Clustering Model for OKU Timur Script Images

Liu Toriko[1], Susan Dian Purnamasari[2]*, Yesi Novaria Kunang[3], Ilman Zuhri Yadi[4], Andri[5]
Faculty of Science and Technology[1], [2], [3], [4], [5]
Universitas Bina Darma
Palembang, Sumatera Selatan, Indonesia
liutoriko031102@gmail.com[1], susandian@binadarma.ac.id[2], yesinovariakunang@binadarma.ac.id[3], ilmanzuhriyadi@binadarma.ac.id[4], andri@binadarma.ac.id[5]

***Abstract*— The OKU Timur is a regency located in South Sumatra Province. In the OKU Timur region there are many historical heritage sites, one of which is the script. In general, script is a system of symbols for writing language. The OKU Timur script is a writing system that is usually used by the local community. This writing system is characterized by its unique characters and has high historical and aesthetic value for the local community. The OKU Timur script is used in daily communication, traditional ceremonies, historical documents, and various other cultural contexts. This research aims to develop a clustering model that is used to efficiently and accurately group Of OKU Timur script images based on certain characteristics. By using techniques in the field of clustering such as the K-Means algorithm this model is developed so that the clustering of OKU Timur script images is made automatically in order to save time and effort. The study employs the K-Means algorithm to divide the data into several clusters, grouping data with similar characteristics into one cluster and data with different characteristics into another. This research is also expected to contribute to preserving digital culture so that the development of OKU Timur characters can be passed on to future generations.**

Keywords*— *OKU Timur Script, Clustering, K-Means

I. Introduction

East OKU Regency is located in South Sumatra Province and borders Ogan Ilir Regency, Ogan Komering Ulu Regency, Ogan Komering Ulu Selatan Regency, and Lampung Province. In East OKU, there are many historical sites, one of which is Aksara. Aksara refers to letters or scripts that function as symbols of sound (phonemes). Aksara is also known as a "writing system." Over time, aksara has come to mean a system of visual symbols that appear on various media, such as paper, stone, trees, wood, or fabric, to convey expressive elements of a language [1]. Currently, the development of information technology is progressing rapidly and encompasses many aspects of various fields of life. This advancement results in the availability of vast and diverse data, covering industries, economics, science, technology, and various other areas of life [2]. One of the technologies that can make life easier for humans is clustering

Clustering is the process of grouping data into various clusters where similar objects are placed in one cluster, while dissimilar objects are placed in different clusters. Each cluster contains data that is as similar as possible to each other, and the degree of similarity is usually measured based on distance. Therefore, each object in one cluster must have similar characteristics, while objects in other clusters must have different characteristics. This definition assumes the existence of several important parameters that represent similarities or differences between clusters [3].

The goal of this clustering process is to organize the data into several groups, so that similar data is placed in one group while different data is placed in another group [4]. The basis of the clustering concept is to group a number of objects into clusters, with the aim of forming groups that have high similarity among the objects within them and significant differences from the objects in other groups [5].

The clustering algorithm that is currently developing is K-Means. K-Means is a non-hierarchical clustering method that aims to divide data into several clusters, where data with similar characteristics are grouped together, while data with different characteristics are placed in separate clusters. The K-Means algorithm falls under the category of partitioning clustering, which separates data into distinct parts. The K-Means algorithm is very popular due to its ability to cluster large amounts of data and outliers very quickly [6]. K-Means has relatively low computational complexity, making it fast to apply to large datasets [7].

Until now, there has been no research that specifically focuses on the development of a clustering model for OKU Timur script images. The absence of prior research presents a unique challenge in finding suitable information for the development of this model. The development of this clustering model is necessary to make the system more efficient and accurate in grouping OKU Timur script images and to preserve the digital culture of using the OKU Timur script for future generations. Based on the existing issues, the researcher is conducting a study titled "Clustering Model for OKU Timur Script Image"

II. Research Methodology

In the development of the clustering model for OKU Timur script images, it will certainly be explained and developed in the form of a flowchart, which is a diagram that clearly and structurally illustrates the steps of a process.

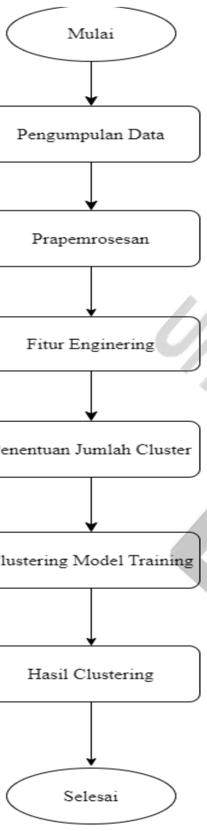


Figure 1. Flowchart

A. *Data Collection*

The data on the OKU Timur script was obtained from the Traditional Leader and Script Expert of OKU Timur. The data collection was previously conducted by another team from the ISRG (Intelligent System Research Group), and after the data was collected, there were 228 characters of the OKU Timur script. Before performing clustering, several processes must be carried out, such as preprocessing.

B. *Preprocessing*

In the preprocessing stage conducted on the raw data, the goal is to improve data quality so that clustering can be performed more effectively and yield good groups. The processes involved include downloading and extracting data, then saving, printing the path to the directory, and organizing, checking, and processing the list of image files to prepare them for the K-Means model training.

C. *Feature Engineering*

Feature engineering is the process of creating and transforming additional features from raw data to improve model performance, such as VGG16. In the context of VGG16, feature engineering involves extracting features from the image representations processed by the model, as well as adding new features or transforming existing ones to enhance the model's ability to recognize patterns or classify data.

D. *Determining the Number of Cluster*

Next, the stage of determining the number of clusters is conducted, as the researchers identified approximately 228 characters of the OKU Timur script, with 102 respondents collected. These 228 characters are grouped into 228 clusters, each representing a specific characteristic of the OKU Timur script. The 228 clusters are derived from 19 letters multiplied by 12 punctuation marks. This is the most critical process in clustering, as it must determine the optimal number of clusters to efficiently and accurately separate the data.

E. *Clustering Model Training*

After obtaining the data and determining the number of clusters, the next step is to train the clustering model. This training is necessary to test which model is suitable for clustering the OKU Timur script images, as there are several models available, such as K-Means, DBSCAN (Density-Based Spatial Clustering of Applications with Noise), and Agglomerative Hierarchical Clustering.

The model suitable for clustering the OKU Timur script images is K-Means because it is simple, fast, efficient, and flexible in determining the number of clusters. In the Clustering Model Training, a K-Means model is created with the number of clusters specified by 'number_cluster' and 'random_state=0' for consistent results. The K-Means model is then trained on the image features stored in 'featurelist', which have been converted into a NumPy array. The purpose of this is to group images based on character similarity into the predetermined clusters.

F. *Clustering Result*

This stage involves analyzing the clustering results, observing how the data is grouped, and using the resulting dataset to train deep learning and transfer learning models for classifying the OKU Timur script images.

III. Result and Discussion

A. *Data Collection*

At this stage, the data collection process for the OKU Timur script has involved local traditional leaders and script experts, covering information about the usage and characteristics of the script. Previously, the ISRG (Intelligent System Research Group) team had collected preliminary data regarding the OKU Timur script, including its use in cultural contexts and technical aspects.

Additional data from the traditional leaders and script experts aims to deepen the existing understanding. The obtained data is then processed into a questionnaire to be distributed to respondents. The image below is an example of the OKU Timur script questionnaire that will be filled out by respondents, consisting of 12 punctuation marks of the OKU Timur script.

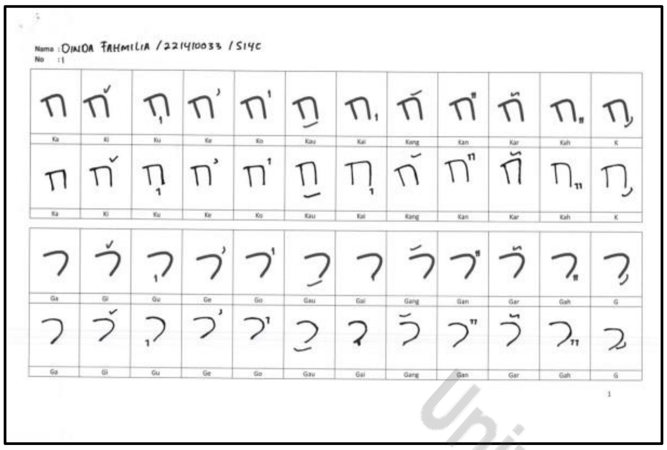


Figure 2. Example Questionnaire for the OKU Timur Script

After being filled out by the respondents, the questionnaire aims to identify the uniqueness of each respondent's writing. This questionnaire contains 228 characters of the OKU Timur script and was completed by 102 respondents. Below is an example of the filled questionnaire from the respondents.

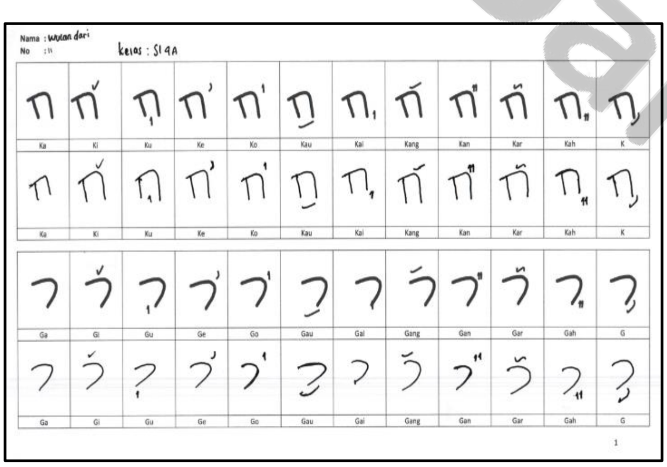


Figure 3. Filled Questionnaire Example

B. Preprocessing

Preprocessing is an important step in data clustering. In this study, 10 sample questionnaire forms were used, with a total of 228 character image types, resulting in a total of 2280 images collected. The number of images taken was then inputted as 2280 images. However, during preprocessing, this was reduced to 2031 images, which were then divided into 228 classes. This process is conducted to prepare the data for use by the model. Several stages are performed during preprocessing, including :

1. Data Downloading and Extraction

```
!wget
"https://www.dropbox.com/scl/fi/rrcu64k0ej6o71zp0
ftku/aksara_okut.zip?rlkey=k6quer7g9guem8uyabj73y
xur&st=4y5ktmce&dl=0"-O aksara_okut.zip
```

This research begins by downloading data files from an external URL using the 'wget' command. This command allows the system to automatically download files from the internet and save them in a specified location; in this case, the downloaded file is aksara_okut.zip

```
import os
import zipfile
local_zip = '/content/aksara_okut.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('/content/')
zip_ref.close()
```

Next, the file extraction process is carried out using the zipfile module in Python. This module is used to read and extract content from ZIP files. The extraction process involves opening the downloaded file, after which all contents are extracted to a specified directory ('/content/'). After that, to ensure the extraction is successful, the program will display a list of files extracted from that folder.

2. Saving and Printing Path to Directory

```
import os
# Path ke direktori tempat gambar-gambar
diekstrak
imdir = '/content/aksara_okut'
targetdir = '/content/output'
print("Path ke direktori gambar:", imdir)
```

This process aims to save and print images in a directory that will later be used in processing. In the initial stage, the task is to save the path to the directory containing the images to be extracted ('/content/aksara_okut'). Then, the path to the destination directory where the extracted images will be saved is also stored ('/content/output'). Once the data has been saved, the next step is to print a text indicating the path to the image directory stored in the variable ('imdir'). The output will display the message ("Path to image directory:") followed by the value of the path in the variable ('imdir'). By defining this path, the code prepares the necessary directory structure for subsequent operations in the analysis.

3. Compiling List of Image Files

```
import glob
import os

# Loop over files and get features
# Menyusun daftar file dengan berbagai format
gambar
filelist = glob.glob(os.path.join(imdir,
'**/*.png'), recursive=True) + \
           glob.glob(os.path.join(imdir,
'**/*.jpg'), recursive=True) + \
             glob.glob(os.path.join(imdir,
'**/*.jpeg'), recursive=True) + \
            glob.glob(os.path.join(imdir,
'**/*.bmp'), recursive=True) + \
            glob.glob(os.path.join(imdir,
'**/*.gif'), recursive=True)
filelist.sort()
featurelist = []
labels = []
```

At this stage, the task involves initializing variables for the image directory, which then creates an empty list to store the file names. In this case, the glob function is used to search for all image files in PNG, JPG, JPEG, BMP, and GIF formats within the directory and its subdirectories recursively, adding the results to the file list. This process aims to collect all image files from a specific directory to prepare complete data for analysis.

4. Checking the List of Image Files

```
# Debugging: Periksa apakah filelist kosong
if len(filelist) == 0:
    print(f"No image files found in the directory
{imdir}.")
else:
    print(f"Jumlah file ditemukan:
{len(filelist)}")
```

The next step is to check if the list of image files is empty. If ('filelist') is empty, the code prints a message stating that no image files were found in the directory ('imdir'). If there are image files in ('filelist'), the code prints the number of files found. Based on the process, 2031 image files were found out of 2280 that were input.

This occurred due to issues while processing images, such as corrupted files, unsupported file formats, or problems with the file path that caused some images to be inaccessible. If images cannot be accessed or processed correctly, they may not be included in ('featurelist') and ('labels'), thus reducing the total number of successfully processed images. Additionally, errors in filtering image formats can also cause some images to be missed.

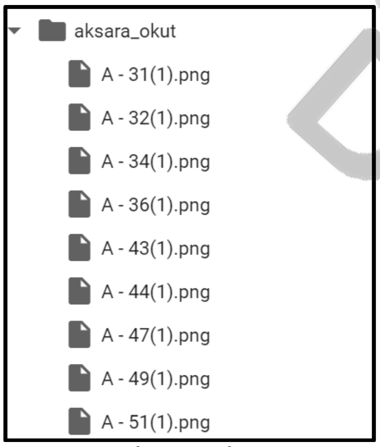
5. Processing List of Image Files

```
import glob
import os
from keras.preprocessing import image
from keras.applications.vgg16 import
preprocess_input
import numpy as np
```

```
# Loop over files and get features
# Menyusun daftar file dengan berbagai format gambar
imdir = '/content/aksara_okut'  # Ganti dengan jalur
direktori sebenarnya
filelist = glob.glob(os.path.join(imdir,
'**/*.png'), recursive=True) + \
           glob.glob(os.path.join(imdir,
'**/*.jpg'), recursive=True) + \
           glob.glob(os.path.join(imdir,
'**/*.jpeg'), recursive=True) + \
           glob.glob(os.path.join(imdir,
'**/*.bmp'), recursive=True) + \
           glob.glob(os.path.join(imdir,
'**/*.gif'), recursive=True)
filelist.sort()
featurelist = []
labels = []

for i, imagepath in enumerate(filelist):
    print("Status: %s / %s" %(i+1, len(filelist)),
end="\r")
    # Periksa apakah file ada sebelum mencoba
memuatnya
    if not os.path.exists(imagepath):
        print(f"Peringatan: File tidak ditemukan:
{imagepath}")
        continue  # Lewati ke file berikutnya

    img = image.load_img(imagepath,
target_size=(224, 224))
    imagepath = imagepath.replace("\\", "/")
    print(imagepath)

    # Use a more robust way to extract the label
assuming it's the directory name
    label =
os.path.basename(os.path.dirname(imagepath))
    labels.append(label)

    img_data = image.img_to_array(img)
    img_data = np.expand_dims(img_data, axis=0)
    img_data = preprocess_input(img_data)
    features = np.array(model.predict(img_data))
    featurelist.append(features.flatten())

print("Labels - ", labels)
```

In the next stage, the initialization of the image directory path is performed, collecting all image files from various formats using glob, and then sorting the file list. In this case, two empty lists are prepared to store image features and labels. After that, each file in the 'filelist' is iterated, loading the images and resizing them to (224, 224). Once the images are loaded at the specified size, labels are extracted from the directory names where the images are located.

Next, the images are converted into arrays and preprocessed to prepare them for use in the model for predicting image features. The prediction results are then stored in the feature list. At the end of the process, the collected label list is displayed. This stage overall prepares the features for further analysis in the context of pattern recognition. The final preprocessing step involves obtaining the resulting images saved in the folder ('aksara_okut'), as shown in the image below.



```
aksara_okut
    A - 31(1).png
    A - 32(1).png
    A - 34(1).png
    A - 36(1).png
    A - 43(1).png
    A - 44(1).png
    A - 47(1).png
    A - 49(1).png
    A - 51(1).png
```

Figure 4. List of Images After Preprocessing

C. *Feature Engineering*

Feature engineering is the process of creating and transforming additional features from raw data to improve model performance, such as VGG16. In the context of VGG16, feature engineering involves extracting features from the image representations already processed by the model, as well as adding new features or transforming existing features to enhance the model's ability to recognize patterns or classifications. The processes involved in feature engineering are divided into two, as follows :

1. Importing Modules in Using the VGG16 Model

```
from keras.preprocessing import image
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import
preprocess_input
import numpy as np
from sklearn.cluster import KMeans
import os, shutil, glob, os.path
from PIL import Image as pil_image
image.LOAD_TRUNCATED_IMAGES = True
```

This process is carried out to import various modules for image processing, utilizing the VGG16 model for feature extraction, performing clustering using the K-Means algorithm, managing images, and allowing for cropped image processing.

2. Creating the VGG16 Model

```
model = VGG16(weights='imagenet',
include_top=False)
```

At this stage, the task is to create the VGG16 model without the final classification layer, so it can be used for feature extraction from images, such as clustering. In this process, there will be no output displayed because the VGG16 model here serves only as an additional feature to ensure the code runs smoothly without errors.

D. *Determining the Number of Cluster*

The stage of determining the number of clusters involves researchers knowing there are approximately 228 characters of the OKU Timur script, with 102 respondents collected. This is the most crucial process in clustering, as it must determine the optimal number of clusters to efficiently and accurately separate the data.

```
# Variables
number_clusters = 228
```

This code defines a variable named (number_clusters) and assigns it the value ('228'). This variable is used to store the number of clusters that will be used in a clustering algorithm, such as KMeans. The output generated from (number_clusters) with the value ('228') will create 228 class files of images that have been processed successfully.

E. *Clustering Model Training*

In this process, model training for clustering is carried out. This training must be performed to test the model suitable for clustering images of the OKU Timur script, and the model used in this process is K-Means.

1. K-Means Model Training

```
# Clustering
kmeans = KMeans(n_clusters=number_clusters,
random_state=0).fit(np.array(featurelist))
```

This code is used to create a K-Means model with the number of clusters specified by 'number_cluster' and 'random_state=0' for consistent results. It also trains the K-Means model on the image features stored in 'featurelist', which have been converted into a NumPy array. The purpose of doing this is to group images based on character similarity into the specified clusters.

2. Creating New Directory

```
try:
    os.makedirs(targetdir)
```

```
except OSError:
    pass
```

This code is used to create a new directory specified by 'targetdir'. The purpose is to ensure the target directory exists without causing an error if the directory already exists.

3. Grouping Images into Subdirectories

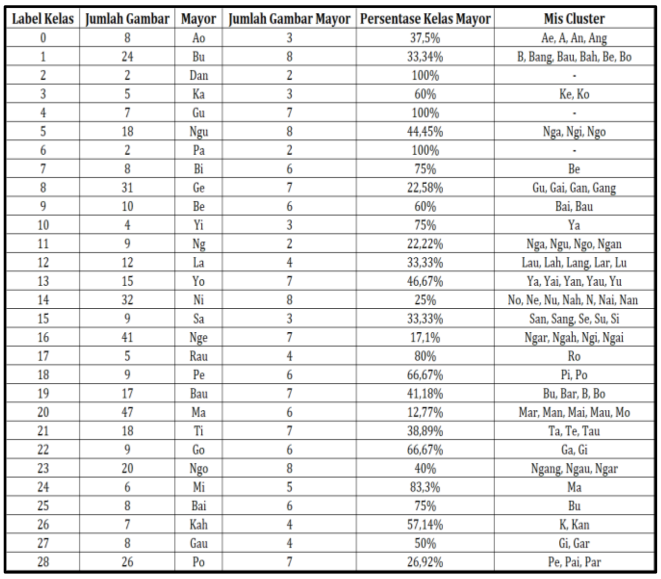
```
print("\n")
for i, m in enumerate(kmeans.labels_):
    try:
        os.makedirs(targetdir+'//'+str(m))
        #print(m)
    except OSError:
        pass
    print("Copy: %s / %s" %(i+1,
len(kmeans.labels_)), end="\r")
    shutil.copy(filelist[i], targetdir +'/'+
str(m) + "/" + str(i) + ".jpg")
```

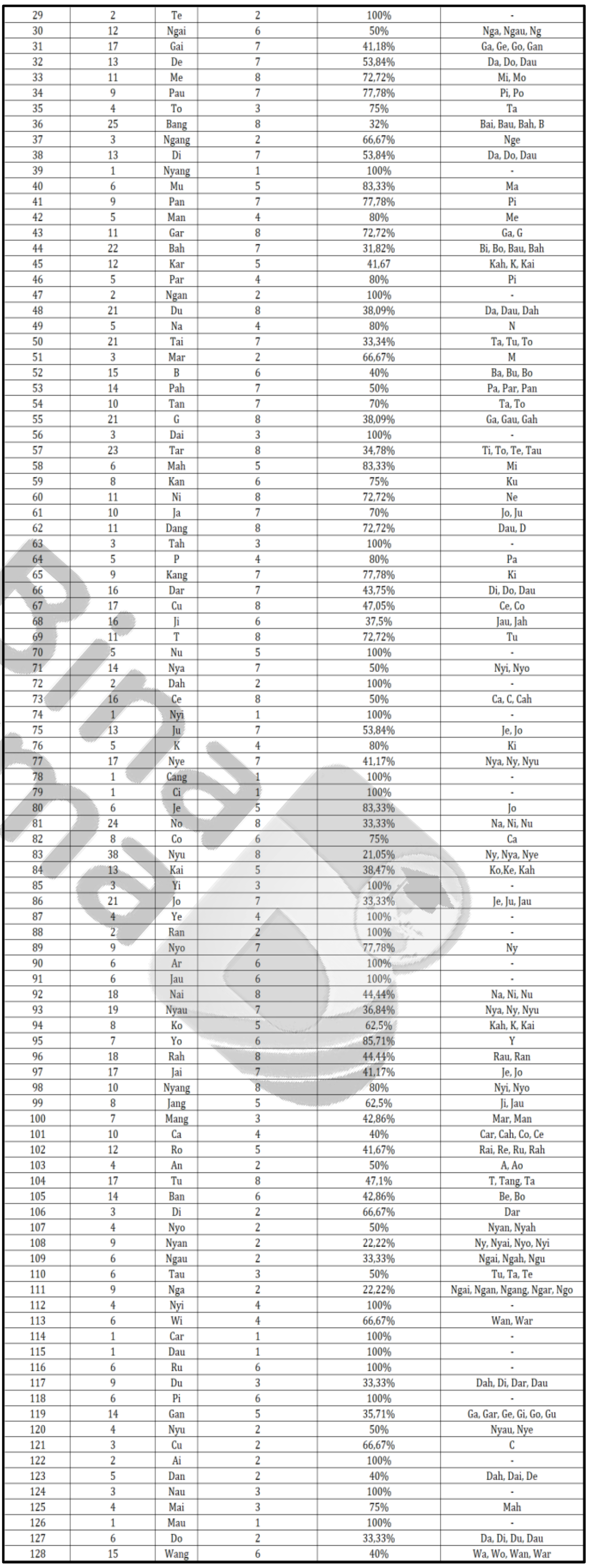
This code is used to group and store images into subdirectories based on the clustering results produced by the K-Means model.

F. *Clustering Result*

After performing clustering, the next step is to examine the results. The visual representation of images in each cluster provides strong visual confirmation of how these images are grouped by the model. This can help in understanding and validating whether the clustering has successfully separated the characters of the OKU Timur script into the predetermined groups. The clustering results of 228 classes are presented in the following table :

TABLE 1. CLUSTERING RESULTS IN 'OUTPUT' DIRECTORY





Label Kelas	Jumlah Gambar	Mayor	Jumlah Gambar Mayor	Persentase Kelas Mayor	Mis Cluster
0	8	Ao	3	37,5%	Ae, A, An, Ang
1	24	Bu	8	33,34%	B, Bang, Bau, Bah, Be, Bo
2	2	Dan	2	100%	-
3	5	Ka	3	60%	Ke, Ko
4	7	Gu	7	100%	-
5	18	Ngu	8	44,45%	Nga, Ngi, Ngo
6	2	Pa	2	100%	-
7	8	Bi	6	75%	Be
8	31	Ge	7	22,58%	Gu, Gai, Gan, Gang
9	10	Be	6	60%	Bai, Bau
10	4	Yi	3	75%	Ya
11	9	Ng	2	22,22%	Nga, Ngu, Ngo, Ngan
12	12	La	4	33,33%	Lau, Lah, Lang, Lar, Lu
13	15	Yo	7	46,67%	Ya, Yai, Yan, Yau, Yu
14	32	Ni	8	25%	No, Ne, Nu, Nah, N, Nai, Nan
15	9	Sa	3	33,33%	San, Sang, Se, Su, Si
16	41	Nge	7	17,1%	Ngar, Ngah, Ngi, Ngai
17	5	Rau	4	80%	Ro
18	9	Pe	6	66,67%	Pi, Po
19	17	Bau	7	41,18%	Bu, Bar, B, Bo
20	47	Ma	6	12,77%	Mar, Man, Mai, Mau, Mo
21	18	Ti	7	38,89%	Ta, Te, Tau
22	9	Go	6	66,67%	Ga, Gi
23	20	Ngo	8	40%	Ngang, Ngau, Ngar
24	6	Mi	5	83,3%	Ma
25	8	Bai	6	75%	Bu
26	7	Kah	4	57,14%	K, Kan
27	8	Gau	4	50%	Gi, Gar
28	26	Po	7	26,92%	Pe, Pai, Par
29	2	Te	2	100%	-
30	12	Ngai	6	50%	Nga, Ngau, Ng
31	17	Gai	7	41,18%	Ga, Ge, Go, Gan
32	13	De	7	53,84%	Da, Do, Dau
33	11	Me	8	72,72%	Mi, Mo
34	9	Pau	7	77,78%	Pi, Po
35	4	To	3	75%	Ta
36	25	Bang	8	32%	Bai, Bau, Bah, B
37	3	Ngang	2	66,67%	Nge
38	13	Di	7	53,84%	Da, Do, Dau
39	1	Nyang	1	100%	-
40	6	Mu	5	83,33%	Ma
41	9	Pan	7	77,78%	Pi
42	5	Man	4	80%	Me
43	11	Gar	8	72,72%	Ga, G
44	22	Bah	7	31,82%	Bi, Bo, Bau, Bah
45	12	Kar	5	41,67	Kah, K, Kai
46	5	Par	4	80%	Pi
47	2	Ngan	2	100%	-
48	21	Du	8	38,09%	Da, Dau, Dah
49	5	Na	4	80%	N
50	21	Tai	7	33,34%	Ta, Tu, To
51	3	Mar	2	66,67%	M
52	15	B	6	40%	Ba, Bu, Bo
53	14	Pah	7	50%	Pa, Par, Pan
54	10	Tan	7	70%	Ta, To
55	21	G	8	38,09%	Ga, Gau, Gah
56	3	Dai	3	100%	-
57	23	Tar	8	34,78%	Ti, To, Te, Tau
58	6	Mah	5	83,33%	Mi
59	8	Kan	6	75%	Ku
60	11	Ni	8	72,72%	Ne
61	10	Ja	7	70%	Jo, Ju
62	11	Dang	8	72,72%	Dau, D
63	3	Tah	3	100%	-
64	5	P	4	80%	Pa
65	9	Kang	7	77,78%	Ki
66	16	Dar	7	43,75%	Di, Do, Dau
67	17	Cu	8	47,05%	Ce, Co
68	16	Ji	6	37,5%	Jau, Jah
69	11	T	8	72,72%	Tu
70	5	Nu	5	100%	-
71	14	Nya	7	50%	Nyi, Nyo
72	2	Dah	2	100%	-
73	16	Ce	8	50%	Ca, C, Cah
74	1	Nyi	1	100%	-
75	13	Ju	7	53,84%	Je, Jo
76	5	K	4	80%	Ki
77	17	Nye	7	41,17%	Nya, Ny, Nyu
78	1	Cang	1	100%	-
79	1	Ci	1	100%	-
80	6	Je	5	83,33%	Jo
81	24	No	8	33,33%	Na, Ni, Nu
82	8	Co	6	75%	Ca
83	38	Nyu	8	21,05%	Ny, Nya, Nye
84	13	Kai	5	38,47%	Ko,Ke, Kah
85	3	Yi	3	100%	-
86	21	Jo	7	33,33%	Je, Ju, Jau
87	4	Ye	4	100%	-
88	2	Ran	2	100%	-
89	9	Nyo	7	77,78%	Ny
90	6	Ar	6	100%	-
91	6	Jau	6	100%	-
92	18	Nai	8	44,44%	Na, Ni, Nu
93	19	Nyau	7	36,84%	Nya, Ny, Nyu
94	8	Ko	5	62,5%	Kah, K, Kai
95	7	Yo	6	85,71%	Y
96	18	Rah	8	44,44%	Rau, Ran
97	17	Jai	7	41,17%	Je, Jo
98	10	Nyang	8	80%	Nyi, Nyo
99	8	Jang	5	62,5%	Ji, Jau
100	7	Mang	3	42,86%	Mar, Man
101	10	Ca	4	40%	Car, Cah, Co, Ce
102	12	Ro	5	41,67%	Rai, Re, Ru, Rah
103	4	An	2	50%	A, Ao
104	17	Tu	8	47,1%	T, Tang, Ta
105	14	Ban	6	42,86%	Be, Bo
106	3	Di	2	66,67%	Dar
107	4	Nyo	2	50%	Nyan, Nyah
108	9	Nyan	2	22,22%	Ny, Nyai, Nyo, Nyi
109	6	Ngau	2	33,33%	Ngai, Ngah, Ngu
110	6	Tau	3	50%	Tu, Ta, Te
111	9	Nga	2	22,22%	Ngai, Ngan, Ngang, Ngar, Ngo
112	4	Nyi	4	100%	-
113	6	Wi	4	66,67%	Wan, War
114	1	Car	1	100%	-
115	1	Dau	1	100%	-
116	6	Ru	6	100%	-
117	9	Du	3	33,33%	Dah, Di, Dar, Dau
118	6	Pi	6	100%	-
119	14	Gan	5	35,71%	Ga, Gar, Ge, Gi, Go, Gu
120	4	Nyu	2	50%	Nyau, Nye
121	3	Cu	2	66,67%	C
122	2	Ai	2	100%	-
123	5	Dan	2	40%	Dah, Dai, De
124	3	Nau	3	100%	-
125	4	Mai	3	75%	Mah
126	1	Mau	1	100%	-
127	6	Do	2	33,33%	Da, Di, Du, Dau
128	15	Wang	6	40%	Wa, Wo, Wan, War

129	7	Da	2	28,57%	Dai, Dang, De, Do, Du
130	2	Ci	2	100%	-
131	1	Nyo	1	100%	-
132	6	Gi	3	50%	Gar, Gu
133	35	Pang	7	18,91%	P, Pa, Pah, Par, Pau, Pe, Po
134	11	Yah	3	27,28%	Y, Yai, Yan, Yi, Yar, Ye
135	4	Cau	3	75%	Ci
136	1	D	1	100%	-
137	10	Bo	4	40%	B, Ban, Bau
138	11	Nyau	5	45,45%	Ny, Nyar, Nyu
139	10	Ai	5	50%	Au, An, A, Ar
140	6	Y	3	50%	Ya, Yan
141	6	Ngar	5	83,3%	Ngang
142	3	Ru	2	66,67%	R
143	1	Ngah	1	100%	-
144	2	Ne	2	100%	-
145	6	Pai	5	83,3%	Pah
146	1	Ju	1	100%	-
147	4	Ci	4	100%	-
148	12	W	7	58, 34%	Wah, Wai
149	6	Mo	2	33,33%	Ma, Mah, Mi, Mu
150	12	Sang	3	25%	Sa, Sai, San, Se, Su
151	15	Au	4	26,67%	Ai, Ang, A, Ar, Ai, Ao
152	5	Ny	2	40%	Nyai, Nyang, Nyar
153	1	Gang	1	100%	-
154	10	Kau	3	30%	K, Kai, Kan, Kar, Ke
155	2	L	2	100%	-
156	1	Can	1	100%	-
157	9	La	3	33,33%	Lan, Lar, Lau, Lu
158	1	Bar	1	100%	-
159	5	Cang	4	80%	Ca
160	7	Ah	3	42,86%	Ai
161	4	Nyai	2	50%	Nyah, Nyau
162	10	Tang	8	80%	Tar
163	8	Yu	3	37,5%	Ya, Yar, Ye, Yi
164	9	Ba	7	77,78%	Bah
165	24	Le	6	20,69%	Lah, Lai, Lan, Lang, Lo, Li, Lar
166	11	Sar	4	36,37%	Sa, San, Sah, Su
167	1	Au	1	100%	-
168	10	Ra	3	30%	Rai, Ran, Rang, Re
169	2	Dau	1	50%	Dai
170	8	Nyar	2	25%	Nyai, Nyang, Nyan, Nye, Nyi, Nyo
171	1	De	1	100%	-
172	1	Car	1	100%	-
173	5	Mo	3	60%	Man, Mah
174	9	Ngi	6	66,67%	Nga, Ngan
175	22	Do	6	27,28%	A, A, Kah, Kau
176	1	Co	1	100%	-
177	22	San	5	22,72%	Sa, Sah, Sar, Si
178	3	Ri	3	100%	-
179	18	Wan	7	38,89%	Wah, Wa, War
180	6	Ng	3	50%	Nge, Ngan, Ngah
181	14	Ta	6	42,85%	Te, Tan, Tar
182	25	Tang	6	24%	Ta, Te, Tau, Ti
183	3	Can	2	66,67%	Car
184	7	Ran	4	57,14%	Ro, Rai
185	2	Nyi	2	100%	-
186	3	Jo	2	66,67%	Jang
187	30	M	6	20%	Ma, Man, Mo, Mar
188	5	C	3	60%	Ce
189	2	Pu	2	100%	-
190	10	Nang	7	70%	N, Na
191	6	Jan	5	83,33%	Ja
192	2	Yau	2	100%	-
193	6	Gu	4	66,67%	Ge
194	5	Nan	3	60%	Nar
195	10	Jar	7	70%	Ji, Jau
196	1	Cau	1	100%	-
197	7	Nyan	6	85,71%	Ny
198	7	Jah	5	71,42%	Je
199	1	L	1	100%	-
200	7	Nar	6	85,71%	Nah
201	8	Yah	5	62,5%	Ya, Y
202	2	Cai	2	100%	-
203	18	Ke	6	33,33%	Kang, Kan, Ko
204	4	Nyar	3	75%	Ny
205	8	R	6	75%	Ra
206	2	J	2	100%	-
207	2	Nah	1	50%	Na
208	1	Cang	1	100%	-
209	3	Nyah	2	66,67%	Nyau
210	5	N	4	80%	Na
211	6	Ga	5	83,3%	Gu
212	6	Re	5	83,33%	Ru
213	6	Ny	4	66,67%	Nya
214	2	Ho	2	100%	-
215	12	Cah	6	50%	Ca, Cau, Car
216	1	Nya	1	100%	-
217	1	Nye	1	100%	-
218	4	Y	3	75%	Ya
219	1	Rah	1	100%	-
220	1	Si	1	100%	-
221	1	Dau	1	100%	-
222	8	Ki	5	62,5%	Ku, Kar
223	1	Rau	1	100%	-
224	1	Rar	1	100%	-
225	13	Ku	4	30,77%	K, Kan, Kar
226	3	Lai	2	66,67%	La
227	1	Dau	1	100%	-
Total	**2031**		**975**		

The description of the image above is as follows :

1. Class Label is the name of the folder representing the class.
2. Number of Images is the total number of sample images available in each class category for model testing.
3. Major refers to the typeface and punctuation in each class.
4. Number of Major Images is the highest number of images owned by a single class in the directory.
5. Percentage of Major Class is the proportion of the number of images in the majority class to the total number of images across all classes, expressed as a percentage.
6. Miscluster refers to the grouping error where an item is placed in the wrong group.

If the score is below (<5), there are several factors that could contribute to this, such as :

1. Issues with the character's appearance that reduce the clarity of the letters in the writing.
2. The quality of the images created by the respondents is still lacking.

The result of the clustering process for the OKU Timur character images correctly grouped under each class label amounts to 975 characters from a total of 2031 data points. When expressed as a percentage, this means 48% of the images were clustered accurately. This is considered quite good for clustering using the K-Means Algorithm. Below is a sample from the respondents who filled out the form for the OKU Timur character in the letter 'Ng'.

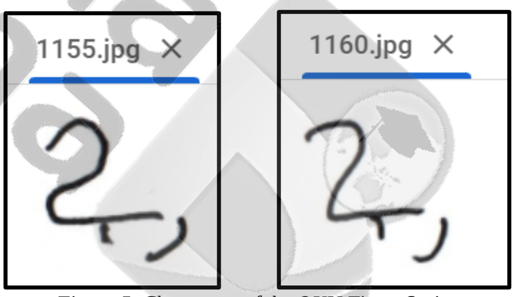


Figure 5. Characters of the OKU Timur Script

IV. CONCLUSION

In this study, a clustering model for the OKU Timur script images has been successfully developed using the K-Means algorithm. This algorithm is effective in grouping the images into several clusters based on visual similarity and extracted feature characteristics. The clustering results show that out of 2,031 available data, 975 characters were successfully grouped correctly. In percentage terms, 48% of these images were accurately clustered according to their respective groups. Despite facing some challenges in grouping images with low similarity, this model provides a strong foundation for further development in automatic script recognition and the preservation of OKU Timur culture.

Recommendations for improving the OKU Timur script image clustering model in the future include enhancing the data grouping process, particularly in the area of punctuation, which lacks accuracy, and ensuring that all data can be read correctly.

Additionally, optimizing the process to shorten the time required for data rendering and experimenting with other algorithms to expedite the processing of larger datasets are advised.

References

[1] E. Roza, "Aksara Arab-Melayu di Nusantara dan Sumbangsihnya dalam Pengembangan Khazanah Intelektual," *TSAQAFAH*, vol. 13, no. 1, p. 177, May 2017, doi: 10.21111/tsaqafah.v13i1.982.

[2] R. Alhapizi, M. Nasir, and I. Effendy, "Penerapan Data Mining Menggunakan Algoritma K-Means Clustering Untuk Menentukan Strategi Promosi Mahasiswa Baru Universitas Bina Darma Palembang," 2020. [Online]. Available: https://journal-computing.org/index.php/journal-sea/index

[3] A. Cluster Provinsi Indonesia Berdasarkan Produksi Bahan Pangan Menggunakan Algoritma K-Means, T. Tendean, and W. Purba, "Analisis Cluster Provinsi Indonesia Berdasarkan Produksi Bahan Pangan Menggunakan Algoritma K-Means," *Jurnal Sains dan Teknologi*), vol. 1, no. 2, pp. 5–11.

[4] Sekar Setyaningtyas, B. Indarmawan Nugroho, and Z. Arif, "Tinjauan Pustaka Sistematis: Penerapan Data Mining Teknik Clustering Algoritma K-Means," *Jurnal Teknoif Teknik Informatika Institut Teknologi Padang*, vol. 10, no. 2, pp. 52–61, Oct. 2022, doi: 10.21063/jtif.2022.v10.2.52-61.

[5] I. Nyoman and M. Adiputra, "Clustering Penyakit DBD Pada Rumah Sakit Dharma Kerti Menggunakan Algoritma K-Means," *INSERT: Information System and Emerging Technology Journal*, vol. 2, no. 2, p. 99, 2021.

[6] M. Benri, H. Metisen, and S. Latipa, "Analisis Clustering Menggunakan Metode K-Means Dalam Pengelompokkan Penjualan Produk Pada Swalayan Fadhila," 2015.

[7] T. D. Pangestu, V. Yose Ardila, M. Suteja, and S. P. Barus, "Klasterisasi Hewan berdasarkan Morfologi dengan K-Means Klastering untuk Memudahkan Pemahaman Taksonomi Hewan Klastering Animals based on Morphology with K-Means Klastering to Facilitate Understanding of Animal Taxonomy."