CS 529 – Assignment 3

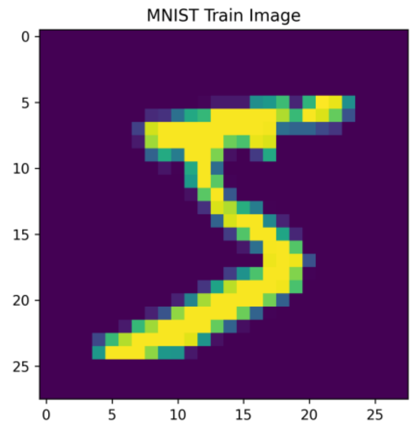
Group: Christopher Gilbert, Ahmaide Awawda, Patrick Jojola, Desiree Forster

04/09/2025

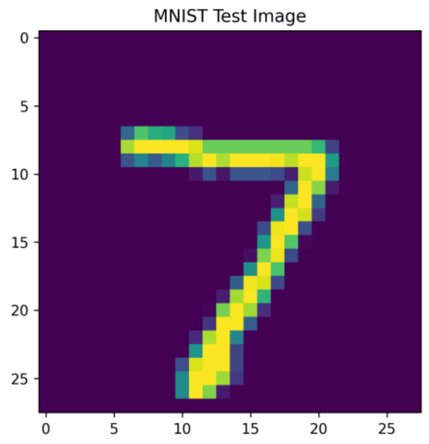
Task 1 (Desiree Forster):

To download and implement the training and test images from MNIST and Fashion MNIST, I simply used the 'idx2numpy' package as suggested in the report prompt description. I saved the files in a directory and set their paths to their respective image_file variable name. Once the paths were set, the images were imported using the 'idx2numpy' package to import them into NumPy arrays. The training and testing plots (same technique provided in the report prompt) are displayed below:

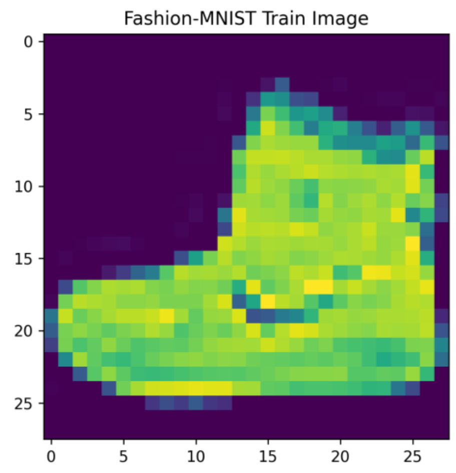
MNIST Train Image:



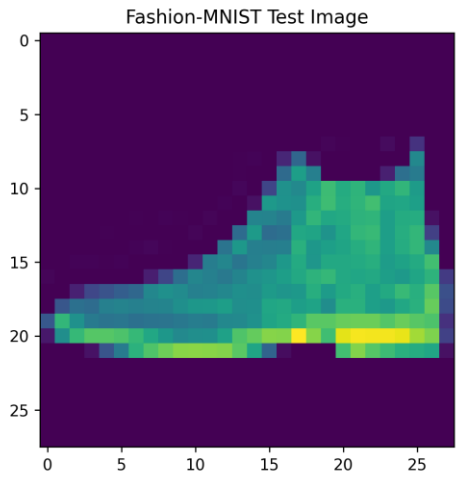
MNIST Test Image:



Fashion MNIST Train Image:



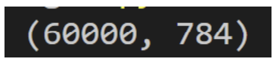
Fashion MNIST Test Image:



Task 2 (Desiree Forster):

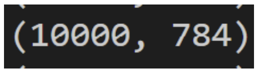
To perform a data transformation by flattening each image into a 1-D array using the NumPy function, reshape, I imported each training and testing images from task 1 and applied the function. To flatten these images, I had to essentially reshape each sample of image 28x28 from a 2-D array so that it instead creates a 1-D array of values for the given number of samples. After reshaping, the printed, flattened values from the terminal are shown below:

MNIST Train Shape:



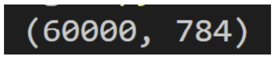
```
(60000, 784)
```

MNIST Test Shape:



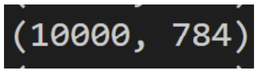
```
(10000, 784)
```

Fashion MNIST Train Shape:



```
(60000, 784)
```

Fashion MNIST Test Shape:



```
(10000, 784)
```

<u>Task 3 (Christopher Beau Gilbert and Patrick Jojola):</u>

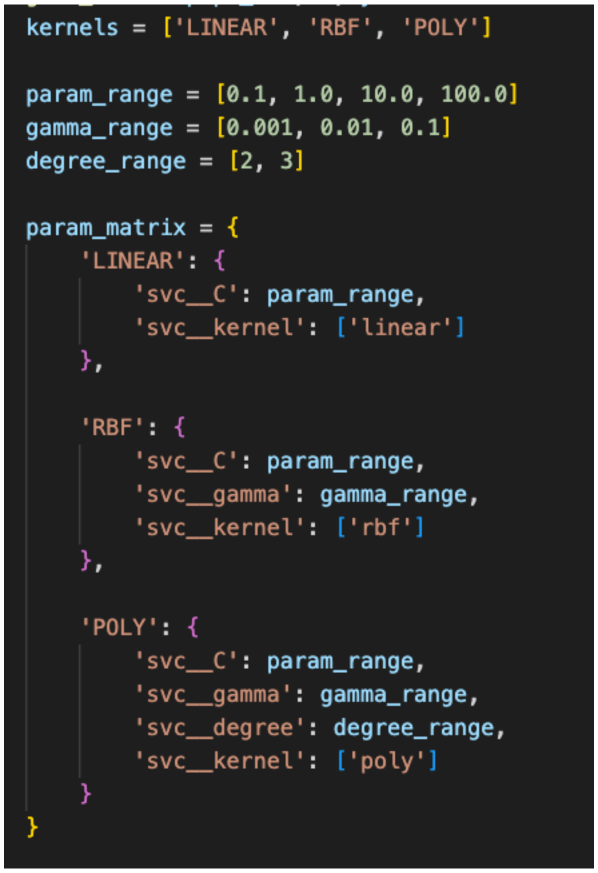
With the training set images downloaded and flattened, the dataset is then put through a sklearn pipeline. The pipeline includes a standard scalar for normalization, PCA object for dimensional reduction, and default SVC control group training. The training is then simply run through a for loop in order to test the dimensional reductions of 50, 100, and 200. An outer for loop also cycles from mnist dataset to the mnest fashion dataset.

There is no implementation of LDA for your assignment due to various limitations of the sklearn LDA functionality. The sklearn implementation of LDA does not permit a dimensionality reduction to a size greater than L –1, where L is the number of possible categorizations.

1. Why does sklearn do this? LDA as a form of dimensionality reduction attempts to maximize the variance between classes. Because of this, the dimensions left from the reduction are the ones which maximize the difference between classes, and select for maximum distance between a class and all others for as many classes as are available, i.e. L-1
2. I don't think it would be out of line to do some form of bagging aggregation where you allow for duplicate classifications. In this case you may want to take classified sets and randomly subdivide them into split categories, i.e. socks_a, socks_b. Doing so would raise the value of L for the LDA algorithm without meaningfully altering the dataset.
3. It may be possible to get around the sklearn restriction of L-1 by doing repeated dimensional reductions. Perform LDA on a dataset, remove those dimensions from the dataset then perform LDA again to create a ranked LDA where you get the dimensions with the most variance, the second most variance, and so forth until you have reasonable number of dimensions to train against.

The best settings for the MNIST dataset were in the poly kernel with dimension 100 and the following parameters (score = 0.95): C = 1.0, degree = 3, gamma = 0.1, and kernel = poly. For the Fashion-MNIST dataset, the best settings were in the rbf kernel with dimension 200 and the following parameters (score = 0.85): C = 10.0, gamma = 0.001, and kernel = rbf.

The following screenshot explains the search space for the hyperparameters:



```
kernels = ['LINEAR', 'RBF', 'POLY']

param_range = [0.1, 1.0, 10.0, 100.0]
gamma_range = [0.001, 0.01, 0.1]
degree_range = [2, 3]

param_matrix = {
    'LINEAR': {
        'svc__C': param_range,
        'svc__kernel': ['linear']
    },

    'RBF': {
        'svc__C': param_range,
        'svc__gamma': gamma_range,
        'svc__kernel': ['rbf']
    },

    'POLY': {
        'svc__C': param_range,
        'svc__gamma': gamma_range,
        'svc__degree': degree_range,
        'svc__kernel': ['poly']
    }
}
```

<u>Task 4 (Ahmaide Awawda):</u>

For this task both datasets (MNIST, and the Fasion-MNIST) were both evaluated by both the principal component analysis (PCA) and the Linear Discriminant Analysis (LDA) for the three kernels (Linear, RBF, and POLY), where the dimensions for PCA were as given (50, 100, and 200), yet for the LDA as the sklearn library can't reduce the dimensionality more than the number of classifications and in order to have three dimensions, the selected dimensions were (2, 5, and 9).

At first, the PCA approach was implemented, where there were 18 results, 9 per each dataset, the confusion metrices for the MINST dataset by the PCA approach for the three given dimensions (50, 100, 200) are as shown in the figures below:

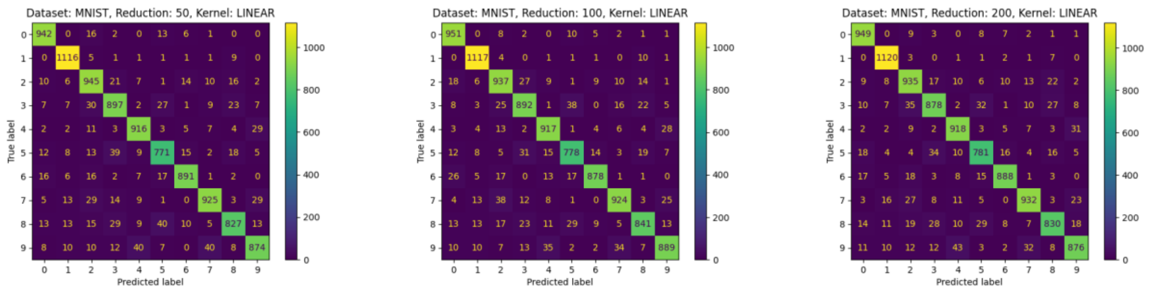


Figure 1: MNSET Dataset with PCA Approach in Linear Kernal

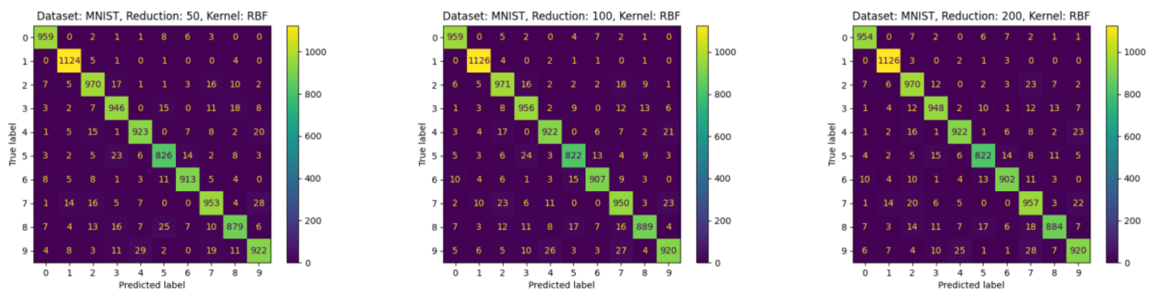


Figure 2: MNSET Dataset with PCA Approach in RBF Kernal

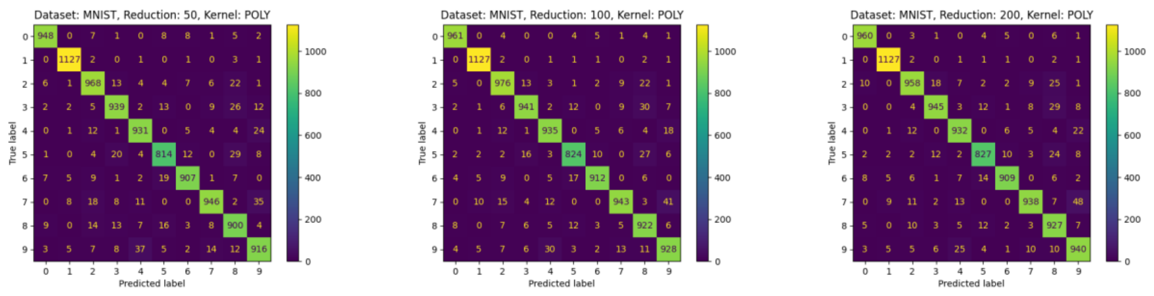


Figure 3: MNSET Dataset with PCA Approach in POLY Kernal

As shown in the previous images, for the first dataset the system performed better for lower dimensions (in general) for all kernels, the RBF and POLY had better performance than the linear kernel, where the RBF performed better in classifying classes with sharper edges like “4” and “4” while the POLY performed better in the more circular image classes like “8”.

Now the PCA approach in implemented the same way for the Fasion-MNSET dataset as the 9 confusion metrices are shown in the figure below:

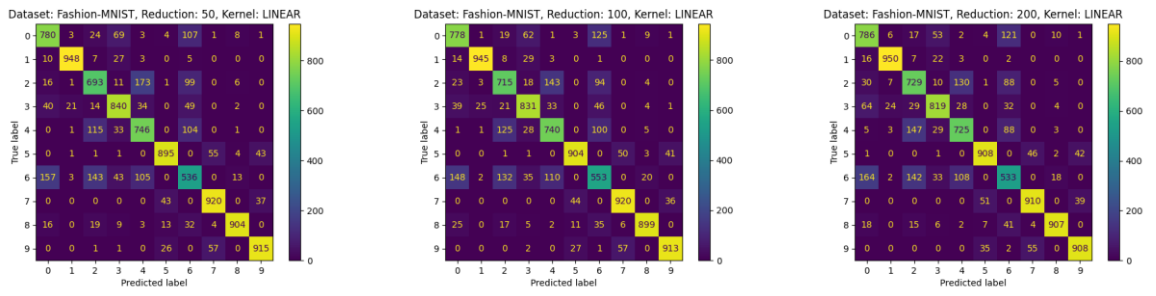


Figure 4: Fasion-MNSET Dataset with PCA Approach in Linear Kernal

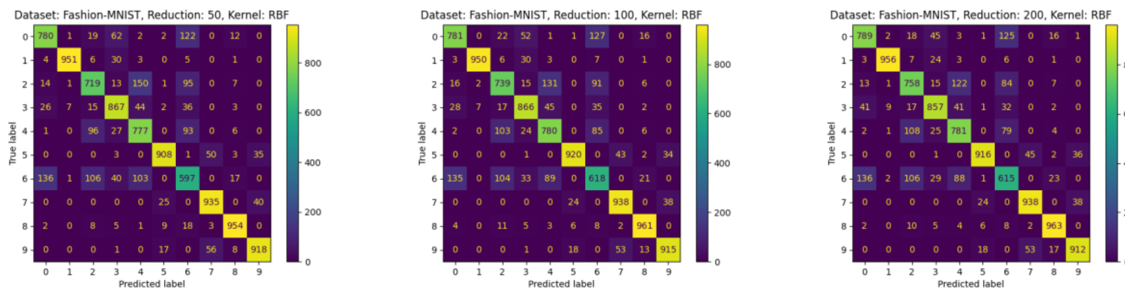


Figure 5: Fasion-MNSET Dataset with PCA Approach in RBF Kernal

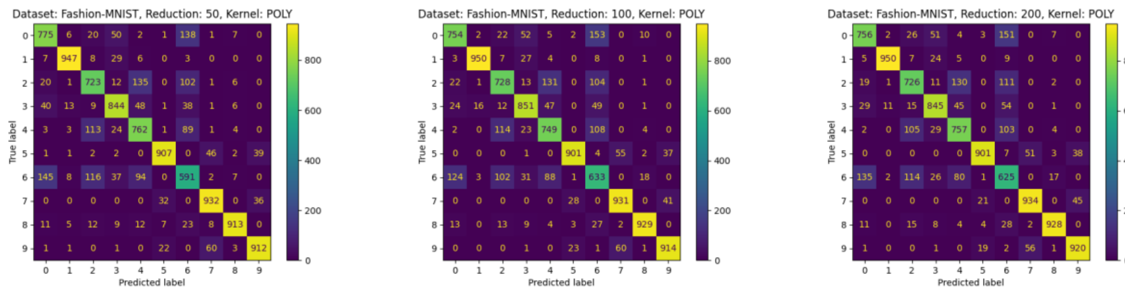


Figure 6: Fasion-MNSET Dataset with PCA Approach in POLY Kernal

The system here showed lower accuracy predictions than the earlier dataset, due to the complexity of the labels in the Fasion-MNSET dataset.

For this dataset the best overall kernel was the RBF followed by the POLY kernel by a slightly difference in the performance evaluation, for this dataset it was noticed that in general the evaluation becomes more accurate with the increase of the number of dimensions.

Now for the LDA analysis it was mentioned that number of dimensions can’t be higher or equal to the number of labels (10 for each dataset). In order to include 36 results as required three dimensions were added for this analysis which were (2, 5, and 9).

The LDA analysis was implemented for the MNSET dataset where the confusion metrics as follows:

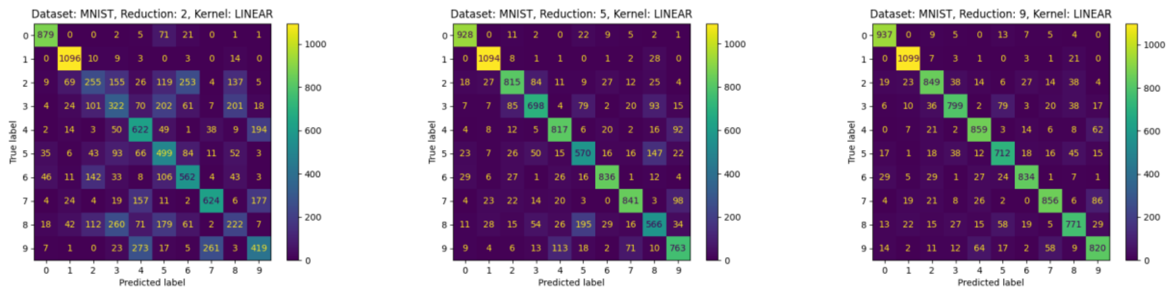


Figure 7: MNSET Dataset with LDA Approach in Linear Kernal

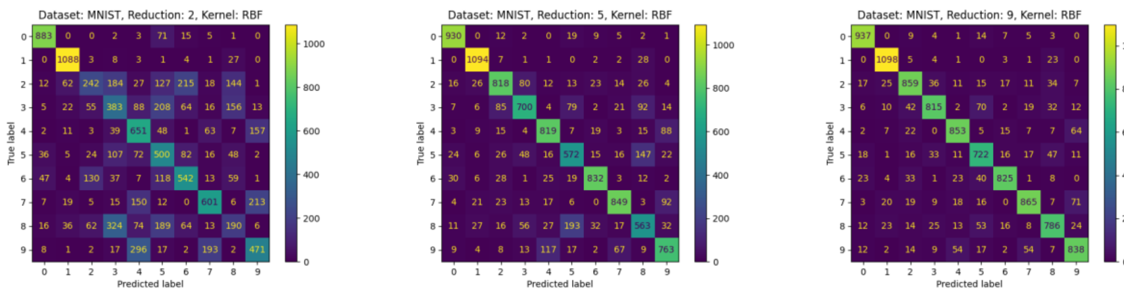


Figure 8: MNSET Dataset with LDA Approach in RBF Kernal

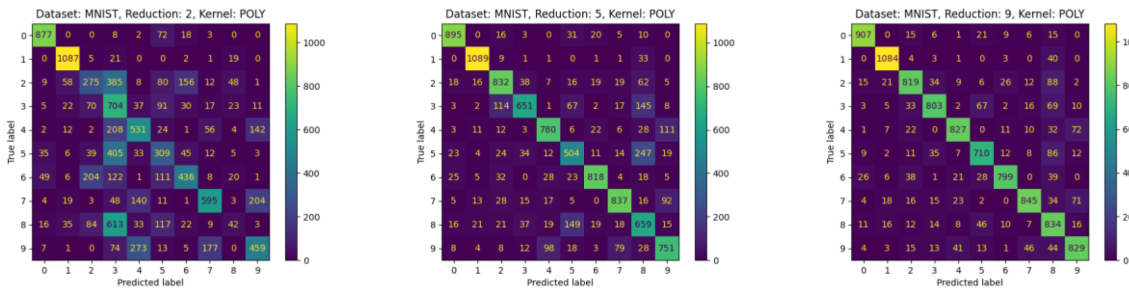


Figure 9: MNSET Dataset with LDA Approach in POLY Kernal

As shown in the previous images the approach gives terrible results when it's only 2 dimensions, it is improved for 5 dimensions, but it performed its best at 9 dimensionality reduction for all kernels. It is noticed that all kernels are pretty much preforming at almost the same level yet with different classifications.

Now finally the LDA approach is being implemented on the Fasion-MNSET as shown below.

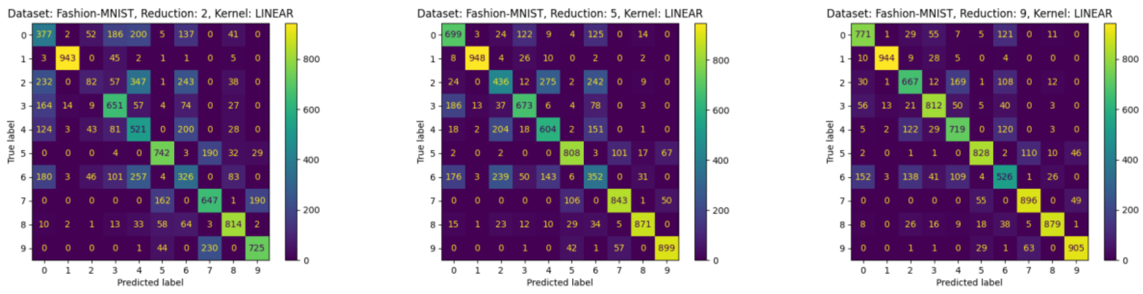


Figure 10: Fashion-MNSET Dataset with LDA Approach in Linear Kernal

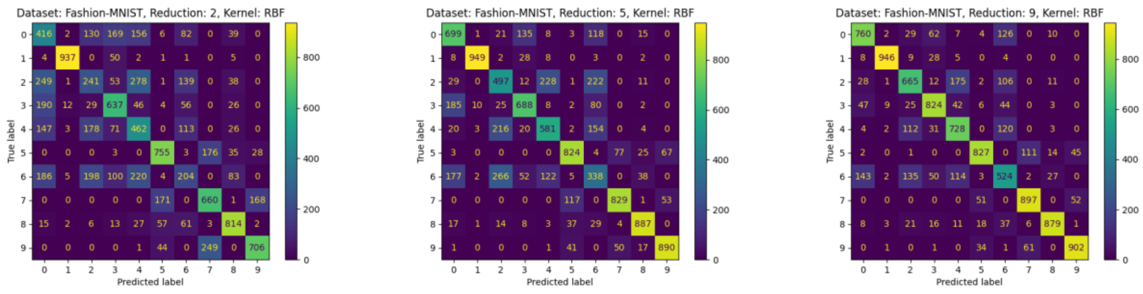


Figure 11: Fashion-MNSET Dataset with LDA Approach in RBF Kernal

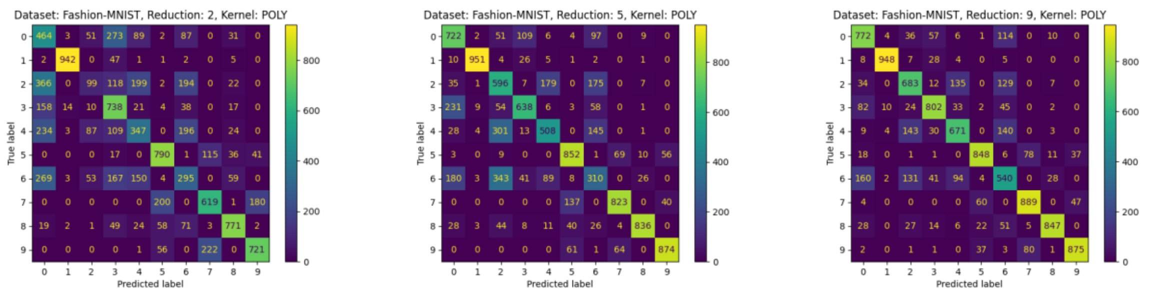


Figure 12: Fashion-MNSET Dataset with LDA Approach in POLY Kernal

Just as the PCA analysis the Fasion-MNSET has less good performance than the MNSET dataset, and again the evaluation performance increases with the increase of the kernels where it performs terrible for both 2 and 5 kernels.

And just like the MNSET dataset this approach has almost a near performance for each kernel with different metrics for each label.

Now the accuracy for each run was also evaluated as for the MNSET dataset it is noted in the table below:

Table 1: MNSET Dataset Accuracy

Dimension	LINEAR	RBF	POLY
PCA 50	0.91	0.94	0.94
PCA 100	0.91	0.94	0.95
PCA 200	0.91	0.94	0.95
LDA 2	0.55	0.55	0.53
LDA 5	0.79	0.79	0.78
LDA 9	0.85	0.86	0.85

Table 2: Fasion-MNSET Dataset Accuracy

Dimension	LINEAR	RBF	POLY
PCA 50	0.82	0.84	0.83
PCA 100	0.82	0.85	0.83
PCA 200	0.82	0.85	0.83
LDA 2	0.58	0.58	0.58
LDA 5	0.71	0.72	0.71
LDA 9	0.79	0.8	0.79

Observations:

- The first MNSET dataset gave way better results in general since the labels were easier to identify for both analyses.

- If we wanted to compare the PCA with the 9-dimension reduction of LDA. The PCA outperformed the LDA when it came to the performance since its implementation had better comparison between the labels.

- The PCA analysis performance saw some differences between the kernels (mainly the LINEAR kernel with the two others) where the RBF and POLY kernels had almost the same performance yet for different values, but the linear one was lower.

- The RBF had better identification when it came to images with sharper edges like “4” and “7” while the POLY kernel had better identification for the circular shapes like “8”.

- The three different dimensionality reductions for the PCA implementations weren’t really effective although there were a few different classifications.

- For the LDA analysis didn’t have a huge performance improvement with the changes in the kernels for all its different dimensions, yet there was a slightly different labeling for each kernel.

- When it came to the dimensionality reduction in the LDA analysis, it was the opposite of the PCA where the number of dimensions played a significant role in the system evaluation.