CS771 Assignment 3 : ML Noobs

(200186): Arnav Gupta; (200204): Aryan Vora
(200539): Kushagra Sharma; (200556): Mandar Wayal
(201050): Tejas R

October 2022

1 Pre-processing

We convert the given image (**RGB** format) into **HSV**. This separates *luma* or *image intensity* from color information, enabling us to compare the intensity components in the image.

The image was then **eroded** (using a kernel of size 5x5, containing ones), which removes the image boundaries. The stray lines are the most affected by erosion and are blended into the background more.

To eliminate the background, we take the top-left pixel (which will definitely belong to the background) and subtract its values from the whole image. This leaves us primarily with alphabets. The image is then **grayscaled** since the color of the alphabet does not help us a lot. However, the boundaries of these alphabets are poorly defined because of erosion, so we apply a **thresholding function**. The threshold for each pixel either completely whitens it (if it has a value different from the background) or blackens it (if it has a value the same as that of the boundary).

To **segment** the three alphabets, we traverse each column and count the number of non-boundary pixels. When this count exceeds a certain percentages ($\frac{20}{150}$ in our case), we assume that a letter has started, and when it falls below that value, we assume that it's finished. We assign the columns in between as the first alphabet. Similar processing to get the second and third alphabets. Suitable **padding** is assigned to each of these pictures so that all the image dimensions are uniform for the CNN to process.

//add images

2 Training

The **2000** training images were converted to **6000** processed grayscale images of shape **(160,157)**, each containing one greek alphabet, using the preprocessing techniques described above. To obtain the corresponding label for each of the 6000 images, we processed labels.txt, and converted the English labels ('ZETA' etc.) into the corresponding one hot encoding (for n images the labels tensor has shape (n,24) in one hot form) for training the model.

We trained the model using a 80-20 **Train Test split**, i.e 1600 of the given images were used for training and 400 were used for testing. Within the 1600 test images, we used 0.1 Validation split(160 images). These fractions were chosen because they are standard choices and an 80-20 split leaves us with enough training data and also a fairly complete test dataset.

The problem of identifying the letters in the captcha has been converted into a problem of classifying a grayscale image of an alphabet, and then getting the alphabets in each image using the predictions of the model.

A **CNN** was used because of its effectiveness on image data, and we tried various model sizes, finally settling on **3 Convolutional+Pooling layers**, with **dropout of 0.2** in two Convolutional layers and finally **two dense layers** with 128 and 24 neurons each (one output neuron per reference alphabet).

We used **early stopping callback**, which stopped training the model if the validation accuracy decreased or did not increase in an iteration. This helped us avoid **overfitting** (along with the dropout layers).

3 Hyperparameter Tuning

We use the model giving us the best validation accuracy using held out validation.

Optimizer	**Learning Rate**	**Training Accuracy**	**Training Loss**	**Val. Accuracy**	**Val. Loss**
Adam	10^{-2}	99.56%	0.0166	99.79%	0.0111
SGD	10^{-5}	38.03%	2.1239	54.58%	1.4816
Adam	10^{-5}	95.74%	0.1508	99.79%	0.0216
Adam	10^{-3}	99.63%	0.0138	99.79%	0.0064

4 Prediction

For predicting the string for the test image, we pre-process and segment the image in the same way we did for images in the training dataset. We split each image into three parts (each containing a single capital greek letter). We ensure that the dimensions of each segmented part is same as those of the images we passed to train our CNN.

Each segmented image is passed to our saved model, which predicts the letter in that part. Finally, we output the names of the three greek letters, as predicted by the saved model.