Assignment 3 (Semi - supervised learning)

Shruti Sharma (MDS202435)
Riya S Huddar (MDS202431)

Introduction

This project focuses on semi-supervised learning using clustering techniques to explore patterns in the FashionMNIST and overheadMNIST datasets. The primary goal is to use K-Means clustering to identify natural groupings within the image data without using label information.

The data is grouped into K-clusters.

- Initially, the image closest to each cluster centroid is selected as a **representative image**, forming a labeled set of K samples. These labels were first used to train a model directly.
- Then, in the **full propagation step**, the label of each representative was assigned to all points in its cluster.
- Finally, in **partial propagation**, labels were assigned only to the top 20% of points closest to each centroid within the cluster.

We also train a **MLP classifier** using only the top 20% most representative samples (closest to cluster centroids) whose labels were propagated from KMeans cluster representatives, and evaluate how accuracy varies with number of clusters (k).

Task 1: Fashion MNIST Dataset

1. Dataset Description

The Fashion MNIST dataset is a popular alternative to the original MNIST. It is a collection of 70,000 grayscale images of fashion products, suitable for image classification tasks in TensorFlow. It consists of 60,000 images for training and 10,000 for testing.

Each image is 28x28 pixels in grayscale, with pixel values ranging from 0 (black) to 255 (white) and categorized into 10 classes:

1. T-shirt/top
2. Trouser
3. Pullover
4. Dress

5. Coat
6. Sandal
7. Shirt
8. Sneaker
9. Bag
10. Ankle boot

Packages used:

- **numpy** – used for numerical operations and array manipulation.
- **matplotlib.pyplot** – used for plotting graphs and visualizing data.
- **KMeans from sklearn.cluster** – used for performing K-Means clustering.
- **LogisticRegression from sklearn.linear_model** – used for training a logistic regression classification model.
- **accuracy_score from sklearn.metrics** – used to evaluate the model's performance using accuracy.
- **fashion_mnist from tensorflow.keras.datasets** – used to load the Fashion MNIST image dataset.

2. Results and Discussion

For K=20:

1. Accuracy using only representative images: 0.6083
An accuracy of 0.6083 suggests a moderate level of performance with the limited information available from just these representative images.

2. Accuracy after full label propagation: 0.6075

- Full label propagation uses the relationships between data points (through similarity or distance measures) to propagate labels across the entire dataset. This means that once the representative images are labeled, the labels are spread across the rest of the data.

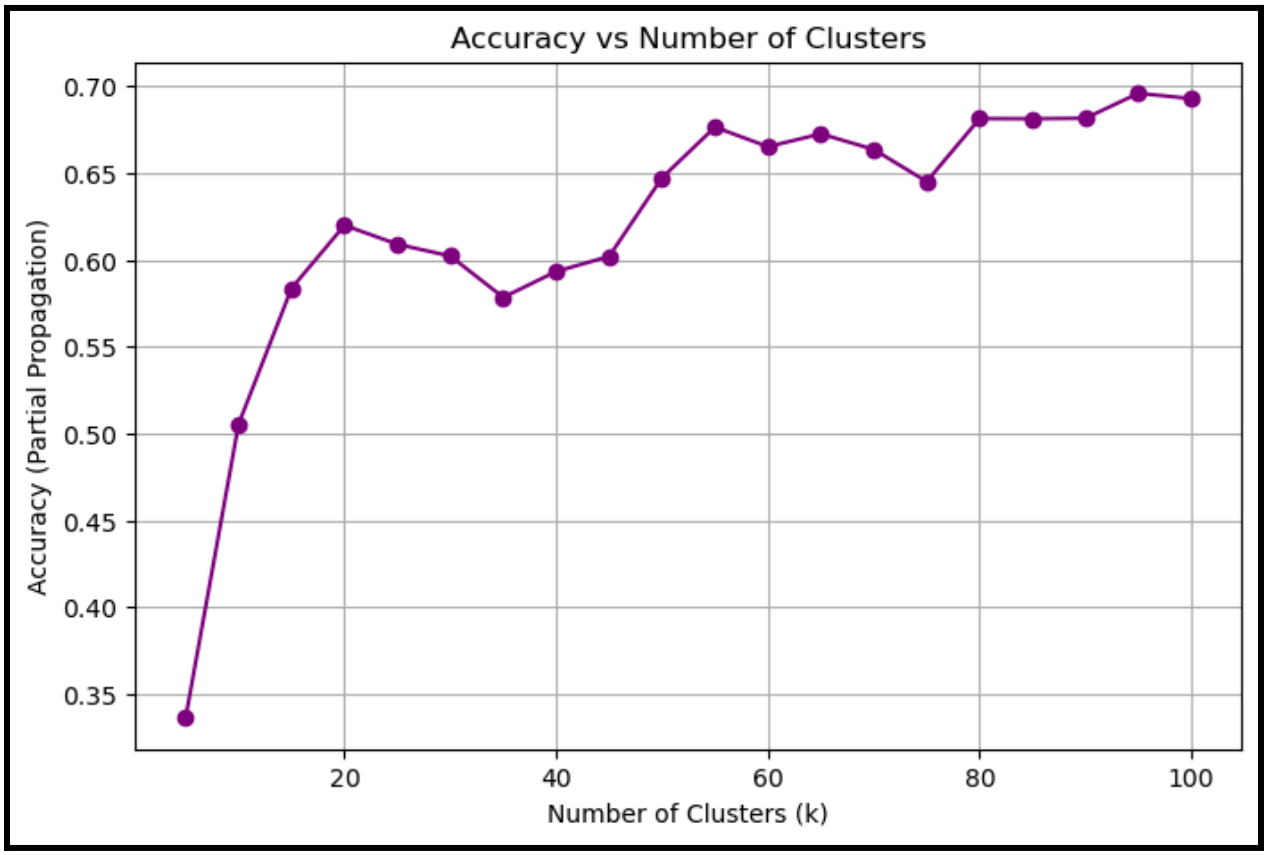
- The accuracy here, 0.6075, is actually slightly lower than the baseline accuracy (0.6083). This suggests that the label propagation, despite using all available data points, may have introduced noise or incorrect label assignments, which negatively impacted performance.

3. Accuracy with partial propagation (top 20% closest points): 0.6199

- Partial label propagation focuses on propagating labels only to the top 20% of the closest points. This technique likely emphasizes the most similar data points to the representative images, which could help propagate more accurate labels.

- The accuracy of 0.6199 is higher than both the baseline (0.6083) and full label propagation (0.6075), suggesting that by limiting propagation to the closest neighbors, you are reducing the potential for label noise and improving the overall performance.

Partial propagation results for more values of K:



The plot illustrates the relationship between the number of clusters (k) and the accuracy of the model (with partial propagation of labels).

Observations:

1. Initial Improvement (k = 10 to 20): As k increases from 10 to around 20, the accuracy significantly improves. This suggests that a higher number of clusters initially helps capture better feature separation for the model.

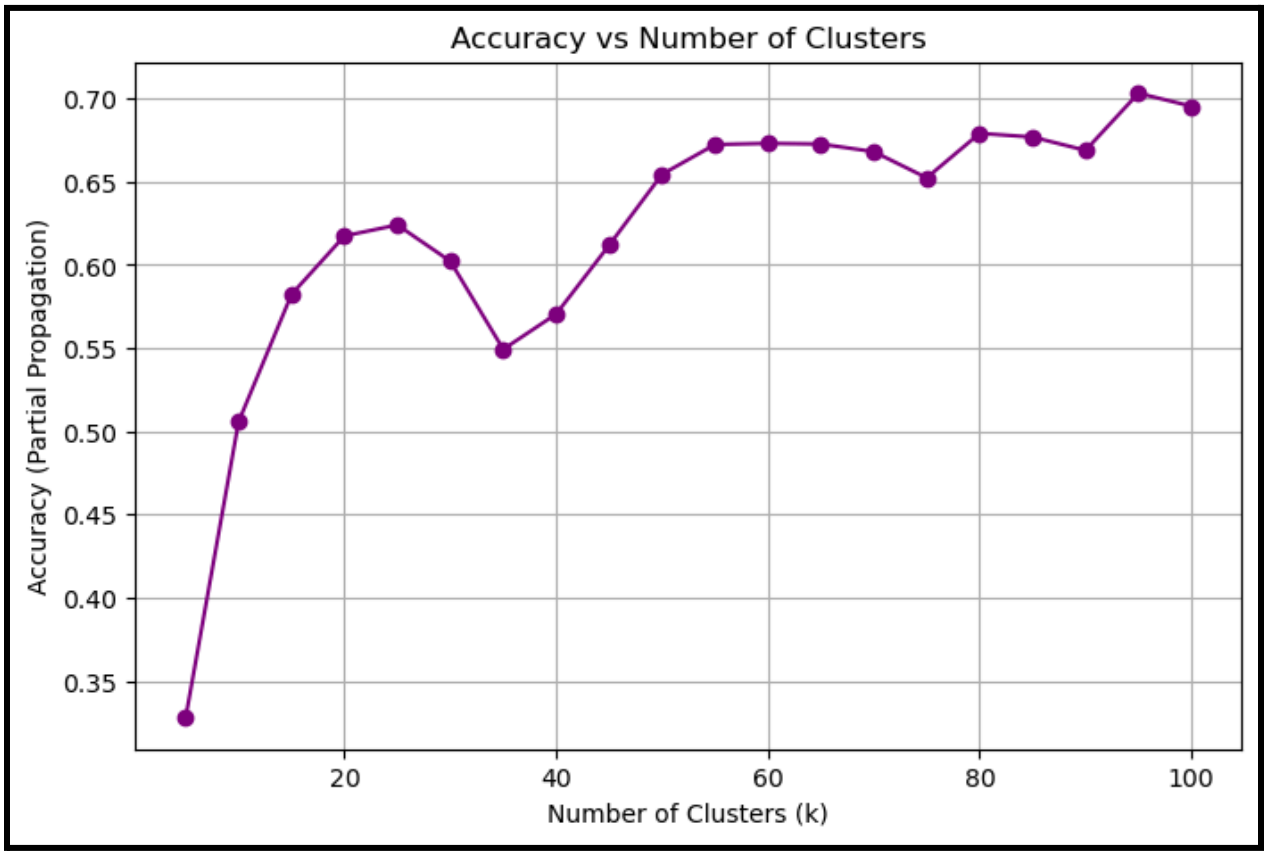
2. Peak Performance (k ≈ 20 to 30): There's a noticeable peak in accuracy, which likely corresponds to the optimal cluster size for the data being used. It indicates that, at this point, the clustering is most effective in dividing the data into meaningful groups for the model.

3. Stabilization (k ≈ 40 to 100): After around k = 30, the accuracy starts to stabilize, with only slight fluctuations in performance. This indicates that increasing k further doesn't

significantly improve the model's performance. It could be that the additional clusters are either redundant or do not add new insights to the model.

4. Possible Overfitting or Diminishing Returns: The accuracy does not continue improving beyond a certain point (about k = 30). This suggests that after this threshold, the model might be overfitting or that the number of clusters exceeds the effective capacity of the data for meaningful segmentation.

MLP classifier results:



The plot illustrates how the accuracy of a MLP model (under partial propagation) varies with the number of clusters (k).

1. **Initial Growth (k = 5 to 25):**

   Accuracy increases sharply from ~0.33 at k=5 to ~0.63 at k=25. Suggests that more clusters initially help capture the structure of the data better, improving label propagation or classification.

2. **Dip (k ≈ 30–40):**

Accuracy drops around k=35 (~0.55), showing a performance degradation. Possibly due to over-segmentation, where too many small clusters reduce generalization or introduce noise.

3. **Stabilization & Plateau (k = 45 to 100):**

   Accuracy steadily rises again and plateaus around 0.67–0.70. From k=50 onward, performance remains relatively stable, suggesting diminishing returns from increasing clusters further.

The results of the MLP model are similar to that of the logistic regression model. As per both the models, the optimal number of clusters seems to be around k = 20 to 30, after which the model's performance plateaus.

3. Conclusion

The results suggest that partial label propagation, which uses only the top 20% closest points for label assignment, leads to the best performance, outperforming both representative images only and full label propagation.

Restricting label propagation to the most similar data points helps reduce noise and improves accuracy. Full label propagation, on the other hand, may introduce incorrect label assignments and slightly decrease the model's performance.

Thus, **limiting propagation to the closest neighbors** seems to be the most effective strategy in this case for achieving better results.

Task 2: Overhead MNIST Dataset

1. Dataset Description

Overhead MNIST, or OMNIST, is a satellite imagery dataset designed to serve as a drop-in replacement for the classic MNIST handwritten digit dataset.

It contains 76,690 grayscale satellite images, each of size 28x28 pixels, representing 10 object classes commonly found in satellite imagery. These classes include:

1. Car
2. Harbor
3. Oil and gas field
4. Parking lot
5. Plane

6. Ship
7. Storage tank
8. Helicopter
9. Baseball diamond
10. Tennis court

Packages used:

1. **numpy** – used for efficient numerical computations and array operations.
2. **struct** – used for interpreting packed binary data (e.g., when reading raw data files).
3. **KMeans from sklearn.cluster** – used for unsupervised clustering of data into k groups.
4. **LogisticRegression from sklearn.linear_model** – used to train a logistic regression model for classification.
5. **StandardScaler from sklearn.preprocessing** – used to standardize features by removing the mean and scaling to unit variance.
6. **mode from scipy.stats** – used to compute the statistical mode (most frequent value) of an array.
7. **matplotlib.pyplot** – used for creating plots and visualizing results.
8. **pandas** – used for data manipulation and analysis using DataFrames.
9. **classification_report, accuracy_score, multilabel_confusion_matrix from sklearn.metrics** – used to evaluate classification model performance.
10. **time as t** – used to measure execution time or implement delays.
11. **MLPClassifier from sklearn.neural_network** – used to create and train a multilayer perceptron (MLP) neural network for classification.

2. Results and Discussion

An important implementation detail that significantly impacted performance was the choice of optimization algorithm in the classifier used for label propagation.

- Without solver='saga': Accuracy remained in the low 20% range.
- With solver='saga': Accuracy increased substantially to the 40–45% range.

saga is a variant of stochastic gradient descent that is particularly well-suited for large-scale datasets and sparse data. Switching to solver='saga' led to a doubling of accuracy, indicating that previous poor performance (in the 20s) was likely due to suboptimal convergence or inability of the solver to handle sparse, high-dimensional inputs effectively.

[k=50]
Accuracy using only representative images: 0.2300469483568075
Full Propagation Accuracy: 0.4535
Top 20% Accuracy : 0.4526

- A **huge jump** from 0.2300 (baseline) to ~0.45 after propagation suggests that representative images alone are very weak (perhaps too few or unrepresentative), and propagation provides significant benefit.

- However, full and partial propagation give very similar accuracy here.

Partial propagation results for more values of K:

[k=20] Completed in 740.25 seconds
  Top 20% Accuracy : 0.4413

[k=30] Completed in 439.52 seconds
  Top 20% Accuracy : 0.4498

[k=40] Completed in 319.22 seconds
  Top 20% Accuracy : 0.4554

[k=50] Completed in 394.86 seconds
  Top 20% Accuracy : 0.4526

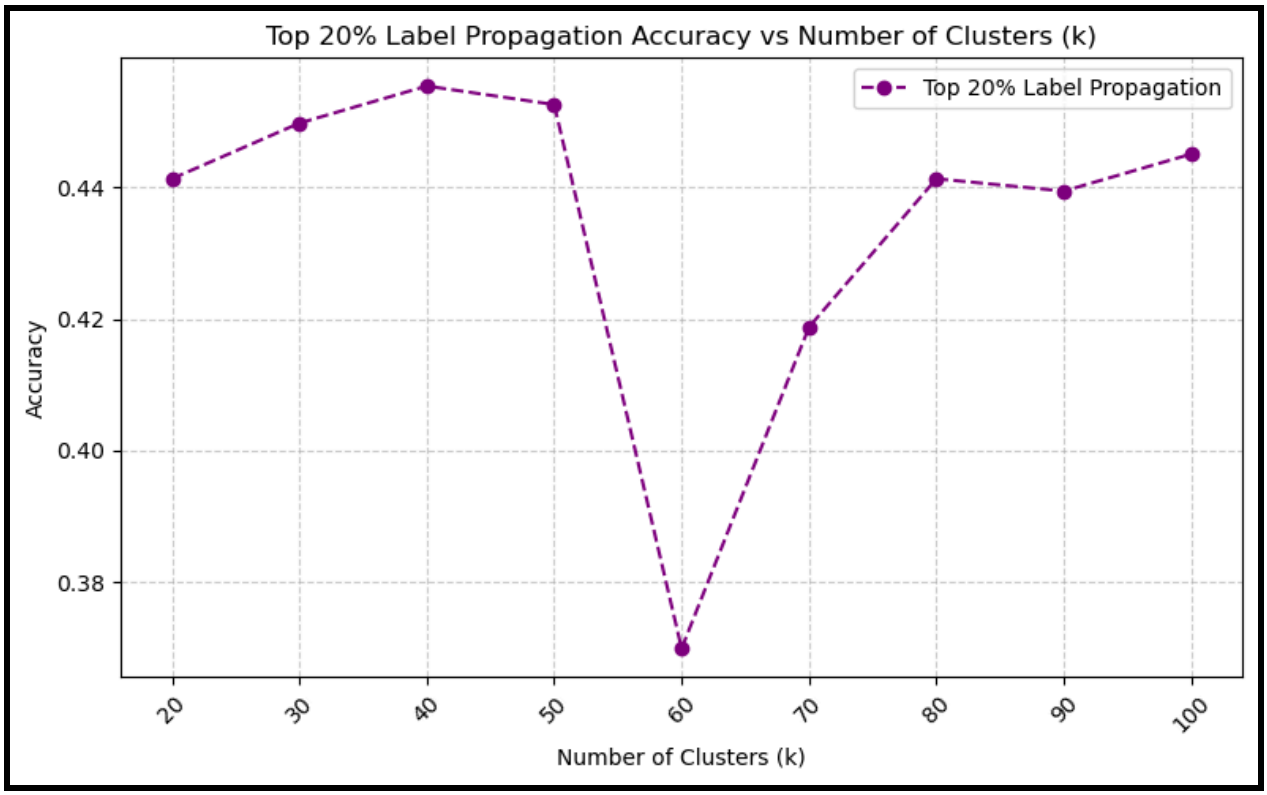
[k=60] Completed in 200.28 seconds
  Top 20% Accuracy : 0.3700

[k=70] Completed in 241.60 seconds
  Top 20% Accuracy : 0.4188

[k=80] Completed in 394.32 seconds
  Top 20% Accuracy : 0.4413

[k=90] Completed in 219.74 seconds
  Top 20% Accuracy : 0.4394

[k=100] Completed in 262.73 seconds
  Top 20% Accuracy : 0.4451



The plot shows how the accuracy of label propagation (restricted to the top 20% closest neighbors) varies as a function of the number of clusters (k). Here's a detailed analysis:

- **k = 20 to k = 40**:
  Accuracy **increases steadily**, peaking around **k = 40**, which achieves the highest accuracy (~0.455).

  Interpretation: At lower values of k, the graph structure may be too sparse or overly local, but as k increases, the graph captures more meaningful structure, improving label propagation.

- **k = 40 to k = 60**:
  There's a **dramatic drop** in accuracy at **k = 60**, hitting the lowest point (~0.37).

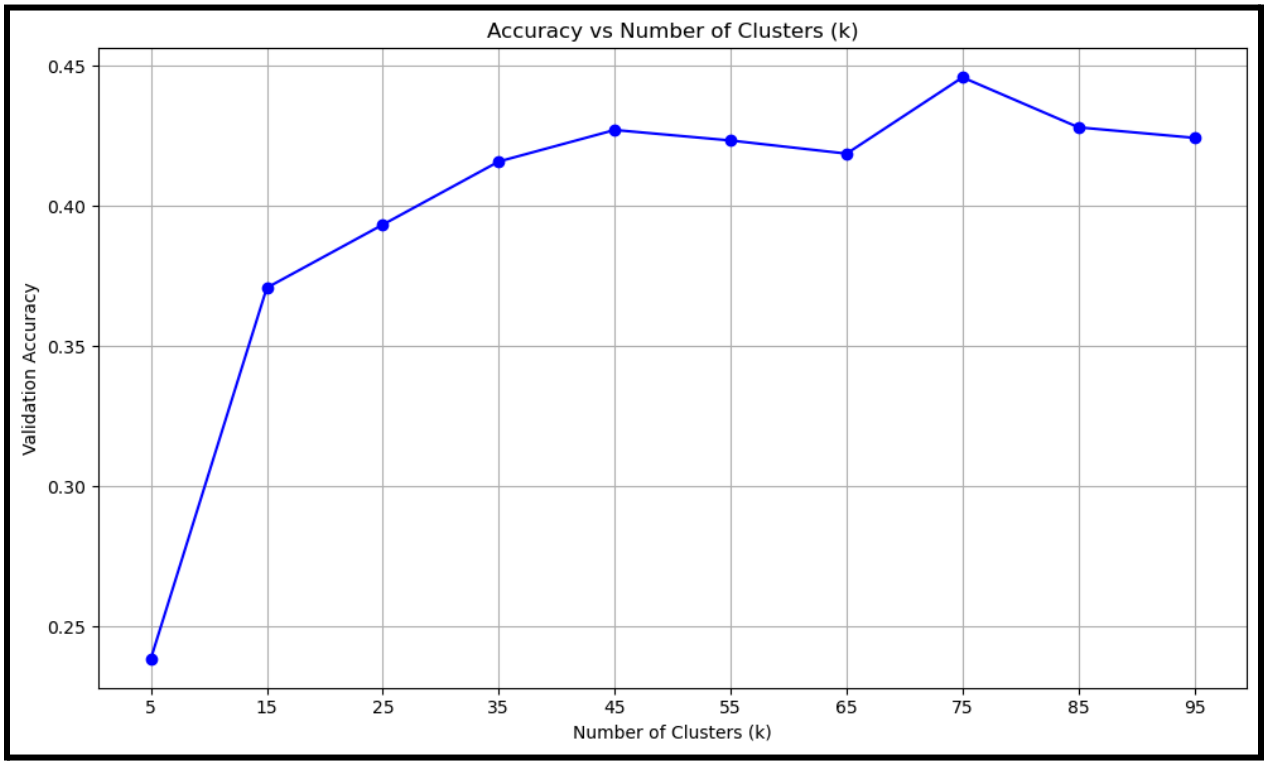
  Interpretation: At this point, label propagation likely begins connecting dissimilar data points, introducing **noise** and harming label quality.

- **k = 60 to k = 100**:
  Accuracy **recovers** gradually from the low at k = 60 and stabilizes around 0.44–0.445 at higher k values.

Interpretation: Increasing k further seems to reduce the negative effect of k = 60, but never quite reaches the peak at k = 40. This suggests that some structural information is regained, but propagation remains slightly noisier compared to the optimal region.

MLP classifier results:

Propagation: 0.2



This plot shows the **Validation Accuracy vs Number of Clusters (k)** for the MLP model with 20% propagation.

1. Initial Steep Increase (k = 5 to 15):

Validation accuracy jumps from **~0.24 to ~0.37**, indicating that even a small number of clusters provides a significant structure to the data that the model can learn from.

2. Steady Gains (k = 15 to 45):

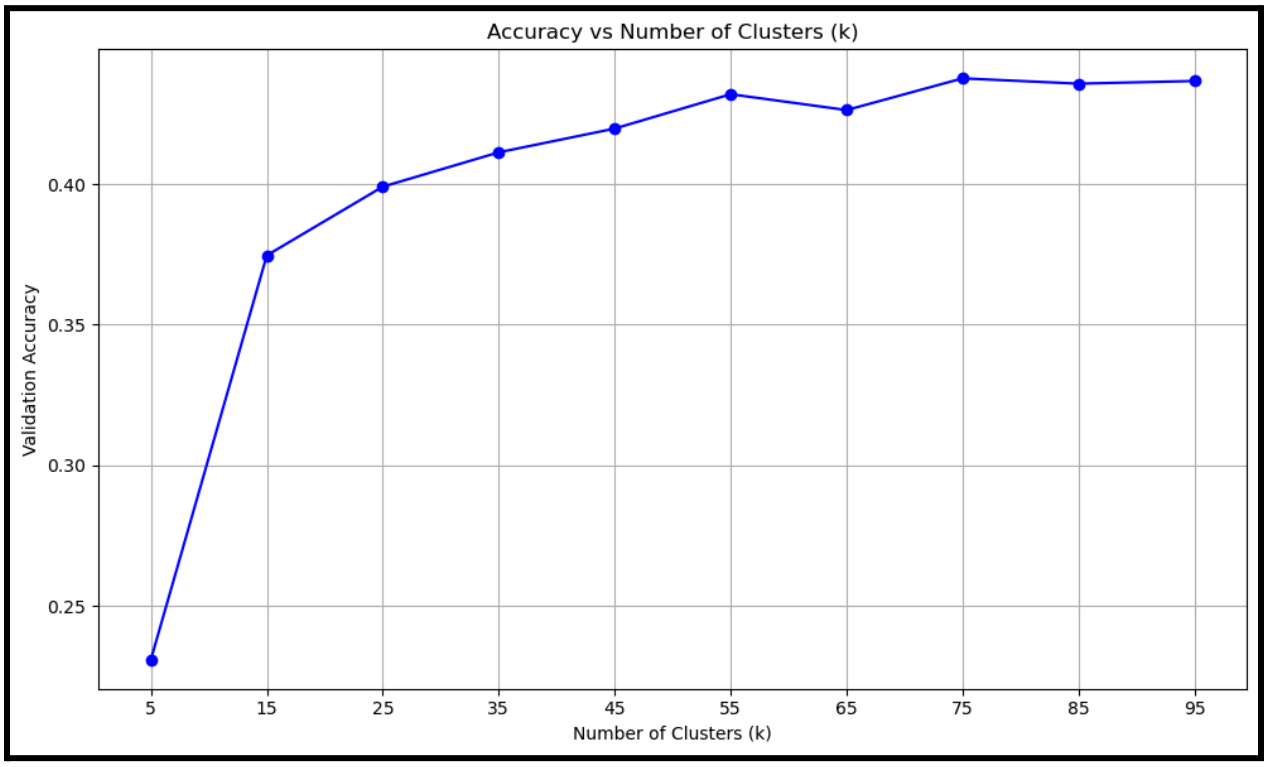
Accuracy rises gradually from **~0.37 to ~0.425**. This indicates that the clusters are becoming more refined and informative for the MLP.

3. Plateauing Behavior (k > 45):

Accuracy stabilizes after thereafter. This suggests that beyond a certain point, additional clusters do not significantly improve the model's ability to generalize. The MLP may be reaching

the limit of how much it can benefit from the cluster-based features, either due to redundancy in the new clusters or because they introduce noise rather than meaningful patterns.

Propagation: 0.3



30% label propagation gives similar results as 20% label propagation.

3. Conclusion

The logistic regression model clearly suggests a non-monotonic relationship between k and accuracy. Moderate values of k (around 30–50, especially k = 40) provide the best trade-off between local fidelity and global connectivity in the graph used for label propagation. Values too low miss important connections; values too high introduce noise. This suggests that careful tuning of k is critical for maximizing semi-supervised learning performance.

The MLP model suggests an optimal range for k appears to be around 45–75, where the balance between feature richness and generalizability is best achieved. Pushing k beyond this may not justify increased computational cost, as the accuracy remains flat.

The overall accuracy on the overhead MNIST dataset is generally lower, likely due to the presence of noisy data. However, this performance could potentially be improved through careful hyperparameter tuning.