**Assignment 2 Report**

CPS 584 - Advanced Intelligent Systems and Deep Learning

University of Dayton

March 27, 2023

Authors:

Connar Hite

Ranjita Piratla

Sean Saud

Monika Somu

T.Shiva Harshith Varma

**1 Introduction**

The purpose of this assignment is to get hands-on experience with the K-Nearest Neighbors (KNN) and multilayer perceptron Neural Networks (NN) methods for image classification using Haar-like features and Deep Learned Features at the fc7 layer of AlexNet. A set of images of flowers is provided which contains two classes: *Rose* and *Tulip*. For each class, 40 training images and 20 testing images are given. 30 additional training images per class are used, which were collected in the previous assignment. Before extracting the Haar-like features, The images are initially gray-scaled and resized to the size of 50 x 50. Then, Haar-like features are extracted from each training image using its integral image. Before extracting the Deep Learned Features at the fc7 layer of AlexNet, the images are resized to 227x227. Next, the KNN algorithm (with K ϵ {1, 3, 5, 7, 9}) is performed on the testing images twice: once using Haar-like features as input, and the other using deep learned features as the input. Then, the NN algorithm is performed twice on the testing images using different numbers of hidden layers and hidden layer nodes. Again, NN is performed once using Haar-like features and again using deep learned features. The accuracy rate of each class is reported (KNN-Haar, KNN-Deep Learned Features, NN-Haar, NN-Deep Learned Features). Next, all the training images are flipped in order to have more training data. Then, the KNN algorithm is performed (with K ϵ {1, 3, 5, 7, 9}) with the additional training images. Finally, the Haar-like features and deep learned features are concatenated for each training and testing image, and the KNN algorithm is performed using the concatenated features. For each system, the input is 30 Haar-like features or 4096 deep learned features of each image, and the output is a predicted class label: *Rose* or *Tulip*. The accuracy rate for each class, as well as the overall accuracy, are reported for each instance of both classification methods.

**2 Problems**

**2.1 Extracting Haar-like Features**

For extracting the Haar-like features from the given images, we first calculated pixel values of the integral image. We implemented a function for doing this. In this function, there are three cases. Firstly, for the first pixel, the value in the integral image remains the same as in the original image.

int\_img(1,1)=img(1,1)

Secondly, for the pixels in the first row and first column, the value is the sum of the current pixel value and previous pixel value of the integral image.

int\_img(i,1)=img(i,1)+int\_img(i-1,1)

int\_img(1,j)=img(1,j)+int\_img(1,j-1)

Thirdly, for the rest of the pixels, we add the pixel value of the current pixel with the integral image pixel to the top and right of the current value. Then subtract the repeated pixel area from it.

int\_img(i+1,j+1)=img(i+1,j+1)+int\_img(i,j+1)+int\_img(i+1,j)-int\_img(i,j)

We then created a matrix of 30 dimensions of estimated coordinates of white regions of the given 30 rectangles. Using this, we found the sum of white and black regions of each of the testing and training data. We then extracted the haar-like features using the given formula for the rectangle sum given an integral image.

**2.2 Extracting Deep Learned Features using AlexNet**

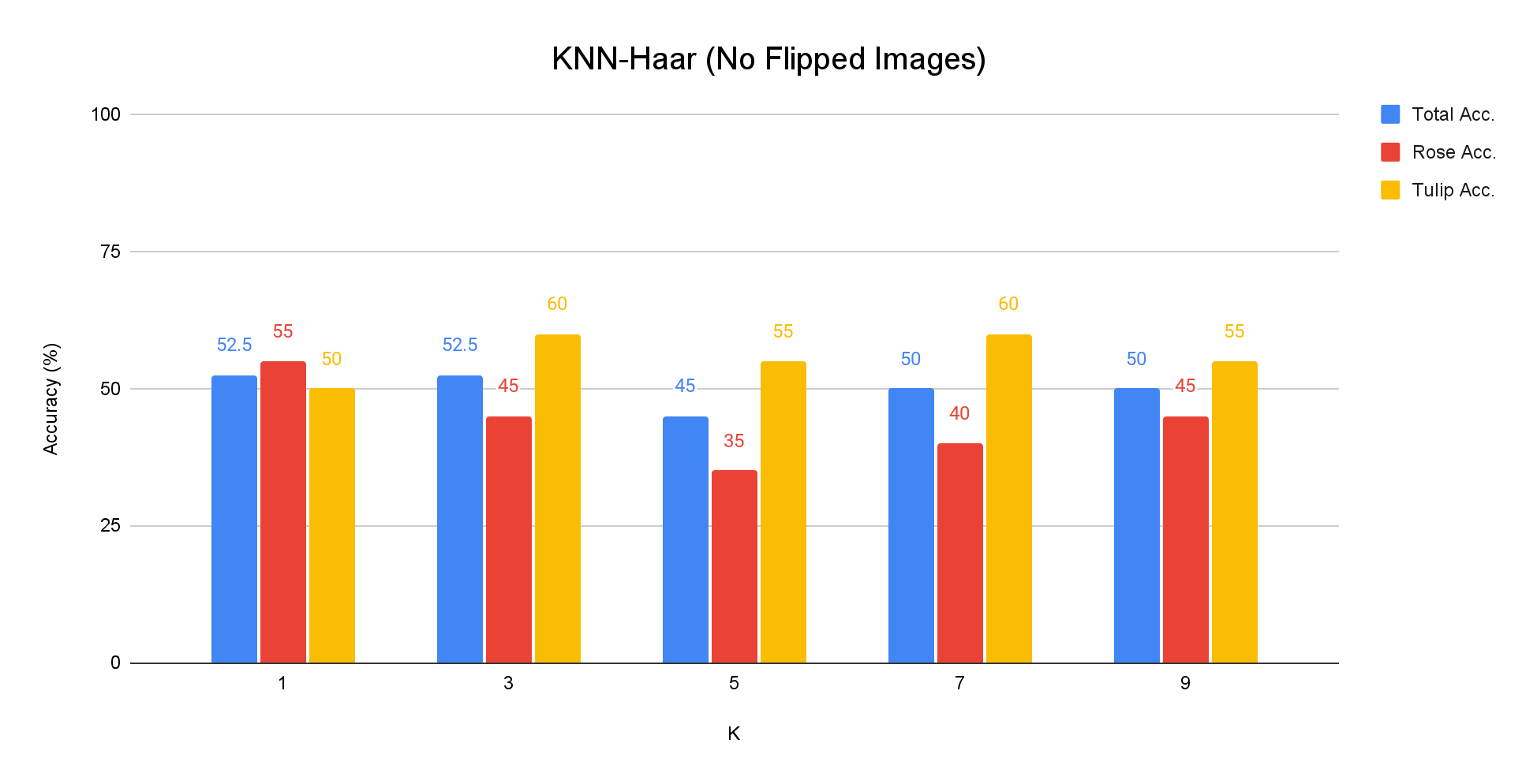
In this question, we were asked to extract the Deep Learned features from the layer fc7 of AlexNet. To do so, we first load the pretrained network: “net”. Tracing “net”, we found layer 18 is fc7. To extract the deep learned features from that particular layer, we used the equation below. Doing this meant that feeding the input images into a simple neural network, with the structure of “net”, resulted in our goal of the Deep Learned features.

net.layers = net.layers(1, 1:18)

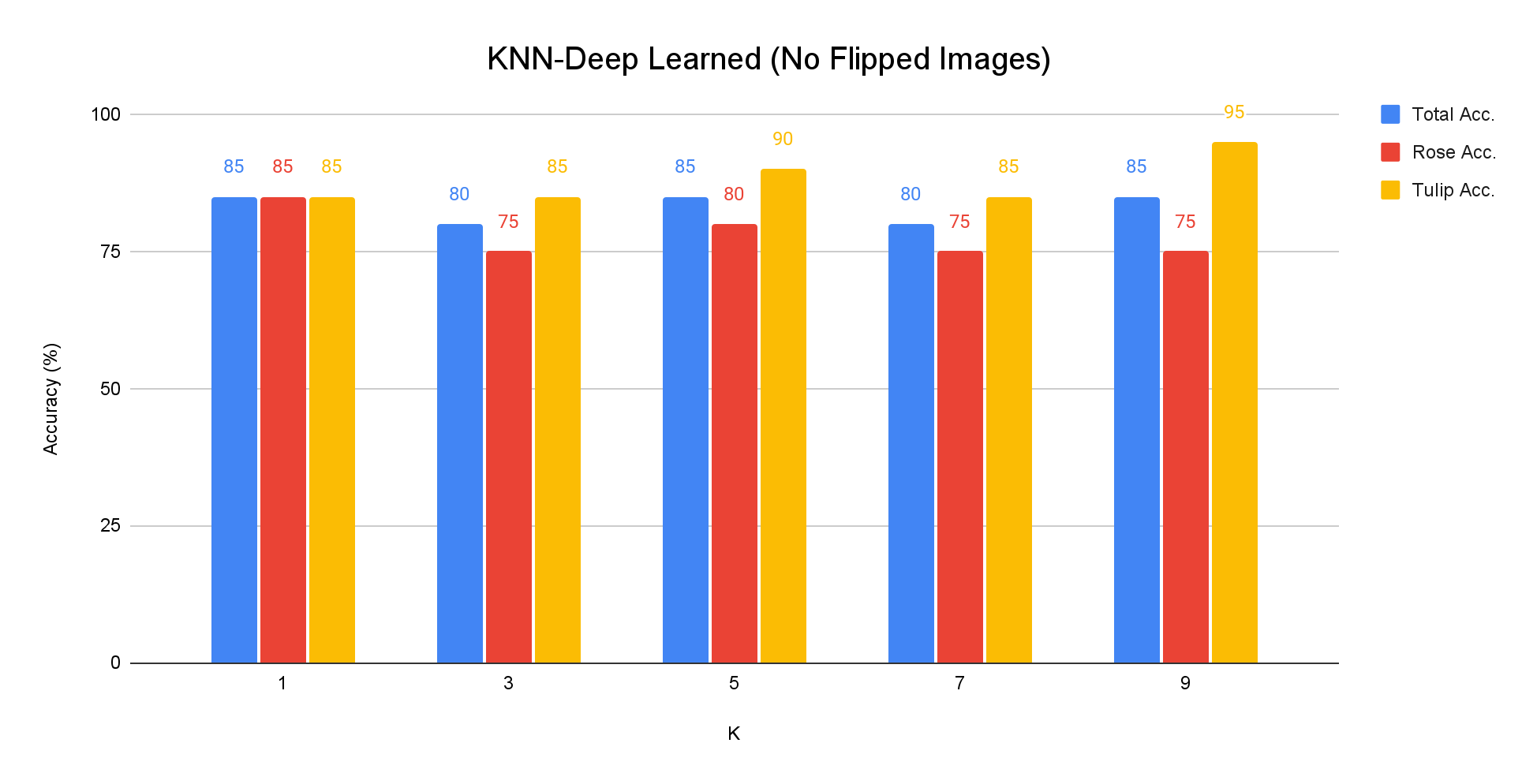
However, the input images for this network are different from the Haar-like features algorithm. Instead of a 50x50 gray-scaled image, the input for the Deep Leaned algorithm is a 227x227 color image. So, each image needs to be resized. Each channel of the color images (red, blue, green) is also normalized based on a provided parameter from AlexNet. Now the input images are ready to be processed.

**2.3 K-Nearest Neighbors (KNN) and Neural Networks (NN)**

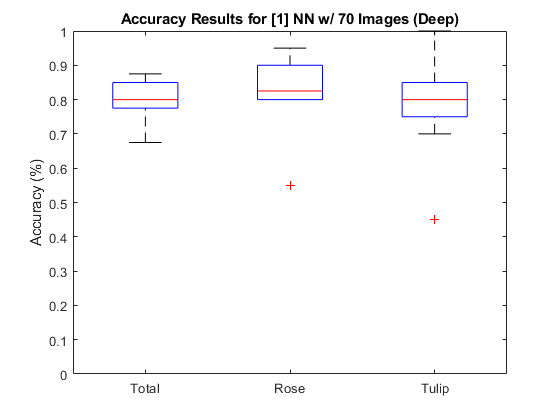
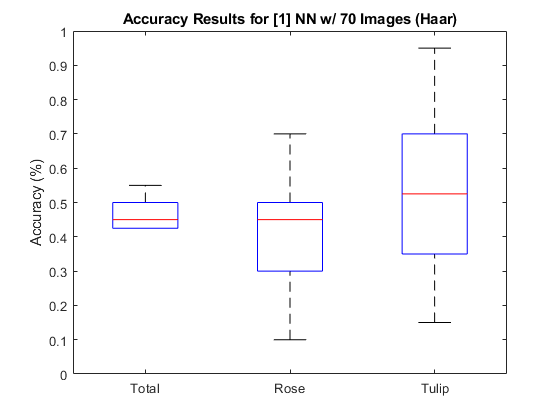
The algorithms used to compute the KNN and NN results are the same as the first assignment. The inputs for them being the Haar-like features and Deep Learned features extracted from the previous sections. The K-Nearest Neighbors algorithm uses K ϵ {1, 3, 5, 7, 9} for each of the problems. As for the Neural Network algorithm, multiple separate tests were performed. Each test has a different number of nodes and hidden layers. To account for the randomness during the training process, each NN is trained and tested multiple times. Doing so allows for a visualization of the consistency of the system. However, this was not done for all tests performed on the Deep Learned features. Given the large number of inputs (4096 per image), the training process takes much longer than with Haar-like features. Thus, some of the later tests were only performed once. There is also the issue of how much memory is being used. If the number of nodes used for the Deep Learned set is too large, an error will occur indicating that too much memory will be used. The KNN algorithm is run for Haar-like features and Deep Learned features, the results of which are shown below in Figure 1 and Figure 2 respectively. The NN algorithm is run for the same set of features, the results of which are shown in Figure 3 and Figure 4. The former shows the results of a NN structure of [1] and the latter shows the results of a NN structure of [5]. Figure 5 shows the results of a NN trained on the Deep Learned features with the largest hidden layer size that would not cause an error: [10].



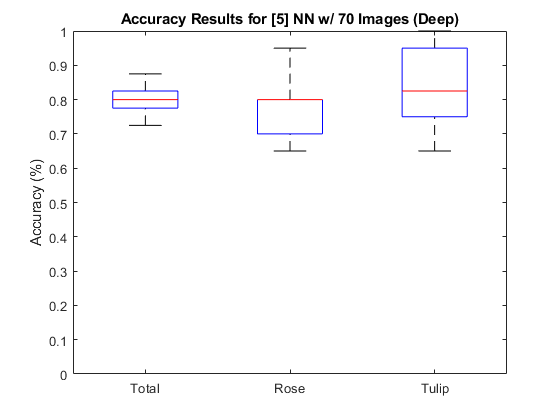
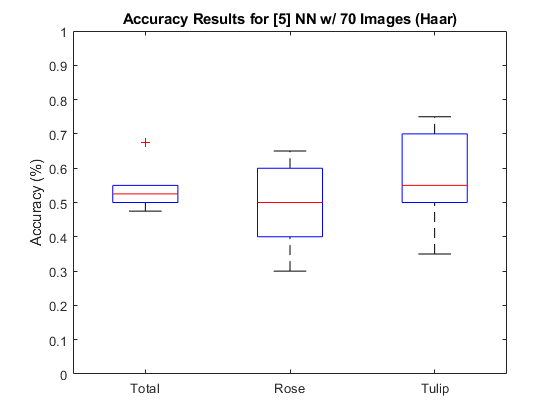
**Figure 1: KNN-Haar Results w/ No Flipped Images**



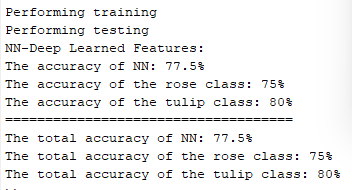
**Figure 2: KNN-Deep Learned Features Results w/ No Flipped Images**

****

**Figure 3: [1] NN Results for Haar (Left) and Deep Learned (Right)**

****

**Figure 4: [5] NN Results for Haar (Left) and Deep Learned (Right)**

****

**Figure 5: [10] NN Results for Deep Learned Features**

**2.4 Additional, Flipped Images (KNN)**

For the classes of Rose and Tulip, we have gathered 30 new pictures, which was done in the previous assignment. These were added to the other 40 training images that had already been provided to create a total of 70 images for each class. This resulted in better results overall. To create a similar effect, we flip our existing images and add them to the training set. This effectively doubles the number of training inputs, without needing to find new images. Shown below is part of the algorithm that adds these images. If we choose to add the flipped images, then the already imported and processed training images are gathered and flipped. We decided to flip them horizontally.

if flipped\_images

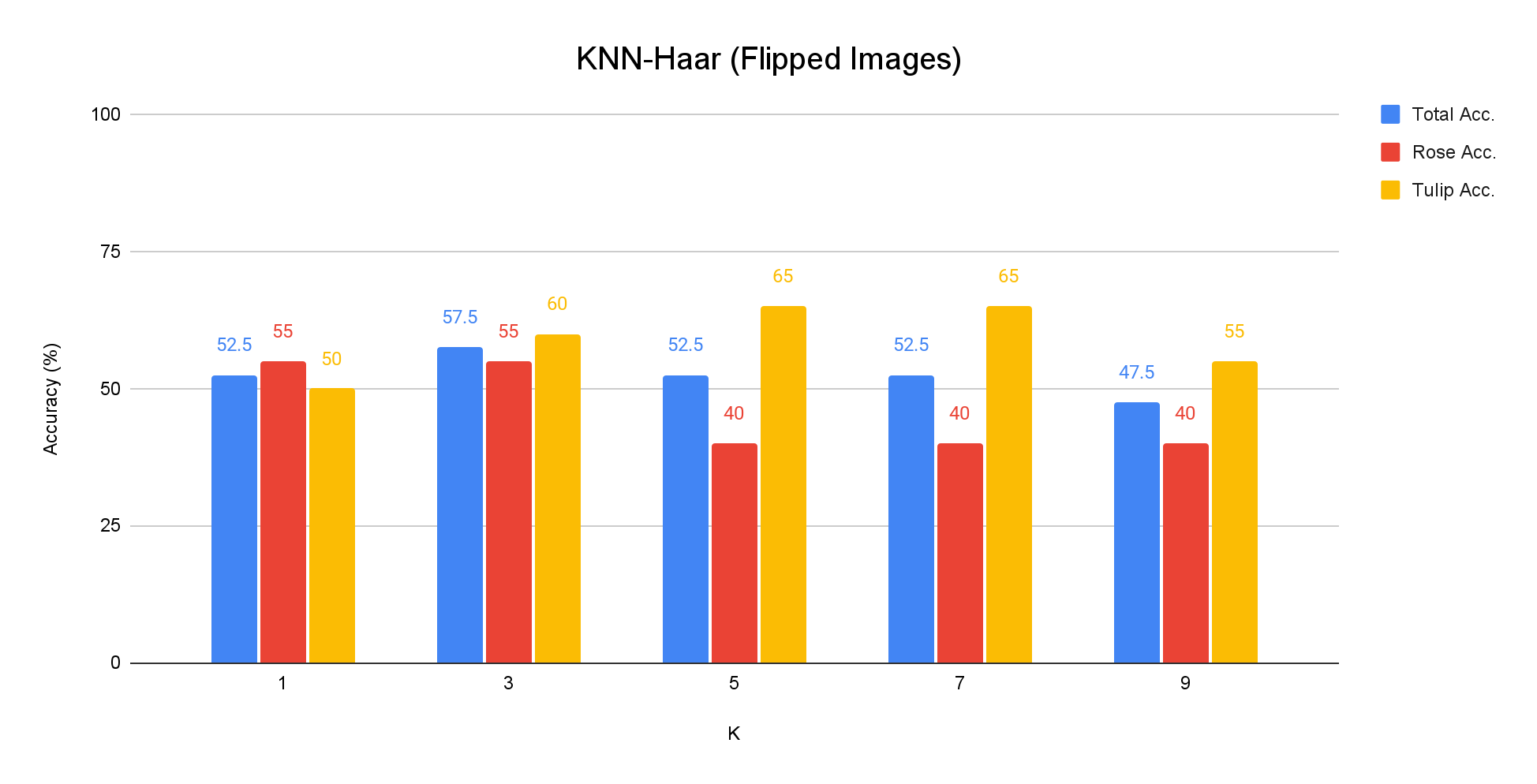
training\_images{(training\_amount\*(i-1))+d+training\_amount/2}=flip(training\_images{(training\_amount\*(i -1)) + d}, 2);

training\_images\_haar{(training\_amount\*(i-1))+d+training\_amount/2}=flip(training\_images\_haar{(training\_amount \* (i -1)) + d}, 2);

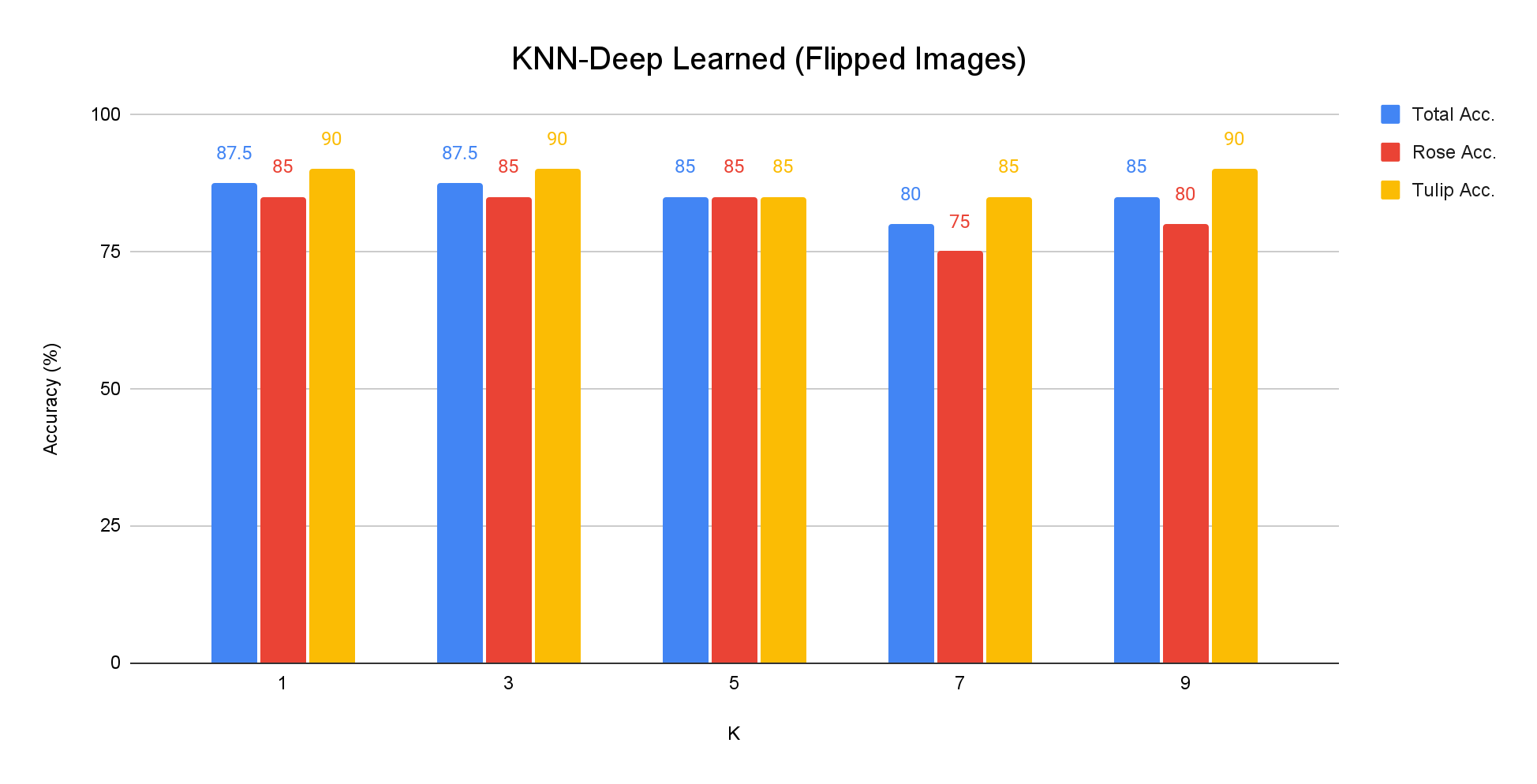
training\_labels((training\_amount \* (i -1)) + d + training\_amount/2) = i-1;

The training images are then processed to create the Haar-like features, and they are saved in the training images haar variable. If the flipped images variable is set to 1, the labels are applied to each image and the flipped images are added to the training set.

The accuracy rate for each class is then reported using Haar-like and Deep Learned features. The accuracy of KNN-Haar and Deep Learned features can be seen in the figures below, Figure 6 and Figure 7 respectively.



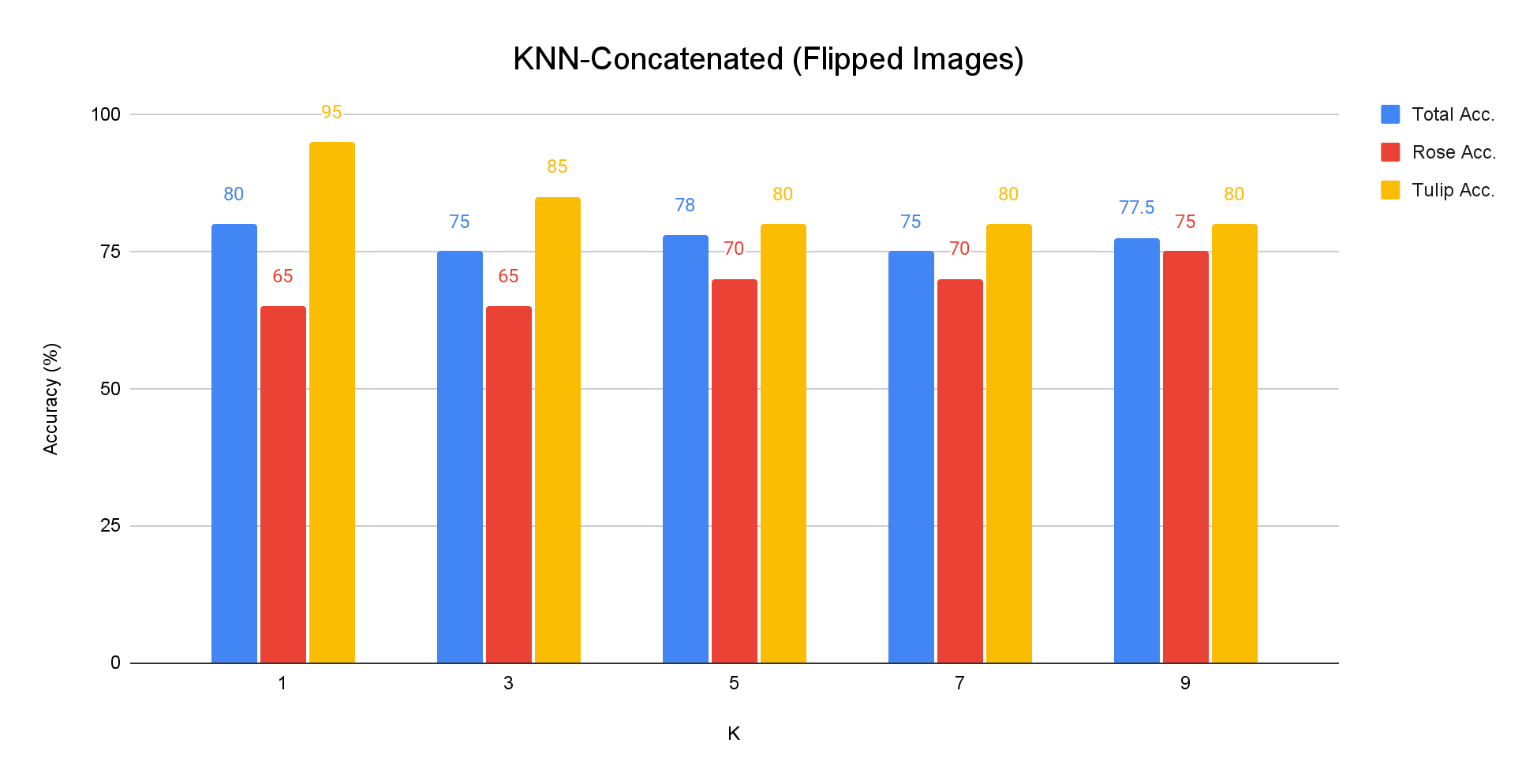
**Figure 6: KNN-Haar Results w/ Flipped Images**



**Figure 7: KNN-Deep Learned Features Results w/ Flipped Images**

**2.5 Concatenated Features (KNN)**

Using the same training set in (2.4), the Haar-like features and the Deep Learned features are horizontally concatenated to be used as input into a KNN system. This results in a 4126 dimensional input for the KNN. Then we performed KNN with different K values. Also we calculated the accuracy rate for each class. The results of which are shown below in Figure 8.



**Figure 8: KNN-Concatenated Features (Haar + Deep Learned Features)**

**3 Issues Encountered & Solutions**

* When training the Neural Network for the Deep Learned features, an error was encountered involving the amount of memory being used for the process. Part of the issue was the size of the hidden layers. The error would only occur if we increased the hidden layer node count too much. To solve this problem, we either needed to decrease the number of input nodes or the size of the hidden layer. Given the task at hand, we chose to keep the size of the hidden layer smaller. This also decreased the training time.
* Each image had a different quality while reporting the accuracy rate of each class by adding 30 more images. The accuracy of classifiers can be greatly influenced by the quality of the images used for training and testing. Low-resolution or poorly captured images may lack the necessary features to distinguish between classes. For training and testing, we made certain that high-quality images were used.

**4 Discussion**

In the KNN systems, the use of Deep Learned features resulted in much higher classification accuracy (around 30%) compared to the use of Haar-like features. Using different values of K yielded slightly different results, but the overall accuracy was consistently around 50% using Haar-like features and 80% using Deep Learned features. While the use of Deep Learned features provides better performance, it comes at the cost of being much more computationally expensive. This is due to there being 4096 features per image, compared to 30 for Haar-like. The Neural Networks did not perform much better, but were generally a couple percentage points better than the KNN. The overall accuracy was generally around 50% using Haar-like features and 80% using Deep Learned features. However, the NN has a degree of randomness when training. Meaning there are two things to consider when evaluating the system: average accuracy and spread of results. The general trend found was that having an increased amount of inputs would decrease the spread of results. This makes the system more consistent. Increasing the hidden layer size provided the best result for the spread of accuracies in the NN. The average accuracy in the Deep Learned systems, as shown in Figure 3 and Figure 4, were roughly the same at about 80%. However, increasing the size of the hidden layer did decrease results spread. For the system with one hidden node, the results varied between about 68% and 90%. The system with five hidden nodes had results that varied between about 72% and 88%. Increasing the node count further drastically increased training time. Thus, for the test done with ten nodes, as shown in Figure 5, only one test was conducted. This makes judging the spread impossible. However, the accuracy falls within the same range as the other tests.

After flipping the training images to have more training data (140 training images per class), the KNN system classified the flowers more accurately. The use of Haar-like features resulted in around 52.5% accuracy with the highest overall accuracy being 57.5% when K = 3. The use of Deep Learned features resulted in around 85% total accuracy with the highest overall accuracy being 87.5% when K = 1 and K = 3. Larger values of K tended to result in lower accuracy when using 140 training images per class. Concatenating the Haar-like features and the Deep Learned features did not improve performance compared to using only the Deep Learned features. It resulted in accuracy rates around 78% with the highest accuracy being 80% when K = 1. The decrease in performance is likely due to the fact that the Haar-like features used do not capture the most significant features of the flowers to help distinguish roses from tulips. Adding the Haar-like features essentially added noise to the data that decreased the performance of the system.

**5 Conclusion**

Using Haar-like features and Deep Learned features at the fc7 layer of AlexNet, we were able to create K-Nearest Neighbor and Neural Network systems to classify images of flowers as either a rose or tulip in this project. Testing using both 70 training images per class and 140 training images (via flipping the images), the KNN system (with K ϵ {1, 3, 5, 7, 9}) and the NN system were trained and tested using both Haar-like features and Deep Learned features. This project demonstrates that the features used as input to a machine learning algorithm can be just as, if not more, important than the machine learning algorithm used. The use of Deep Learned features resulted in better performance than Haar-like features when using a KNN system or a NN system. For this classification task, smaller values of K tended to result in higher performance. It is also shown that increasing the amount of training data leads to a better performance. In addition, this project demonstrates that concatenating “bad” features to “good” features decreases performance compared to using only the “good” features.

Even though the algorithms were able to achieve relatively high performance (when using Deep Learned features in particular), there is always room for improvement. For example, the max memory issue could be further investigated and fixed. One way to do this would be to design a feature selection method that selects only the most important features to be used as input into the classification system. This would decrease the dimensionality and possibly increase performance. If the max memory issue is fixed, it would then be much more feasible to test different hidden layer structures for the NN. Also, different deep feature extraction methods could be tested to try to achieve higher performance.

**5 Member Contributions**

Connar and Sean wrote the code individually, coming together to compare results afterwards. The other members were able to complete the first two problems, but were unable to finish the code individually. Thus, they used Connar and Sean’s completed code as reference. The submitted code was written by Connar, with corrections being made with the help of other group members.

Connar Hite: Wrote the K-Nearest Neighbors (KNN) and Neural Networks (NN) section and

edited paper/powerpoint

Ranjita Piratla: Wrote Additional, Flipped Images (KNN) section

Sean Saud: Wrote the Introduction and Conclusion in report, Results and Conclusion slides in presentation, edited report/powerpoint

Monika Somu: Wrote Extracting Deep Learned Features using AlexNet section

T.Shiva Harshith Varma: Wrote Concatenated Features (KNN) section

**6 References**

Parts of this assignment reference “Assignment 1” and “Lab 5” in Dr. Tam Nguyen’s *Advanced Intelligent Systems and Deep Learning* class (CPS 584).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84-90