Devanagari Digits Classification – Supervised and Unsupervised Techniques

Akash Mittal

Università degli Studi di Milano Statale - UNIMI

B79: Data Science for Economics

Matricola ID : 31909A

Table of Contents

Abstract

This project investigates the application of both supervised and unsupervised learning techniques on the Devanagari handwritten digit dataset. Devanagari is an Indic script used in the northern Indian subcontinent and it is written left-to-right, based on the ancient Brāhmi script. I primarily used the Convolutional Neural Networks (CNN), to classify the digits and compare the performance of simple CNN with pretrained RestNet50 model for same parameters. For unsupervised learning, dimensionality reduction with Principal Component Analysis (PCA) was followed by K-Means clustering to group the digits and evaluate the performance. The combination of PCA-based dimensionality reduction and clustering revealed meaningful patterns in the data, demonstrating the effectiveness of hybrid approaches in digit recognition tasks. But since the image data is high-dimensional, I used T-SNE and UMAP to further visualize and cluster the data which came out to be more prominent than PCA, as also suggested by different studies for high dimensional data.

Keywords: *Devaganari, CNN, RestNet50, PCA, K-Means, T-SNE, UMAP.*

Goal of the Study

The primary objective of this study is to leverage advanced machine learning techniques to accurately classify and distinguish handwritten Devanagari digits. This task is particularly meaningful to me, as I have personally experienced the challenges of differentiating between certain symbols in my script, such as the similar appearances of 1 and 9, or 3 and 6, which can easily be confused due to their flipped orientations. By employing convolutional neural networks (CNNs), I aim to harness their feature extraction capabilities and recognize patterns in the data that might be overlooked by traditional methods. Additionally, I have incorporated unsupervised learning approaches, such as K-means clustering and principal component analysis (PCA), to explore the underlying structures within the digit dataset, allowing me to better understand the relationships between the different classes of handwritten digits.

Dataset

I have used a Hindi/ Sanskrit Handwritten Characters dataset by "*Shailesh Acharya"* and "*Prashnna Kumar Gyawali*", uploaded to the UCI Machine Learning Repository.

Data Type: GrayScale Image

Image Format: PNG

Resolution: 32 by 32 pixels

Actual character is centered within 28 by 28 pixel, padding of 2 pixel is added on all four sides of actual character.

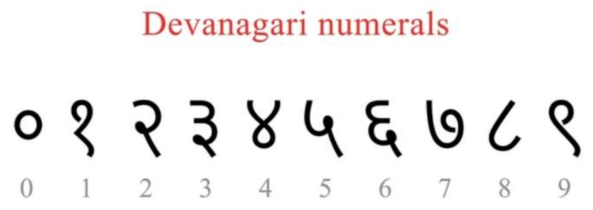
There are **~1700** images per class in the **Training** set, and around **~300** images per class in the **Test** set.

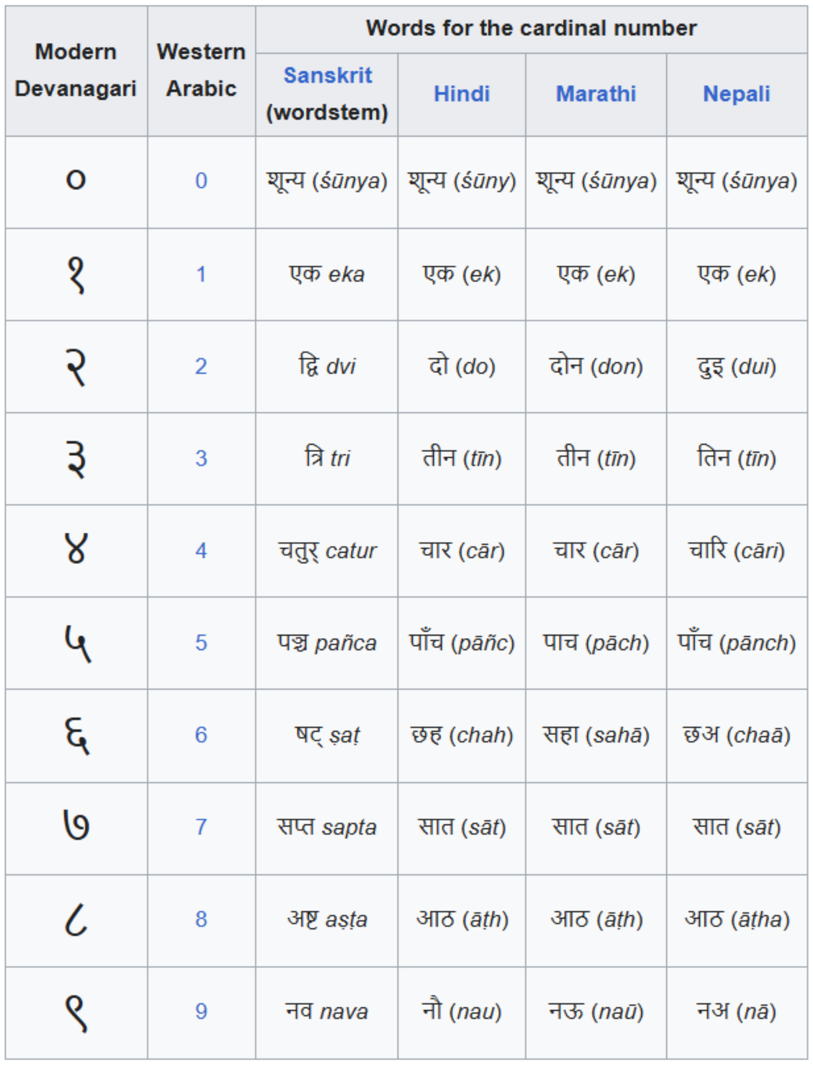
Source (1): *https://archive.ics.uci.edu/ml/datasets/Devanagari+Handwritten+Character+Dataset*

Source (2): *https://www.kaggle.com/datasets/anuragss397/hindi-mnist-data*

Devanagari Numerals[1][2]

The images below show the Devanagari digits/ numerals in their standard form. We can see that one can easily get confused between writing a 1 or 9 or get confused between remembering the orientation of 3 or 6. Further, there are cases in handwritten digits where it becomes difficult to distinguish between a 0 and 8, as in running hand, a 0 is not always perfectly written as a closed shape.



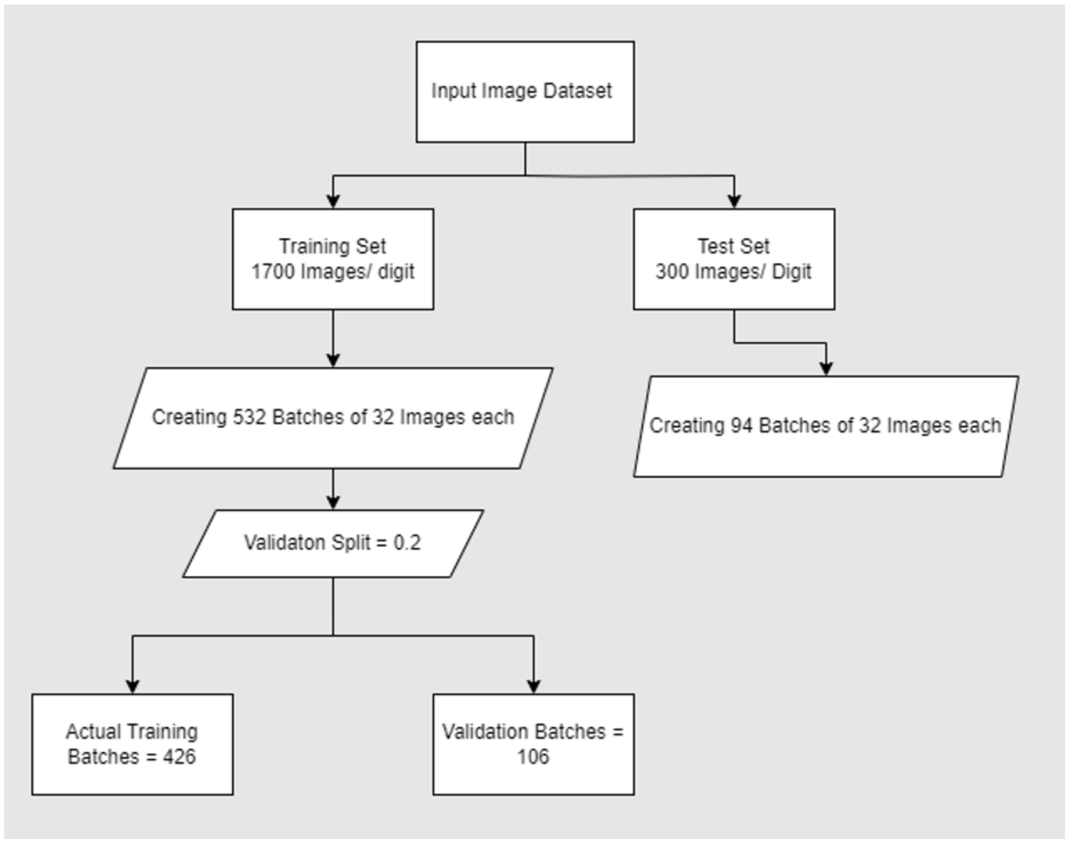


Modern Devanagari	Western Arabic	Words for the cardinal number			
		Sanskrit (wordstem)	Hindi	Marathi	Nepali
०	0	शून्य (śūnya)	शून्य (śūny)	शून्य (śūnya)	शून्य (śūnya)
१	1	एक *eka*	एक (*ek*)	एक (*ek*)	एक (*ek*)
२	2	द्वि *dvi*	दो (*do*)	दोन (*don*)	दुइ (*dui*)
३	3	त्रि *tri*	तीन (*tīn*)	तीन (*tīn*)	तिन (*tīn*)
४	4	चतुर् *catur*	चार (*cār*)	चार (*cār*)	चारि (*cāri*)
५	5	पञ्च *pañca*	पाँच (*pāñc*)	पाच (*pāch*)	पाँच (*pānch*)
६	6	षट् *ṣaṭ*	छह (*chah*)	सहा (*sahā*)	छअ (*chaā*)
७	7	सप्त *sapta*	सात (*sāt*)	सात (*sāt*)	सात (*sāt*)
८	8	अष्ट *aṣṭa*	आठ (*āṭh*)	आठ (*āṭh*)	आठ (*āṭha*)
९	9	नव *nava*	नौ (*nau*)	नऊ (*naū*)	नअ (*nā*)

[1] Image Source (1): https://www.shutterstock.com/image-vector/devanagari-numerals-0-9-their-600nw-2513455439.jpg

[2] Image Source (2): https://en.wikipedia.org/wiki/Devanagari_numerals

Data Loading and Image Visualization


Input Image Dataset
Training Set 1700 Images/ digit
Test Set 300 Images/ Digit
Creating 532 Batches of 32 Images each
Creating 94 Batches of 32 Images each
Validaton Split = 0.2
Actual Training Batches = 426
Validation Batches = 106

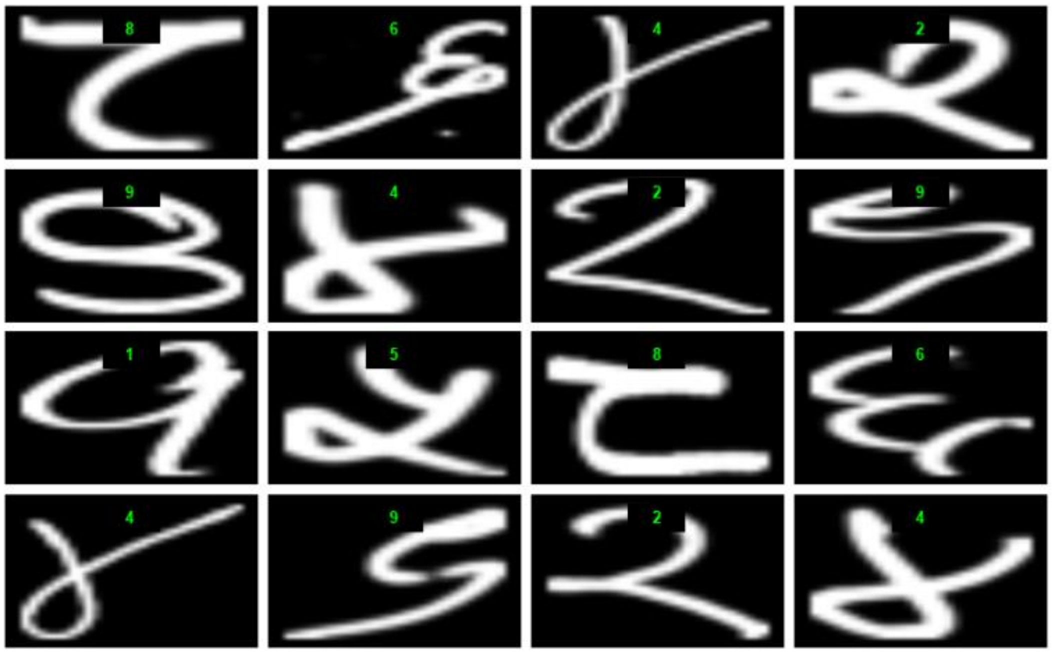


Figure. The image shows 16 random digits chosen from the training set and displayed with their labels.

Supervised Learning Approach

Analysis Using CNN and RestNet50:

1. Loading the training and test datasets
2. Pre-processing and cross validating the loaded dataset for consistency and extracting the labels from the dataset.
3. Normalizing the images before training.
4. Visualizing the images and cross validating the extracted labels.
5. Building a Convolutional Neural Network (CNN) Model, and training on the training dataset. (The image features, e.g. random rotation, random zoom, are randomized to make the model robust. Further, during the training steps, a random dropout of images is used for more robust training of the model).
6. Using RestNet50 with same model features (epochs) and layers for training and comparing the results with the simplified CNN algorithm.

Note: Both the models were training for different learning rates, and a learning rate of 0.001 was chosen to compare the performances of both.

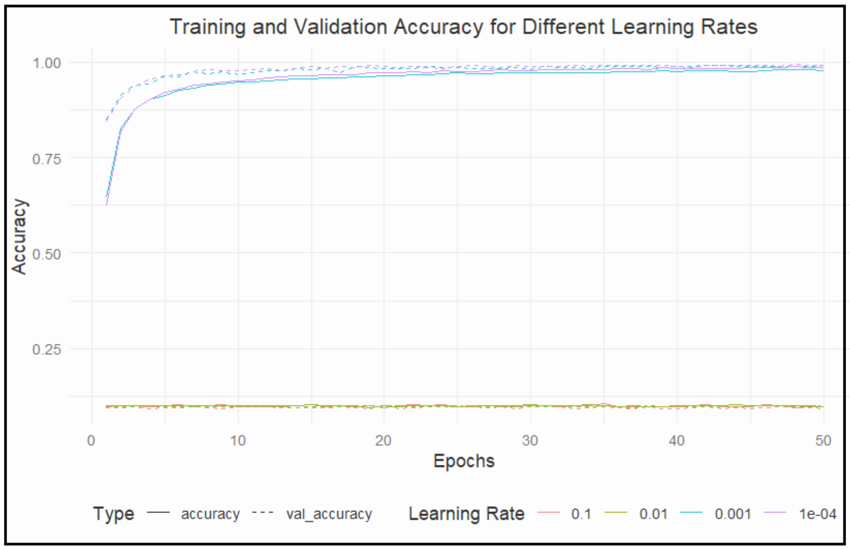
Unsupervised Learning Approach

Analysis using Principal Component Analysis (PCA) and K-Means

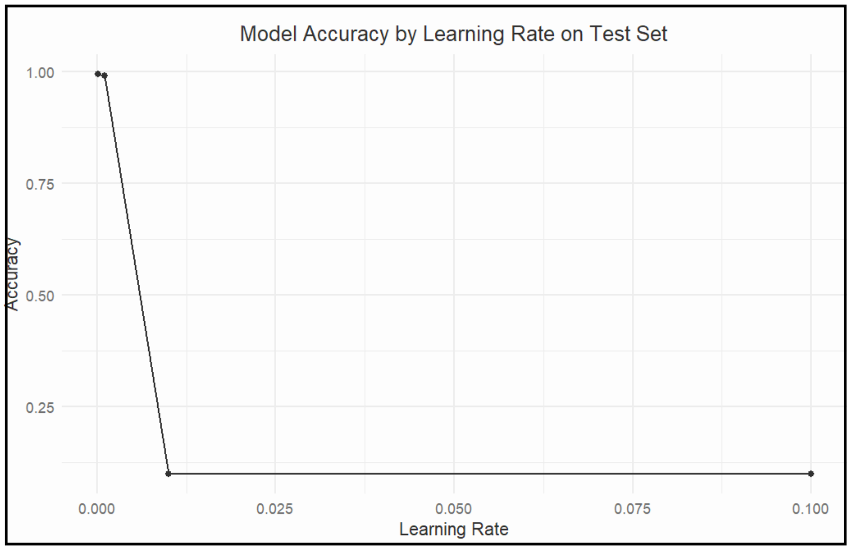
1. Loading the Training and Test Datasets.
2. Normalizing the pixel values of the images to a range of [0, 1] for better performance in PCA, and flattening the images from 2D image arrays to 1D vectors for the PCA.
3. Before feeding the images to PCA, identified and removed the all-zero columns.
4. Performing PCA (on the centred and scaled matric dataset) to reduce the dimensionality of data from **784 to 62**, explaining **~85%** of the variance in the dataset.
5. Visualization of the PCA results and then performing K-Means Clustering on the selected PCA components.
6. In K-means, I set the number of clusters to 10 for the 10 digits and while performing K-Means I made sure that I randomize the algorithm to start with multiple random points and with enough iterations for convergence.
7. Creating a contingency table for analysing how the actual digit labels are distributed across the K-Means clusters. Also plotting the percentage of each digit present in each cluster showing insights about misclassifications.

Key Findings

Performance of CNN with different Learning Rates

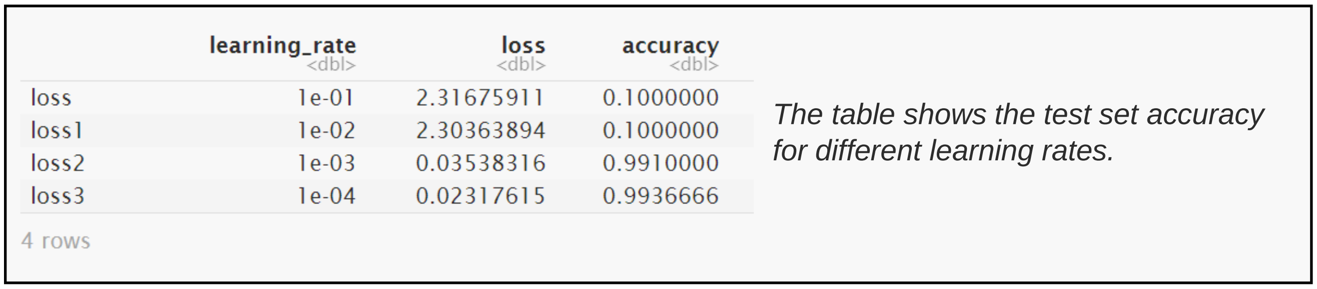


The image shows the training and validation accuracy of the CNN model for different learning rates. The models with a lower learning rate are more accurate than with higher learning rate for the same number of epochs.



The image shows the model accuracy for different learning rates. It shows that accuracy drops for higher learning rates. I choose 4 different learning rates for the algorithm, 0.0001, 0.001, 0.01, and 0.1.

Finally, as it is always a tradeoff between the learning rates, number of epoch and training time. I chose the learning rate of 0.001 and for that model, the accuracy on the test data is **99.1%.**



	learning_rate <dbl>	loss <dbl>	accuracy <dbl>
loss	1e-01	2.31675911	0.1000000
loss1	1e-02	2.30363894	0.1000000
loss2	1e-03	0.03538316	0.9910000
loss3	1e-04	0.02317615	0.9936666

4 rows

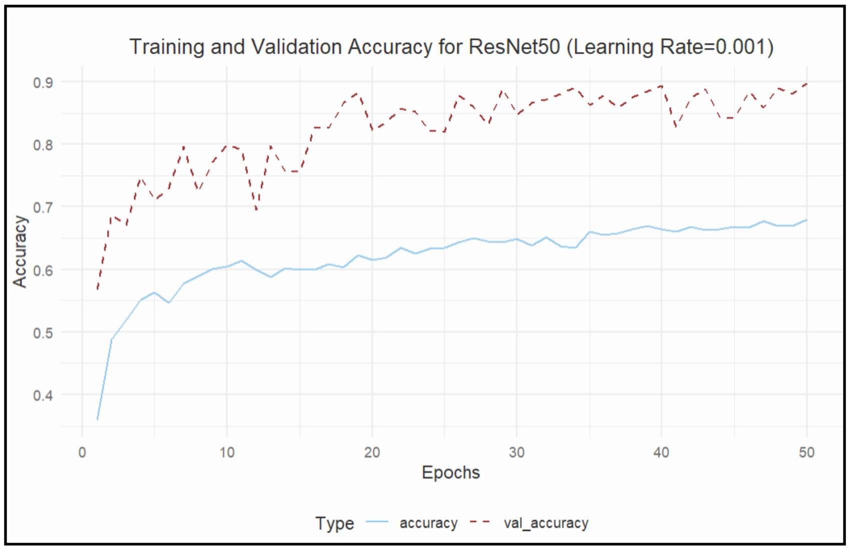
The table shows the test set accuracy for different learning rates.

Performance of RestNet50 (comparable to Simple CNN)

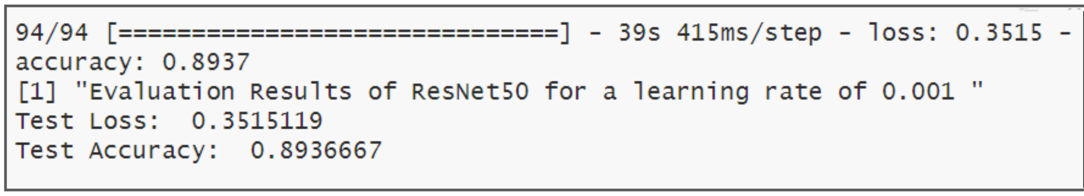
- Accuracy of RestNet50 compared to simple CNN approach

Training the pre-trained ResNet50 model on the digit classification peaked a training accuracy of **68%** with validation accuracy of **85.5%.** But at the same time the test set accuracy came to be **89.36%** for the same number of epochs, and a learning rate of **0.001**.

This comparison between the two models reiterates the known concept in Machine Learning that no one model fits all. In scenarios when a simple CNN model is working better than a highly pretrained ResNet50 model.



The image shows the training and validation accuracy of the ResNet50 model for learning rate 0.001 for the same number of epochs as for CNN.

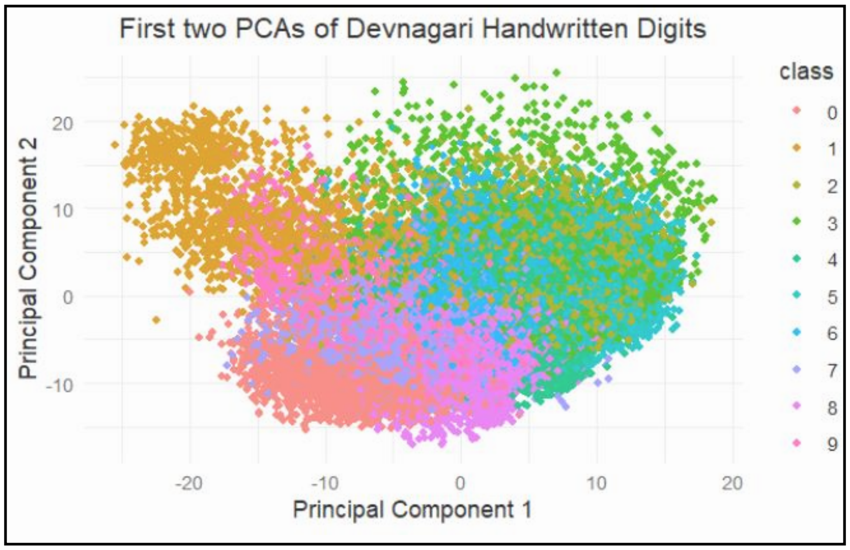


```
94/94 [==============================] - 39s 415ms/step - loss: 0.3515 -
accuracy: 0.8937
[1] "Evaluation Results of ResNet50 for a learning rate of 0.001 "
Test Loss:   0.3515119
Test Accuracy:   0.8936667
```

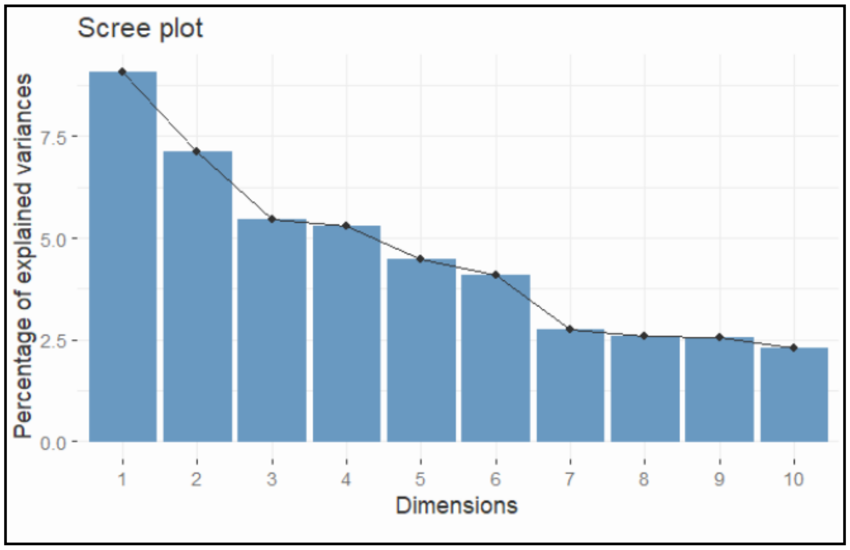
Test set Accuracy of the ResNet50 Model (with learning rate of 0.001)

Performance of PCA + K-Means (unsupervised learning approach)

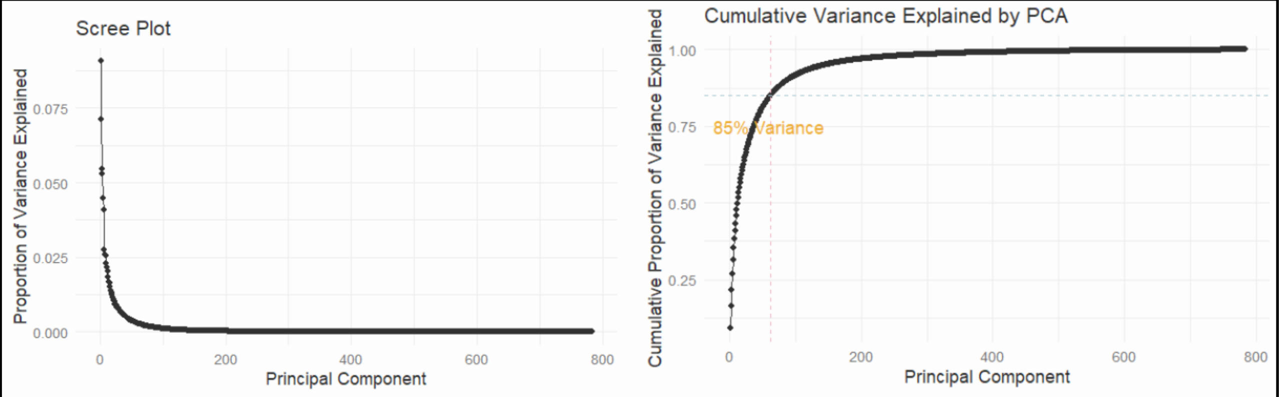
1. Loading the image dataset and checking the dimensions of the images.
   a. Dimension of Image Set (17000, 1024).
2. Preparing it for PCA, by removing all zero columns.
   a. Dimensions of Image set -> (17000, 784).



Performing and visualizing the PCA analysis. Plotting the first two Principal Components as below.



Scree-Plot of first 10 Principal Components with bars showing the percentage of variance explained by each.

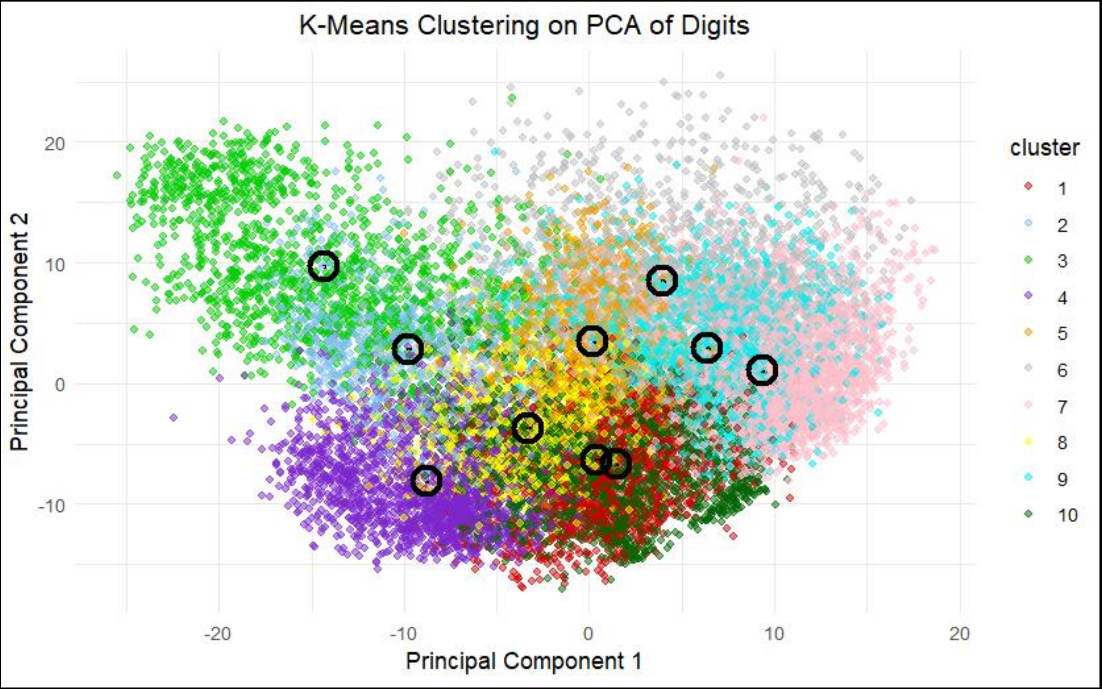


The Scree-Plot and Cumulative Variance Explained by the 784 principal components for the image dataset.

The plot for cumulative variance also marks the point where the principal components explain at least 85% variance in the image dataset.

K-Means Clusters

<u>Visualization Using Principal Components</u>



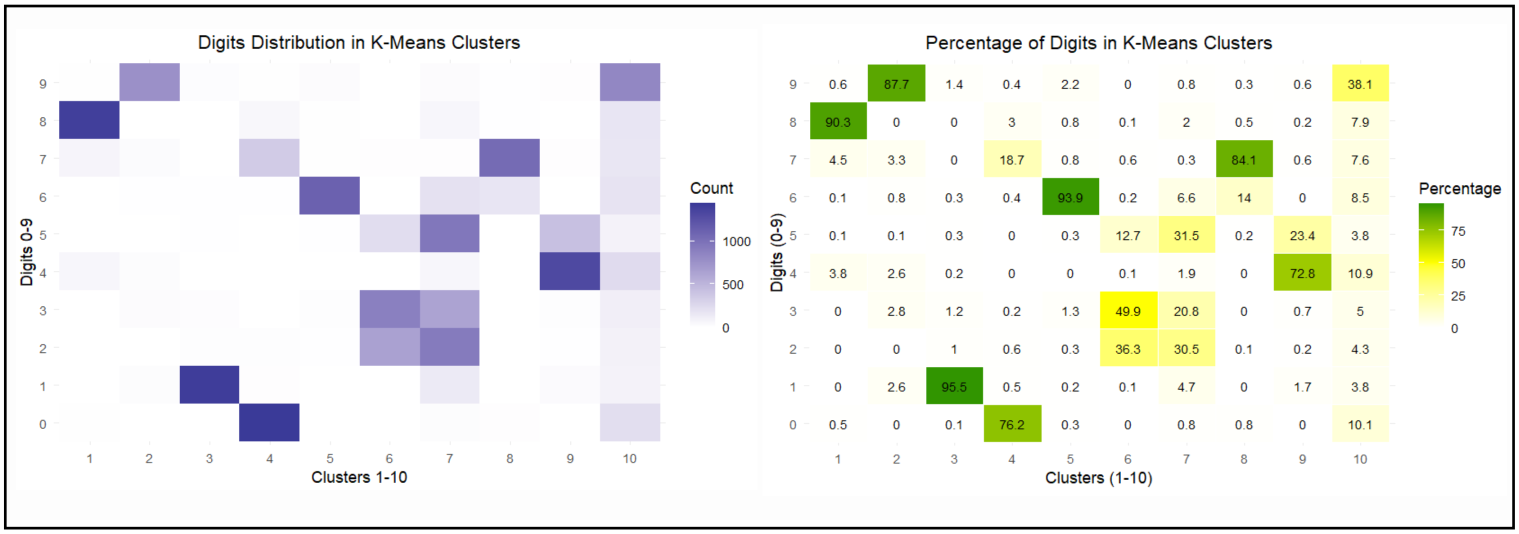
Plotting the 10-Clusters in K-means for the first two principal components.

Most of the clusters looks well separated with a little overlap. Further, the cluster 10 looks spread over other clusters, so does the cluster 7 over other clusters. Since it's a lot of clusters, I analyzed the proportions separately in a heatmap.

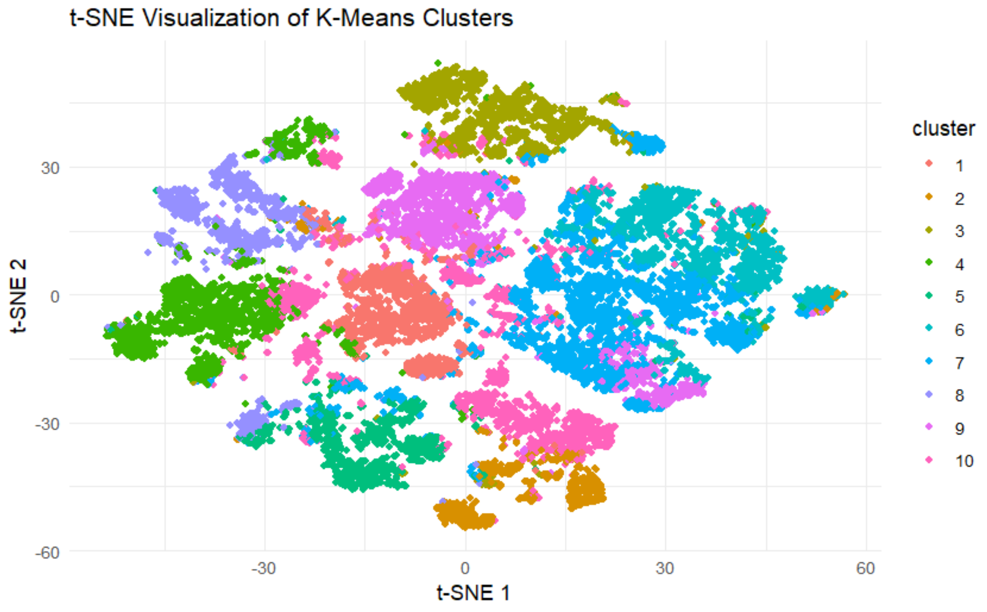
The Heatmap below shows the distribution of each digit in the different clusters. It shows that K-Means clustering was able to cluster most of the digits into separate clusters. Also, it was able to distinguish between digits 3 and 6, as in cluster 5 where there are 93.9% of images are for digits-6, only 1.3% images are of digit-3.

Further, from the heatmaps, we see that K-Means was not able to clearly distinguish between the digits 2,3 and 5 as seen from the proportion in Cluster-7, and Cluster-6 as presented in below table –

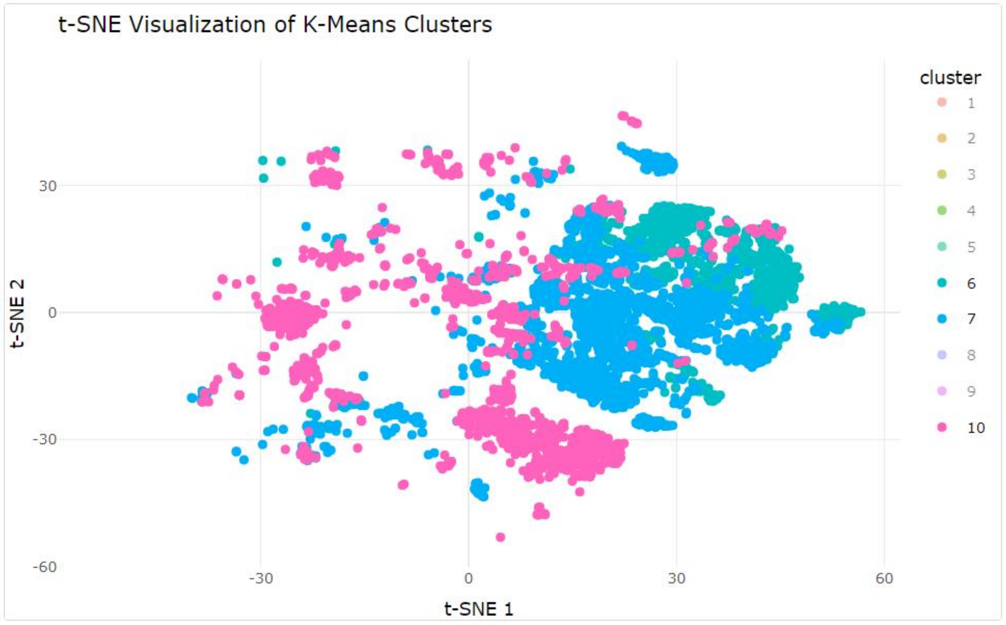
Cluster Number	Digit 2	Digit 3	Digit 5
Cluster-7	30.5%	20.8%	31.5%
Cluster-6	36.3%	49.9%	12.7%



Visualization using T-SNE

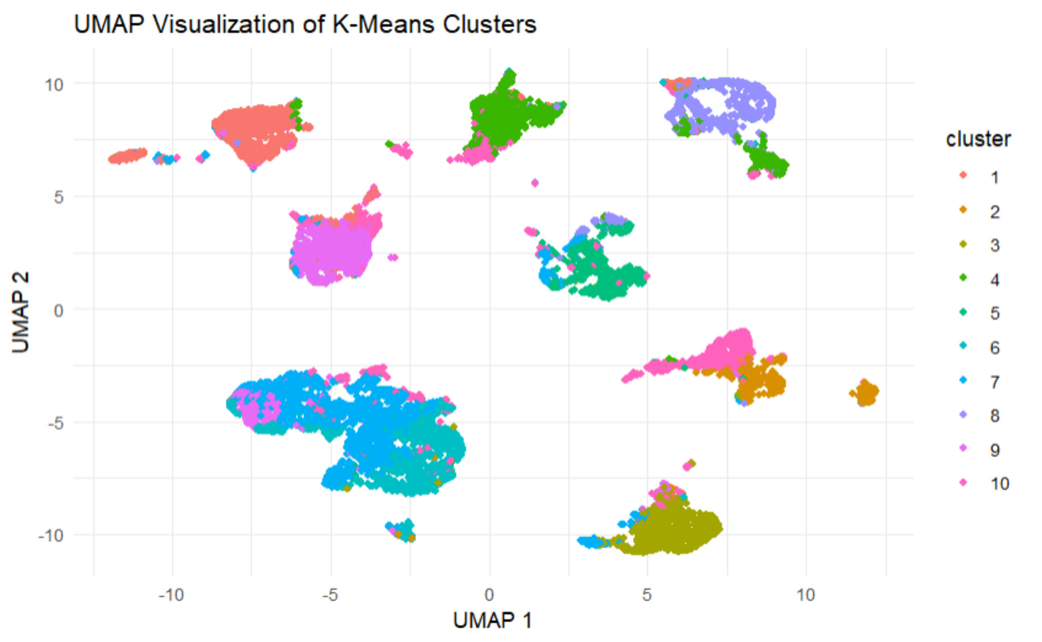


“Plot of All 10 Clusters”

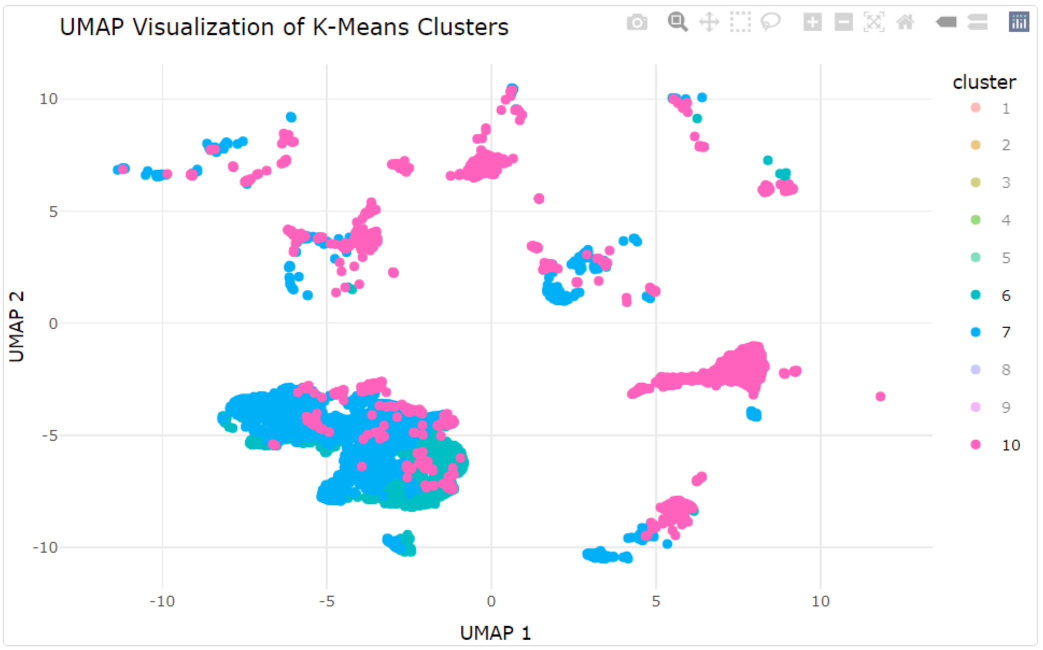


“Plot of Clusters 6, 7 and 10”

Visualization using UMAP



"Plot of All 10 Clusters"



"Plot of Clusters 6, 7 and 10"

The two visualizations using t-sne and UMAP provides a better visualization of the high-dimensional data clusters which was not clear from the overlapping representation in PCA.

We can clearly see the overlapping of the three clusters, 6, 7 and 10.

Brief Introduction of the Models used

1. Convolutional Neural Network (CNN)

CNNs are a class of deep learning models specifically designed for processing structured grid data, such as images. They utilize convolutional layers to automatically and adaptively learn spatial hierarchies of features from input images.

- **Architecture**: CNNs typically consist of:
  - **Convolutional Layers**: Apply filters to capture local features.
  - **Activation Functions**: Non-linear transformations (e.g., ReLU) to introduce non-linearity.
  - **Pooling Layers**: Down-sampling to reduce dimensionality while retaining essential features (e.g., max pooling).

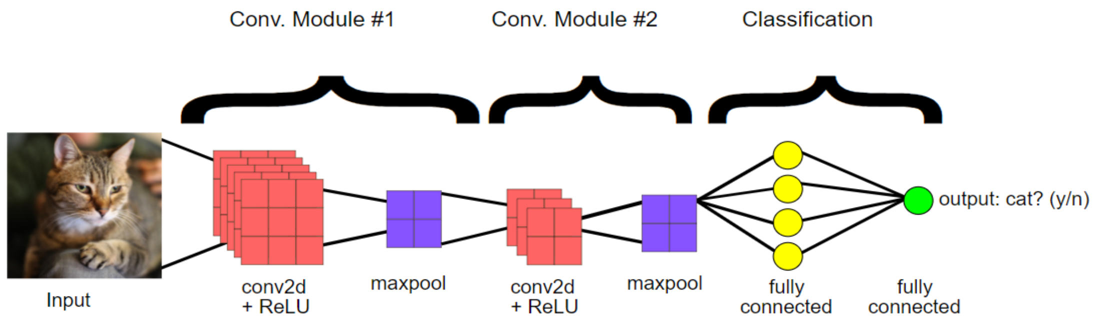


Figure 1 The CNN shown here contains two convolution modules (convolution + ReLU + pooling) for feature extraction, and two fully connected layers for classification.

  - **Fully Connected Layers**: Serve as classifiers by connecting every neuron from the previous layer to every neuron in the current layer.

- **Training**: CNNs are trained using backpropagation and gradient descent, adjusting the weights of filters to minimize the classification error on the labeled training data.

2. ResNet50

ResNet-50 is a convolutional neural network with 50 layers of depth. It is a pre-trained version of the neural network trained on over a million images from the ImageNet database. This pre-trained neural network is able to classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the neural network has learned feature-rich representations for a wide range of images. The neural network has an input image size of 224x224. ResNet50 is a type of CNN that incorporates residual learning through the use of skip connections. These connections allow gradients to flow through the network more effectively during backpropagation, mitigating the vanishing gradient problem that can occur in very deep networks. The "50" in ResNet50 refers to the network's depth, comprising 50 layers. The architecture includes convolutional blocks with batch normalization, ReLU activations, and average pooling.

Residual Learning: ResNet50 introduces shortcut connections that skip one or more layers, allowing gradients to flow more easily during training and enabling the training of much deeper networks. This approach facilitates the learning of residual functions rather than direct mappings.

Architecture:

ResNet-50 comprises:

- Multiple convolutional layers grouped into residual blocks.
- Average pooling layers to down-sample feature maps.
- Fully connected layers for classification at the output.

Input Size: ResNet-50 takes images of size 224x224 pixels and is pre-trained on the ImageNet dataset, enabling it to classify images into 1000 categories and learn robust feature representations.

3. Principal Component Analysis (PCA)

PCA is a dimensionality reduction technique that transforms a high-dimensional dataset into a lower-dimensional form while retaining as much variance as possible thus retaining the significant information. The algorithm computes the covariance matrix of the data and then derives its eigenvalues and eigenvectors. The eigenvectors with the largest eigenvalues represent the directions of maximum variance and are selected to form a new feature space.

Process:

Standardization: Center the data by subtracting the mean and scaling to unit variance.

Covariance Matrix: Compute the covariance matrix to understand the relationships between features.

Eigen-Decomposition: Calculate eigenvalues and eigenvectors of the covariance matrix.

Projection: Select the top K eigenvectors (principal components) to form a new feature space, projecting the original data onto this space.

4. K-Means

K-means is an unsupervised learning algorithm that partitions data into K distinct clusters based on feature similarity. It minimizes the within-cluster variance by grouping similar data points.

Algorithm:

1. Randomly initialize K cluster centroids.
2. Assign each data point to the nearest centroid, forming K clusters.
3. Recalculate the centroids based on the mean of the points in each cluster.
4. Repeat the assignment and centroid recalculation until convergence.

5. T-SNE (t-distributed Stochastic Neighbor Embedding)

T-SNE (t-distributed Stochastic Neighbor Embedding) is a machine learning algorithm used for the dimensionality reduction and visualization of high-dimensional data. It is particularly useful for representing data in two or three dimensions, thus allowing the visualization of complex patterns and structures in the data that might be difficult to interpret in higher dimensions.

How it works:

1. First, t-SNE converts high-dimensional Euclidean distances between data points into conditional probabilities that represent the similarities between points. These probabilities reflect how likely two points are to be neighbours.
2. It uses a uses a Student’s t-distribution in lower-dimensional space (2D or 3D) to model the similarities. The objective is to minimize the divergence between the probability distributions in both the high-dimensional and low-dimensional spaces.
3. Finally, the result is an embedding of the high-dimensional data in a lower-dimensional space, where similar data points are grouped together and dissimilar ones are far apart.

The important point to note is that t-snne focusses to preserve the local structure/ relationship between similar data points, and it may distort the global structure between the points overall.

6. UMAP (Uniform Manifold Approximation and Projection)

UMAP is also a dimensionality reduction technique, and it provides advantage over t-sne as it takes into account scalability, interpretability, and the ability to preserve both local and global data structures. It is also used for visualizing the high-dimensionality data.

How it Works:

1. It starts by constructing a weighted neighbourhood graph, where the data points are represented by nodes, and similarity between nodes is represented by edges. It uses a local distance metric, (Euclidean distance), to define the nearest neighbours of each data point.
2. It then uses the concepts of manifold learning and topological properties, to understand the global structure of data. The local neighbourhood points are combined to form a global structure of the data that reflects both local and global relationships.
3. It then optimizes the low-dimensional projection/ representation (2D or 3D) preserving the structure of graph. It uses a stochastic optimization technique (e.g. Gradient Descent/ Stochastic Gradient Descent) with a better trade-off between local and global structures.
4. Finally, it outputs the low-dimensional representation of data that is used for data visualization and further analysis.

Conclusion

I implemented three approaches, a CNN with 4-convolutional layers, implementation of ResNet50, and then K-means followed by PCA. Based on the comparison of the three models, I found that the CNN with 4-convolutional layers performed the best on the digit classification task with a test-accuracy of ~99%, followed by ResNet50 with test accuracy of ~89% and then K-Means as it was not able to distinguish between three digits considerably. Further, as cited well in literature also, PCA was not able to help in visualization of high-dimensionality data clusters, but T-SNE and UMAP worked well in visualization of the clusters.

The project also highlights the details about the important concept that I had in mind while performing this task to be able to distinguish the digits. As a kid, I was not able to distinguish the images that were flipped or looked the same to me for their way of writing, but the algorithms (even K-means) are able to distinguish them well. However, K-means is not able to distinguish the images for digits 2, 3 and 5 well, which possibly is due to shape similarity and variability in handwritten images in a continuous flow and use of distance metric by K-Means is not able to percept those changes as different images. Further, it is more likely that no one algorithm fits all stand true here also, as a simple 4 layer CNN model worked better than a more complex ResNet50 model on the digit classification (which was trained on 1000 images but not specifically on digits dataset (source)).

Further work

To further strengthen the results of K-means, I propose further analysis using the following approach, implementing the K-Means on the flattened layer from the CNN models. Since the flatten layer will have thousands of thousands of features, performing feature selection (PCA) on the flattened layer and then performing K-Means on the layer, might give better results as the convolutional layers will extract more important features from the image dataset. But at the same time, it adds further computational complexity to the model.

Additionally, random forest and hierarchical clustering can be used to further compare the performances of these model.

References

4. https://www.statlearning.com/
5. https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks
6. resnet50
7. https://keras.io/api/applications/resnet/#resnet50-function
8. https://image-net.org/
9. ImageNet Large Scale Visual Recognition Competition 2014 (ILSVRC2014)
10. https://www.rdocumentation.org/packages/cluster/versions/2.1.6
11. https://app.diagrams.net/ (for creating the flowchart)
12. “How to Use t-SNE Effectively”, Wattenberg et.al, https://distill.pub/2016/misread-tsne/

Appendix

The markdown files for the project are uploaded on the public repository, Github.

Link: https://github.com/sky-akash/StatisticalLearningClassProject