**COMP257-003**

**Unsupervised & Reinforcement Learning**

**Final Project**

Written Report

**Team Gamma**

**Professor: Ashish Gupta**

**301145308 JAEUK KIM​**

**301180583 SOHAM RAJPUT​**

**301209079-Utkarsh Kaushik​**

**301210328 Mirunan Pirabaharan​**

**301221570 Ripudaman​**

**301233539 Noah Tsang**

**1. Rationale for the split ratio**

Empirically, the best results are obtained if we use 20-30% of the data for testing, and the remaining 70-80% of the data for training. In this paper, we provide a possible explanation for this empirical result.

The motivation for using an 80 / 20 split is loosely driven by the Pareto principle (also called the 80–20 rule), which states that 80% of the effect is driven by 20% of causes (and vice versa). The Pareto principle isn’t a mathematically guaranteed property, but many observed phenomena follow the Pareto principle. For example, wealth distribution (80% of all the wealth is held by 20% of people), social media marketing (80% of all social media shares are generated by 20% of posts), agricultural production (20% of farmers produce 80% of crops), etc. Given how ubiquitous the 80–20 rules can be, it is a common train-test split ratio within the ML community.

Reference: <https://medium.com/@nahmed3536/the-motivation-for-train-test-split-2b1837f596c3>

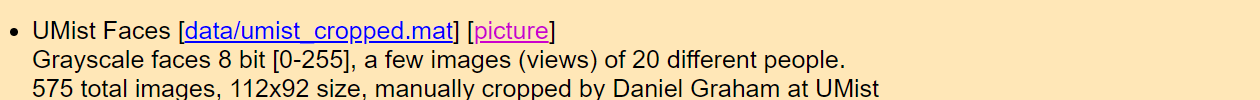
**2. The rationale behind the steps for data pre-processing** 

**1) Standardization (Normalization) using *StandardScaler()***

It calculates the mean and standard deviation of each feature in the training data and then scales (transforms) the features so that they have a mean of 0 and a standard deviation of 1. This process helps in improving the numerical stability of machine learning algorithms and ensures that all features have the same scale.

(1) Consistency in Intensity Values

The dataset has grayscale faces in 8 bit [0-255], 0 (completely black) to 255 (completely white) in an 8-bit image, with values in between representing varying shades of gray. These variations can make it challenging for machine learning algorithms to learn meaningful patterns because they might focus on intensity differences that are unrelated to the content of the images.



<https://cs.nyu.edu/~roweis/data.html>

(2) Numerical Stability

Machine learning algorithms are sensitive to the scale of input features. When you feed raw pixel values into these algorithms without scaling, they may struggle to converge during training or exhibit poor performance. Scaling ensures that the pixel values are within a reasonable range, typically between 0 and 1 or -1 and 1, making the computations more numerically stable.

(3) Computational Efficiency

Many machine learning algorithms, especially neural networks, benefit from having input features that are roughly centered around zero and have similar variances. When the input features are scaled to have similar ranges, it can lead to more efficient training, as the optimization process is less likely to encounter large gradient updates, which can slow down training.

So, our data processing of the data is justified.

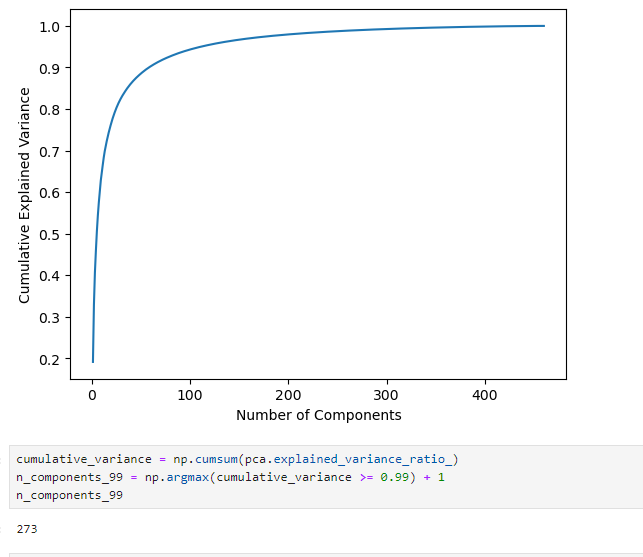
**2) One-Hot encoding using *to\_categorical()***

We have converted the class labels or target values into a one-hot encoded format. One-hot encoding is a representation that turns categorical data (e.g., class labels) into binary vectors, making it easier for machine learning models to work with categorical variables.

**3) Dimensionality Reduction using *PCA***

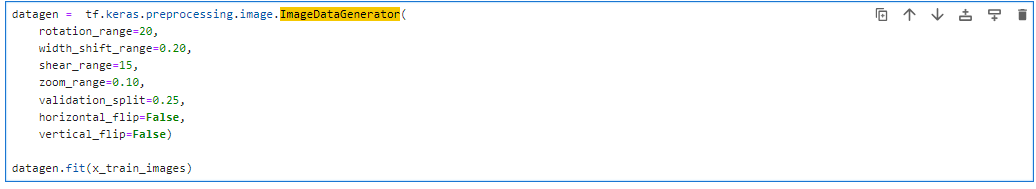
Dimensionality Reduction is executed to reduce the number of features (or dimensions) in a dataset while preserving as much of the original variance or information as possible.

The best value is around 273



**4) ImageDataGenerator**

ImageDataGenerator is used to artificially increase the size and diversity of the training dataset by applying various transformations to the original images.



rotation\_range: It specifies the range (in degrees) of random rotations that can be applied to the images. rotation\_range=20 means that the images can be rotated by up to 20 degrees in either direction.

width\_shift\_range: This parameter controls horizontal shifting of the images as a fraction of the total width. A value of 0.20 means the images can be shifted horizontally by up to 20% of their width.

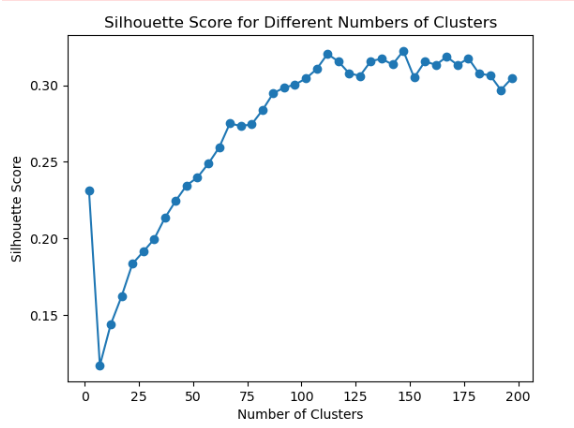
shear\_range: It controls the shear intensity for geometric transformations. A value of 15 indicates that the images may be sheared by up to 15 degrees.

zoom\_range: This parameter specifies the range for random zooming. A value of 0.10 allows zooming in by up to 10%.

validation\_split: It can be used to specify the validation split ratio. In this case, it's set to 0.25, which means that 25% of the augmented data will be used for validation.

horizontal\_flip and vertical\_flip: These parameters control whether horizontal and vertical flips are applied to the images. In this code, both are set to False, meaning no flipping will occur.

**4. The reason why we decided not to use a clustering on the training instances.**



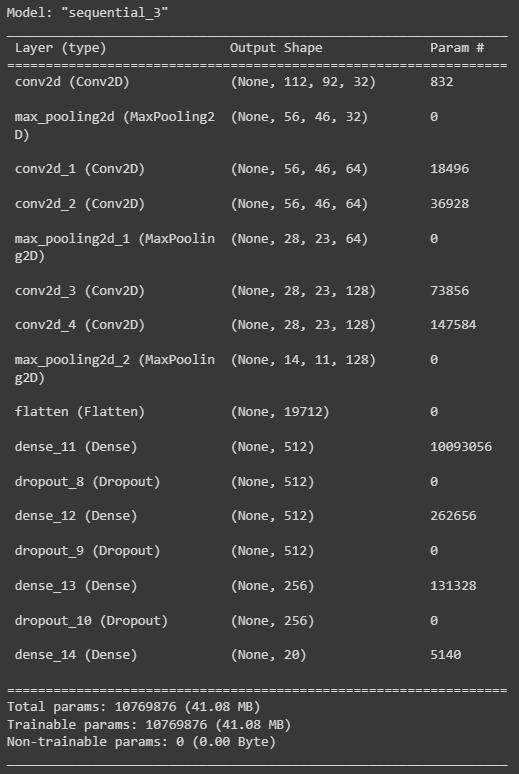
We tried to apply one of the clustering technique, K-means, and evaluated it with Silhouette Score. The problem is the accuracy was not going up and there were too many clusters, n\_clusters = 129, which is not ideal as we know that the dataset has 20 clusters.

Compared with that clustered data, the data without clustering showed better performance (accuracy).  
So, we decided not to use clustering technique for this system.

**5. The architecture our team has selected**

**(Discuss the rationale behind your team’s choice of activation functions, loss function, and how you tuned the hyperparameters of the network model.)**

**1) Summary**



**2) Choice**

Optimization = Adam

Loss function = categorical\_crossentrop

Activation=

"sigmoid" | “relu” | “softmax” for multiclass prediction

**3) Rationale**

**Convolutional Neural Network (CNN):**

CNNs are a suitable choice for image classification tasks like face recognition. They are designed to automatically learn hierarchical features from images, which makes them effective for capturing spatial patterns and textures.

CNN in our system has multiple convolutional layers, followed by max-pooling layers. This is a common architecture for image classification because it helps extract features at different scales while reducing the spatial dimensions.

**Activation Functions:**

"relu" is chosen as the activation function for most of your convolutional layers. ReLU (Rectified Linear Unit) is a popular choice for CNNs because it introduces non-linearity and helps the model learn complex patterns.

"sigmoid" activation is used for some of your dense layers. Sigmoid is often used in binary classification models to produce class probabilities between 0 and 1.

For multiclass classification, "softmax" is a common choice in the final layer, as it provides probabilities for each class.

**Loss Function:**

"categorical\_crossentropy" is a suitable choice for multiclass classification problems, where each sample belongs to one class out of several. Categorical cross-entropy measures the dissimilarity between predicted class probabilities and true class labels, making it an appropriate choice for training the model to predict one class out of 20 possible classes.

**Optimization Algorithm (Adam):**

Adam is a widely used optimization algorithm that combines the benefits of both RMSprop and Momentum. It is known for its efficiency and ability to adapt learning rates for each parameter, making it a suitable choice as an optimizer.

Adam generally works well as a default choice for training deep learning models and often converges faster than other optimizers.

**Dropout Layers:**

Dropout layers are used to reduce overfitting in the model. By randomly dropping a fraction of neurons during training, dropout prevents the model from relying too heavily on specific neurons and encourages more robust learning.

**4) Hyperparmeter**

**We are using “keras tuner”. It simplifies the process of hyperparameter tuning by systematically searching through different combinations of hyperparameter values to identify the configuration that results in the best model performance.**

**6. The results of the trained system**

Test Accuracy: 0.9913

|  |  |
| --- | --- |
| Class 1 Recall: 1.0000  Class 2 Recall: 1.0000  Class 3 Recall: 1.0000  Class 4 Recall: 1.0000  Class 5 Recall: 1.0000  Class 6 Recall: 1.0000  Class 7 Recall: 1.0000  Class 8 Recall: 1.0000  Class 9 Recall: 1.0000  Class 10 Recall: 1.0000 | Class 11 Recall: 1.0000  Class 12 Recall: 1.0000  Class 13 Recall: 1.0000  Class 14 Recall: 1.0000  Class 15 Recall: 0.7500  Class 16 Recall: 1.0000  Class 17 Recall: 1.0000  Class 18 Recall: 1.0000  Class 19 Recall: 1.0000  Class 20 Recall: 1.0000 |

Stratified split is used so that the split training / test dataset have “Balanced Class Distribution” and “Statistical Significance” and avoid bias. However, the dataset is given is too small to achieve the goals. In the dataset, as we know, there are only 20 classes, not big enough for train / test splitting.