|  |
| --- |
| **Image Captcha Recognition** |

**Xiaoguang Wang Qian Wang Hao Zhang**

A53246911 A53235276 A53234732

xiw503@ ucsd.edu qiw018@ ucsd.edu haz351@ ucsd.edu

**Abstract**

Image captcha has long been used as a common protective tool for some websites to tell computer and human apart. Here, we employed the pre-trained AlexNet and combined with joint training method to build a model that can automatically recognize the numbers and characters on the captcha image and received a recognition accuracy of 96% on the most complex occasion with numbers, upper and lower characters. Also, we tried to borrow some technology from traditional computer vision like Gaussian blur to denoise the captcha images and improved the performance of our recognition pipeline.

**1 Introduction**

Captcha here is an acronym for “**C**ompletely **A**utomatic **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part”. It is useful since computers can hardly read the contents like numbers and characters from images like a human. However, thanks to the fast development in the research of computer vision and especially artificial neural network, machines now can better understand the specific semantic information from images. For example, LeNet[1] has proved great success in handwritten digit recognition.



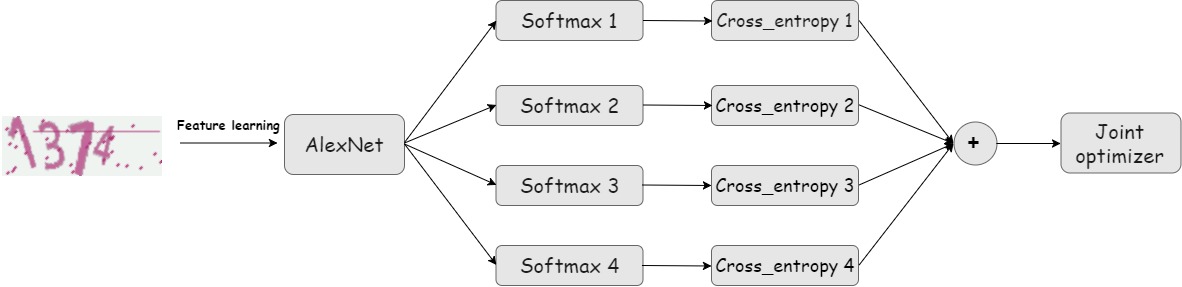
Figure 1: Some examples for captcha images

Inspired from the above ideas and forerunners, we proposed a novel system which can automatically learn useful features from the captcha images and make predictions of the numbers and characters on each digit separately. Figure 2 provides the broad framework for our system.

In this project, we proposed the novel captcha recognition network above and implemented the system on TensorFlow. The main contributions of our work are outlined as follows:

● Modified the existing AlexNet with multiple output layers to make predictions for each digit of captcha code separately and used multi-task learning method to tune the model.

● Combined with some image smoothing methods from traditional computer vision to pre-process the input images to the network and improved performance.

 Figure 2: Network structure for the recognition system (4-digits captcha)

**2 Approach**

In this section, we present some specific description pf the designed network and some implementation details. Specifically, although the method proposed in this project can be easily adapted to any length captcha recognition, we implemented a system for 4-digit here for the convenience to verify the feasibility of our ideas.

**2.1 Model architecture overview**

Figure 2 contains the architecture of the model we built in this project. It is a transformation from traditional AlexNet.

**Feature learning**: We firstly used a pre-trained AlexNet for image feature learning since it has been well trained on a large image dataset and proved to represent images features well with lower dimension feature vectors. We leave the net parameters untouched during the training process. Also, we explored some more modern networks like GoogLeNet, but unluckily, the GPU server seems out of resources to run this large network well. So, we chose AlexNet at last and believe this network can extract necessary features well from the simple captcha images.

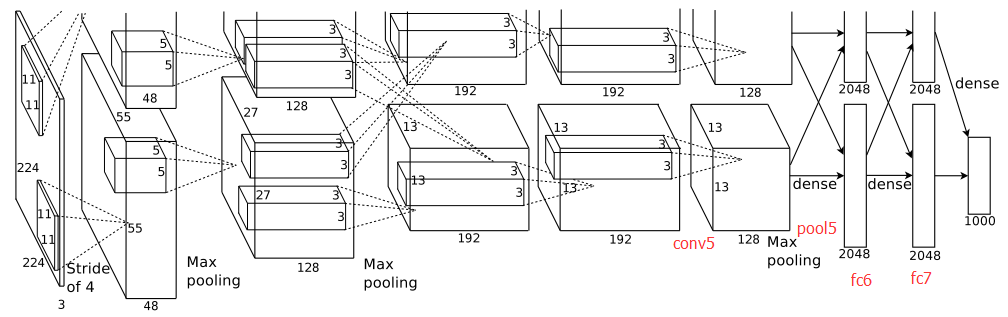


Figure 3: Original AlexNet structure for comparison

**Prediction**: we used one *softmax* layer to predict each digit separately (here we set the captcha length to be 4, so we implemented 4 *softmax* layers). To train this model, we use a method called multi-task learning (we will give specific explanation in the next part).

**2.2 Multi-task learning**

Normally in multiple categories classification problem, we use *softmax* layer to map image features from their feature space to a probability distribution. Then backpropagate the calculated the cross-entropy loss value to tune net parameters. However, here we have an output and get a cross-entropy loss value for each output *softmax* layer.

To handle this problem, we used a method from multi-tasking learning called **joint training**. That is in the training process, instead of backpropagating these loss values four times separately, we backpropagate the sum or the mean value of these loss values and assume that we can get the best prediction if we can minimize this ‘complex loss value’ through some methods like gradient descent.

**3 Experiment**

We built a 4-digit captcha image recognition model to verify the feasibility of our design of the system. Also, the model is implemented with TensorFlow and ran on UCSD DSMLP.

**3.1 Dataset**

We searched the opening datasets on the Internet, but we cannot find a suitable dataset (one at least should have sufficient captcha images and corresponding labels to train our network). But luckily enough, there is a Python library called *captcha* (<https://pypi.org/project/captcha/>), we can use the library to build our own dataset (including training set and test set). Below is a demo of the generated dataset.

Figure 4: Running demo for Python captcha

We can see for each captcha image, we set their file name to be the content on the image, which serve as the supervised information(labels) during the training process. Also, since the dataset is generated and can be controlled strictly on the image format and dataset size, which is very convenient to verify our ideas and model. In this project, we generated 10000 images for each model and used 500 for testing and the rest for training.

**3.2 Evaluation Protocol**

Forimage captcha recognition, what we concern about mostly is the prediction accuracy. So here, we use prediction accuracy on each of the 4 digits as the only metrics to evaluate out models.

More importantly, let’s consider another important problem for our model – robustness from the following two aspects:

● **Complexity of the code**: here, we explored different levels of complexity of the generated captcha code from numbers only (char set size: 10) to numbers + lower case characters (char set size: 36) and lastly the most complex case with numbers + lower and upper case characters (char set size: 62).

● **Noise**: As an improvement to our system, here instead of inputting the generated captcha images directly into the model, we borrowed some technology from traditional computer vision and pre-process the images with some noise cancellation operations like Gaussian blue and median blur. We compared the effect of image smoothing on our model.

Figure 5: Captcha image before and after noise cancellation

Explanation on Gaussian blur

**3.3 Results**

In this part, we will present results we get on our model with different char set complexity: numbers only, numbers + lower case characters, number + lower + upper case characters and a comparison between the results of with or without smoothing operation.

**3.3.1 numbers only**

We only used digits from 0 to 9 to compose a 4-digits captcha image. Below are the accuracies on test set and mean loss value on training set we get during the training process.

Figure 6: Accuracies and loss value during the training process (numbers only)

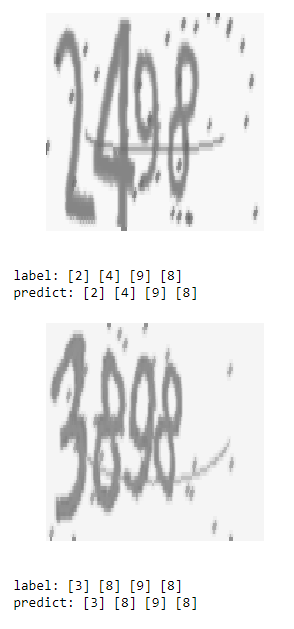
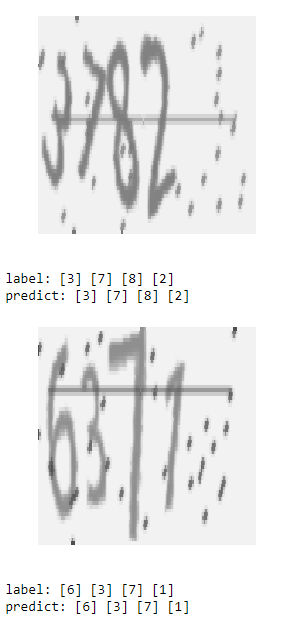
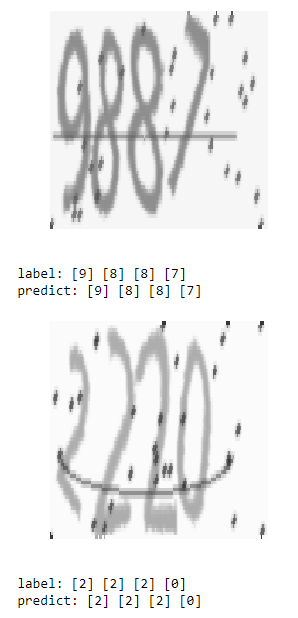


Figure 7: Test demos for image captcha recognition (numbers only)

**3.3.2 numbers + lower case characters**

Here, we used digits from 0 to 9 and lower-case characters from ‘a’ to ‘z’ to compose a 4-digits captcha image. Below are the accuracies on test set and mean loss value on training set we get during the training process.

Figure 8: Accuracies and loss value during the training process (numbers + lower case)

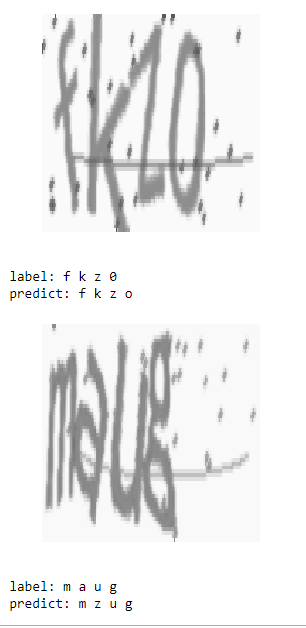
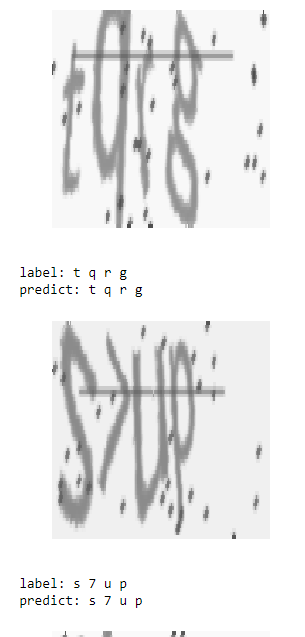
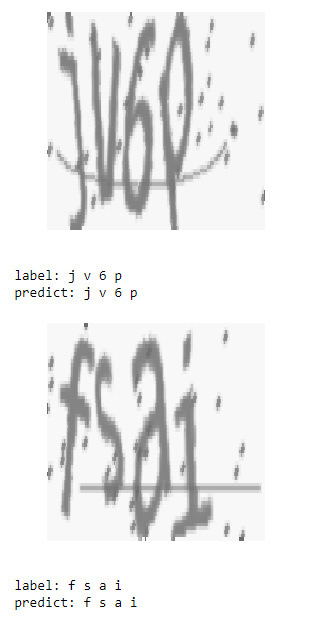


Figure 9: Test demos for image captcha recognition (numbers + lower case)

**3.3.3 numbers + lower and upper case characters**

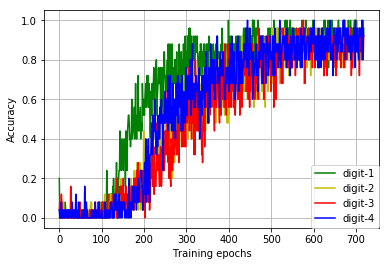
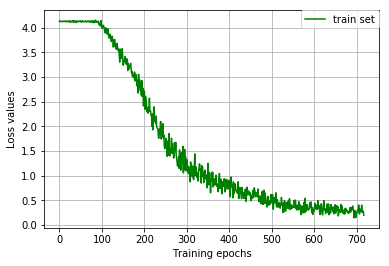
Here, we used digits from 0 to 9 and lower-case characters from ‘a’ to ‘z’ and upper case characters from ‘A’ to ‘Z’ to compose a 4-digits captcha image. Below are the accuracies on test set and mean loss value on training set we get during the training process.

Figure 10: Accuracies and loss value during the training process

(numbers + lower, upper case)

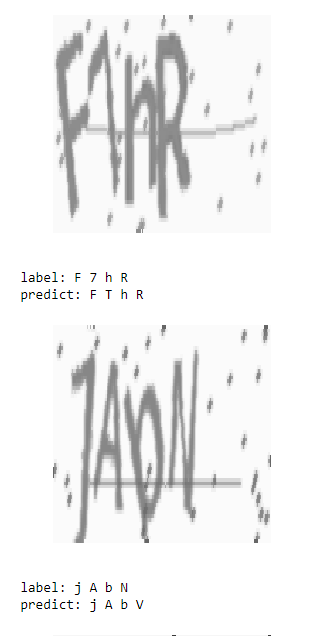
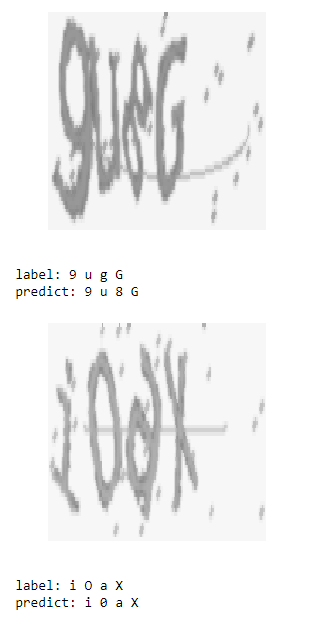
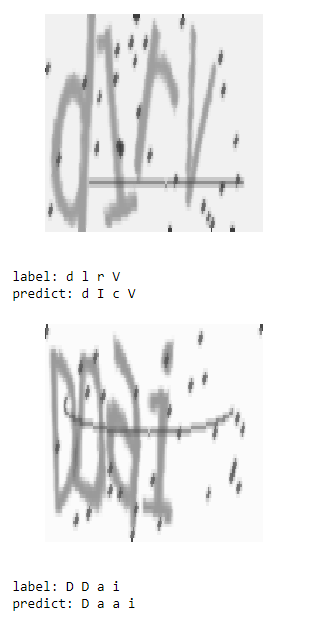


Figure 11: Test demos for image captcha recognition (numbers + lower, upper case)

**4 Conclusions and Discussions**

These instructions apply to everyone, regardless of the formatter being used.

**4.1 Citations within the text**

Citations within the text should be numbered consecutively. The corresponding number is to appear enclosed in square brackets, such as [1] or [2]-[5]. The corresponding references are to be listed in the same order at the end of the paper, in the **References** section. (Note: the standard BibTeX style unsrt produces this.) As to the format of the references themselves, any standard reference style is acceptable, as long as it is used consistently.

As submission is double blind, refer to your own published work in the third person. That is, use "In the previous work of Jones et al. [4]", not "In our previous work [4]". If you cite your other papers that are not widely available (e.g. a journal paper under review), use anonymous author names in the citation, e.g. an author of the form "A.Anonymous".

**5 Reference**

[1] Yann LeCun. (Proc. IEEE 1998) Gradient-Based Learning Applied to Document Recognition.

[2] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. (2012) ImageNet Classification with Deep Convolutional Neural Networks.