DeCaptcha

By: Matthew Gualino, Jeremy Wang

Our Problem

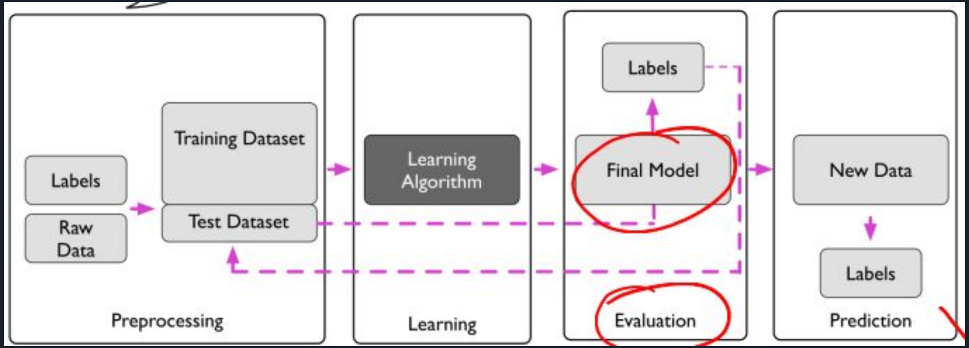
The goal is for this model is to be able to read text from a "Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA)." The dataset I will be using can be found here: Link

Captchas are a pain in the butt to solve (We get it wrong 29% of the time, Stanford), so why don't we go and make an AI to automatically solve these problems with just a copy & paste?

Our Process

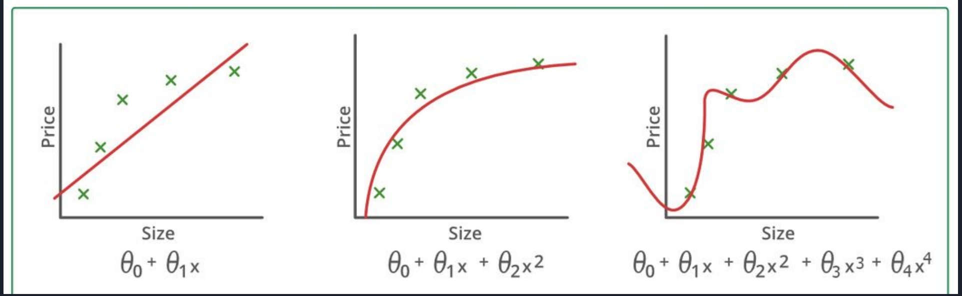
We have split the task up into 3 parts:

- Data processing
- Training & Validation
- Prediction



Some lingo:

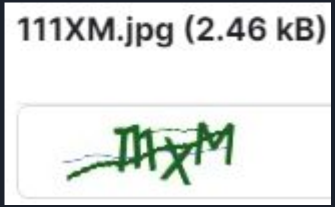
Overfitting:



Bootstrapping : It is any test or metric that uses random sampling with replacement, and falls under the broader class of resampling methods ($1/e \approx .368$)

Epoch: When every training sample is processed, you do epoch multiple times to improve accuracy.

Data Preprocessing (V1)



111XM.jpg (2.46 kB)

Each image is 150x40 pixels. The images consist of texts with many different colors, with the text being listed in the title.

Our data currently is consisted of 113,000 images, and we will be converting this into a csv for ease of use.

We are planning to quantize each image's pixel in a format similar to this:

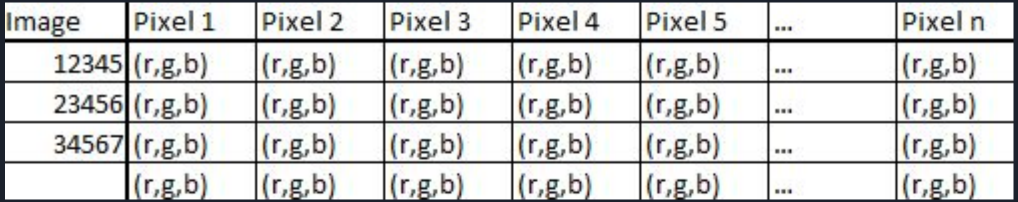
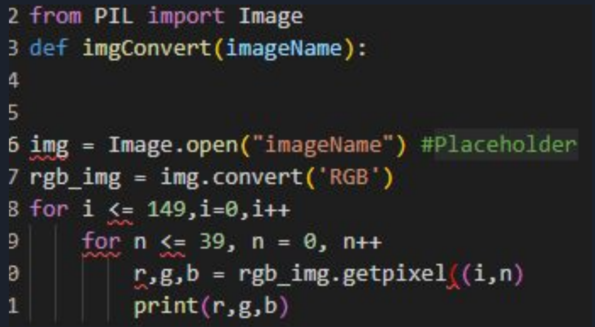


Image	Pixel 1	Pixel 2	Pixel 3	Pixel 4	Pixel 5	...	Pixel n
12345	(r,g,b)	(r,g,b)	(r,g,b)	(r,g,b)	(r,g,b)	...	(r,g,b)
23456	(r,g,b)	(r,g,b)	(r,g,b)	(r,g,b)	(r,g,b)	...	(r,g,b)
34567	(r,g,b)	(r,g,b)	(r,g,b)	(r,g,b)	(r,g,b)	...	(r,g,b)
	(r,g,b)	(r,g,b)	(r,g,b)	(r,g,b)	(r,g,b)	...	(r,g,b)

A little roadblock



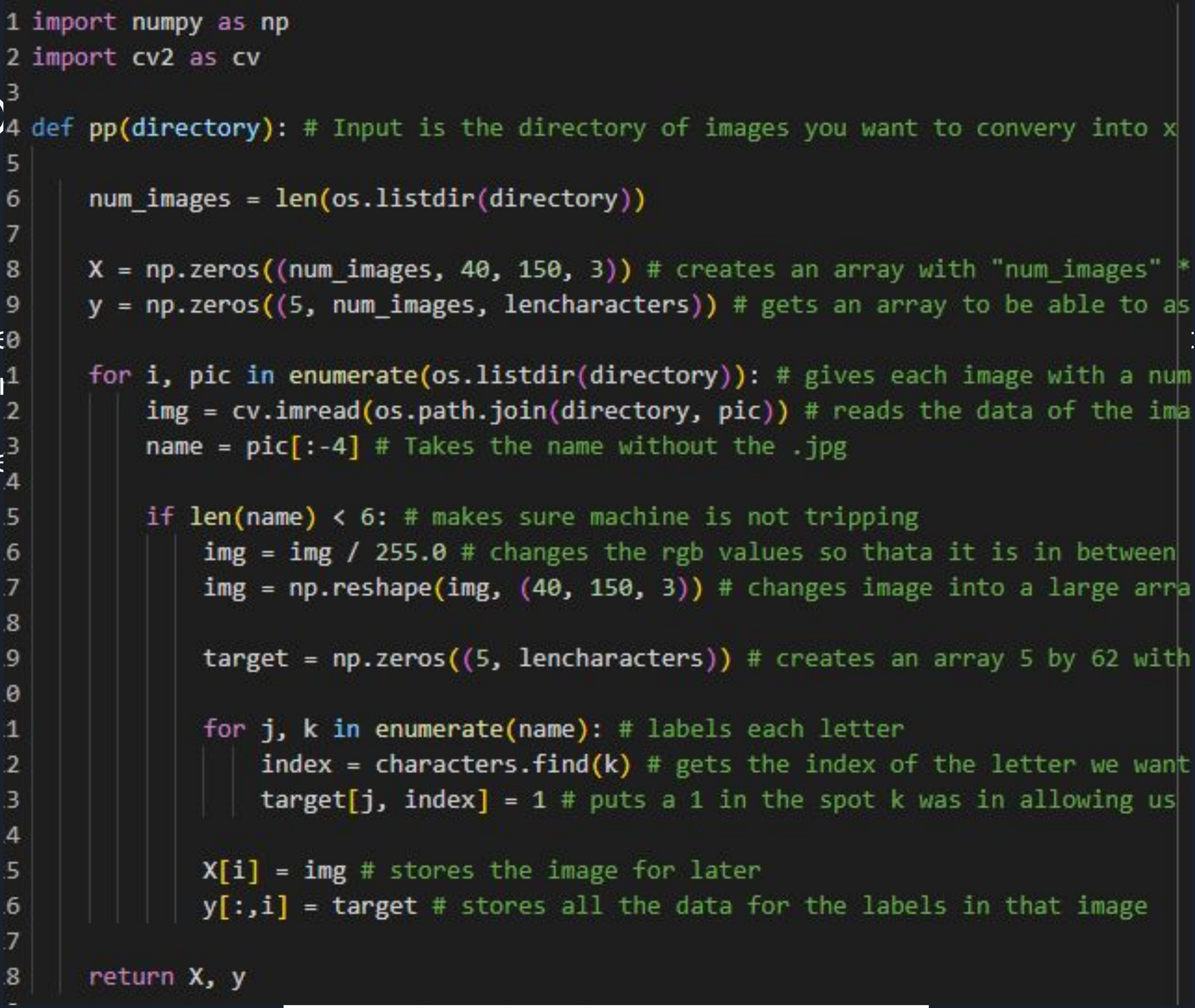
```
from PIL import Image
def imgConvert(imageName):


img = Image.open("imageName") #Placeholder
rgb_img = img.convert('RGB')
for i <= 149,i=0,i++
    for n <= 39, n = 0, n++
        r,g,b = rgb_img.getpixel((i,n)
        print(r,g,b)
```

We decided that instead of bootstrapping we would manually split the images 36% of the data into the validation folder.

We use the OS library's listdir and OpenCV imread methods to fix this issue and to further speed up our preprocessing speeds.

Additionally, we were planning on training our model on the captchas themselves but we realized that we'd need to train the model individually

D



```python
import numpy as np
import cv2 as cv

def pp(directory): # Input is the directory of images you want to convery into x

    num_images = len(os.listdir(directory))

    X = np.zeros((num_images, 40, 150, 3)) # creates an array with "num_images" *
    y = np.zeros((5, num_images, lencharacters)) # gets an array to be able to as

    for i, pic in enumerate(os.listdir(directory)): # gives each image with a num
        img = cv.imread(os.path.join(directory, pic)) # reads the data of the ima
        name = pic[:-4] # Takes the name without the .jpg

        if len(name) < 6: # makes sure machine is not tripping
            img = img / 255.0 # changes the rgb values so thata it is in between
            img = np.reshape(img, (40, 150, 3)) # changes image into a large arra

            target = np.zeros((5, lencharacters)) # creates an array 5 by 62 with

            for j, k in enumerate(name): # labels each letter
                index = characters.find(k) # gets the index of the letter we want
                target[j, index] = 1 # puts a 1 in the spot k was in allowing us

            X[i] = img # stores the image for later
            y[:,i] = target # stores all the data for the labels in that image

    return X, y
```

We :he model

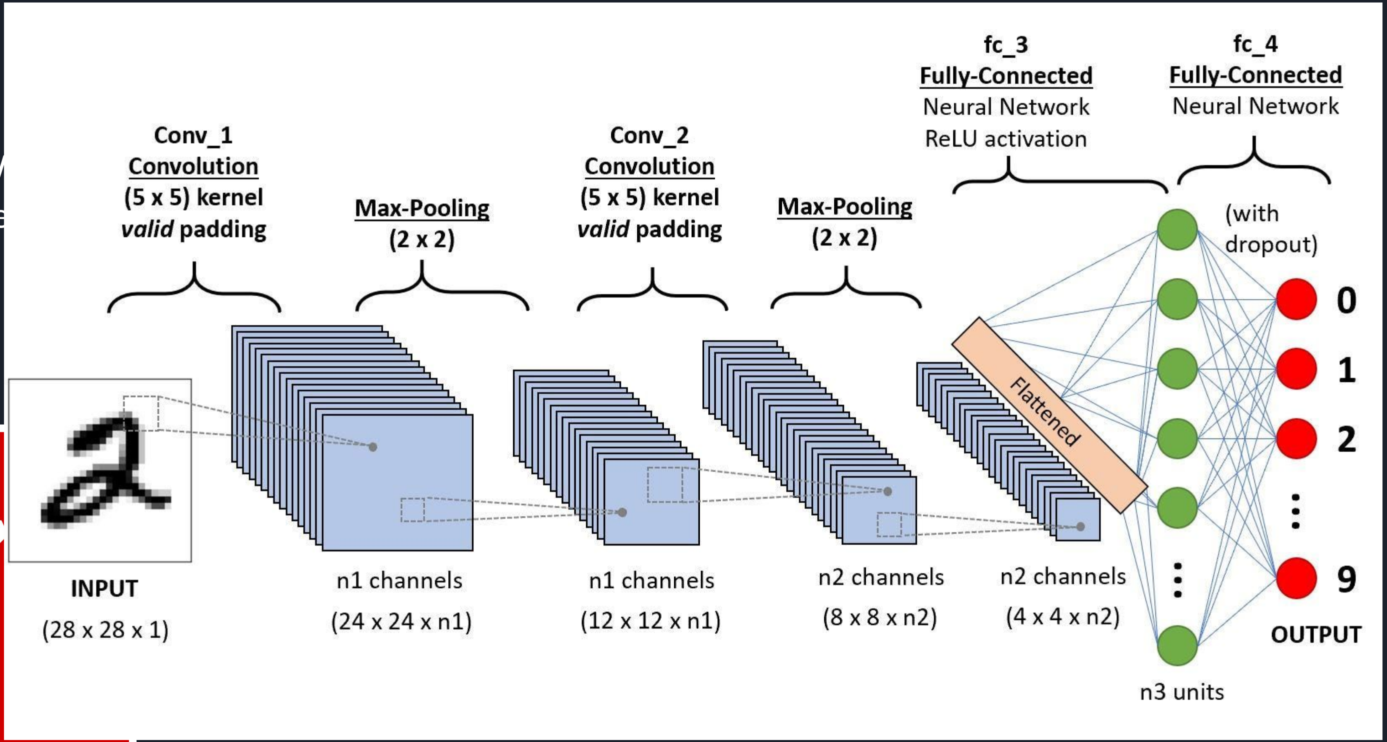
to r

We

Training & Validation

W

be



Results of Training

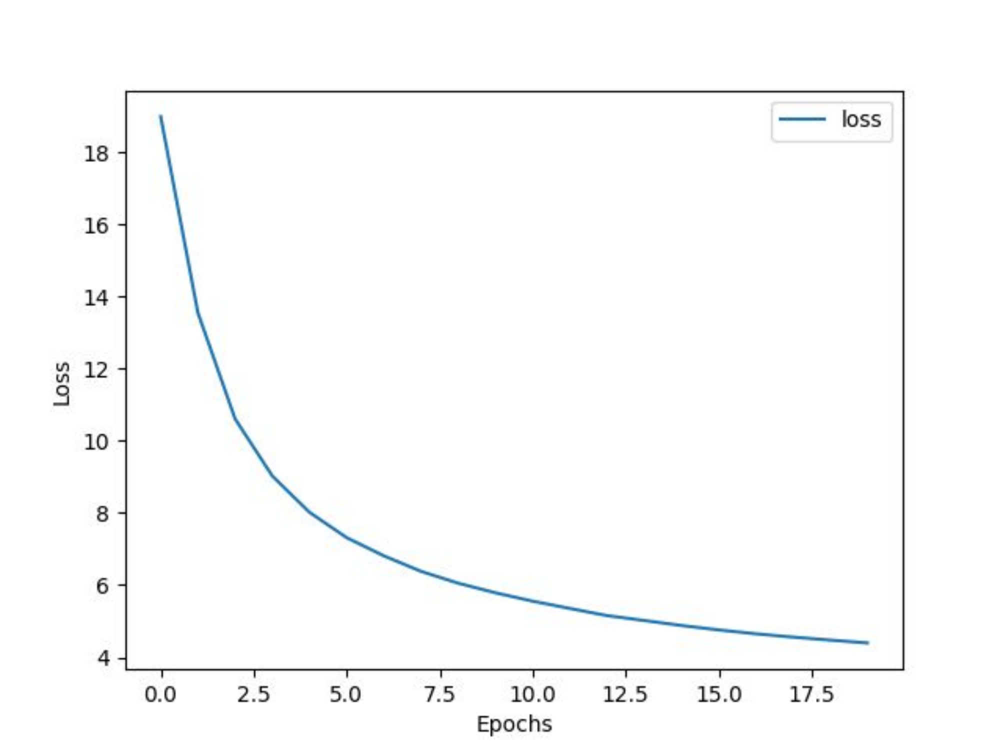
On our first model we started with doing 32 Batches, 20 Epochs, and our optimizer was ADAM. With this model we had a 79.53% success rate (That’s a B baby!)

On our third model we used 128 Batches, 20 Epochs, and our optimizer was RMSProp, and we changed the validation to 20%. With this model we had roughly a 94% success rate.

On our fourth model we used 128 Batches, 20 Epochs, and our optimizer was RMSProp, we changed the validation to 20%, and this time we used weight decay at .2% to prevent overfitting. With this model we had achieved a 97.2% success rate.

Some Graphs:

Model 4



Our Final Product!

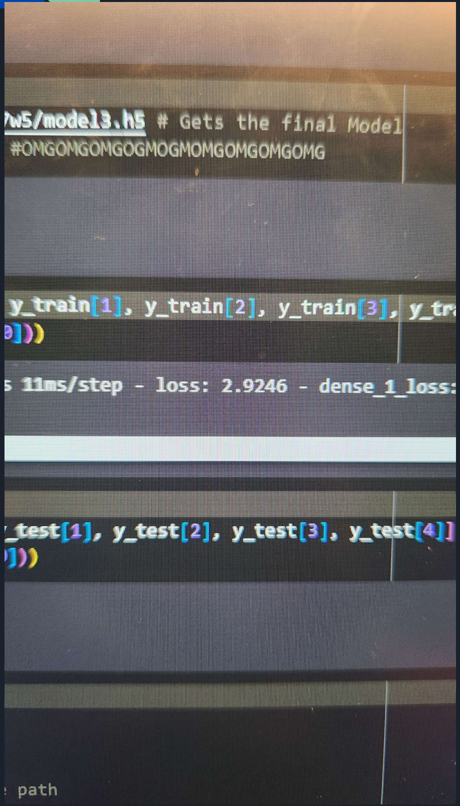
Sources:

https://www.kaggle.com/code/yassineghouzam/introduction-to-cnn-keras-0-997-top-6

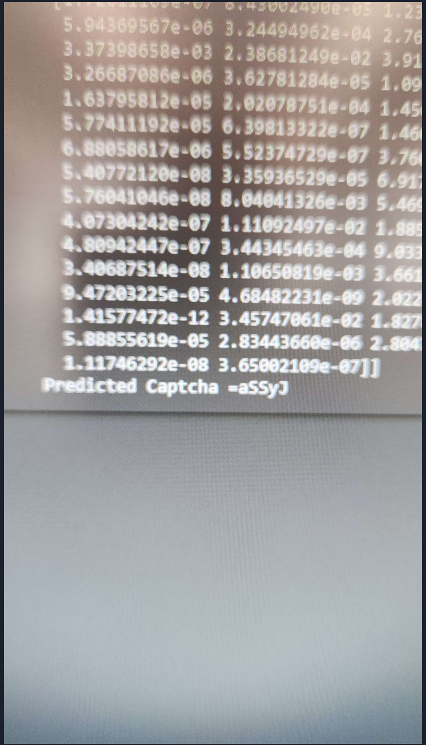
https://medium.com/@manvi./captcha-recognition-using-convolutional-neural-network-d191ef91330e

burszstein_2010_captcha.pdf (stanford.edu)

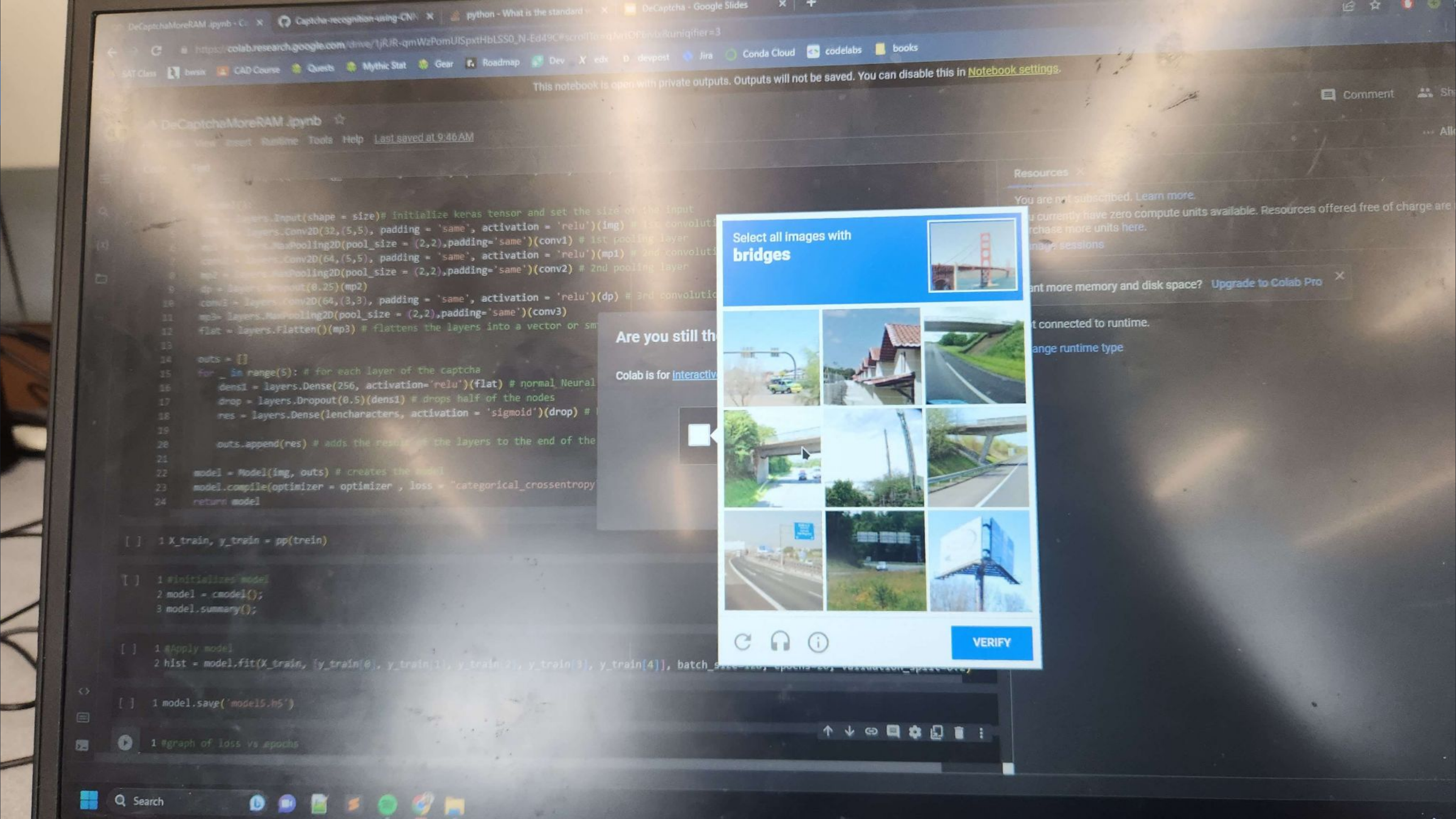
Spent wayy too long tryna get this done



```
/w5/model3.h5 # Gets the final Model
 #OMGOMGOMGOGMOGMOMGOMGOMGOMG

 y_train[1], y_train[2], y_train[3], y_tr
0]))

s 11ms/step - loss: 2.9246 - dense_1_loss:

_test[1], y_test[2], y_test[3], y_test[4]]
]))

: path
```



```
   3.43002490e-05 1.23
 5.94369567e-06 3.24494962e-04 2.76
 3.37398658e-03 2.38681249e-02 3.91
 3.26687086e-06 3.62781284e-05 1.09
 1.63795812e-05 2.02078751e-04 1.45
 5.77411192e-05 6.39813322e-07 1.46
 6.88058617e-06 5.52374729e-07 3.76
 5.40772120e-08 3.35936529e-05 6.91
 5.76041046e-08 8.04041326e-03 5.46
 4.07304242e-07 1.11092497e-02 1.88
 4.80942447e-07 3.44345463e-04 9.03
 3.40687514e-08 1.10650819e-03 3.661
 9.47203225e-05 4.68482231e-09 2.022
 1.41577472e-12 3.45747061e-02 1.827
 5.88855619e-05 2.83443660e-06 2.804
 1.11746292e-08 3.65002109e-07]]
Predicted Captcha =aSSyJ
```



DeCaptchaMoreRAM.ipynb - Co
Captcha-recognition-using-CNN
python - What is the standard
DeCaptcha - Google Slides
colab.research.google.com/drive/1jRJR-qmWzPomUlSpxtHbLSS0_N-Ed49C#scrollTo=qAnOPbvu&uniqifier=3
SAT Class
bwsix
CAD Course
Quests
Mythic Stat
Gear
Roadmap
Dev
edx
devpost
Jira
Conda Cloud
codelabs
books
Comment
Tools
Help
Resources
Search

This notebook is open with private outputs. Outputs will not be saved. You can disable this in Notebook settings.

DeCaptchaMoreRAM.ipynb

Last saved at 9:46 AM

```
img = layers.Input(shape = size)# initialize keras tensor and set the size of the input
conv1 = layers.Conv2D(32,(5,5), padding = 'same', activation = 'relu')(img) # 1st convoluti
mp1 = layers.MaxPooling2D(pool_size = (2,2),padding='same')(conv1) # 1st pooling layer
conv2 = layers.Conv2D(64,(5,5), padding = 'same', activation = 'relu')(mp1) # 2nd convoluti
mp2 = layers.MaxPooling2D(pool_size = (2,2),padding='same')(conv2) # 2nd pooling layer
dp = layers.Dropout(0.25)(mp2)
conv3 = layers.Conv2D(64,(3,3), padding = 'same', activation = 'relu')(dp) # 3rd convolutio
mp3= layers.MaxPooling2D(pool_size = (2,2),padding='same')(conv3)
flat = layers.Flatten()(mp3) # flattens the layers into a vector or sm

outs = []
for _ in range(5): # for each layer of the captcha
    dens1 = layers.Dense(256, activation='relu')(flat) # normal Neural
    drop = layers.Dropout(0.5)(dens1) # drops half of the nodes
    res = layers.Dense(lencharacters, activation = 'sigmoid')(drop) #

    outs.append(res) # adds the result of the layers to the end of the

model = Model(img, outs) # creates the model
model.compile(optimizer = optimizer , loss = "categorical_crossentropy
return model
```

```
X_train, y_train = pp(trein)
```

```
#initialize model
model = cmodel();
model.summary();
```

```
#Apply model
hist = model.fit(X_train, [y_train[0], y_train[1], y_train[2], y_train[3], y_train[4]], batch_
```

```
model.save('model5.h5')
```

```
#graph of loss vs epochs
```

Are you still th

Colab is for interactiv

Select all images with
bridges

VERIFY

You are not subscribed. Learn more.

u currently have zero compute units available. Resources offered free of charge are

rchase more units here.

nage sessions

ant more memory and disk space? Upgrade to Colab Pro

t connected to runtime.

ange runtime type