Captcha Recognition

Team Details:

Team Name: B1W2

Team Member's Names:

1)Mithun P

2)Karmanya Gupta

3)Nandeesh Bhatrai

4)Atharva Keny

Problem Statement:

The task at hand was to develop a model that would act like a captcha detector. The model would receive text captcha made up of alphanumeric characters and would recognize this image and output its corresponding characters. To complete this task we received a dataset of over 100,000 images of captcha along with their corresponding correct characters.

Process:

1)Preprocessing:

Image Preprocessing:

Final Image Characteristics: 40*150 pixels, Grayscale, RGB values of all pixels were normalised between 0 and 1.

Text Preprocessing:

Final Text Characters: 3-D vector of dimensions 5*n*62. Where n represents no. of loaded images. For every loaded image we had 5 1D vectors of 62 length, where each 1D vector corresponded to the alphanumeric character of the label it represented. Ex: To represent 'a' all the other array elements would be 0 except for the first element which would be 1. (First lowercase letters, then capital letters, then digits)

2)Model Chosen:

We chose the CNN model because it is the most convenient model for image processing. We have submitted two .ipynb files with slight variations in the model architecture and the way we trained the model. [We were not able to train the whole data on the model because of out of memory error]

Model 1: Training Data: 60000 images, Test Data: 19000 images, Validation Data: 1000 images

To avoid overfitting we added three layers to introduce noise [random rotation, translation and zoom]. We played around a bit with the architecture and added a few more sets of convolutional2d, maxpooling2d and batch normalisation layers to increase the accuracy of the model. We also used the soft max activation function since we had to do multiclass classification. We also added a dropout layer to drop 20 percent of the images to avoid overfitting. The output is received in 5 layers, each layer representing the prediction for each letter.

Model 2: Training Data: 95000 images Validation Data: 5000 images Test Data: 13000 images

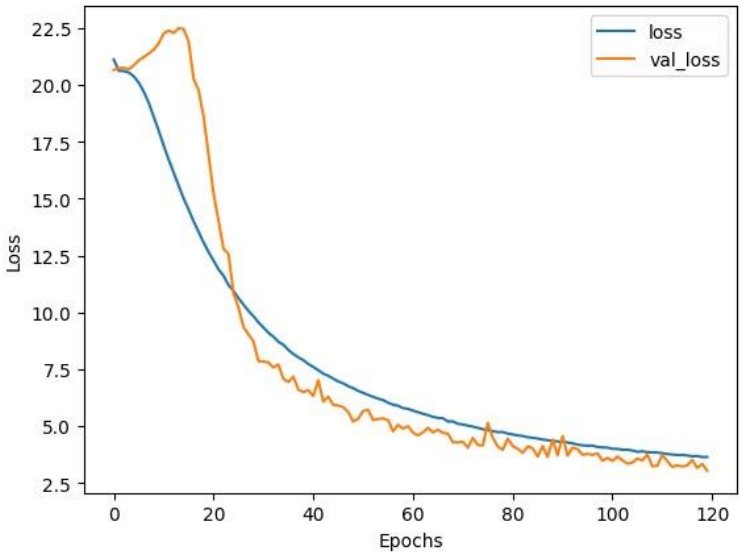
We used batch processing in this model to improve the accuracy, we first loaded 45000 images and trained the model on that for 80 epochs and then trained the remaining 50000 images for 80 epochs. We used the same architecture as Model 1 and same format of Output.

P.S. We found a few images in the dataset for which the answer is incorrect [ex: 44QqE, ymx13, IOnBr] , This may result in incorrect training of the model as well as some places where the model predicted correctly but given answer was wrong. This has resulted in an overall decrease in the accuracy of both the models.

Accuracy and Loss Graph:

Model 1:

(https://colab.research.google.com/drive/1yHnH5pCGm68BeLGlE3z5Psghv2or_S1S?usp=sharing)



Test Accuracy of each Character:

1st 84.79%

2nd 78.35%

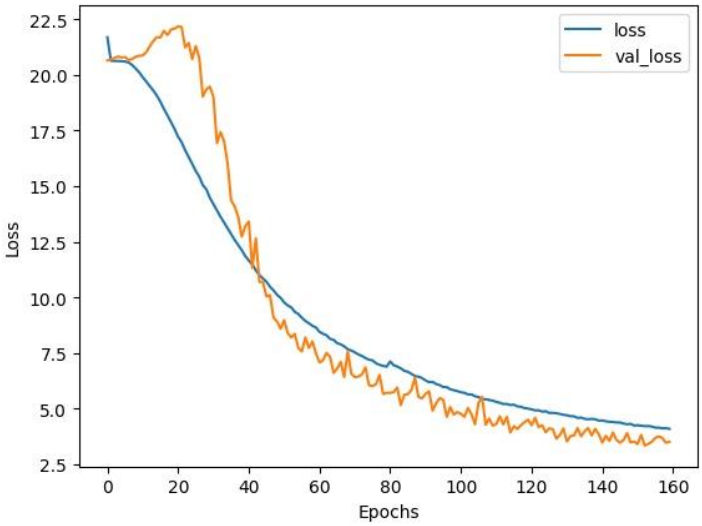
3rd 75.70%

4th 78.65%

5th 84.10%

Total Accuracy (Considering captcha as a whole): 39.51%

Model2:
(https://colab.research.google.com/drive/1w8UtMOvnYjY6reDdeXAcu7mREFb7w7Jc?usp=sharing)



Test Accuracy of each Character:

1st 82.09%

2nd 72.68%

3rd 69.56%

4th 72.60%

5th 81.80%

Total Accuracy (Considering captcha as a whole): 31.11%

Final Model Chosen and Conclusion:

Model 1 is our chosen final model as it has slightly higher accuracy. The reason for choosing this model is it uses various features to prevent common problems like overfitting. The model is giving high individual accuracy for each character, but for the captcha as a whole the accuracy dropped. Captcha is designed so that a machine cannot pass as a human. Therefore even after making such a complex model, the accuracy is relatively small.