Leiden University
Computer Vision
Autumn 2025
Instructor: Hazel Doughty, Lu Cao

Assignment 1: Character Recognition

1 Goal

The goal of this assignment is to recognize the characters 'S', 'T' and 'V'. You will do this using features from the Fourier space and a classifier of your choice. Your solutions will be automarked with a variety of different inputs to test both the correctness and robustness of the system.

2 Data

We have provided you with 10 training images of the letters 'S', 'T' and 'V' as well as 5 test images. We have also provided you with starter code that reads in these images. Run the starter code with the command:

```
python stv_classifier.py test.csv
```

This will automatically load the images listed in `train.csv` and produce a prediction for the images in `test.csv`. Currently the prediction is always 'S', you will change this prediction to be based on image features and machine learning during the project.

3 Features

Your goal in this assignment is to classify the images based on features from the Fourier space of the image. To give you a start, we have already created the functions `get_ftt` and `get_fft_dataset` which perform the fast Fourier transform on a single image and a list of images. The starter code also visualizes the magnitude and phase of the first image in the dataset, so that you can inspect the Fourier space.

In this part, your goal is to change the `get_features` function to get informative image features that can help you distinguish the different characters.

Currently, this function outputs two features per image, the sum of the magnitude and the sum of the phase. These are not very useful features, you need to come up with your own better ones.

Hint: looking at particular regions of the phase and magnitude may be useful.

Note: you cannot use any non-fourier features or base your classification on the RGB pixel space.

4 Classifier

From the features you have produced, train a classifier to classify the input images into categories 'S', 'T', and 'V'. You should then modify the variable `predictions` so that it makes real predictions for the test images based on your trained classifier.

Hint: k-Nearest Neighbour classifier similar to the one you used in the practical session will do a decent job but is not the best.

Hint: you will need to do some iteration between improving the features and improving the classifier to find the best combination.

Note: you are allowed to use libraries for your classifier, *e.g.* `neighbors` from `sklearn`, however you are not allow to use a deep learning *e.g.* `MLPClassifier` from `sklearn.neural_network` or any imports from a deeplearning library such as PyTorch, Tensorflow or JAX.

5 Submission

Submit your new `stv_classifier.py` file to BrightSpace along with any dependencies, your training images, and your `train.csv` file **in a single zip file**. The automarker will run your code with the command:

```
python stv_classifier.py <our_test>.csv
```

where `<our_test>.csv` is a csv file pointing to our hidden test images. It will follow the same format as the `test.csv` we have provided.

Your code should output a new file named `<our_test>_predictions.csv` which has two columns, the first with the header `image_path` which specifies the relative path to the image and the second with the header `prediction` which specifies the prediction of your model for the corresponding image, either S, T or V. The starter code already writes a file in the correct format so if you do not modify this part of the code is will already work fine.

6 Grading

We will test your solution with a number of hidden test images to test the correctness and robustness of your model. The test images will be similar in nature to the test images you have been provided and each will contain a single instance of the digits ‘S’, ‘T’ or ‘V’. Your score will be based on how many of these images your model correctly classifies with the following breakdown:

6 points 80% accuracy on our simple hidden test set. This is very similar to the test images that we have given you, but not exactly the same so you cannot simply train your model on the test set.

1 points 80-100% accuracy on our simple hidden test set

3 points Performance on our more complex test set. We will not tell you what this contains, but think about what your model should be robust to accurately recognize characters formed with different handwriting styles.

-1 point For any adaption we have to do to your code to get it running with the command `python stv_classifier.py test.csv` or to produce a predictions csv file with the correct output as in the example code.

Your final grade is based on performance in each of these areas, with marks given proportionally based on how many images you correctly classify. For instance if your model achieves 75% accuracy on the simple hidden test set and 40% accuracy on the more complex test set your final grade would be (0.75/0.8)*6 + 0*1 + 0.4*3 = 6.825 which will be rounded to 7. Likewise, if you achieved 90% accuracy in the simple test set and 70% on the complex test set, but your code needed some adaptation to run with the correct command, your grade would be calculated as: (0.8/0.8)*6 + (0.1/0.2)*1 + (0.7*3) - 1 = 7.6, rounded to 7.5.

Note: If you use features based on the RGB image pixel space rather than the Fourier space or use a Deep Learning-based classifier you will get a grade of 0.