Cracking Arbitrary Substitution Ciphers with Statistics-Informed Seq2Seq Models

Yiyang Zhang (yzhan627) | Xiangxi Mu (mxiangxi) | Tianze Hua (thua5) | Gary Zheng (yzheng84)

May 8, 2024

Code | Slides | Devpost

Introduction

The substitution cipher is a traditional encryption technique that substitutes units of the plaintext into other units thereby forming the ciphertext. Each substitution cipher is defined by a key, which is essentially a mapping function specifying how to substitute each plaintext unit. One key weakness of substitution ciphers is that they do not hide the statistics of units in the ciphertext thus prone to a type of attack called frequency analysis. While frequency analysis could already crack substitution ciphers, we are interested in whether modern neural networks could also crack substitution ciphers with less hard-coded components.

Concretely, we aim to build a deep sequence model that could recover ciphertexts and produce the underlying plaintexts. We focus on substitution ciphers that operate on characters and numbers but impose no constraints on the keys (in contrast to Ceaser Cipher, which has extra constraints on the key space). We are motivated by the failure case presented in one of the past deep learning final projects, which achieves only 37% on recovering ciphertexts encrypted with arbitrary substitution ciphers. We believe the use of one-hot encoding ignores essential information about the statistics of units from the ciphertext and can be improved with better choices. We adopt the approach proposed in Aldarrab & May, 2021, which includes the unigram frequency information of the ciphertext into the embedding. Furthermore, we explore the possibility of enhancing the transformer encoder blocks with the use of 1-D CNNs as feature extractors.

Related Work

Can Sequence-to-Sequence Models Crack Substitution Ciphers?
Aldarrab & May employ an end-to-end multilingual model to solve the simple substitution on ciphers. The model is trained on multilingual data so that it can obtain end-to-end decipherment without relying on a separate language ID step. To generalize the model's ability to recover from that noise by inserting, deleting, or substituting characters while generating plaintext, they used Sequence-to-sequence model architecture with frequency encoding.

Character-Level Translation with Self-attention
Gao et al. proposed an architecture, conv-transformer. They adapted each encoder block to include an additional subblock, which they applied to the input representations before applying self-attention. The sub-block consists of three 1D convolutional layers, aimed to resemble character-level interactions of different levels of granularity.

CipherBusters: Automatically Busting Classical Ciphers
In this past deep learning final project, Chung et al. investigated using modern deep learning sequence models (RNNs and Transformers) to crack classical ciphers including Ceaser, Vignere, and arbitrary substitution cipher. While they could solve Ceaser ciphers to high accuracy, their models fail to generalize to Vignere and arbitrary substitution ciphers.

Decipherment of Substitution Ciphers with Neural Language Models
Kambhatla et al. uses large pre-trained neural LMs to crack substitution cipher. They modified the beam-search algorithm and used global scoring of the plaintext message. Compared to the traditional n-gram models, this method results in much lower error rates in cracking challenging ciphers.

Word Translation without Parallel Data
This paper showed a method to build a bilingual dictionary between two languages without using any parallel corpora, by aligning monolingual word embedding spaces in an unsupervised way.

Methodology

Data

We use the **wikitext-103-raw-v1** split from wikitext for our training, validation, and testing data. Wikitext is a language corpus extracted from selected Wikipedia articles. For preprocessing, we only kept letters (all switched to lowercase), numbers, and spaces, while removing all other characters and symbols. The following table introduces the sizes of our training, validation, and testing datasets.

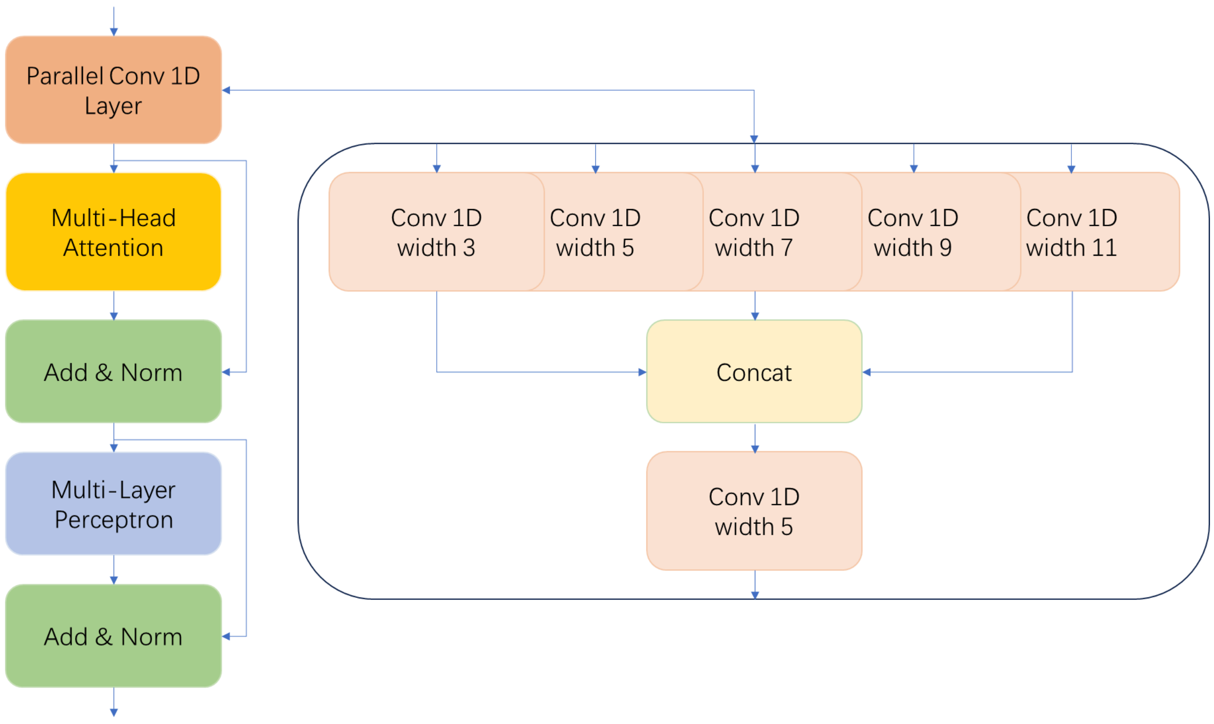
Dataset splits	Number of characters
Train	500,978,592
Validation	1,066,435
Test	1,198,848

Upon training, we plan to dynamically apply our substitution cipher function on the raw text inputs, and encrypt them with different substitution keys.

Model Architecture

Due to the nature of the task – mapping ciphertext characters back to their plaintext counterpart, our model is only required to output the same number of characters as its input. We therefore choose to use an encoder-only transformer architecture, which computes self-attention across its input tokens and outputs a same-length sequence after the last layer. Apart from the usual transformer components, we implement CNN-enhanced encoder blocks such that before the self-attention is computed, we first compute several 1D CNN kernels with different kernel sizes to extract local patterns from the input and then feed that output to the self-attention module. We verified the effectiveness of CNNs in our encoder module in our later ablation studies – CNNs are very good at extracting local patterns and summarizing them into a contextualized representation for single characters which help the encoder to further process.

Below is an illustration of our encoder block (CNN, MultiHead Attention, MLP):

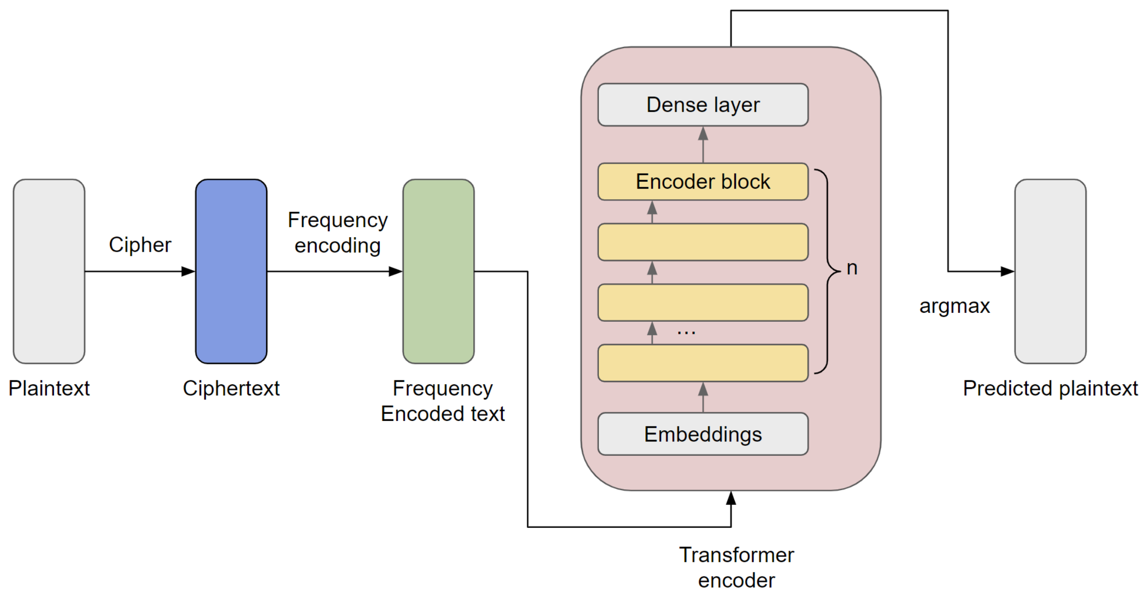


Unigram Frequency Encoder

Because of the enormous possibilities of keys (37!) of the substitution cipher, models in previous works show a suboptimal performance in generalizing across the highly variable input sequences. We adopt one statistics-encoding method used by Kambhatla et al. to help alleviate this problem. Instead of having a dictionary from a specific letter to its embedding, we associate an embedding with the most frequent character in the ciphertext, the second most frequent character, and so on. Thus, the model is learning to recover the ciphertext to plaintext based on a frequency-sorted ciphertext, which helps a lot in our case because the relative frequency of characters in English is relatively stable, and could make the recovering process much easier and generalizable.

Workflow Overview

Below is an illustration of the entire workflow of our model, for training and testing. During training, we randomly initialize a substitution cipher and apply it on a plaintext of specific window size to get the ciphertext. Then we use a frequency encoder to summarize the statistical information in the ciphertext and feed its output to our neural network model. After the transformer encoder, we use 1-hot encoded ground-truth plaintext and cross-entropy function to compute the loss and do backpropagation for training. During inference time, we use an argmax function to extract the predicted labels from the logits.



Metric

We create test and training ciphers from plain text of the wikitext dataset using methods we implemented. We will use the character-level accuracy as an evaluation metric. Under this metric, the accuracy will be the result of dividing the number of correct characters in the predicted sentence by the number of characters in the original plain-text sentence.

Results

We present our results below on plaintext prediction using our models. We compared our results with the baseline accuracy and previous DL projects in Table 1 and explored the accuracy of models with different test/training window sizes in Table 2. Finally, In Table 3, we conducted ablation studies to understand the contribution of each component of the system.

Baselines

In our experiment, we compared our model performance against a baseline accuracy from the unigram frequency decipher. The unigram frequency decipher computes the frequency of each individual character in the given ciphertext, and matches characters with the universal character frequency of the entire dataset. The assumption is that our model should significantly improve the accuracy rates compared to the baseline results. In addition, we also compared our results with the result of a previous 2470 project in 2021 (The only data point they reported was 0.38).

Performance Report

Our model performance can be found in Table 1. The accuracy rates in Table 1 come from models that were trained and tested on the same window size. Compared to the baseline unigram frequency decipher and the previous study CipherBuster, our models performed at a significantly higher accuracy rate.

Window Size	**Baseline**	**CipherBuster**	**Ours**
64	0.270	0.38	**0.649**
128	0.321	0.38	**0.817**
256	0.360	0.38	**0.931**
512	0.392	0.38	**0.96**
1024	0.425	0.38	**0.974**

Table 1: Model performance of different window sizes compared to unigram frequency baseline and Cipher-Busters (DL2021).

Effect of Window Sizes in Training and Inference

We also explored the effect of window size on accuracy rates. We explored different test/train window size combinations, and the results are in Table 2. Intuitively, the model performance was the highest when the test window size was equal to the training window size. However, it is worth noting that for models with longer training window sizes, their performance in predicting shorter texts was also promising.

Test/Train	**64**	**128**	**256**	**512**	**1024**
64	**0.649**	0.616	0.599	0.521	0.495
128	-	**0.817**	0.814	0.743	0.71
256	-	-	**0.931**	0.89	0.862
512	-	-	-	**0.96**	0.948
1024	-	-	-	-	**0.974**

Table 2: Decipherment performance of models trained with different window sizes, on ciphertexts of different lengths.

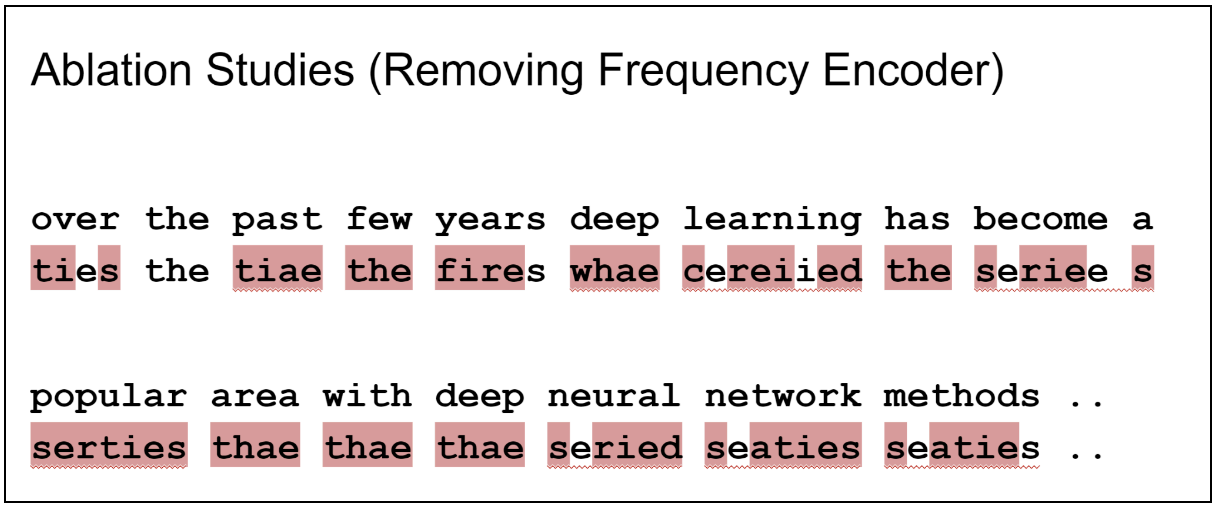
Ablation Studies

Furthermore, we did ablation studies to investigate the role of each module in our architecture.

Window Size	**w/o Freq Encoder**	**w/o CNN**	**w/o Self Attention**	**Full Model**
64	0.339	0.282	0.430	**0.649**
128	0.342	0.325	0.507	**0.817**
256	0.343	0.353	0.585	**0.931**
512	0.346	0.393	0.674	**0.960**
1024	0.346	0.434	0.735	**0.974**

Table 3: Ablation study: Full model performance compared with models with removed components.

When the frequency encoder is removed, we found that the overall accuracy to drop significantly, and they do not improve with a larger window size. Inspecting example outputs from the model, we found that the model used mostly only the word length information (implied by the position of the spaces) to predict the output. For ciphertext words with completely different letters, the model might predict the same plaintetxs for them (“thae” for instance). This is most likely due to the fact that there are 37 factorial numbers of different keys, and the model was having a difficult time generalizing over all the keys in the enormously large key space.

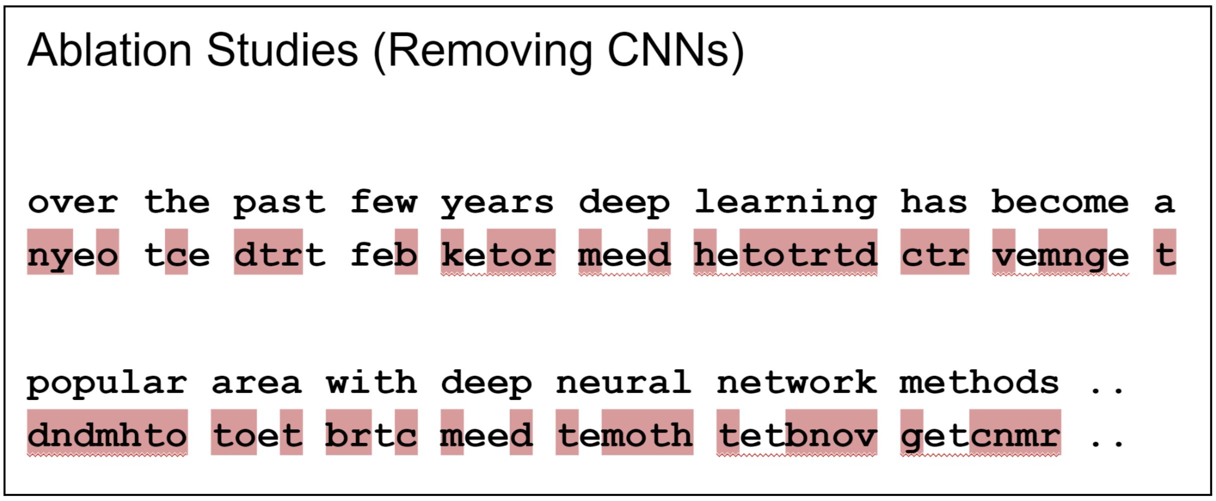


Ablation Studies (Removing Frequency Encoder)

```
over the past few years deep learning has become a
ties the tiae the fires whae cereiied the seriee s

popular area with deep neural network methods ..
serties thae thae thae seried seaties seaties ..
```

When the CNN is removed, we found that the model performs better than without the frequency encoder, but still significantly worse than the whole model (and only comparable to the unigram frequency baseline). However, with self-attention modules, we found that our model can impose a global mapping constraint (the same characters in the ciphertexts will be mapped to the same characters in the predicted plaintext). This consistency property is a desired feature of our decipherment model.

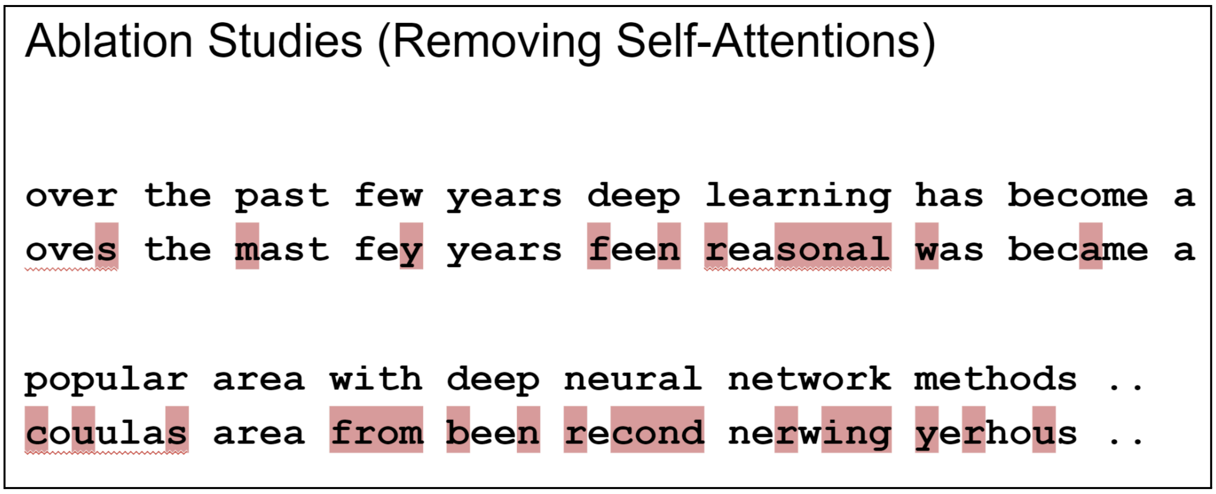


Ablation Studies (Removing CNNs)

```
over the past few years deep learning has become a
nyeo tce dtrt feb ketor meed hetotrtd ctr vemnge t

popular area with deep neural network methods ..
dndmhto toet brtc meed temoth tetbnov getcnmr ..
```

When the Self-Attention module is removed and CNN module is kept, we found that our model performance to be better. This showcases the importance of CNN kernels in extracting and gathering low-level information in the character space and combine them into a meaningful contextualized representations of each single characters. By looking at some model outputs, we found that the model can already output sequences of characters that mimic the appearance of Engish word morphology. However, we found that without the self-attention module, the model will lose the consistency property as it might map the same character in the ciphertext to different characters in the plaintext.



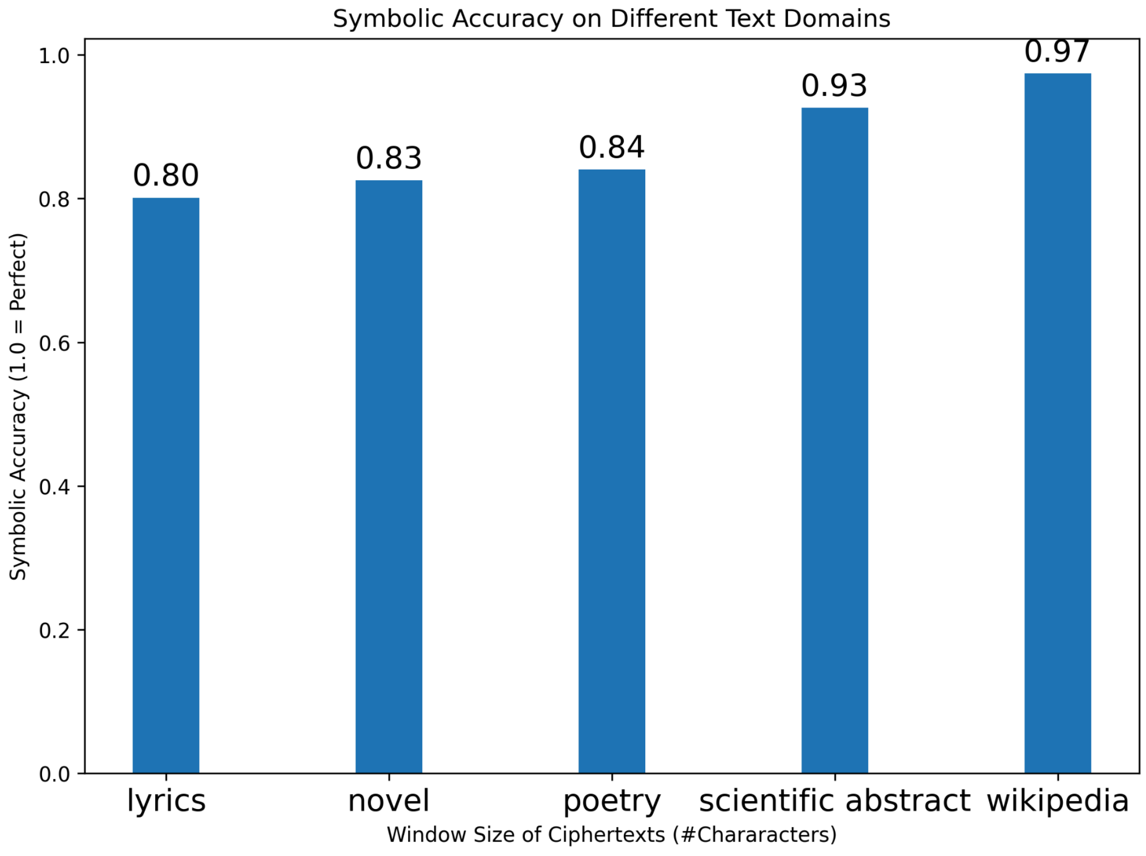
Ablation Studies (Removing Self-Attentions)

over the past few years deep learning has become a
oves the mast fey years feen reasonal was became a

popular area with deep neural network methods ..
couulas area from been recond nerwing yerhous ..

Test on Text of Different Genres

To evaluate our model's functionality, we test our model on samples of text from four different genres. They are song lyrics, novels, poetry, scientific abstract. We test our model on 200 samples for each genre, and then take the average accuracy. Result is shown below.



We can tell from our results that scientific abstract has the best performance among the four genres. The abstract may contain many uncommon and professional words that are unfamiliar for our model, but it should have normal flow of words like the wikipedia text so it should have relatively good accuracy. Song lyrics, the novel, and the poetry don't have a performance as good as the scientific abstract, and the reason could be they don't have a similar style and structure of sentence and words as the wikipedia. The song lyrics and poetry obviously don’t follow the normal way people write texts, and there are usually no clear connections between adjacent sentences. The novel may contain many nouns created by the author and many conversations. Thus, we can see that though our model performs generally well for different genres, there are some differences among their accuracies.

Challenges

It was difficult for us to find the latent relationship between the characters. Simply using an attention encoder is not enough, and we can hardly enhance the model's performance above the baseline and the model translates the ciphertexts by their appearance frequency directly. So, we incorporated a convolutional layer into our model to handle deciphering tasks, following approaches proposed in previous works Character-Level Translation with Self-attention and Can Sequence-to-Sequence Models Crack Substitution Ciphers? We have used different channel sizes to capture dependencies from short to long ranges surrounding each character.

Reflections

Our project ultimately turned out as we expected. We even reached our stretch goals in terms of the accuracy of our final model. Originally, we did not implement the CNN-enhanced transformer encoder blocks and the performance was not as good as expected; but after doing a literature review and playing around with the model architecture, we managed to make it perform better. Our execution of the research plan was in accordance with our original plan - and if we were to do this project again we would have faced the same challenges and come up with the same breakthroughs as we did this time.

If we have more time, we wanted to try out including more characters to the substitution cipher (currently it includes 26 lowercase letters, 10 numbers, and a space character). We also thought about removing the space character or applying substitution ciphers at a subword token level (instead of at the character level).

Our main takeaway is to be open-minded about changing the plan - as we progress gradually through the project, we have to make adjustments and architecture changes in order to improve model performance. If we had stuck with the traditional transformer architecture (and only scaled it with more blocks or with a larger hidden size), we wouldn’t have achieved our final results.