DS203 E7 PROJECT

Group 15

-Siddharth Verma[22B2153]
-Vidyanand Kumar [22B2171]
-Kshitij Kumar Pradhan[22B4215]
-Anirudh Gupta[22B4203]

Github repo: https://github.com/SiddharthMaverick/DS_203_Project.git

Problem Description :

Objective: To create a predictive model that utlilises categorization of designs into distinct families based on their shapes. Further,the classification is made the complexity of layouts into Low Complexity, Medium Complexity, and High Complexity, and to develop criteria to determine this complexity, After defining a proper model ,we would use the parameters of height ,width ,complexity and layout area needed and display appropriate layout designs for it

Data: The dataset contains 1183 images that vary in the size and shape of the contour.

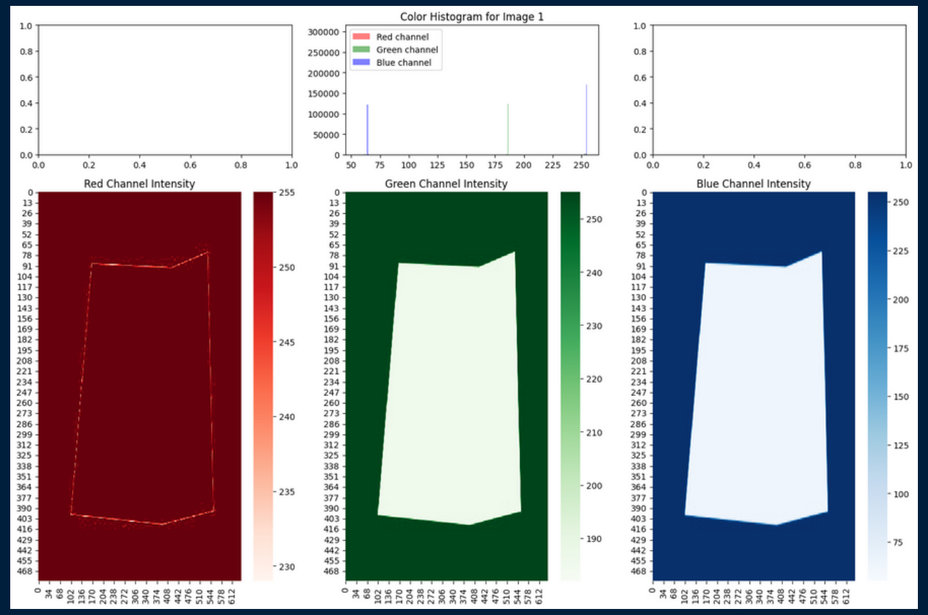
Executive Overview:

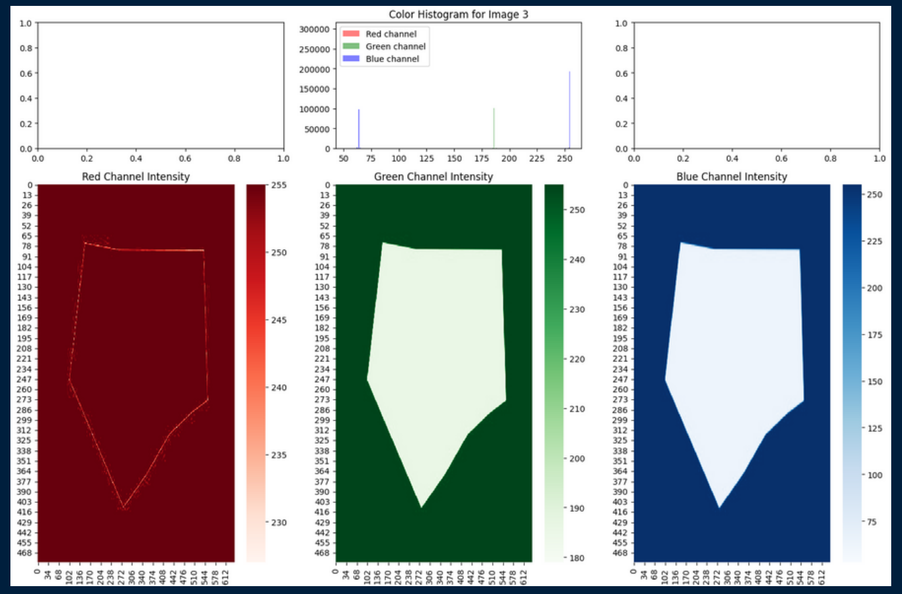
- We have first done an Exploratory Data Analysis of image dataset. In the EDA part we have first done Color channel analysis of image and then we do PCA in which we resized the image and then apply DB-Scan to find the possible clusters of images.

- After performing EDA, we solved the problem using three approaches:
  Approach 1: Unsupervised Complexity Classification using PCA Components and K-means
  Approach 2: By Labelling a Certain amount of Data Set & Mapping it with Efficient Net CNN Model
  Approach 3: Labelling Image Dataset by OpenCV and creating csv of image name, length,width,layout area , number of edges.

Exploratary Data Analysis on Image Dataset

We have 1183 images of 640x480 pixels , we further conduct color channel analysis giving us information about color intensity.

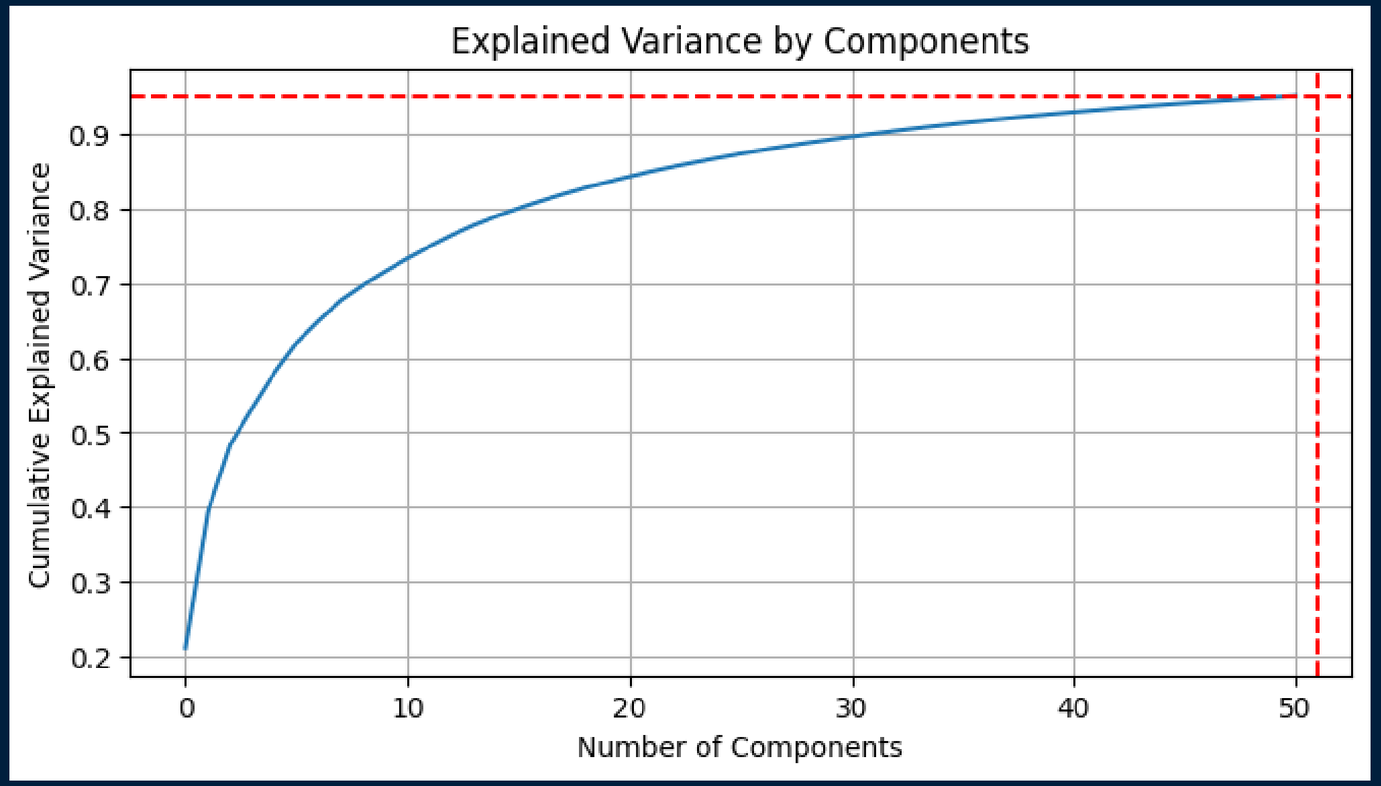




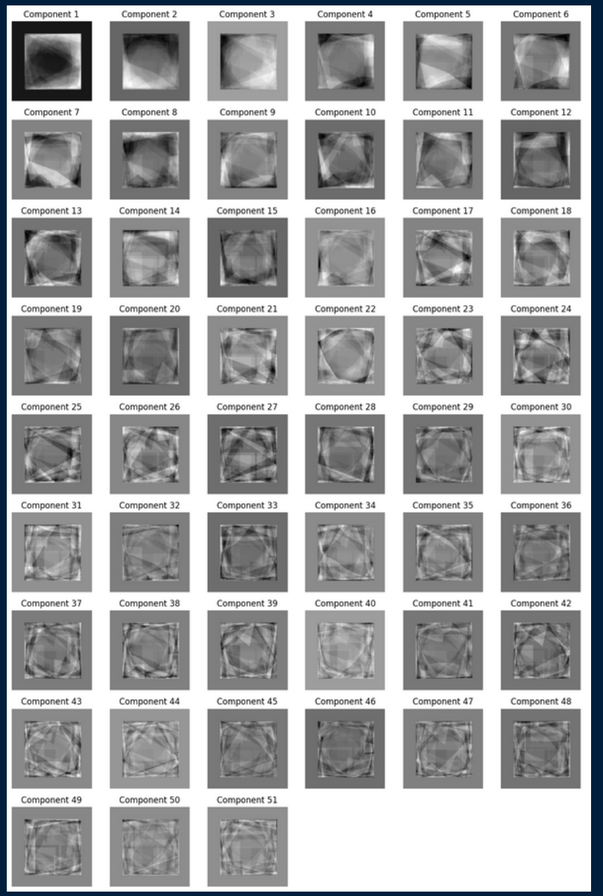
Color channel analysis heatmap & histogram visualization created for EDA

We now conduct PCA analysis on Image Dataset provided.

Note for PCA we are resizing the image to 400x400 then finding number of components for maintaining cumulative variance of 95%

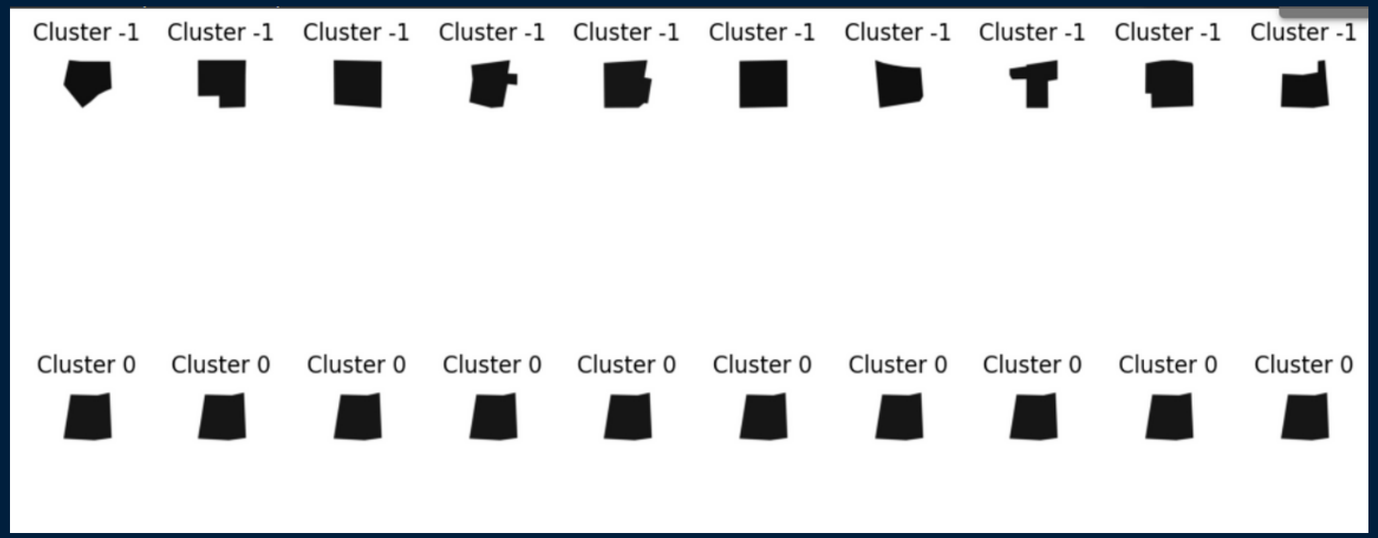


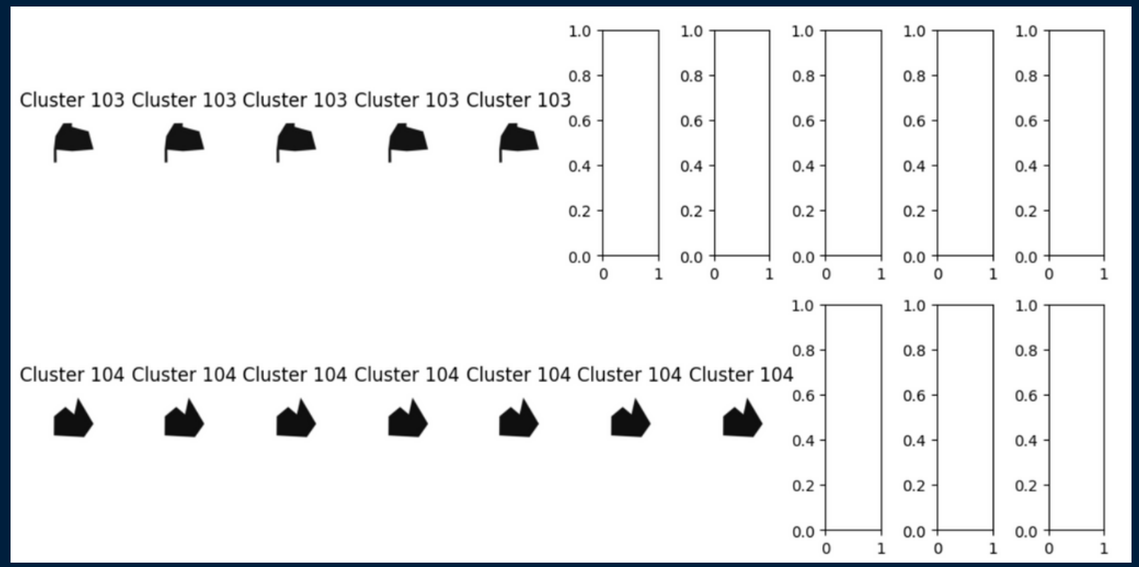
Number of components needed is 51



Visualization of 51 components found on image dataset

We use DB-SCAN to know about the possible clusters we would get ,





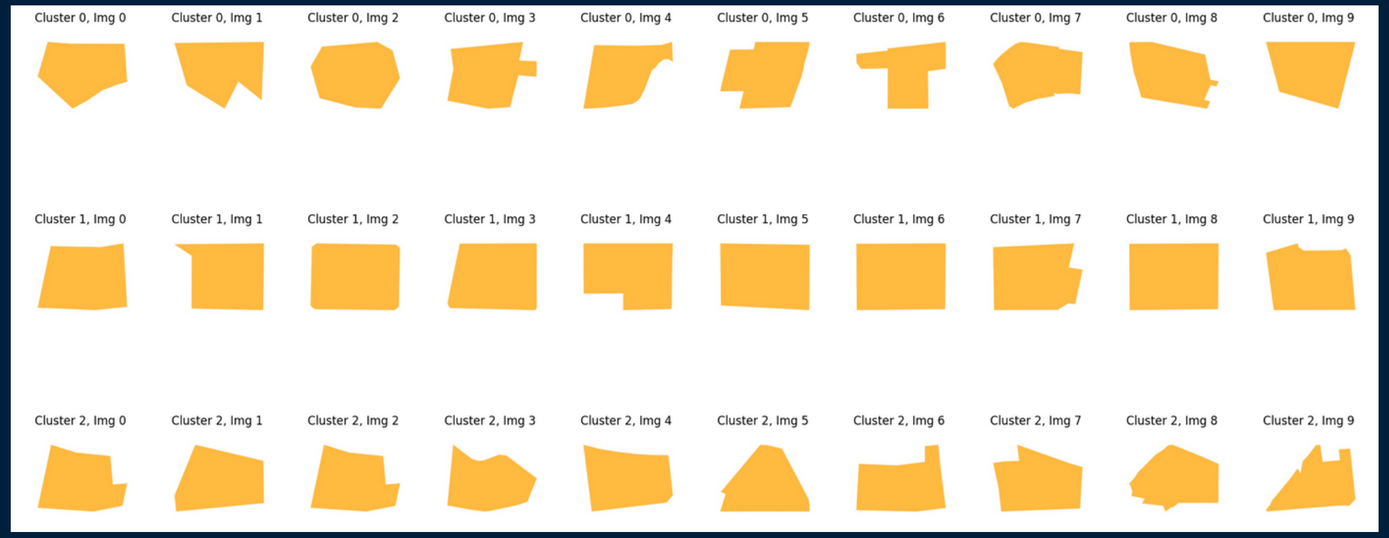
We have found 104 possible clusters . with cluster -1 having images utmost unique ones together i.e. 10 unique ones

Approach 1 : Unsupervised Complexity Classification using PCA Components & K-Means

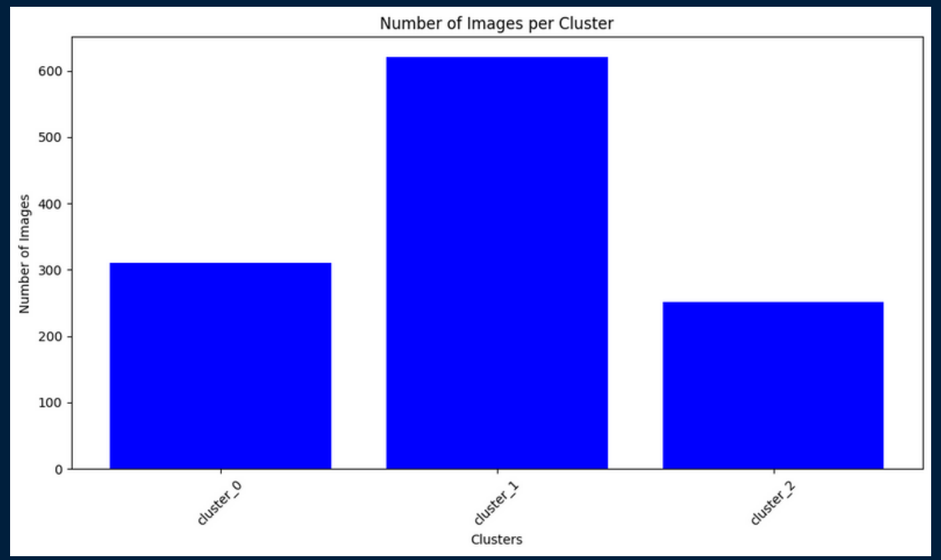
Note: We have here used mathematical definition of complexity rather than any no.shapes or area. Our complexity is based upon a idea that two similar shapes or layout would be closer after PCA transformation.

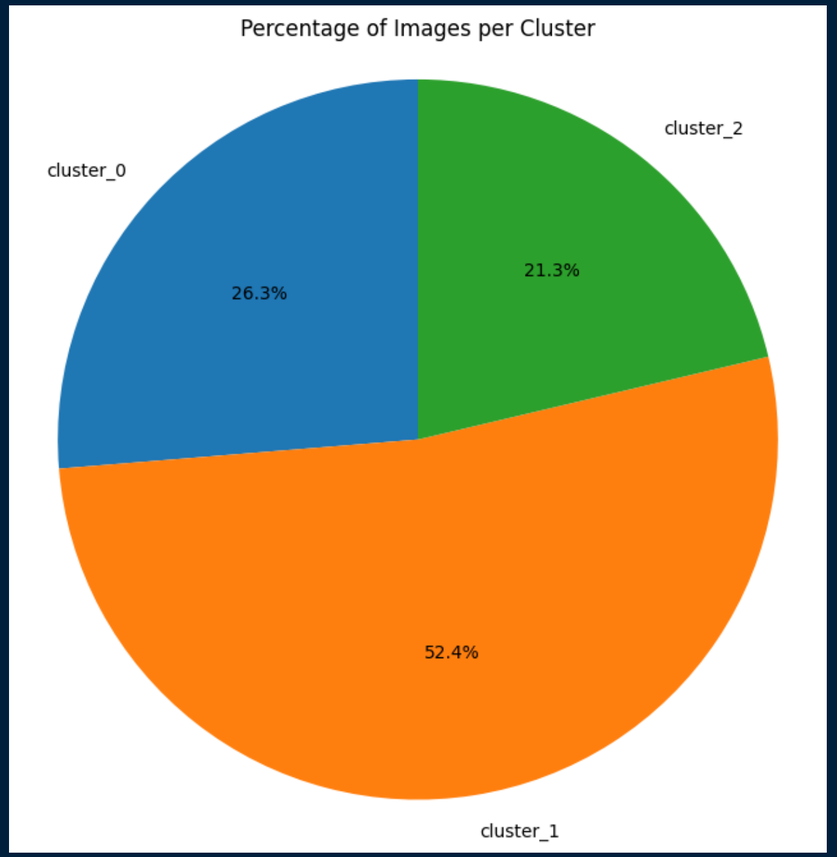
We use the following steps in our 1st Approach:

- We load dataset of images and convert to grayscale and resize them to 400x400 and then finally flatten the images in 1D array.
- We then do PCA-analysis on image dataset by keeping threshold variance of 95% giving 51 components , then transformation is implemented in the perform_pca function.
- Then we cluster the transformed images using K-means algorithm keeping number of clusters as 3 .
- We label each of images with thier cluster name and store them in different folders



Cluster Visualization

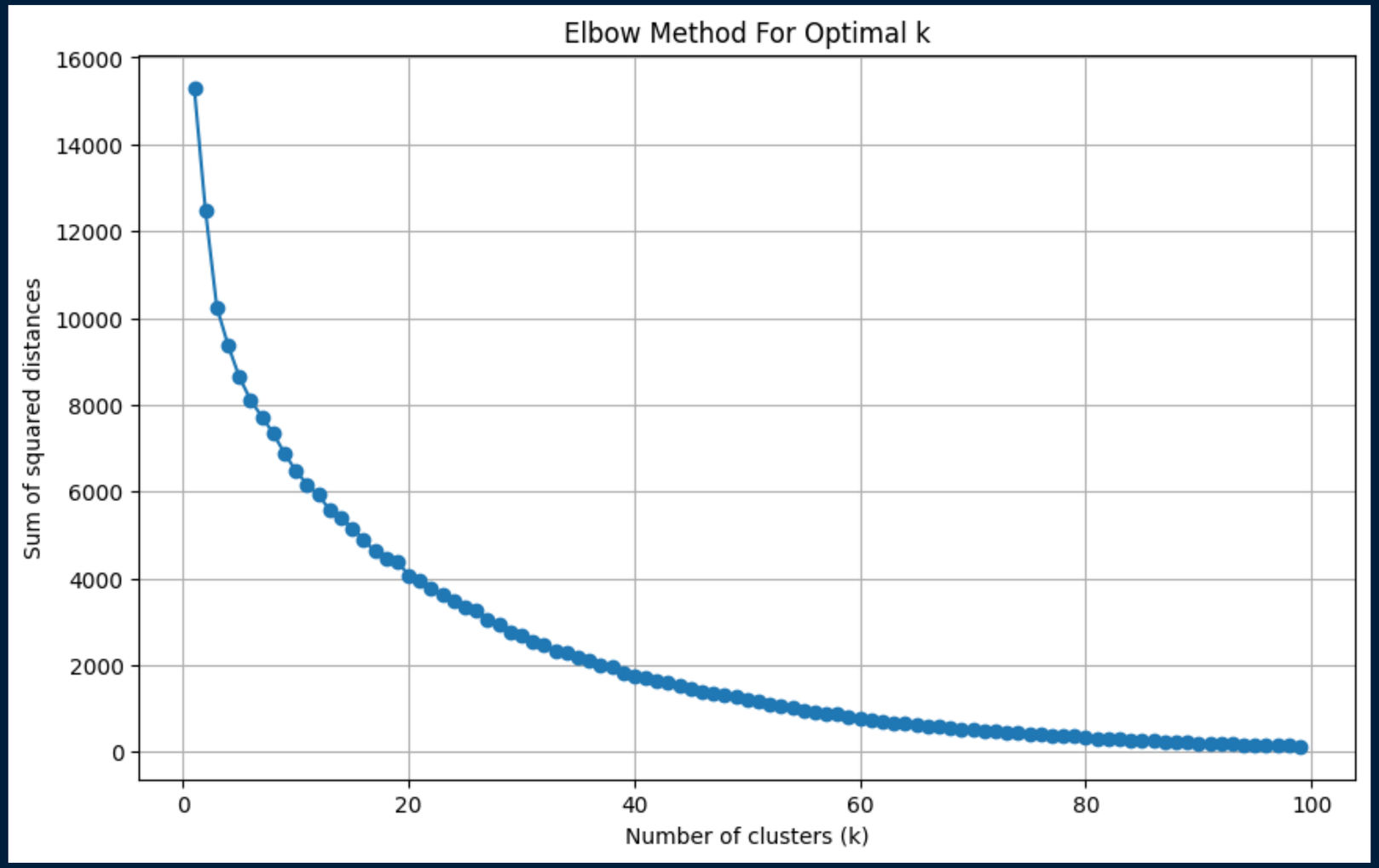




Cluster Percentage

cluster_1 having 52.4%,
cluster_0 having 26.3%
cluster_2 having 21.3%

we have got the following elbow plot for k means while applying by same algorithm .



Note: In DBscan Image analysis we had got 104 clusters, similarly we are getting converging at around 100 in elbow curve, therefore taking K=3 , is not clearly viable when defining complexity via PCA analysis.

Approach 2: By Labelling a Certain amount of Data Set & Mapping it with Efficient Net CNN Model

Manually Labelled Dataset

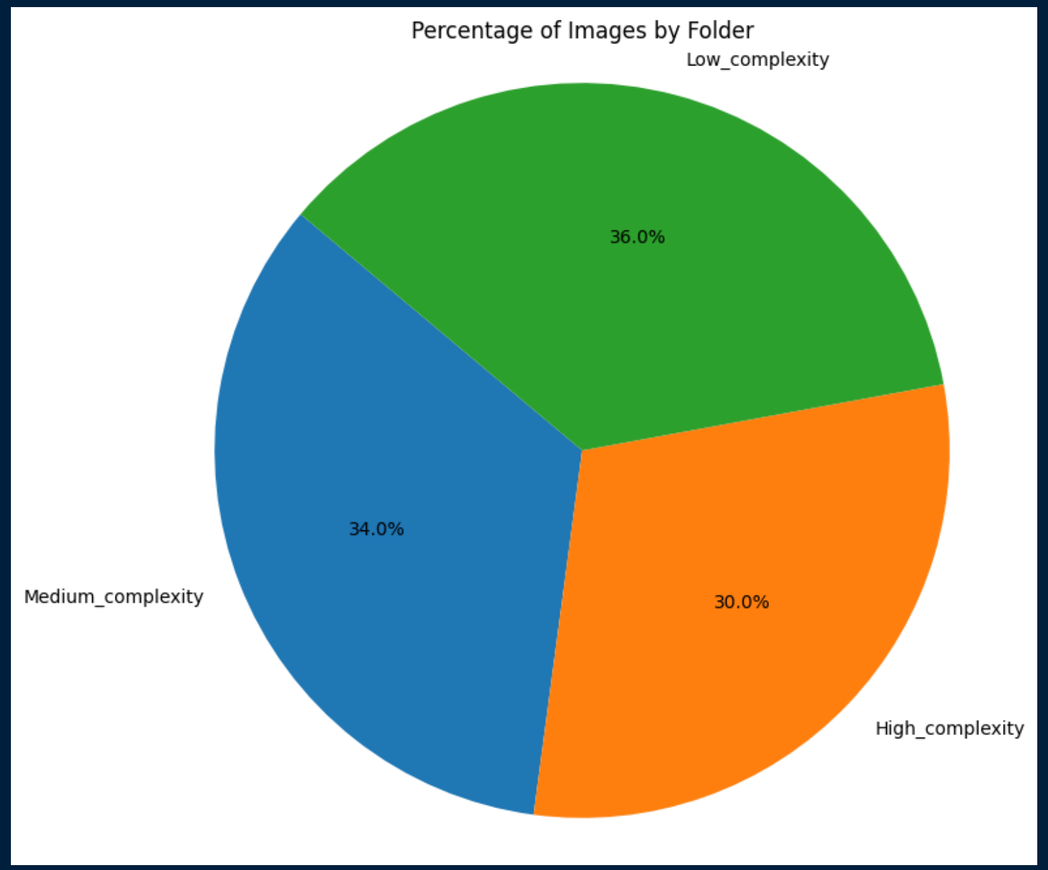


Image Dataset
we created manually by dividing with respect to eye .
We made 161 train images, then 39 validation images using Image Data Generator.

- We have **ImageDataGenerator instance, which is created to automate data preprocessing and augmentation for the training of neural networks. The configurations We have used are as follows:**
  - **Rescaling: Pixel values are scaled to a range of 0 to 1 for normalization.**
  - **Random Rotations: Images will be randomly rotated by up to 40 degrees.**
  - **Shifts: Images will undergo random horizontal and vertical shifts by 20% of their dimensions.**
  - **Shear Transformations: Random shearing by up to 20%.**
  - **Zoom: Random zooming by up to 20%.**
  - **Horizontal Flips: Images may be flipped horizontally.**
  - **Fill Mode: Newly created pixels from transformations will be filled based on the nearest original pixels.**
  - **Validation Split: 20% of the data is reserved for validation.**

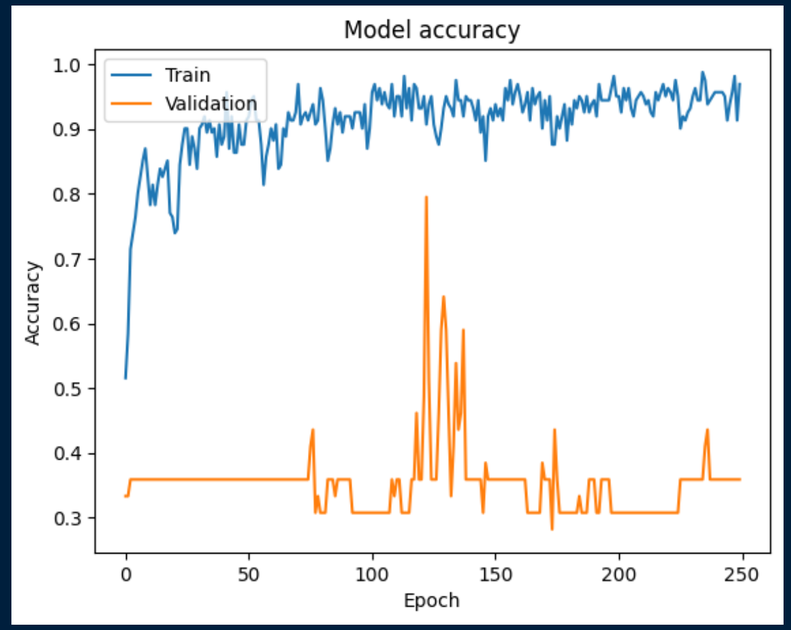
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnetb0 (Functional  (None, 7, 7, 1280)        4049571
 )

 global_average_pooling2d_1  (None, 1280)              0
  (GlobalAveragePooling2D)

 dropout_1 (Dropout)         (None, 1280)              0

 dense_2 (Dense)             (None, 128)               163968

 dense_3 (Dense)             (None, 3)                 387

=================================================================
Total params: 4213926 (16.07 MB)
Trainable params: 4171903 (15.91 MB)
Non-trainable params: 42023 (164.16 KB)
```

Summary of Efficient Model we are using for Training on our manually created dataset

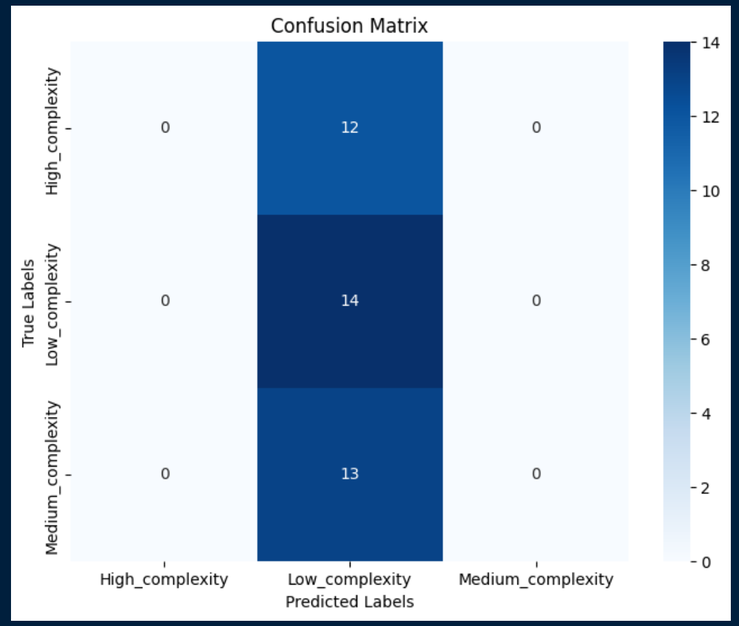
Model Configuration:

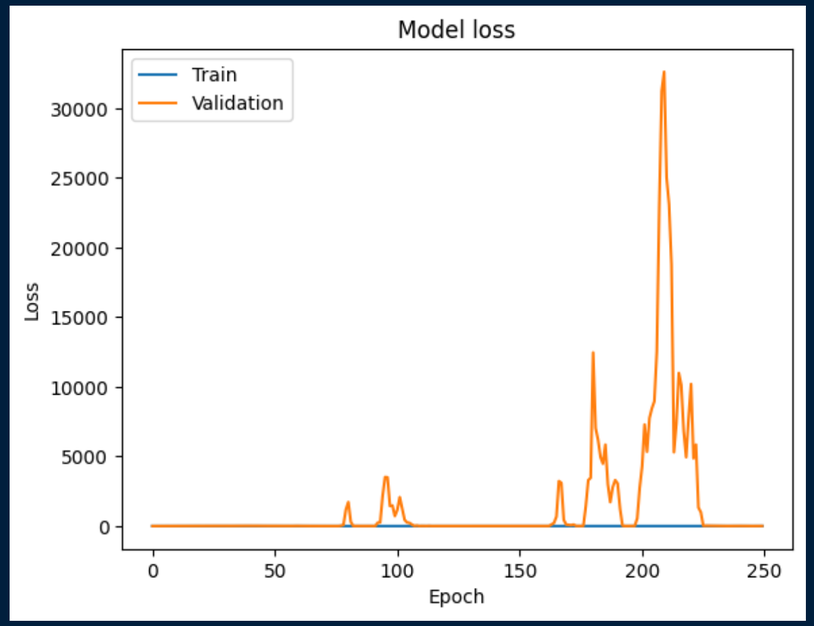
- Trainability: The entire base model is set to be trainable, implying that the weights in the pre-trained EfficientNetB0 will be updated during training on the new dataset, allowing the model to potentially learn more detailed features specific to the new task.
- Things we have added to base model :
  - GlobalAveragePooling2D: Used to reduce the spatial dimensions (height and width) of the output from the base model to a single 128-dimensional vector per image. This helps in reducing the model size and computational cost.
  - Dropout: A dropout layer with a dropout rate of 20% is included to help prevent overfitting by randomly setting a fraction of the input units to 0 at each update during training time.
  - Dense Layers: Two dense layers are added where the first dense layer has 128 units and uses ReLU activation, serving as a fully connected layer that processes features extracted by the base model. The final dense layer has 3 units with a softmax activation function, assuming the task is to classify each image into one of three categories.

Model Accuracy & Loss and Confusion Matrix



Although intially we thought this would be a great idea but the Inference we are getting from confusion matrix and loss plot is that our model is very unpredicitive and is not able to capture the features effectively , therefore we are not proceeding further with this model & approach





Approach 3:Labelling Image Dataset by OpenCV and creating csv of image name, length,width,layout area , number of edges.

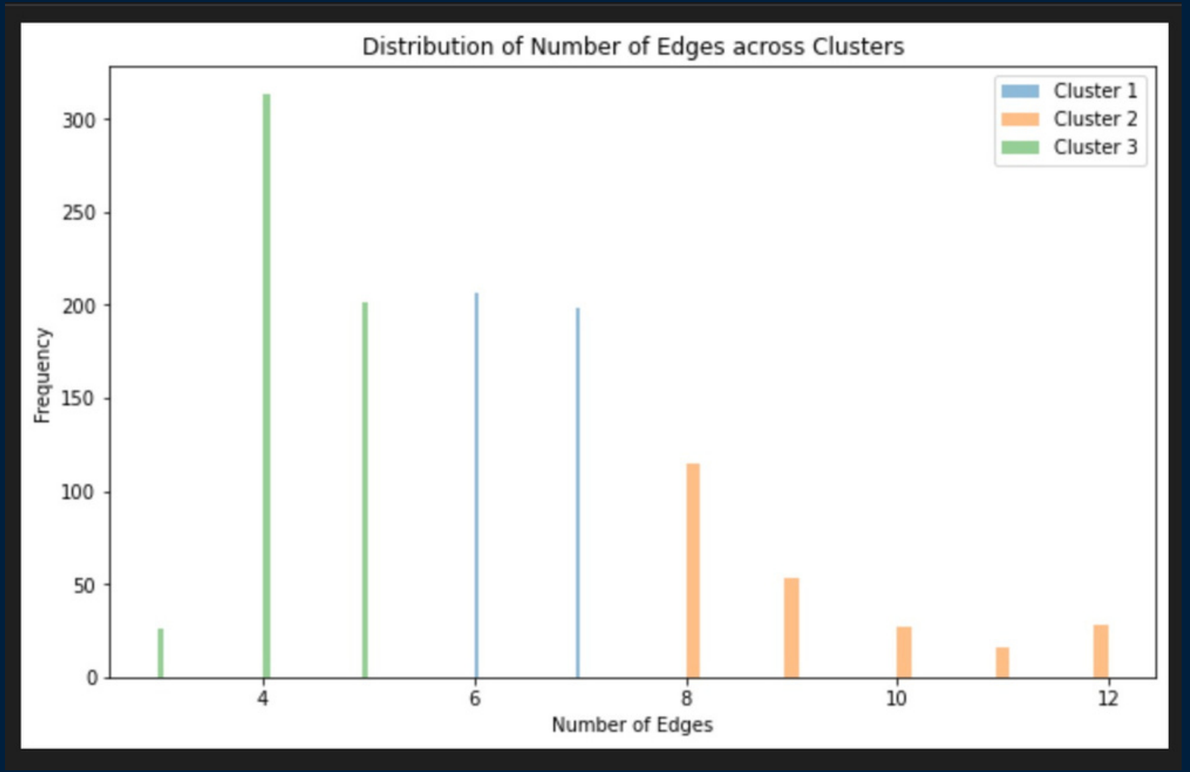
This is the first 10 data points of the Csv file that we obtained

Image Name	Layout Area	Number of Edges	Width	Height	Cluster	Cluster Name
p_0001.jpg	133402	6	446.071743	338.05325	0	Cluster_1
p_0002.jpg	131058.5	5	336	451	2	Cluster_3
p_0003.jpg	104190	6	452.305207	320.301421	0	Cluster_1
p_0004.jpg	100543.5	7	459.393078	318.265612	0	Cluster_1
p_0005.jpg	145848.5	5	330.037877	455.039559	2	Cluster_3
p_0006.jpg	131375.5	7	335.538373	439.711269	0	Cluster_1
p_0007.jpg	136179.5	7	451	336	0	Cluster_1
p_0008.jpg	104661	8	311.194473	461.277574	1	Cluster_2
p_0009.jpg	106586	8	316.332104	454.942854	1	Cluster_2
p_0010.jpg	127351.5	7	438.603466	332.487594	0	Cluster_1

We have defined complexity into three parts by using K means on number of edges here

Plot we get on applying kmeans clustering on number of edges

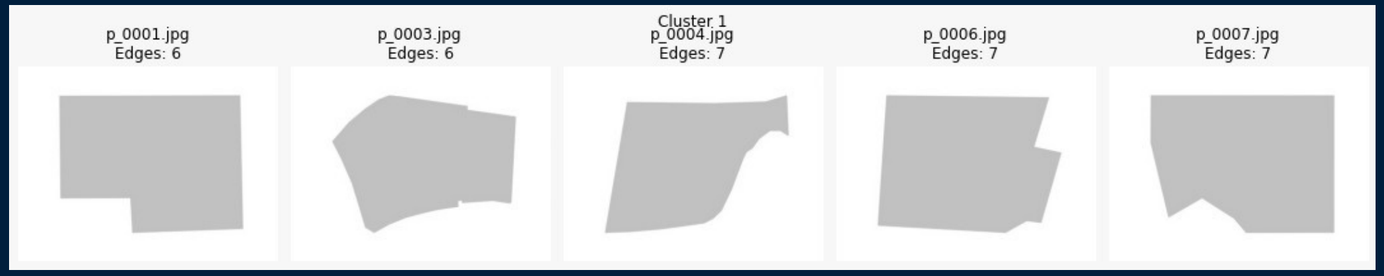
Low complexity: upto 5 edges
Medium complexity :6-8 edges
High complexity: greater than 8



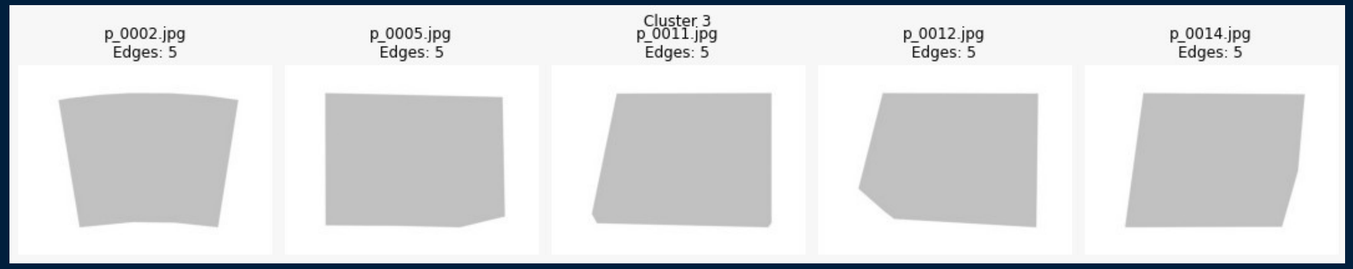
Visualization Clusters we have created

Cluster 2: High Complexity

Cluster 1: Medium Complexity

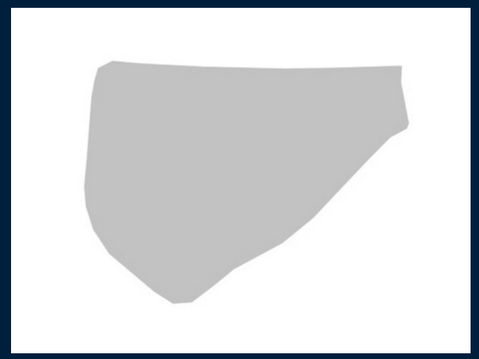


Cluster 3: Low Complexity



Now, we have trained the model in such a way that if we input the width and height of an image from any cluster among the three available clusters, the model will provide 5 similar images from the same cluster of images

For example, if we give this input as width = 336,Height =449, and complexity=”High”
We get 5 similar images as displayed below


Learnings:

- Loading image data involved importing and preparing digital images for analysis and processing within neural networks.
- Neural network was used in image processing to enhance, classify,and generate visual content.
- Image processing metrics and mathematical calculations provided quantitative measures to evaluate and optimize the quality and performance of image algorithms.
- Manipulating gray scale and color in images allows for tight-fitting box plotting and edge detection.

Hurdles:

1. Coding for Various Algorithms of Image Processing: Implementing techniques like image filtering, morphological operations, and color space conversion to manipulate and analyze image features.
2. Debugging the Codes Which Provided Error: Employing robust error handling and code optimization to detect and rectify errors,improving the alogrithmic performance.
3. Selecting the Better Models for the Project: Evaluating and comparing machine learning and deep learning models to identify the optimal model for the given task.
4. Additional Challenges and Solutions: Utilizing data augmentation to diversify the dataset and enhancing model interpretability through feature analysis, visualization, and explanation method

Thank You