrix----
[[167. 52.]
[ 33. 148.]]
Accuracy: 78.75
Precision: 0.7625570776255708
Recall: 0.835
Time: 429.2362320423126 s

For HOG, C = 100 and gamma = 2. For Inception, C = 0.01 and gamma = 0.25. Unlike part 1.5, the SVM is Soft Margin RBF so Inception features are giving an accuracy of 78.75 and HOG is giving 70.75 which is better than Hard Margin SVM. Expectedly, the time taken for Inception is greater than that for HOG. Unlike linear kernels, both HOG and Inception are giving Greater recall than precision in RBF kernels thus we can conclude that this model is a bit more liberal towards predicting positive. Finally, Inception is giving better results than HOG in this model.

**Q1.7) Soft Margin SVM with RBF kernel for HOG and Inception Multi Class**

HOG:

C: 0.01 G: 0.25 Accuracy: 29.35
C: 0.01 G: 2 Accuracy: 26.75
C: 0.01 G: 64 Accuracy: 18.3
C: 1 G: 0.25 Accuracy: 33.85
C: 1 G: 2 Accuracy: 35.95
C: 1 G: 64 Accuracy: 10.85
C: 100 G: 0.25 Accuracy: 33.1
C: 100 G: 2 Accuracy: 36.7
C: 100 G: 64 Accuracy: 11.25
fin_C: 100 fin_gamma: 2
----Confusion Matrix----
[[21. 0. 6. 4. 2. 3. 2. 2. 1. 0.]
[ 2. 8. 5. 4. 6. 1. 1. 1. 5. 2.]
[ 4. 3. 5. 2. 0. 1. 0. 1. 3. 1.]
[ 2. 4. 7. 11. 3. 5. 6. 3. 4. 1.]
[ 0. 14. 5. 9. 20. 3. 1. 2. 7. 3.]
[ 4. 3. 2. 1. 0. 16. 2. 4. 1. 6.]
[ 3. 1. 3. 3. 1. 6. 20. 5. 2. 0.]
[ 3. 0. 2. 2. 2. 1. 4. 12. 6. 0.]
[ 0. 7. 4. 1. 5. 0. 2. 9. 11. 7.]
[ 1. 0. 1. 3. 1. 4. 2. 1. 0. 20.]]

Accuracy, avg: 36.0
Accuracy, class: [50. 27.5 30. 50. 40. 50. 27.5 12.5 20. 52.5]
Precision_micro: 0.36
Precision_macro: 0.36273898909503366
Recall_micro: 0.36
Recall_macro: 0.36
F1_micro: 0.36
F1_macro: 0.3557294116253149
Time: 98.51937556266785 s

Inception:

C: 0.01 G: 0.25 Accuracy: 51.15
C: 0.01 G: 2 Accuracy: 10.65
C: 0.01 G: 64 Accuracy: 10.0
C: 1 G: 0.25 Accuracy: 38.4
C: 1 G: 2 Accuracy: 10.65
C: 1 G: 64 Accuracy: 10.0
C: 100 G: 0.25 Accuracy: 38.4
C: 100 G: 2 Accuracy: 10.65
C: 100 G: 64 Accuracy: 10.0
fin_C: 0.01 fin_gamma: 0.25
----Confusion Matrix----
[[24. 0. 9. 3. 0. 2. 1. 0. 4. 3.]
[ 1. 21. 2. 3. 8. 0. 1. 6. 7. 4.]
[ 3. 0. 7. 4. 0. 0. 0. 1. 0. 1.]
[ 4. 1. 5. 18. 1. 0. 0. 0. 0. 2.]
[ 1. 6. 2. 1. 24. 0. 0. 0. 2. 0.]
[ 0. 0. 1. 0. 0. 29. 2. 2. 1. 2.]
[ 2. 0. 0. 0. 0. 0. 19. 1. 1. 1.]
[ 0. 1. 1. 0. 1. 0. 1. 18. 1. 0.]
[ 0. 8. 1. 2. 5. 1. 0. 7. 19. 0.]
[ 5. 3. 12. 9. 1. 8. 16. 5. 5. 27.]]
Accuracy, avg: 51.5
Accuracy, class: [67.5 47.5 45. 47.5 72.5 60. 45. 17.5 52.5 60. ]
Precision_micro: 0.515
Precision_macro: 0.5699400281408311
Recall_micro: 0.515
Recall_macro: 0.5149999999999999
F1_micro: 0.515
F1_macro: 0.5186844029433619

Time: 647.4610526561737 s

For HOG, C = 100 and gamma = 2. For Inception, C = 0.01 and gamma = 0.25. The overall accuracy for HOG is 36 and for Inception is 51.5. Clearly, Inception is using more time than HOG but is also performing better than HOG. Thus, looking at the accuracy, this model is not very useful for classifying Multi Class data.

**Q1.8) Hard Margin SVM with Poly kernel for HOG and Inception Multi Class**

HOG:

C: 10000000000.0 G: 0.25 D: 3 Accuracy: 32.4
C: 10000000000.0 G: 2 D: 3 Accuracy: 32.4
C: 10000000000.0 G: 64 D: 3 Accuracy: 32.4
C: 10000000000.0 G: 0.25 D: 5 Accuracy: 35.15
C: 10000000000.0 G: 2 D: 5 Accuracy: 35.15
C: 10000000000.0 G: 64 D: 5 Accuracy: 35.15
C: 10000000000.0 G: 0.25 D: 7 Accuracy: 36.0
C: 10000000000.0 G: 2 D: 7 Accuracy: 36.0
C: 10000000000.0 G: 64 D: 7 Accuracy: 36.0
fin_degree: 7 fin_gamma: 0.25 fin_C: 10000000000.0
----Confusion Matrix----
[[20. 1. 5. 4. 2. 5. 3. 4. 1. 1.]
[ 2. 14. 5. 2. 6. 3. 1. 1. 5. 1.]
[ 4. 3. 7. 2. 0. 1. 0. 1. 4. 1.]
[ 2. 4. 2. 11. 3. 4. 5. 4. 3. 2.]
[ 0. 11. 8. 10. 20. 1. 2. 1. 6. 4.]
[ 4. 2. 1. 2. 0. 15. 1. 3. 2. 6.]
[ 3. 2. 4. 4. 1. 6. 20. 4. 3. 0.]
[ 3. 1. 2. 1. 2. 1. 3. 12. 5. 1.]
[ 0. 2. 4. 1. 3. 0. 3. 9. 11. 4.]
[ 2. 0. 2. 3. 3. 4. 2. 1. 0. 20.]]
Accuracy, avg: 37.5
Accuracy, class: [50. 27.5 30. 50. 37.5 50. 27.5 17.5 35. 50. ]
Precision_micro: 0.375
Precision_macro: 0.37487239458345967
Recall_micro: 0.375
Recall_macro: 0.375
F1_micro: 0.375
F1_macro: 0.36984179471117784
Time: 91.69164967536926 s

Inception:

C: 10000000000.0 G: 0.25 D: 3 Accuracy: 64.45
C: 10000000000.0 G: 2 D: 3 Accuracy: 64.45
C: 10000000000.0 G: 64 D: 3 Accuracy: 64.45
C: 10000000000.0 G: 0.25 D: 5 Accuracy: 25.2
C: 10000000000.0 G: 2 D: 5 Accuracy: 25.2
C: 10000000000.0 G: 64 D: 5 Accuracy: 25.2
C: 10000000000.0 G: 0.25 D: 7 Accuracy: 16.15
C: 10000000000.0 G: 2 D: 7 Accuracy: 16.15
C: 10000000000.0 G: 64 D: 7 Accuracy: 16.15
fin_degree: 3 fin_gamma: 0.25 fin_C: 10000000000.0
----Confusion Matrix----
[[32. 0. 8. 1. 0. 0. 0. 0. 1. 0.]
[ 1. 31. 4. 8. 6. 0. 1. 0. 10. 4.]
[ 3. 0. 16. 2. 0. 0. 1. 0. 0. 0.]
[ 3. 0. 8. 27. 2. 1. 0. 1. 0. 1.]
[ 0. 2. 0. 0. 28. 0. 0. 0. 0. 0.]
[ 0. 0. 0. 0. 0. 31. 1. 0. 0. 1.]
[ 0. 0. 0. 0. 0. 0. 27. 1. 1. 1.]
[ 0. 2. 3. 1. 2. 6. 8. 36. 2. 1.]
[ 1. 5. 1. 1. 2. 2. 1. 2. 25. 3.]
[ 0. 0. 0. 0. 0. 0. 1. 0. 1. 29.]]
Accuracy, avg: 70.5
Accuracy, class: [72.5 62.5 90. 67.5 77.5 70. 67.5 40. 77.5 80. ]
Precision_micro: 0.705
Precision_macro: 0.7473777969803206
Recall_micro: 0.705
Recall_macro: 0.7050000000000001
F1_micro: 0.705
F1_macro: 0.7090621411306215
Time: 706.1015276908875 s

For HOG, D = 7 and gamma = 0.25. For Inception, D = 3 and gamma = 0.25. The overall accuracy for HOG is 37.5 and for Inception is 70.5. Clearly, Inception is using more time than HOG but is also performing better than HOG. Thus, looking at the accuracy, this model performs decently for classifying Multi Class data using Inception features.

**Q1.9) Compare the performance of models constructed in previous sections**

- Which feature extraction method yields better results:
  Overall, if the chosen model is suitable, the Inception feature extraction method works better than the HOG feature extraction method. The main reason for this is the high number of features in Inception Feature extraction method.
- Which model and feature combination performed best/worst:
  Super Class Classification:
  The best model is the Logistic regression model with Full batch gradient ascent with an accuracy of 87.75. In SVM, the best model is again soft margin linear kernel SVM with C = 0.01 and using Inception features. The worst performing combination is of Hard margin SVM using rbf kernel with Inception Features with Gamma = 1. The model predicted almost all points as positive and thus got a 60% accuracy.
  Sub Class Classification:
  The best was polynomial kernel with d = 3, g = 0.25 with Inception Features and the worst was soft margin rbf kernel with c = 100, g = 2 with HOG Features
- The effect of C on the decision boundary of SVM:
  The C parameter in SVM controls the amount of error allowed in the model. The greater the value of C, the smaller the margins of the decision boundary and the more the decision boundary tries to fit every point on either side of the boundary and thus creating a non generalizable model and prone to overfitting. In linear kernel, while using HOG features, the C was neither chosen as 0.01 nor 100 but rather 0.1 which is somewhere in the middle to create a generalizable decision boundary which neither oversfit nor underfit the data. In other kernels, C had the same effect on the decision boundary. On the other hand, Inception features selected C as 0.01 in linear kernel but still since this is not a very small value, the decision boundary did not underfit to the data. In other kernels, the C was controlled to be mostly under 1.
- The effect of gamma on the decision boundary of SVM with RBF kernel:
  The greater the gamma, the greater the overfitting is observed, and the smaller the gamma, the greater the underfitting is observed. In RBF kernels, the gamma was chosen to be 2 for HOG features as it was where the decision boundary was shaped the best while Inception features usually required gamma as 0.25 to avoid overfitting and avoid underfitting of the decision boundary.
- The effect of d on the decision boundary of SVM with polynomial kernel:
  The degree of the polynomial kernel controls the shape of the decision boundary to be more and more flexible as the degree increases. The greater the degree goes, the higher the model is prone to overfit new data. The lower the degree, the more the model is prone to underfit to new data. In HOG features, degree 7 was chosen to be the one making the best decision boundary while 3 was chosen for Inception features.
- The effects of different (C, γ) pairs on the decision boundary and tolerance of SVM:

As the value of C increases, the error tolerance of SVM decreases and it creates a hard margin SVM. In Inception features, when the SVM was hard margin, the gamma was chosen as suitable as possible which was 0.25 to try to overcome the overfitting done by the huge C value. The same case was seen in hog in hard margin SVM. In soft margin svm on the other hand, Inception features chose a (0.01,0.25) pair and the increase of either of these parameters caused huge accuracy problems. In HOG features, the pair (100, 2) was chosen. The decrease in C did not affect the decision boundary and tolerance of SVM alot but changing the gamma parameter affected the shape of the decision boundary to either overfit or underfit to the data and cause problems in the accuracy of prediction of new data.

- The effects of (d, γ) pairs on the decision boundary of SVM with polynomial kernel:
  As the value of d increases, the decision boundary is prone to overfitting as the curve created is more flexible and attempts to go through all data points while decreasing d makes the decision boundary prone to underfitting. In inception features, the pair (3, 0.25) is chosen while in HOG, the pair (7, 0.25) is chosen. In HOG decreasing the d from 7 causes underfitting while in Inception increasing d from 3 causes overfitting. The effect of the gamma in this pair is not seen when it is changed and the value remains the same.
- The effect of batch size to performance and training time of logistic regression models:
  In both HOG features and Inception features, the time increased as the batch size was decreased. The time was highest in stochastic while the least in full batch ascent. It is due to the sheer number of times the weights have to be recalculated as the size of the batch is decreased. The performance however is another story. The performance here is given by Full Batch in both HOG and Inception. Even though Full Batch is not supposed to do as good as stochastic or mini batch, it still does good in performance because there is no local minima/maxima in logistic regression one-piece function and all of these ascents converge to one maxima.
- The use of γ parameter for polynomial kernel:
  The gamma parameter in our experiments does not change any performance in our model but usually it is used to scale the features in the polynomial kernel.
- What are the advantages and disadvantages of logistic regression and SVM models compared to each other?
  Firstly, logistic regression works on linearly separable data while there is no such restriction on SVM. Also, the SVM has support vectors which ensures that the linear decision boundary between classes is not just separating the data but is also the best separating line between the data. Logistic regression on the other hand requires an increasing degree of polynomial manually if non linear data is to be fit. The SVM on the other hand can do this using kernels which create non linear decision boundaries using efficient methods. The advantage of Logistic regression in linearly separable data is that

logistic regression takes quite less time than SVM in training. Also, in SVM multiple hyper parameters are added which have to be tweaked which is clearly a disadvantage.

**Q1.10) In what cases NPV, FPR, FDR and F1 score would be more informative compared to accuracy, precision, recall alone? Discuss if accuracy is a reliable metric for subclass and superclass classification tasks.**

We would only use accuracy and precision, recall when there is no class imbalance. NPV, FPR, FDR and F1 score will be used when there is an imbalance in the classes as F1 score uses precision and accuracy both to make a metric which would not be affected by class imbalance. Accuracy is a reliable metric for subclass and superclass classification tasks because in the datasets that we have been given, the classes are not imbalanced and all classes are equally included in the data. Thus accuracy will be a reliable metric for both superclass and subclass.

**Q2.1) SVD  based implementation of PCA to X.**

MSE:  1.56950838762225e-16

Time:  38.3919951915741 s

Since in SVD implementation we get USV components of the original matrix, we can easily get a very accurate representation of the original matrix by applying dot product to all of these USV matrices. Thus, resulting in the low MSE.

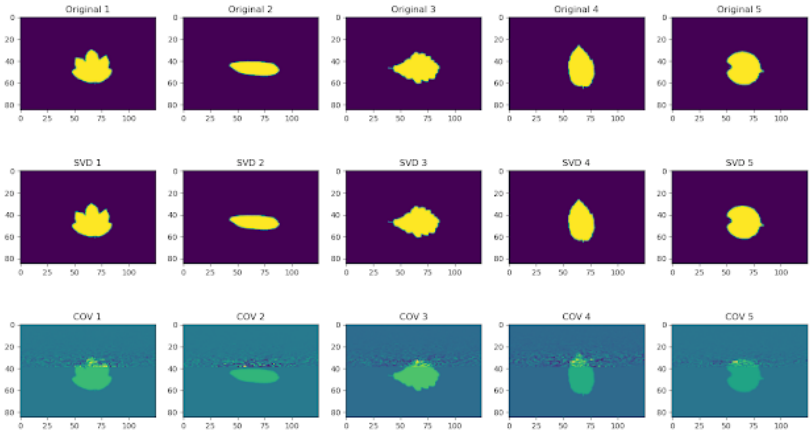
**Q2.2) Covariance  based implementation of PCA to X.**

MSE:  0.025924433896687384

Time: 184.17397046089172  s

Since in Covariance implementation we get the eigen values and eigen vectors of the original matrix. We get the original matrix by applying dot product to the reduced matrix with the eigen value matrix thus getting a not very accurate representation of the original matrix. Also getting the covariance matrix is a time consuming procedure, cov implementation takes more time than svd implementation.

**Q2.3) COV vs SVD images:**




According to the reconstructed images, we can see that the original image and the SVD reconstruction is quite better than the COV one as the reconstructed image in COV is a little distorted and its color scheme is also disturbed. Precision of SVD is obviously better than COV. Also regarding timing, SVD takes less time than COV based reconstruction. SVD implementation is also more numerically stable due to the fact that the eigen values extracted from COV based PCA are extracted from the COVARIANCE matrix of the original matrix. The condition number of the eigen values from the original matrix X is less than the condition value of eigen values from the original matrix itself. The condition value represents how sensitive the value is to change from a little change in the input. Thus, since a little change in the input matrix X, is bound to change the eigen value of COV based PCA by a lot, the numerical stability of SVD paced PCA is greater. MSE is not a reliable method for comparing similarity because it does not account for variability in data. In that case, $R^2$ is a better measure as it creates more generalizable values. It is calculated as (Sum of Square Regression)/(Sum of Square Total).

**Q2.4) Running times COV vs SVD:**

**1 - n samples >> n features:**

In this case, COV based implementation will do better than SVD in terms of time because creating a covariance matrix of very small (feature x feature) size takes less time and then extracting eigen values and vectors from said small matrix takes very little time than extracting eigen vectors from square matrices (sample x sample) and (feature x feature) matrix as in SVD. Thus SVD will take more time in this case than COV.

**2 - n samples == n features:**

In this case, SVD based implementation will do better than COV in terms of time because creating a covariance matrix of very large (feature x feature) size takes more time and then extracting eigen values and vectors from said large matrix takes a lot of time than just extracting eigen values from large square matrices (sample x sample) and (feature x feature) matrix as in SVD. The overhead becomes the covariance matrix that has to be found in COV based

implementation. Even though eigen vectors have to be found from large square matrices (sample x sample) and (feature x feature) matrix, we still do not have to waste time in getting a COV matrix. Thus COV will take a little more time in this case than SVD. As a rule of thumb, covariance will take more time if the matrix for covariance becomes large and large. And that will be when features increase in size.

**3 - n samples << n features:**

In this case, SVD based implementation will do better than COV in terms of time because creating a covariance matrix of very large (feature x feature) size takes more time and then extracting eigen values and vectors from said large matrix takes a lot of time than just extracting eigen vectors from square matrices (sample x sample) and (feature x feature) matrix as in SVD. The overhead becomes the covariance matrix that has to be found in COV based implementation. Thus COV will take more time in this case than SVD.