**Teaching Machines to Recognize CAPTCHA**

**Jusin1), Wilbert Harriman2) , Robin3)**

1Fakultas Ilmu Komputer, Universitas Pelita Harapan

E-mail: [jsnmpc8@gmail.com](mailto:jsnmpc8@gmail.com)1)

2Department of Computer Science, National Tsing Hua University

E-mail: [wilbert@datalab.cs.nthu.edu.tw2](mailto:wilbert@datalab.cs.nthu.edu.tw2))

3Fakultas Ilmu Komputer, Universitas Pelita Harapan

E-mail: robin80huang@gmail.com3)

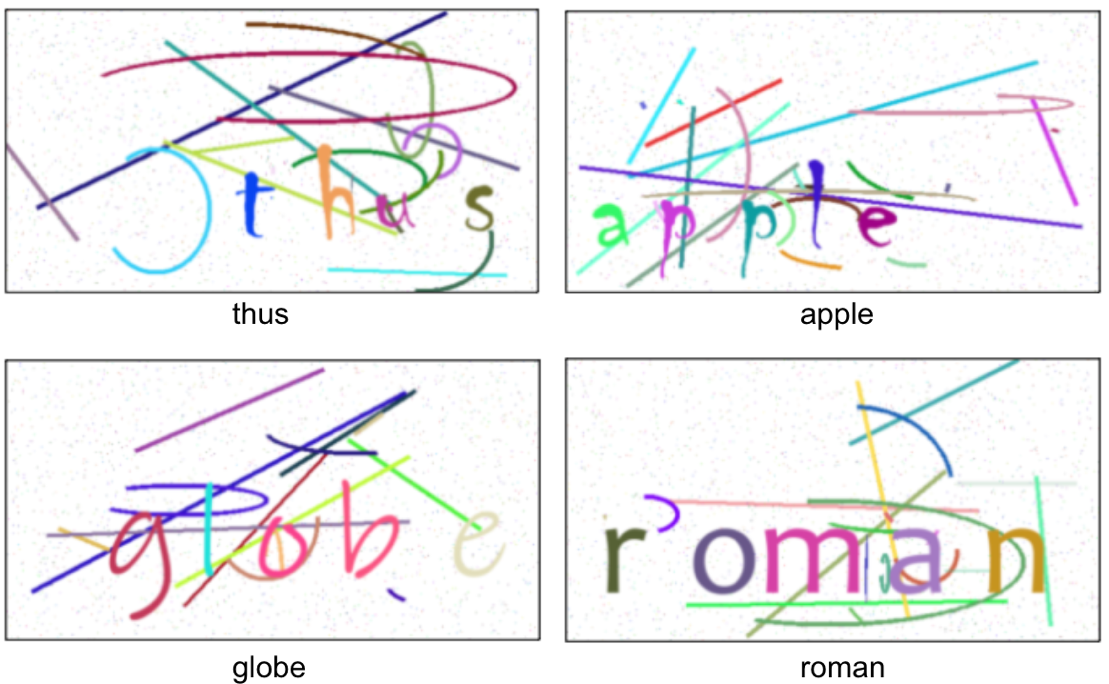
*Abstract – Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) is widely used as a security measure against spam and bot attacks via the Internet. CAPTCHA works by the assumption that it takes human sensory and cognitive skills (that are not present in computers) to successfully identify objects or letters within a noisy graphical environment. In this work, we propose a way to teach machines to recognize CAPTCHAs with deep learning. Our deep learning model uses a Convolutional Neural Network (CNN) encoder to convert CAPTCHA images into vector representations, followed by a Recurrent Neural Network (RNN) decoder to convert vector representations into text. Our model is able to achieve a validation accuracy of 90% after about an hour of training. Code is available at https://github.com/wilbertharriman/tf2-attention-captcha-recognizer.*

***Keywords****: CAPTCHA, deep learning, neural network, supervised learning*

*Abstrak – Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) telah digunakan secara luas sebagai sebuah acuan dalam melawan serangan spam dan bot melalui Internet. CAPTCHA bekerja dengan berasumsi bahwa sensori dan kognitif manusia dibutuhkan (dimana hal ini tidak dimiliki oleh komputer) agar bisa mengenal objek atau tulisan yang terdapat dalam sebuah lingkungan yang memiliki derau (noise) dengan baik dan benar. Kajian ini mengajukan sebuah cara untuk mengajari mesin untuk mengenal CAPTCHA dengan deep learning. Model deep learning ini menggunakan Convolutional Neural Network (CNN) encoder untuk mengkonversi citra CAPTCHA menjadi representasi vektor, kemudiaan dilanjutkan dengan menggunakan Recurrent Neural Network (RNN) decoder untuk mengkonversi representasi vector menjadi tulisan. Model ini mampu mencapai ketelitian validasi hingga 90% setelah dilakukan training selama 1 jam. Kode program tersedia pada alamat URL https://github.com/wilbertharriman/tf2-attention-captcha-recognizer.*

**Kata Kunci** : *CAPTCHA, deep learning, neural network, supervised learning*

**INTRODUCTION**

CAPTCHA is a widely used verification method to differentiate humans from machines. CAPTCHAs can be divided into three groups based on how they interact with users: text-based, image-based, and sound-based. Our work focuses on image-based CAPTCHAs, the most common type of CAPTCHA. A typical image-based CAPTCHA involves the user seeing a noisy image, then identifying the letters in the image in the correct order. Before the advent of deep learning, computers were unable to process and learn from images in the same way as humans could, which is why image-based CAPTCHAs have been so successful. In recent years, convolutional neural networks (CNNs) has become the standard method for learning to recognize patterns from images, while recurrent neural networks (RNNs) have been shown to work well with sequential data such as text or speech. These advances motivated the combined use of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to ****generate high quality captions from images. Recent work proposed an attention-based model that learns to describe the content of images by learning to focus on salient objects while generating the corresponding words in the output sequence [2].

Motivated by recent works, we aim to apply current advances in image captioning to build a model that is able to replicate the human ability to solve image-based CAPTCHAs. We will use “text CAPTCHA” to refer to image-based CAPTCHAs that display text.

In this work, we describe how to combine convolutional neural network (CNN) and attention-based recurrent neural network (RNN) to recognize letters in text CAPTCHAs.

Figure 1. Text CAPTCHAs and their corresponding labels

**METHODS**

**1. Data Collection**

Training a deep learning model requires a large dataset. For our model, we need our training dataset to be in the form of

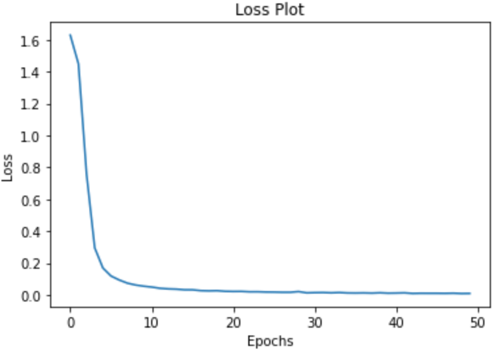
<captcha image, label> tuples (see Fig. 1). Our dataset consists of 120k labeled

text CAPTCHAs for training and validating, as well as 20k unlabeled text CAPTCHAs for testing.

**2. Encoder**

We need an encoder to transform images into meaningful vector representations. Our encoder takes 50x100x3 (height \* width \* channel) RGB images as inputs and turn each image into a 3D tensor of size 3x6x256. Our encoder uses convolutional neural network architecture inspired by AlexNet [4].

**3. Attention-based Decoder**

The purpose of our decoder is to output letters (a-z) one at a time corresponding to the letters in the input CAPTCHA images. Our decoder uses gated recurrent units (GRU) to avoid the vanishing gradient and catastrophic forgetting problems of vanilla recurrent neural networks (RNN). Our decoder uses Bahdanau attention mechanism [3] to help the decoder focus on the relevant parts of the image when outputting letters.

**4. Training**

We split the 120k labeled training set into 100k for training and 20k for validation.

To train our model, we compute the crossentropy loss between our prediction (the output of our decoder) and the true label. Our error value is then used to update our model through backpropagation. We also use teacher forcing [5] to train our decoder. Finally, we train our model for 50 epochs.

**5. Validation**

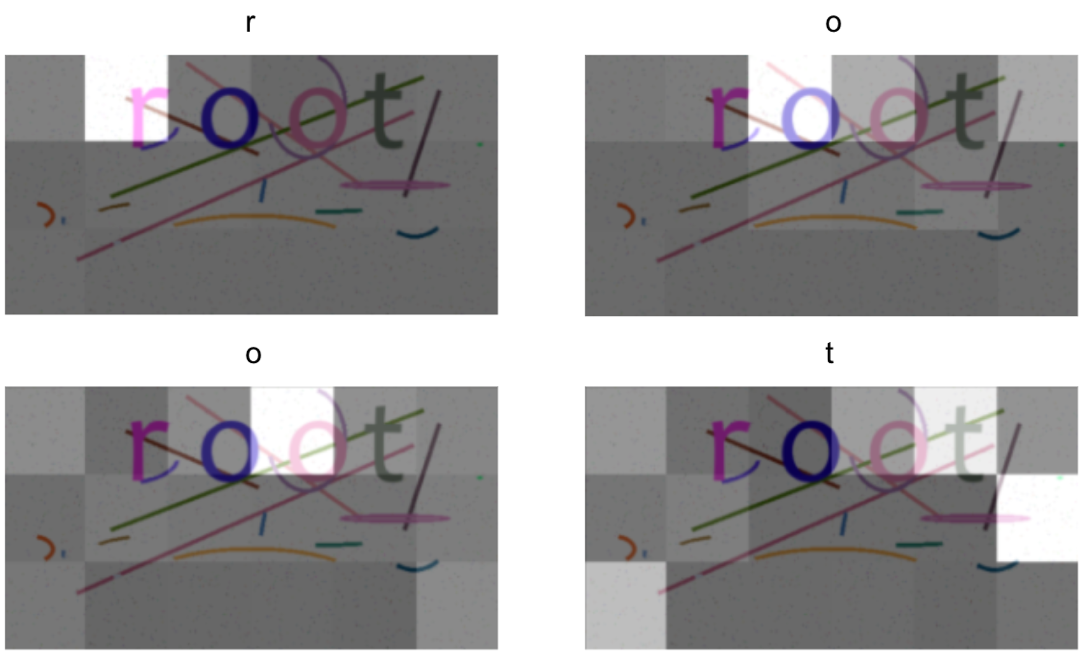
To test our model, we use images from our validation set and compare the output of our model to the label of the validation

set. Our prediction is correct only if the whole word matches the label.

*Figure 2.* Our loss plot after 50 epochs of training



*Figure 3.* Samples from unlabeled text CAPTCHA.

******

*Figure 4.* As the model outputs each letter, its attention changes to reflect the relevant parts of the text CAPTCHA.

**RESULTS & DISCUSSION**

After training for 50 epochs, we plot our loss function for every epoch (see Fig. 2). We see steep decline in our loss function in the first few epochs and stable decline afterward. This indicates that our model is learning well.

In the validation step, our model achieves a validation accuracy of 90.675% which demonstrates its capability to recognize text CAPTCHAs.

When we test our model with unlabeled dataset and sample 10 results, we obtain the following results: **[toe, that, kerry, type, mba, line, von, means, pink, kids]**.

We then verify the sampled results through human labor (see Fig. 3). We find that all 10 predictions align with human perception. This result is remarkable given that not only is the machine able to ignore the noise in text CAPTCHAs, it manages to recognize various fonts within the text CAPTCHAs even though it had no concept of how letters look.

Next, we seek to find out more about how the model works. By visualizing the attention component learned by our model, we can see that our model solves CAPTCHA exactly how a human would (see Fig. 4). Our attention plot shows that the machine is able to focus on the relevant parts of the image when making prediction for each letter.

**CONCLUSION**

In this work, we train a model that is able to correctly identify text in a text CAPTCHA with 90% accuracy. We also show that conventional text CAPTCHA is no longer a good way to tell humans and computers apart.

**REFERENCES**

[[1]GoodFellow, I., Bengio, Y., Courville, A., (2016), Deep Learning. MIT Press]([1]GoodFellow, I., Bengio, Y., Courville, A., (2016), Deep Learning. MIT Press[2]https://proceedings.mlr.press/v37/xuc15.pdf[3]https://arxiv.org/pdf/1409.0473.pdf[4]https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf[5] https://towardsdatascience.com/what-is-teacher-forcing-3da6217fed1c)

[[2]https://proceedings.mlr.press/v37/xuc15.pdf]([1]GoodFellow, I., Bengio, Y., Courville, A., (2016), Deep Learning. MIT Press[2]https://proceedings.mlr.press/v37/xuc15.pdf[3]https://arxiv.org/pdf/1409.0473.pdf[4]https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf[5] https://towardsdatascience.com/what-is-teacher-forcing-3da6217fed1c)

[[3]https://arxiv.org/pdf/1409.0473.pdf]([1]GoodFellow, I., Bengio, Y., Courville, A., (2016), Deep Learning. MIT Press[2]https://proceedings.mlr.press/v37/xuc15.pdf[3]https://arxiv.org/pdf/1409.0473.pdf[4]https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf[5] https://towardsdatascience.com/what-is-teacher-forcing-3da6217fed1c)

[[4]https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf]([1]GoodFellow, I., Bengio, Y., Courville, A., (2016), Deep Learning. MIT Press[2]https://proceedings.mlr.press/v37/xuc15.pdf[3]https://arxiv.org/pdf/1409.0473.pdf[4]https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf[5] https://towardsdatascience.com/what-is-teacher-forcing-3da6217fed1c)

[[5] https://towardsdatascience.com/what-is-teacher-forcing-3da6217fed1c]([1]GoodFellow, I., Bengio, Y., Courville, A., (2016), Deep Learning. MIT Press[2]https://proceedings.mlr.press/v37/xuc15.pdf[3]https://arxiv.org/pdf/1409.0473.pdf[4]https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf[5] https://towardsdatascience.com/what-is-teacher-forcing-3da6217fed1c)

**SOURCE CODE**

ENCODER **(**Convolutional Neural Network encodes captcha images into simpler representation**)**

class CNN\_Encoder(tf.keras.Model):

def \_\_init\_\_(self, embedding\_dim):

super(CNN\_Encoder, self).\_\_init\_\_()

layers = [

Conv2D(96, (3, 3), padding='valid', activation='relu', kernel\_initializer='he\_uniform'),

BatchNormalization(),

MaxPool2D((2, 2), padding='valid'),

Conv2D(128, (3, 3), padding='same', activation='relu', kernel\_initializer='he\_uniform'),

BatchNormalization(),

Conv2D(256, (3, 3), padding='same', activation='relu', kernel\_initializer='he\_uniform'),

MaxPool2D((2, 2), padding='valid'),

Conv2D(384, (3, 3), padding='same', activation='relu', kernel\_initializer='he\_uniform'),

BatchNormalization(),

MaxPool2D((2, 2), padding='valid'),

Conv2D(384, (3, 3), padding='same', activation='relu', kernel\_initializer='he\_uniform'),

BatchNormalization(),

MaxPool2D((2, 2), padding='valid'),

Conv2D(256, (3, 3), padding='same', activation='relu', kernel\_initializer='he\_uniform'),

BatchNormalization(),

Dropout(0.5)

]

self.cnn = tf.keras.models.Sequential(layers)

self.fc = tf.keras.layers.Dense(embedding\_dim, activation='relu')

def call(self, x):

x = self.cnn(x)

x = tf.reshape(x, (x.shape[0], -1, x.shape[3]))

x = self.fc(x)

return x

DECODER **(**Output from attention layer is fed into GRU cells followed by FC layers to convert encoded captchas into strings**)**

class RNN\_Decoder(tf.keras.Model):

def \_\_init\_\_(self, embedding\_dim, units, vocab\_size):

super(RNN\_Decoder, self).\_\_init\_\_()

self.units = units

self.embedding = tf.keras.layers.Embedding(vocab\_size, embedding\_dim)

self.gru = tf.keras.layers.GRU(self.units,

return\_sequences=True,

return\_state=True,

recurrent\_initializer='glorot\_uniform')

self.fc1 = tf.keras.layers.Dense(self.units)

self.fc2 = tf.keras.layers.Dense(vocab\_size)

self.attention = BahdanauAttention(self.units)

def call(self, x, features, hidden):

context\_vector, attention\_weights = self.attention(features, hidden)

x = self.embedding(x)

x = tf.concat([tf.expand\_dims(context\_vector, 1), x], axis=-1)

output, state = self.gru(x)

x = self.fc1(output)

x = tf.reshape(x, (-1, x.shape[2]))

x = self.fc2(x)

return x, state, attention\_weights

def reset\_state(self, batch\_size):

return tf.zeros((batch\_size, self.units))