CS771 Introduction to Machine Learning
Assignment 3

Aditya Kankriya
22111072
adityask22@iitk.ac.in

Haris Khan
22111026
hariskhan22@iitk.ac.in

Srujana Sabbani
22111083
srujanas22@iitk.ac.in

Anshul Sharma
22111009
anshulsh22@iitk.ac.in

Madhav Maheshwari
22111037
madhavm22@iitk.ac.in

Faizal Khan
22111022
faizalk22@iitk.ac.in

1 Questions and Solutions

Describe the method you used to find out what characters are present in the image. Give all details such as the algorithm used including hyperparameter search procedures, and validation procedures. You have to give a detailed explanation even if you used an algorithm/ implementation from the internet – make sure to give proper credit to the person/source from where you took the code/algorithm. There is no penalty for using someone else's code/algorithm but there would be a heavy penalty for doing so without giving proper credit to that person/source.

Solution :

1.1 Pre-Processing Phase:

The given images from the train folder contain extra noise lines in the images and also the characters have been rotated so we should pre-process the data and remove all the noises possible from the images.

Each image in the train folder will go through the following steps:

- The image will be read by using opencv2 (1) imread() and convert the image RGB to HSR using opencv2 cvtColor()
- Subtracted the color from the image to remove the noise pixels from the background
- We use cv2.erode() method which erodes away the boundaries of foreground objects Always try to keep the foreground white by using a 3x3 kernel for 8 iterations. from the dry run we found 8 iterations will remove the obfuscation lines.
- Then converted the images to grayscale images and thresholded it to make a binary image with black and white pixels

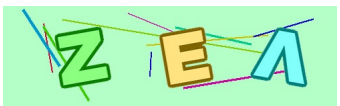

Figure 1: captcha image before pre-processing

Preprint. Under review.

Figure 2: captcha image after pre-processing

1.2 Segmentation Phase

From the pre-processed images in train data, we segment them into character images. Later these images will be used to train the neural network. The approach to detect any letter in the captcha, we traverse vertically i.e along the y-axis. We set Thresh_Minvalue = no_of_pixels_along_y-axis*255 which states that all the vertical line is pure white.

- The algorithm calculates all the pixels(let's say YL_sum) values along the y-axis and checks if(YL_sum < Thresh_Minvalue) then it states that the letter has been started from that point and we record that vertical line and set it as a start of the segment.
- we will continue until we found YL_sum = Thres_Minvalue and then set that axis to the end of the segment.
- This step Will be repeat this step over the captcha and be able to find no of letters written on the captcha.
- After finding each letter we are saving that segment of 20 x 20 pixels in a folder that will be used later to train our model.

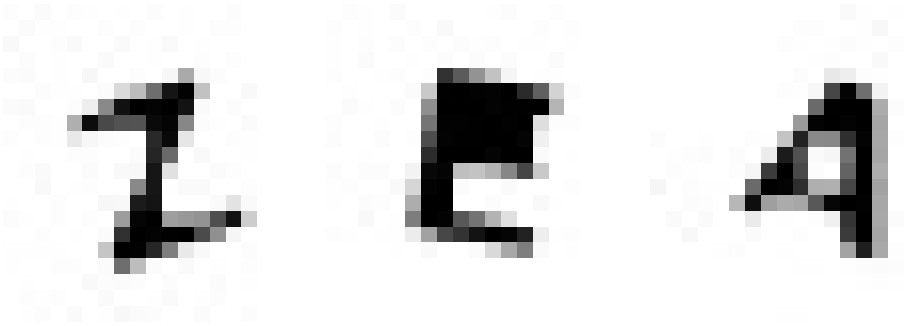

Figure 3: Segmented images (a) Zeta (b) Epsilon (c) Alpha

1.3 Classification Phase

The data of segmented images is split into train and test data with 75% and 25% respectively using scikit (2) library train-test-split(). The segmented images will be fed to CNN for the classification of each character. For building this convolution neural network we use Keras (3). Each character image will be given as input and the output labels were one-hot encoded using LabelBinarizer() using scikit.

- First Convolutional Layer (8, 5x5 Filters with padding('same') and 'relu' activation
- MaxPool with (2x2) stride and (2x2) pool size
- Dropout with 20% input set to zero
- Second Convolutional Layer (8, 5x5 Filters with padding('same') and 'relu' activation
- MaxPool with (2x2) stride and (2x2) pool size
- Dropout with 20% input set to zero
- Then the output was flattened and connected to a fully connected layer with 128 neurons and relu activation
- Final output layer with 24 neurons(output size) and softmax activation.

using scikit grid-search (4) method for hyper tuning to get the best possible parameters for training those are adam as an optimizer and categorical_crossentropy function for the loss. we train the neural network for 40 epochs

1.4 Some HyperParameter Tuning Results

We have tried with Different combinations of Neurons of different layers(Combinations like 8,6,128,etc) and we have Considered the pool size and strides as (2,2) and (2,2) respectively.
For Output Layer with 32 nodes(one for each possible letter/number we predict),we use 24.

We get the Accuracy as Follows:
1-0.9953 accuracy with loss:0.0150
2-0.9952 accuracy with loss:0.0152
3-0.9954 accuracy with loss:0.0147
4-0.9956 accuracy with loss:0.0146

So the Conclusion is ,We have tried with different Neurons and We have Got our best accuracy as 0.9956 with loss 0.0146.

1.5 Train your chosen method on the train data we have provided (using any validation technique you feel is good). You may or may not wish to use the reference images – it is your wish. Store the model you obtained in the form of any number of binary/text/pickled/compressed files as is convenient and write a prediction method in Python in the file predict.py that can take new images and use your model files you have stored to make predictions on that image. Include all the model files, the predict.py file, as well as any library files that may be required to run the prediction code, in your ZIP submission. You are allowed to have subdirectories within the archive.

Solution :

2 Code

Code Link :-https://web.cse.iitk.ac.in/users/madhavm/mlass3/submit.zip

References

[1] `https://docs.opencv.org/2.4/modules/imgproc/doc/imgproc.html`, 2022.

[2] `https://scikit-learn.org/stable/documentation.html`, 2022.

[3] `https://keras.io/`, 2022.

[4] `https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras`, 2022.