UNIVERSITY OF THESSALY

DIPLOMA OF ENGINEERING THESIS

Computational Analysis of Genomic Sequences utilizing Machine Learning

Author:
Dimitrios PARASCHAS

Supervisor:
Dr. Artemis HATZIGEORGIOU

A thesis submitted in fulfillment of the requirements for the degree of Diploma of Engineering

in the

Department of Electrical and Computer Engineering

September 3, 2018

Declaration of Authorship

I, Dimitrios PARASCHAS, declare that this thesis titled, "Computational Analysis of Genomic Sequences utilizing Machine Learning" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"Always take the position that you are to some degree wrong, and your goal is to be less wrong over time."

Elon Musk

Abstract

Dimitrios PARASCHAS

Computational Analysis of Genomic Sequences utilizing Machine Learning

Modern high-throughput sequencing has produced a deluge of sequencing data, that promise the extraction of new insights from the code of life. Due to the enormous size of these data and the accelerated rate of their production, they cannot be manually parsed by humans in a timely fashion. Automated programmatic pipelines are instead required to extract the functions of novel sequences and pinpoint the specific sites they originate from. Machine Learning algorithms and specifically Neural Networks are well suited for this task; they can learn the representation of complex functions from characteristics of sequences with known functionality and efficiently parse and categorize novel ones, as well as find the origin site for each functionality. This project addresses the specific problem of recognizing whether a novel Homo Sapiens DNA sequence encodes one or more proteins, and pinpoints the Translation Initiation Sites if they exist in this sequence. An end-to-end processing pipeline has been created that parses the input sequence, locates existing open reading frames, extracts required features and finally determines whether a protein is encoded by each open reading frame with excellent accuracy, precision, and recall.

Περίληψη

Δημήτριος Παρασχάς

Υπολογιστική Ανάλυση Γονιδιακών Ακολουθιών με χρήση
Μηχανικής Μάθησης

Οι σύγχρονες μέθοδοι αλληλούχισης γονιδιακών ακολουθιών έχουν παράξει έναν τεράστιο όγκο δεδομένων, που μας υπόσχονται την εξαγωγή νέων σημαντικών πληροφοριών από τον κώδικα της ζωής. Εξαιτίας του γιγαντιαίου μεγέθους αυτών των δεδομένων και του επιταχυνόμενου ρυθμού παραγωγής τους, είναι αδύνατο να αναλυθούν χειροκίνητα από τους επιστήμονες. Αντίθετα, απαιτούνται αυτοματοποιημένες διαδικασίες και προγράμματα τα οποία θα εξάγουν την πληροφορία της λειτουργικότητας ακολουθιών που επεξεργάζονται για πρώτη φορά, και θα προσδιορίζουν με ακρίβεια τη θέση στην οποία αυτές εκκινούν. Οι αλγόριθμοι μηχανικής μάθησης και συγκεκριμένα τα νευρωνικά δίκτυα είναι κατάλληλοι για αυτήν την εργασία, εφόσον μπορούν να μάθουν την αναπαράσταση σύνθετης λειτουργίας από χαρακτηριστικά ακολουθιών με γνωστή λειτουργικότητα, καθώς και να αναλύσουν με ταχύτητα νέες, να τις ταξινομήσουν, αλλά και να ανιχνεύσουν το σημείο εκκίνησης για κάθε λειτουργία. Η εργασία αυτή αντιμετωπίζει το πρόβλημα της αναγνώρισης αν μια ακολουθία από DNA του Homo Sapiens κωδικοποιεί μία ή περισσότερες πρωτεΐνες και προσδιορίζει τις θέσεις εκκίνησης της μετάφρασης, αν υπάρχουν σε αυτήν την ακολουθία. Έχει υλοποιηθεί ένα πλήρες πρόγραμμα επεξεργασίας, το οποίο αναλύει την ακολουθία της εισόδου, εντοπίζει τα υπάρχοντα open reading frames, εξάγει τα χαρακτηριστικά του μοντέλου και τελικά καθορίζει για κάθε open reading frame αν κωδικοποιεί κάποια πρωτεΐνη με εξαιρετικά accuracy, precision, και recall.

Acknowledgements

I express my sincerest gratitude to my supervisor Professor Artemis Hatzigeorgiou and my colleague Dr. Nikos Perdikopanis that provided invaluable advice and guidance throughout the research and implementation of this project, as well as all of the great professors I had the opportunity and luck to be taught and influenced by throughout my studies.

Contents

List of Figures

List of Abbreviations

DNA	deoxyribonucleic acid
RNA	**r**ibo**n**ucleic **a**cid
dsDNA	double-stranded DNA
ssRNA	single-stranded RNA
gDNA	genomic DNA
cDNA	complementary DNA
mRNA	messenger RNA
pre-mRNA	precursor messenger RNA
lncRNA	long non-coding RNA
lincRNA	long intergenic non-coding RNA
bp	base pair(s)
GPU	Graphics Processing Unit
ORF	Open Reading Frame
NN	Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
SVM	Support Vector Machine
AUC	Area Under Curve
ROC	Receiver Operating Characteristic

Chapter 1

Translation Initiation Sites

1.1 Proteins and protein synthesis

Proteins are the main structural and functional molecules in the cells of living organisms and consequently one of the most important type of molecules in all of Life.

A few of the vast array of functions performed by proteins within organisms are the catalysis of metabolic reactions, signal transmission, molecule storage, and molecule transportation. Protein assemblies can perform higher level functions in a cell or organism, as exemplified by the flagellum, a lash-like appendage that protrudes from the cell body of some bacteria and eukaryotic cells and provide them with the capability of locomotion, and the motor that drives it. 1.2

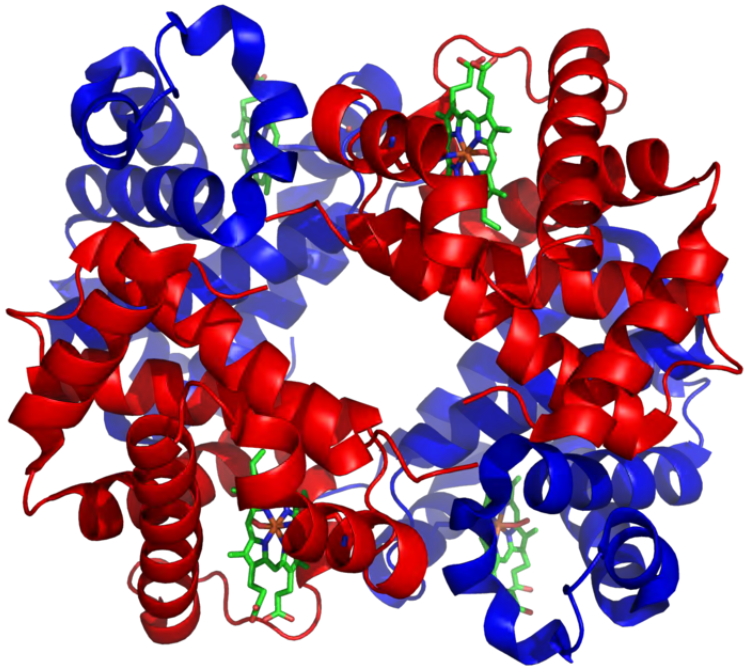

FIGURE 1.1: Structure of human hemoglobin. α and β subunits are in red and blue, and the iron-containing heme groups in green. (Image by Richard Wheeler, Wikimedia Commons)

Proteins are composed of one or more polypeptides, which are long linear chains of amino acid residues, usually called amino acids for short, bonded together by peptide bonds. The

unique sequence of the amino acids for each kind of protein results in a specific three-dimensional structure that determines its activity and function.

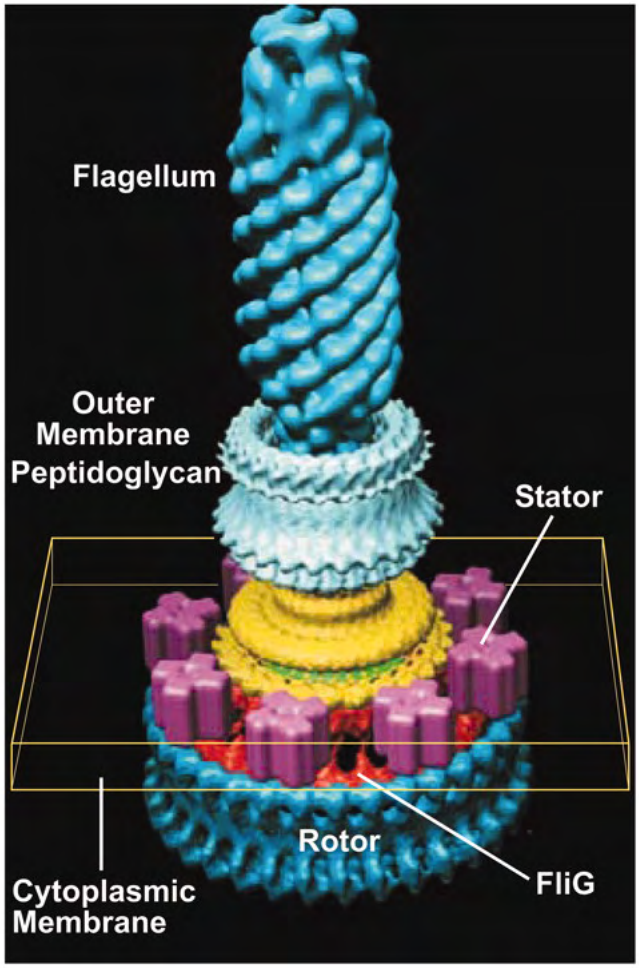

FIGURE 1.2: The overall structure of the Bacterial Flagellar Motor. Each element with different color is a distinct protein. (Image from Xing et al., 2006)

Proteins are synthesized by ribosomes, which are complex molecular machines that link amino acids in the order specified by messenger RNA (mRNA) molecules they parse, in a process called translation, to produce the polypeptides that will result in the final functional protein, after a possible post-translational modification. The precise process of protein generation in cells is a multi step one called protein synthesis.

The information that encodes the exact amino acid sequence of each protein resides originally in the genome of the organism that generates and utilizes the protein, where the genome being the sum of the genetic material of an organism. Eukaryotic organisms such as homo sapiens use DNA strands to store this information. (Alberts et al., 2014)

1.1.1 DNA and RNA

DNA, short for deoxyribonucleic acid, is a macromolecule and specifically a thread-like chain of nucleotides.

Most DNA molecules consist of two biopolymer strands coiled around each other to form a double helix. The two DNA strands are called polynucleotides as they are composed of simpler monomer units called nucleotides. (Alberts et al., 2014) Each nucleotide is composed of one of four nitrogen-containing nucleobases (adenine [A], cytosine [C], guanine [G], thymine [T]), a sugar called deoxyribose, and a phosphate group. The nucleotides are joined to one another in a chain by covalent bonds between the sugar of one nucleotide and the phosphate of the next, resulting in an alternating sugar-phosphate backbone. The nitrogenous bases of the two separate polynucleotide strands are bound together, according to base pairing rules (A with T and C with G), with hydrogen bonds to make double-stranded DNA.

In a double helix, the direction of the nucleotides in one strand is opposite to their direction in the other strand: the strands are antiparallel. The asymmetric ends of DNA strands are called the 5′ (five prime) and 3′ (three prime) ends, with the 5′ end having a terminal phosphate group and the 3′ end a terminal hydroxyl group.

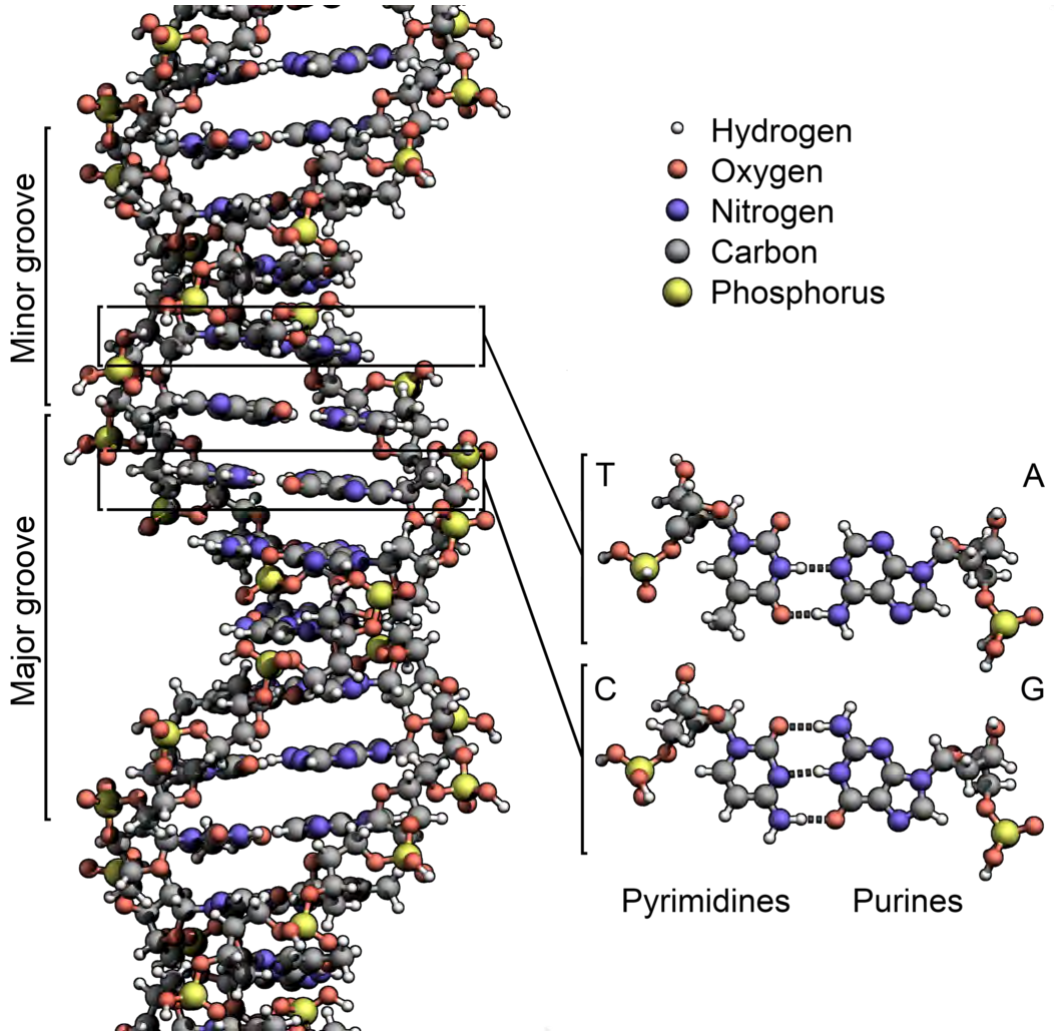

FIGURE 1.3: The structure of the DNA double helix. (Image by Richard Wheeler, Wikimedia Commons)

The information in DNA is organized in units called genes. A gene is defined as a sequence in DNA that codes for a molecule that has a function.

The structure of a protein coding gene consists of many elements of which the actual protein coding sequence is often only a small part. These include DNA regions that are not transcribed as well as untranslated regions of the RNA.

Flanking the open reading frame, genes contain a regulatory sequence that is required for their expression. First, genes require a promoter sequence. The promoter is recognized and bound by transcription factors and RNA polymerase to initiate transcription. The recognition typically occurs as a consensus sequence like the TATA box. A gene can have more than one promoter, resulting in messenger RNAs that differ in how far they extend in the 5′ end. (Mortazavi et al., 2008) Highly transcribed genes have "strong" promoter sequences that form strong associations with transcription factors, thereby initiating transcription at a high rate. Others genes have "weak" promoters that form weak associations with transcription factors and initiate transcription less frequently. Eukaryotic promoter regions are much more complex and difficult to identify than prokaryotic promoters. (Alberts et al., 2014)

Additionally, genes can have regulatory regions many kilobases upstream or downstream of the open reading frame that alter expression. These act by binding to transcription factors which then cause the DNA to loop so that the regulatory sequence (and bound transcription factor) become close to the RNA polymerase binding site. For example, enhancers increase transcription by binding an activator protein which then helps to recruit the RNA polymerase to the promoter; conversely silencers bind repressor proteins and make the DNA less available for RNA polymerase. (Maston, Evans, and Green, 2006)

The transcribed pre-mRNA contains untranslated regions at both ends which contain a ribosome binding site, terminator and start and stop codons. In addition, most eukaryotic open reading frames contain untranslated introns which are removed before the exons are translated. The sequences at the ends of the introns, dictate the splice sites to generate the final mature mRNA which encodes the protein or RNA product. (Bicknell et al., 2012)

Many prokaryotic genes are organized into operons, with multiple protein-coding sequences that are transcribed as a unit. The genes in an operon are transcribed as a continuous messenger RNA, referred to as a polycistronic mRNA. The term cistron in this context is equivalent to gene. The transcription of an operon's mRNA is often controlled by a repressor that can occur in an active or inactive state depending on the presence of certain specific metabolites. When active, the repressor binds to a DNA sequence at the beginning of the operon, called the operator region, and represses transcription of the operon; when the repressor is inactive transcription of the operon can occur (see e.g. Lac operon). The products of operon genes typically have related functions and are involved in the same regulatory network. (Jacob and Monod, 1961)

1.1.2 Transcription

During a process called transcription, an mRNA chain is generated with one strand of the DNA double helix in the genome as a template. In eucaryotes, the original product of the transcription called precursor messenger RNA (pre-mRNA) undergoes a post-transcriptional modification to produce the final mature messenger RNA.

RNA, short for ribonucleic acid, is a polymeric molecule essential in various biological roles in coding, decoding, regulation, and expression of genes. RNA is assembled as a chain of nucleotides, in the same way to DNA, but unlike DNA it is more often found in nature as a single-strand folded onto itself, rather than a paired double-strand. Cellular organisms use messenger RNA (mRNA) to convey genetic information (using the nitrogenous bases

of adenine, cytosine, guanine, and uracil, denoted by the letters A, C, G, and U) that directs synthesis of specific proteins.

The genetic information of mRNA is encoded in the sequence of nucleotides, which are arranged into codons consisting of three nucleotides each. Each codon encodes for a specific amino acid, except for the stop codons, which terminate protein synthesis.

Transcription is divided into the major steps of initiation, promoter escape, elongation, and termination.

Initiation

Transcription begins with the binding of RNA polymerase, together with one or more general transcription factors, to a specific DNA sequence, referred to as a promoter, to form an RNA polymerase-promoter "closed complex". In the closed complex the promoter DNA is still fully double-stranded.

RNA polymerase, assisted by one or more general transcription factors, then unwinds approximately 14 base pairs of DNA to form an RNA polymerase-promoter "open complex". In the open complex the promoter DNA is partly unwound and single-stranded. The exposed, single-stranded DNA is referred to as the "transcription bubble".

RNA polymerase, assisted by one or more general transcription factors, then selects a transcription start site in the transcription bubble, binds to an initiating NTP and an extending NTP (or a short RNA primer and an extending NTP) complementary to the transcription start site sequence, and catalyzes bond formation to yield an initial RNA product.

In bacteria, RNA polymerase holoenzyme consists of five subunits: 2 α subunits, 1 β subunit, 1 β' subunit, and 1 ω subunit. In bacteria, there is one general RNA transcription factor: sigma. RNA polymerase core enzyme binds to the bacterial general transcription factor sigma to form RNA polymerase holoenzyme and then binds to a promoter. (Watson, 2008)

In archaea and eukaryotes, RNA polymerase contains subunits homologous to each of the five RNA polymerase subunits in bacteria and also contains additional subunits. In archaea and eukaryotes, the functions of the bacterial general transcription factor sigma are performed by multiple general transcription factors that work together. In archaea, there are three general transcription factors: TBP, TFB, and TFE. In eukaryotes, in RNA polymerase II-dependent transcription, there are six general transcription factors: TFIIA, TFIIB (an ortholog of archaeal TFB), TFIID (a multisubunit factor in which the key subunit, TBP, is an ortholog of archaeal TBP), TFIIE (an ortholog of archaeal TFE), TFIIF, and TFIIH. In archaea and eukaryotes, the RNA polymerase-promoter closed complex is usually referred to as the "preinitiation complex". (Roeder, 1991)

Transcription initiation is regulated by additional proteins, known as activators and repressors, and, in some cases, associated coactivators or corepressors, which modulate formation and function of the transcription initiation complex.

Promoter escape

After the first bond is synthesized, the RNA polymerase must escape the promoter. During this time there is a tendency to release the RNA transcript and produce truncated transcripts. This is called abortive initiation, and is common for both eukaryotes and prokaryotes. (Goldman, Ebright, and Nickels, 2009) Abortive initiation continues to occur until an RNA product of a threshold length of approximately 10 nucleotides is synthesized, at which point promoter escape occurs and a transcription elongation complex is formed.

Mechanistically, promoter escape occurs through DNA scrunching, providing the energy needed to break interactions between RNA polymerase holoenzyme and the promoter.

In bacteria, upon and following promoter clearance, the σ factor is released according to a stochastic model.

In eukaryotes, at an RNA polymerase II-dependent promoter, upon promoter clearance, TFIIH phosphorylates serine 5 on the carboxy terminal domain of RNA polymerase II, leading to the recruitment of capping enzyme (CE). The exact mechanism of how CE induces promoter clearance in eukaryotes is not yet known. (Mandal et al., 2004)

Elongation

One strand of the DNA, the template strand (or noncoding strand), is used as a template for RNA synthesis. As transcription proceeds, RNA polymerase traverses the template strand and uses base pairing complementarity with the DNA template to create an RNA copy. Although RNA polymerase traverses the template strand from 3′ to 5′, the coding (non-template) strand and newly formed RNA can also be used as reference points, so transcription can be described as occurring 5′ to 3′. This produces an RNA molecule from 5′ to 3′, an exact copy of the coding strand (except that thymines are replaced with uracils, and the nucleotides are composed of a ribose (5-carbon) sugar where DNA has deoxyribose (one fewer oxygen atom) in its sugar-phosphate backbone).

mRNA transcription can involve multiple RNA polymerases on a single DNA template and multiple rounds of transcription (amplification of particular mRNA), so many mRNA molecules can be rapidly produced from a single copy of a gene. The characteristic elongation rates in prokaryotes and eukaryotes are about 10-100 nts/sec. In eukaryotes, however, nucleosomes act as major barriers to transcribing polymerases during transcription elongation. In these organisms, the pausing induced by nucleosomes can be regulated by transcription elongation factors such as TFIIS. (Fitz et al., 2016)

Elongation also involves a proofreading mechanism that can replace incorrectly incorporated bases. In eukaryotes, this may correspond with short pauses during transcription that allow appropriate RNA editing factors to bind. These pauses may be intrinsic to the RNA polymerase or due to chromatin structure.

Termination

Bacteria use two different strategies for transcription termination – Rho-independent termination and Rho-dependent termination. In Rho-independent transcription termination, RNA transcription stops when the newly synthesized RNA molecule forms a G-C-rich hairpin loop followed by a run of Us. When the hairpin forms, the mechanical stress breaks the weak rU-dA bonds, now filling the DNA-RNA hybrid. This pulls the poly-U transcript out of the active site of the RNA polymerase, terminating transcription. In the Rho-dependent type of termination, a protein factor called Rho destabilizes the interaction between the template and the mRNA, thus releasing the newly synthesized mRNA from the elongation complex. (Richardson, 2002)

Transcription termination in eukaryotes is less well understood than in bacteria, but involves cleavage of the new transcript followed by template-independent addition of adenines at its new 3′ end, in a process called polyadenylation. (Lykke-Andersen and Jensen, 2007)

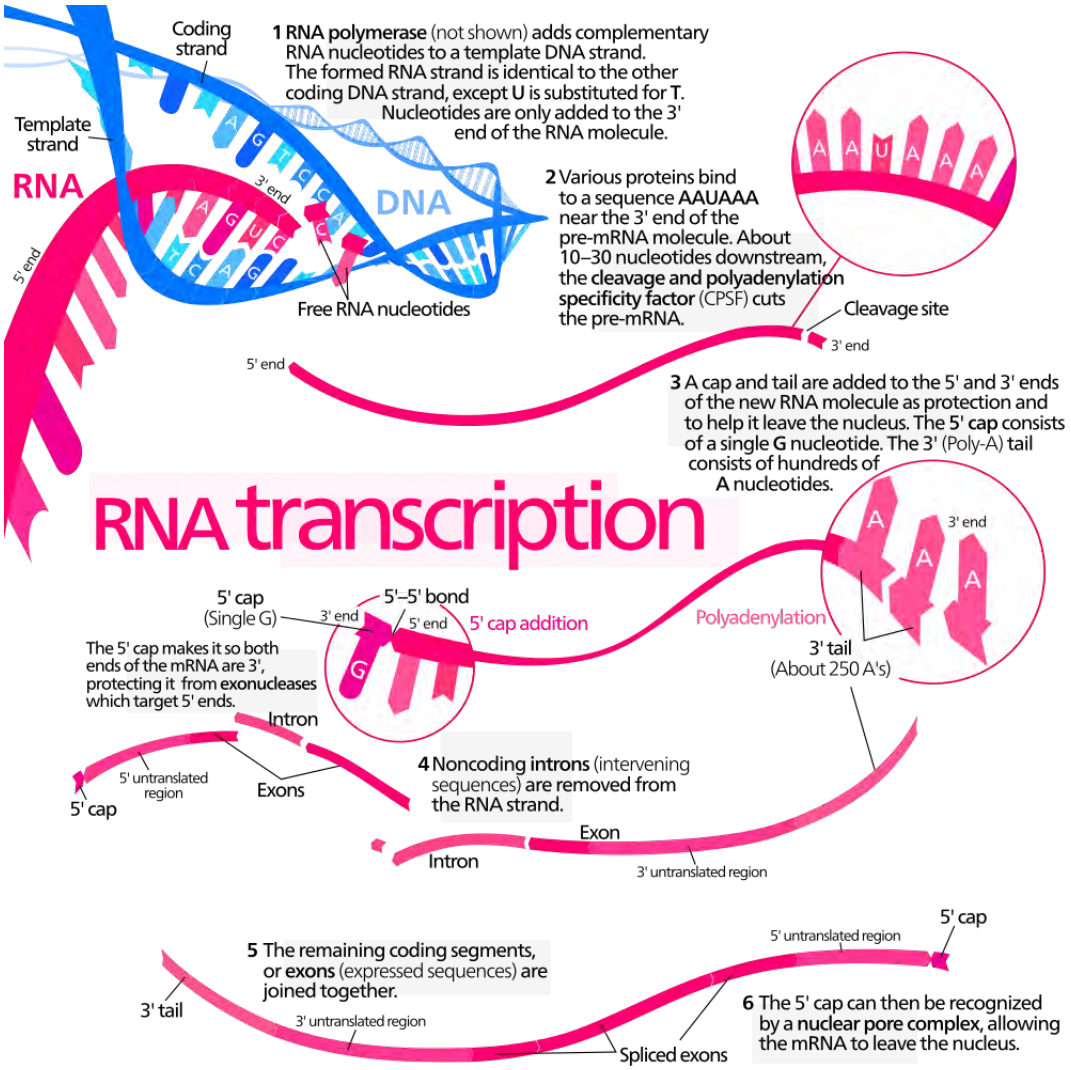

FIGURE 1.4: Diagram of mRNA synthesis and processing. (Image by Kelvin Ma, Wikimedia Commons)

1.1.3 Translation

Translation is the process in which ribosomes in the cytoplasm or endoplasmic reticulum (ER) synthesize proteins.

Specifically, messenger RNA (mRNA) is decoded in a ribosome to produce a specific amino acid chain, or polypeptide. The polypeptide later folds into an active protein and performs its functions in the cell. The ribosome facilitates decoding by inducing the binding of complementary tRNA anticodon sequences to mRNA codons. The tRNAs carry specific amino acids that are chained together into a polypeptide as the mRNA passes through and is "read" by the ribosome.

There are generally three phases in the translation process:

Initiation, where the ribosome assembles around the target mRNA. The first tRNA is attached at the start codon.

Elongation, where the tRNA transfers an amino acid to the tRNA corresponding to the next codon. The ribosome then moves (translocates) to the next mRNA codon to continue the process, creating an amino acid chain.

Termination, in which a stop codon is reached and the ribosome releases the polypeptide.

The basic process of translation is the addition of one amino acid at a time to the end of the polypeptide being formed. This process takes place inside the ribosome. A ribosome is made up of two subunits, a small 40S subunit and a large 60S subunit. These subunits come together before translation of mRNA into a protein to provide a location for translation to be carried out and a polypeptide to be produced. The choice of amino acid type to be added is determined by the genetic code on the mRNA molecule. Each amino acid added is mismatched to a three codon sequence of the mRNA. For each such triplet possible, the corresponding amino acid is accepted. The successive amino acids added to the chain are matched to successive nucleotide triplets in the mRNA. In this way, the sequence of nucleotides in the template mRNA chain determines the sequence of amino acids in the generated polypeptide. Addition of an amino acid occurs at the C-terminus of the peptide and thus translation is said to be amino-to-carboxyl directed.

The tRNA carries genetic information encoded as a DNA sequence from the chromosomes to the nucleolus. The ribonucleotides are "read" by translational machinery in a sequence of nucleotide triplets called codons. Each of those triplets codes for a specific amino acid.

The ribosome molecules translate this code to a specific sequence of amino acids. The ribosome is a multisubunit structure containing rRNA and proteins. It is the "factory" where amino acids are assembled into proteins. tRNAs are small noncoding RNA chains (74-93 nucleotides) that transport amino acids to the ribosome. tRNAs have a site for amino acid attachment, and a site called an anticodon. The anticodon is an RNA triplet complementary to the mRNA triplet that codes for their cargo amino acid.

Aminoacyl tRNA synthetases (enzymes) catalyze the bonding between specific tRNAs and the amino acids that their anticodon sequences call for. The product of this reaction is an aminoacyl-tRNA. In prokaryotes, this aminoacyl-tRNA is carried to the ribosome by EF-Tu, where mRNA codons are matched through complementary base pairing to specific tRNA anticodons. Aminoacyl-tRNA synthetases that mispair tRNAs with the wrong amino acids can produce mischarged aminoacyl-tRNAs, which can result in inappropriate amino acids at the respective position in protein. This "mistranslation" (Moghal, Mohler, and Ibba, 2014) of the genetic code naturally occurs at low levels in most organisms, but certain cellular environments cause an increase in permissive mRNA decoding, sometimes to the benefit of the cell.

The ribosome has three sites for tRNA to bind. They are the aminoacyl site (abbreviated A), the peptidyl site (abbreviated P) and the exit site (abbreviated E). With respect to the mRNA, the three sites are oriented 5′ to 3′ E-P-A, because ribosomes move toward the 3′ end of mRNA. The A-site binds the incoming tRNA with the complementary codon on the mRNA. The P-site holds the tRNA with the growing polypeptide chain. The E-site holds the tRNA without its amino acid. When an aminoacyl-tRNA initially binds to its corresponding codon on the mRNA, it is in the A site. Then, a peptide bond forms between the amino acid of the tRNA in the A site and the amino acid of the charged tRNA in the P site. The growing polypeptide chain is transferred to the tRNA in the A site. Translocation occurs, moving the tRNA in the P site, now without an amino acid, to the E site; the tRNA that was in the A site, now charged with the polypeptide chain, is moved to the P site. The tRNA in the E site leaves and another aminoacyl-tRNA enters the A site to repeat the process. (Griffiths, 2008)

After the new amino acid is added to the chain, and after the mRNA is released out of the nucleus and into the ribosome's core, the energy provided by the hydrolysis of a GTP bound to the translocase EF-G (in prokaryotes) and eEF-2 (in eukaryotes) moves the ribosome down one codon towards the 3′ end. The energy required for translation of proteins is significant. For a protein containing n amino acids, the number of high-energy phosphate bonds required to translate it is 4n-1. The rate of translation varies; it is significantly higher in prokaryotic cells (up to 17-21 amino acid residues per second) than in eukaryotic cells (up to 6-9 amino acid residues per second). (Ross and Orlowski, 1982)

Even though the ribosomes are usually considered accurate and processive machines, the translation process is subject to errors that can lead either to the synthesis of erroneous proteins or to the premature abandonment of translation. The rate of error in synthesizing proteins has been estimated to be between 1/105 and 1/103 misincorporated amino acids, depending on the experimental conditions. (Wohlgemuth et al., 2011) The rate of premature translation abandonment, instead, has been estimated to be of the order of magnitude of 10-4 events per translated codon. The correct amino acid is covalently bonded to the correct transfer RNA (tRNA) by amino acyl transferases. The amino acid is joined by its carboxyl group to the 3′ OH of the tRNA by an ester bond. When the tRNA has an amino acid linked to it, the tRNA is termed "charged". Initiation involves the small subunit of the ribosome binding to the 5′ end of mRNA with the help of initiation factors (IF). In prokaryotes, initiation of protein synthesis involves the recognition of a purine-rich initiation sequence on the mRNA called the Shine-Dalgarno sequence. The Shine-Dalgarno sequence binds to a complementary pyrimidine-rich sequence on the 3′ end of the 16S rRNA part of the 30S ribosomal subunit. The binding of these complementary sequences ensures that the 30S ribosomal subunit is bound to the mRNA and is aligned such that the initiation codon is placed in the 30S portion of the P-site. Once the mRNA and 30S subunit are properly bound, an initiation factor brings the initiator tRNA-amino acid complex, f-Met-tRNA, to the 30S P site. The initiation phase is completed once a 50S subunit joins the 30 subunit, forming an active 70S ribosome. Termination of the polypeptide happens when the A site of the ribosome faces a stop codon (UAA, UAG, or UGA) on the mRNA. tRNA usually cannot recognize or bind to stop codons. Instead, the stop codon induces the binding of a release factor protein that prompts the disassembly of the entire ribosome/mRNA complex and the hydrolysis and the release of the polypeptide chain from the ribosome. Drugs or special sequence motifs on the mRNA can change the ribosomal structure so that near-cognate tRNAs are bound to the stop codon instead of the release factors. In such cases of 'translational readthrough', translation continues until the ribosome encounters the next stop codon. (Schueren and Thoms, 2016)

The process of translation is highly regulated in both eukaryotic and prokaryotic organisms. Regulation of translation can impact the global rate of protein synthesis which is closely coupled to the metabolic and proliferative state of a cell. In addition, recent work has revealed that genetic differences and their subsequent expression as mRNAs can also impact translation rate in an RNA-specific manner. (Cenik et al., 2015)

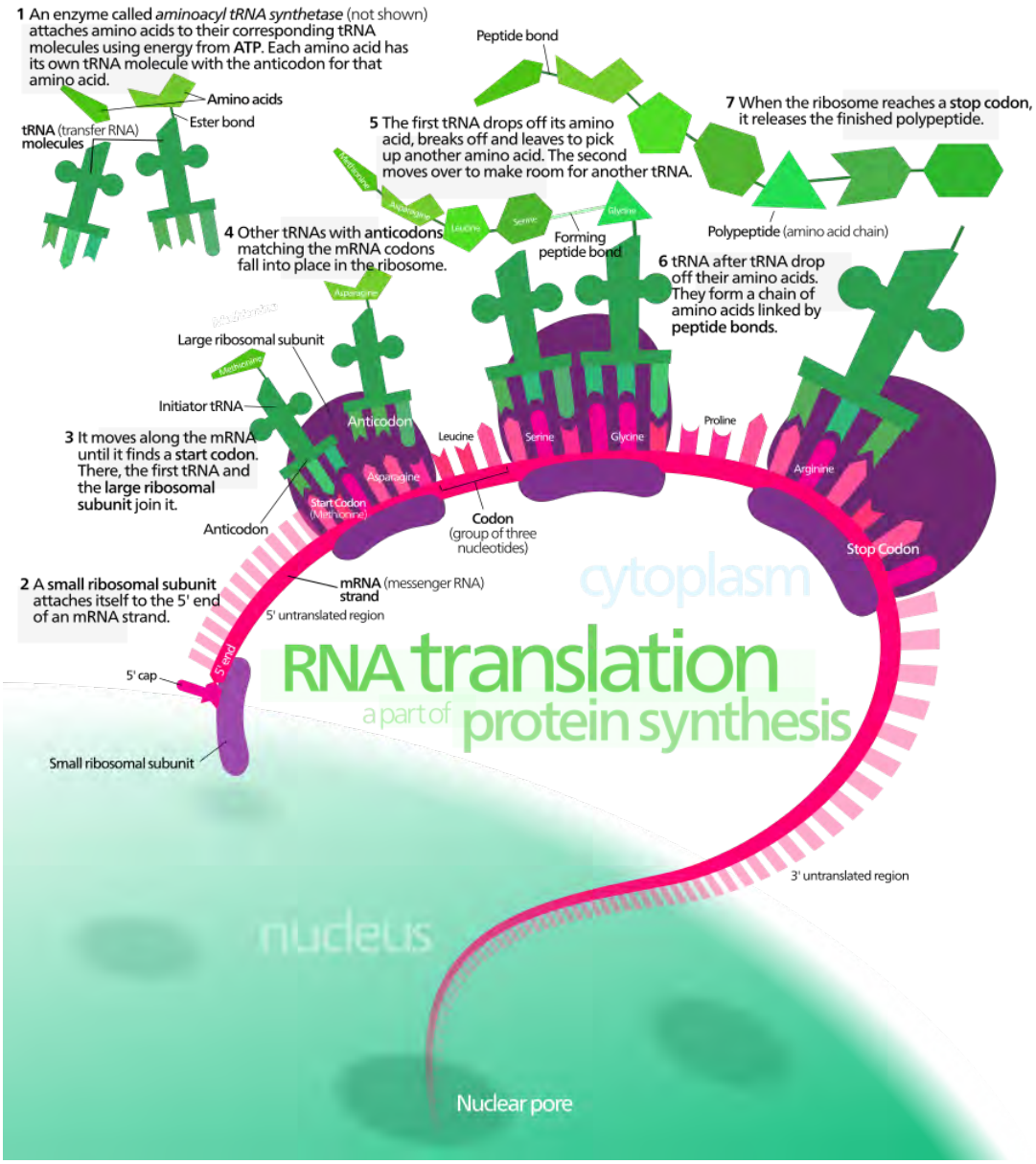

FIGURE 1.5: Diagram of RNA translation. (Image by Kelvin Ma, Wikimedia Commons)

1.2 Translation Initiation Sites identification

In February 2001, the first assembly of the human genome was published by the International Human Genome Sequencing Consortium (Lander, [...], and International Human Genome Sequencing Consortium, 2001). According to this study human genes tend to have small exons (encoding in average only 50 codons) separated by long introns (some exceeding 10 kb). This phenomenon increases the signal-to-noise ratio for algorithms that attempt to facilitate gene prediction, leading to significantly limited accuracy. The performance of such algorithms relies mostly on the availability of coding sequences that can be utilized to develop robust predictive models. The current thesis presents an in silico approach that can readily identify the coding segments of either known or putative genic loci. To this end, the algorithm must be capable of locating all Open Reading Frames (ORFs) located in the queried sequence, and identifying the correct ORF, whose 5′ end is considered the Translation Initiation Site (TIS). The ORF is defined as a stretch of DNA codons that start with the start codon (ATG) and terminate with any of the stop codons (TAA/TAG/TGA). Stop codons are not considered part of the ORF. Since all codons represent triplets of nucleotides, there can be up to three ORFs per single stranded DNA sequence. The immediate flanking loci of coding regions do not encode for proteins and can be considered as non-coding (3′ and 5′ UnTranslated Regions, UTRs). This is a phenomenon that the proposed methodology attempts to exploit since in general, the patterns of nucleotide composition greatly differ between coding and non-coding sequences.

The original work for the identification of TIS in coding sequences dates back to 1987, when Kozak developed the first weight matrix from an extended collection of data (Kozak, 1987). The consensus motif derived from this matrix was GCCACCatgG, describing a G residue following the ATG codon, and a purine, preferably A, three nucleotides upstream, as two highly conserved positions that exert the strongest effect. While attempting to describe what really happens in the cell, Kozak developed the ribosome-scanning model. According to this model, ribosomes initially attach to the specific cap region in the 5′ end of mRNAs and subsequently scan the sequence until they find the first ATG located in an optimal nucleotide context. This is described as the site where the translation of codons into amino acids begins. Although this process characterizes most studied mRNA's, there are some notable exceptions.

In cases where the first ATG codon of the sequence has a less than optimal nucleotide context, it can actually be bypassed by the ribosome, which then initiates translation from a subsequent start codon located in a more optimal nucleotide context further downstream. This phenomenon is also known as leaky scanning. In reinitiation, the translation starts from an ATG codon upstream of the coding region, located in optimal nucleotide context inside the 5′ UTR region and is terminated at the first stop codon, normally in a short downstream distance. Scanning then continues until the authentic ATG codon (start codon) is reached.

During the last decade, a plethora of computational methods has emerged aiming to facilitate the distinction between coding and non-coding sequences. Even though the amount of available implementations is quite large, only a small fraction provides source code for stand-alone usage, while the vast majority can only be accessed through web servers of limited throughput capacity. Most importantly, the vast majority of the implementations use outdated programming languages and methodologies, thus signifying the need for an open source and streamlined implementation based on a modern toolset.

Chapter 2

Deep Learning

2.1 Machine Learning

Machine Learning is a field of computer science that uses statistical techniques to give computer systems the ability to learn from data, or to progressively improve performance on a specific task without being explicitly programmed how to accomplish that.

A core objective of a learner is to generalize from its experience. (Bishop, 2006) Generalization in this context is the ability of a learning machine to perform accurately on new, unseen examples/tasks after having experienced a learning data set. The training examples come from some generally unknown probability distribution, considered representative of the space of occurrences, and the learner has to build a general model about this space that enables it to produce sufficiently accurate predictions in new cases.

The computational analysis of machine learning algorithms and their performance is a branch of theoretical computer science known as computational learning theory. Because training sets are finite and the future is uncertain, learning theory usually does not yield guarantees of the performance of algorithms. Instead, probabilistic bounds on the performance are quite common. The bias-variance decomposition is one way to quantify generalization error.

For the best performance in the context of generalization, the complexity of the hypothesis should match the complexity of the function underlying the data. If the hypothesis is less complex than the function, then the model has underfit the data. If the complexity of the model is increased in response, then the training error decreases. But if the hypothesis is too complex, then the model is subject to overfitting and generalization will be poorer. (Alpaydin, 2010)

In addition to performance bounds, computational learning theorists study the time complexity and feasibility of learning. In computational learning theory, a computation is considered feasible if it can be done in polynomial time. There are two kinds of time complexity results. Positive results show that a certain class of functions can be learned in polynomial time. Negative results show that certain classes cannot be learned in polynomial time.

Today, Machine Learning is employed in a vast range of computing tasks where designing and programming explicit algorithms with good performance is difficult or infeasible, such as image recognition, facial recognition, speech recognition, automated translation, email spam filtering, network intrusion detection, and driverless driving. More recently, a great number of tasks in Bioinformatics, Biotechnology, and Medicine have also been tackled with Machine Learning, such as gene expression, sequence allignment, metabolic pathways modeling, and protein structure and function prediction.

2.2 Neural Networks

Neural Networks (NNs) are machine learning algorithms vaguely inspired by the biological neural networks that constitute animal brains. Neural networks have favorable properties that set them apart and beyond other machine learning algorithms. The most important of them are the ability of improve performance with scale and the ability to improve performance with increased amounts of training data. These characteristics, along with their inherent property of being able to be expressed by simple linear algebra operations, so they can be run very efficiently on off-the-self Graphics Processing Units, lead to their adoption for almost all tasks in the modern machine learning pipelines.

A Neural Network (NN) is composed of a collection of connected units or nodes called artificial neurons which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it.

In common neural network implementations, the signal at a connection between artificial neurons is a real number, and the output of each artificial neuron is computed by some non-linear function of the sum of its inputs. The connections between artificial neurons are called 'edges'. Artificial neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Artificial neurons may have a threshold such that the signal is only sent if the aggregate signal crosses that threshold. Typically, artificial neurons are aggregated into layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first layer, called the input layer, to the last layer, called the output layer, possibly after traversing the layers multiple times.

2.2.1 Multi-layer Perceptron

The most used neural network algorithm is the exceedingly effective and powerful Multi-layer Perceptron (MLP). The Multi-layer Perceptron is a supervised machine learning algorithm that learns a function $f(\cdot) : R^m \rightarrow R^o$ by training on a dataset, where m is the number of dimensions for input and o is the number of dimensions for output. Given a set of features $X = x_1, x_2, ..., x_m$ and a target y, it can learn a non-linear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers. Figure 2.1 shows a one hidden layer MLP with scalar output.

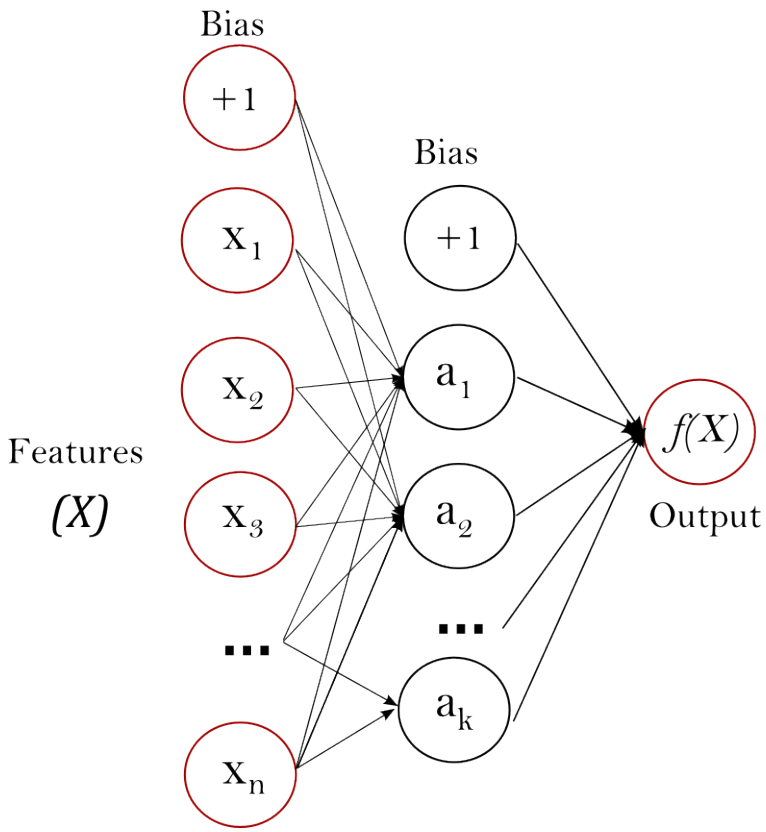

FIGURE 2.1: A single hidden layer Multi-layer Perceptron network (Image from Pedregosa et al., 2011)

The leftmost layer, known as the input layer, consists of a set of neurons $\{x_i|x_1, x_2, ..., x_m\}$ representing the input features. Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation $w_1x_1 + w_2x_2 + ... + w_mx_m$, followed by a non-linear activation function $g(\cdot) : R \rightarrow R$ - like the hyperbolic tan function. The output layer receives the values from the last hidden layer and transforms them into output values.

2.3 Convolutional Neural Networks

Convolutional Neural Networks are very similar to ordinary Neural Networks, as they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other and they still have a loss function on the last, fully-connected layer. On the other hand, Convolutional Neural Networks, or ConvNet for short, architectures make the explicit assumption that the inputs are images or other data with spacial relationship between the distinct features, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network.

Convolutional Neural Networks take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: width,

height, depth. The neurons in a layer will only be connected to a small region of the layer before it, instead of all of the neurons in a fully-connected manner. Moreover, the final output layer would have reduced dimensions, because by the end of the ConvNet architecture we will reduce the full image into a single vector of class scores, arranged along the depth dimension. 2.2

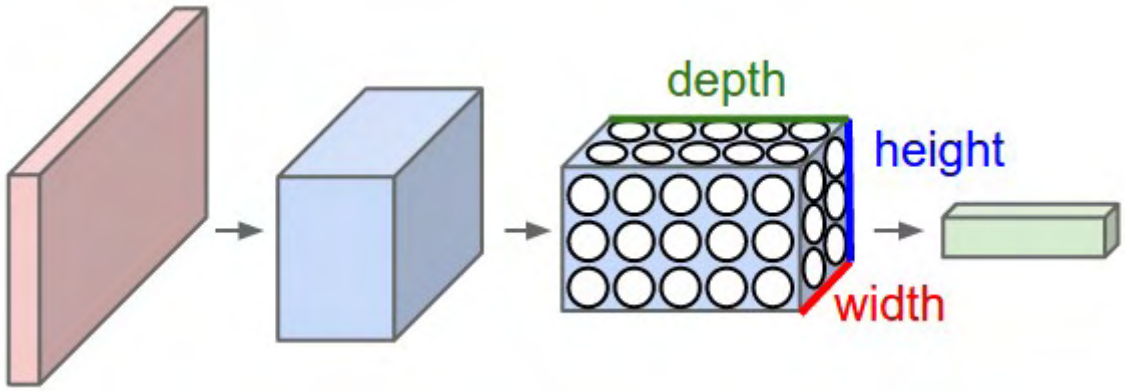

FIGURE 2.2: A Convolutional Neural Network arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In the case of an image, the red input layer holds the image data, so its width and height would be the dimensions of that image, and the depth would be 3, one for each of Red, Green, and Blue channels. (Image from Stanford CS231n: Convolutional Neural Networks)

2.3.1 ConvNets Layers

A simple ConvNet is a sequence of layers, and every layer of a ConvNet transforms one volume of activations to another through a differentiable function. We use three main types of layers to build ConvNet architectures: Convolutional Layer, Pooling Layer, and Fully-Connected Layer, the latter being the same used in regular Neural Networks. We stack these layers to form a full ConvNet architecture.

2.3.2 Convolutional Layer

The Convolutional layer is the core building block of a Convolutional Network that does most of the computational heavy lifting. The Conv layer's parameters consist of a set of learnable filters. Every filter is small spatially, along width and height, but extends through the full depth of the input volume. For example, a typical filter on a first layer of a ConvNet might have size 5x5x3, i.e. 5 pixels width and height, and 3 because images have depth 3, the color channels. During the forward pass, we slide, or more precisely, convolve, each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position. As we slide the filter over the width and height of the input volume we will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position. Intuitively, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color on the first layer, or eventually entire honeycomb or wheel-like patterns on higher layers of the network. Now, we will have an entire set of filters in each

Conv layer, and each of them will produce a separate 2-dimensional activation map. We will stack these activation maps along the depth dimension and produce the output volume.

When dealing with high-dimensional inputs such as images, as we saw above it is impractical to connect neurons to all neurons in the previous volume. Instead, we will connect each neuron to only a local region of the input volume. The spatial extent of this connectivity is a hyperparameter called the receptive field of the neuron; equivalently this is the filter size. The extent of the connectivity along the depth axis is always equal to the depth of the input volume. It is important to emphasize again this asymmetry in how we treat the spatial dimensions, width and height, and the depth dimension: The connections are local in space, along width and height, but always full along the entire depth of the input volume.

Having explained the connectivity of each neuron in the Conv Layer to the input volume, we will now discuss how many neurons there are in the output volume and how they are arranged. Three hyperparameters control the size of the output volume: the **depth**, **stride** and **zero-padding**.

The **depth** of the output volume is a hyperparameter: it corresponds to the number of filters we would like to use, each learning to look for something different in the input. For example, if the first Convolutional Layer takes as input the raw image, then different neurons along the depth dimension may activate in presence of various oriented edges, or blobs of color. We will refer to a set of neurons that are all looking at the same region of the input as a depth column. 2.4

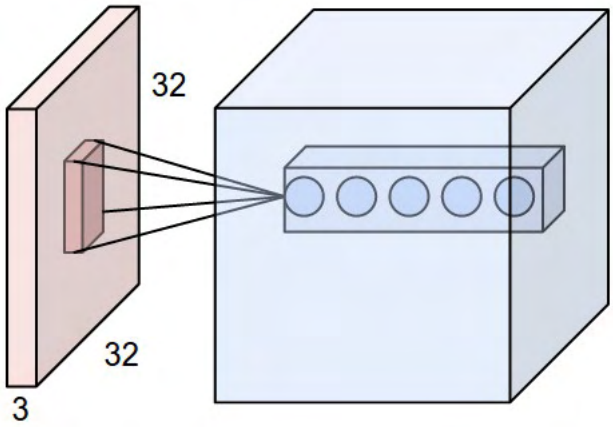

FIGURE 2.3: An example input volume in red, with the dimensions $32 \times 32 \times 3$ of CIFAR-10 images, and an example volume of neurons in the first Convolutional layer. Each neuron in the convolutional layer is connected only to a local region in the input volume spatially, but to the full depth, i.e. all color channels. There are multiple neurons (5 in this example) along the depth, all looking at the same region in the input. (Image from Stanford CS231n: Convolutional Neural Networks)

Subsequently, we must specify the **stride** with which we slide the filter. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2 then the filters jump 2 pixels at a time as we slide them around. This will produce smaller output volumes spatially.

Sometimes it will be convenient to pad the input volume with zeros around the border. The size of this **zero-padding** is a hyperparameter. The nice feature of zero padding is that it will allow us to control the spatial size of the output volumes (most commonly as we'll see

soon we will use it to exactly preserve the spatial size of the input volume so the input and output width and height are the same).

We can compute the spatial size of the output volume as a function of the input volume size $\mathbf{W}$, the receptive field size of the Conv Layer neurons $\mathbf{F}$, the stride with which they are applied $\mathbf{P}$, and the amount of zero padding used $\mathbf{P}$ on the border. You can convince yourself that the correct formula for calculating how many neurons "fit" is given by $(\mathbf{W} - \mathbf{F} + 2\mathbf{P})/\mathbf{S} + 1$.

A parameter sharing scheme is used in Convolutional Layers to control the number of parameters. It turns out that we can dramatically reduce the number of parameters by making one reasonable assumption: That if one feature is useful to compute at some spatial position (x,y), then it should also be useful to compute at a different position (x2,y2). In other words, denoting a single 2-dimensional slice of depth as a depth slice, we are going to constrain the neurons in each depth slice to use the same weights and bias. In practice during backpropagation, every neuron in the volume will compute the gradient for its weights, but these gradients will be added up across each depth slice and only update a single set of weights per slice. Notice that if all neurons in a single depth slice are using the same weight vector, then the forward pass of the Conv layer can in each depth slice be computed as a convolution of the neuron's weights with the input volume –hence the name: Convolutional Layer. This is why it is common to refer to the sets of weights as a filter, or a kernel, that is convolved with the input.

2.3.3 Pooling Layer

It is common to periodically insert a Pooling layer in-between successive Conv layers in a ConvNet architecture. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation. Here we should say that the most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 downsamples every depth slice in the input by 2 along both width and height, discarding 75% of the activations. Every MAX operation would in this case be taking a max over 4 numbers, a little 2x2 region in some depth slice. The depth dimension remains unchanged. Concretely, the pooling layer accepts a volume of size $\mathbf{W_1} \times \mathbf{H_1} \times \mathbf{D_1}$, requires the hyperparameters $\mathbf{F}$ which is their spatial extent, and the stride $\mathbf{S}$, produces a volume of size $\mathbf{W_2} \times \mathbf{H_2} \times \mathbf{D_2}$ where: $\mathbf{W_2} = (\mathbf{W_1} - \mathbf{F})/\mathbf{S} + 1$, $\mathbf{H_2} = (\mathbf{H_1} - \mathbf{F})/\mathbf{S} + 1$, and $\mathbf{D_2} = \mathbf{D_1}$, and introduces zero parameters since it computes a fixed function of the input. 2.4

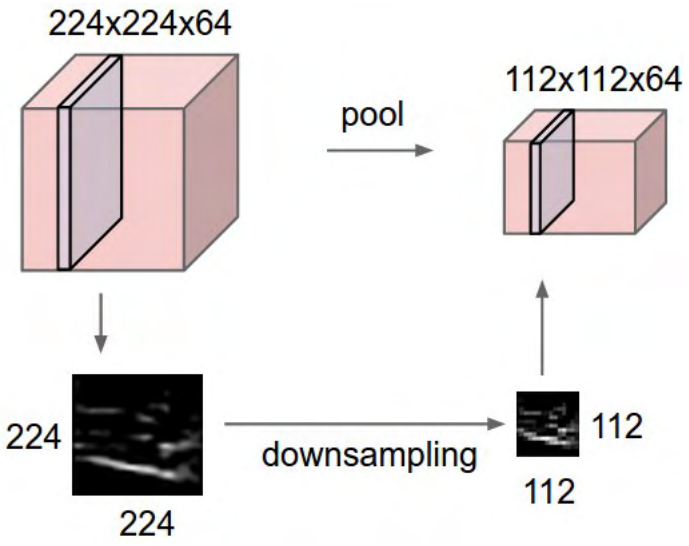

FIGURE 2.4: An input volume of size [224x224x64] is pooled with filter size 2 and stride 2 into the output volume of size [112x112x64]. The volume depth is preserved. (Image from Stanford CS231n: Convolutional Neural Networks)

In addition to max pooling, the pooling units can also perform other functions, such as average pooling or even L2-norm pooling. Average pooling was often used historically but has recently fallen out of favor compared to the max pooling operation, which has been shown to work better in practice.

Chapter 3

Translation Initiation Site prediction

3.1 Overview

The TIS prediction project was designed and implemented with modern software engineering practices in mind. Furthermore, a special care was taken to produce a completely reproducible implementation, which is a prerequisite for scientific progress.

To this end, a significant part of the project was the design and implementation of scripts to automatically download and process genomic data and generate the required datasets used to train and test the neural network models. The other major task was the experimentation with multiple neural network architectures and the search in the hyperparameter space to reach a trained model that generalizes over the data while avoiding overfitting.

3.2 Dataset

The dataset used for the implementation of the TIS prediction programs was generated by the Human genome produced and hosted by the Ensembl genome database project (Zerbino et al., 2018), a joint scientific project between the European Bioinformatics Institute and the Wellcome Trust Sanger Institute. The complete set of the transcription sequences from the "Human genes (GRCh38.p12)" dataset of the "Ensembl Genes 92" database from the Ensembl BioMart service was downloaded programmatically using an auxiliary library that consumes the BioMart's API.

Two sets of examples were required to train the models: the positive and the negative set. The positive set was constructed by downloading the cDNA transcripts that code for protein, along with their annotation metadata. During filtering, the transcripts with APPRIS annotation, and with Transcript support level (TSL) equal to "tsl1", "tsl2", or "tsl3" were selected. A single transcript for each unique gene was randomly selected.

The negatives examples where generated from long intergening non-coding RNA (lincRNA) transcripts of the dataset. During filtering, the transcripts with a GENCODE basic annotation were selected. A single transcript for each unique gene was randomly selected. Additionally, for each lincRNA transcript, all of the open reading frames of the sequence were selected and those with length less than a specified minimum were filtered out and one ORF from the remaining was randomly selected. The exclusion of sequences shorter than the minimum length was necessary to allow for the selection of the model features, 12 for the preceding sequence to the start codon of the Ribosome Signal feature and a variable number for the maximum length of the Coding Potential window feature.

3.2.1 Training and Testing sets

In the first prototype build of the project, we used the labels of 1 and -1 to represent the positive and the negative examples respectively. In the later and currenct implementation, the more general representation of an one hot array was used. The positive and negative examples were then merged and randomly shuffled, before splitting the resulting dataset to a training and a testing set, with 70% and 30% of the examples respectively.

3.3 Features

Two features were used for the creation of the Machine Learning model built. The first, called Ribosome Signal, represents the statistical significance of the coding potential around the start codon of the sequence. A 18-length nucleotide window was extracted from each sequence, consisting of the -12 to +9 nucleotides surrounding the start codon (itself starting at +1). The second feature, called Coding Potential, represents the statistical significance of the coding potential of the conserved motif of the sequence. It utilizes a scanning window that starts 60 nucleotides downstream of the start codon.

The nucleotide sequence windows extracted for the two features were concatenated to a single nucleotide array. In the prototype implementation, the nucleotides of the sequence (A, T, C, G) were mapped to the integers (1, 2, 3, 4). This numeric representation enabled their subsequent transformation to NumPy arrays in order to achieve efficient computations and utilization throughout the Machine Learning pipeline. In the latter and current implementation, to achieve the same goal, we used the representation of one hot arrays, to same as the label data.

3.4 Model Architecture

Two architectures classes of architectures were tested and experimented with in order to achieve the highest prediction accuracy for the target at hand. Both were Neural Network algorithms, using the high-level Keras API (Chollet et al., 2015) running on top of the TensorFlow Machine Learning framework (Abadi et al., 2015). We should note here that an initial implementation that utilized Support Vector Machines (SVM) was glued together but the subsequent neural network implementation was superior in every regard.

3.4.1 Fully-connected Neural Network

The first class of Neural Network architectures that was tested were fully connected networks. Trying to avoid giving additional information to the network by generating special features, simple sequential fully connected architectures were tested, with a varying number of layers and number of neurons for each of them. Increasing the number of layers lead to an increased efficiency of the neural network. The optimal number of hidden layers were 12, each them with 128 neurons, apart from the first and last neural network with 256 and 8 neurons respectively. 3.1

The neural network achieves an accuracy of **0.8660** with testing loss of **0.3161** and Area Under Curve (AUC) of **0.906124**. These metrics fluctuate to a small amount between runs, due to the randomized process of selecting the examples from the initial dataset.

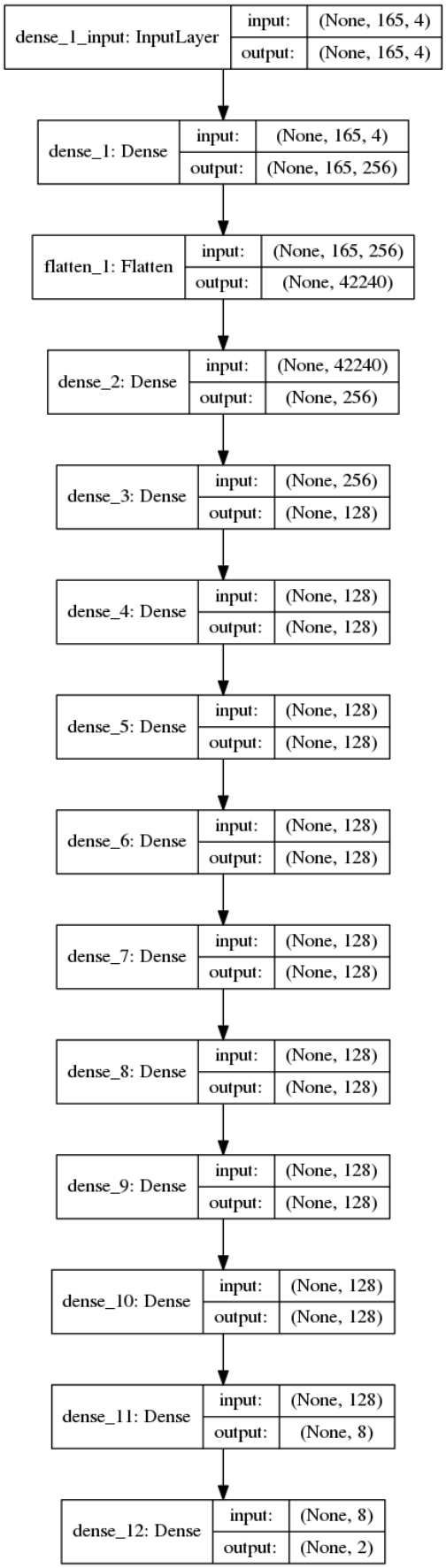

FIGURE 3.1: The best performing fully-connected TIS prediction neural network.

Figure 3.2 shows the training history of the neural network, while figure 3.3 shows the Receiver Operating Characteristic (ROC) curve of the neural network.

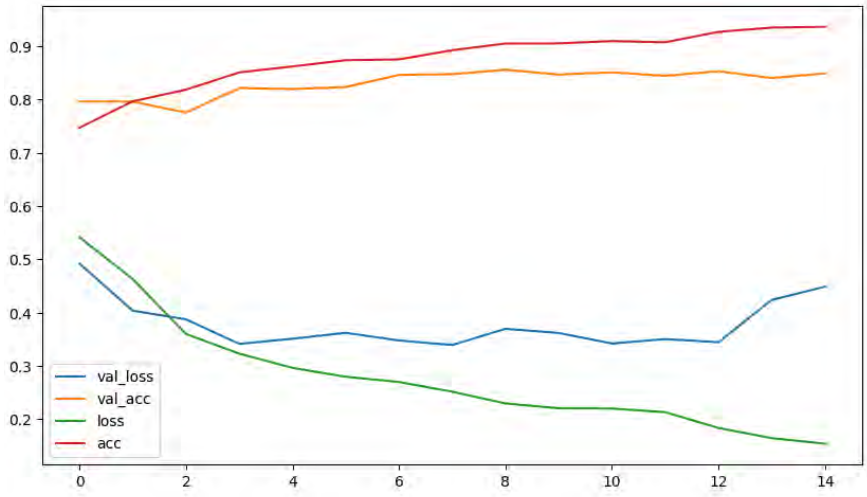

FIGURE 3.2: Training history of the fully-connected TIS prediction NN.

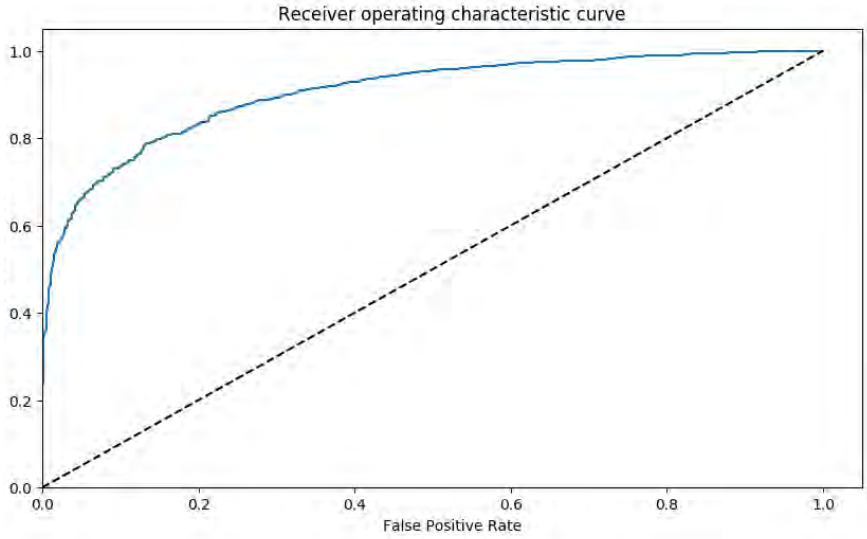

FIGURE 3.3: Training history of the fully-connected TIS prediction NN.

3.4.2 Convolutional Neural Network

The second class of Neural Network architectures that was tested were Convolutional Neural Networks. With the convolutional layer of a CNN, we hope to capture the information of the locality between the nucleotides. Due to the way the translation of the mRNAs is achieved by the ribosome, the exact sequence of the nucleotides is expected to play a significant role to whether an mRNA transcript is coding or non-coding. We adopted the same

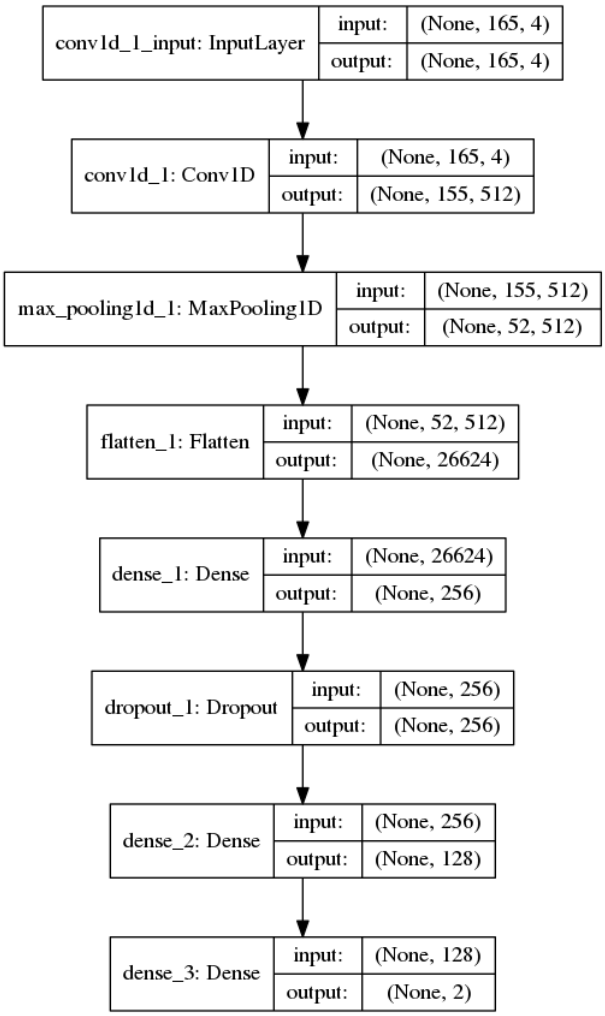

FIGURE 3.4: The best performing fully-connected TIS prediction neural network.

approach that is frequently utilized in CNNs for image classification, that uses an initial convolutional layer, a subsequent pooling layer and then a varying number of fully connected layers. 3.4

The neural network achieves an accuracy of **0.8625** with testing loss of **0.5871** and Area Under Curve (AUC) of **0.879967**. These metrics fluctuate to a small amount, due to the randomized process of selecting the examples from the initial dataset.

Figure 3.5 shows the training history of the neural network, while figure 3.6 shows the Receiver Operating Characteristic (ROC) curve of the neural network.

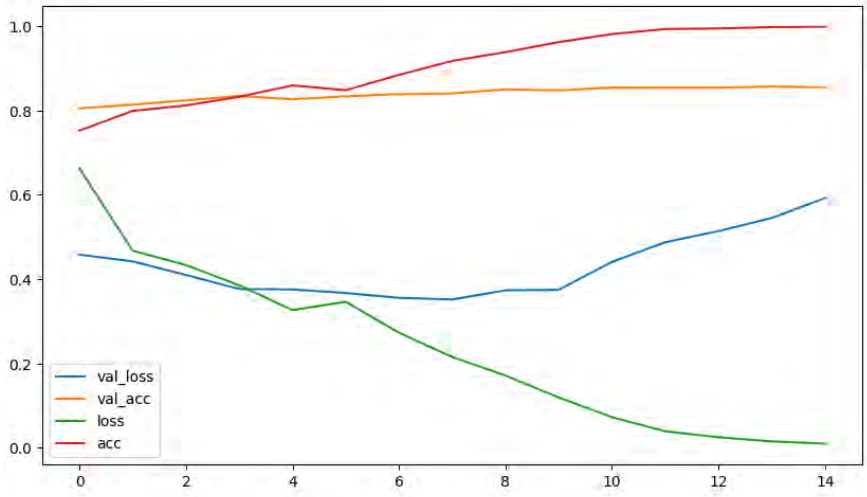

FIGURE 3.5: Training history of the fully-connected TIS prediction NN.

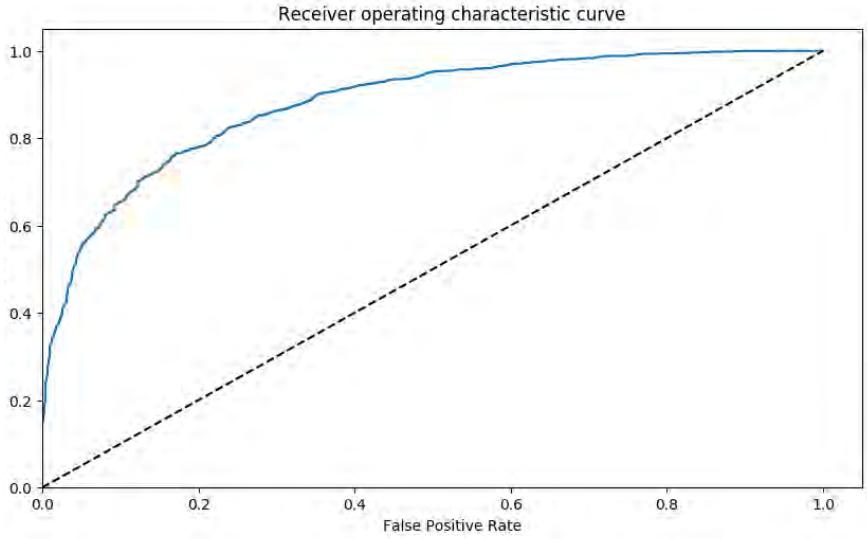

FIGURE 3.6: Training history of the fully-connected TIS prediction NN.

3.5 Conclusion and future work

In this project, we showed how a neural network adapted to the task of Translation Initiation Site prediction can achieve excellent results, that render it useful to practical implementations in genome sequencing and database annotation.

The two classes of neural networks achieved very similar levels of accuracy. A convolutional neural network would be expected to outperform a fully-connected one in cases where the sequences to be classified would be too large to construct a big enough fully-connected NN. In our case and the task at hand, and since the CNN model utilizes an information lossy procedure, the fully connected deep NN is preferred, as it places no limits to further improvements and extension.

3.5.1 Future work

There are two distinct places where there is opportunity of improvement in the predictive power of the classification program.

The first lies in the preparation of the dataset and, specifically, the selection of the training examples and the model features. The former is the idea that the dataset can be augmented by expanding the sets of positive and negative examples. Both sets can be expanded by including all the transcripts for each unique gene. The negative example set can be further expanded by including open reading frames from the untranslated regions of the genome, as well as start codon sites, close to true Translation Initiation Sites, that are not themselves true TIS. This is expected to provide an additional nuance to the training data that will increase the specificity of the network.

The second and most important idea for improving the efficiency of the program, is adopting a Recurrent Neural Network (RNN) architecture for the neural network. The attribute of RNNs of storing information in their internal memory units, makes them a very good candidate of modelling the actual behavior of the ribosome, learning the sequence of distinct steps that lead to a successful translation of coding mRNAs to proteins.

Appendix A

Codon table

Standard genetic code

1st base	2nd base T		2nd base C		2nd base A		2nd base G		3rd base
T	TTT	(Phe/F) Phenylalanine	TCT	(Ser/S) Serine	TAT	(Tyr/Y) Tyrosine	TGT	(Cys/C) Cysteine	T
	TTC		TCC		TAC		TGC		C
	TTA	(Leu/L) Leucine	TCA		TAA	Stop	TGA	Stop	A
	TTG		TCG		TAG	Stop	TGG	(Trp/W) Tryptophan	G
C	CTT		CCT	(Pro/P) Proline	CAT	(His/H) Histidine	CGT	(Arg/R) Arginine	T
	CTC		CCC		CAC		CGC		C
	CTA		CCA		CAA	(Gln/Q) Glutamine	CGA		A
	CTG		CCG		CAG		CGG		G
A	ATT	(Ile/I) Isoleucine	ACT	(Thr/T) Threonine	AAT	(Asn/N) Asparagine	AGT	(Ser/S) Serine	T
	ATC		ACC		AAC		AGC		C
	ATA		ACA		AAA	(Lys/K) Lysine	AGA	(Arg/R) Arginine	A
	ATG[A]	(Met/M) Methionine	ACG		AAG		AGG		G
G	GTT	(Val/V) Valine	GCT	(Ala/A) Alanine	GAT	(Asp/D) Aspartic acid	GGT	(Gly/G) Glycine	T
	GTC		GCC		GAC		GGC		C
	GTA		GCA		GAA	(Glu/E) Glutamic acid	GGA		A
	GTG		GCG		GAG		GGG		G

Amino acids biochemical properties: nonpolar | polar | basic | acidic

Termination: stop codon

FIGURE A.1: Codon table, the standard genetic code. (Template:Codon table, Wikipedia)

Appendix B

Project setup and reproduction

B.1 Python

We use pyenv to install Python, which is a simple and powerful Python version manager, that lets us install the specific version of Python required for this project without interfering with the system's Python installation.

Install pyenv using pyenv-installer:

```
curl -L https://github.com/pyenv/pyenv-installer/raw/master/bin/pyenv-installer | bash
```

Add the following lines to your .profile and reboot, or logout and start a new session:

```
### pyenv
export PATH="$HOME/.pyenv/bin:$PATH"
eval "$(pyenv init -)"
eval "$(pyenv virtualenv-init -)"
```

Run the following command to install any missing requirements for building Python:

```
sudo apt install make build-essential libssl-dev zlib1g-dev libbz2-dev \
    libreadline-dev libsqlite3-dev wget curl llvm libncurses5-dev \
    libncursesw5-dev xz-utils tk-dev libffi-dev
```

Install Python version 3.6.6:

```
pyenv install 3.6.6
```

B.2 pipenv

For package management, we use pipenv, a package and virtual environment management tool, that brings the best of packaging tools for multiple programming languages to Python.

Install pipenv with pip, that is included in the Python installation:

```
pip install pipenv
```

Clone the project git repository, hosted at GitHub.

Move into the project directory and use pipenv to create a virtual environment and install the dependencies:

```
cd TIS_prediction/
pipenv install
```

Bibliography

Abadi, Martín et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: https://www.tensorflow.org/.

Alberts, Bruce et al. (2014). *Molecular Biology of the Cell, Sixth Edition*. http://books.wwnorton.com/books/Molecular-Biology-of-the-Cell. Garland Science.

Alpaydin, Ethem (2010). *Introduction to machine learning*. Cambridge, Mass: MIT Press. ISBN: 978-0-262-01243-0.

Bicknell, Alicia A. et al. (2012). "Introns in UTRs: Why We Should Stop Ignoring Them". In: *BioEssays* 34.12, pp. 1025–1034. ISSN: 1521-1878. DOI: 10.1002/bies.201200073. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/bies.201200073 (visited on 08/20/2018).

Bishop, Christopher (2006). *Pattern recognition and machine learning*. New York: Springer. ISBN: 978-0-387-31073-2.

Cenik, Can et al. (2015). "Integrative Analysis of RNA, Translation, and Protein Levels Reveals Distinct Regulatory Variation across Humans". In: *Genome Research* 25.11, pp. 1610–1621. ISSN: 1088-9051. DOI: 10.1101/gr.193342.115. pmid: 26297486. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4617958/ (visited on 08/19/2018).

Chollet, François et al. (2015). *Keras*. https://keras.io.

Fitz, Veronika et al. (2016). "Nucleosomal Arrangement Affects Single-Molecule Transcription Dynamics". In: *Proceedings of the National Academy of Sciences* 113.45, pp. 12733–12738. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.1602764113. pmid: 27791062. URL: http://www.pnas.org/content/113/45/12733 (visited on 08/20/2018).

Goldman, Seth R., Richard H. Ebright, and Bryce E. Nickels (2009). "Direct Detection of Abortive RNA Transcripts in Vivo". In: *Science* 324.5929, pp. 927–928. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.1169237. pmid: 19443781. URL: http://science.sciencemag.org/content/324/5929/927 (visited on 08/20/2018).

Griffiths, Anthony (2008). *Introduction to genetic analysis*. New York: W.H. Freeman and Co. ISBN: 978-0-7167-6887-6.

Jacob, François and Jacques Monod (1961). "Genetic Regulatory Mechanisms in the Synthesis of Proteins". In: *Journal of Molecular Biology* 3.3, pp. 318–356. ISSN: 0022-2836. DOI: 10.1016/S0022-2836(61)80072-7. URL: http://www.sciencedirect.com/science/article/pii/S0022283661800727 (visited on 08/20/2018).

Kozak, M. (1987). "An Analysis of 5'-Noncoding Sequences from 699 Vertebrate Messenger RNAs". In: *Nucleic Acids Research* 15.20, pp. 8125–8148. ISSN: 0305-1048. pmid: 3313277.

Lander, E. S., [...], and International Human Genome Sequencing Consortium (2001). "Initial Sequencing and Analysis of the Human Genome". In: *Nature* 409.6822, pp. 860–921. ISSN: 0028-0836. DOI: 10.1038/35057062. pmid: 11237011.

Lykke-Andersen, Søren and Torben Heick Jensen (2007). "Overlapping Pathways Dictate Termination of RNA Polymerase II Transcription". In: *Biochimie*. Functional diversity of RNA 89.10, pp. 1177–1182. ISSN: 0300-9084. DOI: 10.1016/j.biochi.2007.05.007. URL: http://www.sciencedirect.com/science/article/pii/S0300908407001307 (visited on 08/20/2018).

Mandal, Subhrangsu S. et al. (2004). "Functional Interactions of RNA-Capping Enzyme with Factors That Positively and Negatively Regulate Promoter Escape by RNA Polymerase II". In: *Proceedings of the National Academy of Sciences* 101.20, pp. 7572–7577. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.0401493101. pmid: 15136722. URL: http://www.pnas.org/content/101/20/7572 (visited on 08/20/2018).

Maston, Glenn A., Sara K. Evans, and Michael R. Green (2006). "Transcriptional Regulatory Elements in the Human Genome". In: *Annual Review of Genomics and Human Genetics* 7.1, pp. 29–59. ISSN: 1527-8204. DOI: 10.1146/annurev.genom.7.080505.115623. URL: https://www.annualreviews.org/doi/10.1146/annurev.genom.7.080505.115623 (visited on 08/20/2018).

Moghal, Adil, Kyle Mohler, and Michael Ibba (2014). "Mistranslation of the Genetic Code". In: *FEBS letters* 588.23, pp. 4305–4310. ISSN: 0014-5793. DOI: 10.1016/j.febslet.2014.08.035. pmid: 25220850. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4254111/ (visited on 08/19/2018).

Mortazavi, Ali et al. (2008). "Mapping and Quantifying Mammalian Transcriptomes by RNA-Seq". In: *Nature Methods* 5.7, pp. 621–628. ISSN: 1548-7105. DOI: 10.1038/nmeth.1226. URL: https://www.nature.com/articles/nmeth.1226 (visited on 08/20/2018).

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Richardson, John P (2002). "Rho-Dependent Termination and ATPases in Transcript Termination". In: *Biochimica et Biophysica Acta (BBA) - Gene Structure and Expression*. Transcription Elongation Control-2002 1577.2, pp. 251–260. ISSN: 0167-4781. DOI: 10.1016/S0167-4781(02)00456-6. URL: http://www.sciencedirect.com/science/article/pii/S0167478102004566 (visited on 08/20/2018).

Roeder, Robert G. (1991). "The Complexities of Eukaryotic Transcription Initiation: Regulation of Preinitiation Complex Assembly". In: *Trends in Biochemical Sciences* 16, pp. 402–408. ISSN: 0968-0004. DOI: 10.1016/0968-0004(91)90164-Q. URL: http://www.sciencedirect.com/science/article/pii/096800049190164Q (visited on 08/20/2018).

Ross, J F and M Orlowski (1982). "Growth-Rate-Dependent Adjustment of Ribosome Function in Chemostat-Grown Cells of the Fungus Mucor Racemosus." In: *Journal of Bacteriology* 149.2, pp. 650–653. ISSN: 0021-9193. pmid: 6799491. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC216554/ (visited on 08/19/2018).

Schueren, Fabian and Sven Thoms (2016). "Functional Translational Readthrough: A Systems Biology Perspective". In: *PLoS Genetics* 12.8. ISSN: 1553-7390. DOI: 10.1371/journal.pgen.1006196. pmid: 27490485. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4973966/ (visited on 08/19/2018).

Watson, James (2008). *Molecular biology of the gene*. San Francisco Cold Spring Harbor, N.Y: Pearson/Benjamin Cummings Cold Spring Harbor Laboratory Press. ISBN: 978-0805395921.

Wohlgemuth, Ingo et al. (2011). "Evolutionary Optimization of Speed and Accuracy of Decoding on the Ribosome". In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 366.1580, pp. 2979–2986. ISSN: 0962-8436. DOI: 10.1098/rstb.2011.0138. pmid: 21930591. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3158919/ (visited on 08/19/2018).

Xing, Jianhua et al. (2006). "Torque–Speed Relationship of the Bacterial Flagellar Motor". In: *Proceedings of the National Academy of Sciences* 103.5, pp. 1260–1265. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.0507959103. pmid: 16432218. URL: http://www.pnas.org/content/103/5/1260 (visited on 08/29/2018).

Zerbino, Daniel R et al. (2018). "Ensembl 2018". In: *Nucleic Acids Research* 46.D1, pp. D754–D761. DOI: 10.1093/nar/gkx1098. eprint: /oup/backfile/content_public/journal/nar/46/d1/10.1093_nar_gkx1098/2/gkx1098.pdf. URL: http://dx.doi.org/10.1093/nar/gkx1098.