Neural Net CAPTCHA Cracker
by
Geetika Garg

Advisor: Dr. Chris Pollett
Committee Members: Dr. Thomas Austin and Mr. James Casaletto
Agenda

• Introduction to Project
• Preliminary Work Summary
• Our Approach
• Neural Networks
• Experiments
• Results
• Conclusion
• Demo
Introduction to Project

• Tried decoding CAPTCHAs using deep neural networks.

• CAPTCHA: Completely automated Public Turing test to tell Computers and Humans Apart.

• Helps to distinguish between humans and computers.

• First mentioned in a paper by Moni Naor [1] in 1996
Characteristics of a CAPTCHA:

• Easy for a human to decode
• Difficult for a Computer to recognize.
Why Decode CAPTCHAs??

- AI Problem
- Security Breach

CAPTCHAAs are critical for security on internet, if they are no more secure, our system won’t be secure and we would have to think of alternatives.
Preliminary Work Summary
● Breaking CAPTCHAs is not a new concept.
● Mori and Malik [2] have broken EZ-Gimpy (92% success) and Gimpy (33% success) CAPTCHAs with sophisticated object recognition algorithms.
● People [3] also have used the following approach for CAPTCHA recognition:
  ○ Preprocessing.
  ○ Segmentation.
  ○ Training the model for individual character recognition.
  ○ Generating sequence with highest probability.
Problems in Segmentation

- Segmentation is difficult as some digits could be overlapping with some other digits.
- Deformity of digits is also a major concern. For example, digit “2” can have a larger loop or just a cusp.
- Character orientation. Characters could be rotated at arbitrary angles making recognition difficult.
- Unknown scale of characters. We do not know how big a character would be. So, it is not known how big the segmentation boxes should be.
Idea behind our approach

- Yann Le Cun used neural networks for handwritten digits recognition in 1990[6].
- Google has published a paper[4] in which they used convolutional neural networks to detect home addresses from street view home plate images.
- One more paper from Google[5] in which they used recurrent neural network to generate caption for an image.

We tried combining these ideas!
Our Approach
Our Approach

- **End to End model.** Systems with multiple modules following conventional approach tend to behave poorly, because each module is optimized independently and the errors between modules compound. We learned an end to end model that predicts directly from pixels.

- **Convolutional neural network** for Image features and,

- **Recurrent neural networks** for generating output sequence.
Neural Networks
Brief History of Neural Network

- Started way back in 1940’s.
- Became unpopular in 1960’s
- Regained popularity in 1980’s
- Recently have become one of the hottest areas in the field of machine learning.
- Applications involve face recognition used by Facebook, Image captioning used by Google etc.
What are Neural Networks

- Inspired from human brain

A simple neural network model

- Many input units and one output unit.
- The inputs are scaled with weights on which an activation function is applied to get the output.
Multilayer neural network
Training Neural Networks

- Backpropagation algorithm.
- The problem is set up as minimization of a loss (objective) function.
- Weights are adjusted using gradients and learning rate.
- Gradients are computed using simple derivative chain rule.
Convolutional Neural Networks are a special kind of multi-layer neural networks.

In 1995, Yann LeCun Et al. introduced the concept of convolutional neural networks [7].

Convolutional Neural Networks are designed to recognize visual patterns directly from pixel images with no preprocessing.
Convolutional layer

Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.

Source pixel

Convolution kernel (emboss)

New pixel value (destination pixel)

\[
\begin{align*}
(4 \times 0) & \\
(0 \times 0) & \\
(0 \times 0) & \\
(0 \times 1) & \\
(0 \times 1) & \\
(0 \times 0) & \\
(0 \times 1) & \\
+ (-4 \times 2) & \\
\end{align*}
\]

\[-8\]
Feature Extraction

- Shared weights: The same filter weights are applied to all the pixels.
- It helps in detecting the same feature at different locations of an image.
- This reduces the number of parameters to be learned.
For Instance, if image size is 200*50, filter size is 5*5 and if there are 32 filters, we have only $32*(5*5 +1)$ (1 for bias) = 832 weights to learn.

- Otherwise it would have been number_of_pixels*number_of_pixels*filters, which would be $200*50*200*50*32 = 3.2$ million.

This is several orders of magnitude larger than what we have in CNNs.
Maxpool Layer

- Typically used after a CNN layer.
- Takes maximum of neighbouring pixels.
- Helps in rotational and translational invariance.
Recurrent Neural Networks

• Feedforward networks accept only fixed sized input and give output of fixed length, whereas RNNs can work with variable length inputs and outputs.
• In RNNs, connections between units have a directed cycle.
• Various applications of RNNs include handwriting recognition and speech recognition.
• “Long Short Term Memory” networks are special kinds of RNNs.
• **Vanishing** and **exploding gradient** makes RNNs difficult to train.
• But in LSTMs, the error gets trapped in memory.
LSTMs

- Uses tanh \( \left( \frac{e^x - e^{-x}}{e^x + e^{-x}} \right) \) activation function.
- Forward and backward LSTMs
Softmax Layer

- Used for generating a probability distribution.
- Used typically in classification problems
- In our model, we use it to estimate probability of every character.
Framework and Dataset
Frameworks Used

- We have used Theano.
- It is a publicly available, flexible library which optimizes, and evaluates mathematical expressions efficiently.
- It is available in Python so was easy to integrate with our project.
- It makes use of GPUs if present making tasks faster.
Frameworks Used

- **Lasagne** is a Python package to train neural networks. It uses Theano internally.
- It implements LSTM. Theano by itself does not have implementation of LSTMs.
- It implements the framework to keep track of all the neural network parameters like weights and biases. It makes it easy to save the parameters and initializes the model with pre-trained weights.
Dataset Used

- Training requires lots of images.
- A standard dataset for CAPTCHAs is not available publicly.
- We generated dataset synthetically.
- Java module to generate CAPTCHAs with randomization of noise, characters and backgrounds.
- We generated fixed as well as variable length CAPTCHA dataset.
- 1 million simple images, 2 million complex images of fixed length(5), and 13 million images of variable length.
A dataset consists of images with the following characteristics:

- An image contains 4-7 characters, if it is a variable length dataset, or 5 characters if it is a fixed length captcha.
- A character could be A-Z, a-z, or 0-9.
- All the images generated are of the same size, i.e., (200*50).

The images include:

- Complex image
- Simple image
Experiments
Convolution layers

- Input image was converted to single channel using “L” conversion.
- Number of filters used - 32
- Size of a filter - 5*5
- Zero padding
- Maxpooling done over 2*2
2 CNN layers to learn image features.
A dense layer and a softmax layer to predict every character.
Softmax layers at the top share weights.
Variable length model

2 CNNs to learn image features.
RNN to generate output sequence.
The same dense layer vector was inputted for every step of RNN.
Training

- Batch Size 1024 (images).
- **Gradient clipping:** We think that exploding gradients were pulling the model too far in different directions. Gradient clipping gave better results which might have helped in dampening the effect of exploding gradients.
- Special id 0 for ‘unk’ was used to signal termination.
- Images of different captcha length were batched together for randomization.
- High learning rates caused instability. So had to gradually decrease it, as the models would get stuck with high learning rates.
<table>
<thead>
<tr>
<th>Type of model</th>
<th>Individual Character Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM fixed length (simple dataset)</td>
<td>99.9%</td>
</tr>
<tr>
<td>LSTM fixed length (complex dataset)</td>
<td>98.48%</td>
</tr>
<tr>
<td>Multiple Softmax fixed length (simple dataset)</td>
<td>99.8%</td>
</tr>
<tr>
<td>Multiple Softmax fixed length (complex dataset)</td>
<td>98.96%</td>
</tr>
<tr>
<td>LSTM variable length with fixed length data</td>
<td>99.5%</td>
</tr>
<tr>
<td>LSTM variable length with variable length data</td>
<td>97.31%</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>Type of model</th>
<th>Sequence Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM fixed length (simple dataset)</td>
<td>99.8%</td>
</tr>
<tr>
<td>LSTM fixed length (complex dataset)</td>
<td>91%</td>
</tr>
<tr>
<td>Multiple Softmax fixed length (simple dataset)</td>
<td>99%</td>
</tr>
<tr>
<td>Multiple Softmax fixed length (complex dataset)</td>
<td>96%</td>
</tr>
<tr>
<td>LSTM variable length with fixed length data</td>
<td>98%</td>
</tr>
<tr>
<td>LSTM variable length with variable length data</td>
<td>81%</td>
</tr>
</tbody>
</table>
Human (Me): 3/10 were wrong
Computer: 1/10 was wrong

Human 2: 2/10 were wrong
Computer: 0/10 were wrong

Human 3: 4/15 were wrong
Computer: 2/15 were wrong
Graphs Generated

- **Training Loss**
  - Graph showing training loss against the number of images.
  - Data来自文件：_training_loss_lstm_website.csv

- **Sequence Accuracy**
  - Graph showing sequence accuracy against the number of images.
  - Data来自文件：_training_acc_lstm_website.csv
Variable length model variants
Conclusion
Deep neural networks showed a really good performance in decoding CAPTCHAs with 80% and 99.8% accuracy for variable and fixed length CAPTCHAs respectively.

CAPTCHAs are not more secure as computers can do better than humans.
Future Work
Future Work

- Will try to work on accuracy using more convolutional layers.
- Will make the system robust by increasing the variety in training data.
Project Links

Demo website:
http://cp-training.appspot.com/

GITHUB:
https://github.com/bgeetika/Captcha-Decoder/
References


QUESTIONS?
THANK YOU !!