Automatic Intron Detection in Metagenomes Using Neural Networks

Bc. Martin Indra

Faculty of Electrical Engineering

Czech Technical University in Prague

Winter Term 2020/2021

Acknowledgements

I would like to thank my supervisor, doc. Ing. Jiří Kléma, Ph.D., for giving me the opportunity to work on this interesting and valuable project and for all his support and guidance.

I would also like to thank Ing. Anh Vu Le for providing me with a lot of valuable data and insights into an existing intron detection pipeline based on support vector machines.

In addition, I would like to thank my wife who supported me during the difficult times.

Abstract / Abstrakt

This work is concerned with the detection of introns in metagenomes with deep neural networks. Exact biological mechanisms of intron recognition and splicing are not fully known yet and their automated detection has remained unresolved.
Detection and removal of introns from DNA sequences is important for the identification of genes in metagenomes and for searching for homologs among the known DNA sequences available in public databases. Gene prediction and the discovery of their homologs allows the identification of known and new species and their taxonomic classification.
Two neural network models were developed as part of this thesis. The models' aim is the detection of intron starts and ends with the so-called donor and acceptor splice sites. The splice sites are later combined into candidate introns which are further filtered by a simple score-based overlap resolving algorithm. The work relates to an existing solution based on support vector machines (SVM). The resulting neural networks achieve better results than SVM and require more than order of magnitude less computational resources in order to process equally large genome.

Keywords: fungi, fungal genomes, neural networks, itron detection, metagenome

Tato práce se zabývá detekcí intronů v metagenomech hub pomocí hlubokých neuronových sítí. Přesné biologické mechanizmy rozpoznávání a vyřezávání intronů nejsou zatím plně známy a jejich strojová detekce není považovaná za vyřešený problém.
Rozpoznávání a vyřezávání intronů z DNA sekvencí je důležité pro identifikaci genů v metagenomech a hledání jejich homologií mezi známými DNA sekvencemi, které jsou dostupné ve veřejných databázích. Rozpoznání genů a nalezení jejich případných homologů umožňuje identifikaci jak již známých tak i nových druhů a jejich taxonomické zařazení.
V rámci práce vznikly dva modely neuronových sítí, které detekují začátky a konce intronů, takzvaná donorová a akceptorová místa sestřihu. Detekovaná místa sestřihu jsou následně zkombinována do kandidátních intronů. Překrývající se kandidátní introny jsou poté odstraněny pomocí jednoduchého skórovacího algoritmu.
Práce navazuje na existující řešení, které využívá metody podpůrných vektorů (SVM). Výsledné neuronové sítě dosahují lepších výsledků než SVM a to při více než desetinásobně nižším výpočetním čase na zpracování stejně obsáhlého genomu.

Klíčová slova: genom hub, neuronové sítě, detekce intronů, metagenom

Čestné prohlášení

Prohlašuji, že jsem zadanou diplomovou práci zpracoval sám s přispěním vedoucího práce a konzultanta a používal jsem pouze literaturu v práci uvedenou. Dále prohlašuji, že nemám námitek proti půjčování nebo zveřejňování mé diplomové práce nebo její části se souhlasem katedry.

....	....
datum	podpis diplomanta

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Indra** Jméno: **Martin** Osobní číslo: **393127**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Bioinformatika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Automatická detekce intronů v metagenomech pomocí neuronových sítí.

Název diplomové práce anglicky:

Automatic intron detection in metagenomes using neural networks.

Pokyny pro vypracování:

1. Seznamte se s obecnou strukturou DNA eukaryot, prostudujte formáty jejich reprezentace v databázích Joint Genome Institute.
2. Vypracujte rešerši stávajícího použití konvolučních a rekurentních neuronových sítí při automatické segmentaci genomu.
3. Navrhněte a implementujte algoritmus pro detekci intronů v genomech hub.
4. Extrahujte introny ze stanovených genomů pomocí výše navrženého algoritmu, reportujte základní statistiky pro jednotlivé genomy, závěry zobecněte přes taxonomické třídy hub. Srovnejte je s existující metodou vycházející z detekce donorů a akceptorů pomocí SVM.
5. Diskutujte použitelnost algoritmu v úloze detekce intronů v metagenomech,
zohledněte zejména hledisko časové náročnosti.

Seznam doporučené literatury:

Lee, B., Lee, T., Na, B., &amp; Yoon, S. (2015). DNA-level splice junction prediction
using deep recurrent neural networks. arXiv preprint arXiv:1512.05135.
Naito, T. (2018): Human splice-site prediction with deep neural networks. Journal
of Computational Biology, pp. 954–961.
Nguyen, Ngoc Giang, et al. (2016): DNA sequence classification by convolutional
neural network. Journal of Biomedical Science and Engineering 9.05 280.

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Jiří Kléma, Ph.D., Intelligent Data Analysis FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **16.09.2020** Termín odevzdání diplomové práce: ______________

Platnost zadání diplomové práce: **19.02.2022**

doc. Ing. Jiří Kléma, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

.	
Datum převzetí zadání	Podpis studenta

Contents

List of Figures

List of Abbreviations

2D	Two dimensional, referring in this thesis to dimensions of data.
DNA	Deoxyribonucleic acid
RNA	Ribonucleic acid
CDS	Coding sequence.
UTR	Untranslated region
SGD	Stochastic gradient descent
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
RCNN	Recurrent Convolutional Neural Network
SVM	Support Vector Machine
ReLU	Rectified Linear Unit
LSTM	Long short-term memory unit in Recurrent Neural Network
BLAST	Basic Local Alignment Search Tool
ROC	Receiver Operating Characteristic
AUC	Area under the curve

1 Introduction

Contents

The kingdom of fungi is largely undiscovered and comprises 2.2–3.8 million species [1]. Only a small fraction of this—around 100 000 species—has been described in the literature [2]. The kingdom plays a significant role in public health, food biosecurity, and biodiversity [3]. It is, therefore, important to develop new, computer-aided methods to bridge the gap between what is known and what is unknown.

A gene is a sequence of DNA nucleotides encoding a gene product. A gene product is a protein or form of RNA. In most organisms, the DNA or RNA sequence that encodes the product is not continuous but interleaved with introns, which are spliced out during gene expression. The remaining parts are called exons [4].

Metagenomics is the study of combined genetic material as retrieved directly from a natural environment. The collection of genetic material retrieved from a sample is known as metagenome, which, as opposed to a single genome, consist of the DNA sequences of multiple organisms.

Gene homology refers to the similarity between the genetic sequences of two genes due to shared ancestry. Two genes with similar DNA sequences and shared ancestry are called homologous. The identification of homologous genes can be used in recognizing individual organisms or kinships in the metagenome.

In a study, forest soil samples were extracted and their genetic material sequenced. Traditional sequence alignment methods were utilized for gene prediction within the metagenome, leading to the identification of a number of fungal species within the sample. But the sample was significantly smaller than expectations based on the estimates of a number of existing fungal species. The present work seeks to improve the results of the gene prediction procedure. For this, it identifies and excises introns from the metagenome and keeps only evolutionarily more preserved exons in the sequences.

The goal of this work is to create a multistep software pipeline that identifies introns in DNA sequences of potentially unknown fungal organisms. The use of deep neural networks is elaborated. Their performances and computational resource usage are compared with previously used methods based on support vector machines and the potential of using the results in a larger gene prediction pipeline.

The work uses a large preexisting annotated fungal DNA dataset. The solution is subdivided into the following sub-goals:

- detection of candidate splice site positions with the search for consensus dinucleotides,
- classification of candidate intron start positions, i.e. donor splice sites,
- classification of candidate intron end positions, i.e. acceptor splice sites,
- combining detected splice sites into intron start-end pairs based on their mutual distance,
- filtering the intron candidates resulting from the previous step by an overlap-resolving algorithm.

1.1 Text Structure

This chapter (1) introduces the work by briefly explaining several important biological concepts and puts the work into its context. Chapter 2 explains biological phenomena related to the work, some concepts and techniques from computer science–basics of recurrent and convolutional neural networks (RCNN) and statistical measures used for evaluation. Chapter 3 talks about further motivation for this work, and Chapter 4 summarizes current research on the topic. Chapter 5 contains an overview of the used data, the statistics computed on the data, their source, and the used file types.

Chapter 6 describes the recurrent convolutional networks used for splice site classification. It also describes the architecture and training of the networks with theoretical and practical reasoning. Chapter 7 talks about how data pre-processing and RCNN training and evaluation were automated for fast and reproducible experimentation. And finally, Chapter 8 talks about the evaluation and achieved results.

Chapter 9 concludes the work and outlines possible further work.

2 Background

Contents

The nucleic DNA molecules or chromosomes of Eukaryotic organisms play many roles in cell biology. Different positions (loci) in a chromosome play different, often overlapping and complex, roles [5]. DNA functions and its structure are not yet fully understood [6].

The previously mentioned complexity opens up the possibility of novel approaches like state-of-the-art machine learning techniques in order to study the DNA structure, its roles and functions, and the detection of already known parts in new DNA sequences. For example, it has been shown that feed-forward neural networks can represent a wide variety of functions [7]. The versatility and effectiveness of artificial neural networks has been practically demonstrated in many fields. Examples of this are general game play [8], face recognition [9], speech recognition [10], or medical image classification [11], among others.

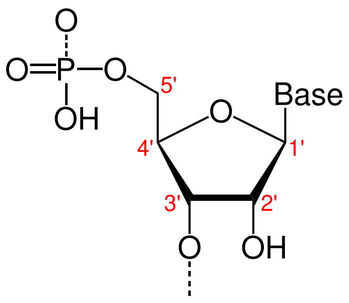

Figure 2.1: Molecular structure of a nucleotide [15]

2.1 DNA and RNA Structure

Polynucleotide is a linear chain of up to 20 different nucleotide monomers joined by the phosphodiester (covalent) bond [12, p. 308, p. 347]. Deoxyribonucleic acid (DNA) and ribonucleic acid (RNA) are two types of polynucleotides that are abundant in natural life. Both have important biological functions. DNA is composed of individual nucleotides Adenine (A), Thymine (T), Cytosine (C), and Guanine (G) [13, p. 4, p. 20, p. 107]. RNA is made from nucleotides Adenine (A), Uracil (U), Cytosine (C), and Guanine (G) [14, p. 11]. RNA molecules first appeared around four billion years ago as a first form of life [12, p. 412].

The DNA molecule is directional due to asymmetry of individual nucleotides [13, p. 42]. See Figure 2.1. Based on The chemical convention of naming carbon atoms in the nucleotide sugar-ring, one side of DNA is called 5'-end and the other side 3'-end. DNA and RNA are synthesized in the 5' to 3' direction [13, p. 167, p. 728]. When referring to relative positions in a DNA sequence, upstream and downstream refer to the 5' and the 3' directions respectively. Similarly, by convention, DNA sequences are usually written and stored in the 5' to 3' direction, unless explicitly stated or needed otherwise.

Chromosomes are enormous DNA molecules which encode the majority of genetic information in fungi and other kingdoms of species. DNA in chromosomes consists of two coiled chains of polynucleotide strands forming a double helix.

Different parts of a chromosome have different functions in a cell. Within chromosomes are continuous parts which form genes, which are nucleotide sequences that encode gene products—either RNA or protein. Genes have a complex structure

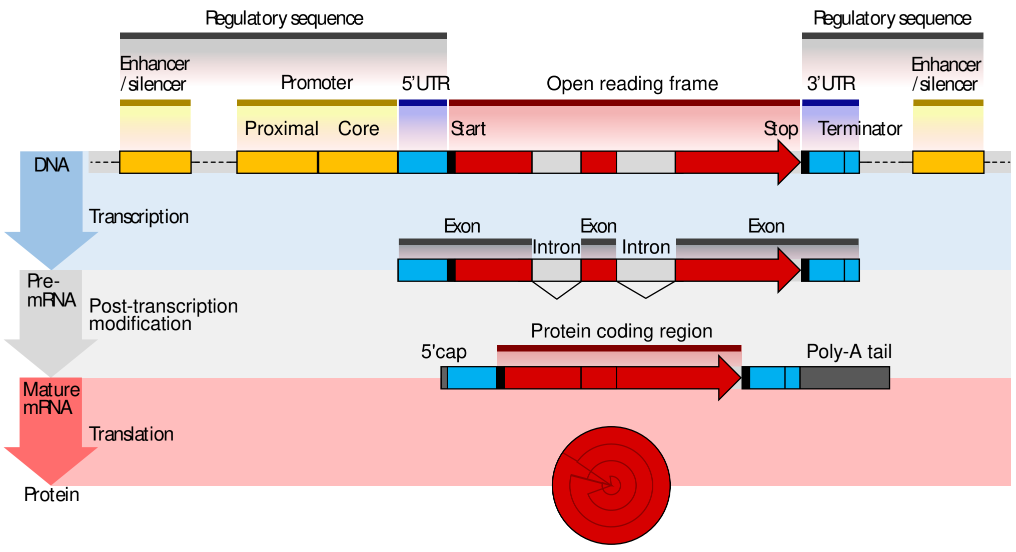

Figure 2.2: Gene structure and processing [17]

that contains regulatory sequences, exons, introns, and other areas. The regulatory sequences of a gene, located at the extremities of the gene, contain a promoter at the 5' side and a terminator at the 3' side of the gene. The promoter and terminator mark the beginning and end of the transcribed region of the gene respectively.

In eucaryotic organisms, gene transcription forms a primary transcript, alternatively called precursor mRNA or Pre-mRNA, which consists of exons and introns and lacks 5' cap and poly(A) tail. 5' cap and poly(A) tail are added at later stages of gene expression [16]. Afterward, introns are spliced out during the post-transcriptional modification and the remaining exons form a final mature mRNA that, in some cases, encodes protein. Both ends of mRNA contain untranslated regions (UTR) 5'-UTR and 3'-UTR enclosing the final protein coding region [17]. The gene structure and processing are illustrated in Figure 2.2.

Within introns, the donor splice site, branch point, and the acceptor splice site are used for splicing. The donor splice site lies at the 5'-end of the intron, the acceptor splice site lies at the 3'-end of the intron, and the branch point lies 18–40 nucleotides upstream from the acceptor site [18]. In the great majority of the cases, the 5'-end of an intron starts with highly preserved `GU` nucleotides and the 3'-end of an intron ends with `AG` nucleotides. The highly preserved dinucleotides, which make first two symbols in an intron (at 5'-end) and last two symbols in an

```
A-TATCATGA
AGTA-CATGG
```

Figure 2.3: Global sequence alignment example

```
ATTATCATGA
  TA-CA
```

Figure 2.4: An example of a sequence (second row) locally aligned to a larger sequence (first row)

intron (at 3'-end), are commonly referred to as consensus dinucleotides. The branch point always contains adenine, but it is otherwise more variable [18]. Cell's splicing machinery is called spliceosomes and removes introns in a sequence of complex steps. The exact end-to-end mechanisms are not yet fully understood [18].

2.2 Biological Sequence Alignment and Search

Sequence alignment is an algorithm which searches for the best scoring alignment of two or more sequences of symbols by inserting gaps into any of the sequences. The resulting alignment contains indels (insertions and deletions), mismatches, and matches. The alignments are usually scored by summing penalties for indels with (mis)match scores at other positions. The (mis)match scores are taken from a substitution matrix and the gaps are frequently scored with a gap-open penalty summed up with a penalty proportional to the gap length. An example pairwise alignment of two sequences is given in Figure 2.3.

Two types of alignments exist: global and local. A global alignment produces sequences of equal length, while a local alignment produces an alignment only of parts of the sequences. See Figure 2.4 with an example of a local pairwise alignment.

There exist exact algorithms like the Needleman-Wunsch algorithm for global alignments [19] and the Smith-Waterman algorithm for local alignments [20]. The exact algorithms have large asymptotic time and memory complexity. An optimized version of the Needleman-Wunsch algorithm has $\mathcal{O}(mn/\log(n))$ asymptotic time complexity, where m and n are lengths of the sequences [21, p. 35]. The asymptotic time complexity of the Smith-Waterman algorithm is $\mathcal{O}(mn)$ [21, p. 40]. This makes their usage for search in large databases unfeasible because their asymptotic time has to be further multiplied by the number of database entries.

For reasons of efficiency, heuristic sequence alignment algorithms are in wide use. These algorithms do not guarantee optimal results but have a much shorter execution time and smaller memory usage. Among the heuristic algorithms is BLAST (basic local alignment search tool), one of the most used algorithms for sequence searching [22].

E-value is the expected number of coincidental matches in the database of a given size with the match score equal or greater than the found score [21, p. 119]. In other words, a large E-value signifies a high probability that a match is only coincidental.

When searching for a sequence in a database, a local alignment, such as BLAST, is performed against all available sequences. All matches with a high enough score giving a sufficiently small E-value are reported as results.

2.3 Neural Networks

The artificial neural network is a network of interconnected artificial neurons. The activation y_q (output) of a simple artificial neuron, q, can be described with Formula 2.1, where $g : \mathbb{R} \to \mathbb{R}$ is the neuron activation function, x_i is either one of the neural network inputs or the activation of the preceding neuron i, $w_{iq} \in \mathbb{R}$ is the weight of the connection from neuron i to neuron q, and $b_q \in \mathbb{R}$ is the bias of neuron q. The weights and biases are learned during ANN training.

$$y_q = g\left(\sum (x_i \cdot w_{iq}) - b_q\right) \tag{2.1}$$

The architecture of a neural network consists of the connections between individual neurons and activation functions, which are specified as meta-parameters [23, p. 193]. Most neural networks could be organized into layers. The neurons in each layer connect only to neurons from preceding layers [23, p. 193].

A neural network is considered deep if it consists of more than three levels of compositions of non-linear operations [24, p. 6]. Therefore, a neural network structured into layers is deep if it comprises more than one hidden layer.

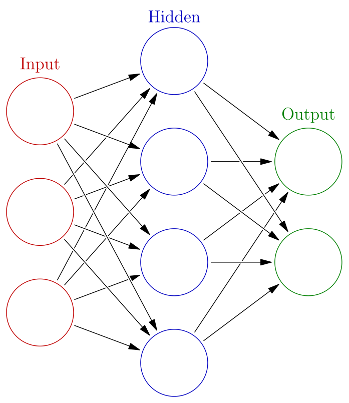

Figure 2.5: Feed forward artificial neural network [25]

Training algorithms, often called neural network optimizers, try to minimize the value of the loss function on training data. The loss function measures how well the network performs in each individual training sample, for example, the difference between truth and prediction.

Neural networks are usually trained with a variation of stochastic gradient descent (SGD) or a derived algorithm [23, p. 149]. SGD repeatedly and randomly selects several training samples called a mini-batch and computes the loss function gradient with respect to trained parameters (i.e. weights and biases) with the back propagation algorithm [23, p. 149]. Backpropagation is an algorithm which computes the gradient of a loss function with respect to network parameters with application of the chain rule [23, p. 201]. A multiple, which is called the learning rate, of the calculated gradient is then subtracted from the parameters [23, p. 150].

A common pattern in the architecture of binary classification neural networks is to have a single output neuron, with an appropriate activation function and a discrimination threshold on output neuron activation.

Convolutional neural networks (CNNs) are a class of neural networks that contain one or more convolution layers (see Figure 2.6). Convolution layers contain neurons with shared weights and a limited perceptive field; the weights of the neurons are shift invariant [23, p. 326]. Thanks to its features, CNNs greatly reduce a number of learnable parameters and are therefore easier to train and less prone to overfitting [23, p. 339].

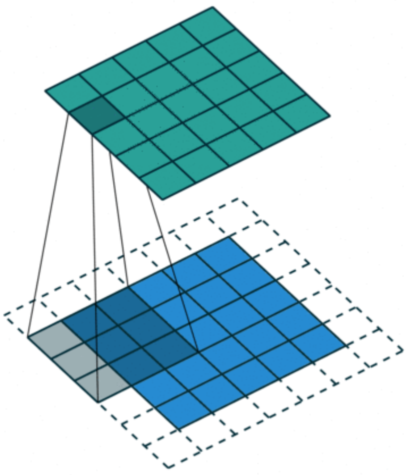

Figure 2.6: Convolutional layer (top) over preceding layer (bottom) [26]

Formula 2.2 provides the activation of neurons in a 2D convolutional layer, where $y_{i,j}$ is the activation of the neuron at position (i, j) in a 2D grid of neurons, $g : \mathbb{R} \to \mathbb{R}$ is the activation function, $k_{m,n} \in \mathbb{R}$ is a shared weight from kernel $k \in \mathbb{R}^{M \times N}$, $x_{i+m,j+n}$ is the output of neuron $(i + m, j + n)$ from the preceding 2D layer, and $b \in \mathbb{R}$ is the shared bias [23, p. 328].

$$y_{i,j} = g\left(\sum_m \sum_n k_{m,n} \cdot x_{i+m,j+n} - b\right) \tag{2.2}$$

Recurrent neural networks (RNNs) are a family of neural networks where the connections between nodes form a directed graph along a sequence [23, p. 368]. In other words, the computational graph of these network contains loops that unfold over a time variable. This architecture allows for the processing of possibly arbitrarily long sequences of data [23, p. 367].

2.4 Evaluation

The four following basic metrics can be estimated for any binary classification algorithm with respect to truth data:

- true positive rate (TPR) – probability of a positive sample being classified as positive by the classification algorithm,

- true negative rate (TNR) – probability of a negative sample being classified as negative,
- false positive rate (FPR) – probability of a negative sample being classified as positive,
- false negative rate (FNR) – probability of a positive sample being classified as negative.

These constants are estimated with formulas $TPR = \frac{TP}{P}$, $TNR = \frac{TN}{N}$, $FPR = \frac{FP}{N}$ and $FNR = \frac{FN}{P}$, where TP, TN, FP, FN, P, and N are the number of true positives, true negatives, false positive, false negatives, the number of positive samples, and the number of negative samples, respectively, and obtained by running the algorithm on a test dataset.

Many other metrics could be derived from these constants.

2.4.1 Accuracy

The accuracy of an algorithm is the probability of a sample being classified correctly, given by Formula 2.3, where p is the prior probability of the positive class.

$$Accuracy = TPR \cdot p + TNR \cdot (1 - p) \tag{2.3}$$

This metric is of limited use in case of highly unbalanced data. For example, an algorithm classifying all samples as negative would have 99% accuracy on data with 99% of the samples being negative.

2.4.2 Precision and Recall

Precision is a fraction of the true positive classifications in all positive classifications given by Formula 2.4, where p is the prior probability of the positive class.

$$Precision = \frac{TPR \cdot p}{TPR \cdot p + FPR \cdot (1 - p)} \tag{2.4}$$

Recall is a fraction of the positive samples classified as positive. Recall is independent of class prior probabilities. See Formula 2.5.

$$Recall = \frac{TPR}{TPR + FNR} = TPR \tag{2.5}$$

Both precision and recall might be calculated directly from TP, FP, TN, FN counts measured on a test dataset. In such a scenario, calculated precision would differ from 2.4 if a fraction of positive sample was different from positive class prior p.

2.4.3 Area Under Curve

The area under curve (AUC) is an area under a receiver characteristic curve (ROC). An ROC curve gives dependence of a true positive rate on the false positive rate obtained on the different threshold settings of a binary classification algorithm.

3 Motivation

DNA sequencing costs are sharply decreasing [27]. GenBank is a genetic sequence database of all publicly available DNA sequences [28]. The number of bases in the GenBank database has doubled approximately every 18 months from 1982 to present (2019), and its database consists of over 366 billion bases [29]. The consequence of ever-cheaper DNA sequencing and exponential sequence database growth is the need for an increased capacity for DNA data handling and analysis. An automated and scalable solution to DNA annotation would partially fulfill this need.

Fungal and bacterial organisms play an important role in various ecosystems including the floor and soil of forests [30][31]. Some fungal species are capable of decomposing cellulose and various biopolymers and are involved in the decomposition of deadwood and litter. The decomposition goes in stages, and in each stage different organisms with varying diversity contribute to the process [31]. Nontrivial dependencies of different fungal organisms and decomposition stages were identified and a lot of the relationships are yet to be known [31]. Decomposed wood and litter is important for some plant species [31]. The presence of wood inhabiting fungi might be a good indicator of overall forest biodiversity [30].

Not only for the previously mentioned reasons, the ability to identify fungal species in metagenomes recovered from soil and other environmental samples is a potent tool in the biological study of forests and other ecosystems. The task of fungal species identification in a metagenome is approached by searching for homologous genes in the DNA sequences by applying sequence alignment algorithms to all known protein and gene sequence databases. The search for homologs does not only allow the identification of known species, but also searches for novel, evolutionary related species.

To achieve good search results, we detect and remove introns in long DNA sequence scaffolds before the search. This should improve protein sequence match scores because introns are spliced before RNA to protein translation. A reason for the improvement is that the match score of alignment algorithms for homologous DNA sequences is the highest in most preserved parts of such DNA sequences—i.e. exons. The functional regions of DNA sequences tend to be more highly conserved than the remaining parts of the DNA [32]. See Section 2.2 for more information on sequence alignment.

4
Current Research

Research on automated splice site detection goes back to at least 1987. For example, work [33] used the probability of occurrence of particular nucleic bases at particular positions for splice site detection.

The hidden Markov model and the AdaBoost classifier to detect an intron splice site were used in [34]. The hidden Markov model was also used in [35] for sequence pre-processing and the support vector machine for intron splice site detection. A multilayered recurrent neural network applied on triplets from original sequence is described in [36]. A convolutional neural network and its interpretation is described in [37], which reports that the network is most sensitive to nucleotides close to a splice site.

A paper [38] reports good performance of recurrent convolutional neural networks, known as DeeperBind, in detecting the protein binding motifs in DNA sequences. DeeperBind encodes DNA sequences with one-hot-encoding. The network consists of one convolutional layer, followed by two LSTM layers, and finally, the LSTM output is connected to two fully connected layers with dropout. The convolutional layers are not followed by max pooling layers common in CNN architectures. Strides of convolution is set to one. The paper reports that the following LSTM layers are able to deal with increased redundancy produced with convolutional layers with stride one and without max pooling.

The paper [39] takes a different approach to RNA sequence encoding. The authors report diminished generalization when learning one-hot-encoded sequences and overcoming this issue by embedding each nucleotide to a four-dimensional dense vector. The input is followed by two RNN layers whose output is fed to a fully connected layer with dropout. The paper compares the performances of ReLU, LSTM, and GRU recurrent units. It is concluded that the addition of more RNN layers does not further improve the performance and that LSTM units show the best performance.

Article [40] describes the use of recurrent convolutional neural networks for splice site detection in the human genome. Two convolutional layers, each succeeded by a max-pooling layer, are followed by a bidirectional LSTM recurrent layer. The output of the RNN layer is fed to two fully connected layers with soft-max at the final output. Dropout is applied on the outputs of all layers. One-hot-encoding is used on the inputs. The paper reports improved performance of the network with the RNN layer as compared with the same network without that layer.

Past research is mostly based on classical methods; the use of deep neural networks for splice site detection is rare. Some methods show almost perfect performance [36]. However, these methods are usually trained and tested on a single or a few organisms—for example, Homo Sapiens—and training-testing data is split on the level of individual samples as opposed to organism or higher taxonomic ranks. The ability of these algorithms to generalize to the DNA sequences of species not present in training data is largely unreported.

It has been described that the splice site recognition process is complex and tissue-specific in humans [41]. This complexity may limit the maximum achievable accuracy of splice site detection algorithms that work with DNA sequences only on out-of-sample organisms.

Convolutional neural networks are used in work [42] for predicting the discrete probability distribution of distances and torsions of amino acid residues based on pre-processed protein sequences. The neural network is structurally similar to the networks used in image-recognition tasks. The predicted distances were used as a seed for a gradient-descent algorithm that predicts the three-dimensional structure of a protein by minimizing the potential. The paper reports the state-of-the-art performance and demonstrates the ability of deep neural networks to predict the physical shapes of biological polymers.

The specifics of fungal DNA are exploited in the thesis. The ability of various neural networks to generalize across various taxonomic ranks is studied. The sheer amount of available annotated data enables the use of more complex neural networks, which, in turn, can learn more subtle or complex features.

5 Data

Contents

Source DNA sequence data and its annotations were downloaded from the Joint Genome Institute[1] [43]. The data contains FASTA files with the DNA sequence scaffolds of individual organisms and GFF files with DNA feature annotations.

As part of this work, all data was uploaded to Google Cloud Storage in a compressed form, and an automated download and extraction script was created.

5.1 File Formats

A FASTA file is a text-based representation of DNA sequences. Such a file contains header lines beginning with character `>`, followed by an identifier of the sequence on successive lines. The sequence itself consists of characters `ATCGN` that represent adenine, thymine, cytosine, guanine, and "any character" respectively. In the data, each sequence within a FASTA file represents a scaffold.

[1]https://jgi.doe.gov/fungi

The sequences in FASTA files represent scaffolds. A scaffold links together a non-contiguous series of genomic sequences, comprising sequences separated by gaps of known length. The linked sequences are typically contiguous sequences corresponding to read overlaps [44].

The following sample is the first four lines of the FASTA file with the DNA sequence of the organism Verticillium dahliae.

```
>Supercontig_1.1
AGTATCATGAAGGAAGAACAAGTTGAGGGACATAATTACCTGGGGTGCGGCGCTTACAAGTAAGGGTCGC
TGGGACATCGACCTGGAGGAGGAGAATCATGTAACGCCCCAGCCCGGTCGTCACCAGGACACCAGGCAGG
ACACCCCGCAGGCGATCGGACGCGCCGCACGGACCCACAGGATCACTCACGTGACCGTGACCAGATCACG
```

General Feature Format (GFF) is a file with DNA feature annotations. The file format has three versions. The latest version 3 is used in the data. GFF is a text-based file in which each line represents a feature annotation. An annotation consists of nine tab-delimited attributes [45]:

- sequence – name of the sequence, scaffold in our case, where the sequence is located,
- source – source of the feature, for example, name an institution,
- feature – type of the feature, only feature type `exon` are used in this work,
- start – 1-based, inclusive offset of start of the feature,
- end – 1-based, inclusive offset of end of the feature,
- score – confidence in validity of the feature,
- strand – DNA strand where the feature is located. It is one of `+`, `-` or `.` for positive, negative, and undetermined respectively,
- phase – phase of CDS features which is always one of 0, 1 or 2,
- attributes – additional feature attributes. Format of this field is not generally determined. In GFF files used in this work, it contains colon-delimited list of attributes, where each is space-delimited attribute name and value.

Phylum	Number of Organisms
Ascomycota	479
Basidiomycota	295
Blastocladiomycota	4
Cryptomycota	1
Chytridiomycota	21
Microsporidia	8
Mucoromycota	38
Zoopagomycota	16

Table 5.1: Phyla in the dataset

Taxonomy Rank	Number of Taxa
Phylum	8
Class	46
Order	124
Family	307
Genus	566
Species	862

Table 5.2: Number of taxa on different taxonomy ranks

The following sample is the first four lines of a GFF file with annotations of the DNA sequence of the organism Verticillium dahliae.

```
Supercontig_1   JGI exon    76  572 .   +   .   name "VDBG_00001T0"; transcriptId 224
Supercontig_1   JGI CDS 406 572 .   +   0   name "VDBG_00001T0"; proteinId 1; exonNumber 1
Supercontig_1   JGI start_codon 406 408 .   +   0   name "VDBG_00001T0"
Supercontig_1   JGI exon    631 1621    .   +   .   name "VDBG_00001T0"; transcriptId 224
```

5.2 Taxonomy

The data consists of eight different phyla, but 89.8% of the organisms (774 of 862) are from Ascomycota and Basidiomycota phyla which make the Dikarya subkingdom. See Table 5.1 with the number of organisms per phylum and Figure 5.1 with a phylogenetic tree of available fungi.

5.3 Data Statistics

In total, 16 067 492 introns, distributed over 5 557 272 individual genes ,were identified from exon annotations on all 862 organisms in the dataset. This number

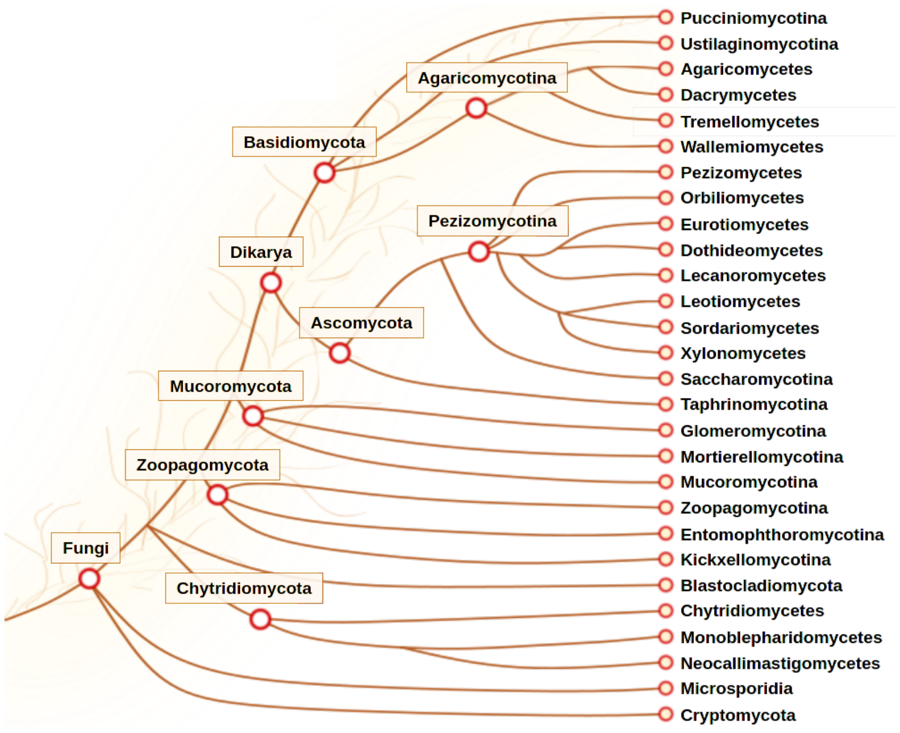

Figure 5.1: Phylogenetic tree of fungi with sequenced genomes as displayed on the website of Joint Genome Institute [43]

includes only introns inside coding sequences, see Section 7.2. Also see Section 7.1 which includes the definition of gene used in this thesis.

The cumulative length of all genes is 8 396 757 084 or 8 371 133 149, with overlaps counted only once. The cumulative length of all introns is 1 296 667 885 or 1 293 833 629, with counting overlaps only once or roughly 15% of genes.

The mean gene length is 1510.9 nucleotides and the mean intron length is 80.7 nucleotides. As much as 93.5% of the introns are 150 nucleotides long or less with a distinct peek in length distribution around 50 nucleotides. See Figure 5.2 with intron length distribution and Figure 5.3 that depicts a box plot of intron lengths on all available phyla.

In this work, only donors and acceptors with consensus dinucleotides were used, see Section 6.2 for more information on this topic.

In total, 15 792 942 donors and 15 828 432 acceptors were identified in the data. All occurrences of `GT` for donors and `AG` for acceptors at positions different from the positions of true splice sites were considered as false donors and false acceptors.

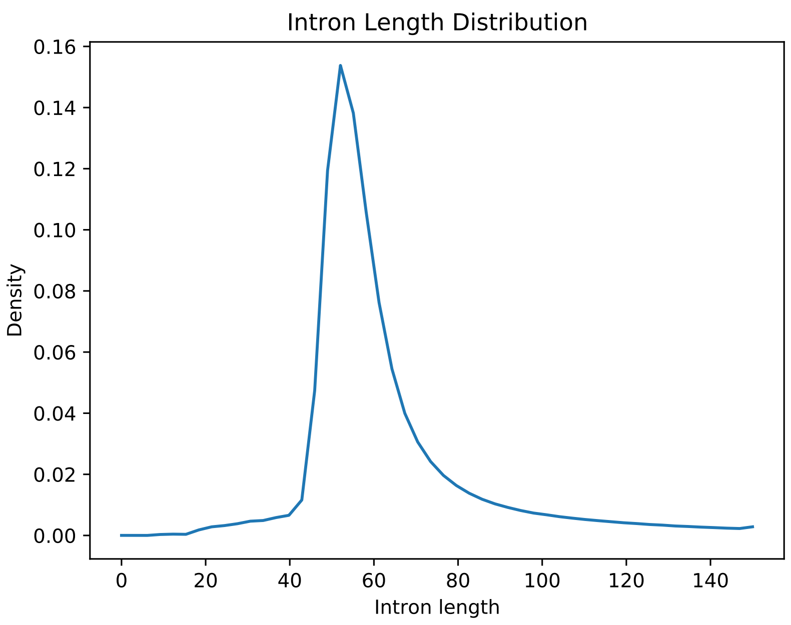

Figure 5.2: Distribution of intron length over all organisms in the dataset

Moreover, 411 393 734 false donors and 498 273 594 false acceptors were found inside genes—this gives the frequency of one false donor per 20.3 nucleotides and one false acceptor per 16.8 nucleotides. False donors are 26.0 times more frequent than true donors, and false acceptors are 31.2 times more frequent than true acceptors in exon-intron areas.

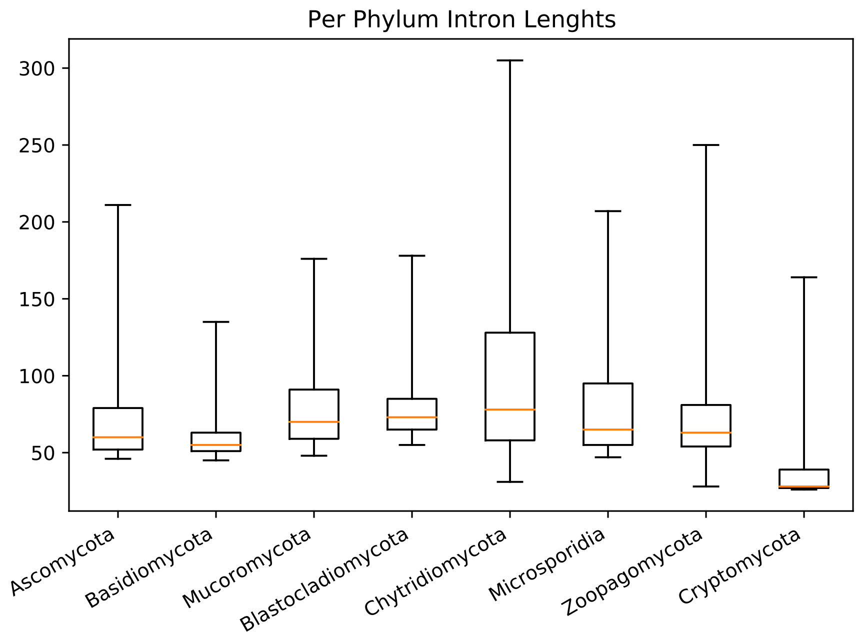

Figure 5.3: Box plot with intron lengths on all phyla. Whiskers extend to 5% and 95% percentiles.

6 Recurrent Convolutional Neural Networks

Contents

This work aims to develop a method based on deep neural networks which would detect and remove introns from the DNA sequences of various fungal organisms.

Removal of introns is done in these steps:

1. all candidate donor and acceptor splice sites are identified based on consensus dinucleotides,
2. these candidate splice sites are classified as true and false splice sites with separate donor and acceptor recurrent neural networks,
3. positive classifications are combined to form candidate introns,
4. candidate introns are assigned a score equal to the multiple of the respective donor and acceptor splice site model confidence,

5. and overlapping candidate introns with non-highest score are filtered out.

Section 6.1 gives an overview of the neural networks used and the overall pipeline. Section 6.2 talks about the selection of candidate splice sites—i.e. locations in source DNA sequences (from a metagenome) which are considered as potential donors and acceptors. These locations are then classified with the splice site networks. Section 6.3 describes how the DNA sequences are encoded in matrices so that they could be used as inputs to the neural networks. Section 6.4 talks about the architecture and other aspects of the splice site classification neural network. Section 6.5 defines the criteria used during network selection. Finally, Section 6.6 describes the training of the neural networks.

6.1 Overview

All critical sections in the aforementioned intron detection and deletion of the pipeline—i.e. two neural networks—are created as part of this work. An existing pipeline from work [46] (which originally used the support vector machines in the critical section) was adapted to obtain a complete pipeline.

Two separately trained neural networks with equal architecture were used for the classification of donor and acceptor splice sites within original DNA sequences (i.e. in full DNA sequence and before transcription to RNA). To accompany the complexity of all regulatory and other sequences involved in intron splicing, convolutional layers were utilized in target neural networks. Recurrent layers with long short-term memory (LSMT) units were used to enable the detection of partially shift invariant inter-dependencies and redundancies between non-adjacent parts of the intron sequences.

The neural networks were trained to distinguish between true and false splice sites. The input to the neural network is a window spanning some distance to both sides around the candidate splice site. The window size has been selected to be large enough so that the network can detect all important features within the DNA sequence. The output of the network is the confidence of the input sample being a true splice site.

The pipeline reported in [46] requires a third model which further filters candidate introns. The need for this additional step was motivated by the high false positive rate of splice site classification models. In this work, the additional filtering step was assessed as counterproductive because splice site classification models based on neural networks have a smaller false positive rate and further filtering of introns would impact the overall intron recall. See Section 8.3 for evaluation metrics.

6.2 Candidate Splice Site Selection

Only splice sites with the consensus dinucleotides were used during the training, evaluation, and testing of the networks. These are `GT` for donor and `AG` for acceptor splice sites. This decision was made because non-consensus splice sites are rare, and therefore, there was not enough training and evaluation data to be included. In fact, its inclusion would drastically increase the number of false positive classifications. It omission has only a minuscule negative effect on the number of false negative classifications. See Section 5.3 which contains more related statistics.

False splice sites included in training, validation, and test datasets are selected only from the gene areas in source DNA sequences. See Section 7.1 for the definition of “gene” used in this thesis. Filtering to genes is motivated by the fact that intra-genetic DNA does not play an important role during sequence alignment and homologous gene search in the larger gene prediction pipeline. Therefore, removal of presumed introns from these intra-genetic areas should not have a large impact on overall results. Only the ability to recognize false splice site within genetic areas was assumed to play an important role. Furthermore, sequences which would be biologically processed as introns might be present in intra-genetic areas and are therefore labeled as negative examples. This would lead to confusion of the trained network as well as a worse performance.

6.3 Sequence Encoding

Splice site classification neural networks were trained to map an input sequence window to a value between 0 and 1. The sequence is encoded as a matrix of dimensions $N \times 5$, where the N rows represent relative positions along the DNA sequence and the columns represent one-hot-encoded nucleotides. Various input window sizes $N = N_{upstream} + N_{downstream}$ were evaluated, see Table 6.1 and Table 6.2. Each nucleotide is encoded as a 5-dimensional vector, with value 1.0 at the dimension of the represented nucleotide and values 0.0 at other dimensions. The first four dimensions represent the nucleotides adenine (A), thymine (T), cytosine (C), and guanine (G). The fifth dimension represents "any symbol" (usually missing or corrupt data).

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.1}$$

Matrix 6.1 illustrates a six-nucleotide long, one-hot-encoded DNA sequence window that reads AGTNAA, which contains the consensus donor dinucleotide GT at position 1.

6.4 Neural Network Architecture

This section reports selected splice site classification network architecture, see Figure 6.1 and explains the theoretical background and motivation behind the input and each selected layer, as well as the empirical findings.

Table 6.1 and Table 6.2 show the dependency between the shape of the input window to the neural network and its performance. The tables clearly demonstrate that the window overlap with the associated intron – located at the downstream direction for donors and upstream direction for acceptors – plays a dominant role and that classifications performed on windows with no overlap with the associated

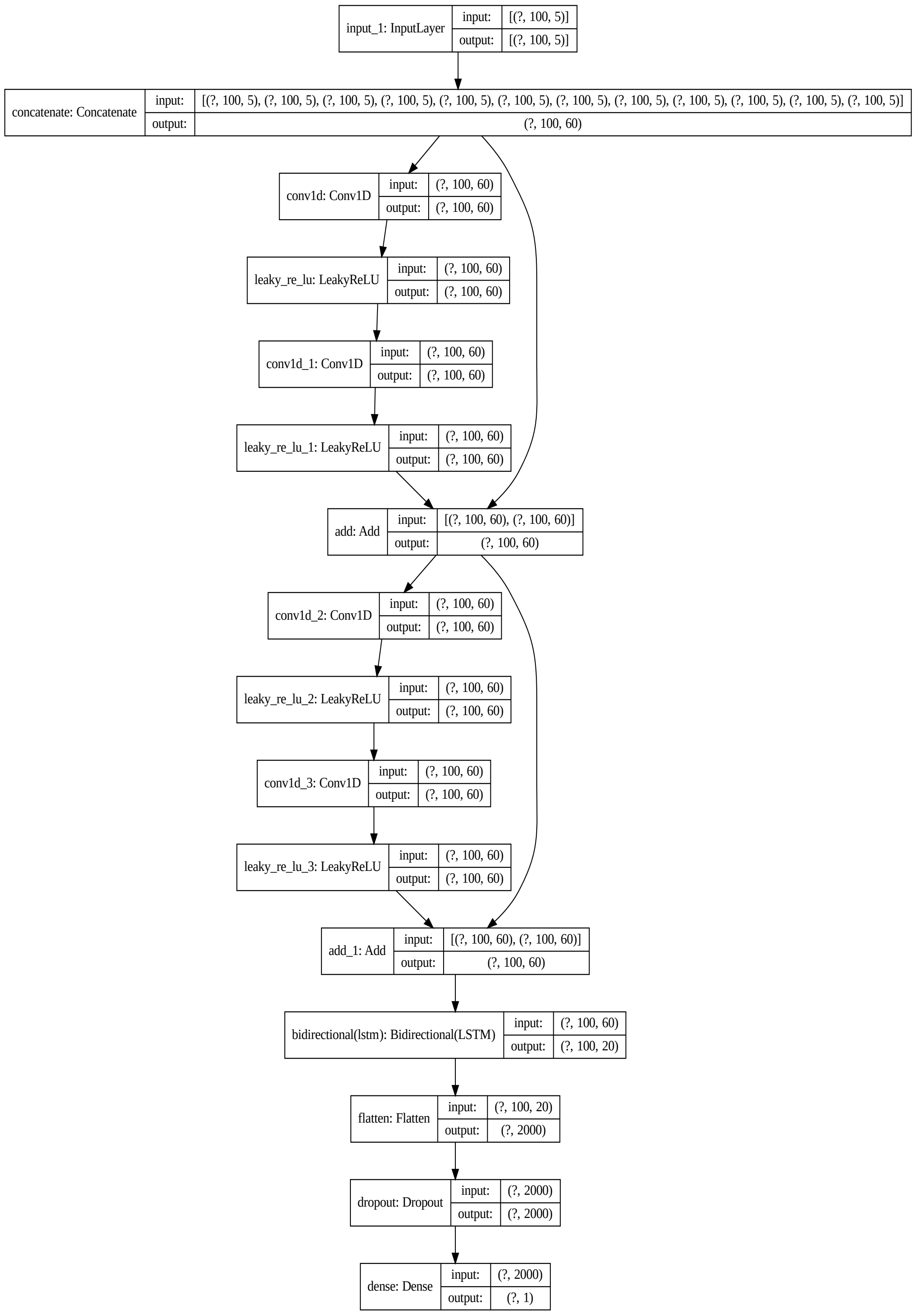

Figure 6.1: Model architecture visualization

	0		**100**		**200**	
	Precision	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
0	-	-	51.8%	85.2%	52.2%	85.4%
100	14.3%	32.2%	62.4%	87.8%	63.3%	88.0%
200	14.6%	34.0%	63.6%	88.1%	64.6%	87.9%

Table 6.1: Dependency between classification performance on donors and the number of nucleotides upstream (rows) and downstream (columns) from the candidate splice site included in the input window.

	0		**100**		**200**	
	Precision	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
0	-	-	10.2%	22.1%	10.4%	24.3%
100	53.1%	86.3%	57.7%	87.2%	56.6%	87.2%
200	56.1%	86.4%	58.6%	87.3%	58.8%	87.4%

Table 6.2: Dependency between classification performance on acceptors and the number of nucleotides upstream (rows) and downstream (columns) from the candidate splice site included in the input window.

intron have very poor performance. This finding is in agreement with the biological understanding of splicing, which is mostly dependent on DNA sequences inside the intron. See Section 2.1 which covers the biological background and Section 8.5 which explores the sensitivity of the trained network at various input positions.

The convolutional layer allows for sparse connectivity between successive layers and parameter sharing by applying the same kernel parameters to a limited receptive field [23, p. 330]. This leads to drastic reduction of the number of trainable parameters which, in turn, decreases the computing time and needed training dataset size. Convolutions are equivariant to translation [23, p. 334] and are therefore theoretically capable of learning various features at different relative positions inside input DNA sequence windows from fewer samples. It is hypothesized that such position-independent features exist in the data, which motivated the use of convolutional layers in the resulting network architecture. It has been empirically verified that convolutional layers improved various network performance metrics at low additional computational costs.

Leaky rectified linear activation functions (Leaky ReLU) were used throughout the network. Use of the ReLU activation function and its variations has many benefits. ReLU was among the most frequently used and successful activation function as of 2017 [47]; it leads to extensive research on them. Networks using ReLU units are more easily trained because they do not have problems with vanishing and exploding gradients [47]. Leaky ReLU was used instead of plain ReLU because it provides similar computational complexity but avoids 0 gradient [48], thereby leading to improved performance.

Skip connections forming residential networks (ResNet) are introduced in the convolutional part of the network. Deep networks may suffer from the vanishing gradient and the degradation problem, which could be resolved by using ResNet [49].

Recurrent neural networks (RNNs) are capable of recognizing same features at various positions in the input sequence and even recognize their repetition and mutual dependence [23, p. 367]. To some extent, it is possible to achieve similar position-independent feature recognition with convolutional layers. However, their processing is shallower and disallows greater interdependence between different positions in the sequence [23, p. 368]. As of 2016, gated RNNs, which include networks with long short-term memory (LSTM) units, were the most effective type of RNNs [23, p. 404]. LSTM units allow the retention of information over large number of time steps without any problems with vanishing or exploding gradients [23, p. 404]. Other works also report the successful use of RNN layers for splice site detection [39]. For these reasons, a bidirectional RNN layer was used in the selected architecture.

The work [39] reports no improvement after adding more recurrent layers. The same finding of no statistically significant improvement after adding another recurrent layer was observed in this work.

The used recurrent layer produces output at each step. This decision has been motivated by the need to distinguish true splice sites that are located exactly in the middle of the input sequence from those located in close proximity in the middle. Bidirectional RNN was used to allow the recognition of both downstream and upstream dependencies.

The aforementioned multilayered architecture allows for the learning of complex, highly non-linear mappings between input DNA sequences and output splice site confidences. The network has 49 921 learnable parameters, which give large learning capacity. However, neural networks with large capacities tend to overfit the training data [50]. This effect was empirically observed and confirmed during experiments on neural network architectures without sufficient regularization techniques. A dropout layer was successfully used in the final architecture to completely overcome overfitting.

Dropout is a technique of omission of a given fraction of neural units in a given layer during training. A different stochastic unit selection is done during each training mini-batch. Classical optimization based on backpropagation is then applied to this reduced network [50]. All neurons are applied with a scaling factor during the later inference of a trained network. Dropout is an approximation of an equally weighted geometric mean of the predictions of multiple neural networks with shared parameters [50]. The number of dropouts and therefore the number of synthetic networks are exponential with the number of neural units [50]. This technique is highly efficient from the point of view of computational resources and prevents overfitting [50].

6.5 Model Criteria

Model selection was based on the following criteria:

- ability of the model to generalize and perform consistently among different fungal organisms,
- good performance metrics, namely its precision and recall,
- model complexity and computational intensity during inference.

The generalization capability of the model was an important aspect because the model would be applied to a wide variety of previously unknown fungal genomes. For this reason, all experimental models were evaluated on individual organisms separately as well as on larger sets of organisms. Consistency was taken into account in this regard.

Precision and recall were used to evaluate the performance of the models. Small false positive rate was emphasized during the development of the network architecture and the selection of binary classification threshold values. This is because consensus dinucleotides are more than an order of magnitude more frequent in target DNA sequences than true splice sites, see Section 5.3 for more details.

Final production computational resource utilization was also an important consideration during the design of the network architecture and pre-processing and post-processing procedures. This aspect is important because metagenomes on which they will be applied are enormous and might go well beyond 10^9 nucleotides. The possible use of the results of this work in an automated online annotation tool is another consideration, and high computational requirement would make such a goal economically unfeasible. This criterion has led to the selection of a sub-optimal network from the point of view of accuracy as networks with larger complexity and window size had slightly better results. But the improvement was not significant and would lead to a large increase in computational requirements.

6.6 Neural Network Training

Stochastic gradient descent (SGD) was used for the optimization during network training. Other optimization techniques, such as the adaptive methods AdaGrad, RMSProp, or Adam, were also tested, but SGD yielded superior results. It has been reported that adaptive methods converge faster during the initial phases of training; but given enough training time, it may lead to worse generalization [51]. This effect was empirically confirmed in this work.

The initial learning rate was set to $\alpha = 0.01$ and multiplied by the factor of 0.2 after every epoch which did not lead to a decrease in validation loss. The training was automatically stopped after 10 successive epochs with no significant decrease in validation loss. The model with the lowest after-epoch validation loss was used during successive evaluation and tests.

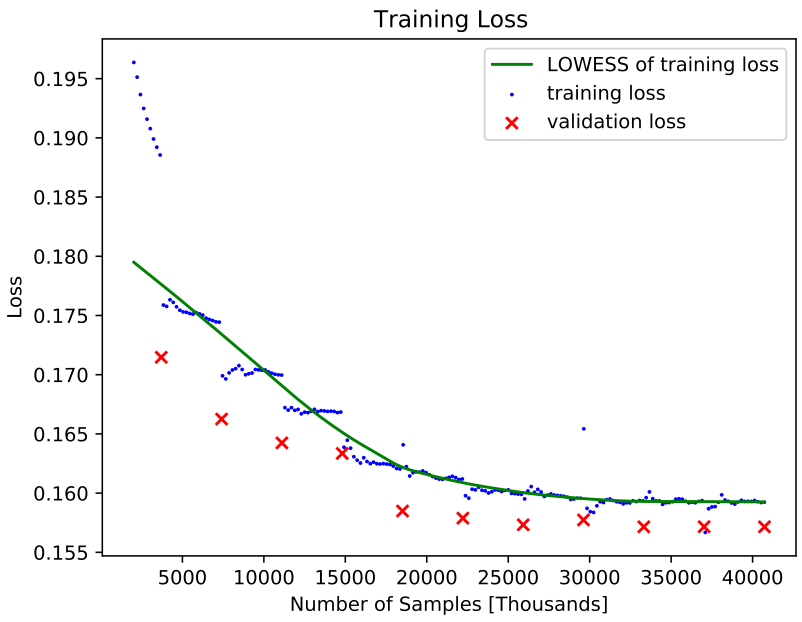

Figure 6.2: Training and validation loss during training of the donor model

Binary cross-entropy was chosen as a surrogate loss function.

It has been reported that small batch sizes ($m \leq 32$) lead to better test results and generalization. For some networks and tasks, this effect might go down to the batch size as small as $m = 2$ [52]. The batch size of $m = 16$ was used during training.

Over three million samples from 769 organisms were used in the training dataset and over 85 000 samples from 85 organisms were used in the validation dataset. See Section 6.7 that talks about validation dataset size selection. After each epoch, training samples were randomly reshuffled. Training and validation loss progression is illustrated by Figure 6.2.

Keras [53] with TensorFlow [54] backend was used for training and prediction. Preprocessed data in the form of Numpy NPZ files with input matrices and desired outputs is used during training, see Section 7.4. Data was loaded gradually from the disk and not kept in memory because of the large number of samples used during the training.

See Chapter 8 for detailed evaluation of achieved results.

6.7 Dataset Size

The validation dataset always consisted of 50% of negative samples and 50% of positive samples. The validation dataset size l was chosen so that the probability of empirical true positive rate (TPR) or empirical true negative rate (TNR) being more than ϵ, different from the true TPR or the true TNR in any of n experiments, was smaller or equal to R. See Section 2.4 for more information on evaluation.

Formula 6.2 gives the maximum probability R_1 of seeing a "bad" TPR or TNR measurement in a single evaluation for the above condition to hold. This is derived from the fact that $2 \cdot n$ measurements need to be performed to obtain empirical TPR and empirical TNR for n different model setups.

Formula 6.3 is derived from Hoeffding's inequality [55] and gives the minimum dataset size l. The right part of the equation is multiplied by 2 because the dataset is split into half between positive and negative samples.

A dataset comprising $l = 75759$ samples is needed for $n = 25$, $R = 0.05$ (5%) and $\epsilon = 0.01$ (1%).

$$R_1 = 1 - (1 - R)^{\frac{1}{2 \cdot n}} \tag{6.2}$$

$$l = \left\lceil 2 \cdot \frac{\log 2 - \log R_1}{2 \cdot \epsilon^2} \right\rceil \tag{6.3}$$

7 Automation

Contents

This chapter describes various automation software developed as part of this work, including data pre-processing and extraction, neural network training, and evaluation. It explains how the work was split into a multi-step pipeline and further describes some technical details that have implications on how the neural networks were trained or evaluated.

Source data, as described in Chapter 5, contains assembled DNA sequence scaffolds in FASTA files and various feature annotations in general feature format (GFF) files. A multistep data pre-processing pipeline was created to extract target training, evaluation, and testing data from the source data. This pipeline is split into multiple steps to reduce its complexity, increase the time-efficiency, and to enable efficient reproducibility with varying configuration parameters like extracted window size. A set of fully automated training, evaluation, inspection, and visualization scripts were also created.

Throughout the work, 0-based indexing is used for sequence positions and start inclusive, end exclusive intervals are used.

All source codes are published in GitHub repository `https://github.com/Indy2222/introns/`. Source data, trained neural networks, and other large binary materials are uploaded to Google Cloud Storage which is linked from the repository.

7.1 Overview

In the text of this chapter, in the source code and in file naming the term “gene” is often used for a continuous area withing a DNA sequence that spans from the beginning of the first annotated CDS to the end of the last annotated CDS of an actual gene. This is a simplification that does not fully correspond to actual genes in all their complexities as understood by biology.

The pre-processing pipeline consists of the following steps:

1. Within each organism, start and end positions of individual genes and introns are extracted. These are collectively referred to as feature, see Section 7.2.

2. Positions of positive and negative examples of donor and acceptor Splice sites are generated. See Section 7.3.

3. Final data samples, which map DNA sequence windows to scores 0.0 or 1.0, are created. See Section 7.4.

The data pre-processing pipeline is implemented in the Rust programming language for its high performance, reliability, and “fearless concurrency” [56][57]. All CPU-intensive parts of the pipeline are parallelized to improve the processing time on multicore computers.

Training, evaluation, inspection, and visualization are described in Section 7.5.

Diagram of the preprocessing, training and evaluation pipeline is depicted in Figure 7.1.

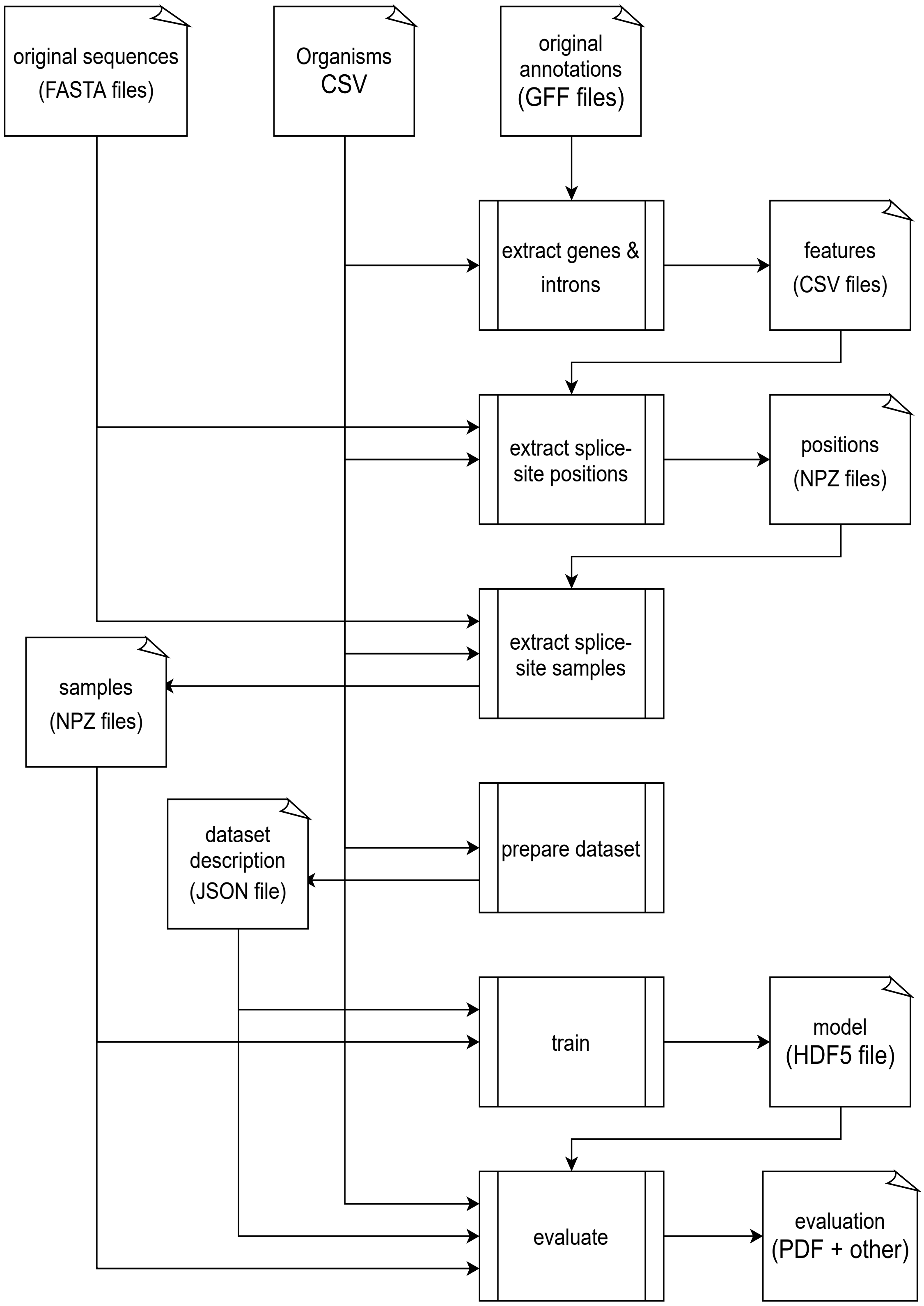

Figure 7.1: Diagram of data extraction and preprocessing, splice site classification model training and evaluation.

```
AGTATCATGAAGGAAGAACAAGTTGAGTGACATAATTACCAGGGGTGCGGCG
```

Figure 7.2: An illustration of a DNA sequence with UTR of exons highlighted in cyan, CDS highlighted in blue, intra-UTR introns (not extracted) highlighted in magenta and intra-CDS introns (extracted) highlighted in red.

7.2 Intron and Gene Extraction

Start and end positions of individual protein coding genes and introns are extracted from the source GFF files by iterating over a sorted list of all annotated coding sequences (CDS). Each CDS has a protein ID attribute, a start position, an end position, and a scaffold name. Gene boundaries are obtained from the start position of its first CDS and the end position of its last CDS. All gaps within the gene not covered by any of its CDS are considered and stored as intros. See Figure 7.2 illustrating extracted and non-extracted introns.

CDS makes a subset of exons but not vice versa, as exons close to both the 5' and 3' sites of a gene contain UTR and may contain only UTR [58]. The use of CDS for intron detection implies that only intra-CDS introns are extracted.

Only genes and introns on the positive strand were considered in the data extraction pipeline and data analysis. This simplifies the pipeline but reduces the amount of extracted data into half. The smaller dataset without reduction of variability in the data does not have any impact on the results due to the large size of the dataset. See Section 5.3 with dataset statistics.

Gene direction is close to random in fungal DNA [59], and it is further supposed that intron-specific differences between two DNA strands are weak or non-existent. This possible effect on the data is further weakened by the non-systematic selection of strands during DNA sequencing. However, some of these assumptions deserve a deeper analysis.

A CSV file with genes and introns is produced for each organism. Each gene and intron has an ID and a parent ID—this allow detailed analysis at later stages.

7.3 Candidate Splice Site Extraction

Positions of all candidate splice sites within genes ,as described by Section 7.2, are detected. Candidate splice sites are divided into donors and acceptors; they are further divided into positive examples—i.e. where true splice sites are located—and negative examples. Not all candidate splice sites are included, see Section 6.2. The intersecting areas of overlapping genes are processed only once to avoid the duplication of candidate splice site positions. Splice sites from all genes are considered within these areas.

Consensus dinucleotide, which does coincide with a respective splice site of an intron of any gene, is not included in the set of negative splice site samples even if it is included in another overlapping gene at a non-splice site position. This prevents inconsistent labeling of training data in cases of alternative splicing and overlapping genes.

Four Numpy NPZ files are generated for each organism:

- positive donor positions,
- positive acceptor positions,
- negative donor positions,
- negative acceptor positions.

Each of these NPZ files maps the scaffold name to a list of 0-indexed positions and feature IDs.

7.4 Training Samples Extraction

Candidate splice site positions, as described in Section 7.3, are used for the extraction of actual training, validation, and testing data. The generation of samples is the first step in the pre-processing pipeline which includes some kind of sub-sampling (i.e. no positions are skipped by the previous pipeline steps). The intermediate data is utilized to avoid the scanning of large amounts of source files with DNA sequences and annotations when regenerating training data on various filtering, sampling, and windowing criteria.

Sample window size, i.e. the number of nucleotides in it, splice site relative offset, the maximum number of examples, and other criteria are all configurable. Stratified sampling is done on groups defined by organism, splice site type, and true/false positivity to avoid any over-representation of organisms. Under this constraint, examples were selected stochastically.

The samples are stored in numbered sub-sub directories as NPZ files map the encoded input sequences to labels. This nested structure is used to obtain good performance on some Linux file systems as the performance on large directories might be very low [60]. A CSV index file, coupling the file paths with sample types and other properties, is created for quick access and further sub-sampling for experiments that do not require a large amount of data.

7.5 Training and Evaluation

There is a script for automatic neural network training, other script for generation and persistence of training-validation-test data split with various other dataset configurations.

A program which automatically evaluates a trained neural network on validation and test datasets was created to allow fast experimentation and comparability among experiments. A PDF with various network performance metrics on whole datasets as well as on individual organisms is produced by the program. This program is extended with more programs that do more detailed inspections, evaluations, and visualization—for example, the analysis of sensitivity of the networks to one nucleotide mutations.

Training, evaluation, and other related automation are implemented in Python for its ease of use, widespread usage among scientists, and abundance of relevant deep learning, statistical and data science libraries.

8 Evaluation

Contents

Detailed evaluation, inspection, and analysis of splice site classification networks, as well as the performance of the overall intron detection pipeline, is reported and discussed in this chapter.

Section 8.1 gives details of test sample selection and puts that into the context of other existing research, specific needs, and goals of this work. Section 8.2 reports several basic per-organism evaluation metrics, such as precision and recall, as well as prediction confidence distributions. Section 8.3 compares the performance of the neural networks with the SVM developed in [46]. Section 8.4 describes the CPU usage of the neural networks and SVM; it also evaluates the possible cost of using the pipeline in the Google Cloud Platform.

Phylum	**Species**	**Organism ID**
Ascomycota	Aspergillus wentii	Aspwe1
Basidiomycota	Mycena albidolilacea	Mycalb1
Blastocladiomycota	Allomyces macrogynus	Allma1
Chytridiomycota	Chytriomyces sp. MP71	Chytri1
Cryptomycota	Rozella allomycis	Rozal_SC1
Microsporidia	Encephalitozoon hellem	Enche1
Mucoromycota	Lichtheimia corymbifera	Liccor1
Zoopagomycota	Coemansia reversa	Coere1

Table 8.1: Organisms included in the test dataset

Section 8.5 analyzes the sensitivity of the neural networks to single nucleotide mutations/swaps and compares that to known biological phenomena. Section 8.6 reports the error rate of the networks on negative samples in proximity to a true splice site. Section 8.7 describes the error rate of the network on introns of various lengths. Section 8.8 evaluates the dependency of donor and acceptor splice site models. And finally, Section 8.9 discusses the overall gene prediction pipeline.

8.1 Datasets

Prior to all experiments, the data was split into training, validation, and test datasets. The performance of splice site classification networks was evaluated on organisms that were not included in the training and validation datasets. The test dataset consists of eight organisms, each belonging to a different phylum. Data from organisms included in training datasets was not used in any experiment or evaluation before the final evaluation reported in this chapter. The selected test organisms are the same as in the work [46] to allow for good comparison. See Table 8.1 which lists all organisms included in the test dataset.

The decision to split datasets on the organism lever rather than the sample level is motivated by the need to test the network ability in order to generalize previously unseen organisms. The network will be used in a pipeline executed on metagenomes containing large number of previously unseen organisms. This sharply contrasts

	NN 100			**NN 400**		
	Precision	**Recall**	**AUC**	**Precision**	**Recall**	**AUC**
Allma1	55.8%	74.0%	96.1%	62.8%	82.3%	97.2%
Aspwe1	53.3%	92.5%	98.3%	68.1%	94.0%	98.5%
Chytri1	49.4%	71.4%	95.5%	65.1%	85.0%	97.7%
Coere1	7.1%	61.8%	85.5%	9.0%	64.7%	87.8%
Enche1	0.8%	100.0%	99.4%	1.0%	95.0%	98.2%
Liccor1	56.2%	89.9%	98.0%	75.8%	94.8%	98.9%
Mycalb1	61.9%	78.1%	95.2%	75.3%	82.4%	96.6%
Rozal_SC1	25.6%	40.3%	86.6%	35.7%	45.3%	90.0%

Table 8.2: Performance of donor classification models. Model NN 100 has input window size of 100 nucleotides, starting at the splice site and going downstream. Model NN 400 has input window size of 400 nucleotides, spanning exactly 200 nucleotides to both sides from the splice site.

with some studies on automated intron detection because it usually reports the results measured on organisms included in the training dataset [37].

8.2 Basic Metrics

One final neural network architecture was selected for the classification of both candidate donor splice sites and candidate acceptor splice sites. See Section 6.4. Two versions of the model with different input window sizes were selected for the final evaluation and usage. Model NN 100 has an input window size of 100 nucleotides and model NN 400 has an input window size of 400 nucleotides. The larger model, unsurprisingly, has better performance but at the cost of roughly 2.9× higher CPU usage per classification. See Table 6.1 and Table 6.2 which compare the performance of the models with various input window sizes (evaluated on the validation dataset during the experimentation phase).

The performances of donor splice site classification models are reported in Table 8.2 and the performances of acceptor splice site models are reported in Table 8.3 [1]. The low precision on some organisms is largely due to the high ratio of the number of candidate splice sites to the number of true splice sites. See Section 5.3 on data statistics. Also see Section 6.2 that describes which splice sites were included in the datasets.

	NN 100			**NN 400**		
	Precision	**Recall**	**AUC**	**Precision**	**Recall**	**AUC**
Allma1	70.6%	80.2%	97.2%	66.3%	83.7%	97.4%
Aspwe1	54.6%	93.4%	98.5%	56.9%	93.9%	98.6%
Chytri1	59.1%	82.0%	97.3%	60.3%	84.9%	97.7%
Coere1	5.9%	46.6%	83.5%	7.1%	57.0%	85.3%
Enche1	0.3%	42.1%	91.9%	0.3%	42.1%	94.3%
Liccor1	68.5%	93.5%	98.8%	72.3%	93.8%	98.9%
Mycalb1	65.4%	80.8%	96.3%	63.9%	83.0%	96.7%
Rozal_SC1	12.0%	22.7%	79.6%	19.5%	30.1%	84.1%

Table 8.3: Performance of acceptor classification models. Model NN 100 has input window size of 100 nucleotides, starting 100 nucleotides upstream from the splice site and going downstream. Model NN 400 has input window size of 400 nucleotides, spanning exactly 200 nucleotides to both sides from the splice site.

The neural networks output values between 0 and 1. This value needs to be compared to a threshold to obtain a binary classification. During the evaluation of all splice site classification models, the threshold was set to 0.5. See Figure 8.1 and Figure 8.2 which depict the distribution of the prediction values on both positive and negative samples of the donor model. Figure 8.3 and Figure 8.4 visualize the same properties on the acceptor model.

The confidence intervals visualized in Figure 8.2 and Figure 8.4 are calculated with Formula 8.1 and Formula 8.2 derived in [61, p. 176].

$$\theta_L = 0.5\chi^2_{2k,\alpha/2} \tag{8.1}$$

$$\theta_U = 0.5\chi^2_{2k+2,1-\alpha/2} \tag{8.2}$$

Kolmogorov–Smirnov statistic computed on the cumulative distribution functions of the prediction values on positive and negative samples is 0.83 at 0.18 for the donor model and 0.79 at 0.18 for the acceptor model. The values imply the high ability of the model to discriminate positive and negative samples at the threshold point of

[1]Only 20 positive donor samples and 19 positive acceptor samples were used during the evaluation of organism Enche1.

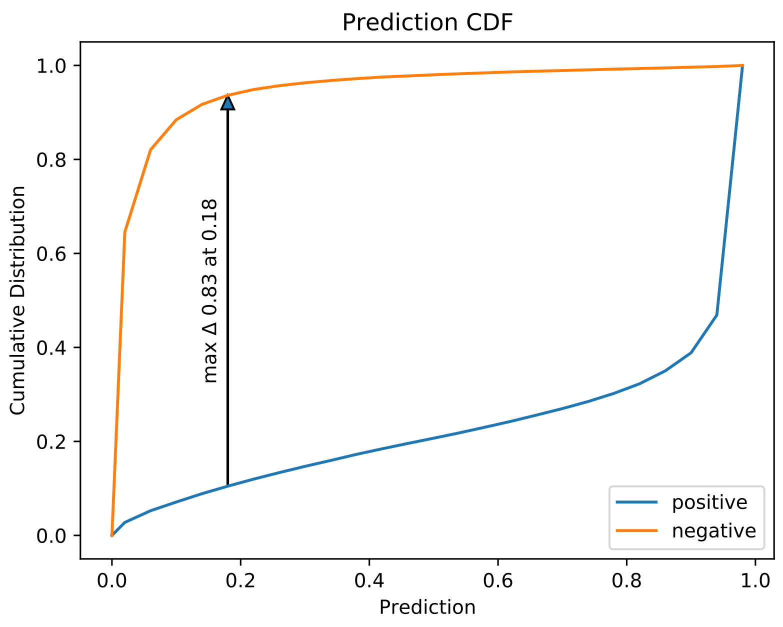

Figure 8.1: Cumulative distribution function with Kolmogorov–Smirnov statistic of prediction values from NN 400 model on positive and negative samples of donor splice sites.

0.18. During the evaluation and production use of the models, a higher threshold of 0.5 was used to compensate for the high positive to negative sample rate ratio.

The prediction distributions were measured over samples from all organisms included in the test dataset. In total, 10 000 positive samples and 10 000 negative samples were included from each organism, except those which did not have enough annotated features. The only organism which was largely underrepresented was Enche1. Prediction distribution on different organisms, especially organisms from different phyla, differ. Likewise, the used sample set does not represent true organism distribution in nature or in extracted metagenomes. This implies the need for a careful interpretation of the plots.

8.3 Comparison with SVM

Table 8.4 compares the true positive rate and true negative rate of two neural networks with different input window sizes and SVM, as initially developed in [46]. The smaller neural network (NN 100) had an input window of 100 nucleotides going

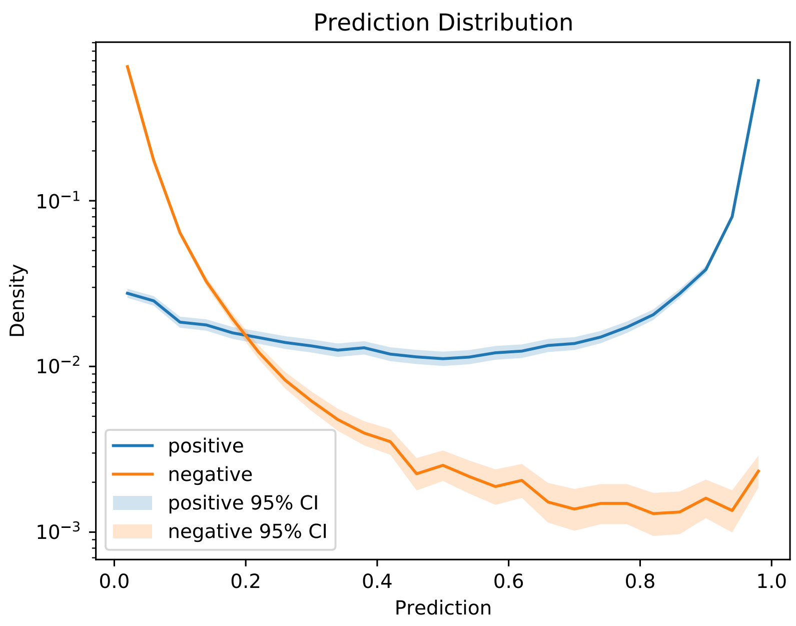

Figure 8.2: Density of prediction values from NN 400 model on positive and negative samples of donor splice sites.

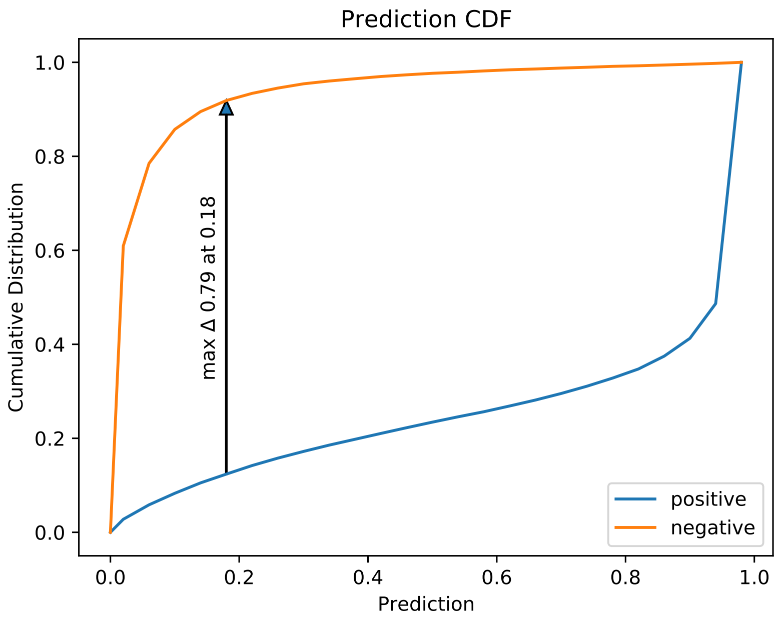

Figure 8.3: Cumulative distribution function with Kolmogorov–Smirnov statistic of prediction values from NN 400 model on positive and negative samples of acceptor splice sites.

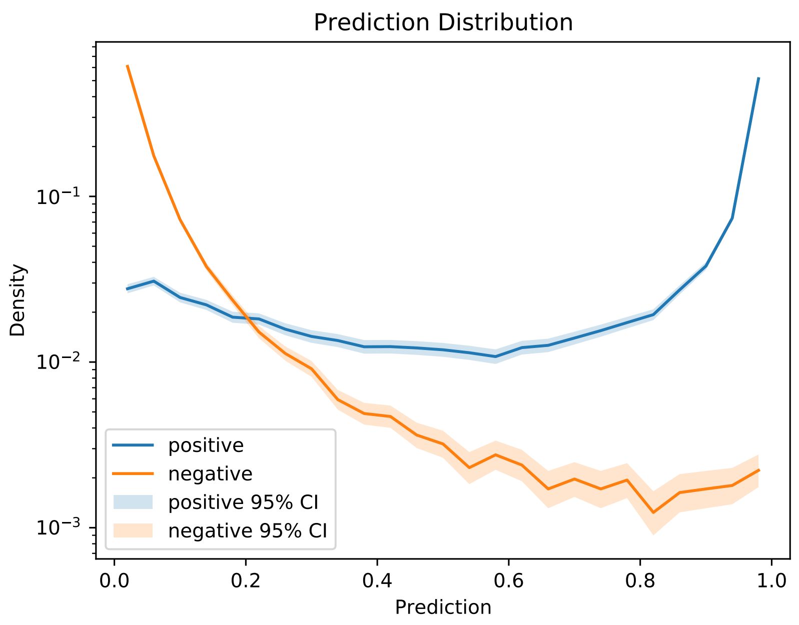

Figure 8.4: Density of prediction values from NN 400 model on positive and negative samples of acceptor splice sites.

from the splice site in the intron direction (downstream in the case of a donor and upstream in the case of an acceptor). The larger neural network (NN 400) had a window size of 400 nucleotides and spanned 200 nucleotides to both directions from the splice site. The SVM used for the evaluation was trained solely on data from the Basidiomycota phylum.

Both neural networks produce much less false positive predictions compared to the SVM. Furthermore, the NN 400 model outperforms the SVM in every measurement, except for *TPR* on donors in the Basidiomycota phylum.

The improvement in *TNR* achieved with the neural networks is very important due to the large ratio of the false candidate splice sites to the true candidate splice sites.

All classification methods in Table 8.4 were evaluated on the same dataset. Negative samples were taken from genetic regions defined as the area between the beginning of the first exon of the gene and the end of the last exon of the gene. These regions are wider than the CDS and intra-CDS regions used in other parts of this work, see Section 7.1.

		Ascomycota			**Basidiomycota**		
		SVM	**NN 100**	**NN 400**	**SVM**	**NN 100**	**NN 400**
Donor	*TPR*	86.6%	84.9%	87.2%	91.6%	86.3%	89.9%
	TNR	94.9%	97.2%	97.7%	95.9%	97.6%	98.1%
Acceptor	*TPR*	83.4%	85.3%	86.4%	88.1%	87.7%	88.8%
	TNR	93.8%	97.7%	97.9%	93.8%	97.8%	97.7%

Table 8.4: Comparison of true positive rate (*TPR*) and true negative rate (*TNR*) between SVM, neural network with window size 100 (NN 100) and neural network with window size 400 (NN 400).

8.4 Computational Intensity

The NN 400 model uses 2.88× more CPU time per inference than the NN 100 model. The SVM uses 46.4× more CPU time per inference than the NN 100 model. This makes the use of the neural networks more practical on large metagenomes.

The NN 100 model uses 1.37 CPU seconds per 1000 predictions, while the NN 400 model uses 3.94 CPU seconds per 1000 predictions. The measurement was performed on the laptop ThinkPad T490 with Intel® Core™ i5-8265U CPU @ 1.60GHz. The measurement was performed with all samples loaded in memory. TensorFlow version 2.3.1 compiled with instructions `AVX2` and `FMA` enabled was used. The measurement included only the CPU time usage of Python call `model.predict(inputs)`.

There is around 10^8 donor and acceptor candidate splice sites in a metagenome of size of 10^9 nucleotides. Using the NN 100 model, all candidate splice sites in this hypothetical dataset could be classified in 137 000 CPU seconds or 9.5 hours on four CPU cores. Both CPU and real time would be much smaller on a server CPU, GPU or TPU.

One hour of a single CPU on an E2 machine type in Google Cloud Platform costs 0.021811 USD [62]. Using this price and ignoring the differences between CPUs, any inefficiencies, or overhead, all introns in the aforementioned hypothetical metagenome could be classified for 0.83 USD.

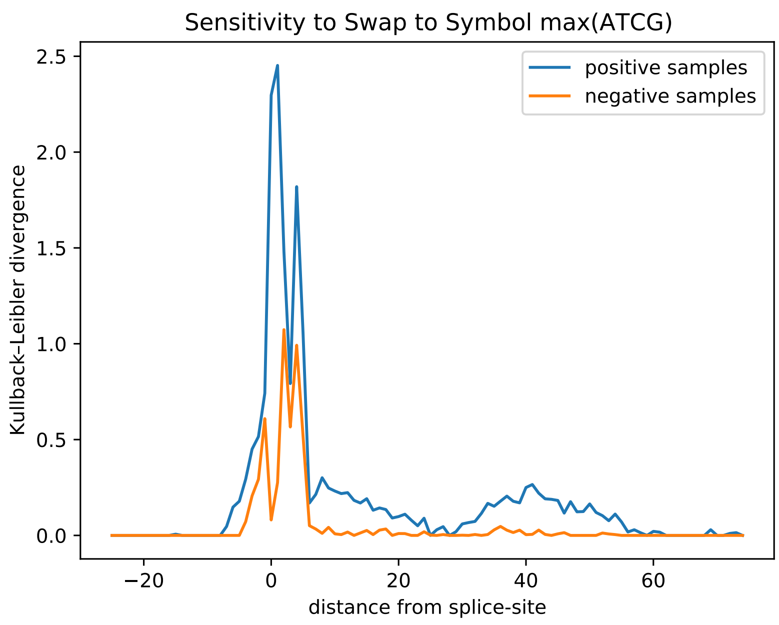

Figure 8.5: Kullback–Leibler divergences between NN 400 donor model inferences on unmodified input sequences and inferences on sequences with single nucleotide modifications at various positions. Position-wise maximum over swaps to adenine, thymine, cytosine, adenine are shown.

8.5 Positional Sensitivity

Figure 8.5 visualizes an estimation of the Kullback–Leibler divergence between the distributions of donor model inferences made on unmodified sequences and sequences with single nucleotide symbol swaps.

The figure illustrates that the network is highly sensitive in the close vicinity of the splice site with decreasing sensitivity in the downstream (intron) direction. Almost no sensitivity could be found to one nucleotide swaps 9 nucleotides and further upstream from the splice site. Part of this sensitivity spike around the splice site is consistent with the literature which reports that the 5'-terminus of the U1 snRNA component of spliceosome binds to the nearly perfect Watson-Crick complement sequence `CAGGURAGU` that spans -3 to +6 around 5'-end of an intron [63].

The second peak of sensitivity is found around 40 nucleotides downstream from the splice site. There is a high frequency of introns of a length between 40 and 75 nucleotides, see Figure 5.2. The branch point is located 18 to 40 nucleotides

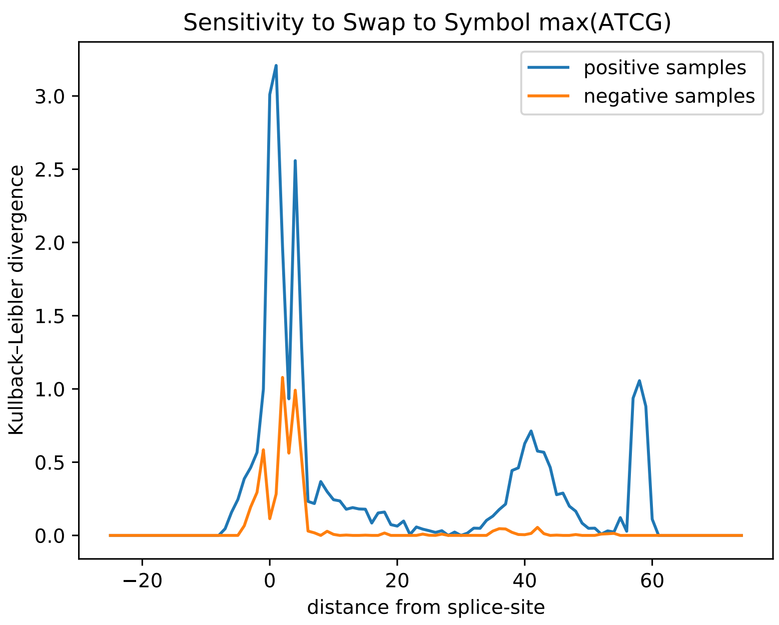

Figure 8.6: Kullback–Leibler divergences between NN 400 donor model inferences on unmodified input sequences and inferences on sequences with single nucleotide modifications at various positions. Position-wise maximum over swaps to adenine, thymine, cytosine, adenine are shown. Positive samples are limited to the splice site of introns of a length 60 nucleotides.

upstream from the acceptor splice site [18]. The relative location of the second peak likely implies that the network is learned to recognize area around branch point. This is further supported by Figure 8.6, where positive samples were limited to the splice sites of introns of a length of 60 nucleotides. Figure 8.6 also displays a sensitivity peak at the location of the acceptor splice site.

Figure 8.7 visualizes an estimation of the Kullback–Leibler divergence between the distributions of acceptor model inferences made on unmodified sequences and sequences with single nucleotide symbol swaps. Figure 8.8 visualizes the divergence on introns of a length of 60 nucleotides. The acceptor sensitivity plot largely overlaps with the donor sensitivity plot, with a notable difference of the sensitivity decreasing more steeply near the opposite splice site and vice versa. Compared to the donor model, the acceptor model is more sensitive to one nucleotide swap in the negative samples.

A similar analysis based on different techniques was done in [37] which used a

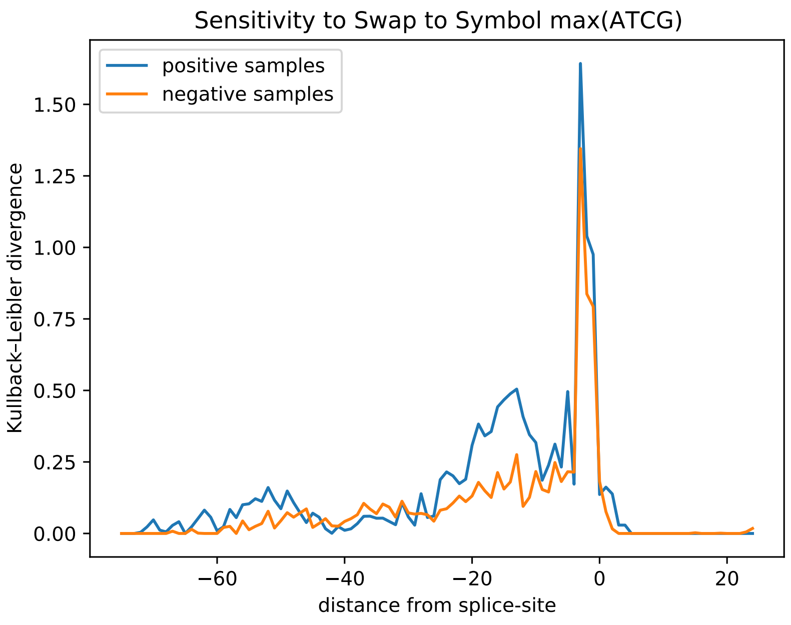

Figure 8.7: Kullback–Leibler divergences between NN 400 acceptor model inferences on unmodified input sequences and inferences on sequences with single nucleotide modifications at various positions. Position-wise maximum over swaps to adenine, thymine, cytosine, adenine are shown.

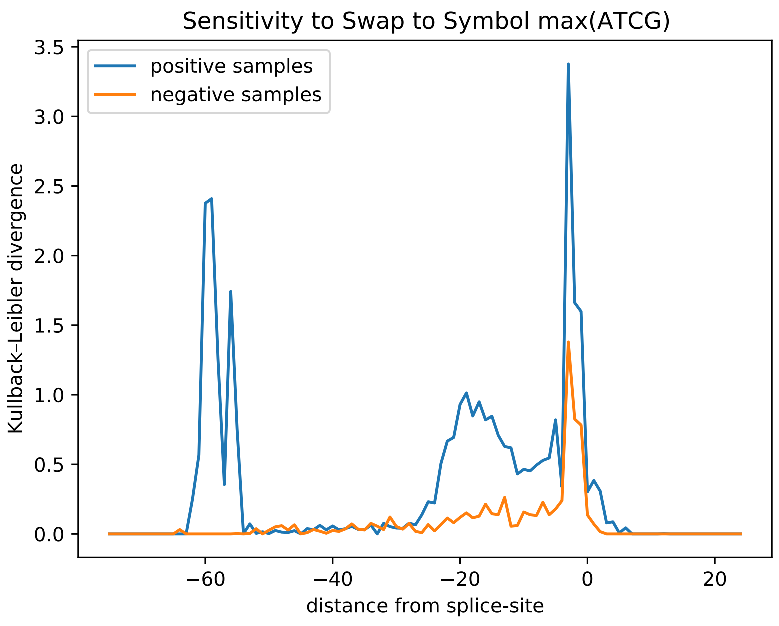

Figure 8.8: Kullback–Leibler divergences between NN 400 acceptor model inferences on unmodified input sequences and inferences on sequences with single nucleotide modifications at various positions. Position-wise maximum over swaps to adenine, thymine, cytosine, adenine are shown. Positive samples are limited to the splice site of introns of a length of 60 nucleotides.

CNN trained and evaluated on Arabidopsis and human. Visualizations reported in that paper, however, differ with the results reported in this section, especially in the areas more than 10 nucleotides distant from the splice sites.

Kullback–Leibler divergence estimation is calculated with Formula 8.3 on two n-tuples of samples i.i.d. drawn from distributions p and q respectively. $\nu_k(i)$ is distance of the i-th sample from the first n-tuple to the k-th nearest neighbor from the second n-tuple; $\rho_k(i)$ is distance of the i-th sample from the first n-tuple to the $k+1$ nearest neighbor from the same n-tuple. This equation was derived from [64], $n = 1000$ and $k = 10$ were used.

$$D_n(p \parallel q) = \frac{1}{n}\sum_{i=1}^{n} log\frac{\nu_k(i)}{\rho_k(i)} + \log\frac{n}{n-1} \tag{8.3}$$

8.6 False Positives in the Proximity to a Splice Site

Figure 8.9 shows the dependence of the donor model performance on negative splice site examples to the distance to the closest real donor splice site. Figure 8.10 displays this dependence only in a narrow neighborhood of true splice sites.

The plots display the mean predicted value and the false positive rate for each distance bin. The mean predicted value is equal to the mean prediction error because only negative samples are used. The measurements are done on bins of size 5 in Figure 8.9 and on bins of size 3 in Figure 8.10. The figures differ due to the overall captured distance range and different bin sizes.

This data was generated on the validation dataset because there were not enough samples in the test dataset to calculate the unbiased report without too much noise. Only splice sites with consensus dinucleotides were included among the negative samples. However, the closest true splice site was selected from the set of all splice sites.

Figure 8.11 and Figure 8.12 visualize the same dependency for the acceptor splice site model. An interesting difference from the donor model is that the error spike around the true splice site is higher and wider.

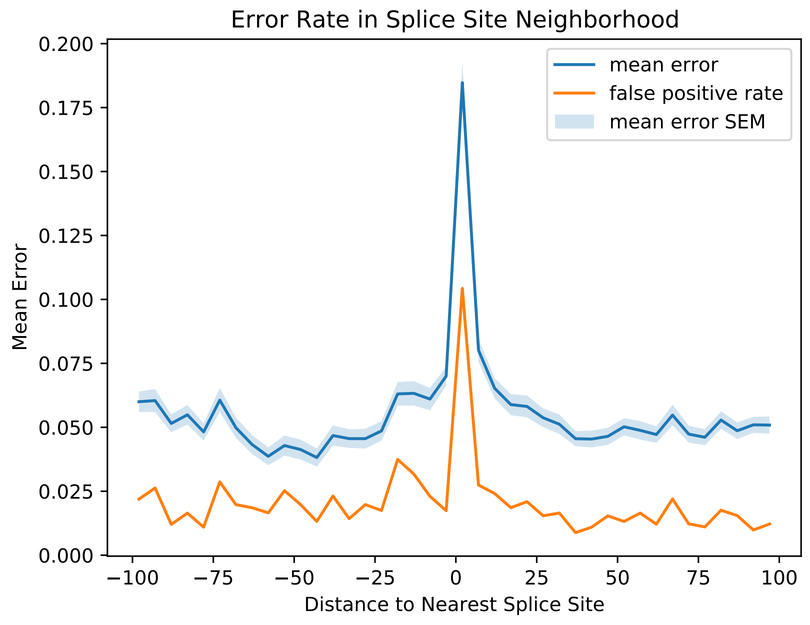

Figure 8.9: NN 400 donor model error rate dependence on the relative position of the nearest true donor splice site.

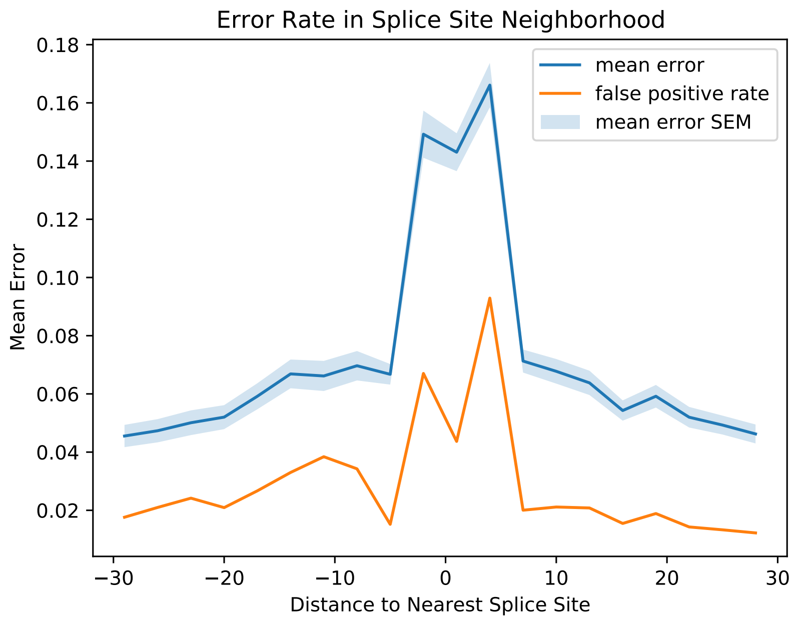

Figure 8.10: NN 400 donor model error rate dependence on relative position of the nearest true donor splice site.

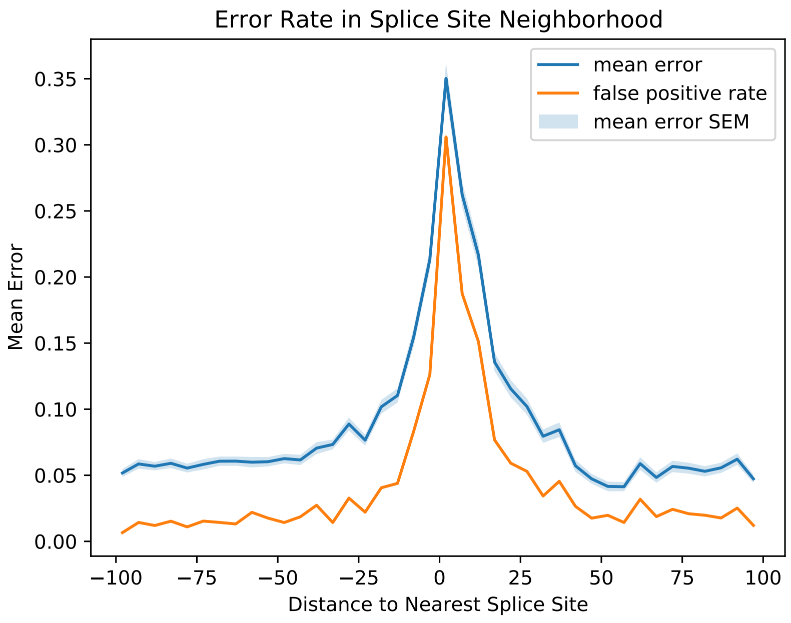

Figure 8.11: NN 400 acceptor model error rate dependence on the relative position of the nearest true donor splice site.

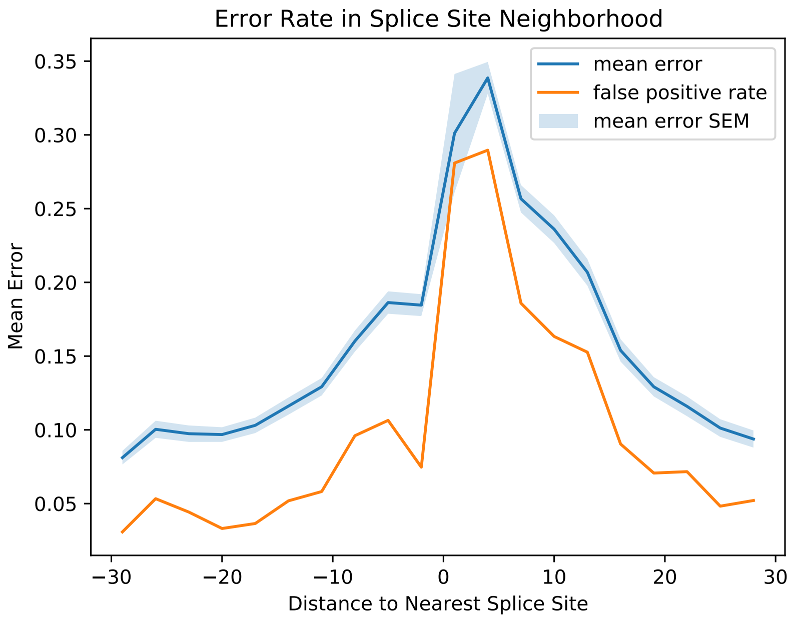

Figure 8.12: NN 400 acceptor model error rate dependence on the relative position of the nearest true donor splice site.

The error spike around true splice sites leads to the higher than uniform presence of false intron detections with an almost perfect overlap with the true introns compared to introns with a low overlap with the true introns.

After the detected splice sites are combined into whole introns, among the overlapping intron detections only those detected with the largest splice site prediction values are kept. It is expected that this would lead to the selection of the true introns in the majority of the cases.

Likewise, a small negative effect on gene prediction from incorrect intron detections which large relative overlap with the true introns is expected.

8.7 Introns of Various Lengths

Figure 8.13 and Figure 8.14 show the dependency of the error rates of donor and acceptor models, respectively, on the splice sites associated with introns of various lengths. The figures demonstrate that the models are systematically not recognizing the splice sites of introns shorter than 40 nucleotides. The error rate is the smallest in the area around 55 nucleotides and goes up for splice sites of longer introns. Low performance on short introns is more pronounced in the acceptor model.

The effect could be, in part, explained by the distribution of intron lengths in the data, see Figure 5.2. Training samples of the models were randomly drawn from the dataset of training organisms. Therefore, the lengths of most of the introns associated with the positive splice sites the model "saw" during training were concentrated around 55 nucleotides.

Introns having lengths shorter than 30 nucleotides are extremely rare in the gene databases across all eukaryotic organisms [65]. Their existence in the data might be a result of incorrect annotation, and it is hypothesized that they do not exist in nature due to the minimum sequence elements needed for their splicing [65]. Introns shorter than 30 nucleotides are rare, but they are present in the data used in this work. In light of this, the high error rate on the splice sites of very short introns could be caused by data quality.

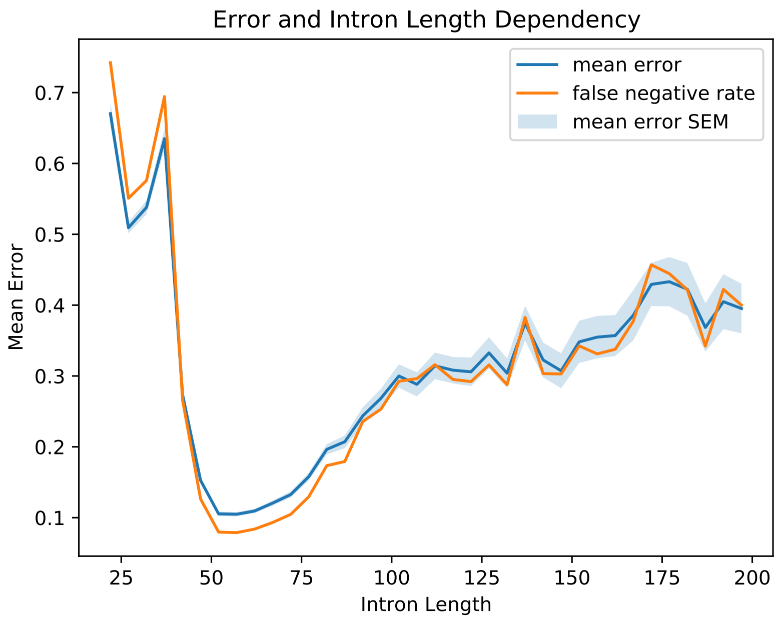

Figure 8.13: Dependence of the NN 400 donor model error rate on positive splice site samples and the lengths of associated introns.

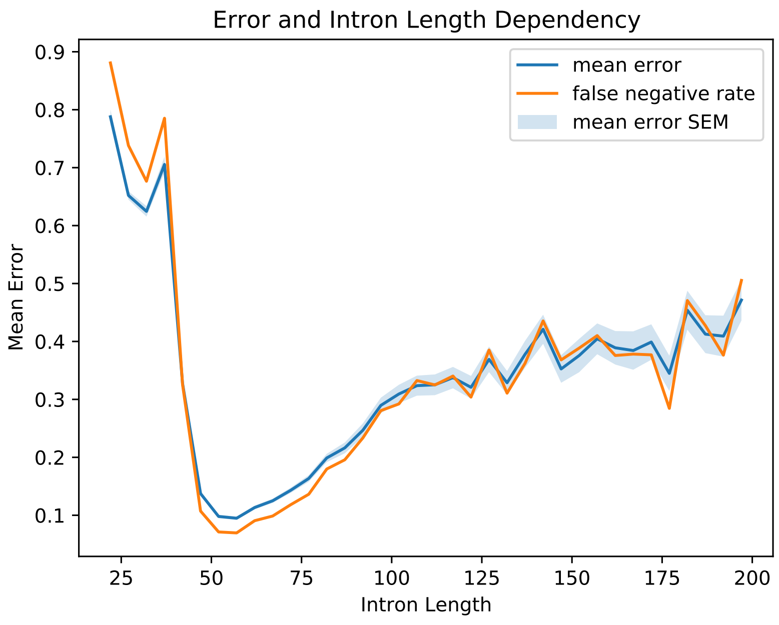

Figure 8.14: Dependence of NN 400 acceptor model error rate on positive splice site samples and lengths of associated introns.

Organism ID	**Correlation**
Aspwe1	0.69
Mycalb1	0.63
Allma1	0.73
Chytri1	0.65
Rozal_SC1	−0.05
Enche1	−0.13
Liccor1	0.64
Coere1	0.66

Table 8.5: The Pearson correlation coefficient of NN 400 donor and NN 400 acceptor model outputs on the splice sites of the same intron.

8.8 Donor and Acceptor Model Correlation

Figure 8.15 depicts the output dependency of the donor splice site model and the acceptor splice site model when predictions are performed on the (opposite) splice sites of the same intron. Per organisms correlations of the outputs are given in Table 8.5.

The correlation is negative only for Rozal_SC1 and Enche1; it is larger than 0.6 for all other test organisms. This might be related to the fact that the performance on the organisms Rozal_SC1 and Enche1 is lower by a big margin compared to all the other test organisms, see Table 8.2 and Table 8.3.

The data illustrates that an intron with a “hard-to-recognize” splice site is of diminished “visibility” even for the model of the opposite splice site. In Section 8.5, it is demonstrated that both the donor splice site model and the acceptor splice site model are sensitive in the region of the opposite splice site—this likely explains the correlations.

If donor and acceptor models were independent, the probability of the detection of an intron would be a multiplication of the probabilities of its splice sites being detected (independently). With strong correlation of donor and acceptor model outputs, the overall likelihood of the detection of a true intron is much higher. This is shown by Table 8.6 that gives a percentage of introns with both splice sites detected by the real models and by hypothetical statistically independent models.

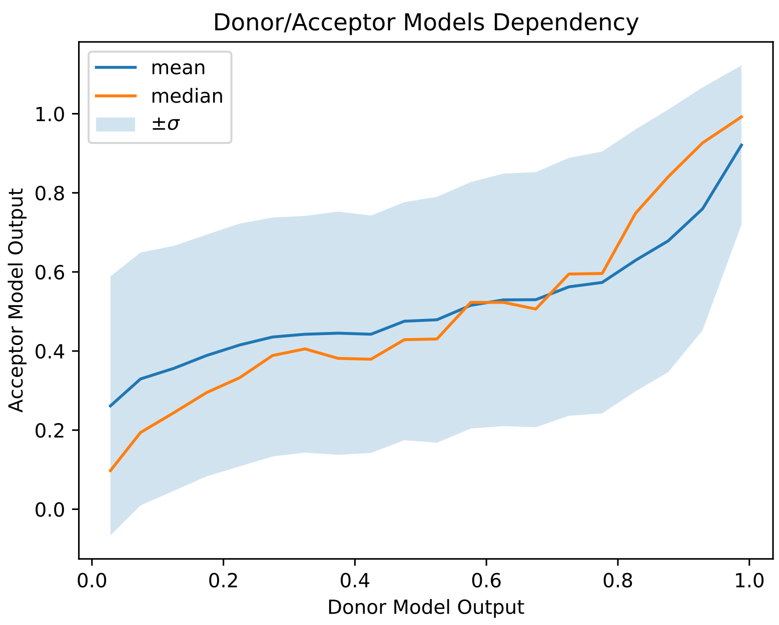

Figure 8.15: Dependency of NN 400 donor and NN 400 acceptor model outputs on the splice sites of the same intron.

Organism ID	Detected Introns	Multiplication
Aspwe1	91.6%	88.0%
Mycalb1	72.8%	65.2%
Allma1	78.0%	69.8%
Chytri1	79.2%	72.7%
Rozal_SC1	12.5%	13.6%
Enche1	42.1%	42.1%
Liccor1	92.0%	89.2%
Coere1	51.1%	37.2%

Table 8.6: The percentage of introns whose splice sites were recognized by both the NN 400 donor model and the NN 400 acceptor model and the percentage of detected introns, if the models were statistically independent but with the same true positive rate.

8.9 Whole Gene Prediction Pipeline

As discussed in Chapter 1 and Chapter 3, the primary purpose of splice site detection models and their combination to the full intron detection pipeline is to improve the performance of a larger gene prediction pipeline, which is utilized to detect gene homologies in novel and known fungal DNA sequences in metagenomes.

Utilization of the intron detection pipeline based on SVM, as developed in [46], has significantly improved the gene prediction by increasing the number of discovered gene homologies when the gene prediction pipeline was executed separately with and without the intron detection.

Based on the results with SVM models and better performance of the neural networks compared to SVM, it is expected that the gene prediction will further benefit with the introduction of the neural network-based intron detection pipeline.

9 Conclusion

Contents

The goal of this work was to develop an algorithm for automated intron detection in fungal metagenomes with the use of neural networks. Emphasis was put on computational requirements and comparison with an existing intron detection pipeline based on support vector machines (SVM).

The resulting pipeline contains two splice site classification models based on deep recurrent convolutional neural networks. As opposed to the pipeline based on SVM, no third intron model is used. The solution outperforms the approach based on SVM and requires 46 times less computational resources for classification. The neural networks also generalize in a better way than the SVM models, and therefore, only one model is used for all phyla.

Up to 91.6% introns on the Ascomycota phylum and 72.8% introns on the Basidiomycota phylum are detected with the neural network-based pipeline.

9.1 Future Work

This work opens a possibility to implement an online, web-based service for intron detection and/or removal. This could be done as a simple webpage where the user uploads a FASTA file and gets a GFF file with automatically annotated introns within minutes. All computations could easily be distributed thanks to the design of the pipeline and characteristics of the data. Low CPU usage of the developed classification models makes this possibility economically feasible, see Chapter 8.

A direction for further investigation is the generalization of the developed models. Interesting results could be obtained by evaluating the model performance on different kingdoms and analyzing the errors it makes. This raises two questions: how well and consistently the model performs when trained and evaluated on distant organisms and whether the same model architecture and approach works well on different kingdoms.

Using the pre-trained models and utilizing transfer learning on organisms with a low amount of annotated data is also a possibility for future investigation.

Using the models in quality assurance procedures of non-automated genome annotation is also an interesting topic worth further exploration. Paying more attention to the annotations that are in disagreement with the model predictions might improve the overall data quality with fixed effort spent.

Performing in vitro or in vivo experiments might be laborious, time-consuming, and expensive. Using the developed in silico methods before the "wet" research is started might save resources, time, and help future research to focus on promising areas. This is yet another area of potential future research.

Biodiversity is our most valuable but least appreciated resource.

— Edward O. Wilson

References

[1] David L Hawksworth and Robert Lücking. “Fungal diversity revisited: 2.2 to 3.8 million species”. In: *The fungal kingdom* (2017), pp. 79–95.

[2] DL Hawksworth. “Global species numbers of fungi: are tropical studies and molecular approaches contributing to a more robust estimate?” In: *Biodiversity and Conservation* 21.9 (2012), pp. 2425–2433.

[3] Nature Microbiology. “Stop neglecting fungi”. In: *Nat Microbiol* 2 (2017), p. 17120.

[4] Bruce Alberts et al. “Molecular biology of the cell”. In: (2018).

[5] Elizabeth Pennisi. “DNA study forces rethink of what it means to be a gene”. In: *Science* 316.5831 (2007), pp. 1556–1557.

[6] Predrag Slijepcevic. “Genome dynamics over evolutionary time:“C-value enigma” in light of chromosome structure”. In: *Mutation Research/Genetic Toxicology and Environmental Mutagenesis* 836 (2018), pp. 22–27.

[7] George Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[8] David Silver et al. “Mastering the game of Go with deep neural networks and tree search”. In: *nature* 529.7587 (2016), p. 484.

[9] Yi Sun et al. “Deepid3: Face recognition with very deep neural networks”. In: *arXiv preprint arXiv:1502.00873* (2015).

[10] Wayne Xiong et al. “Achieving human parity in conversational speech recognition”. In: *arXiv preprint arXiv:1610.05256* (2016).

[11] Dmitrii Bychkov et al. “Deep learning based tissue analysis predicts outcome in colorectal cancer”. In: *Scientific reports* 8.1 (2018), p. 3395.

[12] Robert C King, William D Stansfield, Pamela Khipple Mulligan, et al. *A dictionary of genetics.* Oxford University Press, USA, 2006.

[13] Thomas D Pollard et al. *Cell Biology.* Third Edition. Elsevier Health Sciences, 2016.

[14] Janusz AZ Jankowski and Julia M Polak. *Clinical gene analysis and manipulation: Tools, techniques and troubleshooting.* Cambridge University Press, 1996.

[15] NEUROtiker. *Generic structure of nucleotides.* 2008. URL: `https://commons.wikimedia.org/wiki/File:Nukleotid_num.svg` (visited on 10/18/2020).

[16] GM Cooper and RE Hausman. “The cell: a molecular approach, 2nd edn Sunderland”. In: *MA: Sinauer Associates.[Google Scholar]* (2000).

[17] Thomas Shafee and Rohan Lowe. “Eukaryotic and prokaryotic gene structure”. In: *WikiJournal of Medicine* 4.1 (2017), p. 2.

[18] Suzanne Clancy. “RNA splicing: introns, exons and spliceosome”. In: *Nature Education* 1.1 (2008), p. 31.

[19] Saul B Needleman and Christian D Wunsch. “A general method applicable to the search for similarities in the amino acid sequence of two proteins”. In: *Journal of molecular biology* 48.3 (1970), pp. 443–453.

[20] Temple F Smith, Michael S Waterman, et al. “Identification of common molecular subsequences”. In: *Journal of molecular biology* 147.1 (1981), pp. 195–197.

[21] Wing-Kin Sung. *Algorithms in bioinformatics: A practical introduction.* CRC Press, 2009.

[22] RM Casey. “BLAST sequences aid in genomics and proteomics”. In: *Business Intelligence Network* (2005).

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning.* MIT press, 2016.

[24] Yoshua Bengio. *Learning deep architectures for AI.* Now Publishers Inc, 2009.

[25] Glosser.ca. *Artificial neural network with layer coloring.* 2013. URL: `https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg` (visited on 10/18/2020).

[26] Vincent Dumoulin and Francesco Visin. “A guide to convolution arithmetic for deep learning”. In: *arXiv preprint arXiv:1603.07285* (2016).

[27] M.S. Kris A. Wetterstrand. *DNA Sequencing Costs: Data.* 2019. URL: `https://www.genome.gov/about-genomics/fact-sheets/DNA-Sequencing-Costs-Data` (visited on 10/30/2019).

[28] Dennis A Benson et al. “GenBank”. In: *Nucleic acids research* 41.D1 (2012), pp. D36–D42.

[29] *Release Notes For GenBank Release 233.* GenBank. 2019. URL: `https://www.ncbi.nlm.nih.gov/genbank/release/233/` (visited on 11/09/2019).

[30] M Christensen et al. “Wood-inhabiting fungi as indicators of nature value in European beech forests”. In: *Monitoring and Indicators of Forest Biodiversity in Europe–From Ideas to Operationality* 229 (2005).

[31] Alessia Bani et al. “The role of microbial community in the decomposition of leaf litter and deadwood”. In: *Applied Soil Ecology* 126 (2018), pp. 75–84.

[32] Burkhard Morgenstern et al. “Exon discovery by genomic sequence alignment”. In: *Bioinformatics* 18.6 (2002), pp. 777–787.

[33] Marvin B Shapiro and Periannan Senapathy. “RNA splice junctions of different classes of eukaryotes: sequence statistics and functional implications in gene expression”. In: *Nucleic acids research* 15.17 (1987), pp. 7155–7174.

[34] Elham Pashaei et al. “A novel method for splice sites prediction using sequence component and hidden markov model”. In: *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC).* IEEE. 2016, pp. 3076–3079.

[35] Neelam Goel, Shailendra Singh, and Trilok Chand Aseri. “An improved method for splice site prediction in DNA sequences using support vector machines”. In: *Procedia Computer Science* 57 (2015), pp. 358–367.

[36] Rahul Sarkar et al. “Splice Junction Prediction in DNA Sequence Using Multilayered RNN Model”. In: *International Conference on E-Business and Telecommunications*. Springer. 2019, pp. 39–47.

[37] Jasper Zuallaert et al. “SpliceRover: interpretable convolutional neural networks for improved splice site prediction”. In: *Bioinformatics* 34.24 (2018), pp. 4180–4188.

[38] Hamid Reza Hassanzadeh and May D Wang. “DeeperBind: Enhancing prediction of sequence specificities of DNA binding proteins”. In: *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE. 2016, pp. 178–183.

[39] Byunghan Lee et al. “DNA-level splice junction prediction using deep recurrent neural networks”. In: *arXiv preprint arXiv:1512.05135* (2015).

[40] Tatsuhiko Naito. “Human splice-site prediction with deep neural networks”. In: *Journal of Computational Biology* 25.8 (2018), pp. 954–961.

[41] Jose Mario Bello Pineda and Robert K Bradley. “Most human introns are recognized via multiple and tissue-specific branchpoints”. In: *Genes & development* 32.7-8 (2018), pp. 577–591.

[42] Andrew W Senior et al. “Improved protein structure prediction using potentials from deep learning”. In: *Nature* 577.7792 (2020), pp. 706–710.

[43] Igor V Grigoriev et al. “MycoCosm portal: gearing up for 1000 fungal genomes”. In: *Nucleic acids research* 42.D1 (2014), pp. D699–D704.

[44] Jon Ison et al. “EDAM: an ontology of bioinformatics operations, types of data and identifiers, topics and formats”. In: *Bioinformatics* 29.10 (2013), pp. 1325–1332.

[45] *GFF3*. Generic Model Organism Database. 2016. URL: `http://gmod.org/wiki/GFF3` (visited on 12/03/2020).

[46] Denis Baručić. “Automatic intron detection in fungal genomes using machine learning”. Czech Techincal University in Prague, 2019.

[47] Prajit Ramachandran, Barret Zoph, and Quoc V Le. “Searching for activation functions”. In: *arXiv preprint arXiv:1710.05941* (2017).

[48] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. “Rectifier nonlinearities improve neural network acoustic models”. In: *Proc. icml*. Vol. 30. 1. 2013, p. 3.

[49] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[50] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[51] Ashia C Wilson et al. “The marginal value of adaptive gradient methods in machine learning”. In: *Advances in neural information processing systems*. 2017, pp. 4148–4158.

[52] Dominic Masters and Carlo Luschi. “Revisiting small batch training for deep neural networks”. In: *arXiv preprint arXiv:1804.07612* (2018).

[53] François Chollet et al. *Keras*. `https://keras.io`. 2015.

[54] Martin Abadi et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". In: *arXiv preprint arXiv:1603.04467* (2016).

[55] Wassily Hoeffding. "Probability inequalities for sums of bounded random variables". In: *The Collected Works of Wassily Hoeffding*. Springer, 1994, pp. 409–426.

[56] Nicholas D Matsakis and Felix S Klock. "The rust language". In: *ACM SIGAda Ada Letters* 34.3 (2014), pp. 103–104.

[57] *Rust Programming Language*. 2020. URL: `https://www.rust-lang.org/` (visited on 09/14/2020).

[58] Alicia A Bicknell et al. "Introns in UTRs: why we should stop ignoring them". In: *Bioessays* 34.12 (2012), pp. 1025–1034.

[59] Xiu-Qing Li and Donglei Du. "Gene direction in living organisms". In: *Scientific Reports* 2.1 (2012), pp. 1–4.

[60] Borislav Djordjevic and Valentina Timcenko. "Ext4 file system in linux environment: Features and performance analysis". In: *International Journal of Computers* 1.6 (2012), p. 2012.

[61] Norman L Johnson, Adrienne W Kemp, and Samuel Kotz. *Univariate discrete distributions*. Vol. 444. John Wiley & Sons, 2005.

[62] *VM instances pricing*. 2020. URL: `https://cloud.google.com/compute/vm-instance-pricing` (visited on 11/24/2020).

[63] Laura De Conti, Marco Baralle, and Emanuele Buratti. "Exon and intron definition in pre-mRNA splicing". In: *Wiley Interdisciplinary Reviews: RNA* 4.1 (2013), pp. 49–60.

[64] Qing Wang, Sanjeev R Kulkarni, and Sergio Verdú. "A nearest-neighbor approach to estimating divergence between continuous random vectors". In: *2006 IEEE International Symposium on Information Theory*. IEEE. 2006, pp. 242–246.

[65] Allison Piovesan et al. "Identification of minimal eukaryotic introns through GeneBase, a user-friendly tool for parsing the NCBI Gene databank". In: *DNA Research* 22.6 (2015), pp. 495–503.