ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING

ARTIFICIAL INTELLIGENCE

MASTER THESIS

in

Natural Language Processing

SMALL TRANSFORMERS FOR BIOINFORMATICS TASKS

CANDIDATE

Luca Salvatore Lorello

SUPERVISOR

Prof. Paolo Torroni

CO-SUPERVISOR

Dr. Andrea Galassi

Academic year 2020-2021

Session 1st

A fundamental truth about deep learning is that it enables you
to see the real beauty of Nature. For instance, the more I trained transformers,
the more I got to appreciate harvesting tomatoes under the sun of Sicily.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Summary

Bioinformatics heavily relies on textual informations, like DNA and amino-acid sequences, to perform tasks such as classification of sequences, alignment of noisy reads, phenotype prediction, etc., however this information has been historically analyzed with statistical and traditional machine learning approaches, which, due to their nature, could not exploit positional information.

Although recent trends in bioinformatics are trying to align the techniques to more modern approaches based on statistical natural language processing and deep learning, state-of-the-art neural natural language processing techniques remain relatively unexplored in this domain.

Following the trail of Google's BigBird [54], a transformer-based architecture for long sequences, this work aims at exploring the possibility of using transformers in order to build contextual embedding representations of DNA sequences, to improve performance on bioinformatics tasks such as transcription start site localization, mitochondrial metagenomics and chromatin profile prediction.

Although state-of-the-art performances are achieved by large models, a typical bioinformatics lab has limited hardware resources. For this reason, this thesis will focus on small architectures, the training of which can be performed in a reasonable amount of time, while trying to limit or even negate the performance loss compared to SOTA.

In particular, $\mathcal{O}(n)$ attention mechanisms (such as the one proposed by Longformer [3]) and parameter sharing techniques (such as the one proposed by Albert [24]) are jointly explored with respect to two genetic languages: human genome and eukaryotic mitochondrial genome of 2000+ different species.

Two tokenization approaches are explored, one based on fixed length k-mers and the other, proposed by BigBird, based on random length sequences, both followed by byte-pair encoding. Contextual embeddings for each token are learned via pretraining on a language understanding task, both in RoBERTa [29] (masked language modeling) and Albert (masked language modeling and sentence-order prediction) styles to highlight differences in performance and training efficiency.

The learned contextual embeddings are finally exploited for fine tuning a task of localization (transcription start site in human promoters) and two tasks of sequence classification (12S metagenomics in fishes and chromatin profile prediction, single-class and multi-class respectively). To assess multilinguistic properties, the two tasks are fine-tuned from three initialized states:

- random weights (uninitialized transformer),
- correct language model (ie. TSS localization and chromatin profile using the embeddings learned on the human genome, and metagenomics on the mitochondrial embeddings),
- wrong language model (ie. TSS and chromatin profile on mitochondrial embeddings and vice versa).

The results of the experiments demonstrate that combining $\mathcal{O}(n)$ attention and Albert-style parameter sharing negatively affects performance and that each technique should be used separately, moreover, although a BigBird-style tokenization increases performance by means of dataset augmentation, it's been proven to be very expensive in terms of memory and therefore not viable for large datasets on limited hardware.

Network initialization status before each of the fine tuning tasks was useful for learning (the correct language for each task achieved consistently better than the wrong one and the uninitialized network), and the right language had a positive effect in mitigating overfitting, but these improvements with respect to the uninitialized are not substantial.

Using smaller architectures, near SOTA performances are achieved in all the tasks proposed by BigBird (MLM, promoter region prediction and chromatin profile prediction) on the human genome, and a new SOTA has been achieved for the other tasks (promoter region localization and 12S metagenomics). Moreover further experiments with larger architectures consistently improved the previous SOTA in all cases. In spite of this, none of the chosen fine tuning tasks can be considered complex enough to act as a benchmark for neural network-based bioinformatics.

1.2 Thesis objectives

This thesis tries to achieve the following objectives:

- review and verify the limits of current proposals in bioinformatics for DNA sequences analysis,
- assess the applicability of smaller language models in the context of limited hardware availability,
- determine whether currently explored tasks can be considered viable benchmarks for modern approaches to bioinformatics,
- propose possible directions for future research in bioinformatics, highlighting current limitations and opportunities.

1.3 Thesis contributions

In this thesis applicability of small transformers (small enough to be trained on modern laptops) in the domain of bioinformatics is explored, disproving BigBird's claim of a longer context being crucial for performance in tasks involving biological sequences.

The lack of natural "word boundaries" in DNA sequences is addressed both with traditional k-mer tokenization and BigBird-style tokenization, both techniques are followed by byte pair encoding [43]. Results showed that, although performing better thanks to multiple random alignments, BigBird's word tokenization is too expensive for large datasets to be practical.

Two architectures (Longformer and a Longformer-Albert hybrid) are instantiated in a shallow (2 layers only) setup, and tested on four tasks, by using very small contexts (256 tokens):

- Language understanding (both in RoBERTa and Albert flavors),
- Promoter's transcription start site localization (a more complex version of the promoter region prediction task against which BigBird was tested),
- 12S metagenomics on fishes,
- Chromatin profile prediction (another benchmark used by BigBird).

Test results proved that near state of the art performance can still be achieved on such small architectures (although the hybrid network has too many architectural constraints and performance drops significantly) with reduced input size and that proper weight initialization is beneficial for training, but not crucial on the chosen tasks.

Promoter region localization was trained on human data and high performance in cross-species predictions proved empirically that eukaryotic promoters are very similar across species.

Eukaryotic 12S metagenomics is a tasks with practical value and the good performances achieved can be further improved to build an accurate prediction tool faster than current state of the art classifiers for metagenomics.

Chapter 2

Background

2.1 Bioinformatics background

2.1.1 Biological concepts

Although there is no consensus on the definition of "life", one of the characteristics it must encompass is the capability of transmitting informations across generations. All known life-forms have in common the same information-transfer mechanism, which is expressed by the Central Dogma of molecular biology [7], a paraphrased version of which is the following:

- The biological machinery of a life-form is composed of proteins, which operate on other types of molecules,
- Proteins cannot reproduce and the information required to build them is stored in DNA,
- Information always flows from DNA to protein.

Abstracting from all the details (there is also a third kind of molecule, RNA, involved in information transfer), a sequence of nucleotides (DNA) is translated into a sequence of amino-acids (protein), using the biological equivalent of a look-up table, which is known as genetic code. Each of the 64 (4^3)

possible codons (sequences of three nucleotides) of DNA is unambiguously translated to one of the 22 amino-acids (or a stop signal) and almost every living organism uses the same correspondences, this universality property is explained by multiple theories [20].

A single filament of DNA is called a chromosome and in nature it's always found in a compressed form called chromatin. Life-forms are classified in two domains, based on whether chromatin is free inside the cell (Prokarya) or confined inside a second membrane called nucleus (Eukarya). The following are processes referred to eukaryotes, some informations about prokaryotic mechanisms will be given where relevant inside chapter 3.

A gene is the portion of DNA which is read during translation (although a single gene can be mapped to different proteins) and therefore acts as the information unit. It's always composed by a 5' untranslated region (which contains the binding sites for the biological machinery), the open reading frame (the sequence which will be translated) and a 3' untranslated region (which regulates gene expression).

In eukaryotes an open reading frame is typically composed of subsequences which can either be translated in the final protein (exons) or not (introns) in a process called splicing, moreover gene expression is regulated by extra-genic sequences. Promoters are proximal regulatory sequences located immediately before a gene which enable the translation process, acting as "switches", while distal regulatory sequences (enhancers and silencers, which can be located either before or after a gene) regulate the likelihood (and therefore the quantity of proteins produced).

Mitochondria are organelles of eukaryotes involved in energy and heat production, cell proliferation and programmed death (apoptosis). They possess their own genome (which is more similar to prokaryotes), a slightly modified genetic code and reproduce independently from the main cell body, these characteristics suggest that they were symbionts of a common ancestor to all eukaryotes, which later specialized their function (while the host lost the ability of producing energy on its own) [16].

2.1.2 Bioinformatics concepts

Sequencing is the process of reading a DNA strand or protein into a sequence of character for later bioinformatics use. Since its invention, three generations of DNA sequencing technologies have appeared. In first and second generation devices a single filament of DNA is read by exploiting the biological process of replication.

During replication, the double strand of DNA is denatured and a short sequence, called primer, binds to one of the two filaments and then the replication machinery (various proteins, including the DNA polymerase, DNAP) is recruited. The DNAP adds one nucleotide at the time, starting from the primer and moving from the 5' to the 3' end. In nature primers have fixed patterns, however arbitrary sequences of 18–24bp (base pairs) can be used as primers during in vitro replication and can act as "pattern matching" tools to extract a sequence starting from their complementary and onwards.

In first generation (Sanger sequencing), DNA is replicated in four different chambers, by using a mix of the four "normal" deoxynucleotide-triphosphates (dNTP), along with a dideoxynucleotide-thiphosphate (ddNTP, eg. G) which is also marked (eg. radioactively or with a fluorescent tag). The ddNTP cannot form phosphodiesteric bonds and therefore stop replication as soon as it's added, this produces reads of random length, which however always start with the primer and always terminate with the same ddNTP for each chamber. An electrophoresis run is performed to separate each replicated strand by length and the sequence can be therefore reconstructed.

Let's consider, as toy example, the following reads:

- Chamber containing ddA: `(primer)-??A, (primer)-????A`
- Chamber containing ddT: `(primer)-?T`

- Chamber containing ddC: `(primer)-?????C, (primer)-??????C`
- Chamber containing ddG: `(primer)-G, (primer)-???G`

The reconstructed sequence would be: `(primer)-GTAGACC`.

Second generation devices (Next Generation Sequencing, NGS) still exploit in vitro replication of DNA, however use techniques for reading multiple sequences simultaneously (multiplexing), achieving a much faster reading rate and cheaper results (the human genome required \$3 billions using Sanger sequencing, compared to \$100 for NGS sequencing), for example, the Illumina sequencing technique binds short fragments of DNA to a substrate and replicates them using fluorescent dNTP (each of the four has a different color) which "shine" as soon as they are incorporated by the DNAP. By observing fluorescent clusters in the substrate matrix changing color over time, it's possible to read thousands of strands at the same time, however clusters become harder to distinguish over longer strands, imposing a practical limit of about 300bp for each fragment.

Third generation devices overcome the length limitations of NGS by using different reading technologies (eg. Oxford Nanopore doesn't require in vitro replication, forcing instead DNA strands to pass inside a matrix of nanometer-scale pores while reading changes in electrical fields, which are characteristic of each of the four nNTP, achieving a theoretically "infinite" reading length), however the technology is not yet mature and it's characterized by higher error rates compared to NGS.

Sequencing is a noisy process and, with the exception of third generation techniques, produces short reads. These problems are overcome by software at the first steps of any bioinformatics pipelines. Alignment is the process of reconstructing a longer sequence by merging shorter ones and denoising is the process of correcting reading errors (different algorithms tend however to produce slightly different results, which in fields like metagenomics may affect the outcome of some experiments [33]), an assembly is a sequence which has been both aligned and denoised. An assembly can be built either by using the reads alone (de-novo assembly) or by matching a reference genome (reference-aligned assembly).

A reference genome is a digital sequence which acts as an example genome for a given organism. It doesn't correspond to any existing individual since it's a mosaic assembled from multiple individuals in order to capture the "average" sequence of the entire population. Over time a reference genome can be subject to revision, because newer sequencing technologies allow to fill some gaps caused by previously unknown or unalignable sequences. Other than building assemblies from short reads, the main uses of reference genomes are to provide a coordinate system for genetic variants (eg. to characterize mutations or to provide a compact notation in which, instead of the entire sequence, only the differences with respect to the reference are given) and for queries (eg. with the BLAST [1] algorithm, which matches a given pattern against a database of many references, within a given error threshold).

2.1.3 Promoter region prediction

Promoters are regions of DNA located immediately upstream of a gene, which act as binding sites for transcription-related proteins (ie. RNA polymerases, transcription factors, etc.). In eukaryotes, promoters can be composed of three parts:

- Core: the minimal sequence required to start transcription:
  - transcription start site (TSS),
  - RNA polymerase (RNAP) binding site: there are three RNAP in eukaryotes, each has a different function and binds different sequences,
  - general transcription factor binding sites: sequences which recruit the transcription factor II B (TFIIB), like the TATA-box, the B-recognition element, etc.,

  - various other binding sites which are dependent on the downstream gene.

- Proximal element: regulatory region directly upstream of the promoter, it binds specific transcription factors, acting as an on/off switch for the transcription of a specific gene.
- Distal element (ehnancers and silencers): regions located far away from the gene which bind activator or repressor proteins, modifying the rate of transcription.

Some of the patterns commonly found inside an eukaryotic promoter are:

- TATA-box: located approximately 30bp upstream of the TSS, it binds the TATA-binding protein (TBP) which starts the recruitment process of transcription machinery; although it was one of the first motifs discovered, less than 20% of human promoters have a TATA-box,
- Initiator element (Inr): similar in function to a TATA-box, but more common; if they are both present, recruitment is more efficient, leading to higher transcription rates,
- GC-box: located approximately 110bp upstream of the TSS, it binds various general transcription factors,
- CAT-box: located approximately 60bp upstream of the TSS, it's frequently absent for house-keeping genes (genes which are translated by every cell); it interacts with the GC-box to regulate the quantity of transcription.

The Eukaryotic Promoter Database (EPD) [11, 10] collects all the known promoter sequences of selected organisms (ranging from *Plasmodium falciparum* to *Homo sapiens*) and allows to extract arbitrary length sequences with coordinates centered at the TSS.

Oubounyt et al. [36] and later Zaheer et al. [54] have used neural networks (a convolutional NN and a transformer, respectively) to classify whether a given sequence contains a promoter or not, achieving near perfect performance, however their dataset suffered from two sources of bias:

- every sequence was aligned in the same way with respect to the TSS (-7999 to +8000bp for BigBird),
- the negative samples were created by randomly replacing 60% of the sequences of valid promoters and thus creating “easy” negatives by destroying one or more of the previously described patterns.

2.1.4 Metagenomics

Metagenomics is the study of DNA samples collected from the environment, instead of single individuals, for the purpose of classifying the population in that environment. Due to the intrinsic need of having a shared primer when amplifying sequences, metagenomics reads must be focused on genes which are:

- Shared across all the organisms which need to be analyzed (ie. they must contain highly preserved regions),
- Useful for classification (ie. they must contain hyper-variable regions which can be reliably linked to a specific organism).

These characteristics are possessed by genes which are translated into something which is both important for survival and shared among all species under exam. Ribosomes are macro molecules involved in translation (and therefore shared by all living organisms) which satisfy both conditions, metagenomics exploits genes composing ribosomal subunits for classification of samples:

- The 16S rRNA gene for prokaryotes,
- The 18S rRNA gene for eukaryotes,

- The 12S rRNA gene for mitochondria.

Metagenomics studies are useful in ecology (eg. soil ecosystem analysis [8]), health (eg. human gut microbiome analysis [15]), food designation of origin (eg. cheese produced in a specific region have specific microbiological profiles [53]), forensics (eg. determining the movements of stolen artifacts [39]), etc.

2.1.5 Chromatin profile prediction

The various degrees of condensation of chromatin play an important role in gene regulation. The main components determining the chromatin profile are histones, non-histonic proteins and RNA, which can be used to predict tissue type, transcription-factor binding sites and long term interactions [44].

Some means of identifying chromatin profiles are:

- Type of binding transcription factor (eg. the STAT3 factor, which mediates a pathway involved in embryonic development, immunity and tumor formation),
- Histone profile (histones are “spools” around which the DNA coils, chemical modifications to them, eg. H3K4me3, control the degree of coiling),
- DNAse I hypersensitivity sites (regions not wrapped by histones forming a loop long enough for the DNAse I enzyme, a molecular “scissor” to fit).

Cell differentiation is mainly achieved by means of permanent epigenetic modifications, therefore the same DNA sequence in two different cells (eg. a CD4+ T-lymphocyte and an epithelial cell) will have different chromatin profiles.

Zhou et al. [56] have produced a dataset of DNA sequences belonging to 919 chromatin profiles based on cell type, transcription factor, histone modification, DNAse I hypersensitivity and, since chemical modifications may introduce artifacts, also treatment prior to analysis, by associating each sequence to a multi-label 919-dimensional vector.

2.2 Natural language processing background

2.2.1 Language models

A language model is a function associating a probability to sequence of words in order to characterize a language. Such a function is useful in nearly every natural language processing task, including, but not limited to, text generation, word disambiguation, error correction, translation, etc.

Statistical language models approximate the likelihood of a sequence of words appearing in the language as the conditional probability of observing the last token, given the $n-1$ previous ones, under the assumption that probabilities on *n-grams* are descriptive enough to approximate the entire language and therefore imposing an n^{th} order Markov property on the model.

Although statistical language models can be computed quite easily by counting occurrences in a reference corpus, they suffer from three main drawbacks:

- curse of dimensionality: the models are exponential in size both with respect to the vocabulary and the value of n,
- sparsity: most of the sequences of words have a zero probability of appearing in the language,
- limited observability: the reference corpus doesn't contain all the possible sequences allowed by the language, causing the underestimation of probabilities.

Neural language models exploit the latent representation learned by neural networks to model a language as a continuous vector space in which each word is represented by a point. These word representations are called embeddings because they are a lower-dimensional projection of a larger vector space (the "full" space learned by statistical language models).

2.2.2 Word embeddings

Although training a neural network to predict a traditional n-gram model (ie. given $n-1$ words, predict the likelihood of the next one) can still produce an embedding for the language model, training a skip-gram model (ie. given a central word, learn the probability of its surroundings) yields interesting algebraic properties of the learned embedding, such as mapping semantic relations into linear combinations [31, 37].

One of the main drawbacks of "traditional" word embeddings is the fact that, due to the implicit dimensionality reduction compared to statistical language models, they loose contextual information (ie. the representation of the same word appearing in different contexts, possibily having different meanings, is collapsed to the same point). To overcome this issue, contextual word embeddings, which represent a word in relation to the context it's placed, have been proposed. ELMo [38] exploited the hidden state of a bidirectional LSTM as a contextual word embedding capable of better representing a language model. Since its introduction, other approaches which bypass the training limits of an LSTM (most importantly the sequential nature of training) have been developed.

2.2.3 Attention and transformers

An encoder-decoder [45] architecture is a neural network capable of converting a sequence into another one by using an intermediate vector as shared embedding between the two sequences (ie. the input sequence is encoded into the embedding and the output sequence decodes the embedding into a different domain).

Shortly after their introduction, it became clear that these architectures could greatly benefit, not only from positional informations, but also from mechanisms capable of giving “more importance” to some tokens compared to others. An attention mechanism basically implements a fuzzy look-up operation which, given a key-value dictionary and a query, returns the value (or combination of values) associated to the key which is “more similar” to the query, this concept of similarity can be task-dependent and therefore learned.

Bahdanau et al. [2] propose to compute the similarity score between key and query by first computing an alignment score on query and key, softmaxing it (to obtain values in $[0; 1]$) and then multiplying it element-wise with the values vector. The alignment score is computed as the hyperbolic tangent of the weighted sum of key and query, and then multiplied with a learned matrix. Luong et al. [30], instead, propose to use the dot-product of key and query inside the alignment score computation. In either case, the operations can be performed natively by fully-connected layers of a neural network.

Transformers are an encoder-decoder architecture proposed by Vaswani et al. [48], which completely rely on (Luong-style) attention mechanism, without the limitations of recurrent or convolutional neural networks, achieving higher performance. A typical transformer’s encoder is a stack of blocks, each composed of a multi-head self-attention (ie. multiple attention modules in which key and query are the same vector) fed by the previous layer and followed by a fully-connected layer. Likewise, a typical decoder has two multi-head self-attentions, one fed by the previous layer and the other fed by the encoder output, and a fully-connected layer.

Bidirectional Encoder Representations from Transformers (BERT) [9] is an approach, similar to ELMo’s use of LSTM’s hidden state, to produce highly informative contextual word embeddings which were the key to improving state-of-the-art performance in multiple natural language tasks.

BERT approach is based on a transformer encoder pretrained on a task-agnostic problem to produce word embeddings, which are then fed to a task-specific head (usually a single fully-connected layer with a properly chosen activation function) which is fine-tuned on the desired task. This approach yields better results, compared to training directly the entire architecture, namely:

- Convergence in the final task is achieved faster (in case of limited data, training directly on the task could even be impossible),
- Embeddings can be reused for different tasks in the same language,
- During fine-tuning, the “base” language model is further “specialized” for the specific task because the weights of the lower layers are not frozen.

The encoder is pretrained simultaneously on two language understanding tasks which allow to build a language model, later exploited during fine-tuning:

- masked language modeling (MLM): a random subset of the input is masked and the network is trained to recover the missing tokens based on the context,
- next sentence prediction (NSP): the concatenation of two sentences is given and the network is trained to determine whether they are consecutive or whether they are unrelated (possibly coming even from different documents).

The MLM task is typically performed by masking 15% of tokens and, to mitigate discrepancies of performance between training and prediction, only 80% of those are actually replaced by the `<mask>` token. The rest is either kept as is, or replaced with a random token. In order to reduce overfitting, dynamic MLM was proposed by Liu et al. [29], in which the random masking is recomputed at the beginning of each epoch. Another alternative which greatly reduces overfitting risks of MLM, achieving a faster convergence, is the use of an adversarial network to replace tokens with plausible alternatives instead of `<mask>` or a completely random token, as proposed by Clark et al. [6].

The NSP task has sparked some controversies on its practical utility, with Liu et al. deeming it unnecessary (as long as dynamic MLM is performed on a larger dataset) and Lan et al. [24] proposing a different task, called sequence-order prediction (SOP). In SOP the network is trained to choose whether the first sentence follows the second one, or whether they have been swapped. A dataset for this task is clearly easier to produce and the authors advocate that it leads to a more robust language modeling, because in NSP there is an asymmetry between positive and negative samples:

- a positive sample is constituted by two consecutive sentences, which are therefore spatially and semantically close,
- a negative sample is constituted by two far away sentences, which may present semantical differences.

A network trained on NSP would therefore learn a “shortcut” by discriminating on semantic similarities alone, while in SOP also the negative samples are spatially and semantically close, forcing the network to learn based only on harder clues.

Self-attention mechanisms in transformers have a memory footprint which grows as $\mathcal{O}(n^2)$ with respect to the input length, and, in BERT, each block has an independent attention mechanism. In order to overcome training time and memory limitations, many alternatives have been proposed [13], the ones explored in this thesis will be described in detail on section 4.1.

2.2.4 Word encoding

Any style of word embedding can be ultimately assimilated to a lookup table which maps a word to a point in a vector space. In practice this often leads to two problems:

- Out of vocabulary words: a word not observed in the training corpus must be handled specifically when encountered at runtime (either by replacing it with a special `<unknown>` token or by taking an arbitrary embedding choice, eg. random or based on similarity with other words),
- Large vocabulary size, leading to an increase of memory requirements.

To solve these problems, instead of computing embeddings on entire words, it's common to compute them for sub-word entities (tokens), trading off vocabulary size with input size (with a character-level embedding at the extreme end, solving both problems at the expenses of a huge increase in input size).

Tokenization of subwords can be performed either by splitting words into k-mers (sequences of k characters) or by using an algorithm which exploits the statistical distribution of symbols inside the training corpus to build an "optimal" vocabulary of tokens, according to some criterion. One of such algorithms is byte-pair encoding (BPE) [43], which, given a target vocabulary size, builds the vocabulary in a bottom-up fashion, by starting from the set of alphabet symbol and iteratively inserting a new symbol constituted by the most frequent sequence of two tokens. This process continues until either the target size is reached or all the words are constituted by a single symbol.

Chapter 3

Related work

Traditional machine learning approaches are characterized by the use of hand-crafted features and (usually) the inability of exploiting positional informations.

For example, Lin et al. [28] used a support vector machine (SVM) on hand-crafted positional features to classify whether an input region is a promoter or not.

Bharanikumar et al. [4] proposed a linear regressor classifier to predict the interaction strength of *E. coli* σ^{70} factor (one of the many different promoter-interacting proteins in prokaryotes) to promoter on hand-crafted features (frequency matrices were extracted from a sequence of 13bp centered around the -35 and -10 regions).

Xiao et al. [50] combine ideas from the previous two and used a cascade of two SVMs to predict σ^{70} promoter sites and their strength.

Yang et al. [51] applied gradient boosting in order to classify enhancer-promoter interactions, on a mix of hand crafted features and word embeddings of two input sequences as putative promoter-enhancer pair.

Yousef et al. [52] used a naive Bayes classifier fed with micro RNA (very short RNA sequences acting as silencers for gene expression) sequences pre-processed with a simple (non-neural) convolution to reduce the input size from 110bp to 21bp.

Murakami et al. [32] used a naive Bayes classifier to predict sites for protein-protein interactions (ie. the localization of physical contact between two proteins), using predicted accessibility and a score matrix extracted as conditional probabilities on the sequences via kernel density estimation as input features.

Artificial neural networks are universal approximators which can be trained directly on input features, delegating the network to learn the best processing by itself, without the need of crafting features manually, but at the expenses of higher computational costs. Although achieving consistently better results than traditional machine learning techniques, surprisingly, so far the use of deep learning in bioinformatics has been relegated to novelty methods and proof of concepts. To further complicate the relationship between deep learning and bioinformatics, there is a huge technical gap with respect to other domains (eg. residual connections, applied in computer vision since 2015 have been proposed in the bioinformatics domain only in 2019 [55]).

Wei et al. [49] classified protein subcellular localization (cells possess a very complex mechanism for transport which determines the final location of a given protein by signals which can be, among other possibilities, patterns inside the protein sequence itself, therefore predicting where a protein will end up is a biologically relevant task) by using a stacked autoencoder followed by a fully connected layer, however the input features are still hand-crafted.

Umarov et al. [47] exploited convolutional neural networks, applied to raw input sequences (each base is a one-hot encoding of the four nucleotides), to classify promoter sequences in a wide range of organisms. Their approach was proven to be robust both with respect to the organism and the type of promoter. They also propose to detect which positions are "functionally relevant" for the promoter, by adversarially introducing noise in the input and checking whether the classifier confidence decreases.

Levy et al. [25] trained a variational autoencoder model to extract features from the latent space, using β matrices for DNA methylation sites as inputs, these latent features are embedding-like inputs for downstream tasks (eg. hierarchical clustering of cancer subtypes).

Tampuu et al. used a convolutional neural network with two paths to classify raw (one-hot encoded into 5 classes, ATCG and N for unknown nucleotides) metagenomics contigs (300bp) into human or viral DNA [46].

Busia et al. [5] performed 16S classification of environmental samples using depthwise separable convolutions on raw metagenomics contigs, improving the state-of-the-art (which was previously achieved by long denoising pipelines followed by naive Bayes classification [22]) for noisy reads.

Many of the tasks related to bioinformatics revolve around the analysis of sequences. The main ones are DNA and amino acids, which express their functions based on intra and inter-sequence interactions comparable to natural languages' linguistic representation levels. For example, the gene expression mechanism, involves patterns of DNA at various levels:

- Local patterns, eg. the TATA box, which can be considered lexemes,
- Short distance interactions, eg. between promoter and gene, which form syntagms,
- Long distance interactions, eg. between enhancers and promoters, which can be assimilated to semantic units,
- Extra-sequence interactions, eg. histone methylation, which are akin to pragmatic interactions.

This claim of being able to process biological sequences as languages is backed up by Osmanbeyoglu et al. [35] who performed n-gram analysis (a technique used for author ownership classification, language recognition and plagiarism detection in natural languages) on whole proteome sequences of 970 bacteria species and discovered organism-specific signatures. At the genus level (the second lowest taxonomy) even a unigram model is enough to highlight some signatures. Another important result is that perplexity of prediction of an organism B, using the language model of organism A, is proportional to their phylogenetic distance, a phenomenon observed also in evolutionary linguistics.

Liang [27] performed n-gram analysis on DNA sequences split on k-mers (while the work of Osmanbeyoglu et al. considered a single amino-acid as word), measuring the entropy on the language model of 12 eukaryotic organisms (very far away in the phylogenetic tree, like *Schizosaccharomyces spp.* and *H. sapiens*) and observing that the minimum entropy is reliably achieved in every organism with words large 12–15 k-mers.

Ghandi et al. [14] enhanced the k-mer based approach by proposing new language models based on gapped k-mers (words are large k symbols, l of which are irrelevant for the language model, eg. ATNAGCNN is a word with $k = 8, l = 3$), reducing the overfitting risks on SVM classification.

Recent trends in natural language processing have shifted the approaches from a statistical one (eg. n-gram analysis) to a deep learning based one. Important contributions in this direction are the exploitation of sequential informations, by means of recurrent neural networks, and the learning of embeddings, vector representations which can encode semantic relations as distances between points.

Quang et al. [40] trained an end-to-end neural network to quantify properties of (relatively) long DNA (1000bp) sequences. Their approach is based on a convolutional feature extractor which is then classified by a bidirectional long short-term memory layer. Zhang et al. [55] improve this design by using a deeper architecture with residual connections.

Recently Zaheer et al. classified promoter regions and the chromatin profile (919 classes), achieving new state-of-the-art results with their BigBird [54]. Following the modern transformer pipeline to compute contextual embeddings, they performed a two-phase training, first pre-training it on human genome (version GRCh37 [17]) for language understanding and then fine-tuning it on specific datasets for the final prediction. Another important contribution towards the application of "modern" NLP techniques to bioinformatics is their use of SentencePiece [23] tokenization, instead of the traditional k-mer split.

Chapter 4

Methodology

4.1 Architectures

Everyday bioinformatics tasks based on sequences rely on algorithms such as BLAST or Burrow-Wheeler transform [26], because of this, the typical laboratory is equipped with machines with high memory availability and fast CPUs, and very few algorithms actually exploit GPUs. Although clusters can be rented from providers for training and inference, a cheaper alternative, in the long run, could be to equip the laboratory with a medium-end GPU, such as those used for gaming, and perform both training and inference in-house.

In order to conform with this scenario, two transformer architectures are explored and their parameters are set so that training can be performed on modern, but accessible, GPUs. The training process was tested on three gaming laptops of different generations, equipped with nVidia GeForce RTX3070 (8Gb GDDR6, 2021), RTX2080 Max-Q (6Gb GDDR6, 2018) and GTX960M (2Gb GDDR6, 2016), achieving satisfying results in terms of trainability (with the exception of the last one which could train only on very small batches) and inference times (also for the last one).

4.1.1 Longformer

As already mentioned, attention mechanisms are basically (learned) look-up tables, and the cost of letting any token to attend every other one has a memory requirement of $\mathcal{O}(n^2)$, from a linguistic perspective, however, the context required to give meaning to a token (and therefore required to compute its embedding) is rarely composed by the entire sentence [21].

Beltagy et al. [3] propose, with their Longformer architecture, to build a $\mathcal{O}(n)$ attention mechanism which is built around two linguistic assumptions: context for a token can be dependent on its neighbors, or depend on some "important" tokens, regardless of their position inside the sequence. In order to satisfy these two, seemingly incompatible, assumptions, they propose to use the combination of two self-attention mechanisms:

- Windowed self-attention: each value token can be attended only if lies inside a fixed size window (possibly dilated) centered in the key, akin to convolutions, stacking multiple attention heads can yield larger receptive fields,
- Global self-attention: a subset of key tokens is selected (ie. trained) based on the task and only these are capable to attend to any other token.

Since the inputs and outputs of Longformer's self-attention are the same as a traditional $\mathcal{O}(n^2)$ self-attention, the authors propose to plug their attention into any existing transformer architecture, which can be, not only pretrained on any language understanding task (eg. MLM + NSP, MLM + SOP, etc.), but also "bootstrapped" from the weights learned with the original attention and pretrained for fewer epochs.

Zaheer et al. [54] propose a similar architecture, called BigBird, which adds a random self-attention mechanism to windowed and global self-attentions, however this approach is not backed up by linguistic motivations and the performance gain may not be significant [13].

One of the theoretical results achieved by BigBird, however, is that transformers with sparse self-attention are both universal approximators for functions on sequences (like transformers with dense self-attention) and Turing complete.

4.1.2 Albert

After the rise of BERT's popularity, it soon became evident that the architecture is heavy and pretraining is expensive (or even impossible without enough computing power), causing concerns over environmental impact [42] and equity among researchers [13].

Lan et al.[24] propose to modify BERT's architecture and pretraining tasks in order to greatly reduce memory and time requirements, while preserving the same level of performance.

Albert achieves a reduced number of parameter by exploiting two techniques:

- embedding matrix factorization: the input tokens must be converted into their (non-contextual) embeddings before feeding the transformer, however, the embedding matrix is typically sparse (even when using tokenization algorithms like BPE). This leads to inefficient memory usage which can be overcome by splitting the embedding computation in two steps: the one-hot encoding of tokens is embedded into an intermediate vector space and then mapped to the final target space,

- cross layer parameter sharing: blocks in a transformer are divided into groups and each block inside a given group is forced to use the same weights (both for the self-attention and the fully connected layers). This allows to achieve deeper architectures with a fraction of parameters to train and also applies a regularization effect during training.

Compared to BERT, an Albert network with the same layers has about 5% the number of the original parameters and can be trained almost twice as fast, however, to further increase performance, Lan et al. propose to modify the pretraining task by using sentence-order prediction (SOP, already described in section 2.2.3).

In spite of all these optimizations, Albert's attention still grows quadratically with respect to input length, to overcome this limitation, its attention mechanism can be replaced with Longformer's to achieve a further reduction in model size. For clarity's sake, in the rest of the thesis this hybrid architecture will be called "Longalbert".

4.1.3 Models and hyperparameters

In order to allow training on limited hardware, two candidate models are proposed:

- Longformer:
  - 2 blocks,
  - 8 self-attention heads,
  - 256 maximum input tokens,
  - 16 token attention window size,
  - 256-dimensional output embedding.
- Longalbert:
  - 2 blocks (1 group),
  - 8 self-attention heads,
  - 256 maximum input tokens,
  - 16 token attention window size,
  - 256-dimensional output embedding,
  - 16384-dimensional intermediate input embedding.

The losses and metrics which will used for each of the experiments are briefly defined in appendix A.

In appendix B, additional experiments with larger networks are carried as well.

4.2 Experiments

4.2.1 Language model pretraining

In order to better train the networks in downstream fine-tuning tasks, the proposed models are trained on two different "languages":

- Mitochondrial genome: characterized by a single circular chromosome of limited size (about 15kbp) and with simple regulatory sequences,
- Human nuclear genome: characterized by 22 autosomes and 2 sexual chromosomes, all linear and with a large size (for a total of about 3.1Gbp) and complex regulatory structures.

The mitochondrial language is trained on reference genomes of various species extracted from NCBI's Organelle database [34], while the nuclear language is trained on the latest human reference genome (GRCh38 using the NCBI nomenclature, or hg38 according to the UCSC nomenclature) [18], note that BigBird was pretrained on the previous version (GRCh37/hg19).

There is no natural word-boundary in either of these languages, to solve this problem a traditional k-mer split can be performed. Since the choice of k determines the size of the tokenized documents (which reflects to larger tensors to fit on GPU), but also the perplexity of the learned language model [35], a good compromise must be chosen.

An alternative approach to word-splitting is the one proposed by Zaheer et al. for BigBird [54], which is summarized by algorithm 1.

It's important to notice that either approach has its own drawbacks:

```
words = [];
foreach chromosome do
    for repeats in 0..10 do
        offset = 0;
        start = random(0, 500);
        while start $\leq$ chromosome.length do
            end = min(chromosome.length - start + i, random(0,
              10000));
            sentence = chromosome[start: end];
            i += end;
            j = 0;
            while j < sentence.length do
                old_j = j;
                j += random(50, 500);
                words.append(sentence[old_j: j]);
            end
        end
    end
end
return words;
```

Algorithm 1: Big Bird word splitting algorithm.

- k-mer: every input is aligned in the same way, so there may be tokens (or sequences) not represented in the training set simply because they were misaligned during inference, and the dataset is traversed only once per epoch,

- Big Bird: the entire dataset is traversed multiple times, increasing the memory requirements, and it's not applicable in fine-tuning, causing an asymmetry of conditions observed by the network.

Due to time and memory requirements, a limited number of mitochondrial chromosomes were extracted from the database (which contains a total of about 12 thousands genomes) and processed with various combinations of methods. Finally, train-validation-test splits are performed by extracting a random 10% of the sequences as test set and a further 10% of the remainder as validation set.

- 6-mer: 2000 genomes (1620 train, 180 validation, 200 test),

- 9-mer: 3000 genomes (2430 train, 270 validation, 300 test),
- 12-mer: 4000 genomes (3240 train, 360 validation, 400 test),
- Big Bird-style: 2000 genomes, resampled 10 times (16200 train, 1800 validation, 2000 test).

After either of these word-splitting procedures, BPE is applied to produce tokens which can then be further processed to build the dataset for language understanding tasks. The 6-mer dataset was tokenized with a target vocabulary size of 5000 tokens, which is larger than all the possible combinations ($4^6 = 4096$) and therefore leads to a single token for each 6-mer. 9-mer, 12-mer and Big Bird-style words were tokenized with a target size of 30000 tokens (similar to typical vocabularies used in English and other natural languages).

After tokenization, datasets for two pretraining goals are created:

- Masked language modeling only: the default BERT parameters are used (ie. 15% words masked, 80% of which replaced by the `<mask>` token, 10% of which replaced with a random token and the remainder unchanged),
- MLM and sentence order prediction: a full length MLM sample is prepared, then it's eroded by a random number of tokens on both sides and cut in a random position in the middle, finally the two pieces are swapped with probability 50% as it can be seen in algorithm 2.

Combinations of tokenization approaches and pretraining tasks were tested both on the proposed Longformer and Longalbert models, using the Adam optimizer with parameters: $lr = 10^{-4}, \beta_1 = 0.9, \beta_2 = 0.999$.

Due to time limitations, only the best performing approach identified on mitochondrial genome was applied to nuclear genome. As it will be shown in section 5.1, this setup corresponds to 9-mer tokenization, MLM only training task and Longformer.

```
a = random(1, sample.length / 10);
c = random(sample.length * 9 / 10, sample.length - 1);
b = random(a + sample.length / 10, c - sample.length / 10);
first = sample[a: b];
second = sample[b: c];
swap = random(0, 1);
if swap then
    return (“<cls>” + second + “<sep>” + first + “<sep>”, 1);
else
    return (“<cls>” + first + “<sep>” + second + “<sep>”, 0);
end
```

Algorithm 2: SOP sample preparation algorithm.

4.2.2 Transcription start site localization

As mentioned in section 2.1.3, a promoter always has in its core region a specific position called transcription start site (TSS), which also corresponds to the beginning of the gene regulated by the promoter.

From the Eukaryotic Promoter Database[10], promoter sequences ranging from -4999 to +5000 (with position 0 being the TSS) were downloaded for the following model organisms:

- *Homo sapiens*,
- *Mus musculus*,
- *Gallus gallus*,
- *Danio rerio*,
- *Drosophila melanogaster*,
- *Caenorhabditis elegans*,
- *Zea mays*,
- *Saccharomyces cerevisiae*,
- *Plasmodium falciparum*.

These organisms are some of the most studied in biology and their genomes can be considered relatively well described, moreover they were selected in order to have an increasing evolutionary distance from *H. sapiens*, so that potential correlations between neural network performance and evolutionary distance may be detected.

Each of the 10 thousands bp sequences is split in two at the TSS and each is tokenized separately (using the BPE tokenizer trained on 9-mers), this procedure guarantees that the TSS will always be placed at the beginning of a token and makes dataset preparation easier, but can be a source of discrepancies between training and inference.

The input sequence is built by choosing a random number between 0 and 200 tokens from the sequence before the TSS and the rest from the one after, up to 254 tokens (the first and last are reserved for `<cls>` and `<sep>` respectively, for a total of 256 tokens). The output prediction is the location of the TSS, ie. the chosen random number.

The *H. sapiens* dataset is split according to the following proportions:

- Training set: 81%,
- Validation set: 9%,
- Test set: 10%.

Every other organism is kept unsplit (along with a full version of the *H. sapiens* dataset) for the "evolutionary" experiments.

The learning task is approached as a token-level classification and the goal is to predict the probability for each token to be the TSS. This means that the transformer is equipped with a classification head of 256 neurons, the output of which is normalized using a softmax, and trained with a crossentropy loss. To mitigate overfitting, between Longformer and the classification head, a dropout layer is added.

This architecture is trained only on the human dataset, varying the dropout rate (0%, 10% and 20%) and the type of pretraining Longformer was subject to (human reference genome hg38, 3000 mitochondrial genomes, or no pre-training), to assess whether these parameters affected performance (measured simply as exact accuracy on the prediction).

Training is performed for 10 epochs, monitoring performance on the validation set to check for possible overfitting, and finally accuracy is determined on the human test set.

A further experiment is performed by evaluating accuracy on every model organism to see if evolutionary distance from the organism the network was trained on, plays a role in performance or whether promoter sequences can be considered universal among eukaryotes. This latter hypothesis is backed up by biological facts and, as it will be shown in section 5.2, seems to be the case.

4.2.3 Metagenomics classification

12S metagenomics exploits the 12S rRNA gene (about 1000 bp) of mitochondria to classify the eukaryotic organism they were extracted from. MitoFish [19] is a curated database hosting the mitochondrial sequences of more than 3000 fish species, along with highly standardized annotations for each of them.

For each sample, the 12S rRNA gene is extracted from the full mitochondrial genome, and the dataset is augmented inserting 31 corrupted versions of the same sample (for a total of 32 samples for each species). The mutations are inserted at the character level by choosing a random substitution rate between 0.5% and 5% and randomly flipping base pairs accordingly. This process simulates the sequencing error and therefore each substitution is assumed equally likely (unlike natural mutations for which some substitutions are more likely than others).

After augmentation, each sample is processed by the BPE tokenizer (trained on 9-mers), 254 tokens are extracted from the beginning of the sentence and padded with `<cls>` and `<sep>`. Since with 9-mer tokenization there are between 250 and 260 tokens for the entire gene, discarding the last tokens, in case the entire sequence can't fit the input, shouldn't cause a significant drop in classification performance.

The dataset is finally shuffled and split according to the following proportions:

- Training set: 81%,
- Validation set: 9%,
- Test set: 10%.

As a result, there might be the chance of having classes not appearing in the training set, if all the 32 samples of a given class happen to be split in validation and test sets only. This phenomenon should be unlikely and so it shouldn't introduce biases in performance evaluation.

The learning task is a sequence-level single-label classification, and therefore the transformer is equipped with a classification head possessing one neuron for each of the species in the original MitoFish database (ie. all classes still have a specific neuron in the architecture, regardless of the split they end up being into) and connected to the embeddings of the `<cls>` token alone. The output of the final layer is normalized with a softmax activation and training is performed by optimizing a crossentropy loss.

It's important to note that a proper metagenomics classifier should be further assessed on some additional problematics:

- handling confidence of predictions and unseen classes during training,
- comparing performance against state-of-the-art classifiers on mock communities (ie. artificial samples with known composition),
- assessing performance on real samples (especially in terms of robustness against sequencing errors).

Since these problematics are not directly related to the classification architecture, they are not treated in this thesis.

4.2.4 Chromatin profile prediction

The dataset of Zhou et al. introduced in section 2.1.5 contains more than four million sequences associated with one or more of the 919 chromatin profiles identified in previous experiments. Due to memory limits, only the first 200 thousands samples are extracted from the dataset.

Since the original work exploited convolutional neural networks [56], input sequences in the dataset are one-hot encoded into a 4-dimensional vector for each nucleotide. This encoding is reverted to a string which is then fed to the BPE tokenizer (trained on 9-mers) to produce a sequence of tokens. Of these tokens, the first 254 are extracted and padded by `<cls>` and `<sep>`. Further tests are performed with 512 and 1024 input tokens to verify BigBird's claim of longer sequences being beneficial for performance.

The learning task is approached as a sequence-level multi-label classification. The transformer is therefore equipped with a classification head of 919 neurons using sigmoid activation and trained on binary crossentropy loss. In order to handle class imbalance, the loss for positive samples is upweighted by a factor of 8, as proposed by Zaheer et al. [54].

Previous experiments on promoter localization demonstrated that dropout has minimal effect on performance (see section 5.2), so it was not introduced in this experiment.

Training is performed for 10 epochs, using the following splits:

- Training set: 81%,
- Validation set: 9%,
- Test set: 10%.

Chapter 5

Results

5.1 Pretraining on reference genomes

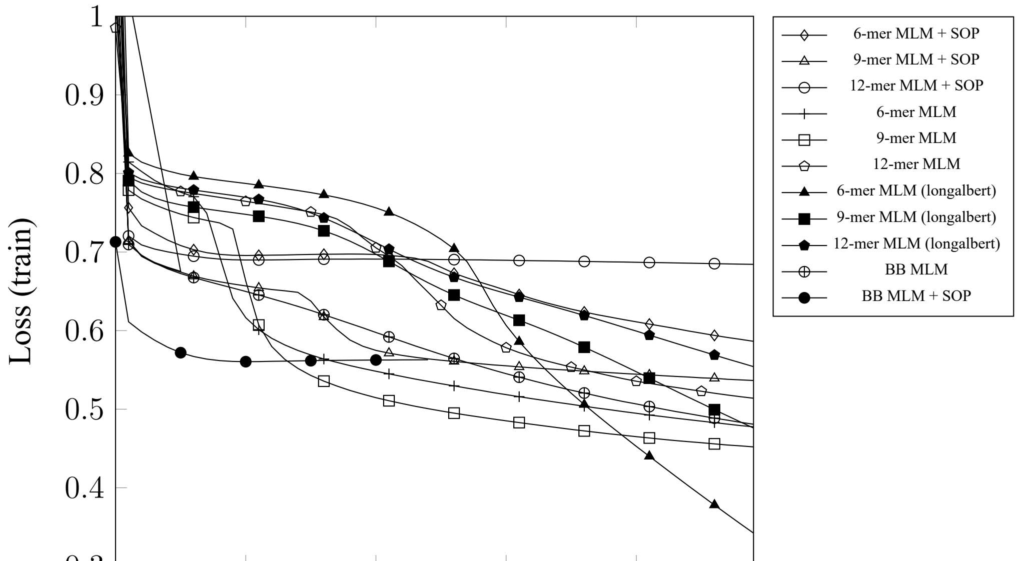

Figure 5.1: Effect of different tokenizations and language modeling tasks on losses (mitochondrial dataset pretraining).

Figures 5.1 and 5.2 compare the training performance of every transformer trained on the mitochondrial dataset. From these, the following considerations can be made:

Mitochondrial language

Model	Tokenizer	Tasks	Accuracy	Perplexity
Longalbert	6-mer	MLM	0.8721	2.8765
Longalbert	9-mer	MLM	0.8652	3.1472
Longalbert	12-mer	MLM	0.8649	5.6703
Longformer	6-mer	MLM	0.8892	1.8346
Longformer	9-mer	MLM	0.8875	**1.7257**
Longformer	12-mer	MLM	0.8856	**1.7797**
Longformer	BB	MLM	**0.9054**	**1.6191**
Longformer	6-mer	MLM SOP	**0.8966** **0.5164**	1.8151
Longformer	9-mer	MLM SOP	0.8948 0.5081	1.7398
Longformer	12-mer	MLM SOP	0.8844 **0.5090**	2.0981
Longformer	BB	MLM SOP	**0.9025** **0.5400**	1.7543

Hg38 language

Model	Tokenizer	Tasks	Accuracy	Perplexity
Longalbert (20 epochs)	12-mer	MLM SOP	**0.8846** 0.4972	2.46
Longformer	9-mer	MLM	0.8785	**2.2363**

Reference values

Model	Tokenizer	Tasks	Accuracy	Perplexity
BigBird (SOTA)	BB	MLM	NA	2.17
Random (baseline)	Any	MLM SOP	0.85 0.5	NA

Table 5.1: Pretraining performance on validation sets (best 3 in bold).

- BigBird-style tokenization converges in fewer epochs and plateaus at a higher accuracy compared to k-mer tokenization, but requires a much longer training time,
- Longalbert networks apparently perform better on the training set, however perplexities on the validation set (figure 5.3) show that this is a consequence of heavy overfitting,
- SOP task improves MLM accuracy quickly in the first few epochs, but then limits it to a plateau which is lower than the one obtained from MLM training alone, hinting to an interaction between the embeddings

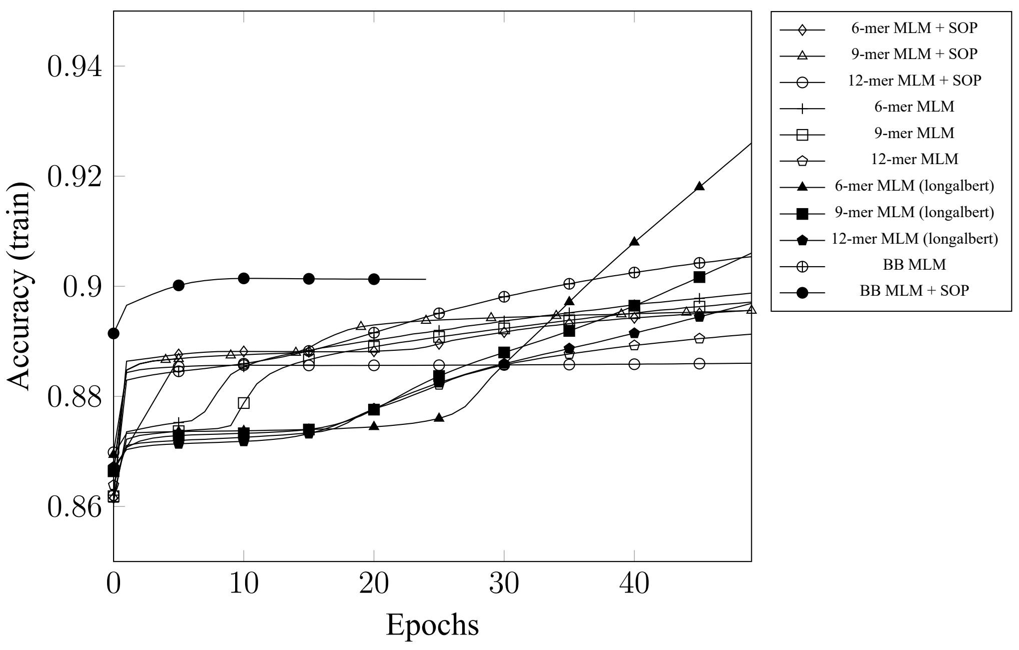

Figure 5.2: Effect of different tokenizations and language modeling tasks on MLM accuracies (mitochondrial dataset pretraining).

learned for the two goals,

- The value of k in k-mer tokenization before BPE affects performance, with 9-mer tokenization being the best of the three tested values (6, 9 and 12).

With 6-mer tokenization of a 4-character alphabet there are only $4^6 = 4096$ tokens, and therefore byte pair encoding with a typical target vocabulary size is basically the identity function, while both 9 and 12 k-mers produce a number of possible tokens which is larger than the target vocabulary size and BPE will split them in order to have better statistical representativeness. As a result, 6-mer tokenization has, not only a reduced input size (256 tokens of 6 bp hold a sequence shorter than 256 tokens of 9 or 12 bp, even after BPE), but also a reduced vocabulary, where each token is potentially reused in a higher number of contexts, increasing ambiguity.

The fact that 9-mer tokenization performs better than 12-mer tokenization, contrary to what could be hypothesized by the work of Liang et al. [27], can

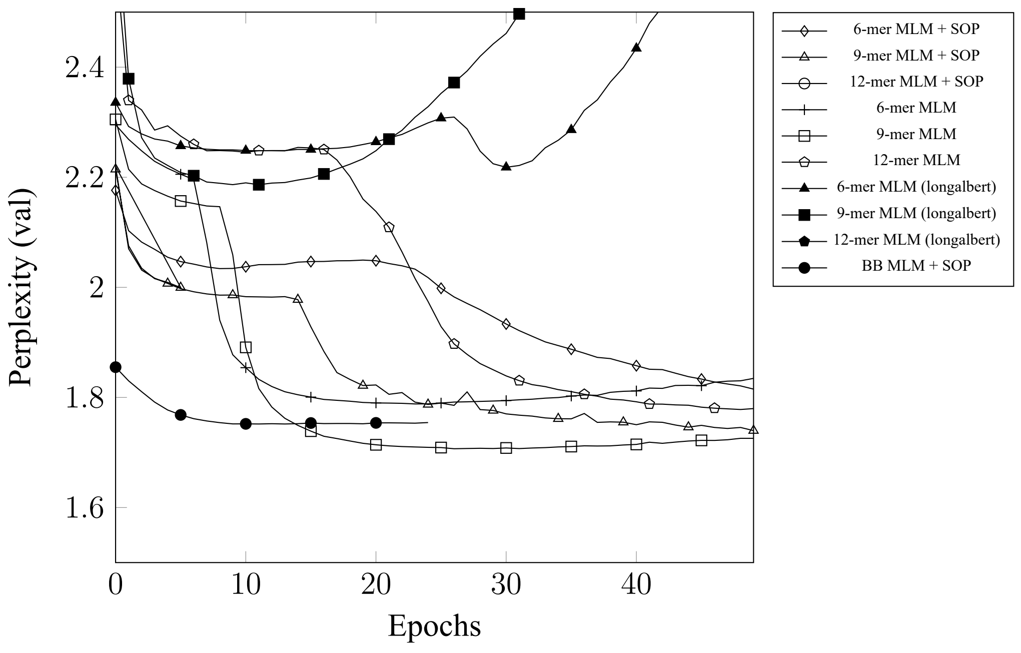

Figure 5.3: Effect of different tokenizations and language modeling tasks on perplexities (mitochondrial dataset pretraining).

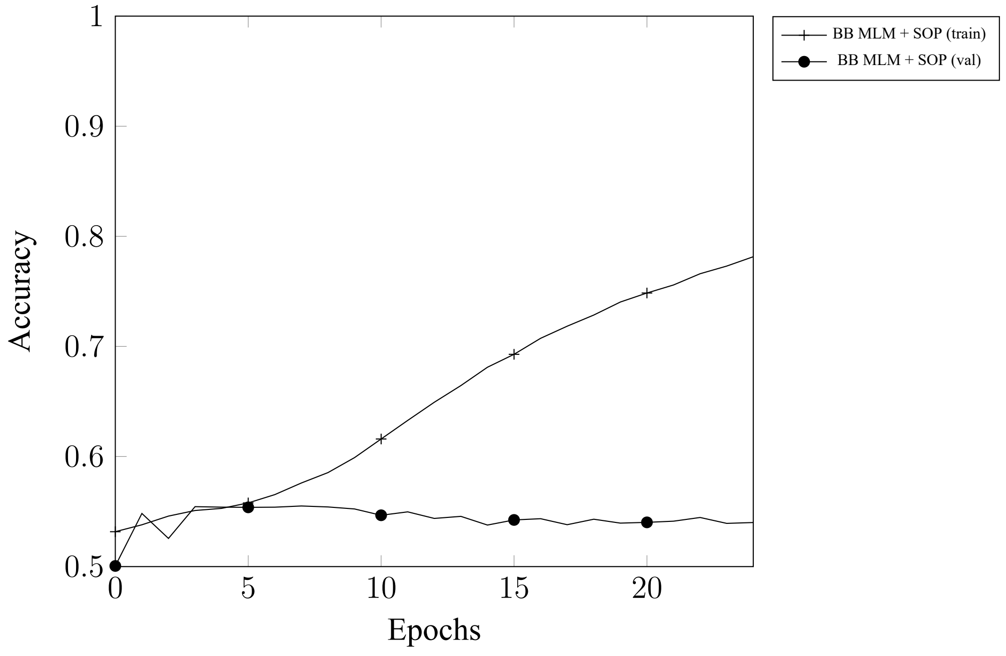

Figure 5.4: Overfitting on SOP accuracy on mitochondrial dataset pretraining (Longformer).

be justified by the fact that BPE alters the probability distribution of each token compared to simple k-mer tokenization, and therefore it's not possible to predict the entropy/perplexity trend after BPE tokenization.

Longalbert's tendency of overfitting can be ascribed to the excessive reduction of parameters in the transformer, caused by the interplay of Albert's shared weights (also for the attention modules) and the use of Longformer's windowed attention. An alternative explanation could be the interaction between Albert's embedding matrix factorization and Longformer's global self-attention training.

Human genome (hg38) training achieved performances similar to BigBird's, in spite of using only 256 input tokens, instead of 4096. Due to the limited amount of time and resources, only two representative cases were tested (table 5.1):

- Longalbert with 12-mer tokenization trained on MLM and SOP achieves a slightly higher accuracy, but also higher perplexity,
- Longformer with 9-mer tokenization trained on MLM only achieves better perplexity, but still higher than its counterpart trained on mitochondrial genomes.

In spite of the low perplexities achieved, qualitative assessment of masked language modeling and sentence order prediction demonstrated that no architecture managed to effectively learn the tasks (since perplexity on mitochondrial genomes is sensibly lower than BigBird's and still can't reliably predict masked tokens, it can be postulated that neither Zaheer et al. achieved good MLM performance). This apparently counterintuitive result can be explained by considering that MLM masks only 15% of the tokens and therefore the perplexity is an overestimate of real performance (since for 85% of the tokens the task is simply to copy them from input to output).

Moreover DNA sequences are not simply languages, but rather meta-languages, in which context can quickly switch from one sub-language to another (eg.

from regulatory DNA to coding DNA as soon as the transcription start site is encountered and from coding DNA to non-coding DNA after a stop codon). These contexts can potentially span hundreds of thousands of base pairs and cannot be fully captured by a transformer (not even a 4096-tokens BigBird) and intra-sequence clues may not be enough for the transformers to “identify” them and learn embeddings optimized for each of them.

Additionally, each of these sub-languages has a heavily imbalanced representativeness inside genomes, for example a gene has one or few representatives inside the entire genome, while a promoter has thousands of them. This observation is coherent with the fact that language modeling on mitochondrial genomes is way more robust (lower perplexity, but still unable to qualitatively predict the correct token) than modeling on human genome, since in the first case there is a small language (15 kbp) instantiated for few thousands organisms, increasing the representativeness of each “sentence” in all the sub-languages contained, while in the second case there is a large language (3.1 Gbp) instantiated for a single organism.

In either case, though, subsequent experiments proved that masked language modeling pretraining is beneficial (although not crucial) for downstream tasks.

The genomic meta-language is also responsible for the total failure of sentence order prediction, with heavy overfitting (as can be seen from figure 5.4) and performances identical to random guessing (table 5.1).

The additional experiments carried in appendix B improved both perplexity and qualitative performance, showing that more complex models can better capture the underlying language, at least for simple genomes such as the mitochondrial one.

5.2 Fine tuning on transcription start site localization

Species	Description	Accuracy
H. sapiens	Humans (baseline)	0.9632
M. musculus	Field mouse	0.9649
G. gallus	Chicken	0.9611
D. rerio	Zebrafish	0.9644
D. melanogaster	Fruit fly	0.9671
C. elegans	Worm	0.9656
Z. mays	Mais plant	0.9629
S. cerevisiae	Yeast	0.9630
P. falciparum	Unicellular eukaryote	0.9658

Table 5.2: Promoter localization performance on different model organisms after training on the *H. sapiens* promoters only (Longformer, 9-mer, 0.1 dropout, Hg38 pretraining, sorted by evolutionary distance from *H. sapiens*).

As previously mentioned, the task of promoter region prediction as performed by Oubounyt et al. [36] and Zaheer et al. [54] is extremely simple and with a single experiment (not described in this thesis) their results has been consistently reproduced, even when using 256 tokens instead of 4096. The increase of performance achieved by BigBird was not achieved, as they claimed, thanks to a longer context, but instead by an increase of expressivity of the overall architecture (a transformer is way more expressive than a convolutional neural network, especially if the latter has very few layers).

The slightly more interesting task of promoter localization was meant to increase complexity and overcome the bias introduced when building the negative samples, however, from figures 5.5, 5.6 and 5.7 it can be noted how near-perfect memorization of the training set is possible, no matter which weights are used to initialize the transformer or which dropout rate is chosen, while a slight drop in performance is seen on the validation set when using the uninitialized network, hinting to a lower generalization capability.

The networks pretrained with MLM on human and mitochondrial genomes don’t present substantial differences, this phenomenon apparently surprising

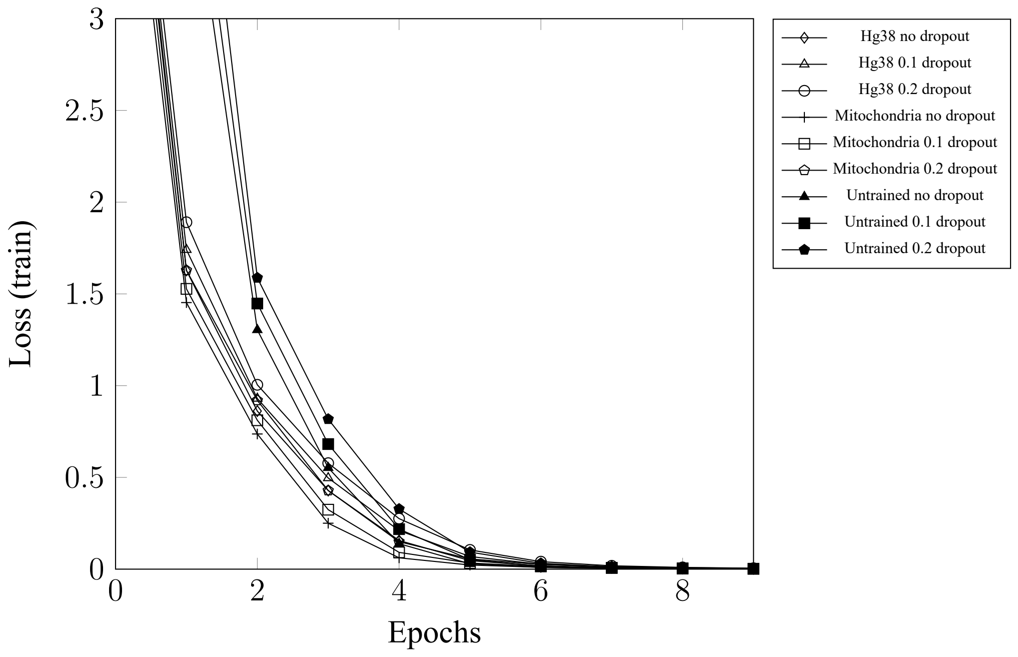

Figure 5.5: Losses on promoter localization.

(mitochondria have a different kind of promoter regions, so sequences of tokens resembling a nuclear promoter were not seen during MLM), is actually explainable by considering that token embeddings learned during pretraining have some metric properties, and so are better fit to capture "similarity" compared to an uninitialized network. This gain however is too small to justify pretraining efforts (especially since MLM pretraining requires days of computations, while promoter localization fine tuning can be performed in few hours).

Final performance was assessed on all the promoters available for each of the chosen model organisms (as such, for *H. sapiens*, training and validation sequences were also considered, giving an overestimate of accuracy) and it's summarized on table 5.2.

It can be noted that, in spite of evolutionary distance from the organism used for training (*H. sapiens*), prediction of promoters in every species achieves the same score. This is coherent with prior biological, linguistical and dataset knowledge about the task:

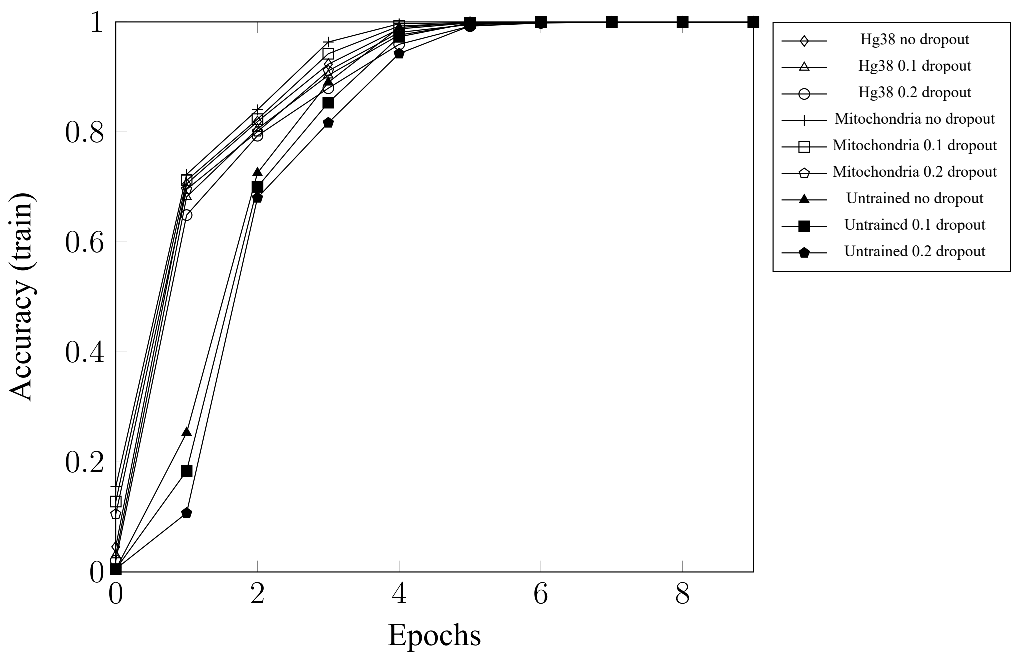

Figure 5.6: Training accuracies on promoter localization.

- biologically, every eukaryote evolved from a shared ancestor, as such the biological mechanisms are shared as well. Even though evolution plays a role in introducing modifications, mechanisms crucial for survival such as promoters tend to remain pretty much unchanged. Furthermore, since training was performed on an evolutionarily recent organism (*H. sapiens*) and then prediction on more ancestral organisms (up to *P. falciparum*), it can be postulated that the transformer is "aware", not only of the ancestral patterns, but also every other one which may have evolved later.

- Linguistically, a promoter is a simple pattern which can be identified by extremely simple tools (eg. regular expressions), therefore a complex architecture like a transformer has little to no difficulty in identifying one, even when matching is not exact (eg. because it's coming from a different species than the one seen during training).

- Regarding data, it's important to note that the Eukaryote Promoter Database

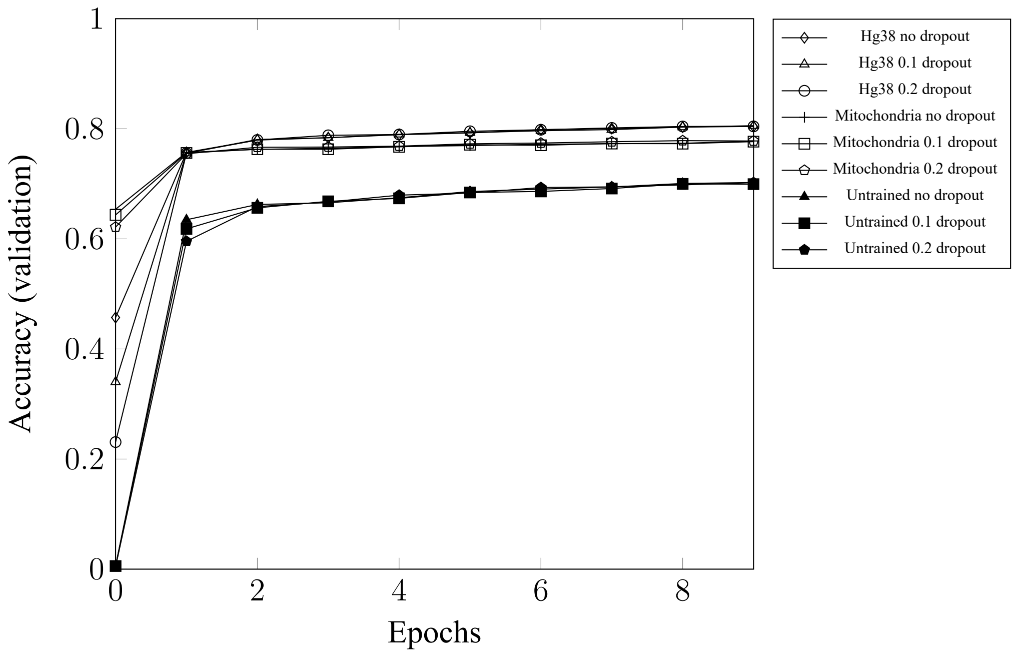

Figure 5.7: Validation accuracies on promoter localization.

focuses mostly on humans, providing way fewer samples for other species. In spite of this imbalance, consistency in performance and a lack of bias hint that all samples have a similar composition, no matter which organism they were taken from.

The experiments in appendix B achieve a slightly lower result, hinting a slight degree of overfitting and further confirming how simple the task of promoter localization is.

5.3 Fine tuning on metagenomics

Pretraining	Loss	Top-1 Accuracy	Top-5 Accuracy
Mitochondria	0.4082	0.8576	0.9851
Untrained	1.1384	0.7008	0.9053
Hg38	1.2261	0.6415	0.9194

Table 5.3: 12S mitochondrial metagenomics performance on test sets (Longformer, 9-mer).

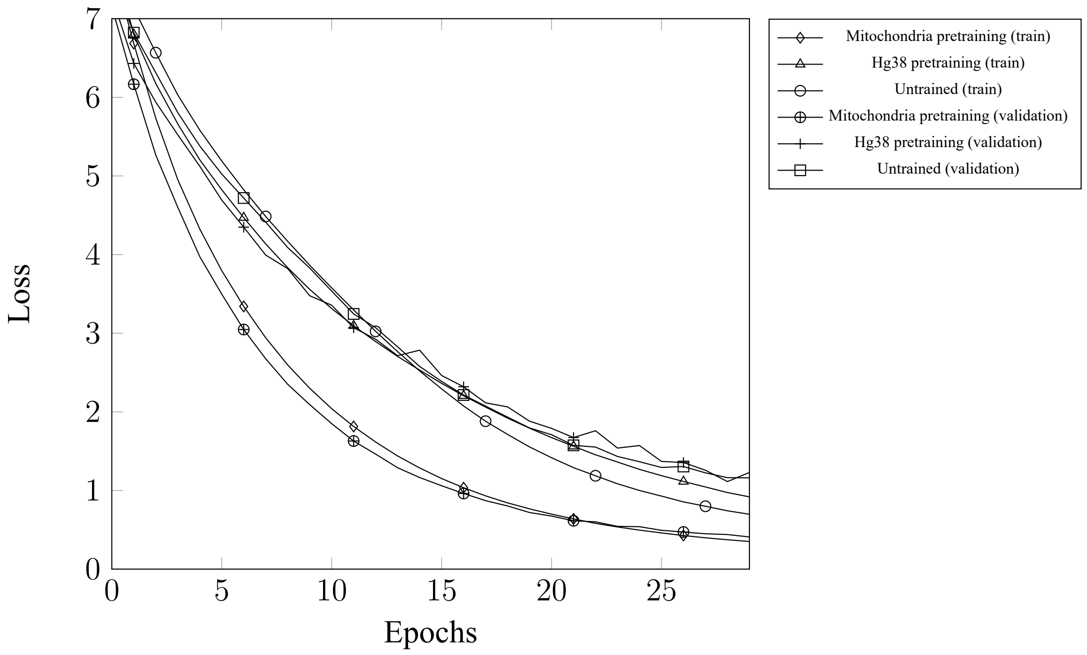

Figure 5.8: Losses on fish metagenomics.

Figures 5.8, 5.9 and 5.10 summarize the training performances in the 12S metagenomics task and table 5.3 presents the final results on the test set.

It can be clearly seen that pretraining on the mitochondrial language provides a sharp improvement, compared to the other two initialization methods. Surprisingly the uninitialized transformer tends to perform slightly better than the one pretrained on the human genome. This peculiar trend (opposite of the one seen with promoter localization) can be explained by the fact that MLM pretraining on the human genome is more difficult and therefore the learned embeddings are, not only less “robust”, but possibly also specialized towards features specific of the human genome which need to be “unlearned” before learning those relevant for metagenomics classification. This hypothesis is even more plausible when considering the connectivity chosen for the transformer: since the classification head is connected only to the embeddings of the `<cls>` token, positional information is lost unless captured by the embedding of `<cls>`. Having achieved a stronger language model, the mitochondrial MLM pretraining is more likely to have captured these informations,

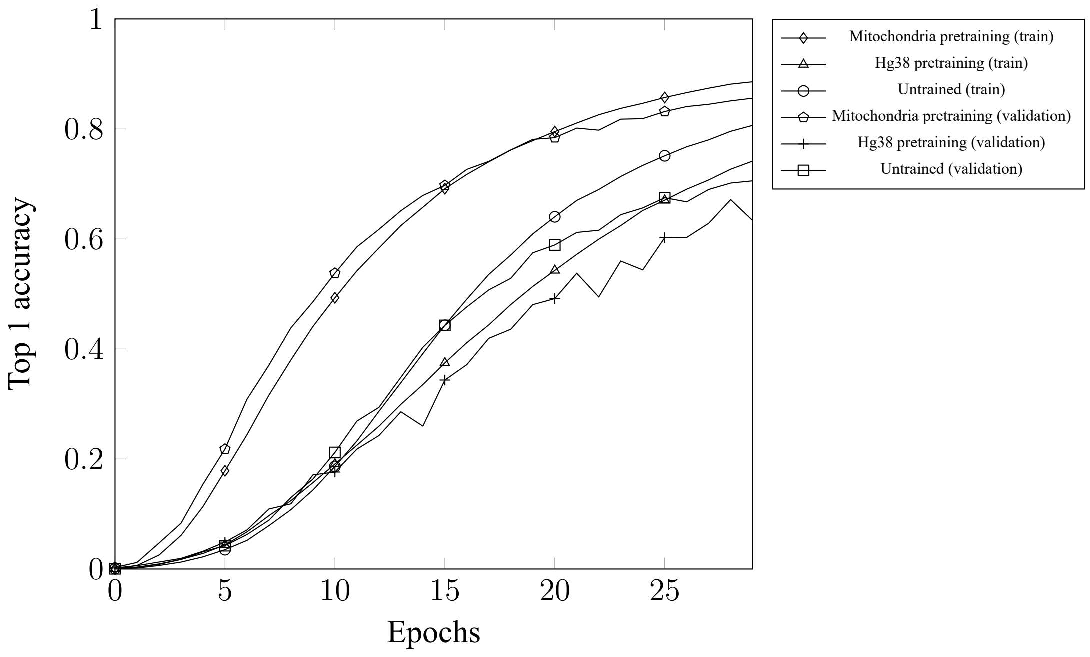

Figure 5.9: Top 1 accuracies on fish metagenomics.

compared to the other two cases.

Top-5 accuracy reaches very good results in all cases, showing that, even if a sample is misclassified, the correct class has almost always a high predicted score.

Taxonomic classification of organisms is hierarchical (the taxa are, from most general to most specific: domain, kingdom, phylum, class, order, family, genus and species) and, even when making mistakes, a good predictor should preserve high accuracy on higher taxonomical levels (ie. it should misclassify something with a “related” entity).

Figures 5.11, 5.12 and 5.13 present three cases (one for each experiment) in which the transformer made an incorrect prediction. Although inconclusive (not all wrong predictions were analyzed), the mitochondrial pretraining tends to misclassify at a lower taxonomical level (mistakes are made below the order level) than the other methods (mistakes are made below the class level), and is therefore better.

It’s important, however, to note that, although there isn’t a general rule

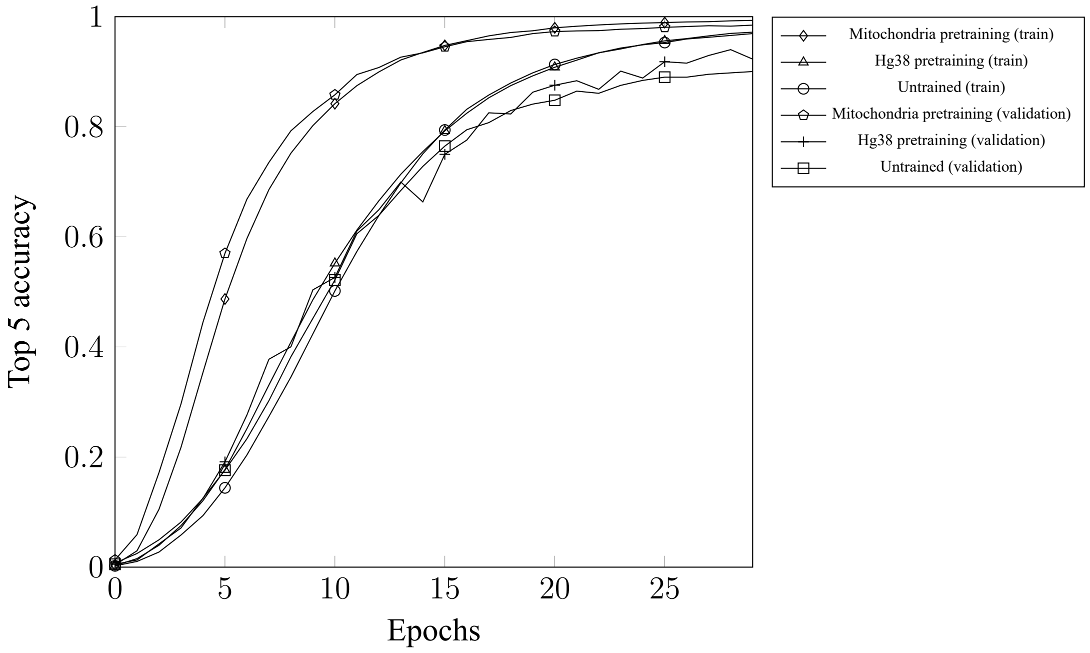

Figure 5.10: Top 5 accuracies on fish metagenomics.

```
Query: <cls> AAAG G CTT GG TCC [Omissis]
True: 2228, Pred: 316
Species: True: Rhynchocypris percnurus, Pred: Bhavania australis
```

Figure 5.11: Wrong metagenomics prediction (mitochondria pretraining). Both fishes belong to the order (*Cypriniformes*).

```
Query: <cls> ACC AGCC TGGTCC [Omissis]
True: 1617, Pred: 2279
Species: True: Nematobrycon palmeri, Pred: Sarda sarda
```

Figure 5.12: Wrong metagenomics prediction (Hg38 pretraining). Both fishes belong to the class (*Actinopterygii*).

```
Query: <cls> AAAGGC ATGG TCCC [Omissis]
True: 2512, Pred: 1193
Species: True: Sinocyclocheilus rhinocerous, Pred: Hyporthodus septemfasciatus
```

Figure 5.13: Wrong metagenomics prediction (untrained). Both fishes belong to the class (*Actinopterygii*).

of thumb for 12S metagenomics, for 16S metagenomics (its analogous for bacterial classification) a good predictor should achieve 100% top-1 accuracy at least at the family level.

The experiments of appendix B greatly improve performance and therefore a larger model, oretrained on the correct genome, can be considered beneficial for metagenomics.

5.4 Fine tuning on chromatin profile prediction

Pretraining	Loss	Precision	Recall	AUC all	AUC DNase	AUC TF	AUC Histone
Hg38	**0.4678**	0.1756	**0.3776**	**0.7161**	0.8570	0.9218	0.8489
Mitochondria	0.6474	0.1883	0.2719	0.6495	0.8604	0.9199	0.8431
Untrained 256 token	0.7042	0.1785	0.2730	0.6251	0.8759	0.9211	0.8563
Untrained 512 token	0.7336	**0.1999**	0.2270	0.6172	0.8833	0.9230	0.8649
Untrained 1024 token	0.7722	0.1994	0.2257	0.6130	0.8918	0.9264	0.8728
DeepSea Baseline					**0.923**	0.958	0.856
BigBird 4096 token					0.921	**0.961**	**0.887**

Table 5.4: Chromatin profile prediction performance on test sets (Longformer, 9-mer).

The results of chromatin profile prediction are shown on figures 5.14 and 5.15, and table 5.4.

It can be noted that weight initialization is irrelevant for global area under curve (AUC), with the exception of pretraining in the human language model which provides a slightly higher performance on validation. However, when AUC evaluation is split in three categories (as proposed by Zhou et al. [56]), chromatin profiles sensitive to DNAse I, chromatin profiles binding specific transcription factors and chromatin profiles silenced by histone methylation, different trends arise:

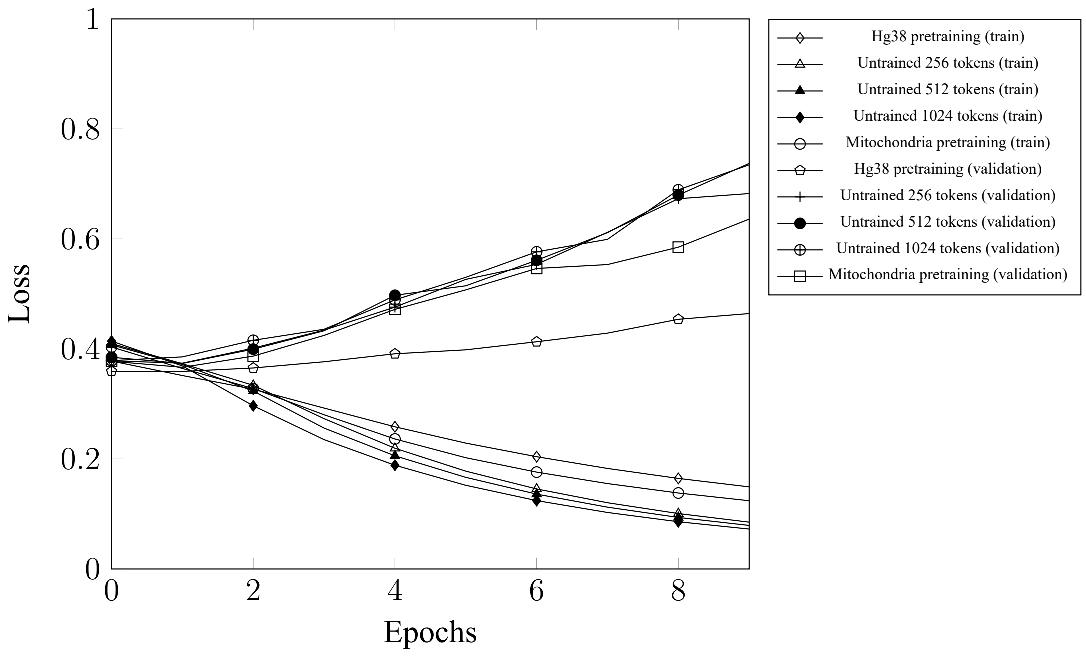

Figure 5.14: Losses on chromatin profile prediction.

- For DNAse I sensitivity and histone methylation sites, the uninitialized transformers perform better than the pretrained ones. Moreover longer input sequences prove to be beneficial for AUC.
- For the transcription factor profiles, sequence length doesn't affect performance significantly and every initialization achieves roughly the same performances.

The fact that input length affects performance only for DNAse I sensisitivity and histone methylation has a biological justification. DNAse I is an enzyme capable of cutting DNA only in a region which is fully uncoiled and forming a portion long enough to allow the enzyme to fit. On the other hand, histones compress long portions of DNA into compact structures (each nucleosome, ie. a histone wrapped with DNA, coils 147 bp and multiple of them interact to further compress the sequence during some types of methylation or uncoil for others).

This correlation between input length and performance, however, is greatly

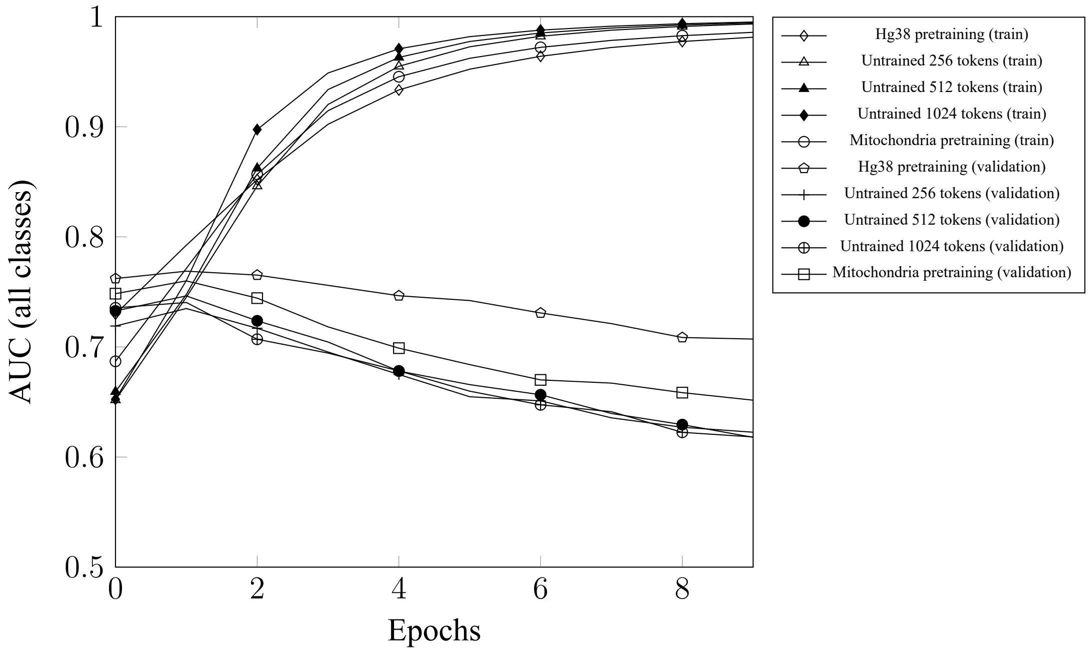

Figure 5.15: Area under curve on chromatin profile prediction.

outweighed by other learnable features, as demonstrated by the experiments carried in appendix B, where a larger model can easily improve state-of-the-art, even when using the "wrong" pretraining (on the mitochondrial language).

As in the case of promoter localization, pretraining efforts are not justified as they do not provide substantial gains (unless AUC is considered for all classes).

The low precision and recall (not reported by the baselines) hint the fact that upscaling the loss for positive samples as proposed by Zaheer et al. did not adequately solve the class imbalance problem.

It's important to note also that the size of the training set was sensibly lower than the one used to train BigBird (about 20 times smaller), and so the decrease in performance with respect to the baselines may also partially be ascribed to this.

Although complex enough to give room for improvements and with a large dataset to leverage for training, the task of chromatin profile prediction mixes classes coming from three extremely different categories (asking in practice to learn three unrelated tasks at once) and has limited practical use (chromatin profile depends on epigenetics, not only sequence informations, so different cell types will have different profiles for the same sequence. A more useful task, exploiting the same datasets, would be to predict a single profile given the sequence and the cell type). For these reason, it can't be considered a good benchmark for bioinformatics architectures, just like promoter prediction or localization.

Chapter 6

Conclusions

6.1 Final remarks

This thesis explored the applicability of small transformers to bioinformatics, achieving good results with a fraction of the size of state-of-the-art architectures. Moreover, by relaxing network size constraints, the same transformers could improve state-of-the-art, while keeping a reduced input size (appendix B).

Both traditional k-mer tokenization, combined with byte-pair encoding, and BigBird-style randomized tokenization, also followed by BPE, are effective representations for the input. Although achieving slightly better results, BigBird-style tokenization is computationally expensive and therefore its use is not justified in practice.

This work explored four tasks under multiple facets, achieving good performance (in spite of having a fraction of the parameters of state-of-the-art architectures such as BigBird) and highlighting problems intrinsic in each task.

More in detail, a language understanding task is trained both using masked language modeling (RoBERTa-style pretraining) and masked language modeling combined with sequence-order prediction (Albert-style pretraining) on two languages: the full human reference genome (build hg38) and the full mitochondrial reference genomes of thousands of eukaryotes. Multiple values for k-mer tokenization were explored and Longformer with 9-mer tokenization, 256 input tokens and MLM-only training achieved best performance (reaching near state-of-the-art perplexity for hg38, obtained by BigBird with an input size of 4096 tokens, and a new baseline for mitochondrial genomes). In spite of good numerical results, qualitative analysis showed unsatisfying results (justified by the intrinsic complexity of the genomic languages).

Promoter region prediction, in the formulation used as a benchmark by BigBird, is ill defined and a more complex task is proposed in the form of transcription start site (the boundary between a promoter and a gene) localization. Extremely good performances are achieved even without pretraining, hinting the fact that promoter localization is not a good benchmark either. Furthermore cross-organism analysis showed that TSS can be reliably localized even on organisms not seen during training.

12S metagenomics is explored by expoiting a database of fish mitochondrial genomes, achieving very good top-5 accuracy and acceptable top-1 accuracy scores (baselines are available for the similar task of 16S metagenomics, but are uncomparable due to the use of different datasets). Using the mitochondrial language model learned during pretraining boosted performance significantly with respect to the untrained Longformer or the one initialized with the hg38 language model (also showing a tendency to make mistakes at a lower taxonomical level, suggesting the achievement of a more solid predictor when the network is initialized with the mitochondrial language model).

The final task of chromatin profile prediction is explored achieving performances slightly lower than the baselines. The use of hg38 language model improves performance when all classes are considered jointly, however when split into three categories, it tends to perform worse than the uninitialized network. Additional tests with an increased input size are performed, revealing a positive correlation between input size and performance relative to DNAse I sensitivity and histone methylation sites (while performance relative to transcription factor binding sites remains roughly constant). Although associated with a high quality dataset with millions of samples, the task of chromatin profile prediction mixes three independent subtasks together and has limited practical value, so it shouldn't be considered a good benchmark for bioinformatics.

6.2 Future work

In spite of data availability, the proposed tasks in which fine tuning was performed are by no means considerable benchmark tasks useful in bioinformatics, due to simplicity (promoter localization), lack of a solid baseline and "real-world" data (12S metagenomics) or the joint learning of very different tasks (chromatin profile prediction). For this reason, an important dimension to explore is the preparation of solid and meaningful benchmark tasks for bioinformatics.

The training data used on this thesis came from reference genomes and databases which provide "exact" sequences, however real data output by a sequencer is associated with uncertainty which can be quantified by a metric known as phred quality score [12] and associated at the single nucleotide level. To the author's knowledge, how to embed this information about uncertainty of reads has not been addressed yet, with the exception of Busia et al. [5] who, simply and arguably, propose to make quality scores a new input channel for their convolutional network (which however wouldn't allow the network to learn to treat these values as confidences, unless some other modifications are done, eg. by modifying the loss function or the network connectivity). One possible solution with a transformer is to modify the attention mechanism in order to weight the query based on the quality of each token, but this raises further questions which need to be answered:

- How to define a "token-level" quality: the phred score is associated with single nucleotides, but each token is composed by many of them. Devising a way to combine single-nucleotide qualities may not be straightforward (especially due to byte-pair encoding), or different approaches may yield different results (eg. taking the average, the maximum, etc.)

- How to apply the weight: phred scores express the probability of error in logarithmic scale and this may or may not require some conversion or derived metric, moreover a “learned” weighting could prove to be more general.
- How to deal with positional properties: a single sequenced read has a distinct quality pattern (eg. in Illumina sequencing, the error is high at the beginning due to calibration, then becomes low and finally rises again as reads become longer and pollute the flow cell) and an assembly (especially a de novo one) may still have some degrees of uncertainty (eg. because for a given position there are very few, disagreeing, overlaps), on the other hand some tasks may be very sensitive to errors in certain positions, but not in other (eg. in molecular clock analysis [41], different regions mutate at different rates, a low quality in a region relevant to the time scale under exam may cause higher uncertainty on the output, while on other regions it may be irrelevant).
- How to deal with training and inference differences, since current databases contain only high quality assemblies, while runtime inputs may have a varying degree of noise.
- Whether or not the approach is generalizable to multiple tasks (and if so, whether an effective pretraining can be devised to address quality at the embedding level instead of the final task).
- And finally whether this approach may propagate bad laboratory practices (when reads have a quality so low to hinder analysis, the entire sequencing is remade, hopefully by a more expert technician or using

better protocols, increasing robustness with respect to quality may lower the care put into making a good sequencing run), or could be actually useful (eg. in fields like archeology or forensics, in which samples start with a low degree of conservation, reflected by a lower quality of reads, and there is usually the impossibility of repeating an analysis).

Another direction yet to be explored is whether a better language model can be achieved in pretraining without increasing the context size, some possible solutions could be:

- Splitting the training corpus into coherent "contexts" (as mentioned in section 5.1, especially for the human reference genome, there are at least two underlying contexts with a very different number of samples for each case) and balancing them, this should produce a "multilingual" contextual embedding, but potentially require a huge corpus for training,
- Using an Electra-style adversarial masking [6] to increase learning performance.

Appendix A

Losses and metrics used

This appendix contains all the mathematical definitions of the functions used for training and evaluation purposes.

In all cases, y_{true} is the true output value, while y_{pred} is the prediction given by the transformers. In order to reduce memory requirements, y_{true} is provided during training as integer, while the value produced by the transformers is a one-hot encoded vector (a vector of all zeros, except a single one at the index corresponding to the integer it represents). The following formulas assume y_{pred} to be the conversion of the output vector back to integer (ie. the index with maximum value).

A.1 Crossentropy loss, softmax activation and perplexity

Entropy is a measure of the information content of a given source, based on the probability of producing each symbol $\mathcal{P}(X)$, and can be interpreted as the average number of bits required to encode each symbol:

$$\mathcal{H}(X) = -\sum_{i=1}^{n} \mathcal{P}(x_i) log(\mathcal{P}(x_i)). \tag{A.1}$$

When a second probability distribution $\mathcal{Q}(X)$ is used to estimate $\mathcal{P}$, a different number of bits are required to encode each symbol and the crossentropy of $\mathcal{P}$ when approximated by $\mathcal{Q}$ is:

$$\mathcal{H}(\mathcal{P}, \mathcal{Q}) = -\sum_{i=1}^{n} \mathcal{P}(x_i) log(\mathcal{Q}(x_i)). \tag{A.2}$$

Since the crossentropy is differentiable and has a minimum when $\mathcal{P} = \mathcal{Q}$, it can be used as loss function for gradient descent, provided that both the of the predictor and the true labels are normalized to produce probability distributions.

If labels are one-hot encoded, they are automatically also a probability distribution (the Kronecker delta function), predictor outputs however need to be explicitly normalized through the softmax activation function, which can be considered a smoothed one-hot encoding:

$$softmax(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_i}}. \tag{A.3}$$

Since each of the K neurons in the layer are exponentiated and then normalized with respect to the entire layer, the output will take non-negative values which sum up to 1 (which are the two properties of a probability distribution).

In natural language processing it's typical to exponentiate the crossentropy in order to get a metrics, called perplexity, with a more intuitive interpretation:

$$PP(\mathcal{P}, \mathcal{Q}) = 2^{\mathcal{H}(\mathcal{P}, \mathcal{Q})}. \tag{A.4}$$

In language modeling perplexity can be interpreted as the average number of tokens the predictor is "perplexed about" in a given context (ie. how many options it considers a possible correct answer), with a value of 1 implying perfect modeling of the language and a value approaching the vocabulary size implying random guessing.

A.2 Accuracies, precision and recall

Accuracy is the simplest and most direct performance metric, measuring the ratio between the correct predictions and total predictions:

$$ACC = \sum_{i=1}^{N} \begin{cases} \frac{1}{N} & \text{if } y_{true}[i] = argmax\{y_{pred}[i]\} \\ 0 & \text{otherwise.} \end{cases} \tag{A.5}$$

In tasks where the prediction can be on many different classes, pure accuracy may be relatively uninformative, because it provides no information about how “close” the prediction is to becoming accurate. Top-k accuracy relaxes the definition in order to consider correct also predictions in which the right class is not necessarily the class with the highest score, but it’s among the top-k (usually $k = 5$) scores:

$$\text{K-ACC} = \sum_{i=1}^{N} \begin{cases} \frac{1}{N} & \text{if } y_{true}[i] \in argsort\{y_{pred}[i]\}[0:k] \\ 0 & \text{otherwise.} \end{cases} \tag{A.6}$$

Precision is a metric measuring how many of the true samples are detected by the predictor:

$$PRE = \frac{\text{true positives}}{\text{true positives} + \text{false positive}}. \tag{A.7}$$

Recall, instead, measures how many of the predicted positives were in fact true positives:

$$REC = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}. \tag{A.8}$$

A.3 Area under the curve (AUC)

In a binary classification task, the output of a predictor will be a real number in a given range (eg. $[0; 1]$ in the case of sigmoid activation functions). To recover the actual $true, false$ prediction, it's required to fix a threshold (eg. everything above 0.5 will be considered $true$ and the remainder will be considered $false$) and its choice affects both false positives and false negatives (ie. an high threshold will reduce false positives at the expenses of an increase of false negatives, and vice versa).

The receiver operating characteristic (ROC) curve is a plot of the true positive rate and false positive rate with respect to all the possible thresholds. The area of the ROC curve (which can be approximated by any numerical method) is:

$$AUC = \int_{x=0}^{1} TPR(FPR^{-1}(x))dx. \tag{A.9}$$

The AUC can be interpreted as the probability of returning a predicted value higher for a random positive sample compared to a random negative sample. As such, a value of 0.5 indicates random guessing and a value of 1.0 perfect classification. In case of multi-label classification tasks, averaging the AUC for each of the variables preserves this interpretation (contrary to other metrics which must consider all the labels at once).

Appendix B

Increasing performance

This appendix contains further experiments aimed at training better performing architectures, which are however too large and therefore not the main focus of this thesis.

B.1 Architectures

- Large Longformer:
  - 4 blocks (instead of 2),
  - 32 self-attention heads (instead of 8),
  - 256 maximum input tokens,
  - 64 token attention window size (instead of 16),
  - 256-dimensional output embedding.
- ELMo [38]:
  - 2 bidirectional LSTM layers,
  - 256 neurons for each layer,
  - all the hidden states of the last layer returned as embeddings.

`Query`: [omissis] **AATCATATC** CAC ACAA **<mask>** ATAT TAAAA **<mask>** TAT TG TTGAG GCC T CG **<mask> <mask> <mask>** ACG CTGC CTG CTT [omissis]

True: [omissis] **TTCCT** CAC ACAA **TA** ATAT TAAAA **ATAT** TAT TG TTGAG GCC T CG **CGAT TGC TT** ACG CTGC CTG CTT [omissis]

Pred: [omissis] **TTTGA** CAC ACAA **AG** ATAT TAAAA **AAAA** TAT TG TTGAG GCC T CG **CGG TTT TT** ACG CTGC CTG CTT [omissis]

Figure B.1: A wrong MLM prediction of the large Longformer. It can still be noticed that almost all of the replaced tokens have the same length as the true tokens.

B.2 Mitochondrial MLM

Model	Accuracy	Perplexity
Longformer (baseline)	0.8875	1.7257
Large Longformer	**0.9062**	**1.5149**

Table B.1: Pretraining performance on validation sets.

MLM is performed on the mitochondrial dataset using the already trained 9-mer tokenizer with the default parameters (15% masked tokens, with 8:1:1 ratio between `<mask>`, random token and unchanged), however the dataset is dynamically rebuilt before each epoch as suggested by Liu et al. [29] and the large Longformer is trained ($lr = 10^{-4}, \beta_1 = 0.9, \beta_2 = 0.999$) for 200 epochs instead of 50.

ELMo is not pretrained for MLM and will be used for downstream tasks uninitialized.

On the test set, performance is slightly better (accuracy: 0.9163, perplexity: 1.4282), but, more importantly, qualitative evaluation produces some results (figures B.1 and B.2).

Query: [omissis] TT TGT **<mask>** TAAAA ACAAAA AG **<mask>** CGATAG ACA CAGTT GAAA **AGAACAAAG** CTT TCT TGG TT TTAT GTTAA **CCTTC <mask>** CAC [omissis]

True: [omissis] TT TGT **TTTT** TAAAA ACAAAA AG **T** CGATAG ACA CAGTT GAAA **CAC** CTT TCT TGG TT TTAT GTTAA **TG TT** CAC [omissis]

Pred: [omissis] TT TGT **TTTT** TAAAA ACAAAA AG **T** CGATAG ACA CAGTT GAAA **AAG** CTT TCT TGG TT TTAT GTTAA **CCTTC G** CAC [omissis]

Figure B.2: An average MLM prediction of the large Longformer.

B.3 Promoter localization

Model	Accuracy
Longformer (baseline)	**0.9575**
ELMo	0.9543
Large Longformer	0.9544

Table B.2: Promoter localization performance on *H. sapiens* test sets.

ELMo is equipped with the classification head connected to all the (flattened) hidden states of the last layer, while the large Longformer connectivity is identical (dimensions aside) to the one described in section 4.2.2.

ELMo and the large Longformer (initialized with the newly trained mitochondrial language model) are trained for 10 epochs on the promoter localization task, achieving identical performances compared to the small longformer. This result further demonstrates the simplicity of the task.

The optimizer hyperparameters for ELMo were $lr = 10^{-3}, \beta_1 = 0.9, \beta_2 = 0.999$, while for large Longformer were $lr = 10^{-5}, \beta_1 = 0.9, \beta_2 = 0.999$.

Model	Loss	Top-1 Accuracy	Top-5 Accuracy
Longformer (baseline)	0.4082	0.8576	0.9851
ELMo	0.4394	0.9164	0.9783
Large Longformer (deep head)	**0.3111**	**0.9172**	**0.9932**

Table B.3: 12S mitochondrial metagenomics performance on test sets.

B.4 12S Metagenomics

Unlike the small transformer (section 4.2.3), the classification head for the large longformer is no longer connected to the `<cls>` embedding only, instead each token contributes to the final classification to provide "lower level" embeddings as classification features. In order to better exploit these features, the classification head is made deeper, with two dense layers (one with 2048 neurons and the hyperbolic tangent activation function and the other with one neuron for each class and the softmax activation function). Before reaching the first classification layer, the embeddings are subject to a 10% dropout rate to reduce overfitting.

For ELMo all the hidden states of the last layer are flattened and fed to the classification head (which is however constituted simply by a dense layer, just like the one used in section 4.2.3, to limit the already long training time).

ELMo is trained for 5 epochs only ($lr = 10^{-3}, \beta_1 = 0.9, \beta_2 = 0.999$), since it started degrading performance on the validation set.

The large Longformer is trained for 30 epochs ($lr = 10^{-5}, \beta_1 = 0.9, \beta_2 = 0.999$, mitochondrial language model pretraining), in order to compensate for the significantly lower learning rate. These additional epochs can still be trained in less than the time required to train ELMo.

Without a deep classification head, large Longformer pretrained with the mitochondrial language model performs still better than its small counterpart, but worse than ELMo (with no pretraining). It can be theorized however that the embeddings can still be further improved (eg. by training on a 12S-only language model, or by further reducing perplexity on the mitochondrial genome) to get the same performances achieved with the deep head, while using a shallow classification head.

B.5 Chromatin profile prediction

Model	Loss	Precision	Recall	AUC all	AUC DNase	AUC TF	AUC Histone
Longformer (baseline)	**0.6474**	0.1883	0.2719	**0.6495**	0.8604	0.9199	0.8431
DeepSea (baseline)					0.923	0.958	0.856
BigBird (SOTA)					0.921	**0.961**	0.887
ELMo	1.3831	**0.2793**	0.2497	0.6100	0.9284	0.9431	0.9188
Large Longformer	1.3027	0.2277	**0.3002**	0.6249	**0.9517**	0.9600	**0.9287**

Table B.4: Chromatin profile prediction performance on test sets.

Just like the previous two experiments, ELMo's hidden states of the last layer are flattened and fed to the classification head, while the large Longformer preserves the connectivity described in section 4.2.4.

ELMo is trained for 10 epochs ($lr = 10^{-3}, \beta_1 = 0.9, \beta_2 = 0.999$) on the chromatin profile prediction task, while large Longformer for 20 epochs ($lr = 10^{-4}, \beta_1 = 0.9, \beta_2 = 0.999$, mitochondrial language model pretraining). The results achieved, although slightly, improve the state of the art for the task.

This result, achieved with 256 input tokens, shows that the positive correlation between input size and performance claimed by BigBird and also shown by experiments on the small Longformer, may be an artifact of "incomplete" learning (ie. the suboptimal transformers may need longer contexts to disambiguate difficult cases, but the optimal one may exploit higher level features to achieve the same results).

Bibliography

[1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] I. Beltagy, M. E. Peters, and A. Cohan. Longformer: the long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[4] R. Bharanikumar, K. A. R. Premkumar, and A. Palaniappan. Promoter-predict: sequence-based modelling of escherichia coli sigma 70 promoter strength yields logarithmic dependence between promoter strength and sequence. *PeerJ*, 6:e5862, 2018.

[5] A. Busia, G. E. Dahl, C. Fannjiang, D. H. Alexander, E. Dorfman, R. Poplin, C. Y. McLean, P.-C. Chang, and M. DePristo. A deep learning approach to pattern recognition for short dna sequences. *BioRxiv*:353474, 2019.

[6] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. Electra: pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[7] F. Crick. Central dogma of molecular biology. *Nature*, 227(5258):561–563, 1970.

[8] R. Daniel. The metagenomics of soil. *Nature Reviews Microbiology*, 3(6):470–478, 2005.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[10] R. Dreos, G. Ambrosini, R. Groux, R. Cavin Périer, and P. Bucher. The eukaryotic promoter database in its 30th year: focus on non-vertebrate organisms. *Nucleic acids research*, 45(D1):D51–D55, 2017.

[11] R. Dreos, G. Ambrosini, R. C. Périer, and P. Bucher. The eukaryotic promoter database: expansion of epdnew and new promoter analysis tools. *Nucleic acids research*, 43(D1):D92–D96, 2015.

[12] B. Ewing and P. Green. Base-calling of automated sequencer traces using phred. ii. error probabilities. *Genome research*, 8(3):186–194, 1998.

[13] Q. Fournier, G. M. Caron, and D. Aloise. A practical survey on faster and lighter transformers. *arXiv preprint arXiv:2103.14636*, 2021.

[14] M. Ghandi, D. Lee, M. Mohammad-Noori, and M. A. Beer. Enhanced regulatory sequence prediction using gapped k-mer features. *PLoS Comput Biol*, 10(7):e1003711, 2014.

[15] S. R. Gill, M. Pop, R. T. DeBoy, P. B. Eckburg, P. J. Turnbaugh, B. S. Samuel, J. I. Gordon, D. A. Relman, C. M. Fraser-Liggett, and K. E. Nelson. Metagenomic analysis of the human distal gut microbiome. *science*, 312(5778):1355–1359, 2006.

[16] M. W. Gray, G. Burger, and B. F. Lang. The origin and early evolution of mitochondria. *Genome biology*, 2(6):1–5, 2001.

[17] GRCh37 - Genome - Assembly - NCBI. URL: `https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.13`.

[18] GRCh38.p13 - Genome - Assembly - NCBI. URL: `https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.39`.

[19] W. Iwasaki, T. Fukunaga, R. Isagozawa, K. Yamada, Y. Maeda, T. P. Satoh, T. Sado, K. Mabuchi, H. Takeshima, M. Miya, et al. Mitofish and mitoannotator: a mitochondrial genome database of fish with an accurate and automatic annotation pipeline. *Molecular biology and evolution*, 30(11):2531–2540, 2013.

[20] E. V. Koonin and A. S. Novozhilov. Origin and evolution of the genetic code: the universal enigma. *IUBMB life*, 61(2):99–111, 2009.

[21] O. Kovaleva, A. Romanov, A. Rogers, and A. Rumshisky. Revealing the dark secrets of bert. *arXiv preprint arXiv:1908.08593*, 2019.

[22] J. Kuczynski, J. Stombaugh, W. A. Walters, A. González, J. G. Caporaso, and R. Knight. Using qiime to analyze 16s rrna gene sequences from microbial communities. *Current protocols in bioinformatics*, 36(1):10–7, 2011.

[23] T. Kudo and J. Richardson. Sentencepiece: a simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

[24] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: a lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[25] J. J. Levy, A. J. Titus, C. L. Petersen, Y. Chen, L. A. Salas, and B. C. Christensen. Methylnet: an automated and modular deep learning approach for dna methylation analysis. *BMC bioinformatics*, 21(1):1–15, 2020.

[26] H. Li and R. Durbin. Fast and accurate short read alignment with burrows–wheeler transform. *bioinformatics*, 25(14):1754–1760, 2009.

[27] W. Liang. Segmenting dna sequence into words based on statistical language model. *Nature Precedings*:1–1, 2012.

[28] H. Lin, Z.-Y. Liang, H. Tang, and W. Chen. Identifying sigma 70 promoters with novel pseudo nucleotide composition. *IEEE/ACM transactions on computational biology and bioinformatics*, 16(4):1316–1321, 2017.

[29] 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.

[30] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[31] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013.

[32] Y. Murakami and K. Mizuguchi. Applying the naïve bayes classifier with kernel density estimation to the prediction of protein–protein interaction sites. *Bioinformatics*, 26(15):1841–1848, 2010.

[33] J. T. Nearing, G. M. Douglas, A. M. Comeau, and M. G. Langille. Denoising the denoisers: An independent evaluation of microbiome sequence error-correction methods. Technical report, PeerJ Preprints, 2018.

[34] Organelle Genome Resource. URL: `https://www.ncbi.nlm.nih.gov/genome/organelle`.

[35] H. U. Osmanbeyoglu and M. K. Ganapathiraju. N-gram analysis of 970 microbial organisms reveals presence of biological language models. *BMC bioinformatics*, 12(1):1–12, 2011.

[36] M. Oubounyt, Z. Louadi, H. Tayara, and K. T. Chong. Deepromoter: robust promoter predictor using deep learning. *Frontiers in genetics*, 10:286, 2019.

[37] J. Pennington, R. Socher, and C. D. Manning. Glove: global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[38] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[39] G. Piñar, C. Poyntner, H. Tafer, and K. Sterflinger. A time travel story: metagenomic analyses decipher the unknown geographical shift and the storage history of possibly smuggled antique marble statues. *Annals of Microbiology*, 69(10):1001–1021, 2019.

[40] 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, 2016.

[41] A. Rieux, A. Eriksson, M. Li, B. Sobkowiak, L. A. Weinert, V. Warmuth, A. Ruiz-Linares, A. Manica, and F. Balloux. Improved calibration of the human mitochondrial clock using ancient genomes. *Molecular biology and evolution*, 31(10):2780–2792, 2014.

[42] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni. Green ai. *arXiv preprint arXiv:1907.10597*, 2019.

[43] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[44] N. C. Sheffield, R. E. Thurman, L. Song, A. Safi, J. A. Stamatoyannopoulos, B. Lenhard, G. E. Crawford, and T. S. Furey. Patterns of regulatory activity across diverse human cell types predict tissue identity, transcription factor binding, and long-range interactions. *Genome research*, 23(5):777–788, 2013.

[45] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*, 2014.

[46] A. Tampuu, Z. Bzhalava, J. Dillner, and R. Vicente. Viraminer: deep learning on raw dna sequences for identifying viral genomes in human samples. *PloS one*, 14(9):e0222271, 2019.

[47] R. K. Umarov and V. V. Solovyev. Recognition of prokaryotic and eukaryotic promoters using convolutional deep learning neural networks. *PloS one*, 12(2):e0171410, 2017.

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

[49] L. Wei, Y. Ding, R. Su, J. Tang, and Q. Zou. Prediction of human protein subcellular localization using deep learning. *Journal of Parallel and Distributed Computing*, 117:212–217, 2018.

[50] X. Xiao, Z.-C. Xu, W.-R. Qiu, P. Wang, H.-T. Ge, and K.-C. Chou. Ipsw (2l)-pseknc: a two-layer predictor for identifying promoters and their strength by hybrid features via pseudo k-tuple nucleotide composition. *Genomics*, 111(6):1785–1793, 2019.

[51] Y. Yang, R. Zhang, S. Singh, and J. Ma. Exploiting sequence-based features for predicting enhancer–promoter interactions. *Bioinformatics*, 33(14):i252–i260, 2017.

[52] M. Yousef, M. Nebozhyn, H. Shatkay, S. Kanterakis, L. C. Showe, and M. K. Showe. Combining multi-species genomic data for microrna identification using a naive bayes classifier. *Bioinformatics*, 22(11):1325–1334, 2006.

[53] M. Zago, T. Bardelli, L. Rossetti, N. Nazzicari, D. Carminati, A. Galli, and G. Giraffa. Evaluation of bacterial communities of grana padano

cheese by dna metabarcoding and dna fingerprinting analysis. *Food Microbiology*, 93:103613, 2021.

[54] M. Zaheer, G. Guruganesh, A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, et al. Big bird: transformers for longer sequences. *arXiv preprint arXiv:2007.14062*, 2020.

[55] H. Zhang, C.-L. Hung, M. Liu, X. Hu, and Y.-Y. Lin. Ncnet: deep learning network models for predicting function of non-coding dna. *Frontiers in genetics*, 10:432, 2019.

[56] J. Zhou and O. G. Troyanskaya. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature methods*, 12(10):931–934, 2015.

Acknowledgements

The author acknowledges dr. Lucchini for validation of biological claims, drs. G. Ruberg, N. Ruberg, Tsiotas and Razzaq for proof-reading and “peer reviewing” this thesis work, and prof. Torroni and dr. Galassi for advisorship and validation of NLP methodology and results.

The author also thanks the members of the DISI department at Università di Bologna and dr. Tsiotas for having provided additional computing power.