LaTeX Project Report COMP 5412

Jacob Davis
HKUST

Abstract

The project showcases the use of deep learning methods to decrypt classical cryptography, mainly ciphers, and outlines a tool for interpreting these scrambled messages. As classical cryptography is no longer used, this project is interesting for the theory, implementation and famous historical examples.

Traditional cracking methods almost entirely involve feature analysis, which motivates the utily of deep learning approaches for generating feature embeddings. However, cryptography has a large focus on long range dependancies, commonly making use of permutaions, which is an active area of research in deep learning. This gives an interesting application to investigate the theory which has potential use in speech recognition, character recognition and image classification.

1. Introduction

An encryption function takes natural language string (plaintext) and produces a scrambled string (ciphertext). A decryption function is the inverse of this encryption function. The aim of these functions is so that the scrambled data can only be interpreted by someone who knows the decryption function.

The puzzle is therefore to decrypt the ciphertext into plaintext without the decryption function.

Historically, there has been lots of demand for encryption and equally for decryption without the key. Notable examples include the Enigma machine used by the Germans during World War II, which was eventually cracked by Alan Turing and his team at Bletchley Park. Another example is the Caesar cipher, which was used by Julius Caesar to encode his messages. Additionally, the famous Zodiac Killer used various ciphers in his letters to the police, some of which have still not been fully deciphered.

Traditionally, decryption is done by feature analysis. The most famous feature being letter frequencies. Statistical measures can of the ciphertext and comparing this to a large coprus can give you information hidden in the ciphertext. For example, the letter ‘e’ is the most common letter and therefore the most common character in a ciphertext is often an ’e’. Similar features like n-gram frequencies or repeated words can be used to further predict the plaintext. Therefore, automating finding these patterns in the data is an interesting application of data science.

The limitating factor in decryption is that, in theory, the ciphertext could map to any plaintext. The choice for possible decryption functions is the power set of the representation of the plaintext (consider the set of bi-jective functions between sequences of letters). However, in practice, the complexity of the function is limited due to ease of use and power of the tools available. For example, a cipher wheel makes it easy to encode but simple to decrypt. That is why modern cryptography creates a line between obfusication and computationally secure. Where any ciphertext can map to any plaintext. This results in the field of deniable cryptography; just because its possible for it to map to a plaintext doesn’t mean that it is computationally feasible. [1] Therefore we can consider a tool that produces the most likely plaintext. For this reason we disregard computationally secure algorithms (algorithms that are proven to take an unreasonable amount of time to solve). In this case, the power of the decryption tools available are unlimited (limited above what is reasonable).

Another challenge with decryption is that random noise is often added to function, be it accidental, depreciated language (e.g. old English) or a purposeful attempt to confuse the decrypter (which is quite a common counter to frequency analysis).

1.1. Related Work and Contribution

For classification, the common approach is to apply statistical measures on ciphertexts. These measures are variable based on the encryption and different languages have different measures too. Some measures doing change when applied to the plaintext and the cipher text. Given a feature space of n statistical methods, you can classify the data using trational clustering and classifying techniques. SVM have been proven to be successful. [2] Here we exploit the benefit of deep learning’s ability to learn its own relations in the data.

For solving, traditional methods include brute force and statistical feature proposal. Using deep learning provides an advantage over brute force as it has ability to account for noise and slight variations. Also as one ciphertext can map to many plaintexts, deep learning gives us a measure for most probable plaintext. Brute force has an unnecessary complexity which we can aim to reduce with deeep learning. Another thing is that it does not generalise well to different classes, we need a whole new brute force model for each cipher class. Statistical feature proposal also doesn't account very well for noise. However, the main disadvantage with statistical feature proposal is how to come up with features, which generally requires expert knowledge of the field. This project removes the need for field experts and historians.

1.2. Theory

Permutation transformations currently are very hard for Deep learning techniques to learn due to the transforms having underlying long range dependencies. Permutaion invariance is curently an active area of research [3] which, this project is reliant on. Therefore the paper investigates how to approach this problem for this specific application.

This area is interesting for its multitude of transferable use cases. Speech recognition is essentially a noisy substitution. Where not everyone sounds the same (different encryptions) but a single person makes the same sounds in different situations (bijectiveness). The information is founded in the similarity between different 'Characters' not in their symbolic meaning.

Other examples include handwriting; different styles look different but have characterise similarity within the same handwriting. There are more use cases on other dimensions than the character level, for example, pixel wise permutation invariant image recognition. There is lots of demand for more robust deep learning techniques. [4]

With the boom of natural language tools, being able to add constraints to applications is more and more useful. The current difficulty and area of interest comes from constraints that have no semantic meaning and it is interesting to investigate how to get the model to understand these constraints.

2. Data

In this project only English plaintext will be considered but the project outline should make it possible to extend to other languages.

The project will use a generated dataset for training and testing. Also, real life examples will be used for qualitative testing.

The generated dataset will come from encrypting a corpus of short English passages.

The dataset will undergo cleaning which will lose some information but make the implementation more clear and the algorithm will learn easier. The data will consist of lower case letters and spaces only. The effect of removing, encrypting or leaving the spaces will be investigated.

Random noise will be added to the data to increase the robustness of the model. Noise will consist of replacing characters with a random character or removing a random character. Given a character, there will be p1 probability it is replaced and p2 probability it is removed.

2.1. Raw Data

In order to give a complete analysis the project uses two differing natural language corpuses. The first is bookcorpus [5] : a list of texts taken from Ebooks. The second is generics_kb [6] : a list of sentences taken from wikipedia and other large corpuses. The bookcorpus is generally longer sections with multiple sentences, the mean length being 67 characters and a standard deviation of 45. The mean length of generics_kb is 52 and a standard deviation of 24.

It is unclear at first whether longer or shorter data entries will produce better results. On one hand, features have more information to investigate and cryptography is much more robust with longer sentences. However, longer entries are harder for deep learning models to converge on and take more computing power.

2.2. Ciphers

The project investigates three classes of cipher. Classing means the same method of encryption was used but the "key" differs for each encryption in the class.It is useful as each class often be considered the same from the veiwpoint of the attacker.

The choice of ciphers include monoalphabetic cipher, polyalpabetic cipher and transposition cipher. These are the general classes of the majority of encryption methods and also gives a varied challenge to investigate.

Substitution Cipher - The most common method of substitution replaces the 26 letters of the alphabet (one letter matches only one other). The key for this cipher is the alphabet shuffled. Then the nth letter of the alphabet maps to the nth letter of the key. It encompasses many common ciphers such as Caesar cipher (DEFGHIJKLMNOPQRSTUVWXYZABC) and Atbash (ZYXWVUTSRQPONMLKJIHGFEDCBA). The key is similar to Cauchy's two line permutation representation.

Vigenere Cipher - The cipher uses a key which is a string of letters of length m. Each letter n in the key corresponds to a substitution cipher whose key is the alphabet shifted to start at n. Then the ciphers from the key are applied to the plaintext in rotation. Therefore, a key of length 1 is a substitution cipher and also notably a key of length the same as the plaintext is a completely secure encryption. The proof is that the there are m! choices of key which each map the plaintext to a different ciphertext. This means in reverse that there exists a key that maps the ciphertext to any plaintext of the same length. There is no way to differentiate between two possible plaintexts. For this reason, we bound the key reasonably.

Column transposition cipher - The plaintext is put into a grid where the columns are the values in the key. The columns are transposed so that they are in order of the key and then the cipher text is outputted.

3. Methods

The project will be split into classification and then solving. Classification is generally considered a solved problem [2] and it makes the solving of each type much easier.

The project will also display a computer vision wrapper for interfacing with images of strings.

3.1. Classification

The project uses a CNN to perform a 2D classification of one-hot encoded vectors. This approach emulates the feature proposal statisticians would have done for traditional methods. The CNN generates a high dimensional feature space and a MLP classifies this.

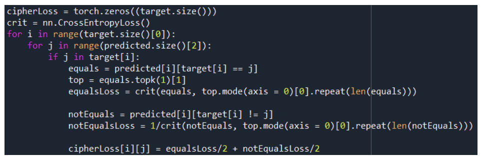
As the information is character level, the one hot encoding serves as the tokeniser. This one-hot encoding is then a 2D map which one can apply convolutions on it. The motivation for convolutions is that the informational dependencies are in the kernel neighbourhood; e.g. positional similarity and character similarity.

3.2. Solving

Solving without the encryption/decryption method involves producing the plaintext from the information in the ciphertext. The encryption/decryption method essentially removes some of the information from the plaintext. However, the missing information can be inferred by using properties of language. The problem statement is then: find a plaintext that is a possible map from the incomplete information in the ciphertext and that makes sense linguistically.

Having done the classification, we can focus on each class of cipher separately, allowing us to better produce an algorithm. This is because each element in a class should be different from each other only by things independent from the problem.

For the substitution cipher it is clear that the applying two ciphers in series is just a different substitution cipher with a different key. Therefore, if we apply our own cipher to the cipher text given, the AI should be able to solve it still. In other words, it doesn't change the problem. Essentially, the value of each i in the ciphertext doesn't matter, what matters is whether it is the same or different from each j in the ciphertext (due to the bi-jectivity). Given a ciphertext we can just label the first character 0 and then the next **unique** character 1 and so on. This is the same problem. For each class we consider similar generalisations.



```
cipherLoss = torch.zeros((target.size()))
crit = nn.CrossEntropyLoss()
for i in range(target.size()[0]):
    for j in range(predicted.size()[2]):
        if j in target[i]:
            equals = predicted[i][target[i] == j]
            top = equals.topk(1)[1]
            equalsLoss = crit(equals, top.mode(axis = 0)[0].repeat(len(equals)))

            notEquals = predicted[i][target[i] != j]
            notEqualsLoss = 1/crit(notEquals, top.mode(axis = 0)[0].repeat(len(notEquals)))

            cipherLoss[i][j] = equalsLoss/2 + notEqualsLoss/2
```

Figure 1. Loss code

3.3. Sequence to sequence

This problem can be thought as a sequence-to-sequence translation, which would be the most general model of the problem and would make the classification redundant. Current success in seq-to-seq translation is in use of transformers [7]. However, as can be seen from the experiments, the transformer approach massively fails to classify the simple substitution cipher. Looking at qualitative results, the transformer manages to recognises the location of "space" but doesn't understand that each character maps bi-jectively to a character. It does have a sense of natural language and semantics in that each word maps to a word and each sentence maps to a sentence. Given a ciphertext string, the model attempts to minimise the negative log likelihood loss between the predicted plaintext and original plaintext.

One way to mitigate this is by using a multi-Loss approach. We add two loss functions to the natural language loss function: L1 (signifying the injectivity) and L2 (representing the surjectivity):

where C is the cross entropy loss. S is the input sentence and S_i is each character. Sî is the set of all the characters in S that map to i.

$$\mathcal{L}1 = \sum_{i=1}^{N} \sum_{j=i} C(S^i, S^j) \mathcal{L}2 = 1/\sum_{i=1}^{N} \sum_{j \neq i} C(S^i, S^j) \quad (1)$$

The code implementation being:

Whilst each loss converges separately, and L1 with L2 converge together, combining the character based constraint and the natural language constraint results in unmeaningful convergence. This shows deep learning's current difficulties to model really discrete relationships. Slight change in the generated sequence results in large difference in losses, so a gradient descent does not find an obvious continuous path.

The main problem is that this model of transformers assumes that the translation function is fixed and works well for problems such as English to French translation. Adding the constraint really mires the models understanding.

3.4. Sequence to classifier to sequence

In an effort to provide a more consistent model we can give the model the underlying understanding of how the ciphers are represented. For substitution, the underlying understanding is that each character maps to a character.

Therefore we build a model with a classifier as the output representing this map. The output should be the one hot encoding of the map. This approach still requires us to write a loss to maintain injectiveness but in this representation, the loss has much clearer meaning and much better results. Also, by using an encoder RNN to embed the sequence we have a 2d space we can do classification on.

To maintain injectiveness, the model finds the maximal matching of the outputted probabilities. Two types of loss are considered which represent applying the loss before the maximal matching and apply the loss after the maximal matching. The performance has a big difference but so does the training cost. This is shown in the experimental procedures.

To remove the noise from the output one could also consider training a decoder to map the separately trained encoder back to natural language. This approach is motivate by the idea that transformers have no backbone understanding so we give it one to ensure that the output is meaningful. there for the transformer only acts a translator, which these models excel at, to interface between natural language and a graph representation.

The main limitation of this approach is that the maximal matching loss is very expensive to train. To get around this, the model does a maximal matching loss every 10 epochs to try to maintain the models compatibility with the injective constraint.For the other epochs, the loss is calculated by the cross entropy of the output from the model before matching and the one hot encoding of the key.

3.5. Generative

Another approach is to treat the problem as a lossy transform. Then similar to a re-colourisation of a grey scale image we can reverse the transform and fill missing data that makes lexical and semantic sense. A GAN is a network typically used for this.

GAN also is a well motivated investigation for cryptography as cryptography is an adversarial problem. One can use pretrained discriminators to decide what is natural language or not with high accuracy.

However, this approach did not give good results. The main difficulty is that it is extremely obvious to the discriminator what is real or generated. Generators and discriminators should be trained at the same rate, however the generator fails to develop with accurate discriminators.

On the other hand, training a discriminator for natural language within a GAN framework is not feasible within the scope of this project.

3.6. Computer vision

By showcasing an image input to the model we can show the extendability and utily of the paper.

Computer vision here is an interesting problem as the symbols may not be anything seen before. The only consideration we can make is the similarity between characters. One can consider the image of a handwritten sentence and think of the potential problems to overcome. ”a” and ”a” may be the same symbolically but pixle-wise different. For classical OCR the symbols are known and the variation between the two characters can be learnt. Therefore, classical OCR disregards some context that can help provide better results.

The formal problem statement is to take an image of a sequence of symbols and return a sequence of numbers $a_1, a_2, ..., a_n$ such that if and only if $a_i = a_j$, it implies the symbols have the same meaning.

Following the same inputs to the other models, we can. The benefit of using embeddings is that there are many unsupervised methods which, also solves the issue of classifying unseen characters.

Even plaintext would be read in as scrambled data by the vision, e.g. ”hello” $- >$ [1,2,3,3,4]. However, as discussed earlier, this does not limit the power of the tool as the tool should still be able to solve a scrambled ”hello”.

This project showcases the usability of the tool by extending it to accept

4. Experiments

The models are trained with varying common hyperparameters. Notable results are shown here, whilst the full results can be found in the supplementary information.

Hyperparameters:

Noise: Applied to the dataset as p1 and p2 (as defined above).

Dataset: What dataset is used.

Size: Size of subset of the data trained on. size = 1 is used to validate the model and loss function.

Hidden size: Sizes of hidden layers within the network

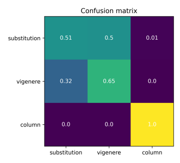
4.1. classification

With a size of 30000 entries. 24000 for train, 6000 for test and 10000 per class: The model performs very well for column transposition and quite well on vigenere and substitution. Notably, substitution and vigenere have large overlaping false positives due to their similarity. The effect of noise on the Book corpus is shown in figure 2. Spaces are not encrypted.

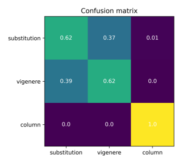
4.2. Sequence to sequence

Qualitative results of first not adding any bijective loss (Figure 3), then with the bijective constraints (Figure 4).

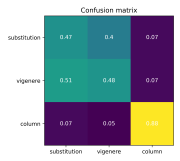
This shows the inability for the model to both do natural language and the constraint. Differently weighting the multi-loss was investigated but had no effect.



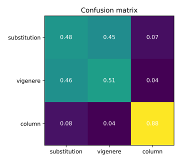
(a) 0, 0



(b) 0, 0.3

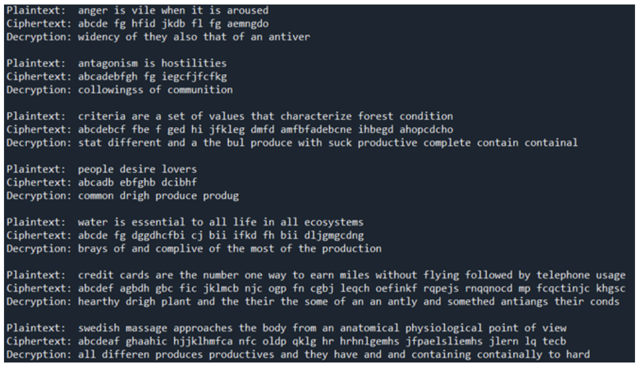


(c) 0.3, 0



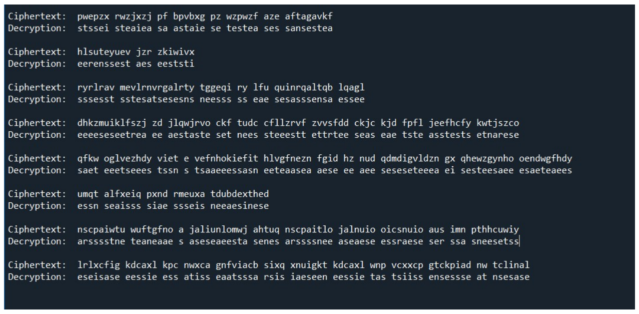
(d) 0.3, 0.3

Figure 2. Classifying bookCorpus with spaces kept as plaintext and p1 = [0, 0.3], p2 = [0,0.3]



```
Plaintext:  anger is vile when it is aroused
Ciphertext: abcde fg hfid jkdb fl fg aemngdo
Decryption: widency of they also that of an antiver

Plaintext:  antagonism is hostilities
Ciphertext: abcadebfgh fg iegcfjfcfkg
Decryption: collowingss of communition

Plaintext:  criteria are a set of values that characterize forest condition
Ciphertext: abcdebcf fbe f ged hi jfkleg dmfd amfbfadebcne ihbegd ahopcdcho
Decryption: stat different and a the bul produce with suck productive complete contain containal

Plaintext:  people desire lovers
Ciphertext: abcadb ebfghb dcibhf
Decryption: common drigh produce produg

Plaintext:  water is essential to all life in all ecosystems
Ciphertext: abcde fg dggdhcfbi cj bii ifkd fh bii dljgmgcdng
Decryption: brays of and complive of the most of the production

Plaintext:  credit cards are the number one way to earn miles without flying followed by telephone usage
Ciphertext: abcdef agbdh gbc fic jklmcb njc ogp fn cgbj leqch oefinkf rqpejs rnqqnocd mp fcqctinjc khgsc
Decryption: hearthy drigh plant and the their the some of an an antly and somethed antiangs their conds

Plaintext:  swedish massage approaches the body from an anatomical physiological point of view
Ciphertext: abcdeaf ghaahic hjjklhmfca nfc oldp qklg hr hrhnlgemhs jfpaelsliemhs jlern lq tecb
Decryption: all differen produces productives and they have and and containing containally to hard
```

Figure 3. NNLLoss



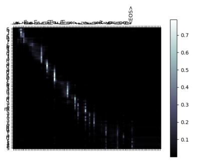
```
Ciphertext:  pwepzx rwzjxzj pf bpvbxg pz wzpwzf aze aftagavkf
Decryption:  stssei steaiea sa astaie se testea ses sansestea

Ciphertext:  hlsuteyuev jzr zkiwivx
Decryption:  eerenssest aes eeststi

Ciphertext:  ryrlrav mevlrnvrgalrty tggeqi ry lfu quinrqaltqb lqagl
Decryption:  sssesst sstesatsesesns neesss ss eae sesasssensa essee

Ciphertext:  dhkzmuiklfszj zd jlqwjrvo ckf tudc cfllzrvf zvvsfdd ckjc kjd fpfl jeefhcfy kwtjszco
Decryption:  eeeeseseetrea ee aestaste set nees steeestt ettrtee seas eae tste asstests etnarese

Ciphertext:  qfkw oglvezhdy viet e vefnhokiefit hlvgfnezn fgid hz nud qdmdigvldzn gx qhewzgynho oendwgfhdy
Decryption:  saet eeetseees tssn s tsaaeeessasn eeteaasea aese ee aee seseseteeea ei sesteesaee esaeteaees

Ciphertext:  umqt alfxeiq pxnd rmeuxa tdubdexthed
Decryption:  essn seaisss siae ssseis neeaesinese

Ciphertext:  nscpaiwtu wuftgfno a jaliunlomwj ahtuq nscpaitlo jalnuio oicsnuio aus imn pthhcuwiy
Decryption:  arsssstne teaneaae s aseseaeesta senes arssssnee aseaese essraese ser ssa sneesetss

Ciphertext:  lrlxcfig kdcaxl kpc nwxca gnfviacb sixq xnuigkt kdcaxl wnp vcxxcp gtckpiad nw tclinal
Decryption:  eseisase eessie ess atiss eaatsssa rsis iaeseen eessie tas tsiiss ensessse at nsesase
```

Figure 4. NNLLoss/2 + L1/4 + L2/4

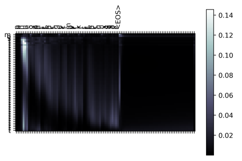
We can visualise the attention mechanism which shows, as we expect, that NNLLoss (Figure 5a) has narrow semantic attention, Whilst the multi-loss has much longer range attention (Figure 5b).

4.3. Sequence to classifier to sequence

The model performs well for solving in this method. As can be seen from 6, the model understands the constraint and attempts to find the natural language. As discussed, the mapping from the embedding to the plaintext prioritises the



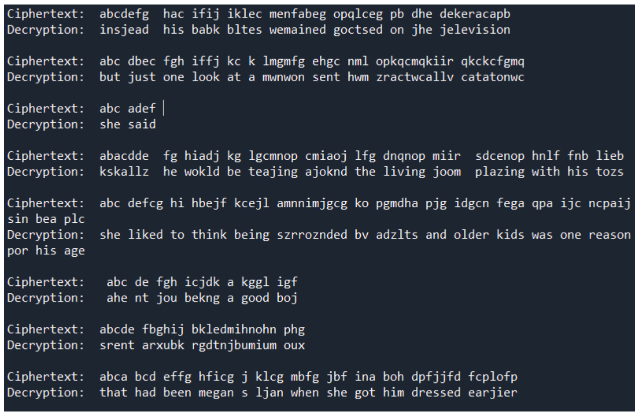
(a) NLLLoss



(b) Multi-loss

Figure 5. Attention between input(x) and output(y)

constraint so some obvious natural language faults appear. The reader can make sense of the result with a little effort. The model was trained for 75 epochs on 10000 entries of the bookcorpus. Note the periodic loss (7) resulting from switching the loss function (every 10 epochs).



```
Ciphertext:  abcdefg  hac ifij iklec menfabeg opqlceg pb dhe dekeracapb
Decryption:  insjead  his babk bltes wemained goctsed on jhe jelevision

Ciphertext:  abc dbec fgh iffj kc k lmgmfg ehgc nml opkqcmqkiir qkckcfgmq
Decryption:  but just one look at a mwnwon sent hwm zractwcallv catatonwc

Ciphertext:  abc adef |
Decryption:  she said

Ciphertext:  abacdde  fg hiadj kg lgcmnop cmiaoj lfg dnqnop miir  sdcenop hnlf fnb lieb
Decryption:  kskallz  he wokld be teajing ajoknd the living joom  plazing with his tozs

Ciphertext:  abc defcg hi hbejf kcejl amnnimjgcg ko pgmdha pjg idgcn fega qpa ijc ncpaij
sin bea plc
Decryption:  she liked to think being szrroznded bv adzlts and older kids was one reason
por his age

Ciphertext:   abc de fgh icjdk a kggl igf
Decryption:   ahe nt jou bekng a good boj

Ciphertext:  abcde fbghij bkledmihnohn phg
Decryption:  srent arxubk rgdtnjbumium oux

Ciphertext:  abca bcd effg hficg j klcg mbfg jbf ina boh dpfjjfd fcplofp
Decryption:  that had been megan s ljan when she got him dressed earjier
```

Figure 6. Encoder to classifier to matching

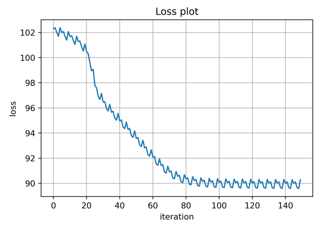


Figure 7. Loss for encoder to classifier model

4.4. Computer vision

Taking the famous cipher text of the zodiac we can assess the end to end ability of the project. The input image is the first page of one of the killers messages. `https://en.wikipedia.org/w/index.php?title=File:Zodiac_Killer_cipher_deciphered_by_Donald_and_Bettye_Harden.pdf`

We input the similarities of each character as a one-hot encoding setting the first character to a then b and so on. The cipher doesn't use any spaces so we use the model best suited to this scenario. The result from the model is:

For the classifier:
substitution - 0.65
vigenere - 0.34
column - 0.01
From the solver:
—-
flikillikpeoplegecaseitivomuch
funitismoiefunthakllinwiudgame
intheforrestbecausemcnisthehos
tdancerouanimalofllltokullsemeh
inggm
—-

It provides a similar solution as on the pdf and shows the capability of the system.

5. Conclusion

The success of the project is mostly good and it maintains this success on small levels of noise. The robustness on naturally noisy language was one of the main motivations for utilising deep learning. Whilst training is quite disruptively slow, using the model is much faster than brute force.

The final tool produced is a image to plaintext program with an analysis of different approaches. The robustness within varying situations is also measured. A large focus of the the project has been to make the code base easy for continued research. The repository has test files, utils and easy ways to add more ciphers and datasets. Large amounts of test data can be generated easily, as is seen in the supplemental information.

The main success is the ability to interpret text input. Further work is needed to extend to be able to reliably take an image input. The main issue to producing research is only good databases are closed source due to historic copyright. Non immediately accessible but research permissible databases could be used, such as DECODE. [8]

Whilst there has been attempts to create more modular injective deep learning architectures, this approach performs well the task required. [9] This project showcases an implementation and investigation of this open problem, which provokes questions and thoughts on why this problem is hard for deep learning.

The main issues is the time to train the classification layer. Finding the maximal matching takes

$$O(n^3) = O(27^3)$$

per prediction. Further investigation could be made to quickly approximate this loss whilst maintaining enough accuracy to not abuse the SGD and reduce the model capacity.

6. Supplementary Material

The project code can be found on GitHub: https://github.com/Jacobdavis12/Deepcrypter

Supplemental figures can be found on the GitHub: https://github.com/Jacobdavis12/Deepcrypter/plots

References

[1] A. Sahai and B. Waters, "How to use indistinguishability obfuscation: Deniable encryption, and more," *SIAM Journal on Computing*, vol. 50, no. 3, pp. 857–908, 2021. 1

[2] N. R. Krishna, *Classifying Classic Ciphers using Machine Learning*. PhD thesis, San Jose State University Library. 1, 3

[3] "Table of features for ciphertexts." https://bionsgadgets.appspot.com/gadget_forms/acarefstats.html. Accessed: 2024-05-7. 2

[4] C. Ivan, "Convolutional neural networks on randomized data," *CoRR*, vol. abs/1907.10935, 2019. 2

[5] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015. 2

[6] "Genericskb: A knowledge base of generic statements," Allen Institute for AI, 2020. 2

[7] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014. 3

[8] M. Héder and B. Megyesi, "The decode database of historical ciphers and keys: Version 2," pp. 111–114, 06 2022. 6

[9] M. Puthawala, K. Kothari, M. Lassas, I. Dokmanic, and M. V. de Hoop, "Globally injective relu networks," *CoRR*, vol. abs/2006.08464, 2020. 6