The Letter Recognition from Sign Language Images

Student ID 160709017
Name Beyza
Surname Kurt

1 Introduction

"The Letter Recognition from Sign Language Images", the project I chose to do for the final project of the CENG 3521 Data Mining course, has been successfully completed and in this report, which models are suitable for images, which ones give better results and the cause-effect relationship between them are examined.

2 TR Sign Language Dataset

The data set is taken from https://www.kaggle.com/berkaykocaoglu/tr-sign-language.

- There are no images of dotted or capped letters (i, ç, ş, ü, ö, ğ).
- It consists of 2 main folders as "train" and "test".
- There are at least 4000 photos for each letter in the "train" folder, and at least 1500 photos for those in the "test" folder.
- Apart from letters, there are 3 more folders named "del", "nothing" and "space" in "train" and "test" folders.

In the data set made ready for use in the project:
- "del", "nothing" and "space" folders have been removed.
- When it was understood that the program could not use all images due to lack of hardware, **the folders of each letter** in the "train" were moved to the main location with the first **2000** images. Thus, "train" and "test" files were removed and the data set in the code was divided into "train" and "test".

Figure 1 below shows the sign language representation of all letters:

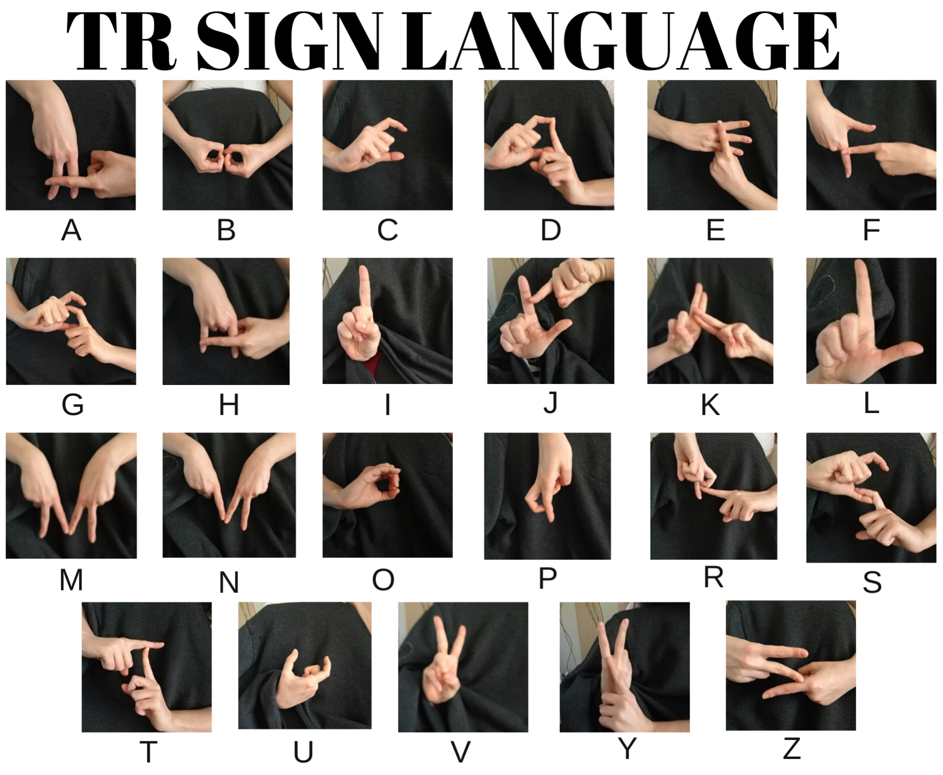

Figure 1: TR Sign Language Images

3 Subjects

Although I use images as data, instead of using libraries like keras that come to mind when it comes to ”image classification”, I wanted to put forward a study by sticking to the sklearn library we use in our homework. Because the digit data set we used during our homework was also a visual data set. I aimed to measure Sklearn’s performance on another visual dataset.

3.1 Data Preprocessing

Because the images were colorful, they can be 3-dimensional arrays. This was not a suitable form to fit into classification models. Therefore, the rgb2gray function of skimage.color has been applied to all images. So they are grayscale and 2-dimensional now.

The states of an image whose original is not square, before and after the transformations are applied can be seen in Figure 2:

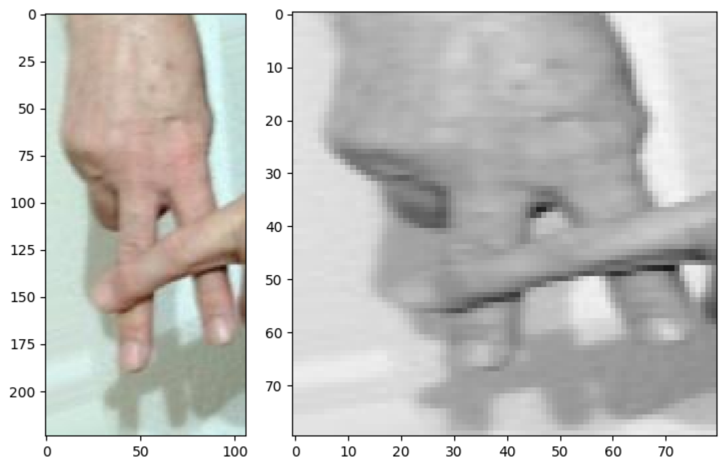

Figure 2: An Image from Data Set; Before and After Transformation

I saved the images that are still not in proper form for the models by bringing them to a one-dimensional form using reshape().
Apart from these, I have examined how the following methods create effects on visuals, but I did not use them. The methods can be found in Section 3.1.1 with sample visuals.

3.1.1 Methods for Image Segmentation

3.1.1.1 Structural Similarity Index

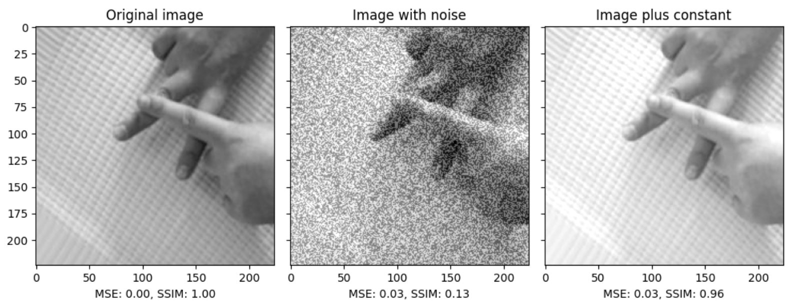

Figure 3: Structural Similarity Index

The explanation of ”Structural similarity index” in Scikit-Image documentation is as follows:
“When comparing images, the mean squared error (MSE)–while simple to implement–is not highly indicative of perceived similarity. Structural similarity aims to address this shortcoming by taking texture into account. The example shows two modifications of the input image, each with the same MSE, but with very different mean structural similarity indices.”
In one of the previous assignments, we added noise to the Digit dataset, but the form with no added noise performed best.

3.1.1.2 Thresholding

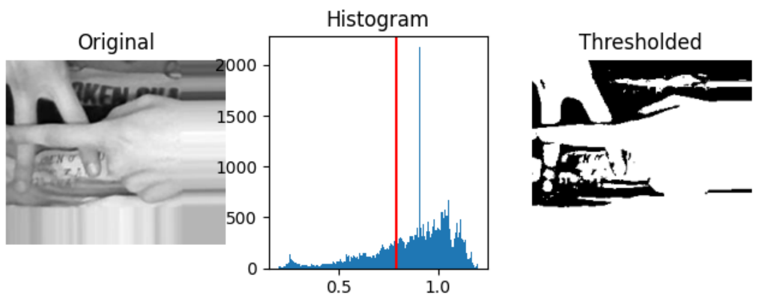

Figure 4: Thresholding

The explanation of "Thresholding" in Scikit-Image documentation is as follows:
"Thresholding is used to create a binary image from a grayscale image. We illustrate how to apply one of these thresholding algorithms. Otsu's method 2 calculates an "optimal" threshold (marked by a red line in the histogram below) by maximizing the variance between two classes of pixels, which are separated by the threshold. Equivalently, this threshold minimizes the intra-class variance."
Although it might seem like it could achieve better results in this image, it was not used because it made most images meaningless.

3.1.1.3 Morphological Snakes

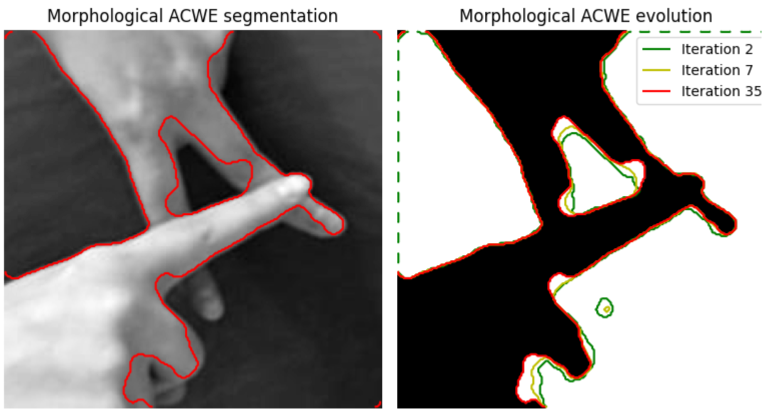

Figure 5: Morphological Snakes

The explanation of "Morphological Snakes" in Scikit-Image documentation is as follows:
"Morphological Snakes use morphological operators (such as dilation or erosion) over a binary array instead of solving PDEs over a floating point array, which is the standard approach for active contours. This makes Morphological Snakes faster and numerically more stable than their traditional counterpart."
Although it might seem like it could achieve better results in this image, it was not used because it made most images meaningless.

3.2 Logistic Regression

It is obvious that a linear classification method will not be effective for a visual data set. However, I still wanted to examine the results of my project by including linear models.In this direction, I used sklearn's linear models "LogisticRegression" and "SGDClassifier". You can see the results I have obtained using these two models, whose performances are almost the same, in Table x.

Model	Number of Sample for Each Letter	Score
LogisticRegression	500	0.058840579710144926
SGDClassifier	500	0.04987142171067184 3

Table 1: Results of The Logistic Regression Models

I expected the results to be low, but I was a little surprised that it was so low. Then I decided to test it using sklearn's digit dataset. You can find the results and the evaluation of the differences in Section 4.

3.3 Neural Network-based Classification

I thought sklearn's neural network model MLPClassifier would do better after linear models. But again I did not get the results I expected. You can see the results in Table 2.

Number of Sample for Each Letter	Hidden Layer Sizes	Score
500	32, 16, 8, 4, 2	0.08492753623188405
500	64, 32, 16, 8, 4, 2	0.09014492753623188
500	128, 64, 32, 16, 8, 4, 2	0.03826086956521739
500	256, 128, 64, 32, 16, 8, 4, 2	0.03826086956521739
500	512, 256, 128, 64, 32, 16, 8, 4, 2	0.03826086956521739

Table 2: Results of The Neural Network-based Classification Models

Looking at the results, we cannot say that the increase in the hidden layer sizes for this data set has a positive effect on the result.
I also tested this model with sklearn's digits dataset. You can find the results and the evaluation of the differences in Section 4.

3.4 Ensemble Learning-based Classification

For Ensemble Learning I used both models of sklearn: "VotingClassifier", "BaggingClassifier".

- For the Voting Classifier, I used MLP classifiers that I created and stored in the previous step.
- I created an MLP classifier with (32, 16, 8, 4, 2) hidden layer sizes and splitted it into 5 equal parts. For the Bagging Classifier, I used them.

You can see the scores of the Voting classifier, the Bagging classifier and the split MLP classifiers I created for use in the Bagging classifier in Table x.

Model	Number of Sample for Each Letter	Score
Voting Classifier	500	0.14115942028985506
Bagging Classifier	500	0.10492753623188406
MLP Classifier 1	500	0.06985507246376811
MLP Classifier 2	500	0.07594202898550724
MLP Classifier 3	500	0.08057971014492754
MLP Classifier 4	500	0.06898550724637681
MLP Classifier 5	500	0.09014492753623188

Table 3: Results of The Ensemble Learning-based Classification Models

If we look at the results of the MLP classifiers on Table 2 we created in the previous step, we can say that the performance of the Voting classifier that has made classification using them is better.
If we look at the results of the MLP classifiers on Table 3, we can say that the performance of the Bagging classifier that has made better classification using 5 estimators in the ensemble.
I also tested this models with sklearn's digits dataset. You can find the results and the evaluation of the differences in Section 4.

4 Testing and Comparing with a Different Data Set

I was not satisfied with the results I got, and I used sklearn's digits dataset to examine the results, both to check the accuracy of the code and to compare the performance of my dataset with another dataset. You can see the results in Table 4.

Model	Score		Model	Score
Logistic Regression	0.9648148148148148		Voting Classifier	0.6
SGD Classifier	0.9351851851851852		Bagging Classifier	0.9277777777777778
MLP Classifier 1	0.85		Bagging MLP Classifier 1	0.4537037037037037
MLP Classifier 2	0.34629629629629627		Bagging MLP Classifier 2	0.8203703703703704
MLP Classifier 3	0.38333333333333336		Bagging MLP Classifier 3	0.08703703703703704
MLP Classifier 4	0.6407407407407407		Bagging MLP Classifier 4	0.3611111111111111
MLP Classifier 5	0.5777777777777777		Bagging MLP Classifier 5	0.8055555555555556

Table 4: Results of All Models on the Digits Dataset

If we compare the results in Table 4 with the results in Section 3, we can see that there is a huge difference between them. But first of all, we can say that **there is no error** in the accuracy of the code.
If we compare the data sets;

- While a digit image that in the Digits data set is 8x8 in size, the images in the TR Sign Language Images data set are mostly 224x224x3. Reducing the dimensions to 80x80 with preprocessing may have caused a loss of accuracy.
- Even after preprocessing, the size of the photos 10 times larger than the digits data set may have decreased the accuracy.
- While there are 10 classes in the Digits dataset, there are 23 classes in the TR Sign Language Images dataset. This may have significantly reduced the accuracy of the estimates in the classification.

5 Conv2D with Keras

Even though I checked it with the Digits data set, I made some experiments by looking at all the important parameters of the models; but the results continued to come low. So I wondered if the problem was that the models were not suitable for images. Thus, under the guidance of the article at https://www.machinecurve.com/index.php/2020/03/30/how-to-use-conv2d-with-keras/, first verify that the model works properly with the CIFAR-10 dataset used in the article. then I run the model by putting my dataset to the appropriate form and review the results. Also, I used the dataset by taking 500 samples from each letter because the performance of my computer was not sufficient before. But this time, I created and used my dataset using all the existing images, without any limitation.

```
Epoch 1/50 403/403 [==============================] - 977s 2s/step -
    loss: 3.1379 - accuracy: 0.0418 - val_loss: 3.1357 - val_accuracy: 0.0441
Epoch 2/50 403/403 [==============================] - 910s 2s/step -
    loss: 3.1354 - accuracy: 0.0475 - val_loss: 3.1359 - val_accuracy: 0.0425
Epoch 3/50 403/403 [==============================] - 907s 2s/step -
    loss: 3.1352 - accuracy: 0.0479 - val_loss: 3.1359 - val_accuracy: 0.0425
Epoch 4/50 403/403 [==============================] - 933s 2s/step -
    loss: 3.1355 - accuracy: 0.0450 - val_loss: 3.1360 - val_accuracy: 0.0425
Epoch 5/50 403/403 [==============================] - 921s 2s/step -
    loss: 3.1353 - accuracy: 0.0447 - val_loss: 3.1360 - val_accuracy: 0.0425
Epoch 6/50 403/403 [==============================] - 914s 2s/step -
    loss: 3.1353 - accuracy: 0.0482 - val_loss: 3.1361 - val_accuracy: 0.0425
Epoch 7/50 403/403 [==============================] - 914s 2s/step -
    loss: 3.1353 - accuracy: 0.0456 - val_loss: 3.1361 - val_accuracy: 0.0425
Epoch 8/50 403/403 [==============================] - 930s 2s/step -
    loss: 3.1354 - accuracy: 0.0420 - val_loss: 3.1360 - val_accuracy: 0.0425
Epoch 9/50 403/403 [==============================] - 913s 2s/step -
    loss: 3.1352 - accuracy: 0.0432 - val_loss: 3.1360 - val_accuracy: 0.0425
Epoch 10/50 403/403 [==============================] - 924s 2s/step -
    loss: 3.1356 - accuracy: 0.0441 - val_loss: 3.1361 - val_accuracy: 0.0425
```

I wanted to experiment with 50 epoch. But in the 10th epocht, I noticed that the value accuracy did not change. I did not continue to run it as this is proof that the progressive score value as low as 0.04 will not change much.

6 Results

Models could not reach adequate accuracy levels with TR Sign Language Images data set. I think the main reason for this is the data set itself. Because when I tried even with a model suitable for pictures, the score was not different from what I got with the models I wrote. Before testing the dataset with Conv2D, I thought it had a big impact that I could not use all the pictures because the performance of my computer or the resources I could use was not sufficient. But if we look at the Section 5, we can see that this is not quite true.

In addition, the increase in total speed was not high even when I ran the code by importing the data from Google Drive via Google Colab due to the inadequate performance of my computer. In fact, the process of importing files from Drive took much longer than on the computer, slowing the process.

Apart from these, the accuracy of the score may have been reduced due to the presence of images of letters similar to representation.

7 References

- sklearn.linear_model.LogisticRegression
- sklearn.linear_model.SGDClassifier
- sklearn.neural_network.MLPClassifier
- sklearn.ensemble.VotingClassifier
- sklearn.ensemble.BaggingClassifier
- sklearn.datasets.load_digits
- How to use Conv2D with Keras?
- The CIFAR-10 dataset