

[placeholder]

An unholy union of Steganography and ML text generation

```
Python 3.6.12 [AMD64, Intel(R) Core(TM) i7-8565U CPU @ 2.80GHz] (default, Sep 9 2020, 08:29:25) [AMD64 v1.10.0 64-bit (AMD64)]
Type "help()" for more information.

Python 3.6.12 -- An enhanced interactive Python.

In [1]:

In [1]: runfile('C:/Users/Krishnasai Addala/Desktop/placeholder/placeholder.py', wdir='C:/Users/Krishnasai Addala/Desktop/placeholder')
Out[1]:
Using TensorFlow backend.
[11:45:10] Downloading image streams to c:/Users/Krishnasai Addala/Desktop/placeholder/placeholder.py...
[11:45:10] Downloading image streams to c:/Users/Krishnasai Addala/Desktop/placeholder/placeholder.py...
Total number of characters: 441
Total words: 24
Total patterns: 541

2020-11-24 14:45:10.447408Z: tensorflow/core/platform/cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX AVX2

test

:: Welcome to [placeholder] ::
1. Encode
2. Decode
3. train model
4. exit

:
```

Krishnasai Addala

2010110842

BTech CSE first year

INTRODUCTION

Placeholder involves encoding text into PNG images by way of steganography.

The text can be entered by the user or generated by the computer by a relatively shallow nlp model.

The model is merely a proof of concept trained for a limited number of cycles on a very

small dataset. This is due to the processor intensive nature of training such a model.

Motivation

The original purpose of this project was to potentially encode passwords into images that could be circulated more freely than standard password texts. This is due to the fact that the image is useless to anybody who doesn't already know about the encoded text.

The nlp model was implemented to give the user the option to generate a 'strong' password from a model trained on a diet of such examples. However the training data in this case thus far is merely some poetry, owing to the difficulty of obtaining caches of strong passwords.

Technologies used

The language used for the project was python 3.6.

The libraries used were

Numpy

Nltk

Sys

Keras

Pillow (PIL)

Tensorflow

The use of tensorflow and keras necessitated the creation of an environment using anaconda navigator, which in turn necessitated the use of an ide that plays well with anaconda (in this case, spyder).

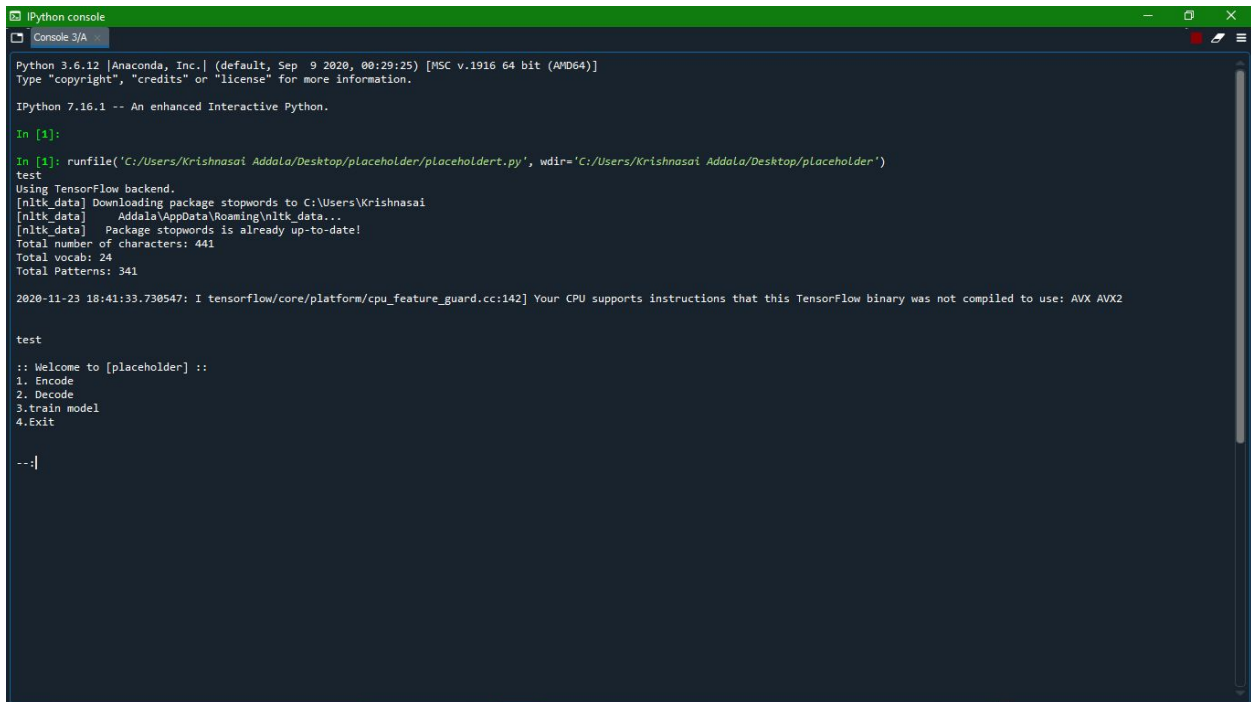
Limitations of the program

[placeholder] can only encode and decode data for .PNG images.

The image names cannot have a '.' except in the extension, due to the way the program extracts the image name from the extension.

The program is trained on a small data set consisting of 2 paragraphs of the poem 'The Raven'. This was done to save time in training as the poem results in a relatively small vocabulary which speeds up the time per epoch.

Screenshots



```
Python 3.6.12 [Anaconda, Inc.] (default, Sep  9 2020, 00:29:25) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.

IPython 7.16.1 -- An enhanced Interactive Python.

In [1]:
In [4]: runfile('C:/Users/Krishnasai Addala/Desktop/placeholder/placeholder.py', wdir='C:/Users/Krishnasai Addala/Desktop/placeholder')
test
Using TensorFlow backend.
[nltk_data] Downloading package stopwords to C:/Users/Krishnasai
[nltk_data] Addala\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!
Total number of characters: 441
Total vocab: 24
Total Patterns: 341

2020-11-23 18:41:33.738547: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX AVX2

test

:: Welcome to [placeholder] ::
1. Encode
2. Decode
3.train model
4.Exit

--:]
```

Start up screen

```
IPython console
Console 3/A

2020-11-23 18:41:33.738547: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX AVX2

test

:: Welcome to [placeholder] ::
1. Encode
2. Decode
3.train model
4.Exit

--:1

Enter image name(with extension) : Capture.PNG
to enter your own data for encoding enter 1
to use computer generated text,enter 2

--:1

Enter data to be encoded : example text
Enter the name of new image(with extension) : egimg01.PNG

:: Welcome to [placeholder] ::
1. Encode
2. Decode
3.train model
4.Exit

--:2

Enter image name(with extension) : egimg01.PNG
Decoded Word : example text

:: Welcome to [placeholder] ::
1. Encode
2. Decode
3.train model
4.Exit

--:]
```

encoding user given input into an image

```
IPython console
Console 7/A

Python 3.6.12 [Anaconda, Inc.] (default, Sep 9 2020, 00:29:25) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.16.1 -- An enhanced Interactive Python.

In [4]: runfile('C:/Users/Krishnasai Addala/Desktop/placeholder/placeholder.py', wdir='C:/Users/Krishnasai Addala/Desktop/placeholder')

test
Using TensorFlow backend.
[nltk_data] Downloading package stopwords to C:/Users/Krishnasai
[nltk_data] Addala\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!
Total number of characters: 441
Total vocab: 24
Total Patterns: 341

2020-11-23 19:03:23.352513: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX AVX2

test

:: Welcome to [placeholder] ::
1. Encode
2. Decode
3.train model
4.Exit

--:1

Enter image name(with extension) : Capture.PNG
to enter your own data for encoding enter 1
to use computer generated text,enter 2

--:2

Random Seed:
" pping rapping chamber door tis visitor muttered tapping chamber door nothing ah distinctly remember "
bleak december sepaaste yyigg ebeer w rrrgr r ooorbleak december sepaaste yyigg ebeer w rrrgr r oooro

this is the data to be encoded
to generate something new, enter 1
|
```

encoding computer generated text into an image

```
IPython console
Console 7/A

Enter the name of new image(with extension) : egimg02.PNG

:: Welcome to [placeholder] ::
1. Encode
2. Decode
3.train model
4.Exit

--:1

Enter image name(with extension) : Capture.PNG

to enter your own data for encoding enter 1

to use computer generated text,enter 2

--:2
Random Seed:
" w vainly sought borrow books surcease sorrow sorrow lost lenore rare radiant maiden angels name leno "
re nameless evermoreeerrrpn ggggaaaaemrrrddoooioore nameless evermoreeerrrpn ggggaaaaemrrrddoooioo

this is the data to be encoded
to generate something new, enter 1
1
please wait a moment
oooi dossilnnccl errrrrr mmeerrrleerrrrr rp
Enter the name of new image(with extension) : egimg04.PNG

:: Welcome to [placeholder] ::
1. Encode
2. Decode
3.train model
4.Exit

--:2

Enter image name(with extension) : egimg02.PNG
Decoded Word : bleak december sepaate yyigg ebeer w rrgp r oooro

:: Welcome to [placeholder] ::
1. Encode
2. Decode
3.train model
4.Exit
```

decoding the text from an image

```
IPython console
Console 8/A

2020-11-23 19:11:16.395558: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX AVX2

test

:: Welcome to [placeholder] ::
1. Encode
2. Decode
3.train model
4.Exit

--:3
Epoch 1/20
341/341 [=====] - 60s 177ms/step - loss: 0.5498

Epoch 00001: loss improved from inf to 0.54977, saving model to model_weights_saved.hdf5
Epoch 2/20
341/341 [=====] - 20s 59ms/step - loss: 0.5970

Epoch 00002: loss did not improve from 0.54977
Epoch 3/20
341/341 [=====] - 19s 57ms/step - loss: 0.7759

Epoch 00003: loss did not improve from 0.54977
Epoch 4/20
341/341 [=====] - 18s 54ms/step - loss: 0.5110

Epoch 00004: loss improved from 0.54977 to 0.51104, saving model to model_weights_saved.hdf5
Epoch 5/20
341/341 [=====] - 19s 54ms/step - loss: 0.5717

Epoch 00005: loss did not improve from 0.51104
Epoch 6/20
341/341 [=====] - 18s 54ms/step - loss: 0.3422

Epoch 00006: loss improved from 0.51104 to 0.34223, saving model to model_weights_saved.hdf5
Epoch 7/20
341/341 [=====] - 18s 54ms/step - loss: 0.5039

Epoch 00007: loss did not improve from 0.34223
Epoch 8/20
341/341 [=====] - 19s 54ms/step - loss: 0.3362

Epoch 00008: loss improved from 0.34223 to 0.33621, saving model to model_weights_saved.hdf5
Epoch 9/20
```

training the model

```
IPython console
Console 8/A

Epoch 00015: loss improved from 0.23974 to 0.22921, saving model to model_weights_saved.hdf5
Epoch 16/20
341/341 [=====] - 19s 55ms/step - loss: 0.2183

Epoch 00016: loss improved from 0.22921 to 0.21835, saving model to model_weights_saved.hdf5
Epoch 17/20
341/341 [=====] - 19s 55ms/step - loss: 0.2152

Epoch 00017: loss improved from 0.21835 to 0.21524, saving model to model_weights_saved.hdf5
Epoch 18/20
341/341 [=====] - 19s 54ms/step - loss: 0.2222

Epoch 00018: loss did not improve from 0.21524
Epoch 19/20
341/341 [=====] - 19s 55ms/step - loss: 0.1922

Epoch 00019: loss improved from 0.21524 to 0.19222, saving model to model_weights_saved.hdf5
Epoch 20/20
341/341 [=====] - 18s 54ms/step - loss: 0.2316

Epoch 00020: loss did not improve from 0.19222

:: Welcome to [placeholder] ::
1. Encode
2. Decode
3.train model
4.Exit

--:4
exiting program

test

In [2]: |
```

Exiting program

How the encoding works

Every byte of data is converted to its 8-bit binary code using ASCII values. Then pixels are read from left to right in a group of 3 containing a total of 9 values. The first 8-values are used to store binary data. The value is made odd if 1 occurs and even if 0 occurs.

For example, in binary the letter a is 01100001, so an example of the possible encoding of the first 3 digits would be (256,255,255).

How the decoding works

To decode, three pixels are read at a time, till the last value is odd, which means the message is over. Every 3-pixels contain a binary data, which can be extracted by the same encoding logic. If the value is odd the binary bit is 1 else 0.

How the machine learning works (to the best of the author's knowledge)

{Once upon a midnight dreary, while I pondered, weak and weary,

Over many a quaint and curious volume of forgotten lore,

While I nodded, nearly napping, suddenly there came a tapping,

As of someone gently rapping, rapping at my chamber door.

"Tis some visitor," I muttered, "tapping at my chamber door

Only this, and nothing more."

Ah, distinctly I remember it was in the bleak December,

And each separate dying ember wrought its ghost upon the floor.

Eagerly I wished the morrow;—vainly I had sought to borrow

From my books surcease of sorrow—sorrow for the lost Lenore

For the rare and radiant maiden whom the angels name Lenore

Nameless here for evermore.}--> the reference data is as such

First the program loads in the reference text and converts it to lowercase. Then using NLTK, the processed data is tokenized and words that exist in the stopwords list (words that the nltk devs have determined hold little value in training such models.

Then the list of words are stored as key-value pairs with numbers in a dictionary. Then the list of inputs is iterated through and characters are converted to numbers in order to create where each sequence starts with the next character in the input data

Then the input sequences can be processed using `numpy.reshape()` and further converted into floats to play nice with the network.

Then the program can one-hot encode the label data using `np_utils` library

Then the model can be constructed. A sequential model using dropout to prevent over training and a densely connected final layer can be constructed as shown in the code.

After compiling the model and saving the weights the model is ready to output text.

The trained model needs to be provided with a random seed. Based on this seed, the model can predict what comes next and convert this output into characters.

Further work

Further developing the project may involve a better data set of strong passwords, larger training batch sizes, further reversible image processing that can be done to add more steps in the decoding process to potentially improve security.

References

Geeksforgeeks

Stackabuse