Data Science

Assignment4(extra point) – A-Z Handwritten Alphabets

Farimah Rashidi – 99222040

1. Introduction

We want to predict handwritten alphabet using machine learning methods.

The dataset contains 26 folders (A-Z) containing handwritten images in size 28*28 pixels, each alphabet in the image is center fitted to 20*20-pixel box. Each image is stored as Gray-level.

2. Directory Exploration

In this section, I initiate the exploration of the input directory to familiarize myself with the files and directories available in the Kaggle environment. I systematically list all files under the '/kaggle/input' directory. This step is crucial for ensuring accurate file path references in subsequent sections of the code, providing a foundational understanding of the dataset's structure.

3. Dataset Loading and Exploration

In this part, I load the handwritten alphabets dataset into a Pandas DataFrame. The resulting DataFrame, denoted as ‘df’, becomes the focal point for data exploration.

I employ functions such as head(), info(), shape, and describe(). These functions collectively showcase the first few rows, data types, dimensions, and summary statistics, offering a comprehensive overview. Here are our results:

```
print(df.head())
```

```
   0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  ...  0.639  0.640  0.641  \
0  0    0    0    0    0    0    0    0    0    0  ...      0      0      0
1  0    0    0    0    0    0    0    0    0    0  ...      0      0      0
2  0    0    0    0    0    0    0    0    0    0  ...      0      0      0
3  0    0    0    0    0    0    0    0    0    0  ...      0      0      0
4  0    0    0    0    0    0    0    0    0    0  ...      0      0      0

   0.642  0.643  0.644  0.645  0.646  0.647  0.648
0      0      0      0      0      0      0      0
1      0      0      0      0      0      0      0
2      0      0      0      0      0      0      0
3      0      0      0      0      0      0      0
4      0      0      0      0      0      0      0

[5 rows x 785 columns]
```

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 372450 entries, 0 to 372449
Columns: 785 entries, 0 to 0.648
dtypes: int64(785)
memory usage: 2.2 GB
None
```

```
print(df.describe())
```

```
                0       0.1       0.2       0.3       0.4       0.5  \
count  372450.000000  372450.0  372450.0  372450.0  372450.0  372450.0
mean       13.523490       0.0       0.0       0.0       0.0       0.0
std         6.740824       0.0       0.0       0.0       0.0       0.0
min         0.000000       0.0       0.0       0.0       0.0       0.0
25%        10.000000       0.0       0.0       0.0       0.0       0.0
50%        14.000000       0.0       0.0       0.0       0.0       0.0
75%        18.000000       0.0       0.0       0.0       0.0       0.0
max        25.000000       0.0       0.0       0.0       0.0       0.0

             0.6       0.7       0.8       0.9  ...         0.639  \
count  372450.0  372450.0  372450.0  372450.0  ...  372450.000000
mean        0.0       0.0       0.0       0.0  ...       0.001616
std         0.0       0.0       0.0       0.0  ...       0.490788
min         0.0       0.0       0.0       0.0  ...       0.000000
25%         0.0       0.0       0.0       0.0  ...       0.000000
50%         0.0       0.0       0.0       0.0  ...       0.000000
75%         0.0       0.0       0.0       0.0  ...       0.000000
max         0.0       0.0       0.0       0.0  ...     252.000000

              0.640         0.641         0.642         0.643  \
count  372450.000000  372450.000000  372450.000000  372450.000000
mean        0.001592       0.001117       0.000929       0.000685
std         0.517297       0.421332       0.419180       0.385566
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000       0.000000
50%         0.000000       0.000000       0.000000       0.000000
75%         0.000000       0.000000       0.000000       0.000000
max       226.000000     229.000000     228.000000     235.000000

              0.644         0.645         0.646         0.647  \
count  372450.000000  372450.000000  372450.000000  372450.000000
mean        0.000596       0.000618       0.000690       0.000239
std         0.319820       0.288942       0.335227       0.134852
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000       0.000000
50%         0.000000       0.000000       0.000000       0.000000
75%         0.000000       0.000000       0.000000       0.000000
max       194.000000     103.000000     198.000000      82.000000

              0.648
count  372450.000000
mean        0.000011
std         0.006554
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         4.000000

[8 rows x 785 columns]
```

```
df.shape
```

```
(372450, 785)
```

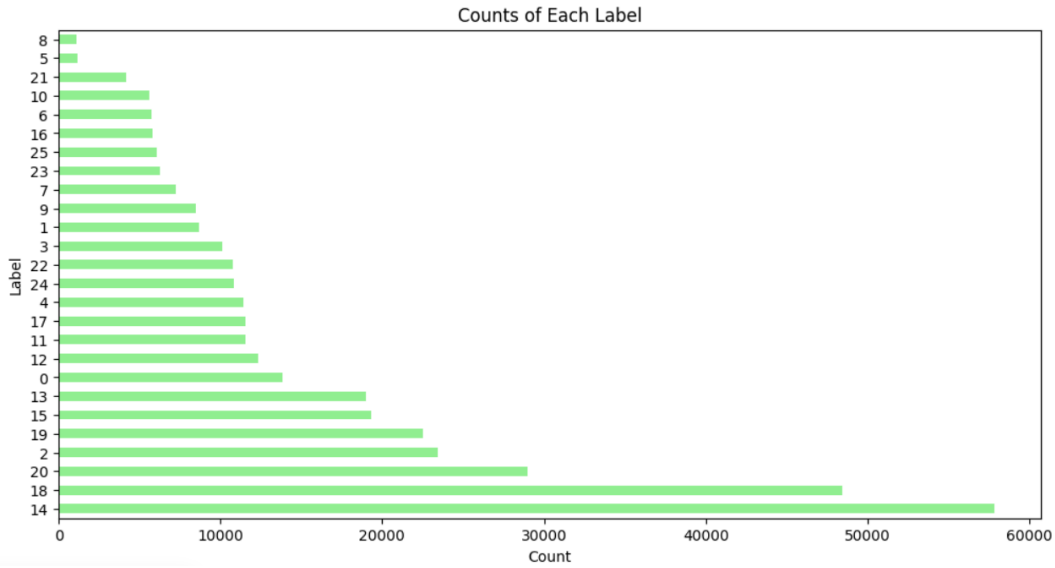
4. Dataset Preprocessing and Visualization

Moving on to dataset preprocessing, I create a duplicate DataFrame named ‘df_copy’ and opt to enhance clarity by renaming the '0' column to 'label'. Let’s look at this part’s result:

	label	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	...	0.639	0.640	0.641	0.642	0.643	0.644	0.645	0.646	0.647	0.648
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 785 columns

Visualization techniques are then employed to gain deeper insights. This bar plot shows the counts of each label:



Additionally, a loop is implemented to showcase one example image per label. Here are some examples of this part:

5. Data Splitting for Machine Learning

In this part, I focus on preparing the dataset for machine learning tasks.

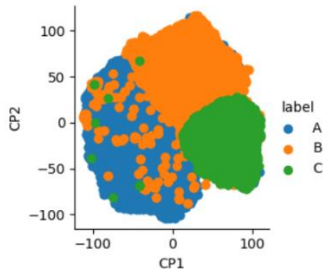
Using the train_test_split function, I meticulously split the data into training and testing sets. A key consideration is maintained through the stratify parameter, ensuring that the class distribution is preserved in both sets. This step lays a solid foundation for subsequent machine learning model development and evaluation, underscoring the importance of balanced data representation in the training and testing phases.

6. T-SNE

In this section, I employed t-SNE to reduce the dimensionality of the handwritten alphabet dataset. The t-SNE algorithm was implemented with two components for visualization purposes, using a random seed for reproducibility. The transformed data, consisting of two new features labeled as CP1 and CP2, was then organized into a DataFrame named `cps_df`. This DataFrame includes the transformed coordinates along with the corresponding labels.

To facilitate interpretation, I assigned meaningful alphabet labels to the numeric values in the 'label' column using a predefined mapping dictionary. This ensures that each point in the scatter plot corresponds to a specific letter of the alphabet, enhancing the interpretability of the visualization.

Finally, I created a FacetGrid to generate a scatter plot of the t-SNE-transformed data. Each point on the plot represents a handwritten alphabet sample. This visualization allows for a qualitative assessment of the spatial distribution of handwritten letters, providing insights into potential clusters or patterns within the dataset. Here is the result of using 30000 data:



Here is the result on 40000 data: