# UNSUPERVISED CIPHER CRACKING USING DISCRETE GANS

Morten Munk

January 2026

AAU CPH - SW9

# Contents

1. Introduction ..... 2
2. Key Concepts ..... 4
   2.1 Caesar Shift ..... 5
   2.2 Vigenère ..... 7
   2.3 GAN ..... 11
3. Related Work ..... 14
   3.1 CycleGAN ..... 15
4. Ideas & Results ..... 17
   4.1 CipherGAN ..... 18
   4.2 Data ..... 21
   4.3 Results ..... 22
5. Criticism ..... 24
6. Relevance to my project ..... 26

# 1. Introduction

## Premise of this paper

*Can a neural network be trained to deduce withheld ciphers from unaligned text, without the supplementation of preexisting human knowledge?*

## Historical context

- Decryption was human-guided

## Unsupervised learning

- Unpaired bank of ciphertext/plaintext
- No knowledge of vocab frequencies or cipher keys
- Large vocab (200 elements)

# 2. Key Concepts

# 2.1 Caesar Shift

**PLAIN:** 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

**3-SHIFTED:** 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

**Example**: hello world → khoor zruog

# 2.1 Caesar Shift

**PLAIN:** `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`

**3-SHIFTED:** `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`

**Example**: hello world → khoor zruog

**Weakness**

- Trivial to solve with frequency analysis

# 2.2 Vigenère

**PLAIN:** `hello world`

**KEY:** `soda`

**KEYSTREAM:** `sodas odaso`

**ENCRYPTED:** `zsolg krrdr`

| | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 |

# 2.2 Vigenère

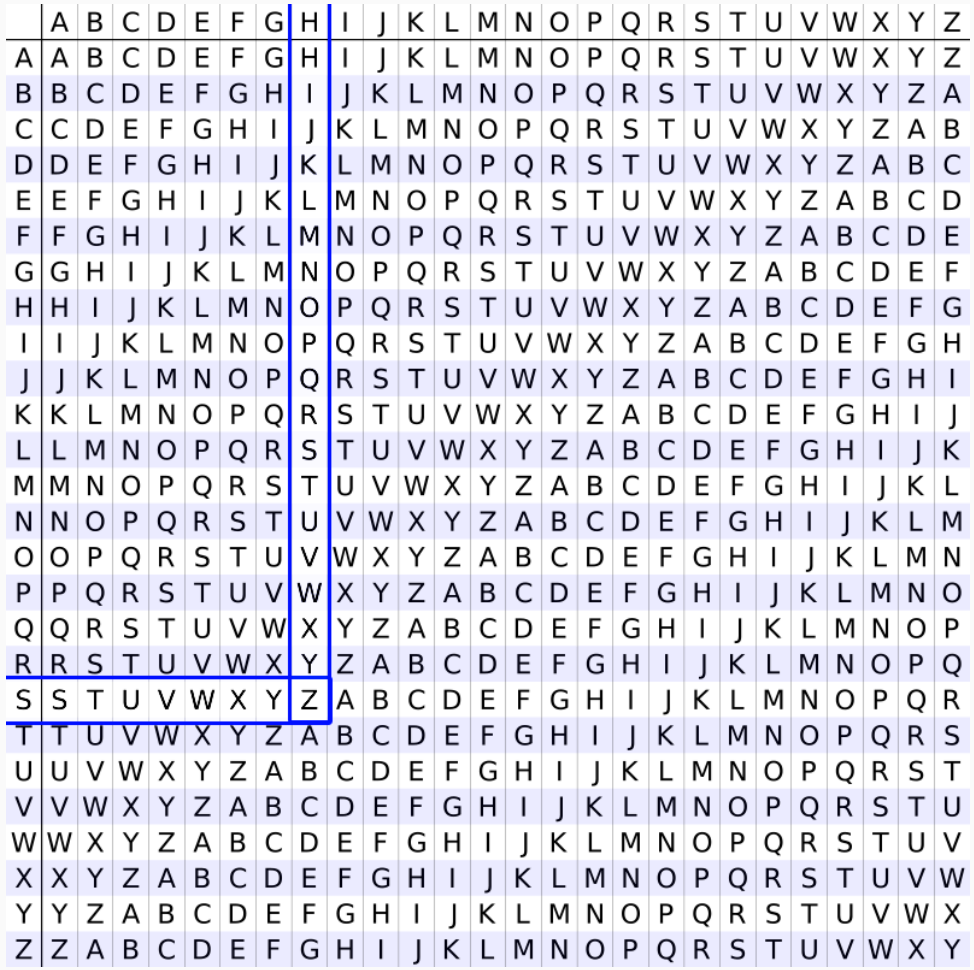
**PLAIN:** `hello world`

**KEY:** `soda`

**KEYSTREAM:** `sodas odaso`

**ENCRYPTED:** `zsolg krrdr`

$(\mathrm{h}, \mathrm{s}) \rightarrow \mathrm{z}$



| | 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 | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 |

**PLAIN:** `hello world`

**KEY:** `soda`

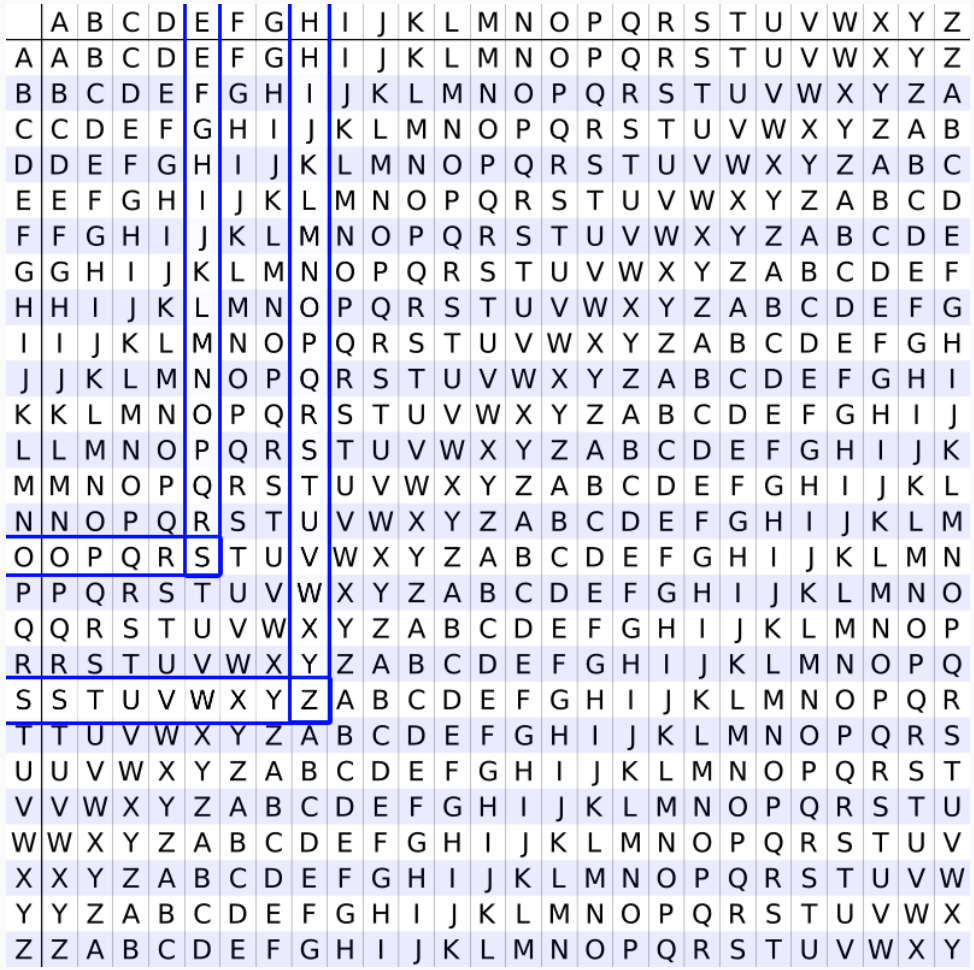
**KEYSTREAM:** `sodas odaso`

**ENCRYPTED:** `zsolg krrdr`

$(\mathrm{h}, \mathrm{s}) \rightarrow \mathrm{z}$

$(\mathrm{e}, \mathrm{o}) \rightarrow \mathrm{s}$

And so on...



| | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 |

**PLAIN:** `hello world`

**KEY:** `soda`

**KEYSTREAM:** `sodas odaso`

**ENCRYPTED:** `zsolg krrdr`

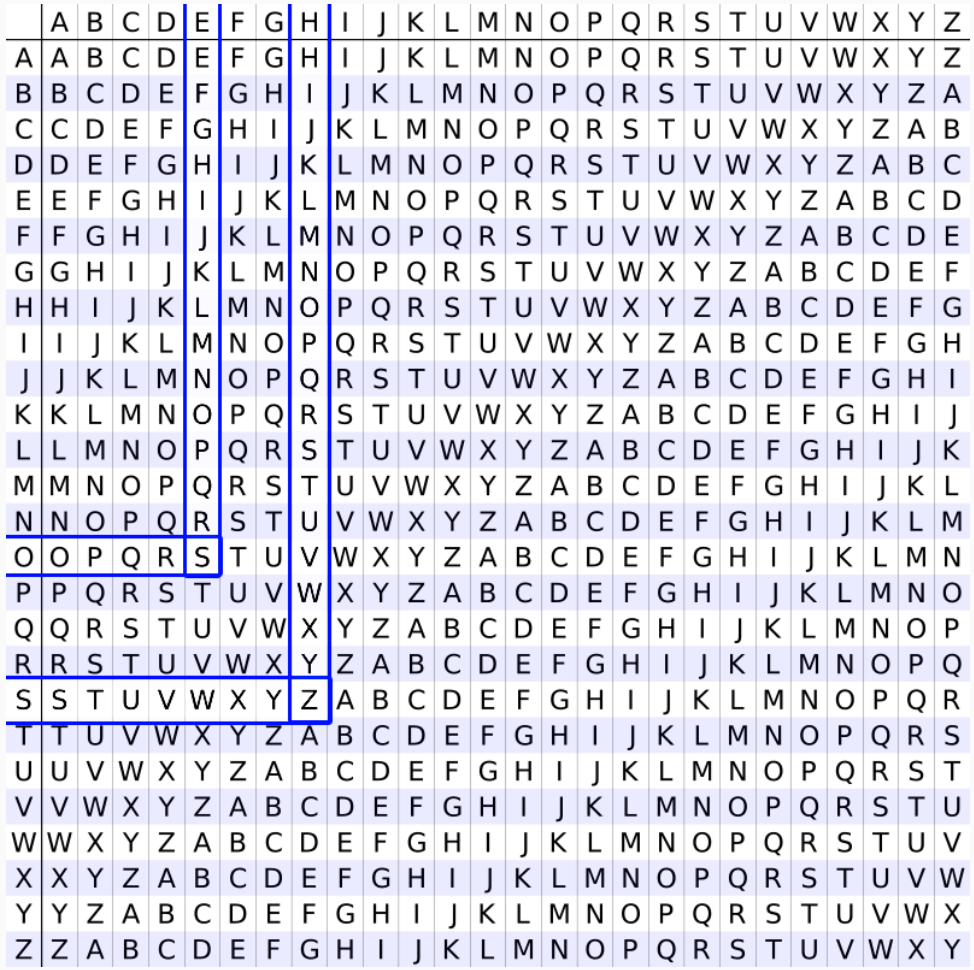
$(\mathrm{h}, \mathrm{s}) \rightarrow \mathrm{z}$

$(\mathrm{e}, \mathrm{o}) \rightarrow \mathrm{s}$

And so on...

**Increased difficulty**

- Frequencies are scrambled



| | 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 | 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 |

**Generator**

- Attempts to produce convincing samples from data distribution
- Denoted as F

**Discriminator**

- Tries to distinguish between true and synthetic samples
- Denoted as G

$$D^* = \arg\max_D E_{x \sim \mathcal{X}}[\log D(x)] - E_{z \sim \mathcal{Z}}[\log(1 - D(F(z)))]$$

$$D^* = \arg\max_D E_{x\sim\mathcal{X}}[\log D(x)] - E_{z\sim\mathcal{Z}}[\log(1 - D(F(z)))]$$

**$D^*$**

- optimal discriminator

**$\arg\max_D$**

- parameters causing highest possible score

**$E_{x\sim\mathcal{X}}$**

- Real data

**$D(x)$**

- Assigned probability
- (e.g. 1 = real, 0 = fake)

**$F(z)$**

- Generator creating fake text

**$1 - D(F(z))$**

- Chance of correctly identifying fakes

$$D^* = \arg\max_D E_{x\sim\mathcal{X}}[\log D(x)] - E_{z\sim\mathcal{Z}}[\log(1 - D(F(z)))]$$

**$D^*$**

- optimal discriminator

**$\arg\max_D$**

- parameters causing highest possible score

**$E_{x\sim\mathcal{X}}$**

- Real data

**$D(x)$**

- Assigned probability
- (e.g. 1 = real, 0 = fake)

**$F(z)$**

- Generator creating fake text

**$1 - D(F(z))$**

- Chance of correctly identifying fakes

**Vulnerable to mode collapse**

- Generator loses diversity - distribution collapses

Figure 2: Discriminators trained on the toy example of recognizing the bottom-right corner of a simplex as true data. From left to right the discriminators were regularized using: nothing; WGAN Jacobian norm regularization; and, the relaxed sampling technique.

## Original GAN can be too strict

- No helpful feedback

## Used by this paper

- Wasserstein Jacobian norm
- Relaxation Sampling

# 3. Related Work

**Two distributions**

- $\mathcal{X}$ & $\mathcal{Y}$

**Two generators**

- $F : \mathcal{X} \to \mathcal{Y}$
- $G : \mathcal{Y} \to \mathcal{X}$

**Two discriminators**

- $D_{\mathcal{X}} : \mathcal{X} \to [0, 1]$
- $D_{\mathcal{Y}} : \mathcal{Y} \to [0, 1]$

**Cycle loss**

- L1 Norm - original text vs. round-trip text
- Forces model to be one-to-one

$$\mathcal{L}_{\text{cyc}}(F, G, \mathcal{X}, \mathcal{Y}) =$$

$$E_{x \sim X}[\| G(F(x)) - x \|_1] + E_{y \sim Y}[\| F(G(y)) - y \|_1]$$

## Consider the losses together

$$\mathcal{L}(F, G, D_y, D_y, \mathcal{X}, \mathcal{Y}) = \underbrace{\mathcal{L}_{\text{GAN}}(F, D_y, \mathcal{X}, \mathcal{Y})}_{\text{forward pass}} + \underbrace{L_{\text{GAN}}(G, D_{\mathcal{X}}, \mathcal{Y}, \mathcal{X})}_{\text{backward pass}} + \lambda * \mathcal{L}_{\text{cyc}}(F, G, \mathcal{X}, \mathcal{Y})$$

## Forward pass

- Generator F: Plaintext to ciphertext for discriminator $D_y$

## Backward pass

- Generator G: Ciphertext to plaintext for discriminator $D_{\mathcal{X}}$

## $\lambda$

- Hyperparameter - good translator vs good detective

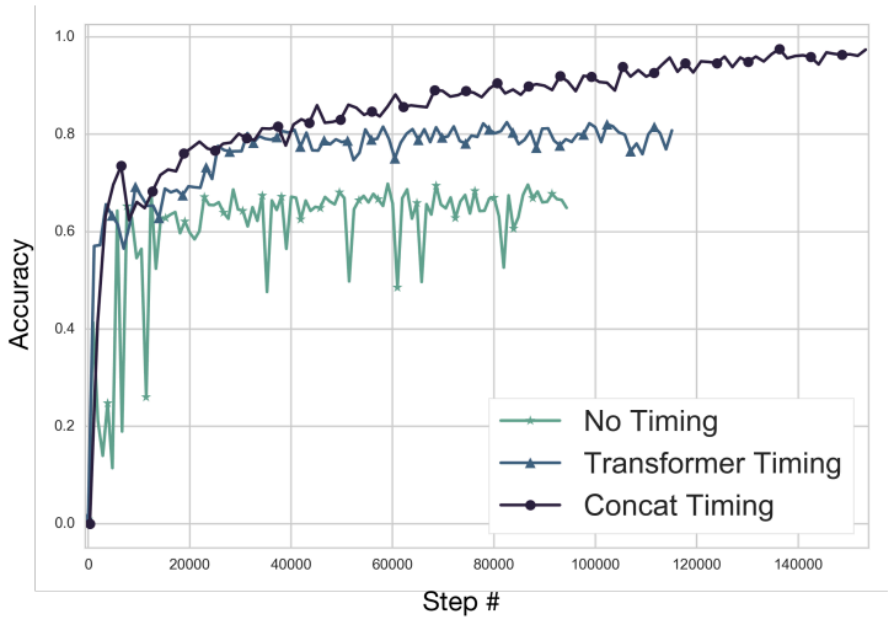
## Cycle

- Ensures diversity - avoid mode collapse

# 4. Ideas & Results

## Words in embedding space

- Discrete choices do not produce gradients



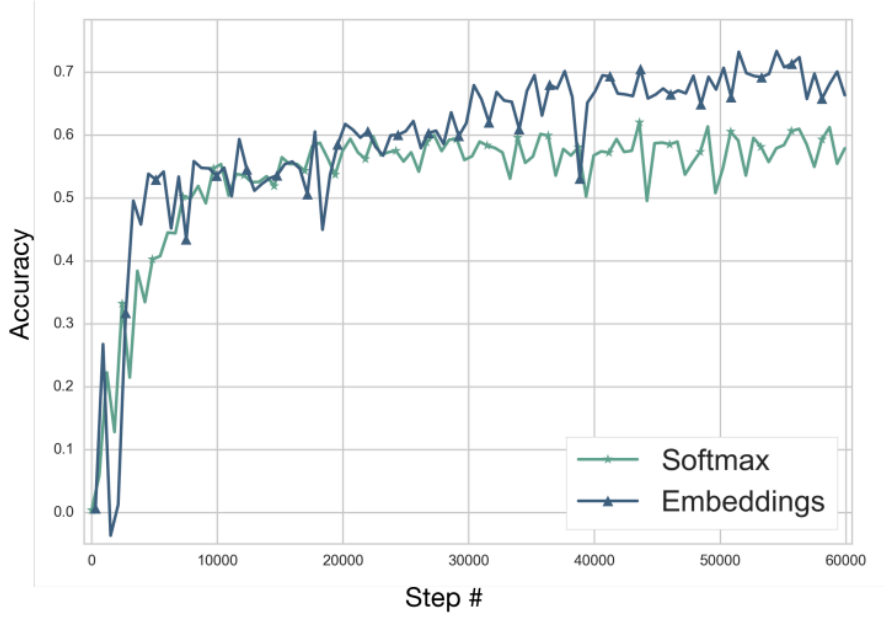
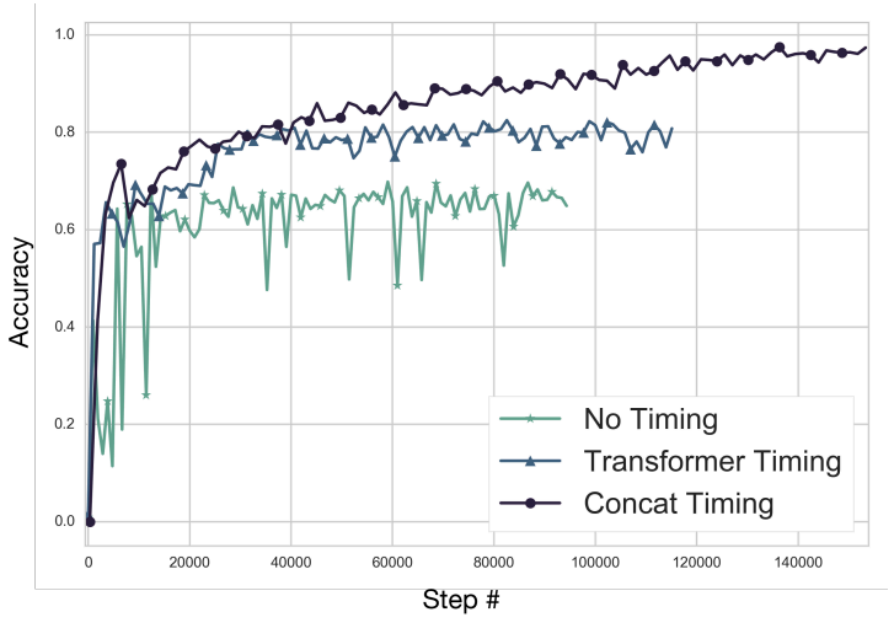




Figure 3: Left: Comparison of different timing techniques for Brown-C Vigenère. Right: Comparison of embedding vs. raw softmax on Brown-W with vocab size of 200.

## WGAN-GP (Jacobian Norm)

- Limit learning rate of discriminator
- Smooth training signal for generator



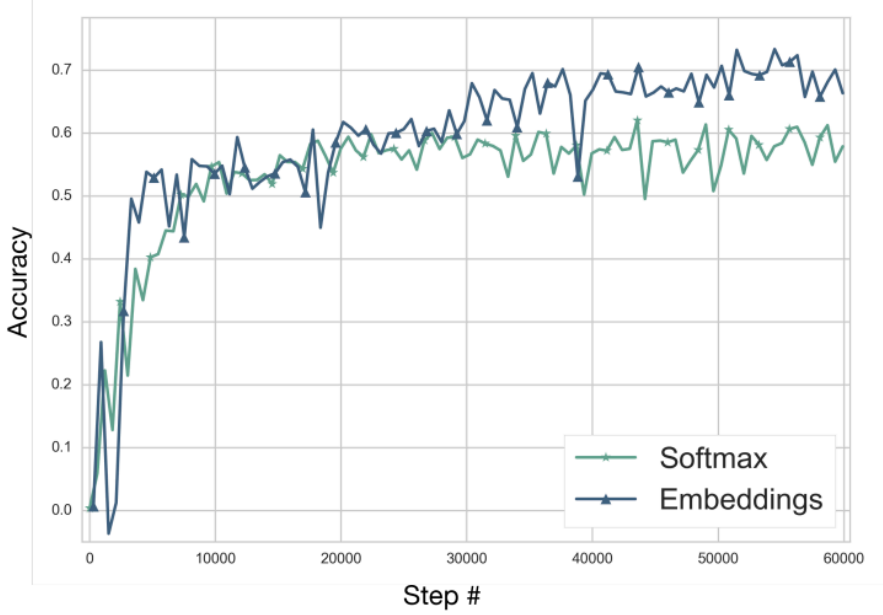
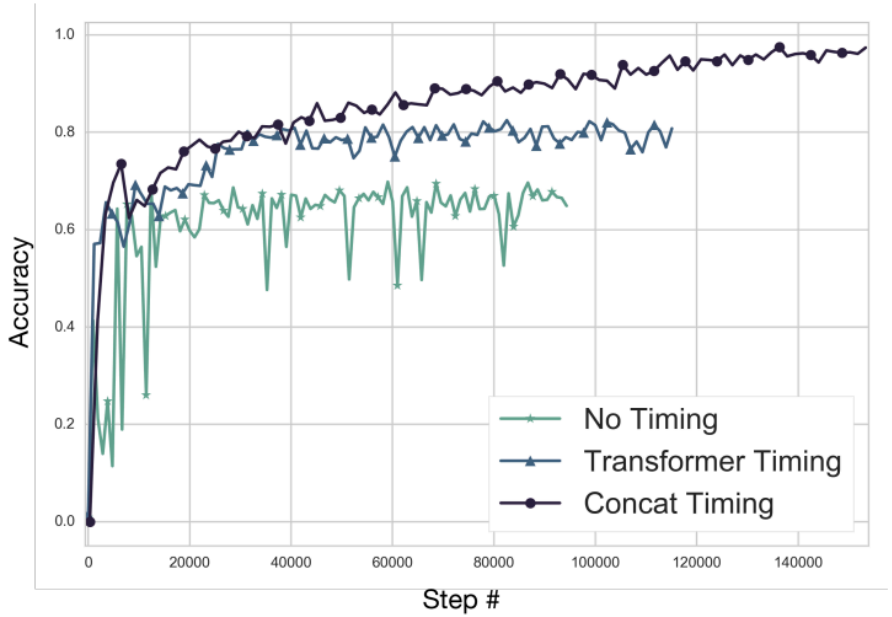




Figure 3: Left: Comparison of different timing techniques for Brown-C Vigenère. Right: Comparison of embedding vs. raw softmax on Brown-W with vocab size of 200.

## Positional embedding (Timing)

- Vigenère relies on positioning
- Tag each letter with position index



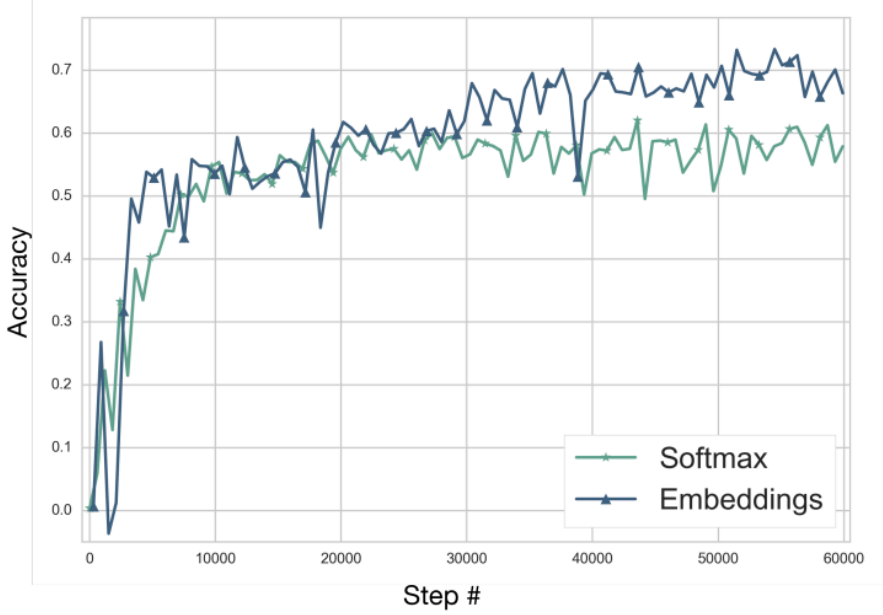




Figure 3: Left: Comparison of different timing techniques for Brown-C Vigenère. Right: Comparison of embedding vs. raw softmax on Brown-W with vocab size of 200.

### Brown English text dataset

- 1st half as CycleGAN $\mathcal{X}$ distribution
- 2nd half as ciphers in $\mathcal{Y}$ distribution

### Brown English-Language corpus

- Natural language plaintext
- Over 1 mil. words
- Both word-level and char-level

### Brown-W

- Top 200 most frequent words
- Rest is blank

# 4.3 Results

| **Data**<br>**Vocab size** | Brown-W<br>10 | Brown-W<br>200 | Brown-C<br>58 | Freq. Analysis (With Key)<br>58 | <br>200 |
|---|---|---|---|---|---|
| Cipher | **Shift/Permutation** | | | | |
| Acc. | 100% | 98.7% | 99.8% | 80.9% | 44.5% |
| Cipher | **Vigenère (Key: "345")** | | | | |
| Acc. | 99.7% | 75.7% | 99.0% | 9.6% (78.1%) | <0.1% (44.3%) |

Table 2: Average proportion of characters correctly mapped in a given sequence. The "Freq. Analysis" column is simple frequency analysis applied to the same corpus our model observes. For Vigenère we also show the score if the key were known (note: the key is left unknown to our model).

| Work | Ciphertext Length | Accuracy |
|---|---|---|
| Hasinoff (2003) | 500 | $\sim 97\%$ |
| Forsyth & Safavi-Naini (1993) | 5000 | $\sim 100\%$ |
| Ramesh et al. (1993) | 160 | $\sim 78.5\%$ |
| Verma et al. (2007) | 1000 | $\sim 87\%$ |

Table 1: Previous results on automated shift cipher cracking with limited ciphertext length.

# 5. Criticism

# 5. Criticism

**Why not test on a real cipher?**

- E.g. Civil War Vigenere cipher
- Real ciphers has typos, etc.

**Sequence length bottleneck**

- 200 chars is short!
- GANs are likely to become unstable

**Why only compare against non-neural approaches?**

- Other neural baselines?

# 5. Criticism

**Why not test on a real cipher?**

- E.g. Civil War Vigenere cipher
- Real ciphers has typos, etc.

**Sequence length bottleneck**

- 200 chars is short!
- GANs are likely to become unstable

**Why only compare against non-neural approaches?**

- Other neural baselines?

**Current better models - (like CausalLM)**

- We know this because we are from the future 😄

# 6. Relevance to my project

# 6. Relevance to my project

## Homophonic Substitution Ciphers

- 1-to-many mappings
- English
- Only lowercase letters
- No spaces

## Comparative Study

- Embeddings
- Heuristics
- Seq-2-Seq

## Increase sequence length with CausalLM

- Linear attention?
- Flash attention?

**Decipherment as Regression: Solving Historical Substitution Ciphers by Learning Symbol Recurrence Relations**

**Nishant Kambhatla** **Logan Born** **Anoop Sarkar**
School of Computing Science, Simon Fraser University
8888 University Drive, Burnaby BC, Canada
{nkambhat, loborn, anoop}@sfu.ca

**Abstract**

Solving substitution ciphers involves mapping sequences of cipher symbols to fluent text in a target language. This has conventionally been formulated as a search problem, to find the decipherment key using a character-level language model to constrain the search space. This work instead frames decipherment as a sequence prediction task, using a Transformer-based causal language model to learn recurrences between

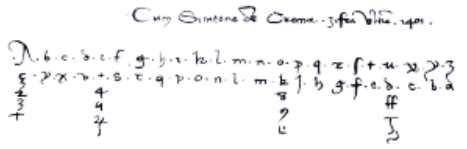


Figure 1: The homophonic substitution key for the *Simeone de Crema* written in Mantua in 1401 AD. The top line maps each character in the alphabet to its reversed-alphabet equivalent; each vowel is substituted by three additional symbols.