Sam Vance, Rylee Albrecht, Brianna White

DNN Captcha Solving

Captchas are one of the most widely used ways of limiting bot accounts, spam, and general bot usage of the internet. The idea is to make a test that can only be passed by a human, but if artificial intelligence is able to solve the puzzle, the security and integrity of these captcha protected sites can be at risk. In this research, we explore the ability of DNN (Deep Neural Nets) to breach two common types of captchas, namely text based captchas and image based captchas.

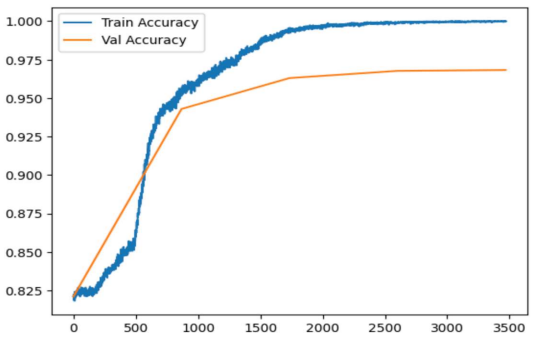
Image based captchas typically show a grid of various images and ask the user to highlight every instance of a specific item (dogs, cats, street lights, cars, etc.) Preliminary research showed that these captchas are classified and not publicly released for research. To create an adequate replacement, we used CIFAR-10, which is a labeled dataset of 32x32 pixel images designed for image classification problems. To simulate captchas, we took groups of 9 images and stacked them into a single 96x96 image. By grouping the images, we could duplicate every image 40x, and then combine them in unique ways to create 40x the amount of unique captchas. For text based captchas, we used a dataset from Kaggle which was made up of 5 character text with distortion applied.

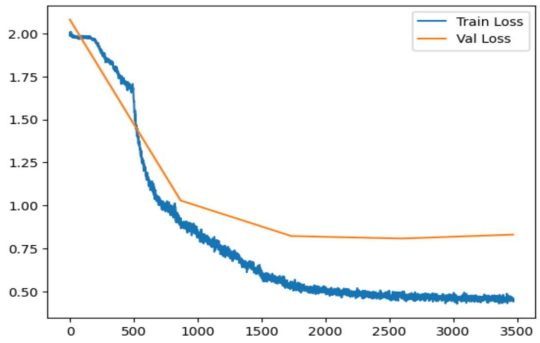
Initial research suggested that using a ResNet-50 pre-trained on the ImageNet dataset would likely provide a solid base model from which we could apply supervised transfer learning techniques to our specific task. Due to the models predicting many items per image, we modified the fully connected layer (the output layer) of the model. The updated output layer outputs a batch size 9x10 tensor. This corresponds to the 9 classifications per captcha, and the 10 classes present in the CIFAR-10 dataset.

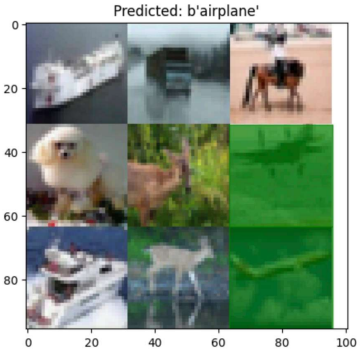
The first experiments training the model resulted in poor performance. The loss didn't drop below 1.9 and the performance on the validation set was ~3/9 correct predictions. After various experiments with learning rates, learning rate schedulers, batch sizes, unfreezing rates, and other hyperparameters, we eventually found the best parameters were as follows:

Learning Rate: .002	Batch Size: 256	Epochs: 4	Unfreeze: 30/165 steps
Lr Step Size: 165	Lr Gamma: 0.8	Learning Scheduler: StepLR	Optimizer: Adam

With these parameters, our results showed validation loss <1.0 and validation accuracy >96%







Shown here is an example of a simulated captcha with the correctly labeled targets highlighted in green.

The graphs of the loss and accuracy show signs of slight overfitting. The train accuracy exceeds the val accuracy by ~3.5% and the val loss is ~.5 greater than the train loss. Our theory is that although each captcha is unique, the individual

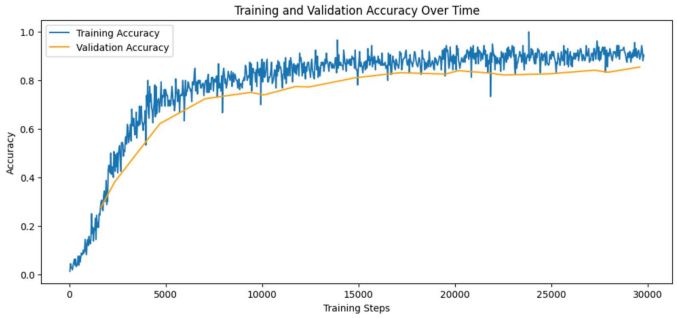
images are repeated 40x throughout the train dataset. This could potentially cause overfitting to the train dataset despite having unique combinations in each of the captchas.

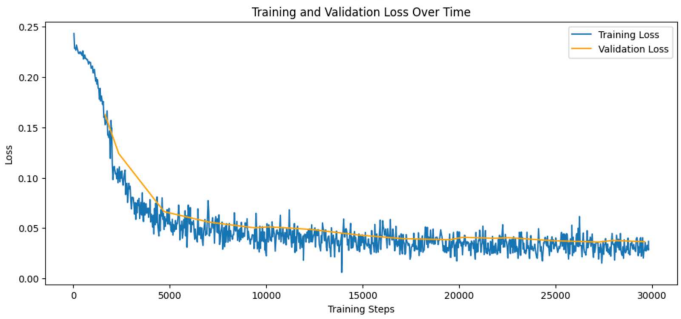
To tackle the text based captchas we tried two different models. We initially thought the ResNet-50 would perform well, as it did for our captchas of 9 images. We also identified the clip-vit-base-patch32 model as a potentially effective model because it is known for being good at image-text matching and retrieval. However, our tests showed that the ResNet significantly outperformed the Clip model. Therefore, we will focus on detailing our implementation of the ResNet-50.

We utilized a similar approach to the previous image based captcha, albeit with slight differences. Namely, we modified the ResNet-50 to output a batch size 5x36 tensor of predictions. This is because each captcha is 5 characters long and there are 26 possible alphabetical characters and 10 possible digits that can be included in each captcha.

One notable issue that we had with this model was lack of training data. The dataset provided by kaggle had only ~1000 images. After doing a train-test split, we were left with only ~800 images to train the model on. The small amount of data led to extreme overfitting and the inability to generalize to the test dataset. To remedy this, we experimented with several data augmentation strategies. First, we tested padding images with whitespace. Our theory was that different sizes of training images would encourage the model to learn a more general solution to the text captcha problem. Second, we cropped the whitespace near the edges of the images and then expanded the images (zoomed in on them). Our hypothesis was that the larger images would help provide even more variety and lead to better generalization than the unaugmented data. With the data augmentation, we were able to double the training data.

The doubling of the data allowed us to experiment with larger batch sizes. The larger batch size combined with a low learning rate seemed to provide a more stable gradient descent and eventually we were able to achieve a validation loss of <.05 with a validation accuracy of >86%





The graphs shown to the side are our results of training with the following hyperparameters:

Learning Rate: 5e-4 Epochs: 15 Batch Size: 32
Optimizer: Adam

It is important to note that these results came from fine-tuning the pretrained weights from the ImageNet. One interesting finding is that despite having more possible categories and less training data, the text based captcha was able to achieve a validation accuracy that was unreachable for the image captcha bot without data augmentation.

In conclusion, we discovered that through fine tuning, a pre-trained ResNet-50 is able to solve both image classification and text based captchas at a high degree of accuracy. This indicates a potential lack of security and developers should probably consider using other ways to limit bot users.