COMPUTER VISION (CAP5415)-PA-3 REPORT

AUTOENCODER & NEAREST NEIGHBOR CLASSIFICATION

Answer 1)

a) AutoEncoder Using Fully connected layers:

1) The encoder architecture includes two layers with 256 and 128 neurons, both followed by ReLU activation functions.
2) ReLU activation is used in both layers of the decoder, which have 256 and 784 neurons, respectively. The compression level is also determined by an intermediate linear layer comprising 32 neurons.
3) Mean Squared Error (MSE) loss is used for training, and the Adam optimizer is used for optimization. A learning rate of 0.001 is used over a period of 10 epochs.

b) AutoEncoder Using Convolution layers:

1) Two convolutional layers, one stride, one padding, and a 3x3 kernel make up the encoder. ReLU activation is applied after every convolutional layer, and it is followed by max pooling with a 2x2 kernel and 2 stride.
2) Three transposed convolutional layers are used by the decoder. Using a 2x2 kernel and 2 strides, the first two layers unsampled, while the final layer uses a 1x1 kernel and 1 stride and does not.
3) Mean Squared Error (MSE) loss is used for training, and the Adam optimizer is used for optimization. A learning rate of 0.001 is applied over a total of 10 epochs.

Results of autoencoders:

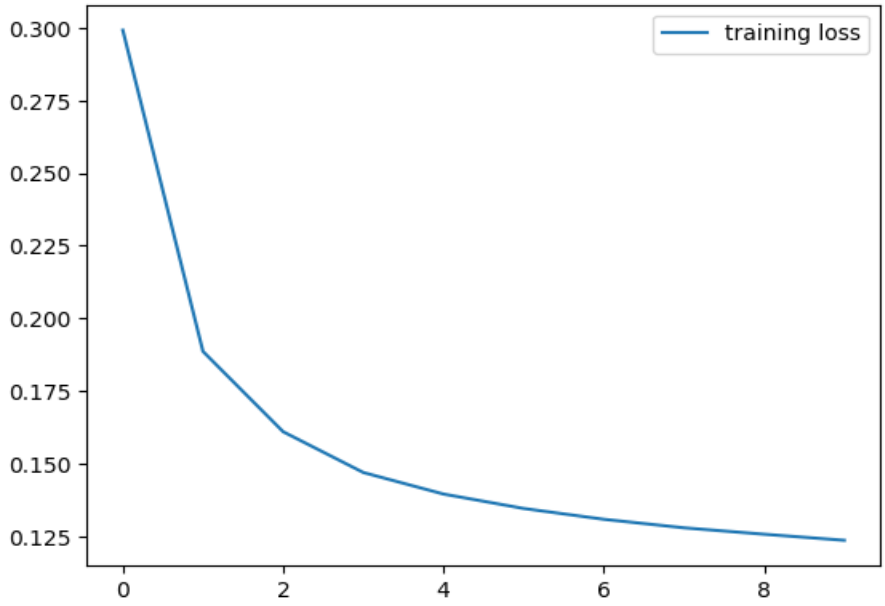


Fig a

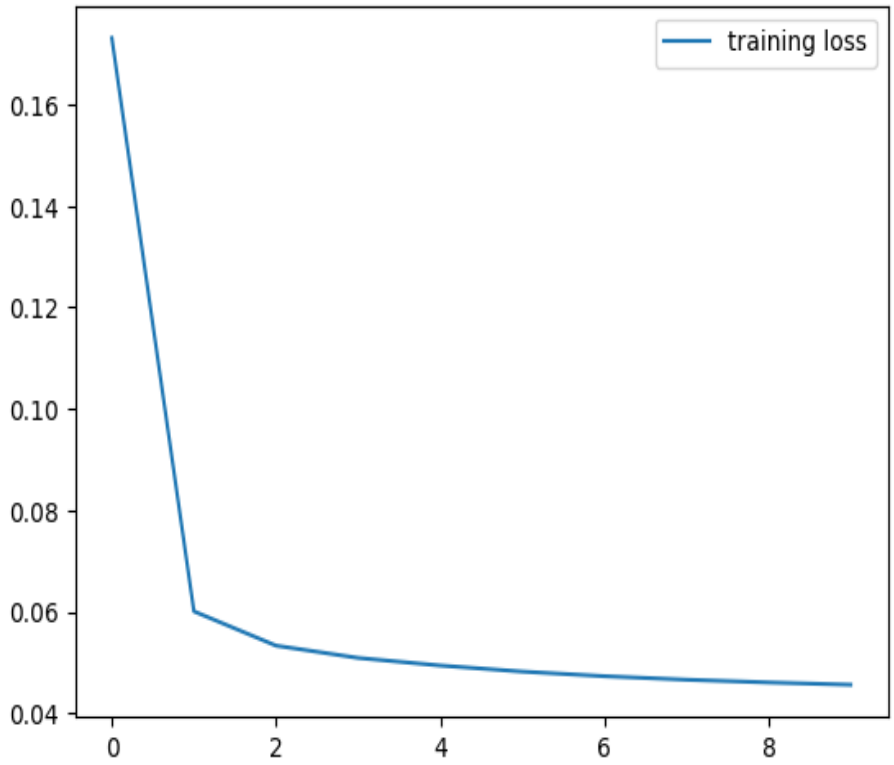
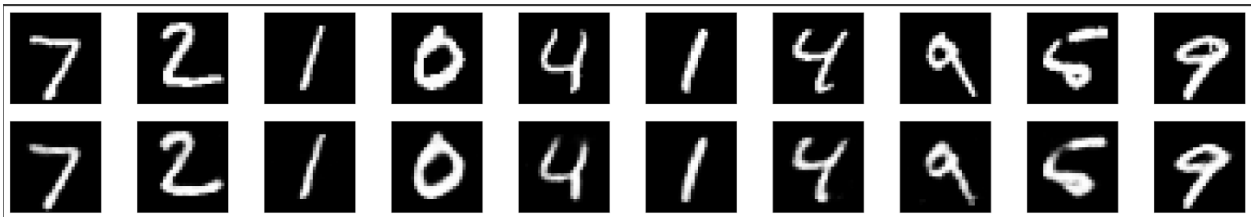


Fig B

The second variation demonstrates a quicker convergence compared to the first, attributed to the adoption of convolutional layers over fully connected layers. However, the ultimate loss values reached by both variations are almost indistinguishable despite their divergent convergence speeds.

The pictures shown show both the original and the rebuilt versions. The reconstructed images in the first variation exhibit some blurriness, particularly in the vicinity of sharp edges. On the other hand, there aren't many noticeable differences between the reconstructed images in the second variation and the originals.



AutoEncoder using Fully Connected Layers

AutoEncoder using Convolution Layers

Answer 2)

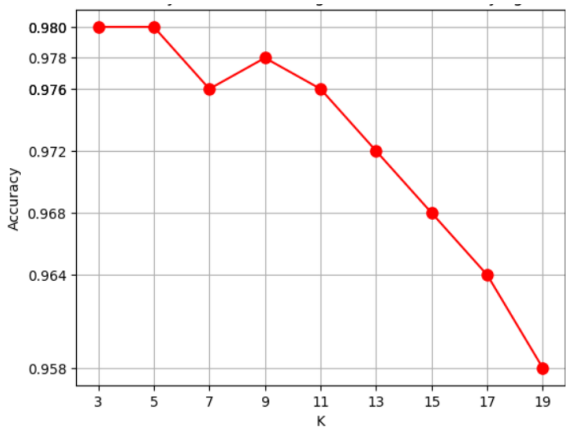
Nearest Neighbor Classification

• Two nearest neighbor classifiers, trained on the Scikit library's digits dataset, are utilized for digit recognition, with a training set of 1297 images and a test set of 500 images.

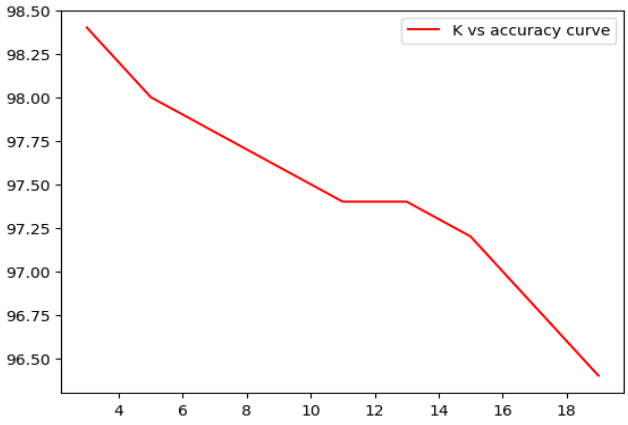
Nearest Neighbor Classifier:
Here, each test image is matched against all training images to determine the image with the smallest L2 or Euclidean distance, and the corresponding label is predicted.
Result:
With L1 distance, the nearest neighbor classifier showed 98% accuracy, and with L2 distance, 98% accuracy.

K-Nearest Neighbor Classifier:
Using this method, the top K images with the lowest L2 or Euclidean distance are determined by carefully comparing each test image to all training photos. The label that appears the most frequently among those chosen K neighbors determines the prediction.
Result:
we have tested the KNN classifier with the different k values. I've plotted the the accuracy vs 'k' value cures for the K-nearest neighbor classifier.

Final Result:



L1 Distance



L2 Distance

As the K value rises in the accuracy curve, it becomes apparent that the classifier's accuracy diminishes, suggesting a trend towards overfitting in such scenarios.
Even said, there is not much overfitting as the accuracy ratings for both distance measurements are still very high. K = 7 appears to be the best hyperparameter selection for this dataset, yielding accuracy of 97.8% for L1 distance and 98% for L2 distance.