Generative Adversarial Networks for Classic Cryptanalysis

Deanne Charan

Generative Adversarial Networks for Classic Cryptanalysis

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Deanne Charan

August 2021

The Designated Project Committee Approves the Project Titled

Generative Adversarial Networks for Classic Cryptanalysis

by

Deanne Charan

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

August 2021

ABSTRACT

Generative Adversarial Networks for Classic Cryptanalysis

by Deanne Charan

The necessity of protecting critical information has been understood for millennia. Although classic ciphers have inherent weaknesses in comparison to modern ciphers, many classic ciphers are extremely challenging to break in practice. Machine learning techniques, such as hidden Markov models (HMM), have recently been applied with success to various classic cryptanalysis problems. In this research, we consider the effectiveness of the deep learning technique CipherGAN---which is based on the well-established generative adversarial network (GAN) architecture---for classic cipher cryptanalysis. We experiment extensively with CipherGAN on a number of classic ciphers, and we compare our results to those obtained using HMMs.

ACKNOWLEDGMENTS

I would like to express my gratitude to my project advisor, Dr. Mark Stamp for his encouragement, guidance and support throughout my graduate studies. He provided the inspiration for this project by introducing the topics covered in this research through his coursework.

A very special thanks to my committee member, to Dr. Fabio Di Troia for helping me finalize my research topic.

I would also like to thank my committee member Dr. Mike Wu for his time and guidance.

Finally, I would also like to thank my friends and family for their support throughout the duration of my Master's at San Jose State University.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Converting information into a ‘‘secret code’’ that is capable of hiding the true meaning is defined as encryption, while decryption is the inverse operation. Cryptography is the art and science of creating encryption and decryption systems. On the other hand, the art of cracking encrypted messages is cryptanalysis [1].

Throughout time, people have realized the importance of keeping critical information secret but accessible to a select few. Many classic ciphers have played a critical role in history. For example, the successful cryptanalysis of the German Enigma and Japanese Purple ciphers [2] shortened the World War II and saved thousands of lives.

Although classic ciphers have been supplanted by more secure modern ciphers, many classic ciphers can be difficult to break in practice. For example, the Zodiac 340 (Z340) cipher was created in the late 1960s by the infamous Zodiac serial killer. In spite of intense cryptanalysis, the Z340 remained unsolved for more than half a century [3].

Modern machine learning techniques have been applied to classic ciphers. For example, Stamp et al. [4] use Hidden Markov models (HMM) to attack classic substitution ciphers. Generative adversarial networks (GAN) have also been proposed for classic cryptanalysis problems [5]. In this research, we carefully compare the capabilities of Hidden Markov Models and suitably modified Generative Adversial Networks as techniques for attacking classic substitution ciphers.

The remainder of this paper is organized as follows. In Chapter 2, we introduce topics relevant to text generation using deep learning as well as some background in cryptography in order to better understand our approach. We then dwell on work that inspired this paper as well as previous work related to this research in Chapter 3. In Chapter 4 the experimental setups in which the models of interest are trained is explained. The results are discussed alongside comparisons with previous work.

Finally, in Chapter 5 we summarize our findings regarding the effectiveness and efficiency of the models studied.

CHAPTER 2

Background

This chapter introduces relevant concepts for a concise understanding of the topics used in this research. We start with Cryptography essentials as a refresher, followed by some theoretical background on the models used. The explanation is limited to the most crucial aspects of the working of the models and the necessary backdrop for each.

2.1 Cryptography

Cryptography is to use a system to encrypt data. That is, the data or a message, is to be kept hidden by the means of confusion and diffusion as per Claude Shannon's Communication Theory of Secrecy Systems [6]. The letters of the original message need to be jumbled up amongst each other, this is diffusion. The letters being substituted for each other with certain rules defined within the cryptosystem, is confusion.

The original message called the plaintext is encrypted via a cipher or cryptosystem, and the resulting encryption is called the ciphertext. If we can convert the ciphertext to the plaintext, this process is called decryption. In classic cryptography, or symmetric cryptography, the processes of encryption and decryption are done by using a key [1]. Figure 1 is a visual depiction of a cryptosystem.

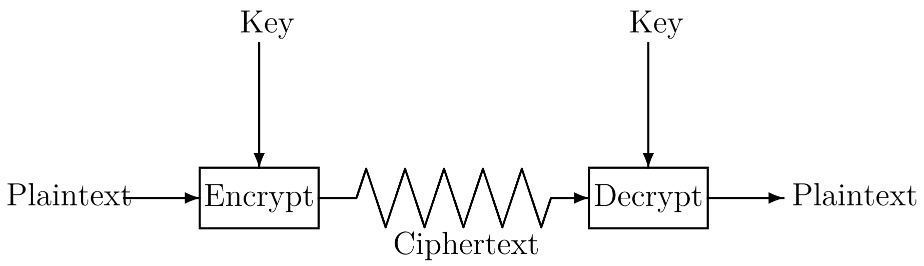

Figure 1: Crypto as a black box

If we view cryptology as some kind of black box [7], it is easy to understand the choice of employing neural networks. Given that in neural networks we feed large amounts of data to the network, guide it towards a certain learning but it devises its own system of understanding the data.

Cryptanalysis is the art of breaking secret codes. In the case of symmetric cryptography this is done by figuring out the key, the plaintext or both. Thus often, a cryptanalyst's goal is to find this key, while machine learning's target is to find a suitable solution in a large space of possible solutions; that is searching in large spaces [8] is common to both cryptanalysis and machine learning.

There are various types of attacks possible in cryptanalysis, depending on the information available to the attacker. The most basic, of course, being the ciphertext alone by itself. If this was not available to the attacker, there would be no need for encryption in the first place. These sorts of attacks are known as ciphertext-only attacks.

2.2 Classic Crypto Basics

Classical encryption involves the usage of a secret key. This means that the parties that need to access the message must know the key in order to do so. The actual channel of transmission though, is unsafe. Therefore, the message is encrypted and near impossible to decrypt without a key. This section introduces several classical substitution ciphers alongside traditional cryptanalytic methods to attack them.

2.2.1 Simple Substitution Cipher

A good example of a substitution cipher is the Caesar cipher. The key size is 26, i.e. the size of the original alphabet of the plaintext. Each symbol corresponds to a new symbol in the cipher text. There is a one-to-one correspondence. In the case of Caesar cipher, each letter is shifted cyclically to correspond to another letter in the cipher text. The key in the following system in Figure 1 is 3 and English letters in the plaintext are mapped to English letters in the ciphertext. The following key is used in the encryption and decryption process.

Table 1: The key of a Caesar cipher with shift 3

Plaintext	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c

Caesar's cipher is usually broken using letter frequency analysis. Letter frequencies, that is how often a letter shows up in a language can be used to match cipher symbols and plaintext symbols. This is why confusion is simply not enough. Diffusion too must be employed as in more advanced ciphers explained in the following Section 2.2.2.

2.2.2 Homophonic Substitution Cipher

In Homophonic Substitution, a plaintext symbol can map to more than one ciphertext symbol [9]. This diffuses the letter frequency statistics across the ciphertext which can be seen in Figure 2. For example, the most common alphabet in English, 'E', can map to 3 ciphertext symbols and have its occurrences distributed over these symbols to throw off frequency analysis.

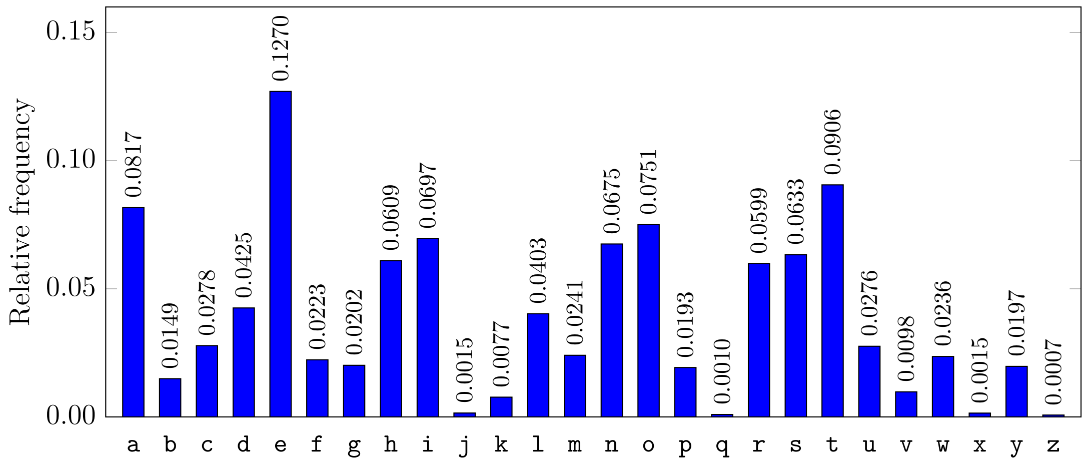

Figure 2: English letter relative frequencies

2.2.2.1 Vigenère cipher

The Vigenère cipher named after a French mathematician, is an example of a polyalphabetic cipher. An example of how it works can be explained starting with the aid of the table in Figure 2

Table 2: The Vigenère key

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Considering an encryption with the key as "CAT" and the secret message

ATTACKNOW.

The first letter of the plaintext is A and the first letter of the key is C, so one must look at row A and column C, which gives the letter C itself. The next letter would be in row T until the column A which once again would be T itself. But for the third letter, one would look at the row T until the column T which would give us the letter M. The rest of the message would be translated as in the following Table 3.

We can see the same letter in the ciphertext such as "P", translates back to more

Table 3: Vigenère enciphering

Key	C	A	T	C	A	T	C	A	T
Plaintext	A	T	T	A	C	K	N	O	W
Ciphertext	C	T	M	C	C	D	P	O	P

than one plaintext symbol, hence this is super effective against frequency analysis type attacks.

2.3 Deep Learning

Neural Networks with a lot of hidden layers and a large number of parameters have been on a steady rise. Especially among larger corporations which can afford to invest in the resource intensive technology. Neural Networks are a series of layers, each of which are comprised of nodes. These nodes perform a series of computations on the input sequence, that converts the input into a desired output. The training phase is what controls and enables this "desirable" result. Depending on the task at hand, if it's classification of the data, the output would be whichever corresponding class, represented by a number that the input sequence belongs to. This is done by "telling" the network, the correct class for each of the training data sequences. Thus, this is called Supervised training [10], where the desired output is labelled for the network via the training data.

Unsupervised training involves feeding in a large amount of data to the Neural Network and it must learn to recognize patterns in this data [11]. Note that in both Unsupervised and Supervised learning, a technique called backpropagation [12] is used to train the model. That is, the parameters under training that are used for the computations in the nodes are updated based on the gradient of the derivatives using the chain rule. The gradient itself is like a feedback signal (that is directed to flow backwards through the nodes, hence "backwards propagation") to denote how close the output of this particular training round was to the actual desirable output.

2.3.1 GPT-2

GPT-2 [13] is a large transformer-based language model with 1.5 billion parameters, trained on a dataset of 8 million web pages. GPT-2 is trained with a simple objective: predict the next word, given all of the previous words within some text. GPT-2 generates synthetic text samples in response to the model being primed with an arbitrary input. The model is chameleon-like—it and adapts to the style and content of the conditioning texts. GPT-2 achieves state-of-the-art scores on a variety of domain specific language modeling tasks.

The model is not trained on any of the data specific to any of these tasks and is only evaluated on them as a final test; this is known as the “zero-shot” setting. GPT-2 outperforms models trained on domain-specific datasets (e.g. Wikipedia, news, books) when evaluated on those same datasets. This makes it perfect for imitating the manner of speaking of a particular style of writing.

2.3.2 Transformers

Transformers of Vaswani et al. [14] have become supremely popular and successful with sequence modelling applications such as text generation. The main operation of a transformer is the self-attention mechanism. The purpose of which is to determine how important all other words in a sentence are w.r.t. a particular word. Since it is a kind of weighted average this makes in highly parallelizable. In the case of text generation, each word is converted into an embedding vector. If we feed this embedding layer into the self-attention layer, the output is another sequence of vectors.

A key reason for using the self-attention mechanism is that it sees the input as a set, not a sequence. This makes sense at an intuitive level, as the ciphertext will belong to one set of mappings withing the cryptosystem. Though it must be noted that self-attention ignores the sequential nature of the input since it is permutation equivariant. This is compensated either by having position embeddings or encodings.

2.3.3 GAN

Generative Adversial Networks which were introduced by Goodfellow et al. [15] consists of a generator and a discriminator. The generator usually produces an image belonging to a target domain, while the discriminator that's been trained on the actual data of that domain, tries to correctly guess whether or not an image is sampled from the domain or generated by the Generator. Traditionally GANs are used for image synthesis and not usually applied to a discrete domain like language, this is addressed in a later section.

2.3.4 CycleGAN

CycleGANs [16] are capable of learning unsupervised translation between two groups of image data. CycleGANs take two pairs of GANs and train them simultaneously. The purpose is to maintain what is known as cycle consistency. A new cycle loss term is added along with the usual loss terms associated with the GAN pair. It is to be noted that the optimization is to be done with both of the pairs of GANs being trained and the cycle loss term. The purpose of the cycle loss can be explained with the help of an example.

Suppose the CycleGAN needs to learn to convert images of nature from summer (X) to winter (Y). The GANs will be broken up as follows. The first generator F learns to generate a winter image of a summer input image, i.e., from X to Y. The discriminator D_Y distinguishes between real images of Y and $\tilde{Y}$. The second GAN pair is trained to convert and distinguish in the opposite direction i.e. from Y to X and between X and $\tilde{X}$ correspondingly. The most important point to note is that the second GAN could aim to convert the images $\tilde{Y}$ to $\tilde{X}$ while the GANs are being trained. Thus, for a cycle of converting one image to winter and then back to summer the images must be consistent, which is known as cycle consistency [16].

2.3.5 CipherGAN

CipherGANs are a neural network architecture based on CycleGANs that are capable of unsupervised machine translation between two "languages" [5]. In this case it is capable of learning to translate between plaintext and ciphertext banks of data of a particular encryption key for certain crypto-systems.

CipherGAN addresses the problem of uninformative discrimination that usually shows up when the data is discrete. Compare this scenario, of a continuous distribution over a discrete distribution, to the data being represented as a standard simplex of dimension k. If a produced sample lies somewhere within the simplex as opposed to a vertex (where the vertex is a possible desired discrete point), the discriminator must be able to distinguish between a point closer to the desired vertex. This problem was addressed in Gomez et al. [5] by using an appropriate regularization term coupled with continuous relaxations of the discrete random variables. The latter was done by having the discriminator operate over the embedding space rather than directly on the Softmax vectors [17]. This was called the relaxed sampling technique and it was capable of dividing the space within the simplex (considering the data is discrete) as illustrated in the Figure 3, taken directly from the CipherGAN paper [5] with the right-most being the most informative. The shaded colors represent the ability to discriminate between a point closer to the desired simplex (the bottom right vertex in this case) and one further away from it, such as in the darker regions.

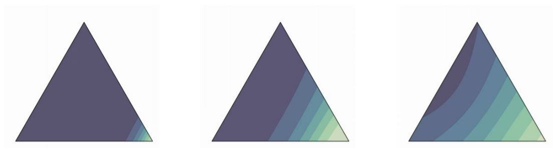

Figure 3: From left to right the discriminators regularized using: nothing; WGAN Jacobian norm regularization; and, relaxed sampling.

Additionally, the discriminator was made to operate in the embedding space instead of over the end vectors from a SoftMax output. This was proven to work and justified with the understanding that the embedding vectors act as continuous relaxations of the discrete variables, these continuous relaxations allow for better feedback during the training phase in the form of a stronger, more informative gradient signal. Thus, addressing the problem called uninformative discrimination.

2.4 HMM

Hidden Markov Models(HMM) are a part of statistical Markov models.The system being modeled is assumed to be a Markov process which can be explained with the help of an example taken from Stamp et al. [18]. Consider a process Y that is observable (such as rings in a tree), but depends on another hidden process (Average annual temperature of that year corresponding to each ring), considered the Markov process (in this case we cannot go back in time therefore it is hidden). That is, the states themselves are not observable, and we can only determine probabillistically which state we are in at a given time, hence the Hidden in HMMs [19]. Suppose a hot year corresponds to a large ring width and the opposite is true for a colder year. Given the size of the ring, a probability can be derived using HMMs for the temperature that year.

Figure 4 is a illustration of the flow of the HMM discussed in the example.

The HMM model can be specified as $\lambda = (A, B, \pi)$ [20].The matrix A represents the probability of transition from one state to another. While the matrix B is the probability the process was in a particular state given the observation. The matrix π is simply the probability of the initializing in either of the states. All three of these matrices are row stochastic, that is, each row satisfies the conditions of a discrete probability distribution.

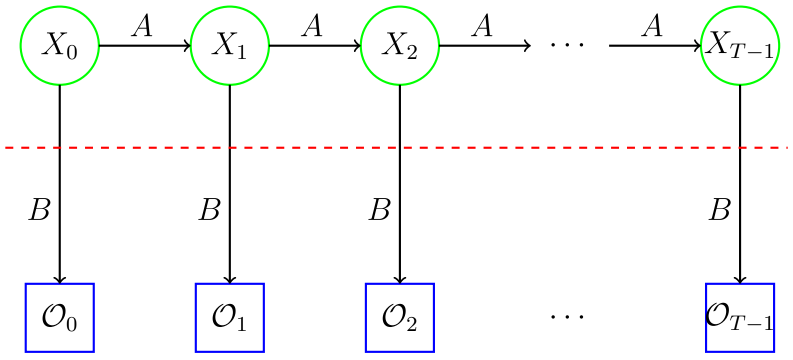

Figure 4: Hidden Markov model

Note that the dimensions of (A, B, π) correspond to $N \times N$, $N \times M$ and $1 \times N$ respectively. Here, N is the number of hidden states(hot or cold year) and M is the number of distinct observation symbols to be determined during training. Training the HMM is a kind of hill climbing algorithm that optimizes these matrices to converge to the right solution. Given the nature of the hill climb problem the initializations or starting values of these matrices before training determines "closest" optima. Thus, random restarts [21] with different initializations are performed and the best HMM is obtained.

2.5 The Zodiac Cipher

The Zodiac was a serial killer in the late 60s in California. He sent encrypted letters describing his crime. He has not been caught yet and one of his letters the Z340 had long haunted cipher experts across the globe. A recent claim had stated that only the first 8 lines were to be considered. And the entire message is a reverse homophonic substitution cipher. However this leaves room for a lot of possible alternate solutions that make perfect sense in English that all fit the reverse homophonic substitution conditions [3]. Because a single cipher symbol can map to more that one overlapping plaintext symbol, there can be a very large set of possible solutions, of which many are likely to make coherent sense in English along with the 1 of these many possible solutions that Bauer which can be seen in the Table 4.

Table 4: Bauer's Z340 putative decryption of first 8 lines

Row																	
1	0 H	1 E	2 R	3 E	4 I	5 T	6 I	7 S	8 I	9 K	10 I	11 L	12 L	13 B	14 O	15 T	16 H
2	17 N	4* I	18 G	19 H	20 T	21 A	22 N	23 D	24 D	25 A	26 Y	27 .	28 I	29 L	30 I	31 V	32 E
3	$\boxed{19}$ B	33 Y	34 T	35 H	36 E	18* G	37 U	38 N	$\boxed{14}$ B	25* A	$\boxed{20}$ R(R)	32* E	12* L	21* A	39 I	$\boxed{0}$ M	40 ␣
4	41 S	$\boxed{4}$ O	$\boxed{4}$ Q	42 U	6* I	5* T	43 W	$\boxed{29}$ I	7* S	44 H	4* I	22* N	18* G	$\boxed{18}$ F	$\boxed{2}$ O	$\boxed{30}$ R	$\boxed{15}$ G
5	45 A	46 M	36* E	$\boxed{18}$ T	$\boxed{39}$ O	47 B	48 E	$\boxed{16}$ O	$\boxed{10}$ V	49 E	50 R(P)	8* I	18* G	51 I	52 S	$\boxed{9}$ ␣	53 M
6	4* $\boxed{I}$	43* W	2* R	6* I	$\boxed{50}$ S	5* T	22* $\boxed{N}$	54 $\boxed{I}$	29* L	16* O	55 C	9* K	50* S	$\boxed{3}$?	$\boxed{15}$ N	$\boxed{24}$ O	$\boxed{20}$ W
7	21* A	$\boxed{49}$ N	18* G	30* R	56 Y	23* D	57 A	15* N	$\boxed{37}$ G	$\boxed{35}$ E	58 R	14* O(U)	7* S	27* .	$\boxed{39}$ ␣	$\boxed{12}$ ␣	10* I
8	20* W	14* O	15* N	$\boxed{40}$ T	$\boxed{31}$ C	$\boxed{48}$ H	21* A	22* N	18* G(E)	45* A	17* N	26* Y	39* O	18* F	59 G	$\boxed{12}$ A	46* M(E)

If a solution can be found, that minimizes the number of cipher symbols used as reverse homophones to say, 0: that would be a regular homophonic substitution cipher. However, this solution was recently proved false by a more plausible solution to the Z340 which was a complex polyalphabetic substitution cipher along with a some kind of diagonal transposition after breaking up the ciphertext into chunks.

CHAPTER 3

Related Work

3.1 GPT-2

Ciphertext to plaintext can be seen as a kind of unsupervised language translation as mentioned even in the CipherGAN paper, thus a generative model GPT-2 [13] was considered for plaintext generation. The model which has been trained on a large corpus of text data available on the internet is capable of producing coherent sounding English. The transformer based architecture has good positional understanding and a very large parameter set. Therefore, while training it for a specialised task, one must only fine tune it on a smaller data set in order to obtain reasonably compelling results.

Thus, the model was trained to imitate the style of writing of the Zodiac and even produced several "secret messages" based on different seed words. One could conclude that the messages make sense to an English reader, albeit with a few grammatical mistakes, that could be taken as purposeful attempts to mislead attackers. Secondly, what came about was a very characteristic and convincing style of writing, which meant that the text produced was not too general to be a possible kind of writing that is like secretive messages.

Broadly, the two main criteria of this experiment, are applicable to even general classic cryptanalysis in the following way. The first criteria, was the "Englishness" of these messages generated, that is they should convey a meaning. The second criteria were to minimize the number of inconsistencies of plaintext to ciphertext translation with respect to the cipher key resulting from the generated plaintext in order to achieve an optima.

3.2 CipherGANs

After having experimented with text generation using neural networks another type of generation was considered. As explained in the background section, the

CipherGAN model is trained for a particular key. Ciphertext samples are generated with a chosen key for a particular enciphering method. These, along with plaintext samples are fed into the GAN pairs of the CycleGAN as the corresponding X and Y distributions. The CipherGAN's accuracy is evaluated on a part of the corpus excluded from the training source [14] via which the ground-truth accuracy for each of the two distributions is obtained.

The vocabulary of the CipherGAN is the number of symbols it can understand or translate. Since it converts all symbols into integers and then finally has an embedding vector scheme, it is capable of learning a very large set of both plaintext and ciphertext symbols. It must be noted that the embedding vectors are trained as parameters of the network.

3.3 HMMs

The ability of an HMM to attack the Vigenère cipher was studied in Stamp et al. [4]. The finding that the matrix A converges when the cipher key length is equal to the number of hidden states N, is intuitive to understand when one considers the Vigenère encryption process as elaborated in the Table 3 in Chapter 2. It is almost as if each of the letters of the key, when repeated across the length of the plaintext, behave as the distinct hidden states with unique rules of converting each of the letters that are fed through these hidden states.

The HMMs were trained by feeding in the ciphertext data into them. The Brown corpus [22] was used after "cleaning" the textual data off punctuation, etc. to reduce it to 26 (or 27) symbols. The B matrix was observed to converge with $M = 26$ observation symbols.

Getting the matrices to converge required a decent amount of training with random restarts [21]. Stamp et al. [4] used 100 random restarts. The random restarts which shuffle about the initial values of the matrices, increase the chances of finding the global optima as opposed to the local optima after training at the point of convergence. Another key factor for convergence is the amount of training data, which in this case was the ciphertext itself.

Although, an unlimited amount of ciphertext would be ideal for training purposes, in most real case scenarios ciphertext is limited. Therefore, finding a lower-bound on the amount of ciphertext required for convergence was studied. They experimented on keywords of different sizes followed by experiments on the minimum amount of ciphertext needed for each of these keyword sizes.From their findings, Stamp et al. [4] concluded that HMMs were more informative than the CipherGAN with respect to a understanding the underlying encryption process.

CHAPTER 4

Experiments

4.1 Data

The Brown corpus English text dataset [22], is used for both the HMM and the CipherGANs. For the CipherGAN, when a model is being trained for one particular key, from the corpus is selected two batches one of which is passed through encryption for ciphertext generation and are fed to the CipherGAN as the X and Y distributions. For the natural language plaintext data, taken from the brown corpus which consists of over one million words in more than 50,000 sentences, the top k most frequent words are included as part of the vocabulary and the rest are represented as unknown tokens. The average sentence length is 20 tokens, while the max and min are 180 and 1 respectively. The capability of the CipherGAN to crack codes not only at character level but at word level too, must be noted over here.

4.2 Overview

The experiments consist of applying CipherGANs to classic encryption systems in order to study their capabilities exhaustively. A brief overview of the following experimental setups is as follows. The CipherGAN can be trained to attack a cipher at either a character level or word level as explained, thus that is one of the hyper-parameters. The other main hyper-parameter is the vocabulary size, or the number of symbols used in the classic encryption system. The ground-truth accuracy of the CipherGAN is noted at each training step of the CipherGAN, and it is the accuracy that is the metric of focus in these experiments as that is what would be useful in applying the CipherGAN to crack a cipher system. The next hyper-parameter is the encryption system itself, along with its unique key. For example, in a Vigenère encryption system, there are a choice of multiple keys of multiple sizes and for each of these keys must a new CipherGAN be trained. The Table 5 summarizes the

Table 5: Hyper-parameter combinations experimented with

	Vocabulary size	
Cipher type/ Vocab type	chars	words
Vigenère Cipher (key range 3 through 7)	26 58 100	200 500 1000 2000 5000
Shift Cipher/ Ceasar's Cipher	26 100	200 2000 5000

hyper-parameters; the combinations of which have been thoroughly experimented on in this report. To explain for example, one can see a total of 8 types of experiments for the Vigenère cipher alone, this whole set of experiments were performed repeatedly for each of the unique Vigenère keys of varying sizes.

4.3 Setup

The architecture is adapted from the Gomez et al. [14]. It is run on 2.20 GHz processor with a 16Gb total memory configuration over a virtual machine on Google Colab on a Linux system. The Generators are fed a sequence of vectors in embedding space. The embedding vectors are jointly trained as parameters of the model with 256 dimensions. The output of the Generators is a Softmax distribution over the vocabulary while that of the Discriminators are a scalar output. The loss for each GAN pair is $\mathcal{L}_{GAN}(F, D_Y, X, Y) = \mathbb{E}_{y \backsim Y}[(D_Y(y \cdot W_{Emb}^{\top}))^2] + \mathbb{E}_{x \backsim X}[(1 - D_Y(F(x \cdot W_{Emb}^{\top}) \cdot W_{Emb}^{\top}))^2] + \alpha \cdot \mathbb{E}_{\hat{y} \backsim \hat{Y}}[(1 - \|\nabla_{\hat{y}} D_Y(\hat{y})\|_2)^2]$ while the total loss of the CipherGAN is $\mathcal{L}_{Total}(F, G, D_X, D_Y, X, Y) = \mathcal{L}_{cyc}(F, G, X, Y) + \mathcal{L}_{GAN}(F, D_Y, X, Y) + \mathcal{L}_{GAN}(G, D_X, X, Y)$

The regularization coefficient for the cycle loss is $\lambda = 1$. Squared loss instead of log likelihood which is usually used in GANs for each of the GAN pairs. Adam optimizer [23] is used with learning rate $2e - 4$, $\beta 1 = 0$ and $\beta 2 = 0.9$. The percentage of the corpus used for training is at 0.9 while the rest is used for testing.

The HMMs are trained using data from the brown corpus converted to ciphertext. For the particular homophonic substitution crypto system, the A and B matrix reveal the cipher key length and cipher key respectively. N is increased or decreased until the matrix A converges and this reveals the cipher key length to be equal to N. Then the expected frequency statistics are compared for each of the states in the B matrix and the 'shift' is the corresponding keyword letter [3]. Then the minimum ciphertext required for convergence is reduced gradually in successive experiments.

4.4 Discussion

The ability of the HMM from Vobbilisetty et al. [24] for HMM on homophonic substitution cipher as seen in Figure 5 can be compared to that for the Vigenère cipher. The graph describes the success rate as it climbs and falls based on the data size which here is the ciphertext available and the number of random restarts.

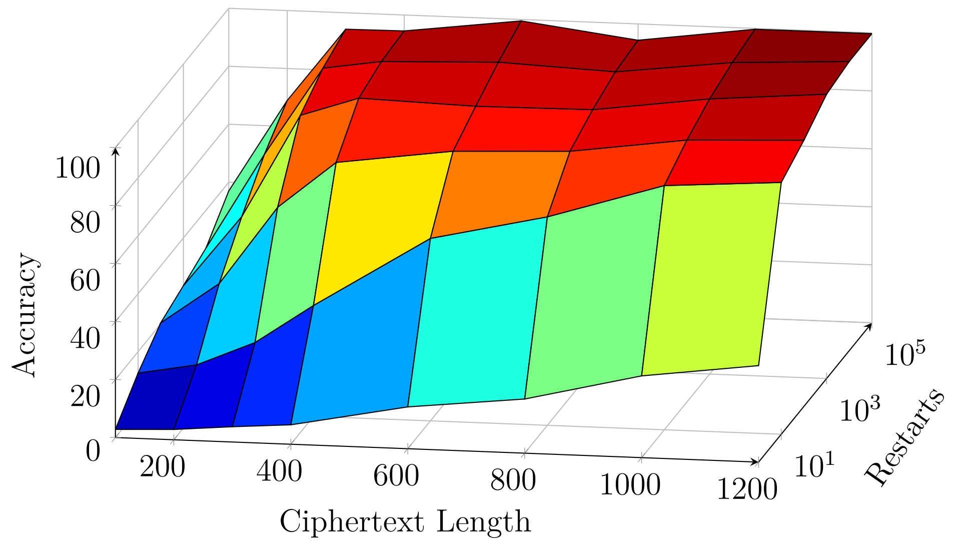

Figure 5: Accuracy vs data size vs restarts (200 iterations)

A result with a little more depth of the underlying cipher system is from Stamp et al. [3] where they also include the key length in the discussion as can be seen in

Table 6 for an HMM trained with 100 random restarts.

Table 6: HMM training results

Keyword	Keyword length	Minimum ciphertext
IT	2	175
DOG	3	250
MORE	4	450
NEVER	5	1200
SECURE	6	1400
ZOMBIES	7	1300

The CipherGAN that can be trained for both words as tokens as well as on the character level has an accuracy of mentioned in the paper. However, while replicating the experiments, the loss was minimized and after stabilization the accuracy was found to be 60% where it was 75% in the paper which is as per the Table 7 [14].

Table 7: Accuracy Of CipherGAN vs data

Word/Char (Vocab Size)	Brown-C (10)	Brown-W (200)	Brown-C (58)
Shift	100%	98.7%	99.8%
Vigenère	99.7%	75.7%	99.0%

Note that vocabulary is the set of characters used to formulate the plaintext.

The ciphertext amount when compared to HMMs is far, far larger. The total training samples used is at around 50,000 and testing is at 5000. Though the maximum number of characters per record is at 100, even so, this is incomparably larger than the amount of ciphertext used in training the HMMs. However the strengths of the CipherGAN are that this amount of data works for even much larger keyword sizes and larger vocabulary size of 200.

The Table 8 compares the hyper-parameters over the various experiments and the resulting metrics with respect to a Vigenère Cipher with key size 3.For Character level the following table is the accuracy of the network for every thousand steps vs the

Table 8: Accuracy Of CipherGAN for number of steps vs vocabulary size with characters

Vocab/ Steps	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
26	6.79	7.74	4.04	2.88	3.75	5.70	3.61	6.29	6.58	8.12
58	0.17	6.06	10.98	8.10	9.90	8.24	6.64	6.82	6.71	6.87
100	6.18	5.41	6.48	6.11	6.14	5.33	4.71	2.03	7.14	5.58

Table 9: Accuracy Of CipherGAN for number of steps vs vocabulary size with words

Vocab/ Steps	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
200	8.32	36.79	51.94	51.44	53.50	57.12	57.43	57.80	57.83	57.82
500	0.04	8.15	39.89	42.22	40.30	36.38	47.80	50.95	51.78	52.29
1000	0.04	17.75	18.58	9.40	28.78	34.93	38.36	41.74	43.34	45.60
2000	0.00	4.89	0.00	0.00	0.00	6.15	17.84	25.24	22.78	22.24
5000	13.21	5.41	2.59	0.08	3.64	0.00	0.00	0.00	0.00	0.00

vocabulary size of the plaintext. In this case it is the number of characters (26 letters of the alphabet for the first column, some punctuation marks and special characters for the second column). Some of hyper-parameters were purely experimental. The vocabulary size in these remain limited to the maximum possible number of characters, but also include the additional blank token.

The Table 9 compares the hyper-parameters over the various experiments and the resulting metrics with respect to a Vigenère Cipher with key size 3 but at word level. That is, the vocabulary of the plaintext is actually directly English words in this case from the Brown corpus. The upper limit on the number of words being trained on is kept to 5000 which has a coverage of 98.5% of the vocabulary in a teen novel as per a well established study on the English language [25].The Table 8 is the accuracy of the network for every thousand steps vs the vocabulary size of the plaintext in words.

Following the above tables, surface graphs were plotted as in Figure 6 and Figure 9, corresponding to Table 8 and Table 9 respectively. All future visualizations of the experiments are based on corresponding tables like Table 8 and Table 9.

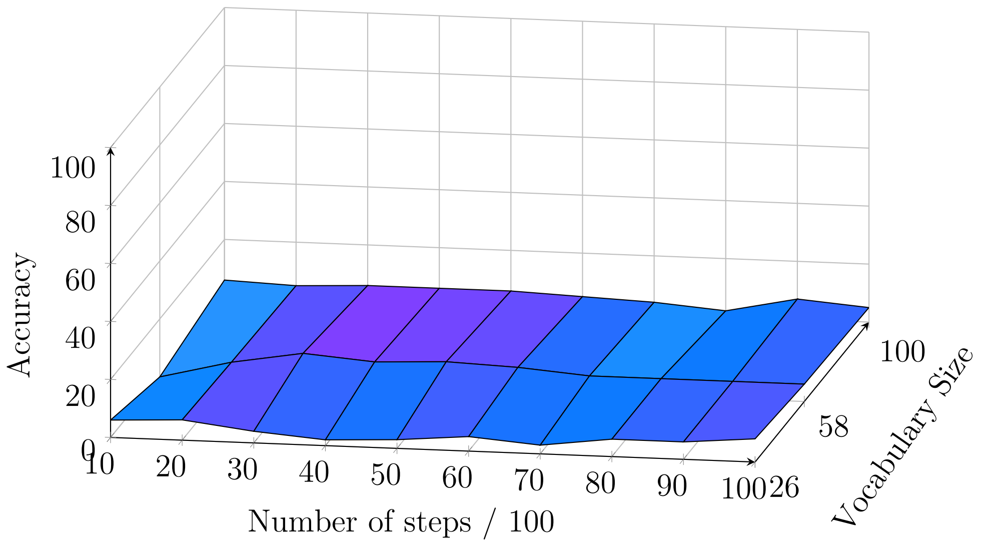

Figure 6: For vocab type char and key size 3: Training Steps vs Accuracy vs Vocab size

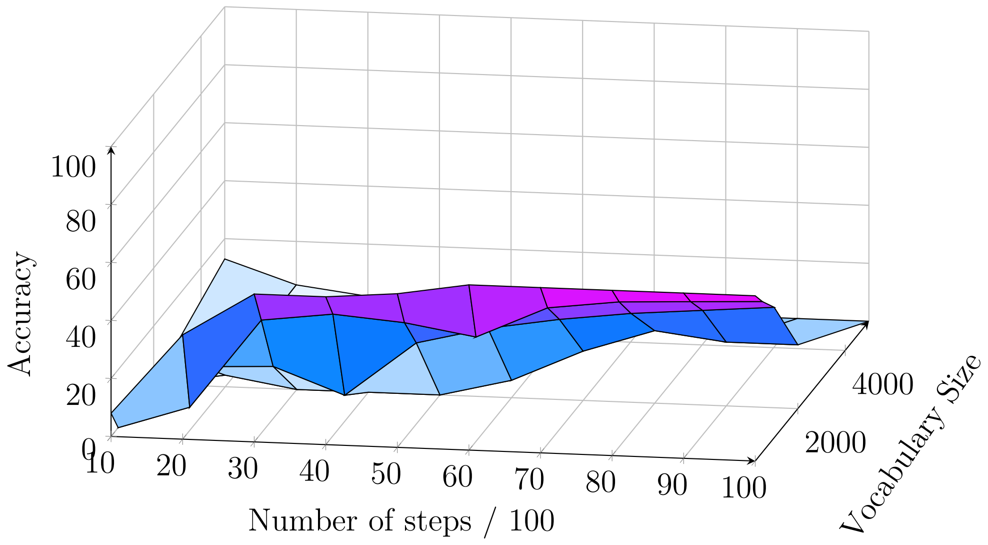

Figure 7: For vocab type words and key size 3: Training Steps vs Accuracy vs Vocab size

The following graphs plot the results of multiple extensive experiments of training the CipherGAN network from scratch in order to study the effects of increasing the key size. A key size of up to 7 was studied on several runs of the CipherGAN and the metrics are as studied in the previous two tables. Recall that, the vocab type can be char level or word level, which denote the tokenization of the English language data.

The graphs in Figure 10 and Figure 11 are the surface plots with the number of training steps on the x-axis, the percentage accuracy on the y-axis and the vocabulary size on the z-axis for Vigenère cipher key size 4.

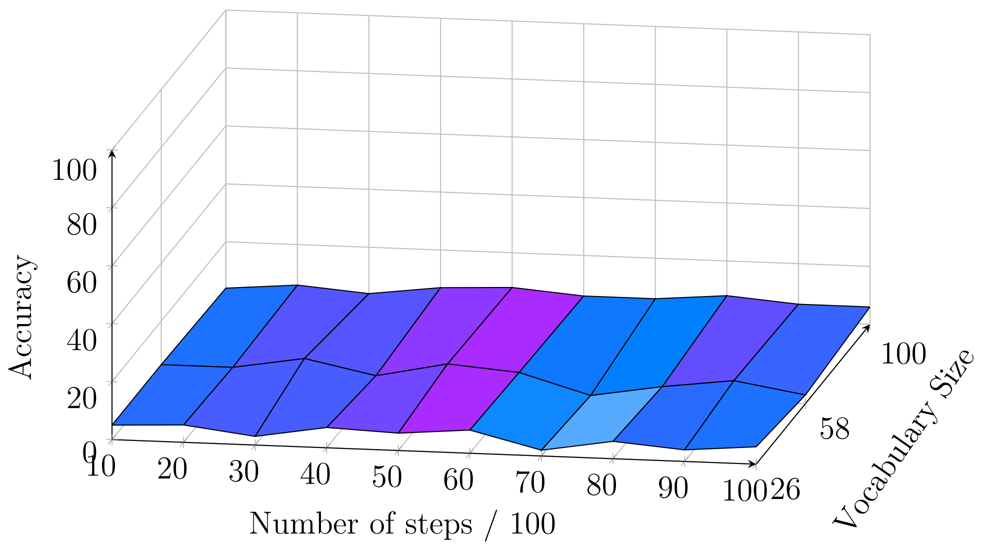

Figure 8: For vocab type char and key size 4: Training Steps vs Accuracy vs Vocab size

The graphs in Figure 10 and Figure 11 are the surface plots with the number of training steps on the x-axis, the percentage accuracy on the y-axis and the vocabulary size on the z-axis for Vigenère cipher key size 5.

The graphs in Figure 12 and Figure 13 are the surface plots with the number of training steps on the x-axis, the percentage accuracy on the y-axis and the vocabulary size on the z-axis for Vigenère cipher key size 6.

The graphs in Figure 14 and Figure 15 are the surface plots with the number of training steps on the x-axis, the percentage accuracy on the y-axis and the vocabulary size on the z-axis for Vigenère cipher key size 7.

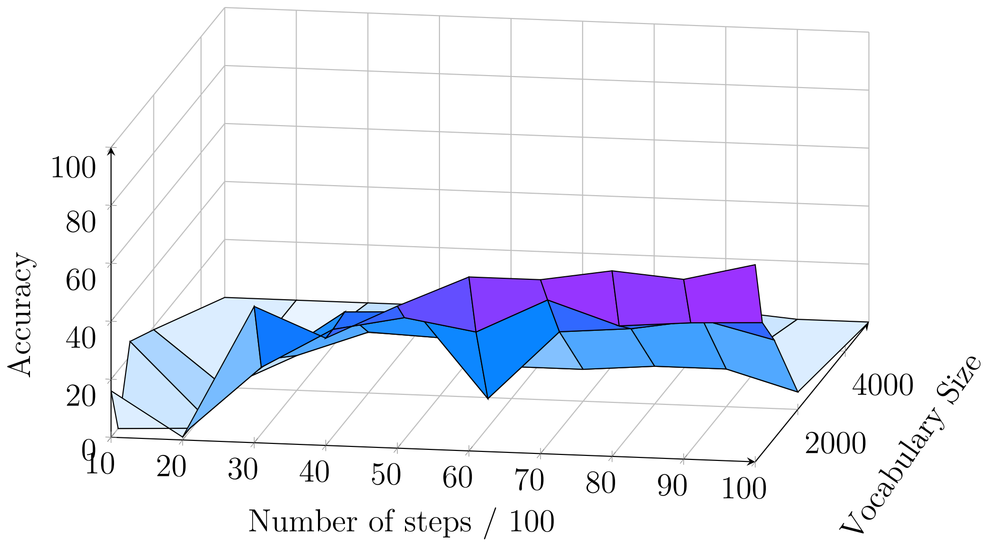

Figure 9: For vocab type words and key size 4: Training Steps vs Accuracy vs Vocab size

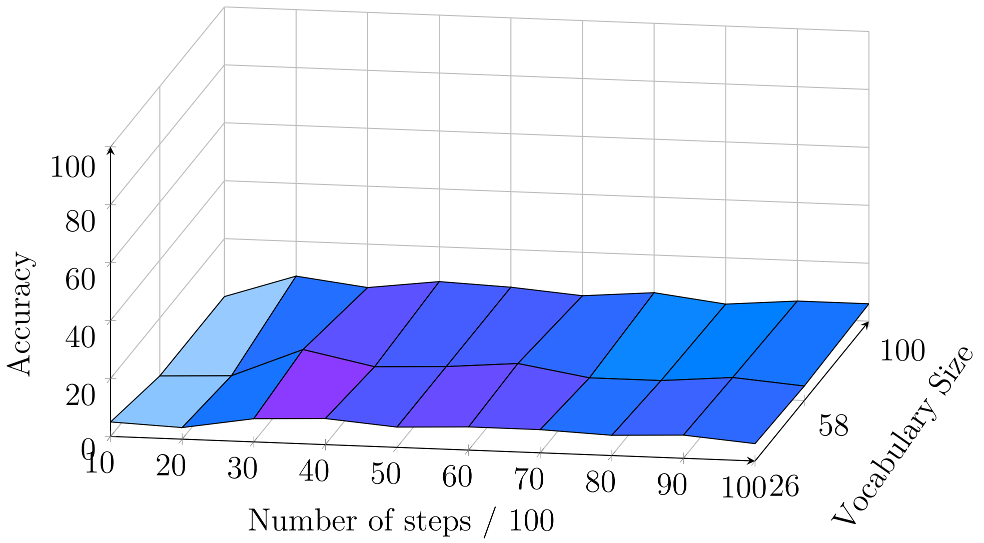

Figure 10: For vocab type char and key size 5: Training Steps vs Accuracy vs Vocab size

The following Table 10 summarizes the results for all the previous experiments. Note that the maximum accuracy throughout the training period is considered and not just the accuracy at the final training step. The ac curacies are plotted for key

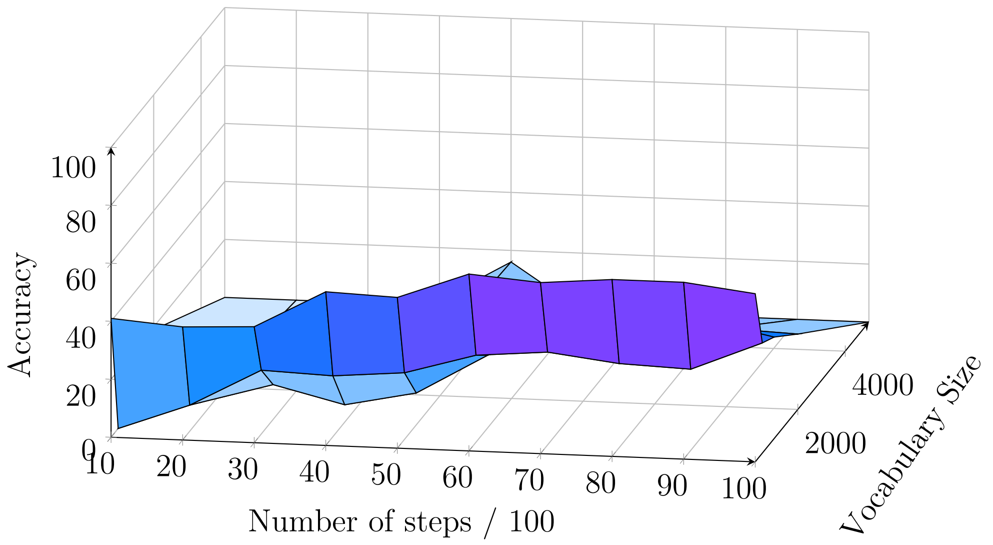

Figure 11: For vocab type words and key size 5: Training Steps vs Accuracy vs Vocab size

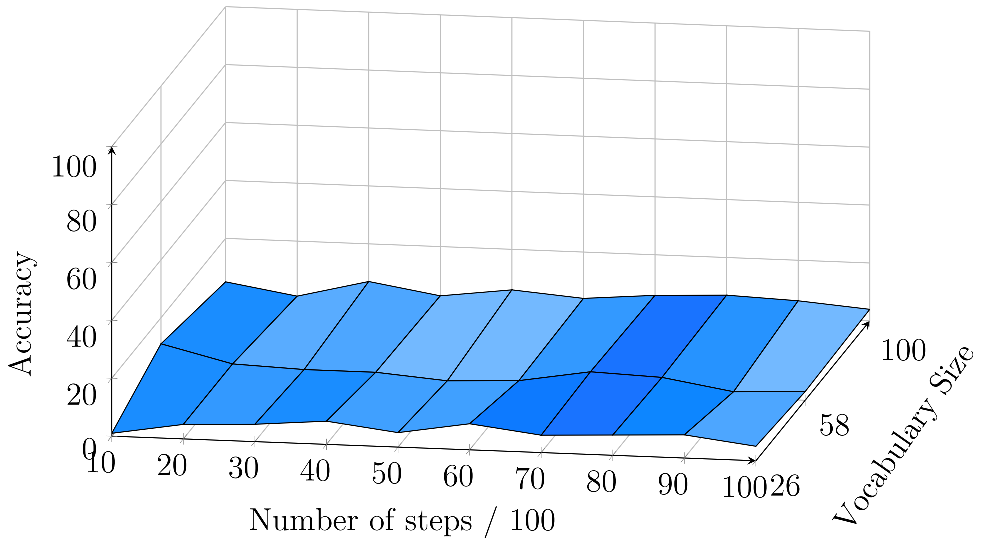

Figure 12: For vocab type char and key size 6: Training Steps vs Accuracy vs Vocab size

size vs vocab size.

From the experiments and previous work [24], it can be seen that the HMMs perform better with limitations in training resources and especially so for a limited

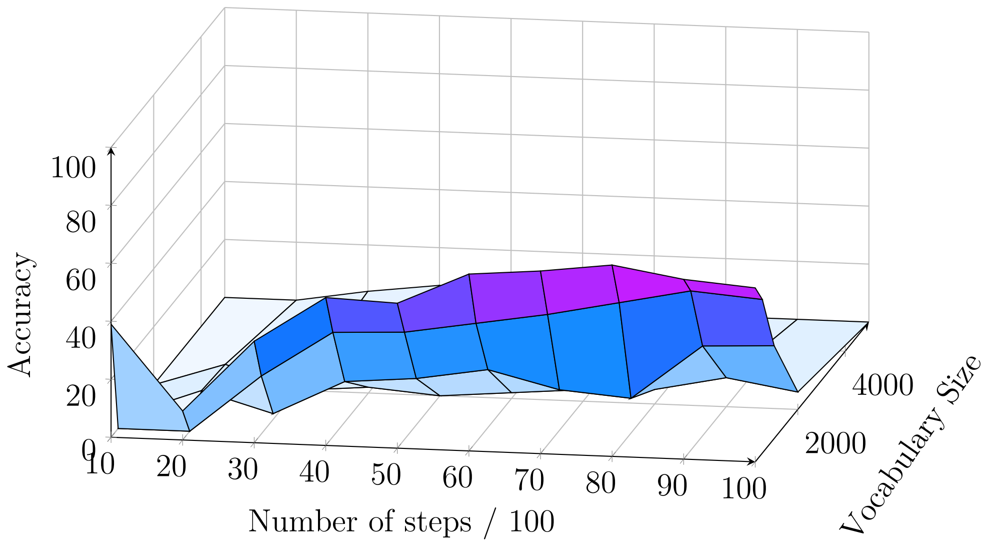

Figure 13: For vocab type words and key size 6: Training Steps vs Accuracy vs Vocab size

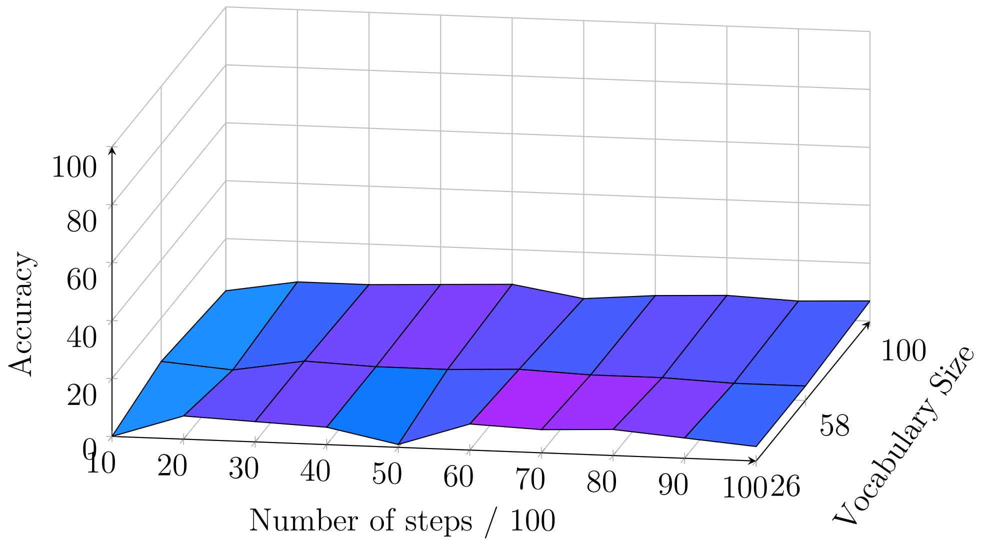

Figure 14: For vocab type char and key size 7: Training Steps vs Accuracy vs Vocab size

amount of ciphertext at the char level. Thus, the CipherGANs were further experimented to investigate if higher accuracies were achievable for networks aimed at cracking a computationally "easier cipher", the shift cipher or Caesar's cipher as

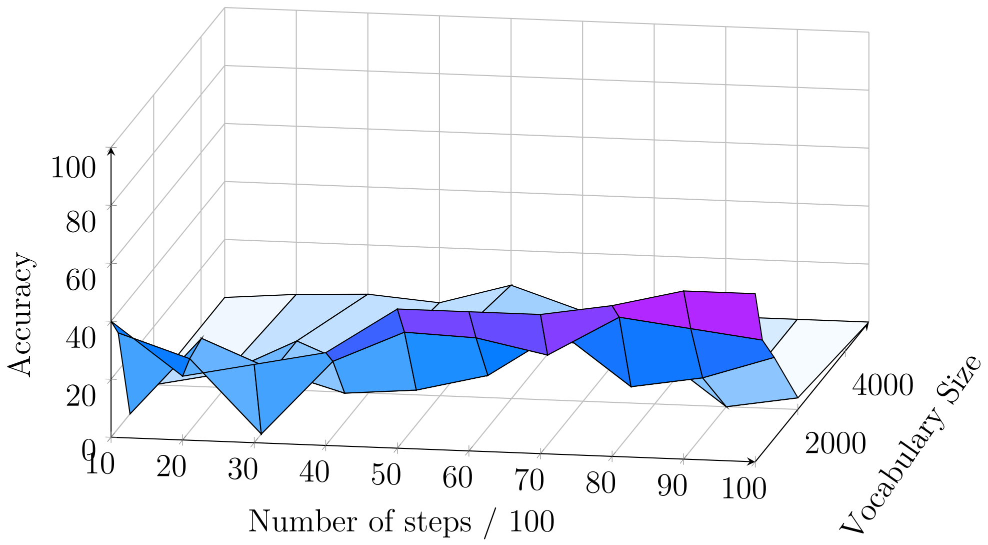

Figure 15: For vocab type words and key size 7: Training Steps vs Accuracy vs Vocab size

Table 10: Maximum accuracy achieved while training

Key Size/ Vocab (Char/Word)	26c	58c	100c	200w	500w	1000w	2000w	5000w
3	8.12	10.98	7.14	57.83	52.29	45.60	22.78	13.21
4	8.11	9.12	8.96	68.87	50.75	40.11	21.43	2.15
5	9.37	11.95	8.15	61.52	38.04	40.85	27.66	16.01
6	9.35	11.95	7.71	66.84	55.77	32.06	10.83	7.42
7	9.68	7.63	8.98	58.39	45.94	34.04	20.84	8.11

explained in Section 2.2.1 of Chapter 2.

The graphs in Figure 16 and Figure 17 are the surface plots with the number of training steps on the x-axis, the percentage accuracy on the y-axis and the vocabulary size on the z-axis for a Shift cipher with a key of 3.

A summary of which can be found in Table 11.

Table 11: Maximum accuracy achieved while training for shift cipher

Vocabulary size (type: c/w)	26c	100c	200w	2000w	5000w
Accuracy (%)	9.31	7.83	68.30	44.57	15.00

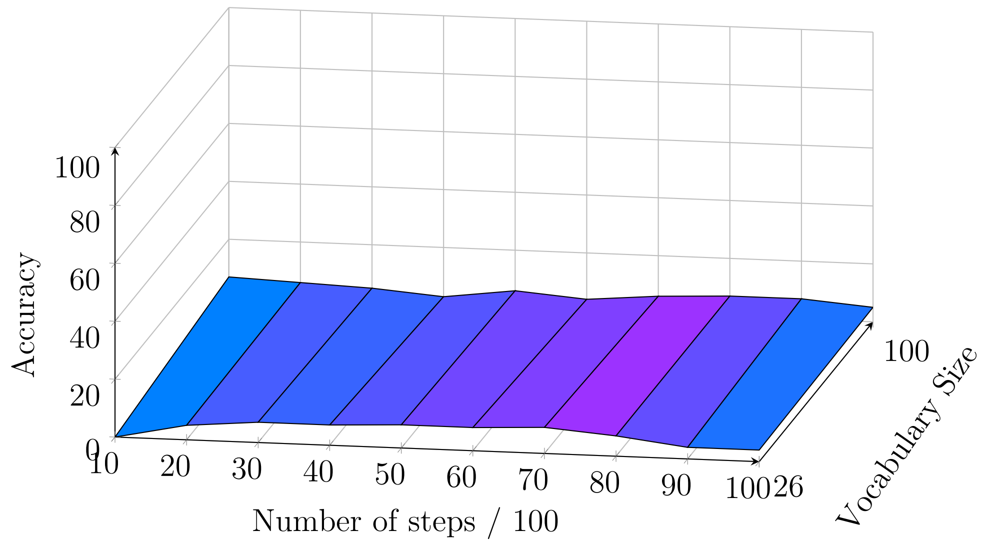

Figure 16: For vocab type 'char' and key 3: Training Steps vs Accuracy vs Vocab size

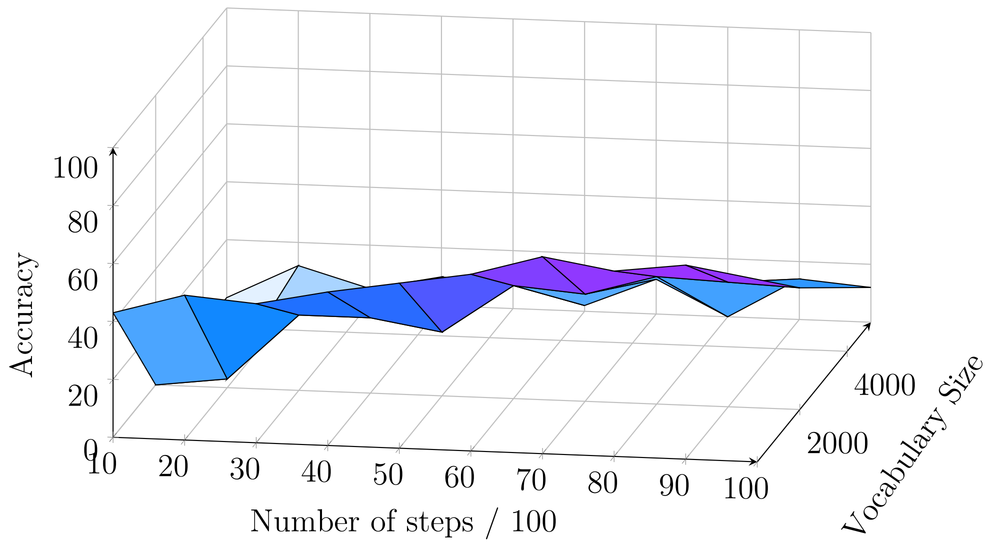

Figure 17: For vocab type 'words' and key 3: Training Steps vs Accuracy vs Vocab size

CHAPTER 5

Conclusion

For simple substitution encryption the models both displayed good results, however there already are many resource-wise cheaper methods for solving simple substitution ciphers, such as the basic frequency analysis. Polyalphabetic ciphers were the answer the next step in the development of cryptology in order to flatten frequency analysis. Therefore, this paper's study of applying machine learning to polyalphabetic ciphers is a crucial gap in the literature that has been filled.

The HMM models with various numbers of random restarts did well on these polyalphabetic ciphers [24]. The CipherGAN too proved to be powerful and capable of solving the polyalphabetic Vigenère cipher. However, the large amount of training data and training steps must be kept in mind and is a serious shortcoming, especially when compared to the HMMs that are capable of cracking the code with a relatively small amount of ciphertext [3].

Thus although the CipherGANs are a compelling demonstration of unsupervised language translation their applicability to cryptanalysis is debatable. They may be more suitable for general unsupervised translation for language generation purposes. Though, it must be noted that the CipherGANs capacity to solve for relatively large vocabulary sets, is remarkable. This could have applications in advanced cryptography which commonly use S-boxes that act as a sort of scale-up or scale-down for the vocabulary set in the encryption and decryption process.

Future work could try more advanced polyalphabetic ciphers and compare and test the advantages of the CipherGAN over other models. Additionally transfer learning can be experimented with on the CipherGAN against various encryption systems to see if it can learn each system and use that to crack a similar system. Such as, keeping the enciphering method common but having different keys, or with closely related polyalphabetic crypto-systems. Thus, the efficiency and effectiveness of the classic HMMs and the deep learning model CipherGAN were compared for classic cryptanalysis.

LIST OF REFERENCES

[1] F. L. Bauer, *Decrypted secrets: methods and maxims of cryptology.* Springer Science & Business Media, 2002.

[2] T. Dao, ‘‘Purple cipher: Simulation and improved hill-climb attack,’’ http://www.cs.sjsu.edu/faculty/stamp/papers/180H.pdf, 2005.

[3] P. K. Basavaraju, ‘‘Heuristic search cryptanalysis of the zodiac 340 cipher,’’ in *Master's Projects.*, vol. 56. Department of Computer Science, San Jose, 2009. [Online]. Available: https://doi.org/10.31979/etd.4krr-6vhr

[4] M. Stamp, F. D. Troia, and J. Huang, ‘‘Hidden markov models for vigenère cryptanalysis,’’ in *Linköping Electronic Conference Proceedings*, vol. 149, no. 11, 2018, pp. 39--46, accessed on: Feb. 10, 2021. [Online]. Available: https://ep.liu.se/ecp/149/011/ecp18149011.pdf

[5] A. N. Gomez, S. Huang, I. Zhang, B. M. Li, M. Osama, and L. Kaiser, ‘‘Unsupervised cipher cracking using discrete gans,’’ *arXiv preprint arXiv:1801.04883*, Jan. 2018.

[6] C. E. Shannon, ‘‘Communication theory of secrecy systems,’’ *The Bell System Technical Journal*, vol. 28, no. 4, pp. 656--715, 1949.

[7] M. Stamp, *Introduction to machine learning with applications in information security.* CRC Press, 2017.

[8] M. M. Alani, ‘‘Applications of machine learning in cryptography: A survey,’’ in *Proceedings of the 3rd International Conference on Cryptography, Security and Privacy*, ser. ICCSP '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 23–27. [Online]. Available: https://doi.org/10.1145/3309074.3309092

[9] A. Dhavare, R. M. Low, and M. Stamp, ‘‘Efficient cryptanalysis of homophonic substitution ciphers,’’ *Cryptologia*, vol. 37, no. 3, pp. 250--281, 2013. [Online]. Available: https://doi.org/10.1080/01611194.2013.797041

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ‘‘Imagenet classification with deep convolutional neural networks,’’ *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097--1105, 2012.

[11] H. B. Barlow, ‘‘Unsupervised learning,’’ *Neural Computation*, vol. 1, no. 3, pp. 295--311, 1989.

[12] P. J. Werbos, ‘‘Backpropagation through time: what it does and how to do it,’’ *Proc. of the IEEE*, vol. 78, no. 10, pp. 1550--1560, 1990.

[13] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, ‘‘Language models are unsupervised multitask learners,’’ *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, ‘‘Attention is all you need,’’ *arXiv preprint arXiv:1706.03762*, 2017.

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, ‘‘Generative adversarial nets,’’ in *Advances in Neural Information Processing Systems*, vol. 27, no. 5, 2014, pp. 2672--2680.

[16] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, ‘‘Unpaired image-to-image translation using cycle-consistent adversarial networks,’’ in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2223--2232.

[17] E. Jang, S. Gu, and B. Poole, ‘‘Categorical reparameterization with gumbel-softmax,’’ *arXiv preprint arXiv:1611.01144*, 2016.

[18] M. Stamp, ‘‘A revealing introduction to hidden markov models,’’ *Department of Computer Science San Jose State University*, pp. 26--56, 2004. [Online]. Available: https://www.cs.sjsu.edu/~stamp/RUA/HMM.pdf

[19] R. Cave, ‘‘Hidden markov models for english,’’ *Proceedings Symposium on the Application of Hidden Markov Models to Text and Speech*, pp. 16--56, 1980.

[20] L. Rabiner and B. Juang, ‘‘An introduction to hidden markov models,’’ *IEEE ASSP Mag.*, vol. 3, no. 1, pp. 4--16, 1986.

[21] T. Berg-Kirkpatrick and D. Klein, ‘‘Decipherment with a million random restarts,’’ in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 874--878.

[22] W. N. Francis and H. Kucera, ‘‘Brown corpus,’’ *Dept. of Linguistics, Brown Univ., Providence, Rhode Island*, vol. 1, 1964.

[23] D. P. Kingma and J. Ba, ‘‘Adam: A method for stochastic optimization,’’ *arXiv preprint arXiv:1412.6980*, 2014.

[24] R. Vobbilisetty, F. Di Troia, R. M. Low, C. A. Visaggio, and M. Stamp, ‘‘Classic cryptanalysis using hidden markov models,’’ *Cryptologia*, vol. 41, no. 1, pp. 1--28, 2017.

[25] N. Schmitt and M. Mccarthy, *Vocabulary: Description, Acquisition and Pedagogy.* Cambridge University Press, 1997.