Bachelor Thesis

Transformer Model for Genome Sequence Analysis

Noah Hurmer

Supervisors: Dr. Mina Rezaei, Hüseyin Anil Gündüz, Martin Binder
Date: July 27^{th}, 2022

Declaration of Originality

I confirm that the submitted thesis is original work and was written by me without further assistance. Appropriate credit has been given where reference has been made to the work of others.

Munich, July 27^{th}, 2022

. . . . . .
Noah Hurmer

Abstract

A multitude of problems faced in genomic research are subject to a scarcity of high quality labeled data, and attaining such often requires expensive and time consuming physical experimentation under laboratory conditions. However, through technological advances enabling high throughput sequencing, an abundance of unlabeled data is now readily available. To take advantage of this, the bioinformatics field has sought to apply semi-supervised learning methods through representation learning on genomic tasks, often adapting models originally developed in Natural Language Processing for use on genomic data. One such model is known as the *Transformer Encoder* and we will investigate its ability to produce useful representations training on unlabeled data in order to taxonomically classify read-level nucleotide sequences downstream. To this end, the success of bacteriophage identification is evaluated under self-supervised, semi-supervised as well as strictly supervised training using different label scarcity scenarios. Additionally, alterations to the method of tokenization and the pretext task during self-supervised training are explored for their benefits on the model's accuracy. While conducted experiments imply that this application of the *Transformer* model on genomics data may not yet be perfected to its full maturity, they also show that this architecture provides an increase in classification accuracy of the downstream task compared to existing methods at the cost of more resource intensive training. Moreover, it appears that the model lends itself well to transfer learning, as the gained representations seem to generalize well to another dataset of different origin. The viability of the *Transformer Encoder* approach, especially for generalization to a broader panel of divergent tasks, merits further research.

Contents

1 Introduction **3**
- 1.1 Understanding Genomic Data . . . 3
- 1.2 Thesis Structure and Contributions . . . 4

2 Learning Representations from Genomics Data **5**
- 2.1 Motivation and Related Work . . . 5
- 2.2 Deep Learning in NLP . . . 6
  - 2.2.1 Attention Mechanisms . . . 6
  - 2.2.2 Transformer . . . 7
  - 2.2.3 BERT . . . 8
- 2.3 DNABERT . . . 9
  - 2.3.1 Data Preprocessing . . . 9
  - 2.3.2 Pretraining . . . 9
  - 2.3.3 Finetuning . . . 10

3 BERT Application Experiments for GenomeNet **11**
- 3.1 Virus Phage/Non-Phage Dataset . . . 11
- 3.2 Experiments . . . 12
  - 3.2.1 Scaled Semi-Supervised Trials . . . 12
  - 3.2.2 VIRBERT . . . 15
  - 3.2.3 VIRBERT-mask8 . . . 16
  - 3.2.4 VIRBERT-stride3 . . . 16
  - 3.2.5 DNABERT6 . . . 17
  - 3.2.6 Finetuning . . . 17
- 3.3 Baselines . . . 18

4 Results **19**
- 4.1 Pretraining . . . 19
- 4.2 Linear Evaluation . . . 19
- 4.3 Semi-Supervised Learning . . . 20
- 4.4 Transfer Learning . . . 22

5 Conclusion and Future Direction **23**
- 5.1 Discussion . . . 23
- 5.2 Improvement Ideas . . . 24
  - 5.2.1 Tuning . . . 24
  - 5.2.2 NSP-like Task . . . 24
  - 5.2.3 Preprocessing and Tokenization . . . 25

List of Abbreviations **27**

List of Figures **28**

List of Tables **29**

Bibliography **30**

A Appendix **37**
A.1 Task-Specific Sequence Creation 37
A.1.1 Examples for Evaluation 37
A.1.2 Examples for Finetuning 37
A.2 Definitions 37
A.2.1 AdamW 37
A.2.2 Cross Entropy Loss 38
A.2.3 Metrics 38
A.2.4 Task Level Types 39
A.3 Self-genomenet 39
A.4 HPO 40
A.5 Training Time 41
A.6 Self-Supervised Trials 41

1. Introduction

1.1 Understanding Genomic Data

A genome is defined as the complete set of genetic material of an organism and contains all of the instructions to build and maintain itself. This information is encoded as DNA, a four letter language often referred to as the *language of life*. Each letter represents a nucleotide base - adenine (A), thymine (T), guanine (G), and cytosine (C) - and the specific order of these nucleotides in a sequence defines specific traits or functions [NIH, 2020]. Deciphering this code is far from trivial and what we do know originates from experimentation and computational analysis [Brown, 2002]. With the rise of complex Deep Learning (DL) methods and scientific progress in fast and automated sequencing of genomes, a foundation to substitute some of the slow and expensive experimentation with accurate computational analysis has been laid in order to annotate genomes, identify genes and determine their function.

Since encoding information into a sequence can be seen as much as a language as our human communication tools, genomic DL models often have direct origins in Natural Language Processing (NLP) [Asgari and Mofrad, 2015]. Powerful NLP models employ self-supervised representation learning, a method to gain knowledge from an abundance of unlabled data. This is meant to understand meaning, grammar and structure of a language and therewith assist in solving downstream tasks where the limited amount of task-specific data would inhibit the recognition of such structures. Training on omics-data[1] poses a very similar issue, as obtaining annotated examples of genome sequences requires manual experimentation that can often be very expensive, whereas sequencing genomes via Next Generation Sequencing (NGS) [Buermans and den Dunnen, 2014] and established databases such as *GenBank* [Sayers et al., 2019] offer a multitude of available sequences that can be used for self-supervised training.

The *GenomeNet* project intends to identify and create Deep Learning architectures well suited for analysis of genomic data. Such models are meant to advance the capabilities to identify new genomic structures, impute missing nucleotides and classify genomic data under sparse label conditions [BMBF, 2020].

[1] Omics is the study of multiple biological fields ending in -ome and generally aims to analyze biological molecules [Vailati-Riboni et al., 2017]. Usually, this involves data of sequential nature.

1.2 Thesis Structure and Contributions

In accordance with the goals of *GenomeNet*, the objective of this thesis is to analyze the potential of semi-supervised learning through a Bidirectional Encoder Transformer architecture (*BERT*) on virus genome data. To this end, models based on *DNABERT* [Ji et al., 2021] are trained in a self-supervised manner on a collection of virus genomes. A small scale test framework is created in order to deduce optimal hyperparameter settings and explore different preprocessing, tokenization and pretext task strategies. Apart from the original setup of *DNABERT* [Ji et al., 2021], two differing strategies are pursued and pretrained full scale. Thereafter, the gained representations are then finetuned and evaluated through the downstream classification task of identifying bacteriophages from short (read-level) sequences of nucleotides. This task is performed over various label scarcity scenarios and sequence lengths. A Hyperparameter Optimization (HPO) framework is implemented, with which some finetuning parameter optimization is performed for semi-supervised training. The accuracy over this task is then compared to a baseline model, pretrained on the same data (*self-genomenet* by Gündüz et al. [2022]). Additionally, the transfer learning capability of the *DNABERT* model structure is explored.

2. Learning Representations from Genomics Data

2.1 Motivation and Related Work

Deep Learning has shown improvements over previous methods in a variety of applications in bioinformatics [Zhang et al., 2020]. These include fields such as biomedical image processing [Pereira et al., 2016, Chakravarty and Sivaswamy, 2019], disease prediction and modeling [Guo et al., 2018, Hwang et al., 2017], drug property or interaction prediction [Aliper et al., 2016, Yan et al., 2021] as well as function or specific binding site prediction from omics data [Alipanahi et al., 2015, Zhou and Troyanskaya, 2015]. Many methods proposed to be applied on omics sequences use recurrent or convolutional networks (RNN/CNN) in order to capture patterns and dependencies in the data. Trabelsi et al. [2019] show that combinations of the two, such as a CNN-BiLSTM [Hu et al., 2019, Quang and Xie, 2016], especially with an additional encoding network as used in an architecture they name ECBLSTM, regularly outperform others. Representation learning via adapted language models has been used to improve upon the performance of multiple downstream tasks such as sequence annotation, splice and binding site identification and enhancer prediction [Clauwaert and Waegeman, 2022, Ji et al., 2021, Le et al., 2019]. Iuchi et al. [2021] summarize existing representation learning applications on biological sequences.

Self-supervised models have prevailed in NLP, as they take advantage of readily available amounts of unlabeled data in the form of texts to pretrain model weights and representations in a self-supervised manner. These can then be finetuned with a much smaller task-specific labeled dataset [Devlin et al., 2018, Radford et al., 2018, Peters et al., 2018]. Such approaches aim to learn an understanding of the language and show an improvement over supervised models, especially if annotated training data is very limited [Wang et al., 2020]. The architecture of a Bidirectional Transformer Encoder as introduced by Devlin et al. [2018] is among the self-supervised NLP models that have been found to be useful in representation learning of genomics data and therefore implemented multiple times with differing downstream tasks in mind [Rives et al., 2020, Ji et al., 2021, Le et al., 2021, Mo et al., 2021, Avsec et al., 2021].

Concerning the model evaluation task of identifying bacteriophages, which is conducted during the experiments in this thesis, Ho et al. [2022] provide an overview of conventional approaches to such a task.

2.2 Deep Learning in NLP

Most sequence transduction models prior to the introduction of the *Transformer* architecture were built of either convolutional or recurrent networks wrapped in an encoder-decoder structure [Cho et al., 2014, Sutskever et al., 2014, Kalchbrenner et al., 2016]. A common task for NLP sequence transduction is neural machine translation (NMT). There the encoder embeds the input sequence or its subparts for the decoder to reconstruct a sequence in an autoregressive manner in another language. In order to tackle long range dependency issues associated with such network structures, attention mechanisms were introduced. These mechanisms typically connect encoder and decoder with additional sets of trainable weights, allowing the decoder to be influenced more or less by specific states of the encoder model.

2.2.1 Attention Mechanisms

Attention mechanisms, as introduced by Bahdanau et al. [2014], can be generalized as a way to compute a weighted sum of representation states. This vector of weights - also called a context vector comprised of weighted annotations - helps focus a model onto the more important parts of a given input for a specific output. Moreover, it allows a model to learn further reaching dependencies of an input sequence, tackling the vanishing gradient issue of RNN architectures [Bahdanau et al., 2014, Bengio et al., 1994, Pascanu et al., 2013]. These attention weights are dependent upon a given query (q) and a set of key value pairs (K, V), and computed by a softmax function. Attention is therefore calculated as in the following:

$$Attention(q, K, V)_i = \frac{exp(f_{att}(q, k_i))}{\sum_j exp(f_{att}(q, k_j))} v_i \tag{2.1}$$

or in matrix notation:

$$Attention(Q, K, V) = softmax(f_{att}(Q, K))V \tag{2.2}$$

where f_{att} represents the attention or alignment function. One such attention mechanism is referred to as dot-product attention, where the alignment function is given as [Luong et al., 2015]:

$$f_{att}(Q, K) = K^T Q \tag{2.3}$$

Performing NMT with an encoder-decoder model that incorporates an attention mechanism, the previous decoder state would represent a query, whereas the encoder states provide keys and values.

2.2.2 Transformer

While CNN based models such as *ByteNet* and *ConvS2S* [Kalchbrenner et al., 2016, Gehring et al., 2017] already do away with some of the sequential computations of LSTM or GRU architectures, thus leading to better parallelization and therewith addressing some of the training time demand downsides, Vaswani et al. [2017] argue to completely forgo any convolutional or recurrent layers and to rely solely on attention. The proposed model architecture of the *Transformer* combines only attention and feed-forward layers in both encoder and decoder, allowing for parallel encoding of all tokens in an input sequence. However, the removal of convolution and recurrence leads to the loss of positional information of the input tokens, therefore the token representations are combined with positional encoding vectors. These can either be learned or defined by a combination of sine-cosine functions [Vaswani et al., 2017].

The *Transformer* consists of 6 identical encoder and decoder blocks. Each encoder block includes a multi-head self-attention layer followed by a fully connected feed-forward layer, both possessing a residual connection (see figure 2.1). The decoder differs with the inclusion of another layer which performs attention between the output of the encoder stack and the previous decoder attention layer.

Self-Attention Opposed to other attention mechanisms that map importance of encoder states to an output (decoder) state, in self-attention the queries, keys and values all come from the same input sequence. This allows a model to tend to specific positions of a sequence more when computing the current (query) token representation of that same input sequence. Self-attention therefore results in the identification of dependence structures within a sequence and has been proven useful in language tasks and language modelling [Merity et al., 2016, Paulus et al., 2018]. Self-attention carries the benefit of smaller computational complexity compared to recurrent or convolutional layers [Vaswani et al., 2017]. Additionally, the path length for signals between dependencies within a model is shorter, which improves the ability to learn long range dependencies [Hochreiter et al., 2001]. The transformer uses self-attention in all layers except for the connected attention layer in the decoder. The regular multi-head self-attention layer in the decoder performs masked self-attention, which allows the model to only attend to previous tokens.

Multi-Head Attention In order to learn different dependency structures, the *Transformer* employs multi-head attention. Instead of a single attention function of full dimensionality, queries, keys and values are projected h times into representations of smaller dimensionality of $\frac{d_{model}}{h}$ using different, learned projections [Vaswani et al., 2017]. A scaled dot-product attention function

$$f_{att}(Q, K) = \frac{QK^T}{\sqrt{d_{keys}}} \tag{2.4}$$

is then applied onto each projected triple of queries, keys and values before concatenating the results into a dimension of $h * d_{values}$. Finally, the result is linearly projected once more (figure 2.1). Each multi-head attention layer therefore includes h number of trainable projection matrices for queries, keys and values of the respective dimensions as well as a single trainable matrix of full model dimension.

$$MultiHead(M) = concat(head_1, ..., head_h)W^O \quad (2.5)$$
$$\text{where } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

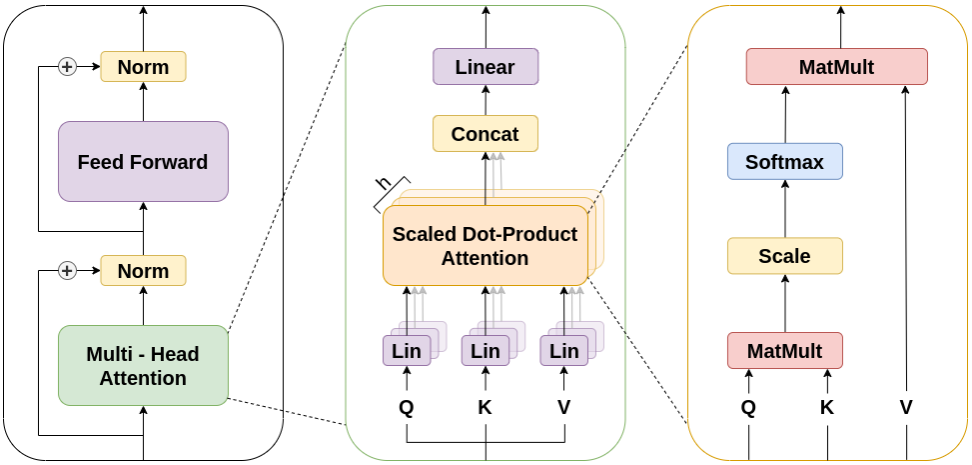


Figure 2.1: **Left:** Pipeline of a single Transformer Encoder block. **Middle:** Inner workings of a Multi-Head Attention Layer of dimensionality h. **Right:** Procedure of Scaled Dot-Product Attention employed during self-attention in the Transformer.

2.2.3 BERT

Similar to other popular self-supervised NLP models such as *GPT* or *ELMo* [Radford et al., 2018, Peters et al., 2018], the Bidirectional Encoder Representations from Transformers architecture [Devlin et al., 2018] is constructed with self-attention layers from the *Transformer*. Specifically, *BERT* - base stacks 12 *Transformer* encoder blocks and raises the number of attention heads from 8 to 12 (each with 64 dimensions) compared to the implementation of Vaswani et al. [2017], which causes an increase in representation size to 768. *BERT* operates in a self-supervised manner which consists of creating representations by pretraining on an unlabled text corpus just like the other models mentioned above. The learned representations are later finetuned when training on task specific data. Critically however, Devlin et al. [2018] claim an improvement over other models due to the bidirectionality of self-attention enabled by its pretext objective of a masked language model, during which input tokens are randomly masked and then predicted by an attached classification

network (mlm-head) (see figure 2.2). This pretext task allows self-attention to be trained on the entire sequence instead of just on preceding tokens, as done in a left to right architecture [Radford et al., 2018]. The tokens defined to be predicted by the model are simply replaced by a designated *MASK* token, a random different token or left unchanged (by a ratio of 8:1:1) when training the entire model with cross-entropy loss (see Appendix A.2.2). To account for inter-sentence relationship importance of certain downstream tasks such as Question Answering, *BERT* also includes a second pretext task of Next Sentence Prediction (NSP). Here, the model is to predict whether or not two input sentences follow each other in the source from which they were extracted. To realize this, two sentences are input separated by an *SEP* token and a classification model head attached to the *CLS* token is trained on this task.

2.3 DNABERT

As opposed to Mo et al. [2021] or Le et al. [2021], who combine a *BERT*-based architecture with other model types, Ji et al. [2021] more directly adapt *BERT* for application on genomics data with their implementation of *DNABERT*. It is pretrained on sequences extracted from the human genome in order to learn regulatory code in gene expression. They report improvements in multiple downstream tasks as well as display the explainability benefits of a self-attention based model.

2.3.1 Data Preprocessing

In order to input genome data into a *BERT* architecture, it has to assume a language-like structure of sequences made up of separated tokens. In order to account for genome sequences that regularly surpass the input length capability of a *BERT* model (512 tokens), subsequences have to be created. Ji et al. [2021] attain these in two different ways. They split genomes into non-overlapping subsequences of sampled length and also randomly sample subsequences of sampled length from a genome. These are subsequently tokenized using all permutations of k-mer representation, which creates tokens like a sliding window with a stride of 1 (see figure 3.2).

2.3.2 Pretraining

Similarly to other *BERT* variations or experiments such as *roBERTa* and *DistilBERT* [Liu et al., 2019, Sanh et al., 2019], *DNABERT* does away with the pretext task of NSP. Representations are therefore created solely with help of the masked language model. Not only do above mentioned experiments show at least similar performance without NSP, removing it allows for full length input sequences of 512 tokens and eliminates the requirement of structural knowledge about the sequences input into the model during self-supervised training.

Because tokens are created using k-mer representation of stride 1, a k-mer is identical to its preceding and succeeding in all but one character respectively. A token could therefore be correctly inferred by its neighbors every time. To prevent this, instead of randomly sampling a percentage m of tokens to mask, Ji et al. [2021] opt to mask k consecutive tokens per sampled masking location. In order to keep the total ratio of masked tokens the same, this sampled masking location percentage is dropped to $\frac{m}{k}$. k refers to the length of the k-mer.

2.3.3 Finetuning

Using different datasets, *DNABERT* is compared to previous state of the art performance on the tasks of prediction of promoter regions, finding of splice and transcription factor binding sites. Ji et al. [2021] claim improvements over existing methods such as *DeePromoter*, *DeepSEA* and *SpliceFinder* [Oubounyt et al., 2019, Zhou and Troyanskaya, 2015, Wang et al., 2019]. Additionally, transfer learning to other than the organism used in pretraining is conveyed through tasks on ENCODE ChIP-seq datasets of Mouse DNA [Stamatoyannopoulos et al., 2012].

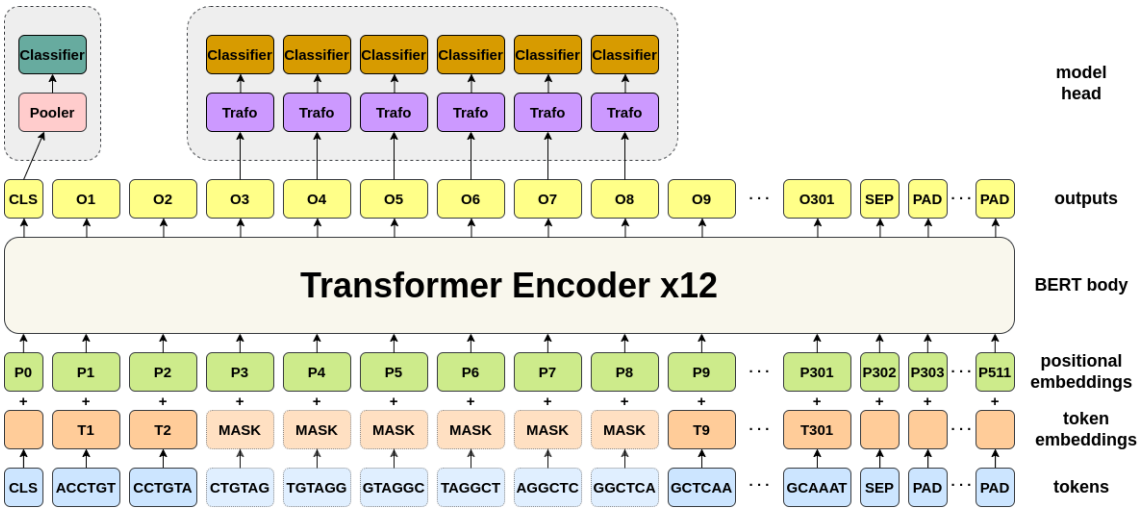


Figure 2.2: Visualization of the *(DNA)BERT* pipeline for an input example of 301 tokens. The right model head represents the mlm training head attached during pretraining, while the left model head is used for sequence classification and therefore present during finetuning. *Trafo* represents a linear layer with GELU activation and layer normalization, whereas *Pooler* here refers to a linear layer with tanh activation.

3. BERT Application Experiments for GenomeNet

The aim of the following experiments is to compare the performance of the *BERT* architecture on genome level tasks (see Appendix A.2.4) to other representation learning models. To this end, three different adaptations of the *BERT*-based model *DNABERT*, as introduced by Ji et al. [2021], are proposed. These and the baseline model are pretrained in a self-supervised manner on a virus genome dataset. Models are evaluated after training them in a supervised manner with different nucleotide length and label scarcity scenarios on the task of identifying whether or not a nucleotide sequence is an excerpt of a bacteriophage genome. In this section the training procedures of the different *BERT* models as well as the baselines that are compared to in chapter 4 will be outlined.

3.1 Virus Phage/Non-Phage Dataset

All self-supervised learning models are trained and evaluated on a compilation of virus genome sequences. On August 2nd, 2021, all available virus genome data on *GenBank* [Sayers et al., 2019] was downloaded and divided into two taxonomic classes, bacteriophages and other viruses. The resulting data was split into train, validation and test sets by approximate ratios of 70%, 20% and 10%. This collection of approximately 40k FASTA-files includes about 1 billion nucleotides for the bacteriophage class and 0.5 billion for other viruses. All of the following self-supervised models are pretrained on nucleotide sequences created from unlabeled data of the train split. Splitting and class-annotation was done by Gündüz et al. [2022], resulting data was simply used here.

Evaluation

This dataset poses a binary classification task of nucleotide sequences between viruses that are bacteriophages and viruses that aren't. Bacteriophages are viruses that only attack bacterial cells [Kasman and Porter, 2021]. They are very abundant in nature, often target species specific and quite diverse in their genomic organization [Simmonds and Aiewsakun, 2018]. In metagenomic sequence applications, taxonomic classification is often useful when applying NGS methods to analyze DNA collected from a given environment [Pust and Tümmler, 2021]. Identifying bacteriophages can be especially difficult, as they lack universal marker genes [Roux et al., 2020]. In comparison to other classification tasks that exhibit themselves on a range of nucleotides such as identifying transcription factor binding - or splice sites [Ji et al., 2021], these classes are genome wide. The goal is therefore to infer

genome-level information from a short sequence.

Because NGS methods regularly return sequences of shorter length, 150- and 1000-nucleotide sequences are randomly sampled without overlap from the test split to be used evaluating this downstream task (see table 3.1). Macro-averaged recall $Recall_M$ (in %) and F_1-score are then used to measure model performance (see Appendix A.2.3).

seq length	150nt	1000nt
non-phage	102027	35306
phage	86381	61101
total	188408	96407
ratio	1.18	0.58

Table 3.1: Number of examples to predict in the evaluation set for the phage/non-phage task of each input length along with the class distribution of those examples. See Appendix A.1.

3.2 Experiments

First, the setup of the aforementioned scaled test framework is outlined and its results discussed. Based thereon, the three different setups used to pretrain a *BERT*-based model are described. Since Ji et al. [2021] report a k of 6 to perform the best in their experiments, all setups share the use of 6-mers. Additionally, training was halted after 100k steps every time for comparability and resource reasons (see section 4.1).

3.2.1 Scaled Semi-Supervised Trials

Since it is not given that the pretraining parameters used by Ji et al. [2021] work optimally with data and downstream tasks different to theirs, it was deemed necessary to investigate the impact of some hyperparameters to the model's performance. These include learning rate, masking percentage, weight decay and warmup percentage. Moreover, it was desired to benchmark some alterations to the process regarding data preprocessing, k-mer creation and masking. Therefore, a scaled down testing framework was created, which would allow for approximate inference of implemented changes without the need to train full models.

Setup This framework is realized by pretraining the *bert-small* configuration on *huggingface*, a scaled down version of *BERT*, which consists of only 4 stacked *Transformer* encoder blocks, only 8-dimensional multi-head attention layers as well as a smaller representation dimensionality of 512 [Wolf et al., 2020, Bhargava et al., 2021].

While the batch size is kept at what the full models would have, a down-sampled amount of input sequences are trained on, and the maximum number of steps is set to 10k. Throughout all trials masking is the only pretext task used. Similarly, hyperparameters - apart from weight decay - of the optimizer *AdamW* (see Appendix A.2.1) are not changed from the original *DNABERT* model implementation.

As pretraining loss only describes the model's ability to predict masked tokens, it is not inherently a metric indicating the model's proficiency of any specific downstream task. Therefore, all pretrained trial models have a linear classifier attached, and only this layer is trained in a supervised manner on the 150nt long phage/non-phage task with the same fixed parameters for all models. The accuracy achieved by each trial on the validation data was mainly used to judge model performance. Detailed information about the trial setup is available in the Appendix A.6.

mask %	stride	#nt/loc	%nt hidden	% for k=6
15	1	1	$\frac{15}{k}$	2.5
15	1	3	$\frac{45}{k+2}$	5.625
15	1	5	$\frac{75}{k+4}$	7.5
15	1	6	$\frac{90}{k+5}$	8.1818
15	$\frac{k}{2}$	$\frac{k}{2}$	15	15
15	k	k	15	15

Table 3.2: Comparison of different masking and tokenization strategies experimented with during scaled self-supervised trials with respect to their implication of nucleotides hidden from the model during mlm training. Shown here are the hidden nucleotides per masked location (#nt/loc), the percentage of nucleotides hidden overall for any k (%nt hidden) and for a k of 6. The first row corresponds to *virBERT*, the second to *virBERT-mask8* and the fifth to *virBERT-stride3* introduced in section 3.2.

Trials The main concern facilitating the desire to test different setups, is the masking technique of Ji et al. [2021]. As stated in 2.3, because k consecutive tokens are masked $\frac{15}{k}$ % of the time, the amount of distinct nucleotides hidden from the model is also only $\frac{15}{k}$ %. On the k-mer representation level, one can in theory infer $k-1$ characters of all of the masked tokens via its preceding and succeeding tokens, leaving the prediction of the one hidden nucleotide per masking location down to a choice of one out of the four existing bases (compare table 3.2). It is however not at all guaranteed that a model learns this relationship, as the k-mers are converted to representing vector tokens before they are fed into the model. Apart from simply increasing the masking percentage, trial setups that mask more consecutive tokens per location (mask k+x -setups) as well as setups that create k-mers with higher

strides of $\frac{k}{2}$ and k are experimented with in order to counteract this effect. On the data preprocessing side, higher input sequence length lower bounds (and lower sequence length upper bounds for higher stride trials) as well as a different input sequence length distribution are examined. These changes are based on the notions that input sequences of lengths less than 3 tokens are not advantageous and that more differentiated input lengths may result in better generalization to tasks of different input lengths. Additionally, input sequences are created with different ratios between non-overlap cut and sampled subsequences. A description of how this is usually performed in *DNABERT* is available in section 3.2.2. Overall, 36 different setups are trialed (see Appendix A.6).

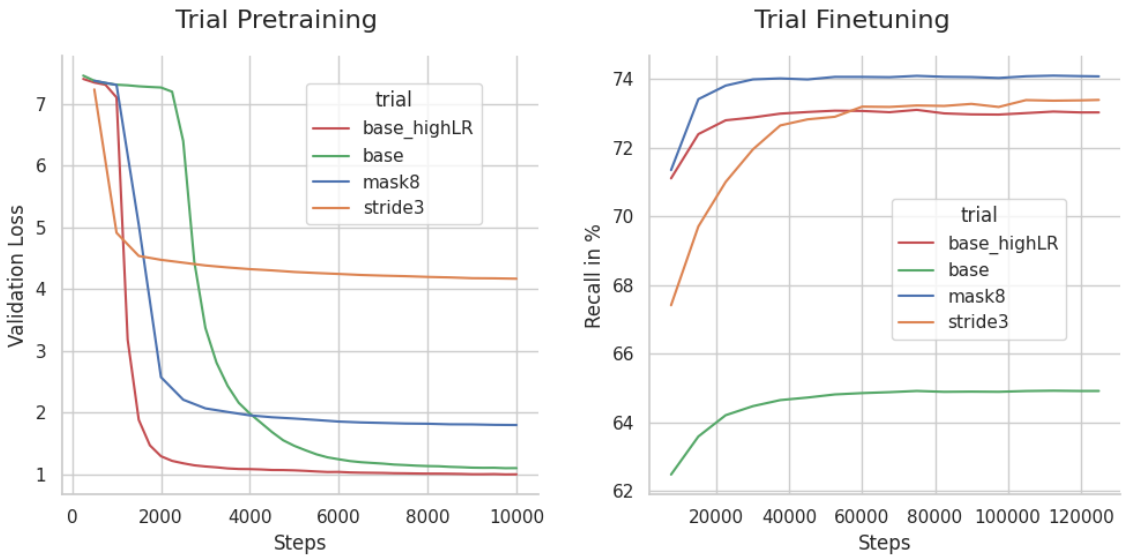


Figure 3.1: Pretraining and Finetuning for some selected trials of the setup described in section 3.2.1. *base* (green) here represents the scaled down version of *virBERT* with all the same training parameters, while *mask8* (blue) and *stride3* (orange) do so for the other two variants pursued full scale. *base_highLR* (red) poses as a baseline to the higher learning rate setups of *stride3* and *mask8* with an equal learning rate of $1e^{-3}$ and is equal to *base* otherwise. **Left:** Cross-entropy loss of mlm on a validation set during self-supervised pretraining. **Right:** Class averaged recall during supervised finetuning with frozen representation model layers on the 150nt virus phage/non-phage classification task.

Results It is important to mention that these trial runs do not represent a systematic or exhaustive form of hyperparameter optimization. They merely serve as indications for possible benefits that changes to the pretraining setup may have. Because alterations in data preprocessing force dataset creation anew and since the sets are sampled down to 10%, the runs are not necessarily trained on the exact same examples. Additionally, increasing model performance on a single downstream task

may come to the detriment of other tasks. That being said, increasing learning rate up to $1e^{-3}$ as well as warm-up percentage to 10% seems beneficial throughout all setup variations. The base masking rate of 15% as well as weight decay of 0.01 generally perform best. Increasing masking percentage during training showed a decrease in model accuracy on the evaluation task. While no masking setup is consistently better throughout the range of learning rates applied, it can be seen that mask+2 setups are on par or outperform standard setups of equal learning rate and produce the best performing model of all stride 1 setups. Higher k-mer stride setups typically do not perform as well as the standard setups for higher learning rates, they do however improve on this task when input sequences are kept shorter (lower sequence length upper bound). Even more so, when the model's input capability is hard limited to accept less tokens[1]. Generally, these setups exhibit a higher accuracy delta during finetuning.

3.2.2 VIRBERT

The first *DNABERT*-based model, henceforth referred to as *virBERT*, is a recreation of the training procedure used by Ji et al. [2021], complete with the suggested self-supervised training parameters and data preprocessing. Instead of training on sequences of the human genome, they are replaced by ones created from the train split of the above-mentioned virus dataset.

In accordance with the approach of Ji et al. [2021], training subsequences are compiled in two ways. Firstly, all genome sequences are split without overlap. The lengths of these splits are sampled between 5 and 510 nucleotides with a 50 percent bias towards the full length of 510. Secondly, subsequences are randomly cut from each full sequence. The length is likewise sampled as stated above. A ratio of 2 between the number of sampled to cut subsequences is set, thus there are twice as many subsequences sampled than non-overlap splitting creates for each genome.

This approach yields ∼7.7M subsequences for all available virus genome data in the train split. It was chosen to not balance pretraining data by class or length per sequence, as per definition of the applied use of a self-supervised model, this information is not necessarily available. Additionally, different contigs of the same genome were each simply treated as a distinct sequence[2]. The created subsequences are then transformed into 6-mers and shuffled.

All training hyperparameters were kept exactly the same as in the original implementation of Ji et al. [2021]. Therefore, the model was set to train over 200k steps with a batch size of 2000 (achieved with gradient step accumulation). A masking

[1]Inputting sequences of shorter token length than the maximum the model is configured for, leads to appended padding tokens. Limiting the model to an input sequence token length closer to that of the actual input sequences, consequentially removes unnecessary tokens and attention/network connections.

[2]The FASTA-files downloaded and used here for training sometimes contain multiple unconnected sections (contigs) of an organism's genome, which have not been fully sequenced, connected or arranged.

probability of 15% is selected for the first 100k steps, increasing to 20% thereafter. *AdamW* (see Appendix A.2.1) is used as the optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 1e^{-6}$. Model weight decay is set to 0.01. A learning rate of $4e^{-4}$ is linearly increased over the first 10k steps (warm-up of 5% of total steps) and then linearly decreased back to 0.

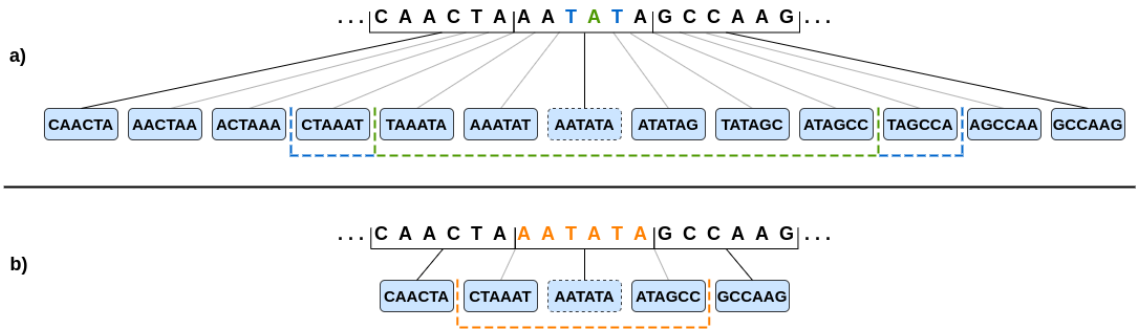


Figure 3.2: This figure shows how input tokens are created as k-mers from an excerpt of a nucleotide sequence. The token with a dashed border represents a sampled masking location for the mlm during pretraining. **a)** Tokenization for 6-mer creation using all permutations (stride of 1), therefore representing the method of *DNABERT6*. A sequence of 18 nucleotides creates 13 tokens this way. The dashed green box shows tokens masked for the defined masking location for the base *virBERT* setup. The blue box represents the mask range extension performed for the *virBERT-mask8* setup. Distinct nucleotides hidden from the model during this masking operation are displayed in a color coordinated fashion in the sequence above. **b)** 6-mer tokenization for the *virBERT-stride3* setup, creating only 5 tokens from 18 nucleotides. Similarly as in *a)*, the orange box shows which tokens are masked and the hidden distinct nucleotides are highlighted.

3.2.3 VIRBERT-mask8

Following the scaled pretraining trials, it was selected to train a full scale mask+2 *DNABERT* variant. This model masks 8 consecutive tokens, and a total of 15% of tokens overall. This rate is not increased during training. The learning rate is increased to $1e^{-3}$ and the number of warm-up steps to 20k. It is again set to train for 200k steps and all parameters not mentioned remain the same as for *virBERT*. Compared to that model, training data is created with a higher nucleotide length lower bound of 36 instead of 5.

3.2.4 VIRBERT-stride3

As a third model, it was chosen to use a k-mer tokenization with a stride of 3 (see figure 3.2). Due to the fact that this approach results in a model that can handle longer input sequences in terms of distinct nucleotides and, inspired by the positive

effect limiting the amount of input tokens to the model had in the self-supervised trials (see section 3.2.1), it was chosen to not only train with sequences of maximum length of 1000nt (332 6-mers with stride 3) but also hard-limit the model to 340 input tokens. This longer nucleotide length input allows for prediction of both task defined sequence lengths without the need to concatenate representations, as is necessary in order to predict 1000nt sequences in all stride 1 k-mer models. Genome subsequences are created in much the same way as in *virBERT-mask8*, the only difference being the higher nucleotide length upper bound of 1000. These are tokenized as 6-mers with a stride of 3. This variant is also trained with a learning rate of $1e^{-3}$ and 20k warm-up steps. 3 consecutive tokens are masked in all 5% sampled masking locations, leading to a total masked token percentage of 15% and 6 hidden nucleotides per location. This setup carries the benefit of deceased training time and resources compared to the other *BERT*-based methods (see Appendix A.5).

3.2.5 DNABERT6

Since Ji et al. [2021] made versions of *DNABERT* pretrained on the human genome available, it is possible to investigate the transfer learning capabilities of this architecture on the same virus task without the need to retrain any model on different pretraining data. While their experiments already show some transfer learning through region level tasks on a different mammalian species, the mouse genome can be confidently aligned on 40% of the human genome [Consortium, 2002], the same cannot be said for the virus genome. *DNABERT6* will be finetuned and evaluated on the same 150nt and 1000nt long phage/non-phage classification task.

3.2.6 Finetuning

To finetune a *BERT* model for a sequence level classification task (as done here with the phage/non-phage task), the pretext task network heads and all but the first token representations are scraped. This starting token (*CLS*-token) is meant to collect sequence level information and only the *CLS* token is then fed through an additional projection layer to a classifier in order to predict the class of an input sequence (see figure 2.2). All three *virBERT* models are finetuned with randomly cut, class annotated subsequences of 150 and 1000 nucleotide length sequences over three different label availability scenarios. The input sequences are balanced by class and to a degree also by genome length. Further details are described in Appendix A.1. Because sequences of length 1000nt, tokenized as k-mers with a stride of 1, surpass the input token limitation of *BERT*-base, these sequences are split into two, passed through individually, and then the sequence representations are concatenated again[3]. This is done for all *DNABERT*-based models on the 1000nt task except for *virBERT-stride3*, which can handle inputs up to 1024nt.

[3]This is achieved with a fully connected layer and a ReLU activation function.

3.3 Baselines

Performance of the self-supervised *virBERT*-models will be compared to a baseline constructed by *self-genomenet*, a self-supervised model proposed by Gündüz et al. [2022], which is trained on the same virus data. The model is comprised of stacked encoder (CNN) and context (LSTM) networks and learns representations through the prediction of reverse complement nucleotide sequences. A more detailed description is available in the Appendix A.3. The reported results below are taken directly from those experiments and are not recreated. As mentioned in section 3.1 however, the training data is identical and the test examples are created equally. Comparisons to other self-supervised models, such as *CPC* [Oord et al., 2018] and different language models, on this task is also available through the experiments of Gündüz et al. [2022]. It is important to mention here, that the *BERT*-base model (and therefore all *DNABERT* variations) has a representation size of 768 compared to the baseline at 512. This not only creates a larger fully connected layer for linear evaluation (see section 4.2), but is generally expected to contribute to better model performance.

Additionally, performance of fully supervised *virBERT* training is provided. This *supervised-virBERT* model is not pretrained, therefore only randomly initialized before it is finetuned on the task data.

4. Results

4.1 Pretraining

All *virBERT* pretraining runs exhibit similar behaviour as reported by Ji et al. [2021] for *DNABERT6*. There is a sharp validation loss decrease within the first 20k steps followed by a much more gradual decrease thereafter (see figure 4.1). Validation loss here refers to the mlm cross-entropy loss on a validation set, which is simply a smaller set of sequences created equally as the training set. For *DNABERT6*, training was halted after 120k steps, citing loss plateauing as the reason. While figure 4.1 shows an indication of continual loss decrease even at 100k steps, *virBERT* models were only pretrained to 100k steps due to resource constraints. As a result, masking percentage increase in *virBERT (base)* was not facilitated, does however also not show an increase in model performance in the scaled self-supervised trials (see section 3.2.1). Figure 4.1 also indicates higher validation loss convergence points for the *mask8* and *stride3* variants of *virBERT*, which is expected as their mlm setup hides more nucleotides overall (compare table 3.2 and section 3.2.1). Pretraining was conducted for 190h on 8 nvidia-A100-40Gb GPUs for *virBERT* & *virBERT-mask8* and 141h on 5 of the same GPUs for *virBERT-stride3*, which shows that *DNABERT*-based models are much more resource intensive than any of the baselines (see Apendix A.5).

4.2 Linear Evaluation

	$Recall_M$	F_1
self-genomenet	88.6	0.916
virBERT	97.8	0.986
virBERT-mask8	93.8	0.963
virBERT-stride3	92.6	0.956

Table 4.1: Model performance on the phage/non-phage classification task for 1000nt long input sequences under linear evaluation of pretrained self-supervised models reported in class averaged *recall* (in %) and F_1-score.

As is common for evaluating self-supervised models, the representations gained while pretraining are applied in a linear classification setting, during which all layers of the representation networks are frozen and only a linear classifier layer is trained. Since the variants *virBERT* and *virBERT-mask8* operate in the split and concatenate function for 1000nt inputs however, there is a linear layer with ReLU activation on

top of the two gained *CLS* tokens of a sequence before the linear classifier in order to combine the two output tokens to a single representation. Since this layer is not present while pretraining, it is also not frozen here. Input sequences are created with 100% label availability for this purpose. Table 4.1 compares the *virBERT*-variants performance to the baseline under such linear evaluation.

Figure 4.1: **Left:** Cross-entropy loss of mlm on the validation set during self-supervised pretraining of the 3 different *virBERT* model setups. **Right:** Class averaged *recall* during supervised finetuning of the 3 *virBERT* models with frozen representation model layers on the 1000nt virus phage/non-phage classification task for linear evaluation.

4.3 Semi-Supervised Learning

Here, all models are finetuned on the phage/non-phage classification task mentioned in section 3.1. Three different label availability scenarios are artificially created by limiting access to a specific subset of FASTA-files during training. These splits result in exactly the same files for all models, so the available labels are also the same throughout. As in the semi-supervised protocol of Henaff [2020], 1% and 10% labeled data is used. Additionally, a very sparse label setting of 0.1% is trained on. For all *virBERT* models, input sequences are created in a way as to maximize the amount of samples (up to a limit) without over-representation of longer genomes and repetition of shorter ones (see Appendix A.1). Models are always finetuned with the same length sequences as they are evaluated. Tables 4.2 and 4.3 compare model performance for 150 and 1000 nucleotide length sequences, respectively. Hyperparameters for supervised finetuning of *virBERT* models are decided on with the aid of small HPO studies (see Appendix A.4).

	10%		1%		0.1%	
	$Recall_M$	F_1	$Recall_M$	F_1	$Recall_M$	F_1
self-genomenet	78.2	0.785	75.3	0.751	67.2	0.700
supervised-virBERT	71.8	0.710	67.6	0.673	62.4	0.608
virBERT	**85.7**	**0.851**	**82.2**	**0.821**	**77.8**	**0.780**
virBERT-mask8	85.0	0.845	81.7	0.812	76.7	0.762
virBERT-stride3	80.8	0.801	75.1	0.757	65.6	0.654

Table 4.2: Performance results through semi-supervised training on sequences of 150 nucleotide length. Displayed are class-averaged accuracy $Recall_M$ and binary F_1-score. The percentages represent the three label availability scenarios during finetuning on the phage/non-phage virus task.

Throughout all 6 scenarios, *virBERT*-based models show superior performance compared to the baseline with the base *virBERT* implementation appearing the best on average overall. Table 4.2 shows that while the *mask8* variant is very similar in performance to the base *virBERT* model, the *stride3* variant produces less successful class predictions for 150nt input sequences. In table 4.3 it is however evident, that the *stride3* variant exhibits similar or better accuracy than the other variants with 1000nt input sequences. Since the *stride3* model variant has a shorter input length it trains notably quicker than the other *virBERT* models (*stride3* trains for 37% and 63% of the time spent by the *base* tokenization versions for 150 and 1000nt, respectively). The pretrained *virBERT*-model manifests an about 20% increase in recall over the strictly supervised baseline throughout all scenarios, generally increasing in the label scarcer setups of 1% and 0.1%. The *virBERT* model also shows an impressive accuracy on the low label scenario of 0.1%, surpassing *self-genomenet* by about 16% and 20% in recall for 150 and 1000nt, respectively.

	10%		1%		0.1%	
	$Recall_M$	F_1	$Recall_M$	F_1	$Recall_M$	F_1
self-genomenet	94.0	-	85.9	-	73.1	0.846
supervised-virBERT	81.5	0.871	77.2	0.867	70.6	0.773
virBERT	97.9	0.986	**94.4**	**0.968**	**87.8**	**0.930**
virBERT-mask8	97.2	0.983	91.7	0.953	81.7	0.901
virBERT-stride3	**98.1**	**0.988**	90.9	0.949	87.2	0.927

Table 4.3: Comparison of semi-supervised training performance with a longer input sequence length of 1000 nucleotides. As in table 4.2, performance is reported in $Recall_M$ and F_1-score over the three different label availability scenarios.

4.4 Transfer Learning

This section is to investigate the capability of a self-supervised model to be used on data of different origin to the one it was pretrained on. In this case, models are trained on data from organisms of different biological classification in a self supervised manner before finetuning them on the phage/non-phage classification task. Ji et al. [2021] provide a pretrained *DNABERT6* model, which is identical to *virBERT* except for the applied training data. They pretrain on sequences extracted from the human genome. Gündüz et al. [2022] already report transfer learning performance of the *self-genomenet* model for this virus task. There, the model is pretrained on all available bacteria genomes from *GenBank* [Sayers et al., 2019]. Since the two models are not pretrained on the same data, table 4.4 does not necessarily allow for direct comparison, it merely indicates each model's ability to effectively transfer gained representations through self-supervised training onto tasks of different data origin.

		150nt		1000nt	
model	pt-data	$Recall_M$	F_1	$Recall_M$	F_1
DNABERT6	human	79.2	0.799	96.6	0.978
self-genomenet	bacteria	-	-	97.0	-

Table 4.4: Transfer learning performance of the Transformer architecture represented by *DNABERT6*, a *virBERT* eqivalent, pretrained on human genome sequences by Ji et al. [2021] and the baseline pretrained on a collection of bacteria genomes.

5. Conclusion and Future Direction

5.1 Discussion

In this thesis we have explored the effectiveness of representation learning through a Bidirectional Transformer Encoder architecture for genome data. To this end, different variations of an adapted *BERT* model are pretrained in a self-supervised manner on nucleotide sequences extracted from virus genomes. These models are then finetuned on a binary taxonomic classification task under various label scarcity scenarios.

In general, it can be concluded that the pretrained Transformer Encoder proves to be a useful tool in analyzing genome sequences on a genome level task. We have shown that the DNA input adaptation of *BERT*, implemented by Ji et al. [2021] as *DNABERT* for use with human genome sequences, is also capable in learning representations from virus genome sequences. Our virus pretrained version, referred to as *virBERT*, outperforms the given baseline on all input length and label availability setups on the task of identifying bacteriophages from read-level length genome sequences. To reiterate however, a direct comparison between the model architectures' success is not exactly possible, since they differ in representation size by 33%. The *virBERT* realization following the original setup of *DNABERT6* also outperforms both of the permutations (*mask8* and *stride3*) trialed on this task on average. Therefore it seems that increasing the amount of nucleotides hidden from the model during the self-supervised mlm training with these masking or tokenization alterations does not increase model performance, partially contradicting learnings from the scaled trials (see section 3.2.1). While the *stride3* variant was theorized to perform better on 1000nt input sequences because it can handle such lengths without splitting the input sequences, there is no directly observable evidence to support this. It does however provide the same level of accuracy with a much lower resource demand in both pretraining and finetuning than the base *virBERT* model on this input length. Generally, it has to be acknowledged that the Transformer Encoder model is very resource and training time intensive, even when compared to other self-supervised models for genome sequence analysis (see table A.5).

An interesting observation of the conducted experiments is that all models trained in these experiments over-predict the bacteriophage class throughout every setup. Possibly the model learns to classify more noisy input sequences as phages, as these might be more diverse in short genome excerpts. However, this may indicate that the performance measures on the 1000nt task are skewed, as the distribution of prediction data might be in favor of such a behavior (see table 3.1).

For a more definitive evaluation of this model architecture, it will be necessary to investigate its performance on more differentiated downstream tasks. This could

include tasks on virus data such as performed by Wu et al. [2021], who attempt to classify bacteriophages as virulent or temperate, predicting a phages host as done by Boeckaerts et al. [2021] or determining whether or not a virus can infect humans [Bartoszewicz et al., 2021]. It should however, definitely also extend to tasks on other organisms and tasks of strictly sequence or token range level (see Appendix A.2.4) such as the *T6SS*-dataset provides [Li et al., 2015]. Training the Transformer Encoder architecture is rather expensive and real-world applications such as classifying genomic sequences extracted from an environment can include DNA from multiple different organisms. Hence, it may be of interest to examine the performance of a model that has been pretrained on a collection of data from different biological kingdoms/domains/classes.

5.2 Improvement Ideas

5.2.1 Tuning

While the *virBERT* models represent a considerable increase in performance on the low label availability scenarios of the applied virus classification task compared to the baseline, finetuning the pretrained self-supervised models exhibits a tendency to overfit very quickly. Altering the hyperparameters weight decay, dropout rate and learning rate does not seem to have a major impact on this behavior and/or lead to a less accurate model. More exhaustive HPO on these label scarce scenarios may lead to better training behavior and model performance. Because decreasing model complexity typically also combats this, an argument may be made to explore a smaller *BERT* size in this context. The *bert-medium* setup on *huggingface* may be suitable and also provides for a more fair comparison to baselines such as *self-genomenet* with the same representation size of 512 [Bhargava et al., 2021].

5.2.2 NSP-like Task

Although some conducted experiments on *BERT* show no performance gain from NSP for downstream language tasks [Liu et al., 2019], this additional pretext task was originally designed to capture sequence level understanding and specifically trains the *CLS* token during pretraining. As mentioned in section 3.2.6, all but this *CLS* token of the output of a *BERT* model is discarded during sequence classification with the standard implemented model head. With this in mind, it seems possible that reinstating some sort of sequence level pretext task might raise the model's performance on sequence or genome-wide downstream tasks such as the bacteriophage identification used in the above experiments. This would not necessarily have to manifest itself in prediction of whether or not two sequences directly follow each other [Mo et al., 2021], but could take on the form of predicting if two sequences originate from the same genome or similar. Implementing this might however exclude some available data from pretraining as it requires some existing form of annotation of the sequences.

5.2.3 Preprocessing and Tokenization

Stride3 Improvements Since the *virBERT-stride3* variant performed on par with the standard model for 1000nt inputs, and because hiding more consecutive tokens during pretraining did not show an increase in performance on the downstream task, it may be of interest to train a *stride3* model while masking less tokens in series during self-supervised training. The scaled trials also revealed an increase in 150nt classification accuracy for a *stride3* model setting when pretraining sequence length distribution is more diverse. Incorporating some of these changes may result in a model that performs best on this task over all input length scenarios whilst consuming less time to train.

Input Length Limitation An obvious downside of the use of a *BERT*-based architecture and a k-mer tokenization method is the length limitation of input sequences. Depending on the stride of the k-mer creation, *DNABERT* models and its derivatives are limited to specific sequence lengths, especially during pretraining. This might inhibit the discovery of very long range dependency structures within genomic sequences and thereby render the model useless or less performant on long input sequence downstream tasks. Another recently implemented *BERT* use for a similar task attempts to tackle this issue with the inclusion of an upstream encoding network [Shang et al., 2022].

Model-based Genome Sequence Segmentation Opposed to the typical word-part tokens used in NLP, the created k-mers in *DNABERT* are not only overlapping but also random. As the input sequences and its lengths are sampled, the k-mers and therefore also the embeddings input into the model change just by sampling a sequence one nucleotide later. Assuming the removal of this randomness improves the model's understanding of the input *language*, it may be beneficial to prefix a *language creation* model. Liang [2012], for example, attempts to segment DNA into words using a statistical model. Another approach could be a *minmax* algorithm that creates as few distinct but as long as possible segments. Because this would create segments of varying length, it probably also considerably increases vocabulary size. Whether or not this is beneficial is however unclear.

Acknowledgements and Reproducibility

This reasearch has been made possible by the contribution of Yanrong Ji, Zhihan Zhou, Han Liua and Ramana V Davuluri with their publication of the *DNABERT* model code.

Sincere thanks goes to the supervisors of this thesis (namely Dr. Mina Rezaei, Hüseyin Anil Gündüz and Martin Binder) for support and guidance throughout the project as well as to the *GenomeNet* team for its cooperation, especially towards René Mreches and Philipp Münch of the HZI for helpful discussions.

This work has been funded in part by the German Federal Ministry of Education and Research (BMBF) under Grant No.01IS18036A, Munich Center for Machine Learning (MCML). Support in computational resources have also been provided under Prof. Dr. Bernd Bischl of the Statistical Learning and Data Science chair at LMU Munich.

All code required to reproduce the experiments conducted can be found on github at *https://github.com/minimops/DNABERT*. A link to downloadable pretrained models is also made available there.

List of Abbreviations

Expression	Representing
BERT	Bidirectional Encoder Representations from Transformers
BiLSTM	Bidirectional LSTM model
CNN	Convolutional Neural Network
DL	Deep Learning
DNABERT	BERT model adapted for DNA as input
ECBLSTM	Embedding-Convolution-BiLSTM
GELU	Gaussian Error Linear Units
GPU	Graphics processing unit
GRU	Gated-Recurrent-Units
HPO	Hyperparameter Optimization
LSTM	Long-Short-Term-Memory
mlm	Masked Language Model
NGS	Next-Generation Sequencing
NMT	Neural Machine Translation
NN	Neural Network
nt	Nucleotide(s)
Phage/Non-Phage	Bateriophage/other Virus
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Networt
tanh	Tangent Hyperbolic Function

List of Figures

2.1 Inner workings of a Transformer Encoder block 8
2.2 Visualization of the *(DNA)BERT* pipeline 10

3.1 Pretraining and Finetuning of scale trial runs 14
3.2 k-mer tokenization for *virBERT* 16

4.1 Pretraining and Finetuning of *virBERT* models 20

A.1 Hyperparameter importance during finetuning 40
A.2 Trials of base tokenization 42
A.3 Trials with more consecutively masked tokens 43
A.4 Trials of higher k-mer stride 43

List of Tables

3.1 Evaluation Datasets 12
3.2 Comparing different masking setups 13

4.1 Model performance during linear evaluation 19
4.2 Model performance on 150nt inputs 21
4.3 Model performance on 1000nt inputs 21
4.4 Transfer Learning performance 22

A.1 Selected Hyperparameters 40
A.2 Pretraining Time 41
A.3 Training Time during supervised finetuning 41
A.4 Trials of base tokenization 42
A.5 Trials with more consecutively masked tokens 43
A.6 Trials of higher k-mer stride 43

Bibliography

T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

B. Alipanahi, A. Delong, M. Weirauch, and B. Frey. Predicting the sequence specificities of dna- and rna-binding proteins by deep learning. *Nature biotechnology*, 33, 07 2015. doi: 10.1038/nbt.3300.

A. Aliper, S. Plis, A. Artemov, A. Ulloa, P. Mamoshina, and A. Zhavoronkov. Deep learning applications for predicting pharmacological properties of drugs and drug repurposing using transcriptomic data. *Molecular Pharmaceutics*, 13(7):2524–2530, 2016. doi: 10.1021/acs.molpharmaceut.6b00248. URL `https://doi.org/10.1021/acs.molpharmaceut.6b00248`. PMID: 27200455.

E. Asgari and M. R. K. Mofrad. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PLOS ONE*, 10(11):1–15, 11 2015. doi: 10.1371/journal.pone.0141287. URL `https://doi.org/10.1371/journal.pone.0141287`.

Ž. Avsec, V. Agarwal, D. Visentin, J. R. Ledsam, A. Grabska-Barwinska, K. R. Taylor, Y. Assael, J. Jumper, P. Kohli, and D. R. Kelley. Effective gene expression prediction from sequence by integrating long-range interactions. *bioRxiv*, 2021. doi: 10.1101/2021.04.07.438649. URL `https://www.biorxiv.org/content/early/2021/04/08/2021.04.07.438649`.

D. Bahdanau, K. Cho, et al. Neural machine translation by jointly learning to align and translate. arxiv preprint arxiv: 1409.0473. 2014.

J. M. Bartoszewicz, A. Seidel, and B. Y. Renard. Interpretable detection of novel human viruses from genome sequencing data. *NAR Genomics and Bioinformatics*, 3(1), 02 2021. ISSN 2631-9268. doi: 10.1093/nargab/lqab004. URL `https://doi.org/10.1093/nargab/lqab004`. lqab004.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 5:157–66, 02 1994. doi: 10.1109/72.279181.

P. Bhargava, A. Drozd, and A. Rogers. Generalization in NLI: Ways (not) to go beyond simple heuristics. In *Proceedings of the Second Workshop on Insights from Negative Results in NLP*, pages 125–135, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.insights-1.18. URL `https://aclanthology.org/2021.insights-1.18`.

BMBF. Genomenet – entwicklung und evaluierung von genomenet für die de novo identifizierung von noch unbekannten genomischen strukturen und zur probabilistischen dna-sequenzimputation - dlr gesundheitsforschung, Apr 2020. URL https://www.gesundheitsforschung-bmbf.de/de/genomenet-entwicklung-und-evaluierung-von-genomenet-fur-die-de-novo-identifizierung-von-10890.php.

D. Boeckaerts, M. Stock, B. Criel, H. Gerstmans, B. De Baets, and Y. Briers. Predicting bacteriophage hosts based on sequences of annotated receptor-binding proteins. *Scientific Reports*, 11, 01 2021. doi: 10.1038/s41598-021-81063-4.

T. Brown. *Chapter 7, Understanding a Genome Sequence*. Oxford: Wiley-Liss, 2002. URL https://www.ncbi.nlm.nih.gov/books/NBK21136/.

H. Buermans and J. den Dunnen. Next generation sequencing technology: Advances and applications. *Biochimica et Biophysica Acta (BBA) - Molecular Basis of Disease*, 1842(10):1932–1941, 2014. ISSN 0925-4439. doi: https://doi.org/10.1016/j.bbadis.2014.06.015. URL https://www.sciencedirect.com/science/article/pii/S092544391400180X.

A. Chakravarty and J. Sivaswamy. Race-net: A recurrent neural network for biomedical image segmentation. *IEEE Journal of Biomedical and Health Informatics*, 23(3):1151–1162, 2019. doi: 10.1109/JBHI.2018.2852635.

K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, Oct. 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL https://aclanthology.org/D14-1179.

J. Clauwaert and W. Waegeman. Novel transformer networks for improved sequence labeling in genomics. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 19(1):97–106, 2022. doi: 10.1109/TCBB.2020.3035021.

M. G. S. Consortium. Initial sequencing and comparative analysis of the mouse genome. *Nature*, 420(6915):520–562, 12 2002. doi: 10.1038/nature01262.

J. Devlin, M.-W. Chang, K. Lee, and K. N. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 2018. URL https://arxiv.org/abs/1810.04805.

J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In *International conference on machine learning*, pages 1243–1252. PMLR, 2017.

H. A. Gündüz, M. Binder, X.-Y. To, R. Mreches, P. C. Münch, A. C. McHardy, B. Bischl, and M. Rezaei. Self-genomenet: Self-supervised learning with reverse-complement context prediction for nucleotide-level genomics data, 2022. URL https://openreview.net/forum?id=92awwjGxIZI.

D. Guo, M. Li, Y. Yu, Y. Li, G. Duan, F.-X. Wu, and J. Wang. Disease inference with symptom extraction and bidirectional recurrent neural network. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 864–868, 2018. doi: 10.1109/BIBM.2018.8621182.

O. Henaff. Data-efficient image recognition with contrastive predictive coding. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4182–4192. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/henaff20a.html.

S. F. S. Ho, N. Wheeler, A. D. Millard, and W. van Schaik. Gauge your phage: Benchmarking of bacteriophage identification tools in metagenomic sequencing data. *bioRxiv*, 2022. doi: 10.1101/2021.04.12.438782. URL https://www.biorxiv.org/content/early/2022/05/24/2021.04.12.438782.

S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In S. C. Kremer and J. F. Kolen, editors, *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press, 2001.

S. Hu, R. Ma, and H. Wang. An improved deep learning method for predicting dna-binding proteins based on contextual features in amino acid sequences. *PLOS ONE*, 14:e0225317, 11 2019. doi: 10.1371/journal.pone.0225317.

U. Hwang, S. Choi, H.-B. Lee, and S. Yoon. Adversarial training for disease prediction from electronic health records with missing data. *arXiv preprint arXiv:1711.04126*, 2017.

H. Iuchi, T. Matsutani, K. Yamada, N. Iwano, S. Sumi, S. Hosoda, S. Zhao, T. Fukunaga, and M. Hamada. Representation learning applications in biological sequence analysis. *Computational and Structural Biotechnology Journal*, 19:3198–3208, 2021. ISSN 2001-0370. doi: https://doi.org/10.1016/j.csbj.2021.05.039. URL https://www.sciencedirect.com/science/article/pii/S2001037021002208.

Y. Ji, Z. Zhou, H. Liu, and R. V. Davuluri. DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. *Bioinformatics*, 37(15):2112–2120, 02 2021. ISSN 1367-4803. doi: 10.1093/bioinformatics/btab083. URL https://doi.org/10.1093/bioinformatics/btab083.

N. Kalchbrenner, L. Espeholt, K. Simonyan, A. v. d. Oord, A. Graves, and K. Kavukcuoglu. Neural machine translation in linear time. arXiv, 2016. doi: 10.48550/ARXIV.1610.10099. URL https://arxiv.org/abs/1610.10099.

L. Kasman and L. Porter. Bacteriophages. *StatPearls [Internet]. Treasure Island*, September 2021. URL https://www.ncbi.nlm.nih.gov/books/NBK493185/.

N. Q. K. Le, E. K. Y. Yapp, Q.-T. Ho, N. Nagasundaram, Y.-Y. Ou, and H.-Y. Yeh. ienhancer-5step: Identifying enhancers using hidden information of dna sequences via chou's 5-step rule and word embedding. *Analytical Biochemistry*, 571: 53–61, 2019. ISSN 0003-2697. doi: https://doi.org/10.1016/j.ab.2019.02.017. URL https://www.sciencedirect.com/science/article/pii/S0003269719300788.

N. Q. K. Le, Q.-T. Ho, T.-T.-D. Nguyen, and Y.-Y. Ou. A transformer architecture based on BERT and 2D convolutional neural network to identify DNA enhancers from sequence information. *Briefings in Bioinformatics*, 22(5), 02 2021. ISSN 1477-4054. doi: 10.1093/bib/bbab005. URL `https://doi.org/10.1093/bib/bbab005`. bbab005.

J. Li, Y. Yao, H. Xu, L. Hao, Z. Deng, R. Kumar, and H.-Y. Ou. Secret6: A web-based resource for type vi secretion systems found in bacteria. *Environmental Microbiology*, 17, 01 2015. doi: 10.1111/1462-2920.12794.

W. Liang. Segmenting dna sequence into words based on statistical language model. *Nature Precedings*, 2012. doi: 10.1038/npre.2012.6939.1.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Bkg6RiCqY7`.

T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1166. URL `https://aclanthology.org/D15-1166`.

S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer Sentinel Mixture Models. *arXiv e-prints*, art. arXiv:1609.07843, Sept. 2016.

S. Mo, X. Fu, C. Hong, Y. Chen, Y. Zheng, X. Tang, Z. Shen, E. P. Xing, and Y. Lan. Multi-modal Self-supervised Pre-training for Regulatory Genome Across Cell Types. *arXiv e-prints*, art. arXiv:2110.05231, Oct. 2021.

NIH. A brief guide to genomics, Aug 2020. URL `https://www.genome.gov/about-genomics/fact-sheets/A-Brief-Guide-to-Genomics`.

A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding, 2018. URL `https://arxiv.org/abs/1807.03748`.

M. Oubounyt, Z. Louadi, H. Tayara, and K. T. Chong. Deepromoter: Robust promoter predictor using deep learning. *Frontiers in Genetics*, 10, 2019. ISSN 1664-8021. doi: 10.3389/fgene.2019.00286. URL `https://www.frontiersin.org/article/10.3389/fgene.2019.00286`.

R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL `https://proceedings.mlr.press/v28/pascanu13.html`.

R. Paulus, C. Xiong, and R. Socher. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HkAClQgA-.

S. Pereira, A. Pinto, V. Alves, and C. A. Silva. Brain tumor segmentation using convolutional neural networks in mri images. *IEEE Transactions on Medical Imaging*, 35(5): 1240–1251, 2016. doi: 10.1109/TMI.2016.2538465.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://aclanthology.org/N18-1202.

M.-M. Pust and B. Tümmler. Identification of core and rare species in metagenome samples based on shotgun metagenomic sequencing, fourier transforms and spectral comparisons. *ISME Communications*, 1:2, 03 2021. doi: 10.1038/s43705-021-00010-6.

D. Quang and X. Xie. DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences. *Nucleic Acids Research*, 44(11):e107–e107, 04 2016. ISSN 0305-1048. doi: 10.1093/nar/gkw226. URL https://doi.org/10.1093/nar/gkw226.

A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

A. Rives, J. Meier, T. Sercu, S. Goyal, Z. Lin, J. Liu, D. Guo, M. Ott, C. L. Zitnick, J. Ma, and R. Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioRxiv*, 2020. doi: 10.1101/622803. URL https://www.biorxiv.org/content/early/2020/12/15/622803.

S. Roux, D. Páez-Espino, I.-M. A. Chen, K. Palaniappan, A. Ratner, K. Chu, T. B. K. Reddy, S. Nayfach, F. Schulz, L. Call, R. Y. Neches, T. Woyke, N. N. Ivanova, E. A. Eloe-Fadrosh, and N. C. Kyrpides. IMG/VR v3: an integrated ecological and evolutionary framework for interrogating genomes of uncultivated viruses. *Nucleic Acids Research*, 49(D1):D764–D775, 11 2020. ISSN 0305-1048. doi: 10.1093/nar/gkaa946. URL https://doi.org/10.1093/nar/gkaa946.

V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

E. W. Sayers, M. Cavanaugh, K. Clark, J. Ostell, K. D. Pruitt, and I. Karsch-Mizrachi. GenBank. *Nucleic Acids Research*, 48(D1):D84–D86, 10 2019. ISSN 0305-1048. doi: 10.1093/nar/gkz956. URL https://doi.org/10.1093/nar/gkz956.

J. Shang, X. Tang, R. Guo, and Y. Sun. Accurate identification of bacteriophages from metagenomic data using Transformer. *Briefings in Bioinformatics*, 23(4), 06 2022. ISSN 1477-4054. doi: 10.1093/bib/bbac258. URL https://doi.org/10.1093/bib/bbac258. bbac258.

P. Simmonds and P. Aiewsakun. Virus classification – where do you draw the line? *Arch Virol. 2018 Aug*, 2018.

J. Stamatoyannopoulos, M. Snyder, R. Hardison, B. Ren, T. Gingeras, D. Gilbert, M. Groudine, M. Bender, R. Kaul, T. Canfield, E. Giste, A. Johnson, M. Zhang, G. Balasundaram, R. Byron, V. Roach, P. Sabo, R. Sandstrom, A. Stehling, and L. Adams. An encyclopedia of mouse dna elements (mouse encode). *Genome biology*, 13:418, 08 2012. doi: 10.1186/gb-2012-13-8-418.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf.

A. Trabelsi, M. Chaabane, and A. Ben-Hur. Comprehensive evaluation of deep learning architectures for prediction of DNA/RNA sequence binding specificities. *Bioinformatics*, 35(14):i269–i277, 07 2019. ISSN 1367-4803. doi: 10.1093/bioinformatics/btz339. URL https://doi.org/10.1093/bioinformatics/btz339.

M. Vailati-Riboni, V. Palombo, and J. J. Loor. *What Are Omics Sciences?*, pages 1–7. Springer International Publishing, Cham, 2017. ISBN 978-3-319-43033-1. doi: 10.1007/978-3-319-43033-1_1. URL https://doi.org/10.1007/978-3-319-43033-1_1.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

R. Wang, Z. Wang, J. Wang, and S. Li. Splicefinder: ab initio prediction of splice sites using convolutional neural network. *BMC Bioinformatics*, 20, 12 2019. doi: 10.1186/s12859-019-3306-3.

S. Wang, M. Khabsa, and H. Ma. To pretrain or not to pretrain: Examining the benefits of pretrainng on resource rich tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2209–2213, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.200. URL https://aclanthology.org/2020.acl-main.200.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, Oct. 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

S. Wu, Z. Fang, J. Tan, M. Li, C. Wang, Q. Guo, C. Xu, X. Jiang, and H. Zhu. DeePhage: distinguishing virulent and temperate phage-derived sequences in metavirome data with a deep learning approach. *GigaScience*, 10(9), 09 2021. doi: 10.1093/gigascience/giab056. URL `https://doi.org/10.1093/gigascience/giab056`. giab056.

X.-Y. Yan, P.-W. Yin, X.-M. Wu, and J.-X. Han. Prediction of the drug–drug interaction types with the unified embedding features from drug similarity networks. *Frontiers in Pharmacology*, 12, 2021. ISSN 1663-9812. doi: 10.3389/fphar.2021.794205. URL `https://www.frontiersin.org/article/10.3389/fphar.2021.794205`.

Y. Zhang, J. Yan, S. Chen, M. Gong, G. Dongrui, M. Zhu, and W. Gan. A review on the application of deep learning in bioinformatics. *Current Bioinformatics*, 15, 07 2020. doi: 10.2174/1574893615999200711165743.

J. Zhou and O. Troyanskaya. Predicting effects of noncoding variants with deep learning-based sequence model. *Nature methods*, 12, 08 2015. doi: 10.1038/nmeth.3547.

A. Appendix

A.1 Task-Specific Sequence Creation

A.1.1 Examples for Evaluation

The process of creating input sequences for prediction is mimicked from the process employed in *self-genomenet*. For genome sequences shorter than the input length times a limit per genome of 64, a starting nucleotide is sampled out of the first 10% of the sequence. From this point on, as many as possible examples are drawn without overlap. For longer genome sequences, 64 examples are randomly extracted without overlap.

A.1.2 Examples for Finetuning

Nucleotide sequences of the desired length are sampled for each FASTA-file up to a maximum cap without replacement. For genome sequences that cannot accommodate this number of examples, they are drawn with replacement until they do, up to a defined limit of how many repetitions there are to be. Examples are created with these set parameters and then balanced by class. Exact calls that lead to the datasets used can be found on *github*. For 150nt length inputs, sequence creation results in $\sim 8M$ examples for 10% of lables, $\sim 2M$ for 1% and $\sim 500k$ for 0.1%. For inputs of 1000nt length, $\sim 2M$ examples are created for 10% of lables, $\sim 500k$ for 1% and $\sim 166k$ for 0.1%. The lower number of examples here is mainly chosen for memory reasons and is not a limit to how many could be created.

A.2 Definitions

A.2.1 AdamW

AdamW represents the applied optimizer during training of the above experiments. It is an amendment of *Adam*, in which weight decay is decoupled from the gradient-based update. This optimizer was proposed by Loshchilov and Hutter [2019]. Its algorithm is depicted in Algorithm 1.

Algorithm 1 AdamW

given: $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, \lambda \in \mathbb{R}$
initialize: time step $t \leftarrow 0$, parameter vector $\theta_{t=0} \in \mathbb{R}^n$, first moment vector $m_{t=0} \leftarrow 0$, second moment vector $v_{t=0} \leftarrow 0$, schedule multiplier $\nu_{t=0} \in \mathbb{R}$
repeat
    $t \leftarrow t + 1$
    $\nabla ft(\theta_{t-1}) \leftarrow \text{SelectBatch}(\theta_{t-1})$ ▷ select batch and return the corresponding gradient
    $g_t \leftarrow \nabla f_t(\theta_{t-1})$
    $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$ ▷ here and below all operations are element-wise
    $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
    $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ ▷ β_1 is taken to the power of t
    $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ ▷ β_2 is taken to the power of t
    $\nu_t \leftarrow \text{SetScheduleMultiplier}(t)$ ▷ can be fixed, decay, or also be used for warm restarts
    $\theta_t \leftarrow \theta_{t-1} - \nu_t \left(\alpha \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon) + \lambda \theta_{t-1} \right)$
until *stopping criterion is met*
return optimized parameters θ_t

A.2.2 Cross Entropy Loss

$$CE\left(t, f(s)\right) = -\sum_{i}^{C} t_i log\left(f(s)_i\right) \tag{A.1}$$

where t are labels and $f(s)$ are class probabilities

A.2.3 Metrics

Binary F_1-score:

$$Precision = \frac{TP}{TP + FP} \quad Recall = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{A.2}$$

$$TP = True\,Positives;\ FP = False\,Positives;\ FN = False\,Negatives$$

where baceriophages represent the positive class.

Macro averaged Recall ($Recall_M$) in %:

$$Recall_M = 100 \times \frac{\sum_{i=1}^{n} Recall(c_i)}{n} \tag{A.3}$$

$$\text{for } c_i \in C \quad \text{Set of Classes} \quad \text{and} \quad n \quad \text{Number of Classes}$$

A.2.4 Task Level Types

The concept of different task levels is used throughout. It refers to what the class of the task encompasses. *Genome-level* tasks therefore constitute a classification of a sequence into a class that refers to the entire genome (identifying bacteriophages would be an example of this), whereas tasks of *token-range* (or *region*), like the recognition of certain protein binding sites, imply that the labeled class of the sequence refers to a window containing a limited number of tokens only. Examples of *token-level* tasks may include imputing of a few nucelotides or recognizing short mutations and predicting following genome sections an example of a *strictly sequence-level* task.

A.3 Self-genomenet

Gündüz et al. [2022] propose a contrastive self-supervised architecture of stacked encoding and recurrent networks to learn representations of nucleotide sequences. This model aims to exploit the fact that one strand of DNA is always accompanied by its reverse-complement, a sequence running in opposite direction with complementary nucleotides. Here, genome sequence excerpts of varying length are split into two non-overlapping subsequences. The convolutional encoding network then further splits these into shorter, overlapping patches for which representations are learned. All patches of the second subsequence are, however, turned into their respective reverse-complements beforehand. Both sequences are concurrently fed into a convolutional encoder network and then an LSTM based context network. The pretext task manifests itself in a linear prediction network, where learned embeddings are to predict embeddings of the opposite (following) subsequence contrastively. Other patches of the same minibatch are used as negative examples. This approach claims high efficiency through the model symmetry given by training on a sequence and its reverse-complement as well as sharing weights of the context and prediction networks.

The model is trained with a single convolutional layer of 1024 filters with $k = 24$ and $s = 6$, leading to patches of length 24 and overlap of 6, followed by an LSTM layer with 512 hidden units. A batch size of 512 was used for self-supervised training.

A.4 HPO

The implemented Hyperparameter Optimization framework is built on the package *optuna* [Akiba et al., 2019]. For this setup, some finetuning parameters for *virBERT* models are selected with studies of about 40 trials per model. These parameters include learning rate, weight decay, dropout rate, batch size and warm-up percentage. While the 10% label availability data split is used, examples are sampled down to a quarter of what the full set would have. As figure A.1 also shows, batch size seems least important to model performance throughout. Hence an equal batch size of 256 is selected for 150nt inputs and 64 for 1000nt inputs. All model variants prefer a lower dropout rate of 10%. Table A.1 depicts the parameters selected for 150nt input sequences over the 10% label availability scenario. Notably, other parameters have also been experimented with full scale. These performed best however.

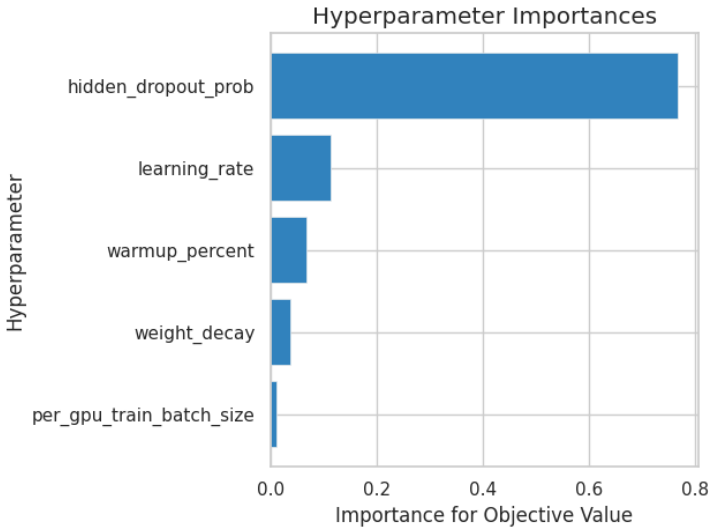


Figure A.1: Importance of hyperparameters during finetuning on the 150nt phage/non-phage task for *virBERT*.

model	batch size	dropout%	learning rate	weight decay	warm-up%
virBERT	256	0.1	$1.6e^{-5}$	1.0	0.2
virBERT-mask8	256	0.1	$2.0e^{-5}$	0.1	0.2
virBERT-stride3	256	0.1	$3.0e^{-5}$	0.1	0.2

Table A.1: Hyperparameters selected for the 3 different *virBERT* variations during finetuning on the 150nt phage/non-phage task with 10% label availability.

A.5 Training Time

model	Training time	#GPUs	GPU Type	accumulated time
self-genomenet	158	-	RTX 2080 Ti	-
virBERT	190	8	A100	1520
virBERT-mask8	190	8	A100	1520
virBERT-stride3	141	5	A100	705

Table A.2: A rough estimation of the time spent pretraining the different models in hours. For *virBERT* models, time spent validating the model during training is included. The time reported for *self-genomenet* includes finetuning on the 0.1% label task.

	150nt		1000nt		
model	#GPUs	steps/h	#GPUs	steps/h	steps/h/gpu
virBERT(-mask8)	1	4870	2	3564	1782
virBERT-stride3	1	13158	1	4478	4478

Table A.3: Comparison of the achieved training steps per hour between the different tokenization approaches of *base* and *stride3*. Batch sizes are equal per input length task. This is taken from finetuning the pretrained models in a supervised fashion on the phage/non-phage classification task with 10% label availability. Time spent validating during training is included.

A.6 Self-Supervised Trials

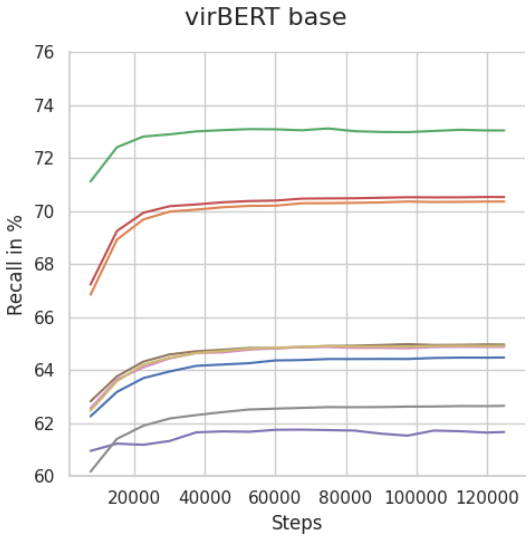
After pretraining each trial (on average 13h on 4 A100 GPUs), a linear layer with a softmax classifier is attached to each model. Only this layer is then trained in a supervised manner with 4M class-balanced examples of 150 nucleotide length on the virus phage/no-phage task. This is done over 2 epochs with a learning rate of $2e^{-5}$ and a dropout rate of 0.3. We evaluate a trial's performance on 1.5M examples created from the validation split of the data.

Masking more tokens per masking location was attempted by masking k+2, k+4 and k+5 tokens in a row, hiding 3, 5 and 6 nucleotides per k+x masked tokens, respectively. For trials with higher strides of 3 and 6, the masking strategy was adjusted to always hide 6 nucleotides from the model, so to 3 successive tokens for a stride of 3 and just 1 for a stride of 6. Dataset preprocessing changes are explored in multiple ways. Since a sequence cut length lower-bound of 5 nucleotides, as Ji et al. [2021] suggests, is theorized to be low, as a sequence of 5 creates an empty

input for 6-mers and in general results in lower length sequences on average, higher cut length lower-bounds of 12 and 36 are examined. A bias of 0.3 towards maximum length input sequences, compared to Ji et al. [2021] implementation of a bias of 0.5, is also tested. Additionally, the effect of the ratio between non-overlap cut sequences and randomly sampled ones is explored, ranging from no non-overlap sequences to a ratio of 3 (times more samples sequences). Because models with inputs tokens of higher stride need longer length sequences to attain the same input lengths, they generally have a higher ratio of sampled sequences to achieve the same amount of examples to train on. The maximum input lengths are however also changed in some of the higher stride trial runs, reducing the number of nucleotides per sequences to 1000 and 510.

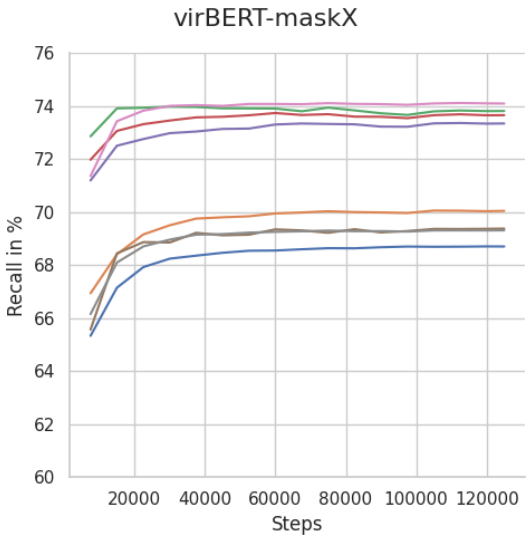
Results

The three figures below (A.2, A.3, A.4) show the accuracy on the 150nt phage/non-phage task of some of the trials conducted. The plots depict supervised learning with frozen representation layers and the tables the respective trial's change to its pretraining parameters. Over all figures, *low* refers to $3e^{-4}$, *base* to $4e^{-4}$, *med* to $6e^{-4}$ and *high* to $1e^{-3}$ in terms of learning rate. If not specified, the learning rate is that of base.



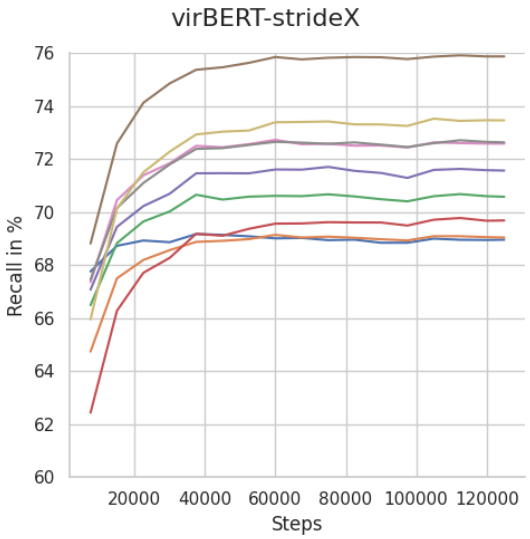
id	description
base	base virBERT
1	mask% 20
2	med lr
3	low lr
5	lower weight decay
8	higher low_b
9	higher ratio
12	med lr, fluid mask%
13	high lr

Figure A.2: Different setups for the standard *virBERT* setup. A comparison of runs base, 2, 3 and 13 reveals $1e^{-3}$ to be the most suitable learning rate. Trial 1 (vs. base) shows that increasing masking percentage does not improve accuracy (even when fluid: 12 vs. 2). The higher sequence input lower bound length trial (8) appears no different to base, whereas increasing the ratio of randomly sampled to non-overlap cut input sequences (9) performs worse.



id	description
7	m8, med lr
14	m8, med lr, mask% 20
15	m10, med lr
19	m8, high lr
20	m10, high lr
21	m11, high lr
26	m8, lr $2e^{-3}$
30	= 19 but warm-up% 10

Figure A.3: Comparison of some of the trials where more than k tokens are masked. mx refers to the number of consecutively masked tokens. Similarly to base trials, performance increases with learning rate up to $1e^{-3}$ (19, 20, 21), whereas an even higher learning rate (26) performs considerably worse. While they are never far apart, *mask8* setups emerge as the best variant.



id	description
11	s3, base lr
16	s6, base lr
17	s3, high lr
18	s6, high lr
25	s3, upp_b 1000nt
28	s3, upp_b 510nt, cap
29	s3, upp_b 510nt
31	s3, upp_b 1000nt, cap
33	= 31 but lower bias

Figure A.4: *stride-x* variant trials of *virBERT*. Trials 25 through 33 also employ the *high* learning rate. A tokenization with a stride of 3 performs better than a stride of 6 in these trials. Trial 9 and 25 show, that reducing the upper bound of input sequence lengths closer to that of the task increases prediction accuracy. Similarly, the more diverse input length distribution of trial 33 with a lower bias towards maximum length inputs of 0.3 hints at the same effect. Additionally, trial 28 (vs. 29) and trial 31 (vs. 25) both demonstrate that limiting the number of tokens a model accepts also achieves better results.