Alexandria University

Faculty of Engineering

Computer and Systems Engineering Department

Assignment #1: Heart Failure Classification Problem

CSE: Pattern Recognition Assigned: Sun, Mar. 02, 2025 Due: Tue, Mar. 11, 2025

Submitted by:

Omar Hani Bishr - 21010891

Mohamed Hassan Mohamed Qabari Hassan - 21011115

Mohamed Mohamed Ibrahim Ali Hassan – 21011211

Instructor: Prof. Dr. Marwan Torki

TA: Eng. Ismail El-Yamany

Approach

Dataset Preparation and Preprocessing

We used the **ORL Face Dataset**, which contains grayscale images of 40 individuals, each with 10 images of size 92×112 pixels. The following steps were taken to prepare the data:

1. **Image Flattening:**
   Each image was converted into a 1D vector of size $10304=92\times112$10304=92×112, and all 400 images were stacked to form a **data matrix** of shape **(400, 10304)**.
2. **Label Vector:**
   Labels were extracted from folder names (e.g., `s1`, `s2`, …, `s40`) and stored in a vector of shape **(400,)** with values from 1 to 40, representing subject IDs.
3. **Normalization:**
   All pixel values were normalized to the range [0, 1] by dividing by 255.
4. **Train/Test Split (Per Subject):**
   To ensure equal subject representation:
   a. Odd-indexed images (1st, 3rd, 5th, 7th, 9th) were used for **training** — 5 images per subject.
   b. Even-indexed images (2nd, 4th, 6th, 8th, 10th) were used for **testing** — 5 images per subject.
   c. Final shapes:

i. Training set: **(200, 10304)**
ii. Test set: **(200, 10304)**
iii. Labels: integers from 1 to 40

Principal Component Analysis (PCA)

To reduce dimensionality while preserving facial structure, we implemented PCA from scratch with the following steps:

1. **PCA Class Implementation:**
    a. `fit(X)`:
        i. Centers the data and computes the **covariance matrix**.
        ii. Extracts **eigenvalues and eigenvectors**, then sorts them in descending order.
        iii. Saves them for later use.
    b. `reduce_dimensions(X, alpha)`:
        i. Selects the **minimum number of components** such that the retained variance ≥ `alpha` (e.g., 0.9 = 90%).
        ii. Projects the data onto the selected eigenvectors (PCA space).
        iii. Returns the reduced representation.
    c. `visualize_reconstruction(...)`:
        i. Reconstructs and visualizes faces from reduced PCA space to compare with originals.
2. **Training and Testing Transformation:**

a. Applied PCA to both **training** and **testing** sets separately for multiple `alpha` values: **[0.80, 0.85, 0.90, 0.95]**.
b. Printed the number of principal components retained for each `alpha`.

KMeans Clustering on PCA-Reduced Data

To group similar facial features without supervision, we implemented a **custom KMeans algorithm** and evaluated clustering accuracy:

1. **Custom KMeans Implementation:**
    a. Randomly initialized K centroids from training data.
    b. Iteratively:
        i. Assigned data points to the nearest centroid.
        ii. Updated centroids based on the mean of assigned points.
        iii. Checked for convergence using a tolerance threshold.
2. **Cluster-to-Label Mapping:**
    a. Mapped each cluster to the **most common true label** in that cluster using `Counter`.
    b. Predicted labels were then compared to true labels to compute clustering **accuracy**.
3. **Experiment Setup:**

a. Ran KMeans on each reduced training set for PCA `alpha ∈ {0.80, 0.85, 0.90, 0.95}`.
b. Tested with cluster counts `K ∈ {20, 40, 60}`.
c. Calculated accuracy using mapped labels.

4. **Visualization:**
    a. Created two line plots:
        i. **Accuracy vs K** for each alpha.
        ii. **Accuracy vs Alpha** for each K.
    b. Helped visualize the tradeoff between dimensionality reduction and clustering performance.

Gaussian Mixture Model (GMM)

Objective

To implement a robust and numerically stable Gaussian Mixture Model (GMM) from scratch that can cluster high-dimensional data using the Expectation-Maximization (EM) algorithm with multiple initializations and early stopping.

Key Components of the Approach

1. **Initialization Strategy**
    a. Means initialized using random data samples with added noise to avoid duplicates.

b. Covariances initialized using scaled identity matrices based on data variance.
c. Weights initialized with random non-uniform values to break symmetry.

2. EM Algorithm

a. **E-step**: Calculates responsibilities using log-probabilities for numerical stability.
b. **M-step**: Updates weights, means, and covariances using weighted statistics.
c. Covariance matrices are regularized to prevent singularities and ensure positive definiteness.

3. Numerical Stability Techniques

a. Uses `logsumexp` to prevent underflow in soft assignments.
b. Adds `reg_covar` and additional regularization in exceptional cases (e.g., low determinant).
c. Handles singular matrix errors via fallback mechanisms in both E-step and scoring.

4. Multiple Initializations & Model Selection

a. Runs `n_init` times with different initializations.
b. Selects the model with the highest log-likelihood score on the training data.

5. Convergence & Early Stopping

a. Monitors log-likelihood difference.

b. Stops early if no improvement for `early_stop` consecutive iterations.

6. **Prediction & Scoring**
    a. Predicts cluster labels using maximum posterior (responsibility).
    b. Computes total log-likelihood of data under the current GMM parameters.

Approach for Dimensionality Reduction with Autoencoder Embeddings

This **convolutional autoencoder (CAE)** compresses input images into a **low-dimensional embedding** (7×6×1 = **42D**), then reconstructs them.

Key Steps:

1. **Encoder (Dimensionality Reduction)**
    - Stacks **convolutional layers + max-pooling** to progressively downsample the image.
    - Final embedding: **7×6×1 tensor** (42D flattened).
2. **Decoder (Reconstruction)**
    - Uses **transposed convolutions** to upsample the embedding back to the original size.

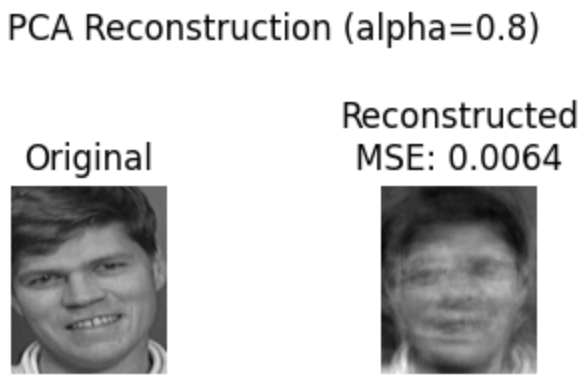
3. **Usage for Embeddings**
    - After training, discard the decoder.
    - Use encoder(x) to extract **compressed embeddings** (42D vectors) for downstream tasks (e.g., clustering, classification).

Why It Works

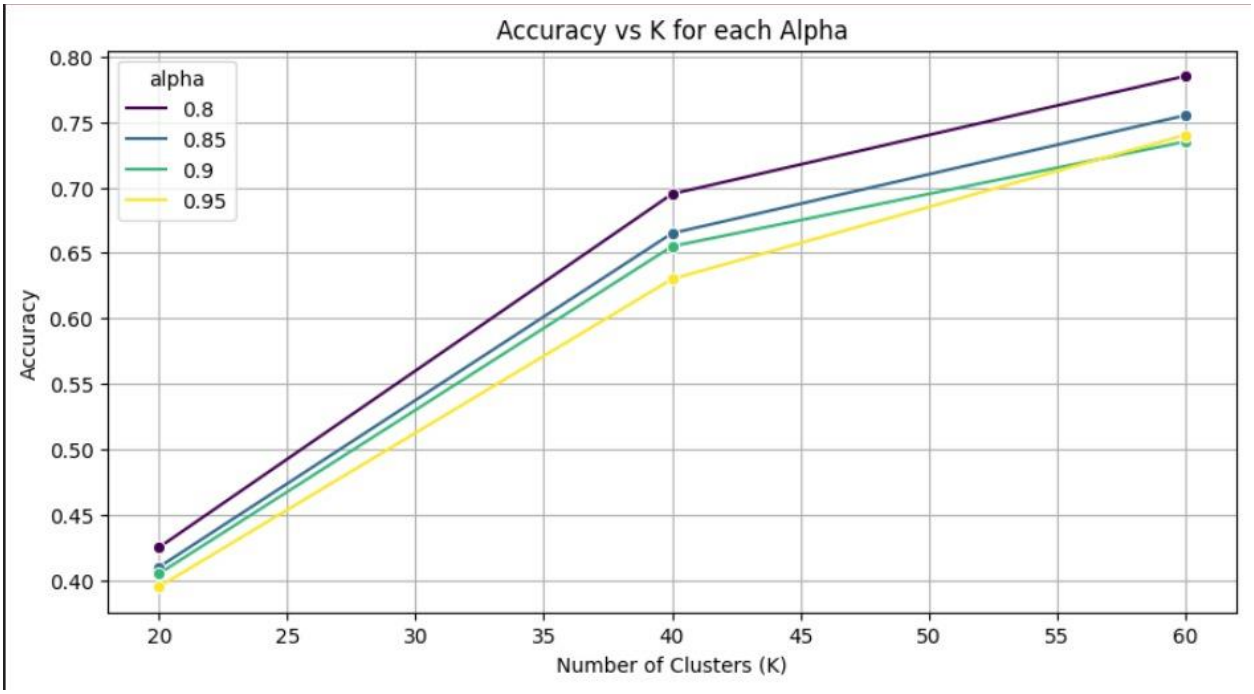
- The bottleneck layer (7×6×1) forces the network to learn a **compact representation**.
- Embeddings retain critical features while reducing dimensionality.

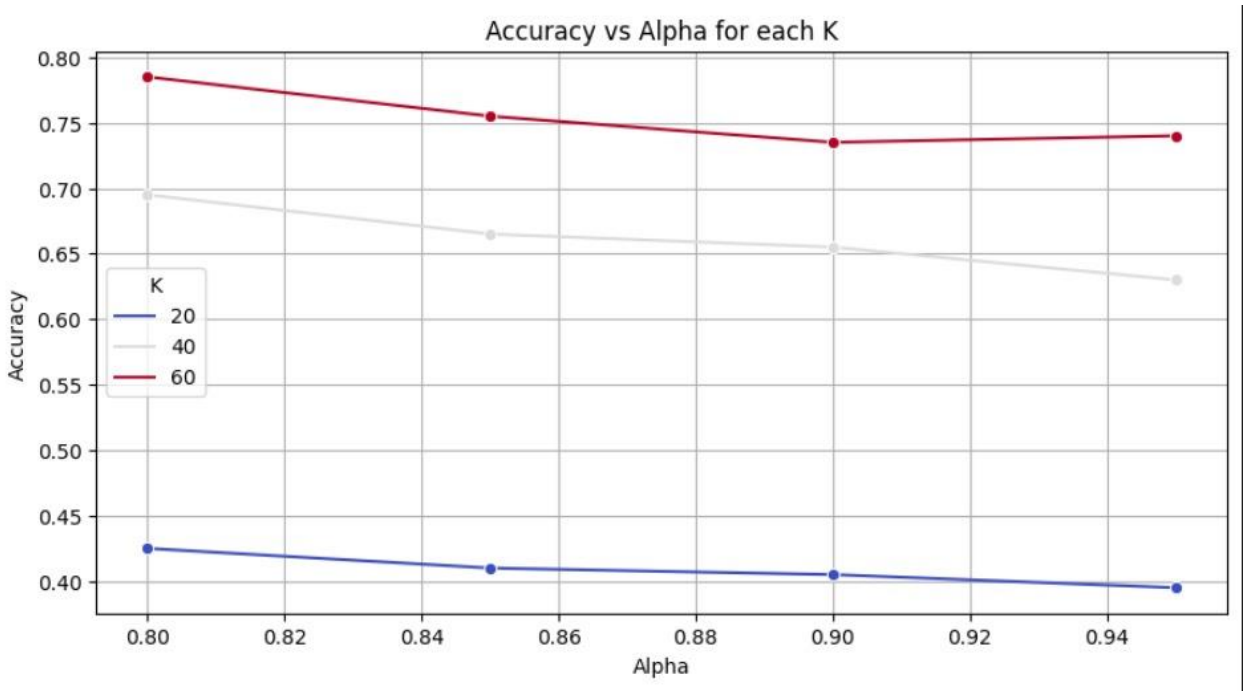
Results

Using PCA


PCA Reconstruction (alpha=0.8)
Original
Reconstructed
MSE: 0.0064

K-means


Accuracy vs K for each Alpha
alpha
0.8
0.85
0.9
0.95
Accuracy
Number of Clusters (K)
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
20
25
30
35
40
45
50
55
60

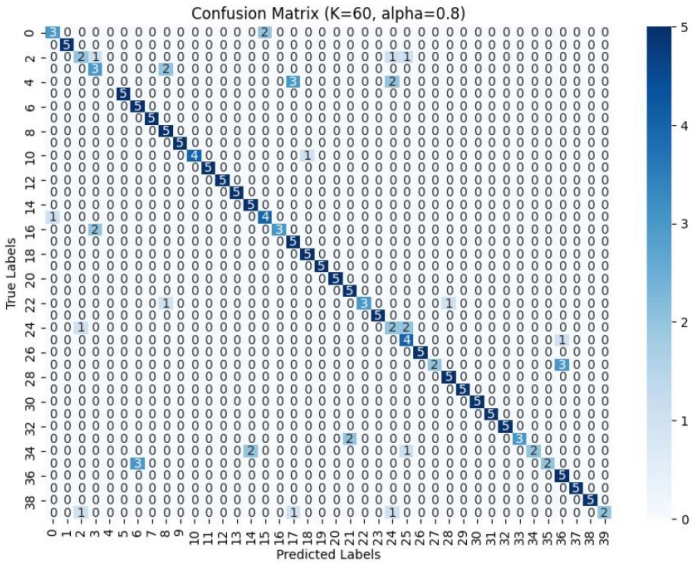


Best parameters -> Alpha: 0.8, K: 60

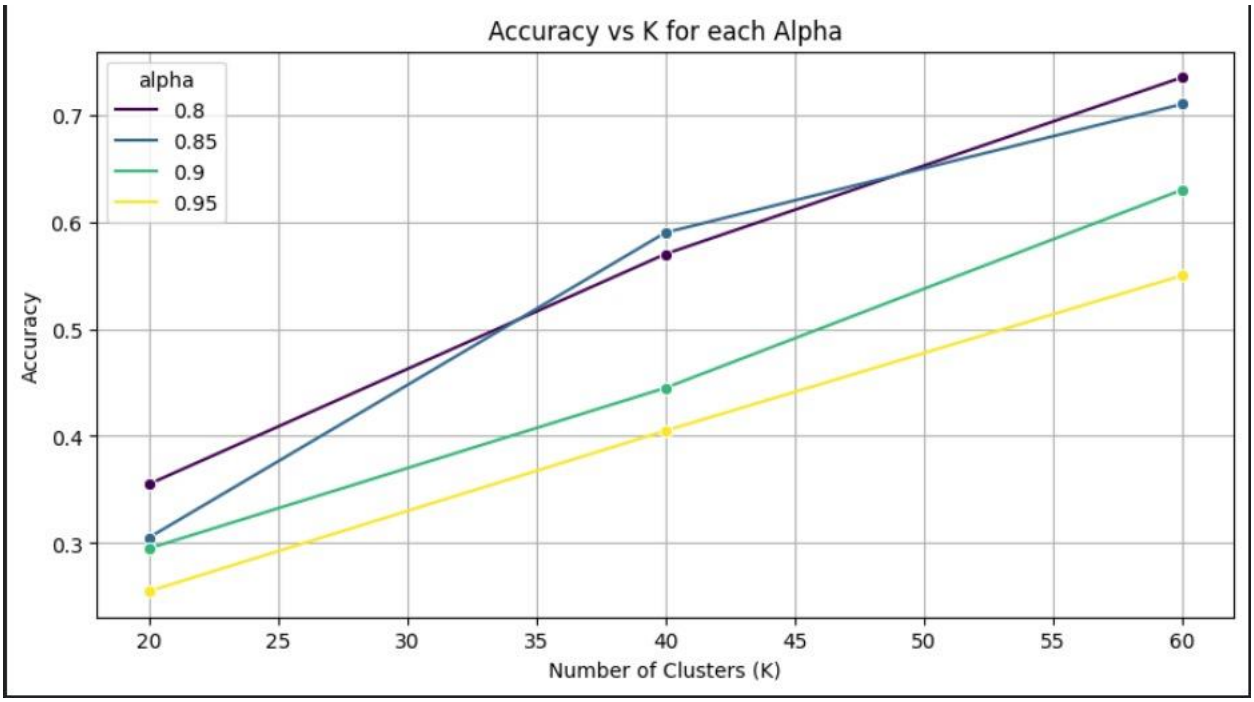
Results for K=60, alpha=0.8:

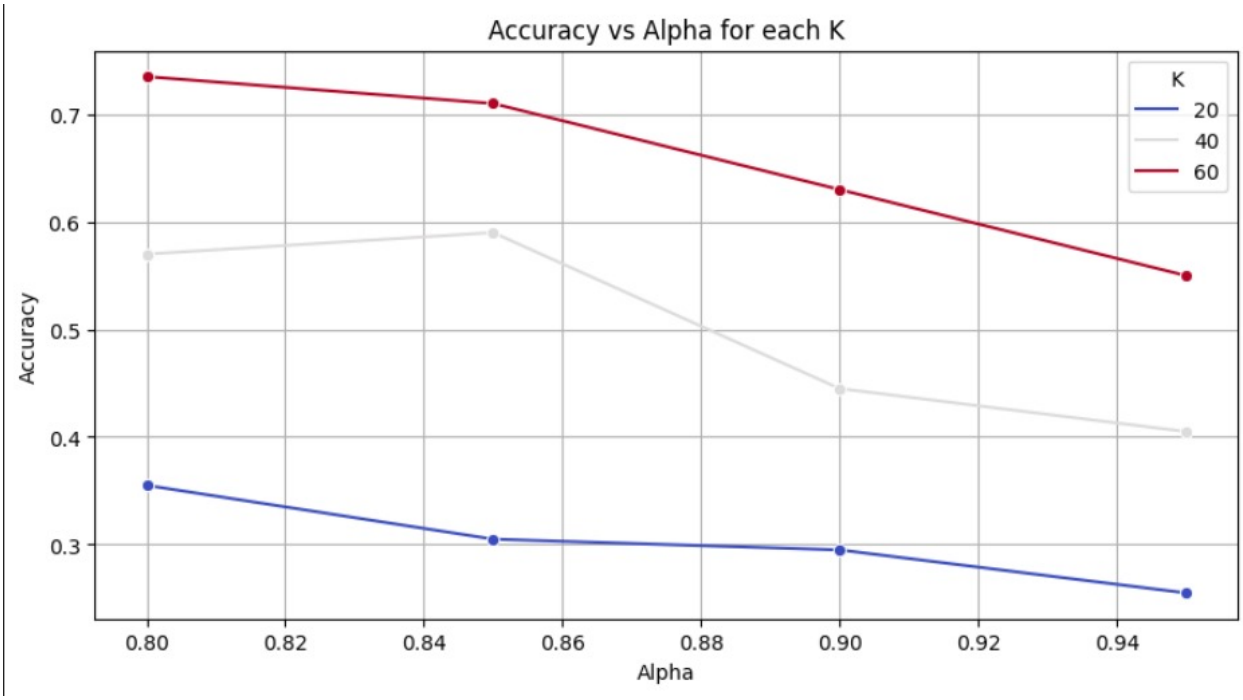
→ Accuracy: 0.8200

→ F1 Score: 0.8060


Confusion Matrix (K=60, alpha=0.8)
True Labels
Predicted Labels

GMM

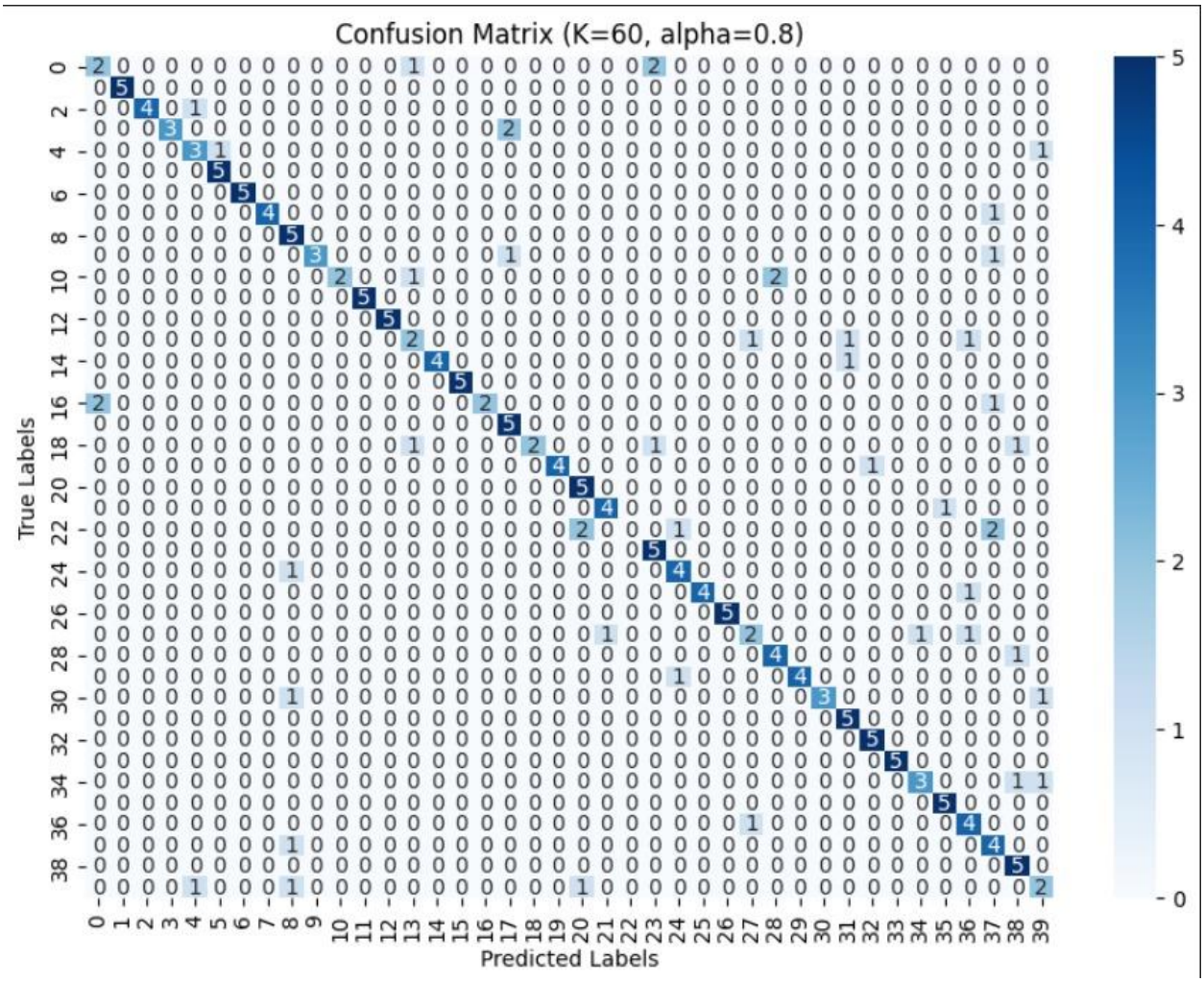

Accuracy vs K for each Alpha
alpha
0.8
0.85
0.9
0.95
Accuracy
Number of Clusters (K)



Results for K=60, alpha=0.8:

→ Accuracy: 0.7650

→ F1 Score: 0.7524

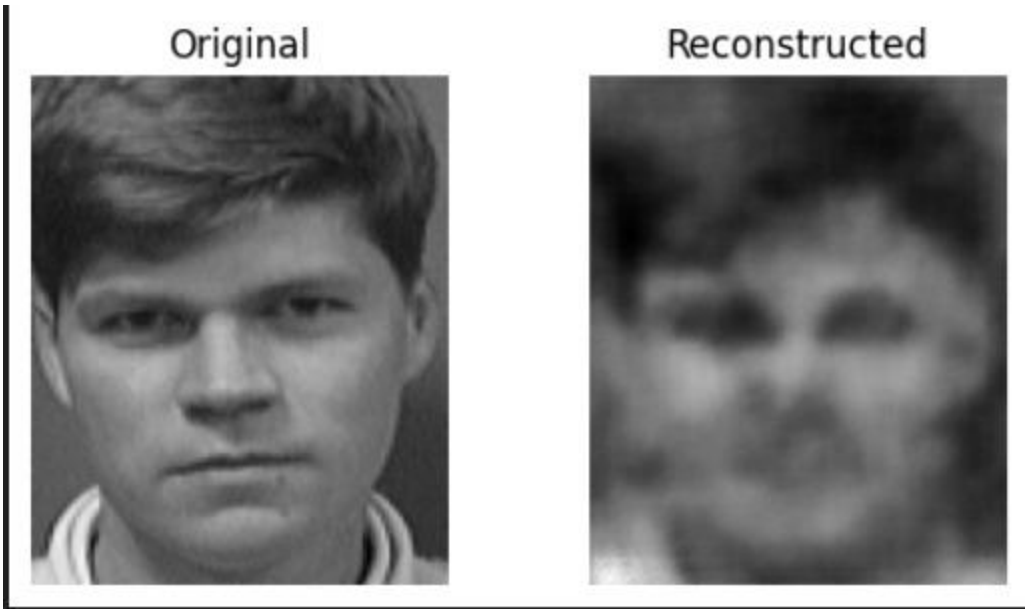


PCA-based Clustering (K = 60, α = 0.8)

Algorithm	Accuracy	F1 Score
KMeans	0.8200	0.8060
GMM	0.7650	0.7524

Using Auto-Encoder

Test Loss: 0.0073

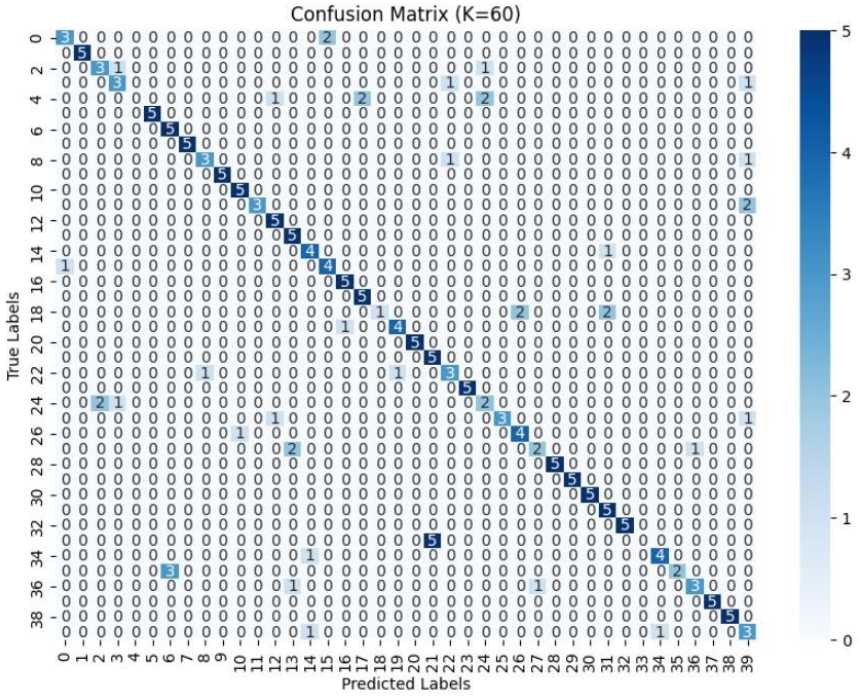


K-means

Results for K=60:

→ Accuracy: 0.7700

→ F1 Score: 0.7452

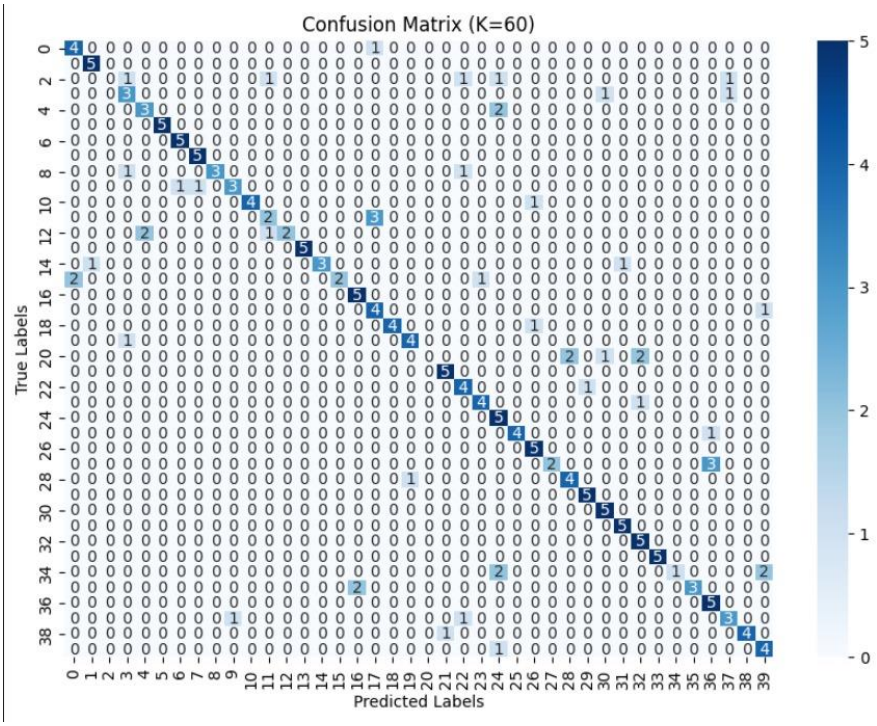


GMM

Results for K=60:

→ Accuracy: 0.7450

→ F1 Score: 0.7194



K: 20 → Accuracy: 0.4200

K: 40 → Accuracy: 0.6200

K: 60 → Accuracy: 0.7400

Autoencoder-based Clustering (K = 60)

Algorithm	Accuracy	F1 Score
KMeans	0.7700	0.7452
GMM	0.7450	0.7194

Analysis

Both in K-means & GMM

Can you find a relation between alpha and classification accuracy?

as alpha increases, accuracy decreases

Can you find a relation between K and classification accuracy?

as K increases, classification accuracy increases.

Using PCA

K-means

Best parameters -> Alpha: 0.8, K: 60

GMM

Best parameters -> Alpha: 0.8, K: 60

Using Auto-Endor

K-means

Best parameters -> K: 60

GMM

Best parameters -> K: 60