**Group Assignment 2: Deep Learning**

**Task 1**

The MNIST dataset, consisting of 784-dimensional handwritten digit images, was analysed using Principal Component Analysis (PCA) to reduce dimensionality and explore digit separability. PCA is effective for visualising high-dimensional data as it reduces the original 784 dimensions into a manageable 2D space, capturing the directions of highest variance.

Mathematically, PCA involves computing the covariance matrix of the centered data X:A black background with white text

Description automatically generated

where **X** is an n×784 data matrix after mean-centering (each feature has zero mean). PCA then applies eigendecomposition to this covariance matrix, to obtain principal components:

Below are the plots from this PCA on the MNIST data set. The left shows the raw PCA plotted for 10,000 random samples, the left shows the same 10,000 samples but with centroids and 95% confidence ellipse outlines.

A diagram of a colorful explosion

Description automatically generated with medium confidenceA diagram of multiple colored lines

Description automatically generated

Below is a histogram of the data from a LinearSVC that we implemented to assess the separability of MNIST digit pairs in 2D PCA space. It trains the classifier with 10,000 iterations for each pair, calculates their separation accuracy, showing a mean accuracy of 0.841. A graph of a number of pairs

Description automatically generated with medium confidence

**Observations from the Plots and Data**

The histogram illustrates the classification accuracy of digit pairs using an SVM classifier on 2D PCA projected data, with a mean accuracy of 0.841. Pairs like (0, 1), (0, 7), and (1, 4) achieve accuracies near 1.0, indicating strong separability due to distinct visual features, such as 0s circular shape versus 1s vertical line. Conversely, pairs like (3, 5), (3, 8), and (7, 9) have lower accuracies, suggesting similarities, like the curves of 3 and 8, make them harder to distinguish.

A black and white screen with numbers

Description automatically generated

The table of PCA results shows digit 1s centroid at (6.71, 3.38) with low PC1 variance (1.32), indicating a tight cluster, while digit 0s centroid at (7.77, 1.75) with higher PC1 variance (13.70) suggests a broader spread. Digits 4, 7, and 9 have similar PC2 values (4.37 to 4.63), hinting at overlap, unlike the distant digits 1 and 0.

The scatter plot visualises these clusters: digit 1 (orange) clusters left (PC1: 5 to 0), digit 0 (blue) right (PC1: 10 to 15), both distinct. Digit 4 (purple) is bottom right, while digits 3 (red), 5 (brown), and 8 (yellow) overlap centrally. The second scatter plot, with centroids and 95% confidence ellipses, confirms digit 1s isolation, digit 7s partial distinction, and dense overlap among digits 4, 5, 9, and others (0, 2, 3, 6, 8) on the right.

**Classes That Can Be Linearly Separated**

Digits 1 and 0 are linearly separable in the 2D PCA space, as their clusters are distinctly positioned with minimal overlap, supported by close to 1 SVM accuracies. Digit 4 shows partial separability, particularly from digits like 1 and 0, but overlaps with others like 5 and 9. Digits 3, 5, and 8, with significant overlap in both scatter plots and lower accuracies (around 0.6), cannot be linearly separated here, nor can pairs like 7 or 9 due to proximity and shared features.

PCA effectively visualises the MNIST dataset by reducing dimensionality and revealing clustering patterns. While digits 1 and 0 are linearly separable, the overlap among digits like 3, 5, and 8 underscores PCA’s limitations in fully separating all classes in 2D, suggesting that higher dimensions or non linear methods might enhance separability for these challenging digits.

* <https://builtin.com/data-science/step-step-explanation-principal-component-analysis>
* <https://www.datacamp.com/tutorial/svm-classification-scikit-learn-python>

**Task 2**

* Demonstrate that your algorithm converges to a good local minima.
* Plot the training error curve vs the number of iterations.
* Show what feature has been learned and discuss why? (Demonstrate as an image with the same size as inputs).
* Repeat this training/testing procedure to classify different pairs. Re-port the accuracies of 5 pairs in a table and discuss why some are easier to classify than others.

**Task 3**

* Report the classification accuracy obtained after training is completed on both train and test data.
* Further using visualisation tools (e.g., Tensorboard is one option) plot the training and testing curves, that are the training and testing data’s classification accuracy rates vs the iterations/epochs of the training algorithm (ADAM). Include a figure of these curves in your report and discuss your observations about the learning curve behaviours.
* Discuss how the number of layers/parameters affects the classification accuracy.
* Train four different networks with more hidden layers, for example, 3, 4, 5 and 7 hidden layers (the choice is yours here to make a good conclusion). Choose an appropriate width, i.e., number of neurons per layer, to achieve good accuracy and feasible training time.
  + How do they compare to the former MLP you implemented?
  + How many parameters (weights/biases) do these models have?
* Compare the classification accuracies of these networks with the previous MLP.
* By plotting a graph or in a table, show the accuracy vs depth vs complexity (number of parameters) of all five trained MLPs with different depths/widths.
* Discuss the results, i.e., how the number of layers/parameters affect the classification accuracy, and provide a conclusion.

**Task 4**

* Report the classification accuracy obtained after training was completed on both train and test data.
* Further using visualisation tools (e.g. Tensorboard is one option) plot the training and testing curves, that are the training and testing data’s classification accuracy rates vs the iterations/epochs of the learning algorithm (ADAM). Include a figure of these curves in your report and discuss your observations about the learning curves’ behaviours by plotting a graph or a table, show the accuracy vs depth vs complexity (number of parameters) of the CNNs.
* For this part, you need to train four additional CNNs of different depths and widths (again,  your choice) and report the results in a table. Discuss the results and provide a conclusion.
* In addition, discuss and analyse the differences in terms of performance, number of model parameters (i.e., weights/biases) and training/testing times between CNNs and MLPs. Provide a conclusion.
* For these discussions, you should compare your results in Table 4.2

**Task 5**

* Once your CNN (in Question 4) is trained, access its filters via, e.g.‘tf.get\_collection’ and plot them on a grid for each layer.
* What pat-terns do you observe, and why?
* In addition, plot the activations of each layer (i.e., the output of each conv layer in your CNN) for two images chosen from digit-classes ‘2’ and ‘9’. Discuss your observations.
* Create deep dream images for digit classes 2 and 9 and show these in your report. Discuss how each deep dream image shows the model’s sensitivity to the patterns that appear in the input images relative to each output class (category).

**Task 6**

* Comment on your results and explain what makes the cases of and λ=1 particularly special? λ=0
* Compare the performance of MTL models to single-task networks. Discuss important considerations when using MTL and its pros and cons