UNIVERSIDADE ESTADUAL DE MARINGÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE INFORMÁTICA

GUSTAVO HENRIQUE FERREIRA CRUZ

Orientadora: Profa. Dra. Josiane Melchiori Pinheiro

IDENTIFICAÇÃO DE REGIÕES CODANTES EM SEQUÊNCIAS DE DNA USANDO REDES NEURAIS CONVOLUCIONAIS

Maringá, PR

2022

UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

GUSTAVO HENRIQUE FERREIRA CRUZ

IDENTIFICAÇÃO DE REGIÕES CODANTES EM SEQUÊNCIAS DE DNA USANDO REDES NEURAIS CONVOLUCIONAIS

Monografia apresentada ao Curso de Ciência da Computação da Universidade Estadual de Maringá como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Profa. Dra. Josiane Melchiori Pinheiro

Maringá, PR
2022

Gustavo Henrique Ferreira Cruz

IDENTIFICAÇÃO DE REGIÕES CODANTES EM SEQUÊNCIAS DE DNA USANDO REDES NEURAIS CONVOLUCIONAIS

Monografia apresentada ao Curso de Ciência da Computação da Universidade Estadual de Maringá como parte dos requisitos necessários para a obtenção do grau em Bacharel em Ciência da Computação.

Aprovada em Maringá, __ de Maio de 2022.

Profa. Dra. Josiane Melchiori Pinheiro
Universidade Estadual de Maringá
Orientadora

Profa. Dra. Valéria Delisandra Feltrim
Universidade Estadual de Maringá - UEM
Examinadora

Prof. Dr. Yandre Maldonado e Gomes da Costa
Universidade Estadual de Maringá - UEM
Examinador

Agradecimentos

Agradeço primeiramente a Deus por me conceder forças de chegar até onde cheguei. Agradeço também minha família por ter me apoiado, incentivado e estar comigo desde o começo, me apoiando em todas as decisões da minha vida.

Agradeço a Universidade Estadual de Maringá pela oportunidade de contribuir com a ciência e pelo desenvolvimento pessoal que me forneceu em todo esses anos de curso.

Agradeço aos professores do DIN, que contribuíram grandemente para a minha formação acadêmica e me forneceram novos horizontes para se explorar, pesquisar e desbravar. Em especial a professora doutora Josiane Melchiori Pinheiro, que desde o primeiro ano me ajudou, conduziu e ensinou o que é a tríade do mundo universitário e ao professor Daniel Kikuti que, junto ao Programa de Educação Tutorial (PET), me introduziu e guiou pelo mundo da extensão e pesquisa.

Agradeço, ainda, aos meus amigos que sempre estiveram comigo ao longo da graduação, bem como meus colegas de turma.

Resumo

Este trabalho buscou implementar uma rede neural convolucional para identificar regiões codantes e não codantes em sequências de DNA. Os dados foram obtidos do GenBank, tendo sido coletados 7003 sequências do fungo *Colletotrichum* na proteína Actina e 3878 sequências do fungo *Diaporthe* na proteína Beta Tubulina. A construção do código se deu pela criação de diversos módulos. No primeiro módulo os dados recuperados do GenBank são filtrados para que apenas as informações relevantes para o contexto do trabalho permaneçam. No segundo módulo são geradas as possibilidades de íntrons, éxons e *neithers* (uma mistura de íntrons e/ou éxons). No terceiro módulo as sequências são transformadas para imagens em escala de cinza e normalizadas, redimensionando todas para um tamanho médio calculado, e, com isso, é feita a divisão dos dados para treino e teste, que são enviados para o modelo que utiliza do aprendizado supervisionado. Além disso, criou-se uma função para refazer a remontagem da sequência com base nas predições do modelo, métrica esta que exige uma compatibilidade integral entre a sequência final obtida pelo modelo e a esperada para ser considerado um acerto. A criação da rede neural se deu pela utilização da biblioteca Tensorflow e Keras, com o uso de um par de camadas de convolução e *Max Pooling*, seguida de uma camada de *Flatten*, duas camadas densas e uma camada densa com função de ativação *softmax* para a classificação. Os resultados obtidos pelo modelo (para um treinamneto de 1000 épocas) obtiveram uma acurácia promissora, de 0,8513 para o *Colletotrichum* e 0,6987 para o *Diaporthe*, porém, menor que a de outros modelos para a base de dados equivalente. Além disso, após usar o modelo para predizer íntrons ao longo de uma sequência e, remonta-la retirando as regiões não codantes, notou-se uma acurácia ainda menor que o de outros trabalhos, de 0,1628 para o *Colletotrichum* e 0,0000 para o *Diaporthe*. A probabilidade de acerto de uma predição do modelo, somada a distorções que ocorrem nas sequências ao realizar redimensionamento das imagens. contribuiu fortemente para que os resultados na métrica de remontagem não fossem promissores.

Palavras-chave: Aprendizado de Máquina. Redes Neurais Convolucionais. *Splicing Sites*.

Abstract

This work seeks an implementation of a convolutional neural network to identify coding and non-coding regions in DNA sequences. The database was obtained from GenBank, with 7003 sequences of fungi *Colleotrichum* in the Actin protein and 3878 sequences of fungi *Diaporthe* in the beta tubulin protein. The code architecture was based on the creation of multiple modules. In the first module the data retrieved from GenBank is filtered to that only information relevant to the job remains. In the second module are generated the possibilities of introns, exons and neither (a mixture of introns and/or exons). At the third module the sequences are transformed to grayscale image and then normalized, resizing them all to a calculated average size, and, with that, the division of the data for training and validation, which are sent to the model that uses the learning supervised to train. In addition, a function was created to redo the reassembly of the sequence based on model predictions, a metric that requires full compatibility between the final sequence obtained by the model and the one expected to be considered a hit. The neural network was created by the Tensorflow and Keras libraries, using a pair of convolutional layers and max pooling, followed by a flatten layer, two dense layers and a last dense layer with a softmax activation function to classify the classes. The results obtained by the model (on 1000 epochs trainning) obtained a promising accuracy of 0.8513 for *Colletotrichum* and 0.6987 for *Diaporthe*, however, lower than that of other models for the equivalent database, moreover, after using the model to predict introns along a sequence and, reassembling it removing the non-coding regions, an even lower precision was noted than that of other works, of 0.1628 for *Colletotrichum* and 0.0000 for *Diaporthe*. The probability of the model being correct in a prediction, added to the distortions that occur in the sequences when resizing the images, contributed strongly to the results in the reassembly metric not being promising.

Keywords: Machine Learning. Convolutional Neural Network. *Splicing Sites*.

Lista de Ilustrações

Figura 2.1 – Figura ilustrativa da localização e composição do DNA. 5
Figura 2.2 – Figura ilustrativa dos éxons e íntrons no DNA, RNA e mRNA. 6
Figura 2.3 – Figura da tabela de tradução de códons. 7
Figura 2.4 – Figura ilustrativa dos problemas de classificação e regressão. 9
Figura 2.5 – Figura ilustrativa de um neurônio biológico. 10
Figura 2.6 – Figura ilustrativa de um neurônio matemático. 11
Figura 2.7 – Figura ilustrativa dos diferentes modelos de redes neurais. 13
Figura 2.8 – Figura ilustrativa do processo de convolução para uma imagem 5x5, *Kernel* 2x2 e passo 1x1. 15
Figura 2.9 – Figura ilustrativa da Matriz de Confusão. 17
Figura 3.1 – Figura representativa da arquitetura modular criada. 20
Figura 3.2 – Figura ilustrativa da construção das possibilidades de íntrons e éxons. 21
Figura 3.3 – Figura ilustrativa do processo de conversão das bases nitrogenadas em meio textual para uma imagem bidimensional (cores invertidas). 22
Figura 4.1 – Gráfico ilustrativo da evolução da acurácia do modelo durante o treinamento de 100 épocas para o *Colletotrichum*. 26
Figura 4.2 – Gráfico ilustrativo da acurácia do modelo durante o treinamento de 100 épocas para o *Diaporthe*. 27
Figura 4.3 – Gráfico ilustrativo da acurácia do modelo durante o treinamento de 1000 épocas para o *Colletotrichum*. 27
Figura 4.4 – Gráfico ilustrativo da acurácia do modelo durante o treinamento de 1000 épocas para o *Diaporthe*. 28

Lista de Tabelas

Tabela 3.1 – Dados obtidos do Genbank para construção da base de dados. 19
Tabela 4.1 – Medidas obtidas pelo modelo durante o treinamento de 100 épocas para o *Colletotrichum*. 25
Tabela 4.2 – Medidas obtidas pelo modelo durante o treinamento de 100 épocas para o *Diaporthe*. 26
Tabela 4.3 – Comparação da acurácia obtida para 100 e 1000 épocas de treinamento. . . 28
Tabela 4.4 – Medidas obtidas na métrica de remontagem de sequência. 28

Lista de Abreviaturas e Siglas

PET	Programa de Educação Tutorial
AM	Aprendizado de Máquina
IA	Inteligência Artificial
ENIAC	*Eletronic Numerical Integrator and Computer*
CRF	*Conditional Random Fields*
CNN	*Convolutional Neural Network*
DNA	Ácido Desoxirribonucleico
RNA	Ácido Ribonucleico
mRNA	Ácido Ribonucleico Mensageiro
A	Adenina
C	Citosina
G	Guanina
T	Timina
U	Uracila

Sumário

1 Introdução 1
1.1 Justificativa 2
1.2 Objetivos 3
1.3 Organização do Trabalho 3
2 Revisão Bibliográfica 4
2.1 Conceitos da Biologia 4
2.1.1 DNA 4
2.1.2 *Splicing* e tradução 5
2.2 Conceitos da Computação 7
2.2.1 Aprendizado de Máquina 7
2.2.1.1 Aprendizado supervisionado 8
2.2.1.2 Aprendizado Não Supervisionado 9
2.2.1.3 Aprendizado Por Reforço 9
2.2.2 Redes Neurais 10
2.2.2.1 O Neurônio Biológico e o Matemático 10
2.2.2.2 Funcionamento das Redes Neurais 12
2.2.2.3 Funções de ativação 13
2.2.3 Redes Neurais Convolucionais - CNN 14
2.2.4 Métricas de Desempenho 16
2.3 Trabalhos Relacionados 17
3 Desenvolvimento 19
3.1 Materiais e métodos 19
3.2 Módulos 20
3.2.1 Módulo de Seleção 21
3.2.2 Módulo de Processamento 21
3.2.3 Módulo de Preparação e Treinamento 22
3.2.4 Métrica de Remontagem das Sequência 23
4 Resultados 25
4.1 Resultados Experimentais 25
4.2 Discussão dos resultados 28
5 Considerações Finais 30
5.1 Conclusão 30
5.2 Trabalhos Futuros 31

Referências 32

1 Introdução

Ao longo do tempo a computação evoluiu significativamente até chegar aos dias atuais, sendo aplicada em praticamente todas as áreas de conhecimento. Desde a criação do ENIAC (*Electrical Numerical Integrator and Calculator*) em 1946, os pesquisadores, cientistas, engenheiros e entusiastas da computação trabalham incessantemente para obter um *hardware* cada vez mais poderoso. Essa evolução possibilitou o desenvolvimento de diversas aplicações que resolvemproblemas cada vez mais específicos. Com o passar dos anos, os problemas mais fáceis, isto é, aqueles cuja solução computacional poderia ser pensada de maneira relativamente imediata foram se extinguindo e problemas extremamente difíceis, computacionalmente e humanamente falando, emergiram. Perto da data da criação do computador, já existiam pesquisas do que mais tarde se tornaria o que hoje conhecemos como a inteligência artificial (IA). Um exemplo é a pesquisa de McCulloch e Pitts (1943), que descreveu a criação do primeiro modelo de neurônios artificiais e protagonizaria o início da "gestação da inteligência artificial" no século XX.

Ao contrário do desenvolvimento dos computadores, que seguem a Lei de Moore[1], a inteligência artificial teve um declínio muito grande quando os pesquisadores perceberam que os problemas que poderiam ser resolvidos por ela se limitavam ao escopo do "simples", isto é, problemas que não precisassem de um aprendizado/processamento complexo, envolvendo memória de curto ou longo prazo e reconhecimento de padrões extremamente específicos. Com a dose de realidade que emanou dos problemas mais complexos e interessantes, a evolução da IA se deu através da elucidação de que, para solucionar problemas complexos, a abordagem precisava de conhecimento total ou de grande parte do cenário sendo tratado.

Problemas mais e mais complexos começaram a ser tratados, diferentes modelos e métodos de aprendizado de máquina (AM) foram criados, buscando melhores resultados para solucionar problemas e, somando-se a isso o crescimento do poder computacional, os grandes desafios começaram a serem superados.

Analogamente à evolução da computação, a biologia e a biotecnologia tiveram grandes avanços em meados do século XX. Como ocorreu em 1953, quando houve a descoberta de que o DNA é composto por uma dupla hélice de nucleotídeos. Este foi o alicerce para pesquisas que mais tarde permitiram entender, mesmo que de maneira parcial, o que fundamenta todo indivíduo: a evolução e genética.

Um processo que é muito importante para a manutenção da vida do indivíduo é o processo de *splicing* (CHOW *et al.*, 1977), que mapeia quais informações da cópia do DNA, o RNA mensageiro (mRNA), serão mantidas para a síntese de proteínas. Dos nucleotídeos do mRNA,

[1] Proposta em 1970 por Gordon E. Moore, a regra afirma que a velocidade/capacidade de processamento dos computadores dobraria a cada dois anos. Existem muitas variáveis a serem consideradas neste aspecto, mas, em termos gerais, a lei continua sendo válida até os dias atuais.

nem todos os formam proteínas, muitas partes dele são descritas como íntrons - regiões que não irão participar do processo de síntese de proteínas. Por outro lado, as regiões que de fato serão usadas, são chamadas de éxons.

Este trabalho pretende continuar os estudos iniciados por Ferreira (2019) e explorados por Santos (2021) e Cruz e Menossi (2021), partindo de uma abordagem diferente das propostas anteriores: fazer uso do modelo de redes neurais artificiais para tentar criar um software capaz de identificar regiões codantes do mRNA com uma boa acurácia.

1.1 Justificativa

Os especialistas da área da biologia e biotecnologia ainda têm de realizar a identificação das regiões de *splicing* manualmente, de forma que usam a proteína que foi gerada para realizar o processamento inverso e descobrir as regiões de *splicing*, ou procuram outros métodos que tentem identificar a partir de exemplos próximos (SONNENBURG *et al.*, 2002). Os resultados de trabalhos como o de Ferreira (2019) ou Sonnenburg *et al.* (2007), que fazem uso do algoritmo *Conditional Random Fields*, apontam para padrões na identificação de íntrons.

Modelos de redes neurais artificiais atualmente têm demonstrado ótimos resultados quando aplicados em problemas de reconhecimento de padrões em sequências de textos, e, no contexto da identificação de regiões de *splicing*, existem trabalhos cujos resultados são muito promissores.

O trabalho de Albaradei *et al.* (2020), conseguiu uma acurácia de 96,91% em seu software que faz uso de um modelo baseado em CNN para sequências de *Homo sapiens* e precisões acima de 90% para outros quatro organismos (*Arabidopsis thaliana*, *Oryza sativa japonica* e *Carnorthabditis elegans*). O trabalho de Naito (2018), conseguiu uma acurácia de 92,91% (*Donor Site* - GT) e 89,45% (*Acceptor Site* - AG) na identificação de regiões de *splicing* para o *Homo Sapiens*, utilizando um modelo de CNN e 93,33% e 89,87% utilizando um modelo híbrido de CNN e LSTM.

A principal motivação é a criação de um software de fácil acesso e uso que seja capaz de identificar regiões de *splicing* nos mais diversos organismos e nas mais diversas proteínas, utilizando um modelo genérico.

Além disso, o processo de extração de características para o problema não é intuitivo - por se tratar de bases nitrogenadas. Neste aspecto, o uso de um modelo com aprendizagem profunda, capaz de aprender representações de forma automática, é muito útil e remove a necessidade de criar e extrair características para o modelo.

Como diferencial em relação aos trabalhos já existentes, buscou-se uma modelagem capaz de receber uma sequência de entrada de fungos filamentosos e devolver a sequência remontada com os íntrons já identificados e removidos. Grande parte das abordagens atuais

apenas identificam *splicings sites*, mas não íntrons inteiros.

Tendo em vista os bons resultados obtidos por outros autores, a crescente utilização de modelos de redes neurais convolucionais e a extração de características de forma automática, a CNN foi o modelo de AM escolhido para a implementação nesse trabalho.

1.2 Objetivos

O objetivo deste trabalho foi a criação de um modelo de AM, baseado em redes neurais artificiais, que possa identificar as regiões codantes do DNA e auxiliar no processo de identificação de regiões de *splicing*. Como objetivos específicos pretende-se:

- Modelar o problema de identificação de íntrons e éxons como um problema de aprendizado de máquina supervisionado.
- Treinar um modelo de redes neurais artificiais, mais especificamente uma rede neural convolucional.
- Realizar o treinamento usando pelo menos duas espécies de fungos filamentosos.
- Validar os resultados obtidos, comparando os resultados com outras abordagens.

1.3 Organização do Trabalho

Na seção 1 é apresentada uma breve contextualização do problema. Na seção 2 é onde são apresentados todos os conceitos necessários para a compreensão do trabalho, bem como trabalhos relacionados. Na seção 3 é descrito o desenvolvimento propriamente dito do trabalho. Na seção 4 são apresentados os resultados obtidos pelo trabalho. Por fim, na seção 5 são apresentadas as conclusões e trabalhos futuros.

2 Revisão Bibliográfica

Este trabalho constitui-se na aplicação de técnicas de Aprendizado de Máquina a um problema da biologia, especificamente, a identificação de íntrons e éxons. Sendo assim, nesta seção serão apresentados os conceitos básicos necessários de ambas as áreas para que haja o entendimento do trabalho, bem como trabalhos relacionados.

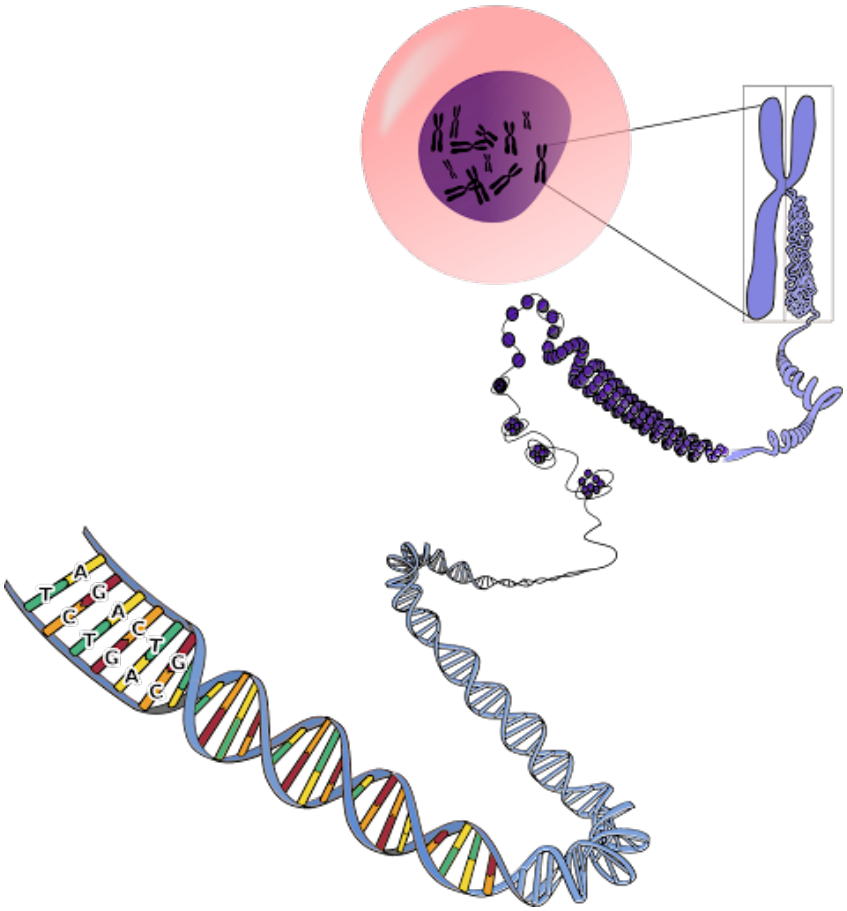
2.1 Conceitos da Biologia

Para total compreensão do problema, nesta seção serão apresentados conceitos sobre o gene, DNA, RNA, processo de transcrição e geração de proteína - *splicing* e tradução.

2.1.1 DNA

O ácido desoxirribonucleico (ADN), ou DNA, como é mais comumente conhecido (sigla abreviada do termo em inglês *deoxyribonucleic acid*), descoberto oficialmente pelos pesquisadores Francis Crick e James Watson em 1953, é uma estrutura vital para o funcionamento dos seres vivos, armazenando informações para a construção de proteínas (o principal responsável por possibilitar a sobrevivência de cada célula), além de coordenar o desenvolvimento e funcionamento dos seres vivos, transmitindo características hereditárias para os descendentes. Está localizado principalmente nos cromossomos e mitocôndrias das células. Os segmentos do DNA que possuem informações genéticas são denominados genes, descritos inicialmente por Gregor Mendel (1822-1864), e o restante do DNA é responsável por estruturar e regular essa informação genética. O DNA é constituído por estruturas que se repetem ao longo de seu corpo, estas chamadas de nucleotídeos. O DNA é um par de moléculas associadas que formam uma dupla hélice, na qual os nucleotídeos são unidos por ligações de fosfodiéster. A dupla hélice é ligada por pontes de hidrogênio entre as bases, sendo que as quatro bases encontradas no DNA são: Adenina (A), Citosina (C), Guanina (G) e Timina (T), de forma que a Adenina se liga à Timina e a Citosina se liga à Guanina no processo conhecido como ligação dos pares das bases. Por meio dessas ligações é possível armazenar uma fita complementar que possibilita a replicação do DNA (WATSON *et al.*, 2015). A Figura 2.1 ilustra a localização e composição do DNA.

Figura 2.1 – Figura ilustrativa da localização e composição do DNA.



Fonte: Pixabay.

2.1.2 *Splicing* e tradução

Para ocorrer a síntese de proteínas, o DNA é replicado e gera o RNA mensageiro (mRNA), onde a base Timina é substituída pela Uracila (U). Uma vez criado, ele é liberado no citoplasma[1] da célula até encontrar os ribossomos[2], onde acontecerá o processo de tradução (WATSON *et al.*, 2015). O processo de síntese de proteínas transforma as sequências de mRNA em proteínas, porém, nem toda a sequência dá origem à proteínas, algumas partes não são codantes - os chamados íntrons, enquanto que as regiões codantes são chamadas de éxons (SHARP, 2005).

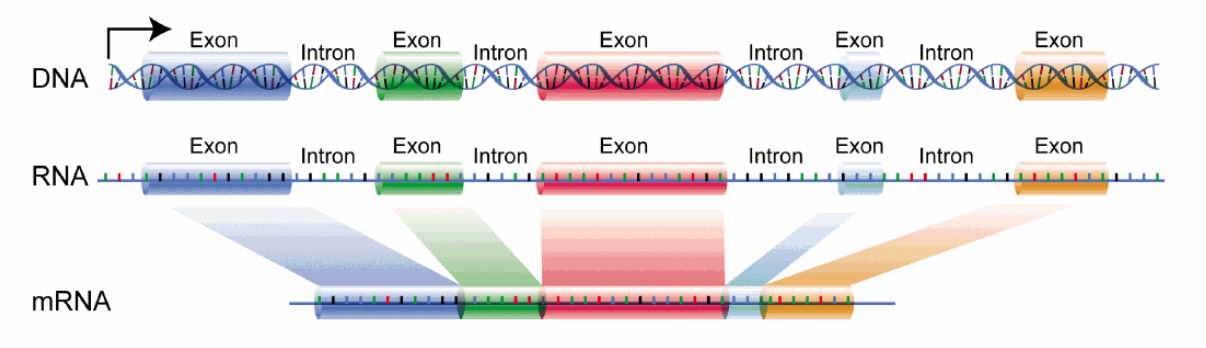
Todos os organismos eucariontes (que possuem células complexas contando com um núcleo bem definido que envolve o material genético e diversas estruturas para o seu funcionamento) possuem íntrons em suas estruturas genéticas e antes que possam ser formadas as proteínas, ocorre a retirada dos íntrons e ligação do éxons remanescentes - no processo conhecido como *splicing* (CHOW *et al.*, 1977).

Na síntese de proteínas, há padrões de nucleotídeos quase invariáveis no processo de

[1] Área do interior de uma célula, que abrange todas as estruturas dela.
[2] Estruturas no interior da célula responsáveis pela produção de proteínas.

identificação de íntrons, esses padrões são chamados de "sequências consenso", que é descrito pela presença das bases GU no início e AG no final (ALBERTS *et al.*, 2017). Um exemplo ilustrativo da localização dos éxons do RNA em relação ao DNA é mostrado na Figura 2.2.

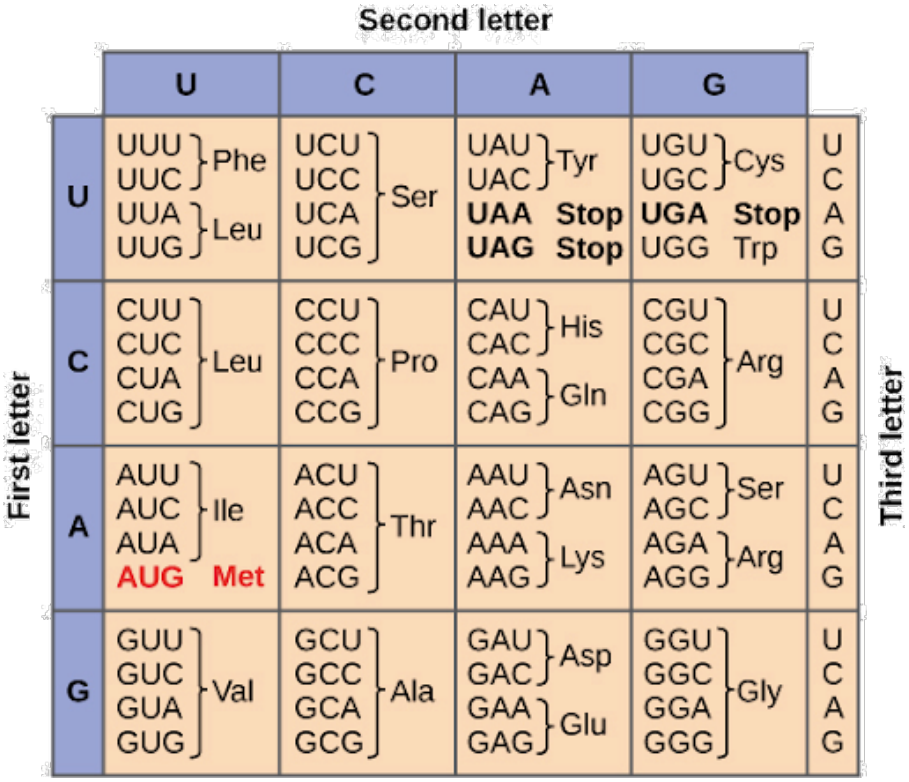
Figura 2.2 – Figura ilustrativa dos éxons e íntrons no DNA, RNA e mRNA.



Fonte: Wikimedia Commons.

Uma vez que o processo de *splicing* é concluído, a tradução é iniciada. Este é um processo que consiste em traduzir uma sequência de nucleotídeos do mRNA em uma sequência de aminoácidos chamados códons. A Figura 2.3 mostra possíveis combinações de trincas e os correspondentes aminoácidos traduzidos por cada uma. Dentro das traduções, o códon AUG, produtor da Metionina (Met), está destacado pois representa o códon de início de tradução, e as trincas UAA, AUG, e UGA representam os possíveis códons de finalização do processo de tradução.

Figura 2.3 – Figura da tabela de tradução de códons.



Fonte: "O código genético", OpenStax College, Biology (CC BY 3.0).

2.2 Conceitos da Computação

A seguir serão introduzidos e explicados os conceitos relacionados à Aprendizado de Máquina (AM), Redes Neurais, Redes Neurais Convolucionais e Métricas de Desempenho.

2.2.1 Aprendizado de Máquina

Uma área da computação, que possui contribuições nas mais diversas áreas é a Inteligência Artificial (IA). A IA pode ser definida como o campo da ciência da computação dedicada à automação do comportamento inteligente (STUBBLEFIELD; LUGER, 1992). O Aprendizado de Máquina (AM) é uma subárea da IA, que tem como objetivo construir sistemas capazes de obter conhecimento de forma automática (MONARD; BARANAUSKAS, 2003). O AM nem sempre esteve atrelado diretamente à computação, como o modelo proposto por Warren McCulloch e Walter Pitts em 1943 (MCCULLOCH; PITTS, 1943) que, baseado na estrutura biológica do neurônio, apresentava um modelo de AM baseado em neurônios artificiais. Em 1956, John McCarthy, Minsky, Claude Shannon e Nathaniel Rochester definiram Inteligência Artificial como uma área de pesquisa e Allen Newell e Herbert Simon apresentaram o primeiro programa de raciocínio - o *Logic Theorist*. Entre 1952 e 1969 ocorreu o grande “boom” do desenvolvimento da IA.

Diversos pesquisadores começaram a desenvolver programas projetados para resolver problemas de maneira similar aos humanos, como o *General Problem Solver*, desenvolvido por Newell e Simon, ou o *Geometry Problem Solver*, por Nathaniel e Herbert. Em 1952, Arthur Samuel apresentou uma série de programas feitos para jogar damas que eram capazes de armazenar informações e aprender. Herbert Simon fez, em 1966, uma previsão que os computadores dentro de dez anos conseguiriam provar um teorema matemático significativo e vencer um campeão de xadrez. Porém, no final do século XX, uma dose da realidade foi dada a todos: os problemas reais, definidos pela intratabilidade humana e pela NP-completude[3] na computação começaram a surgir e a criação de uma máquina superinteligente vira um mito.

Com a evolução da computação somada aos modelos de AM cada vez mais complexos e eficientes, um novo cenário foi visto no horizonte: a possibilidade de resolver problemas complexos, mesmo que não seja de maneira exata 100% das vezes. Atualmente, o AM pode contar com uma quantidade de dados imensa em redes de aprendizagem extremamente complexas - o que nos traz: a IA aplicada no planejamento autônomo e escalonamento (pela NASA), uma IA capaz de vencer um campeão mundial de xadrez (pela IBM), microcirurgias auxiliadas por robôs, aplicativos de reconhecimento de linguagem humana e resolução de problema (como a Siri da Apple).

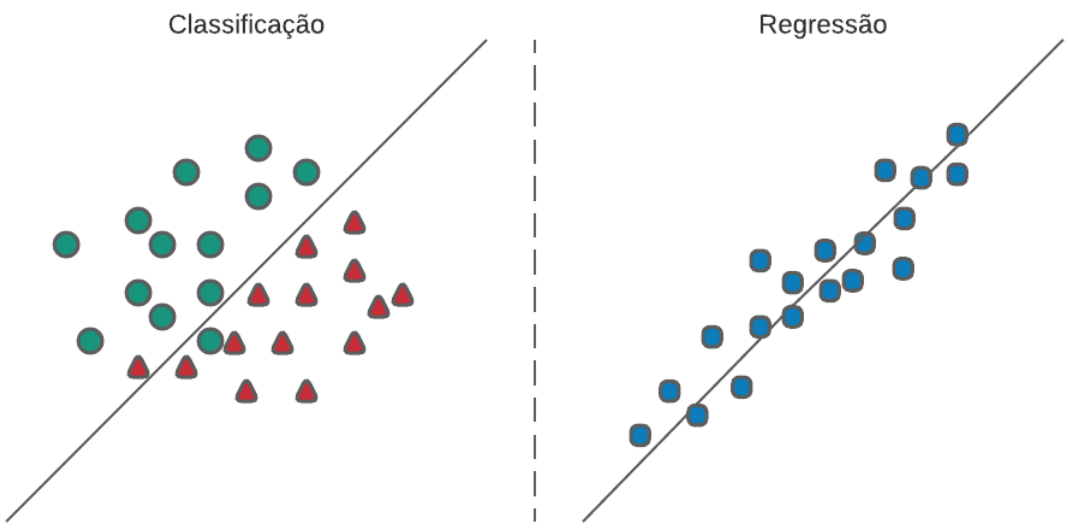
O AM pode ser dividido em três principais tipos de abordagens: aprendizado supervisionado, aprendizado não supervisionado e aprendizado por reforço.

2.2.1.1 Aprendizado supervisionado

O aprendizado supervisionado é uma das formas de aprendizado mais comuns e mais utilizadas atualmente. Ela se baseia na ideia de ensinar à máquina partindo de exemplos já existentes e bem consolidados, trazendo não só a pergunta ou os dados de interesse, mas também a classificação/rótulo desses exemplos. Como exemplo desde tipo de aprendizado, podemos citar a identificação de rostos em fotos tendo à disposição milhares de fotos de diferentes pessoas, juntamente com a identificação (rótulo) dessas pessoas. Nesse tipo de aprendizado, de acordo com Monard e Baranauskas (2003), o modelo é treinado com base em um conjunto de exemplos e seus respectivos rótulos, a fim de possibilitar a construção de um classificador que possa determinar com êxito a classe de exemplos ainda não rotulados. Quando o rótulo esperado dos exemplos é discreto, como o nome da pessoa ou a sua etnia, por exemplo, tem-se um problema de classificação (Figura 2.4). Quando o rótulo é um valor contínuo, como o valor de venda de uma casa, por exemplo, tem-se um problema de regressão (Figura 2.4) (CHOWDHURY, 2003). Algumas técnicas famosas de aprendizado supervisionado incluem: regressão linear, regressão logística, redes neurais ou k-vizinhos mais próximos.

[3] Definição da ciência da computação para problemas com tempos de solução intratáveis.

Figura 2.4 – Figura ilustrativa dos problemas de classificação e regressão.



Fonte: Próprio Autor

2.2.1.2 Aprendizado Não Supervisionado

O aprendizado não supervisionado é uma técnica de aprendizagem que aprende padrões a partir de dados não rotulados. Diferente do aprendizado supervisionado, no qual os dados são rotulados previamente por um humano como, uma "maçã" ou "laranja", o aprendizado não supervisionado utiliza-se de uma auto-organização ou auto-classificação a qual captura padrões presentes nos dados formando agrupamentos (*clusters*) de instâncias semelhantes. Uma forma de medir a distância ou similaridade entre as instâncias faz com que o algoritmo possa encontrar padrões nos dados, sendo indicado para uso quando deseja-se encontrar valores, classificações ou padrões e não se conhece previamente de maneira clara quais são eles (MCGREGOR *et al.*, 2004).

2.2.1.3 Aprendizado Por Reforço

O aprendizado por reforço é uma técnica de aprendizagem que consiste no mapeamento de estados em ações de forma que por meio do mapeamento dessas ações, busca-se maximizar ou minimizar um determinado valor, conhecido também como recompensa. Não é necessário que o agente (ou modelo a ser treinado) saiba inicialmente das ações a serem tomadas, mas ele deve ao longo do treinamento descobrir quais ações o levam a obter os melhores valores de retorno. Essa técnica também é inspirada em um conceito da psicologia, o qual recompensas e punições são direcionadas a um agente dependendo de suas ações no ambiente. Com a repetição desses passos é esperado que o agente (nomenclatura comumente usada no AM para descrever o objeto de estudo responsável por predizer) passe a entender ações que evitam punições e maximizam recompensas no ambiente que o cerca (GéRON, 2019).

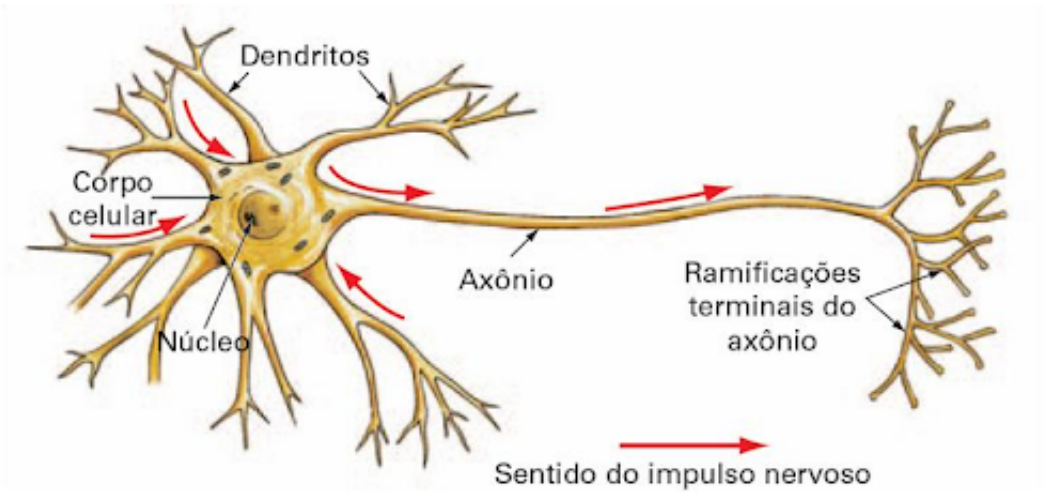
2.2.2 Redes Neurais

As redes neurais artificiais são modelos computacionais inspirados no sistema nervoso central dos animais, responsáveis por captar e transmitir informações para o corpo inteiro. Analogamente a uma rede neural biológica, as redes neurais artificiais são compostas por neurônios que são sensíveis a algumas entradas e a união deles define uma resposta a ser fornecida pelo modelo. A primeira rede neural aplicada a um problema real foi a "MADALINE", criada para reconhecer padrões binários, em 1959, por Bernard Widrow e Marcian Hoff. As redes neurais atualmente são utilizadas em larga escala para as mais diversas finalidades, o que gerou inúmeros modelos de aplicação com inúmeros focos específicos.

2.2.2.1 O Neurônio Biológico e o Matemático

O neurônio é uma estrutura de vasta importância tanto para o modelo de redes neurais quanto para qualquer animal. Estima-se que o cérebro humano de uma pessoa "normal" e "saudável" possui cerca de 86 bilhões de neurônios. O neurônio biológico é especialista na transmissão de informações e todos estão conectados por centenas de bilhões de conexões entre eles, formando uma enorme rede de comunicação: a rede neural biológica. Um neurônio biológico é dividido em três partes principais: o corpo celular - do qual surgem ramificações, os dendritos e os axônios, como é ilustrado a seguir.

Figura 2.5 – Figura ilustrativa de um neurônio biológico.



Fonte: Data Science Academy. Deep Learning Book. Capítulo 4.

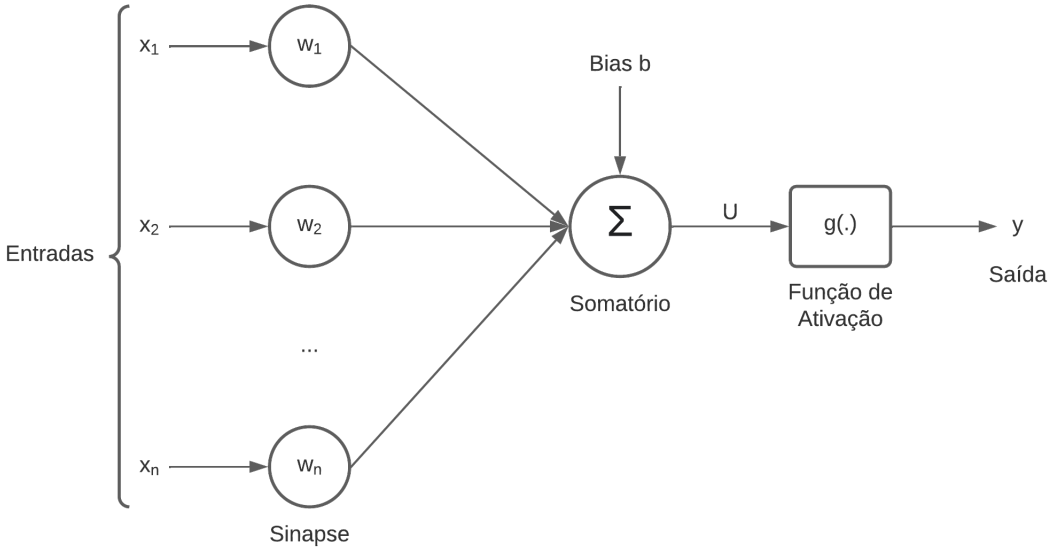
Cada neurônio possui um corpo central com diversos dendritos e apenas um axônio. Os dendritos são receptores de sinais elétricos de outros neurônios. O corpo central é o "processador" das informações recebidas e o axônio é o transmissor da informação gerada.

O funcionamento do neurônio é dado da seguinte forma: os sinais elétricos são gerados externamente, através da visão, toque, papilas gustativas, etc, e caminham pelos axônios. Se eles

forem superiores a um limiar de disparo, o chamado *threshold*, eles são transmitidos a seguir, caso contrário, são bloqueados. Os valores/pesos com quais o limiar é atingido é afetado conforme a região do cérebro onde aquele neurônio se encontra e mudam conforme a vida útil do neurônio - e de seu portador. Podemos encarar o funcionamento de uma rede neural como uma grande cadeia que gera inúmeros bits de informação que possuem um significado para o animal, mesmo que ainda sejamos incapazes de decifrar com precisão essa informação.

Um neurônio matemático possui a mesma ideia de funcionamento. Proposto em 1943 por Warren McCulloch e Walter Pitts (MCCULLOCH; PITTS, 1943), ele é um componente, entre inúmeros outros iguais dentro de uma rede neural, que calcula a soma ponderada de várias entradas, aplica uma função e envia (ou não) o resultado gerado adiante. Na Figura 2.6 é mostrada uma ideia do modelo de um neurônio matemático.

Figura 2.6 – Figura ilustrativa de um neurônio matemático.



Fonte: Baseado em (ACADEMY, 2022)

O modelo matemático como proposto acima se resume a um modelo simplificado do biológico, o neurônio recebe um ou mais sinais e devolve um único sinal de saída, os dendritos e axônios são representados matematicamente apenas pelas sinapses e a ligação deles é o peso sináptico. As entradas, descritas por $x_1, x_2, \ldots, x_n$ na Figura 2.6, são multiplicadas por seus respectivos pesos, resultando em uma unidade homogênea bidimensional de valores, que são enviados a uma função, que simula o processo de *threshold*, que avaliará se o limiar daquele neurônio foi atingido e, caso tenha sido, a saída gerada é um sinal de verdade (1), caso contrário, é um sinal de falso (0).

A rede neural, e por consequência os neurônios, está suscetível a funções de ativações para definir seus *thresholds*, que podem ser definidas das mais diversas formas, de maneira a otimizar a predição obtida.

No neurônio matemático, temos as seguintes convenções e nomenclaturas:

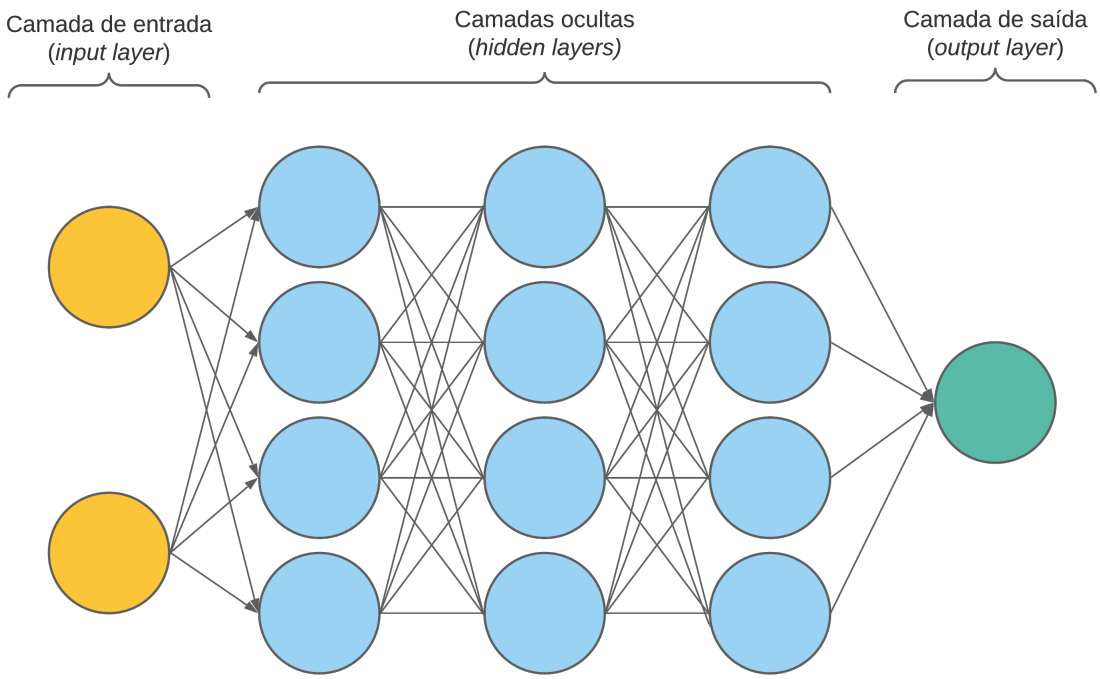
- Os dados de entrada são definidos como os sinais de entrada, definidos por { x_1, x_2, ..., x_n }, que são sinais exteriores normalmente normalizados, vindos de outros neurônios ou diretamente dos inputs (entradas) dos dados a serem processados pelo modelo.
- Os pesos são definidos como os pesos sinápticos { w_1, w_2, ..., w_n }, que irão multiplicar cada respectivo valor dos sinais de entrada.
- A soma ponderada da multiplicação dos pesos pelos sinais de entrada são feitas no combinador linear { Σ }.
- O limiar de ativação { ϑ } é a especificação do limite para que o combinador linear gere um disparo de ativação.
- O potencial de ativação { u } é a diferença entre combinador linear e o limiar de ativação, quando positivo, gera um potencial excitatório, senão, inibitório.
- A função de ativação { g } é a função responsável por limitar a ativação da saída.
- Por fim, o sinal de saída { y } é o valor final a ser passado adiante na rede neural.

2.2.2.2 Funcionamento das Redes Neurais

As redes neurais são compostas por camadas, contendo inúmeros neurônios em cada uma. Diversos tipos de redes foram criadas, mas, em geral, a estrutura básica de uma rede neural se define da seguinte forma (ilustrado na figura 2.7):

- A camada mais à esquerda é a camada de entrada de dados (*input layer*).
- A camada mais à direita é a camada por onde o resultado da rede é dado (*output layer*).
- A(s) camada(s) internas (*hidden layers*) são definidas conforme a necessidade para a resolução do problema e são compostas por camadas ocultas de neurônios, podendo contar com inúmeras camadas cuja entrada é dada de uma camada de neurônios para outra, bem como a sua saída.

Figura 2.7 – Figura ilustrativa dos diferentes modelos de redes neurais.



Fonte: Próprio Autor.

Quando o modelo de aprendizado de máquina utiliza níveis hierárquicos de camadas, com diversas camadas ocultas (*hidden layers*), ele é caracterizado como um modelo de aprendizagem profunda.

Além disso, em redes neurais, o treinamento se dá em épocas (ou *epochs*, em inglês). Cada época é uma iteração sobre os dados de entrada de forma aleatória.

2.2.2.3 Funções de ativação

Criadas com o objetivo de fornecer ao neurônio um mecanismo matemático para quando ele dará resultado (será ativo) ou não - funcionando de maneira análoga a um interruptor, as funções de ativação permitem que o modelo tenha a capacidade não-linear de processamento. Existem diversos tipos de funções de ativação, cada uma possibilita o modelo mapear as soluções de uma forma única. Algumas das principais funções de ativação existentes hoje são descritas a seguir:

- Linear: é a função de ativação mais simples dentre as mais comumente usadas. Ela se baseia em uma função linear e apenas aplica um fator de multiplicação ao valor que recebe.
- Sigmoid: é uma função eficiente quando se trata de não-linearidade, produzindo valores dentro do intervalo [0,1].

- Softmax: é uma função que generaliza a função sigmoid para casos não-binários - sendo muito utilizada para definir problemas de classificação com multi-classes, já que sua característica é produzir valores no intervalo de [0, 1] onde a soma de todos os valores gerados é igual a 1. Desse modo, a saída gerada pode ser interpretada como uma distribuição de probabilidade.

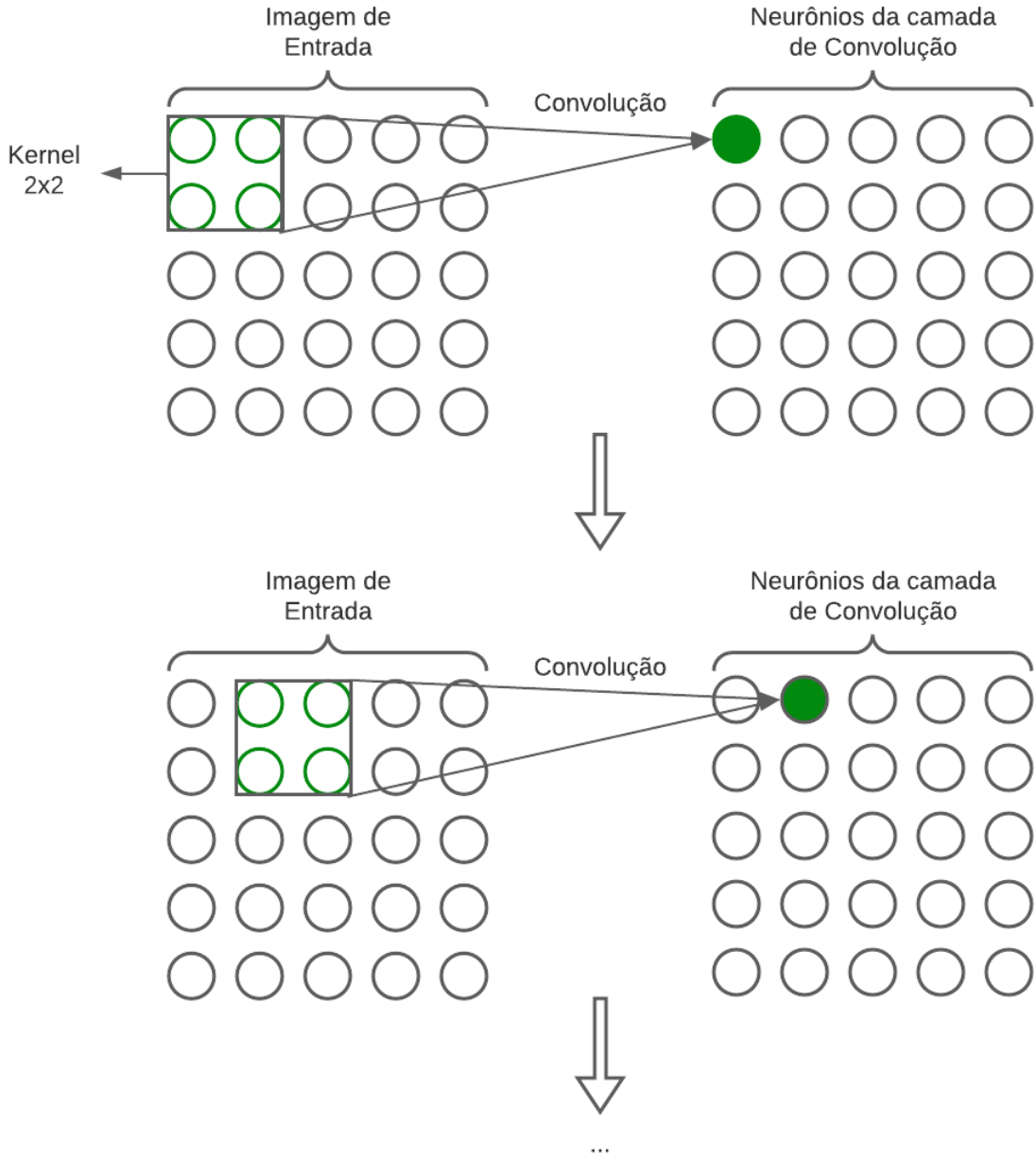
- ReLU: abreviação para *rectified linear unit* (unidade linear retificadora), ela produz resultados dentro do intervalo $(0, \infty)$. Para valores negativos, ela retorna o valor zero, enquanto que, para valores maiores que zero é o próprio valor que é retornado. Atualmente é uma função de ativação padrão em diversas bibliotecas que implementam redes neurais (como a Keras, que foi utilizada neste trabalho) e vastamente utilizada, fornecendo um ganho de velocidade de processamento em função da sua simplicidade computacional.

2.2.3 Redes Neurais Convolucionais - CNN

Proposta por LeCun *et al.* (1998), é um modelo de rede neural de aprendizado profundo criado com enfoque inicial de ser aplicado na detecção de padrões em imagens, sendo capaz de compreender de forma automática padrões comuns (as chamadas *features*) nas imagens. A limitação de *hardware* na época do seu desenvolvimento fez com que ela só obtivesse adesão pela comunidade conforme a tecnologia se desenvolveu. Agora, com placas de processamento gráfico ou *Graphics Processing Unit* (GPU) cada vez melhores e mais acessíveis, a CNN tem ganhado grande força, sendo aplicada tanto em problemas envolvendo imagens, quanto em problemas adaptados, como é o caso deste trabalho.

A principal propriedade da CNN é a convolução. A convolução é uma técnica proposta por LeCun *et al.* (1998) que se baseia em técnicas do processamento de imagens (e, por consequência, na visão humana), a replicação dos campos receptivos humanos, nos quais cada neurônio tem a tarefa de analisar uma parte isolada da imagem de entrada (ACADEMY, 2022). Uma convolução é a extração de características de uma imagem a partir de uma janela deslizante de tamanho NxM que será aplicada por toda a imagem. Essa janela deslizante é descrita na CNN como o *kernel*, o valor padrão para o *kernel*. O costuma ser 2x2, mas pode ser alterado conforme a dimensionalidade das imagens da entrada. Esse processo é exemplificado na Figura 2.8

Figura 2.8 – Figura ilustrativa do processo de convolução para uma imagem 5x5, *Kernel* 2x2 e passo 1x1.



Fonte: Próprio Autor.

Outra camada que é muito comum ser aplicada junto à de convolução é a de *Pooling*, no qual, a partir de uma janela deslizante como a do *kernel* (pode-se customizar tanto o tamanho dessa janela quanto o seu passo - isso é, ao aplicar a janela, quantos *pixels* serão deslocados tanto horizontalmente quanto verticalmente), é aplicado um calculo matemático sobre os *pixels* contidos dentro da janela deslizante. Normalmente, são aplicados três tipos de *pooling*:

- Max: dentro dos *pixels* contidos na janela deslizante, o resultado será o valor do *pixel* com maior valor.

- Min: análogo ao anterior, porém, ao invés do maior valor, considera-se o menor.
- Avg: dentro dos *pixels* contidos na janela deslizante, o resultado será uma média simples entre os valores deles.

É comum encontrar a aplicação de múltiplas camadas de Convolução e *Pooling* em pares quando se têm uma CNN. Essa abordagem garante a extração de automática características profundas dentro da imagem.

Ao final da rede convolucional, comumente aplica-se camadas densas para fazer o mapeamento das características obtidas para valores de classes (quando o problema é de classificação) ou valores contínuos (quando o problema é de regressão).

2.2.4 Métricas de Desempenho

Para avaliar o desempenho de modelos de AM, criou-se diversas medidas com enfoques de mensurar a qualidade dele. As mais comumente utilizadas e difusas pela comunidade são a acurácia e erro, matriz de confusão, precisão, revocação e medida F1.

A medida acurácia é a representação da porcentagem de acertos do modelo pela quantidade total de predições realizadas, de forma que:

$$Acuracia = \frac{QuantidadeDeObservacoesClassificadasCorretamente}{TotalDeObservacoes}$$

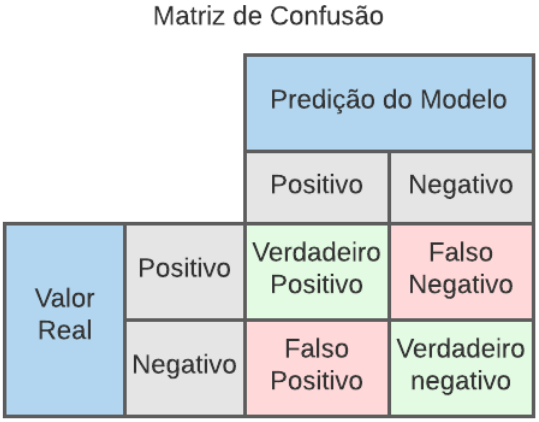
Já o erro, é definido como o complemento da acurácia, isto é:

$$Erro = 1 - Acuracia$$

A matriz de confusão é uma matriz bidimensional, indexada em uma dimensão pela classe verdadeira de um objeto na outra pela classe que o classificador atribui (KOHAVI; PROVOST, 1998). Dessa forma, a matriz de confusão é formada por quatro possíveis valores:

- Verdadeiro Positivo (*True Positive* - TP): valores positivos classificados corretamente como positivos pelo modelo.
- Verdadeiro Negativo (*True Negative* -TN): valores negativos classificados corretamente como negativos pelo modelo.
- Falso Positivo (*False Positive* - FP): valores negativos classificados erroneamente como positivos pelo modelo.
- Falso Negativo (*False Negative* - FN): valores positivos classificados erroneamente como negativos pelo modelo.

Figura 2.9 – Figura ilustrativa da Matriz de Confusão.



Fonte: Próprio Autor.

A partir da matriz de confusão, é possível definir algumas medidas de desempenho:

- Precisão: a precisão é definida contabilizando quantos dos valores classificados pelo modelo como verdadeiro eram de fato verdadeiros. É definida pela fórmula:

$$Precisao = \frac{TP}{TP + FP}$$

- Revocação: também é conhecida como *recall* ou sensibilidade. É calculada considerando quantos dos valores que deveriam ser classificados como verdadeira, realmente foram classificados como verdadeiros por ele. É definida pela fórmula:

$$Revocacao = \frac{TP}{TP + FN}$$

Uma vez definidos os valores de precisão e revocação, podemos calcular a medida F1, conhecida por ser a medida harmônica entre ambas precisão e revocação. É definida pela fórmula:

$$F1 = \frac{2 * (precisao * revocacao)}{precisao + revocacao}$$

2.3 Trabalhos Relacionados

Identificação das regiões de *splicing* ainda é um tema pouco explorado e que carece de ferramentas eficazes e de fácil utilização, mas que possui algumas contribuições significativas, como o trabalho de Sonnenburg *et al.* (2007), que apresenta um modelo vetorial de predição de

regiões codantes ou Pashaei e Aydin (2018), que apresenta um modelo baseado na Cadeia Oculta de Markov para identificar as regiões codantes ou, ainda, trabalhos que voltam o problema para a identificação usando redes neurais, como o trabalho de Albaradei *et al.* (2020), no qual foi criado um modelo usando redes neurais convolucionais para a predição de regiões de *splicing* em sequências de espécies de Homo sapiens, Oryza sativa japonica, Arabidopsis thaliana, Drosophila melanogaster, e Caenorhabditis elegans, apresentando resultados satisfatórios.

O problema de identificação de regiões codantes já foi abordado anteriormente por Ferreira (2019), que apresentou e explorou o tema, abordando uma técnica de aprendizado de máquina supervisionada utilizando o algoritmo *Conditional Random Fields* (CRF) para aprender e predizer as regiões codantes das sequências obtidas por ele. A pesquisa mostrou resultados bastante promissores, o que resultou na aplicação de mais pesquisas, conduzidas por Santos (2021) e Cruz e Menossi (2021), trazendo ainda a aplicação do CRF para a identificação de regiões codantes. Os resultados obtidos por esses autores foram satisfatórios, cada um com seus próprios adendos para o desenvolvimento do tema, como a identificação por meio de trincas por Santos (2021). ou o impacto de sequências degeneradas por (CRUZ; MENOSSI, 2021), ou, ainda, a criação de uma métrica baseada na remontagem de sequências vindas dos exemplos, em ambos os trabalhos.

Trabalhos como o de Albaradei *et al.* (2020) e Naito (2018) utilizam abordagens que fazem uso de CNN com outros modelos, atingindo os melhores resultados que constam na literatura observada. O trabalho de Albaradei *et al.* (2020) descreve ainda uma conversão de regiões de *splicing* para imagens, através da conversão das quatro bases presentes no DNA para listas unidimensionais de quatro posições, cada posição representando uma das possíveis bases, e, com a soma de diversas listas, gera-se uma imagem de saída. Essa modelagem de conversão de bases para imagem foi forte inspiração para o desenvolvimento do Módulo de Preparação e será explorada nele.

3 Desenvolvimento

Esta seção descreve todos os passos e informações necessárias para a replicação do estudo desenvolvido, bem como estrutura os procedimentos e técnicas utilizados. Na seção 3.1 são expostas as ferramentas que foram utilizadas no trabalho, incluindo linguagem e banco de dados. Na seção 3.2 é discutido como foi feito o pré-processamento dos dados e a criação da rede neural, bem como a geração de métricas de desempenho.

3.1 Materiais e métodos

Para o desenvolvimento do trabalho, optou-se pelo uso da linguagem Python, que já possui uma vasta coleção de bibliotecas e pacotes que auxiliam no pré-processamento de dados e na criação das mais diversas redes neurais e modelos de aprendizado de máquina. Para o pré-processamento dos dados, foram utilizadas as bibliotecas NumPy, BioPython e Pillow, enquanto que para o treinamento da rede neural convolucional, bem como realizar predições, expor medidas de desempenho e salvar e carregar o modelo, utilizou-se a biblioteca Tensorflow - que permite o treinamento das mais diversas redes neurais, podendo customizar as camadas envolvidas no processo de treino. Foram utilizadas ainda as bibliotecas Pickle e Random, para, respectivamente, salvar e carregar os dados dos módulos criados e para seleções aleatórias nos conjuntos de dados.

Foi utilizada a base de dados do GenBank - um repositório de dados público, mantido e gerenciando pelo *National Center for Biotechnology Information*, amplamente utilizado pelos cientistas das mais diversas áreas, possuindo informações genéticas de sequências de DNA das mais diversas espécies de seres vivos (desde fungos filamentosos, como os escolhidos para este trabalho, até sequências de *Homos sapiens*). Do GenBank, foram recuperados dados de dois fungos filamentosos: o fungo *Colletotrichum* e do fungo *Diaporthe*. As informações sobre ambos os arquivos são descritas na Tabela 3.1. A escolha dos fungos se deu pela aplicação deles em trabalhos passados, como o de Cruz e Menossi (2021) e Santos (2021), que obtiveram bons resultados com a base de dados utilizando o CRF.

Tabela 3.1 – Dados obtidos do Genbank para construção da base de dados.

Fungo	Proteína	# amostras	Tamanho (MB)
Colletotrichum	Actina	7003	16,5
Diaporthe	Beta Tubulina	3878	10,8
Total		10881	27,3

Para recuperar as sequências do GenBank e montar a base de dados, de forma a se equiparar à utilizada neste trabalho (os dados recuperados no trabalho podem não ser os mais

atuais na plataforma), foram utilizadas as seguintes *strings* de busca:

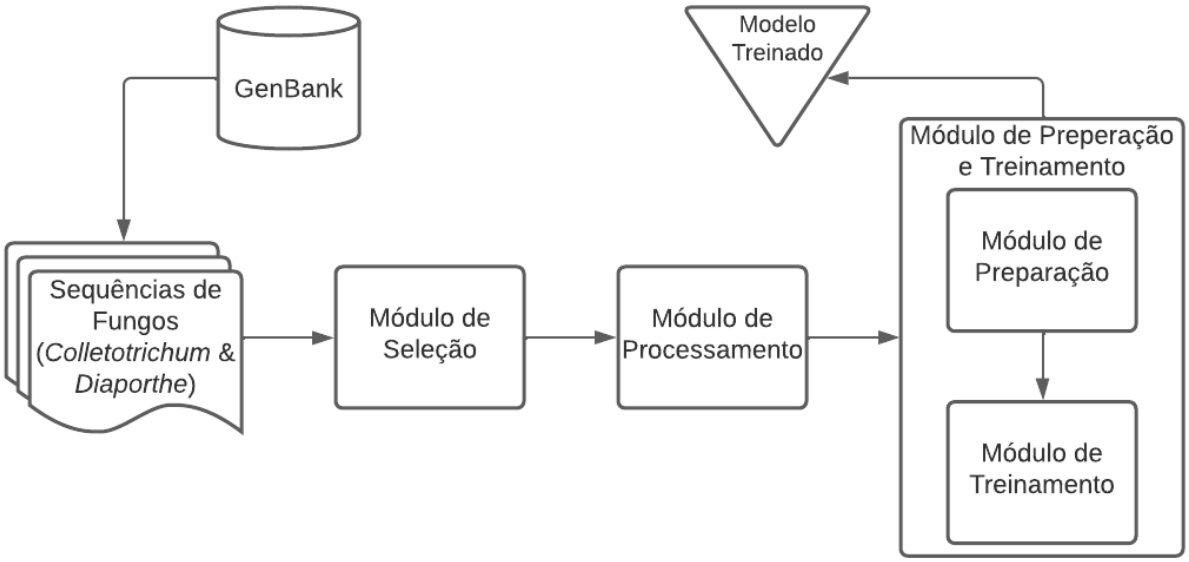
- *Colletotrichum*: *colletotrichum actin partial CDS*;
- *Diaporthe*: *diaporthe beta-tubulin partial CDS*;

As *strings* recuperam todos os registros dos fungos filamentosos *Colletotrichum* e *Diaporthe* do GenBank das proteínas actina e beta-tubulina, respectivamente. Há ainda, a restrição de "*partial CDS*", que filtra por sequências que não necessariamente começam de forma a codar uma proteína (em outras palavras, a sequência pode começar "cortada" no meio de uma trinca e/ou terminar cortando uma proteína).

3.2 Módulos

A construção do código e a estruturação da arquitetura abordada deu-se por meio da criação de módulos de desenvolvimento, como ilustrado na Figura 3.1. Foram criados três módulos principais, além de uma função para a criação de uma métrica de desempenho "real", isso é, considerando a/s resposta/s fornecida/s pela rede como correta/s apenas se ela for idêntica a resposta esperada - maiores detalhes sobre essa métrica são explorados na Seção 3.2.4.

Figura 3.1 – Figura representativa da arquitetura modular criada.



Fonte: Próprio Autor.

Os módulos de Seleção e Processamento são adaptados do trabalho de Cruz e Menossi (2021), que explorou esse tipo de abordagem na estruturação do problema.

3.2.1 Módulo de Seleção

O módulo de seleção é onde se concentra o pré-processamento das sequências "cruas" dos arquivos do GenBank. Cada registro do GenBank conta com inúmeras informações que, para propósitos do trabalho, não são úteis. Então, neste primeiro passo, filtra-se as sequências de forma a manter apenas as informações relevantes, sendo elas: a sequência completa daquele registro, as posições de cada exón daquela sequência, as sequências de éxons, as posições das sequências de íntrons daquela sequência e, por fim, as sequências de íntrons. Por fim, o dado gerado é salvo em um arquivo intermediário.

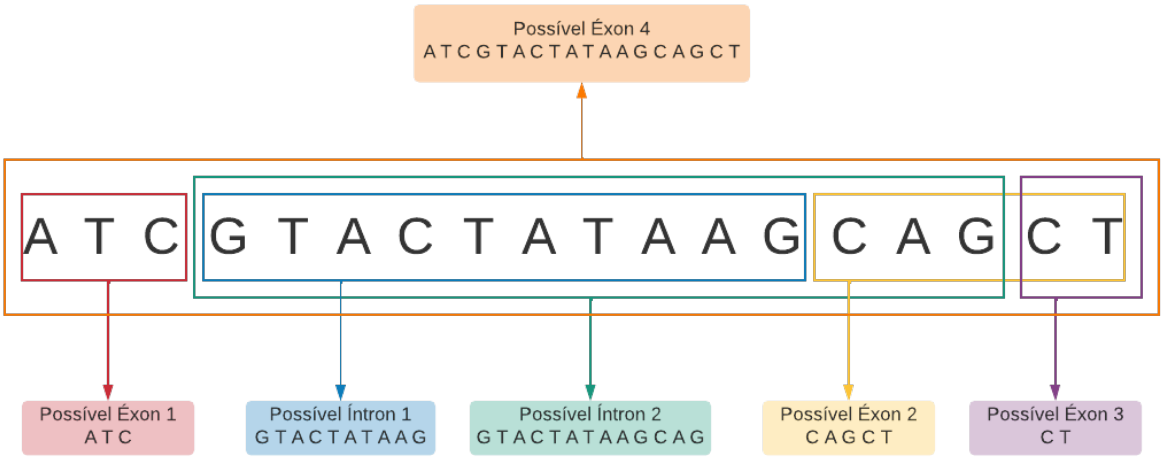
É importante ressaltar ainda que, registros cuja sequência ultrapasse trezentas bases nitrogenadas foram desconsiderados - uma vez que registros grandes acarretarão em um grande processamento nos módulos posteriores, bem como os que possuem bases degeneradas[1].

3.2.2 Módulo de Processamento

O módulo de processamento é o módulo responsável por gerar as possibilidades de sequências de íntrons, éxons ou *neithers* - uma classe que descreve uma mistura de íntrons e éxons.

A entrada deste módulo é o arquivo intermediário gerado no módulo anterior. Para cada sequência que é recebida, são geradas todas as possibilidades de íntrons e éxons daquela sequência, como é descrito na figura 3.2.

Figura 3.2 – Figura ilustrativa da construção das possibilidades de íntrons e éxons.



Fonte: Próprio Autor.

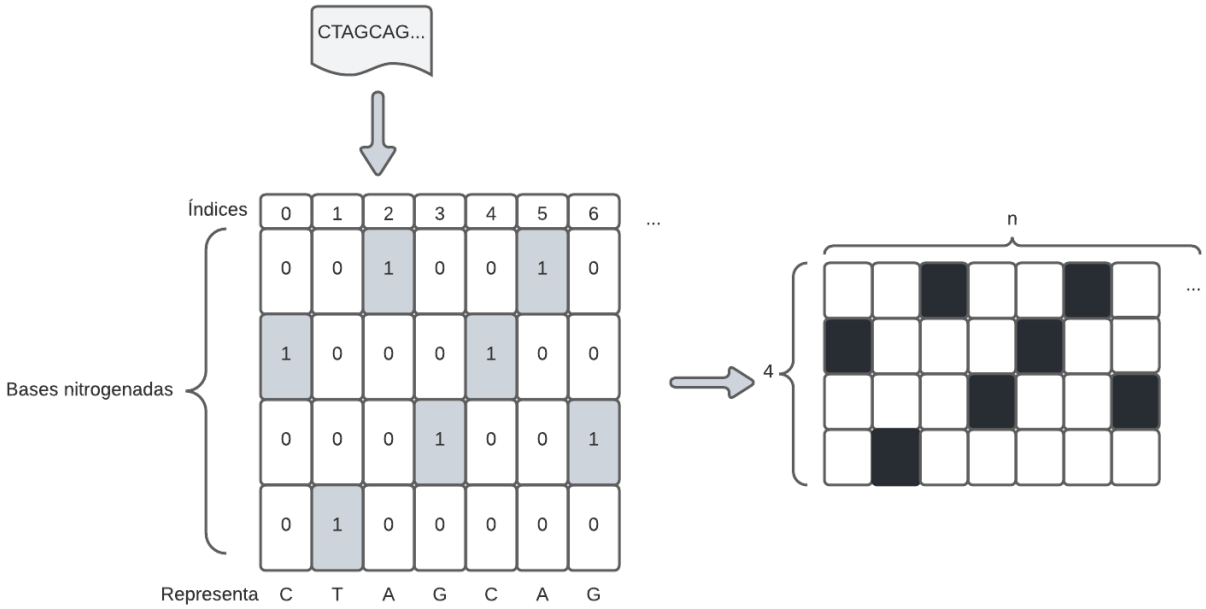
[1] Uma base degenerada é uma falha comum no processo de decodificação da proteína para a sequência, onde uma proteína pode ter sido gerada por múltiplas bases e o pesquisador não foi capaz de determinar qual a correta. Em termos da sequência, uma base degenerada é representada por uma letra que pode corresponder a duas ou mais bases diferentes.

3.2.3 Módulo de Preparação e Treinamento

O módulo de preparação e treinamento é dividido em duas etapas.

Na primeira etapa os dados coletados a partir do módulo anterior são preparados para serem usados pelo modelo, neste caso, a CNN do Tensorflow. O processamento principal acontece em função de que os padrões dos dados recebidos se dá por meio de bases nitrogenadas em meio textual, o que, para a CNN não é uma entrada válida - tanto por não estar em um formato matricial, quanto pela falta de normalização das sequência, pois elas não estão em um tamanho padronizado. Para contornar estes problemas, gera-se uma imagem com quatro "*pixels*" de altura e N "*pixels*" de largura, onde N é o tamanho da sequência obtida, ou a quantidade de bases nitrogenadas da sequência. O processo de conversão se dá como segue: toda base é substituída por uma lista unidimensional homogênea contendo uma escala de cinza (neste caso, 0 indica preto e 1 indica branco) de forma que a Adenina fique com uma lista da forma [1,0,0,0], a Citosina com [0,1,0,0], a Guanina com [0,0,1,0] e a Citosina com [0,0,0,1]. Por fim, todas as listas geradas são juntadas em outra lista, formando uma matriz que representa a "imagem" formada pela sequência. Este processo é ilustrado na Figura 3.3. Além disso, é calculada uma média de tamanho das sequências e, então, as imagens de todas as sequências são redimensionadas de forma a ficarem com a largura do tamanho da média calculada.

Figura 3.3 – Figura ilustrativa do processo de conversão das bases nitrogenadas em meio textual para uma imagem bidimensional (cores invertidas).



Fonte: Próprio Autor.

Após, os *neithers* e éxons são unificados em uma classe chamada de "not-intron" e, então, é realizada a separação entre os registros da base para treino e validação, separando uma taxa de cada uma das classes para validação, definida pela constante *separation_rate*, definida no início

do código. Por fim, as listas são re-arranjadas de forma a deixar todos as classes misturadas ao longo da lista final.

Após toda sequencia passar por esse processo, os dados estão (junto com as suas respectivas classes) prontos para serem enviados para treinamento do modelo.

Na segunda etapa é feito o treinamento propriamente dito. Nessa etapa, define-se as camadas da rede neural, bem como o otimizador e parâmetros para o treinamento, como as épocas e dados de validação.

Para as camadas da rede neural, utilizou-se as seguintes configurações:

- Uma camada de convolução, com o tamanho de entrada igual aos definidos para as imagens geradas, *kernel* de tamanho 2x2 e função de ativação ReLU.
- Uma camada de *Max Pooling*, com tamanho do *pool* de 2x2.
- Uma camada de *Flatten* para realizar a transformação das imagens para um agregado homogêneo unidimensional.
- Uma camada densa, com função de ativação ReLU e 64 neurônios.
- Uma camada densa, com função de ativação ReLU e 32 neurônios.
- Uma camada densa, com função de ativação *Softmax*, contendo dois neurônios, para a separação entre as possíveis classes.

Para o otimizador do modelo foi usado o Adam e para a função de custo a *Categorical Crossentropy*.

A rede conta com apenas uma camada de convolução e *pooling* pois a altura das imagens não permite a inserção de novos valores, a menos que o *kernel* seja retraído, o que acarretaria em uma perda do aprendizado de características da rede. Em seguida, para que seja possível analisar a imagem, e suas características aprendidas pelo modelo, segue uma camada de *flatten*, para que seja possível usar as imagens no estado atual para camadas densas que serão responsáveis por mapear as características nas classes de saída disponíveis.

3.2.4 Métrica de Remontagem das Sequência

Para que fosse possível avaliar o modelo em cenários "reais", foi criada a métrica de remontagem de sequência. A métrica busca avaliar não apenas a identificação das regiões codantes, mas também, da reconstrução de uma sequência de entrada. Primeiramente, a sequência de entrada é desmembrada em possíveis íntrons, os quais serão um a um enviados para o modelo e ele avaliará, devolvendo um resultado negativo ou positivo para a classe "Intron". Quando positivo, essa sequência que delimita o íntron será removida da sequência original (caso ela ainda exista

por completo na sequência e não foi removida por outra instância de íntron encontrado). Ao final do processo, temos uma sequência inteira que sofreu um processo similar ao *splicing*. Então, essa sequência é comparada base à base com a resultante esperada. Se a compatibilidade das sequências for integral, então é computado um acerto. Como resposta final, é calculada a acurácia e o valor é devolvido para o usuário.

4 Resultados

Nesta seção são apresentados e discutidos os resultados alcançados, bem como é feita a comparação entre os resultados obtidos com os de outros trabalhos.

4.1 Resultados Experimentais

O primeiro treinamento realizado foi com um número fixo de 100 épocas para o treinamento e com o tamanho dos dados de entradas para cada passo da época gerenciado automaticamente pela biblioteca.

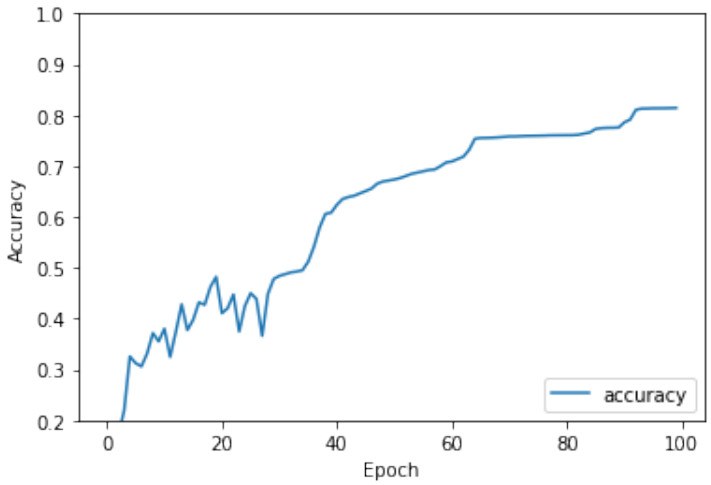
Para o fungo *Colletotrichum*, que contava com 7003 sequências na proteína actina, os resultados obtidos são apresentados na Tabela 4.1:

Tabela 4.1 – Medidas obtidas pelo modelo durante o treinamento de 100 épocas para o *Colletotrichum*.

Rótulo	Quantidade	% exemplos na base	Medida F1
Intron	10384	0.42	0.9982
Not-Intron	14359	0.58	0.9982
Total	24743	1.00	

A acurácia final do modelo para o *Colletotrichum* em 100 épocas de treinamento foi de 0,8122. A curva da acurácia gerada pelo modelo ao longo do treinamento é ilustrada na Figura 4.1

Figura 4.1 – Gráfico ilustrativo da evolução da acurácia do modelo durante o treinamento de 100 épocas para o *Colletotrichum*.



Fonte: Próprio Autor.

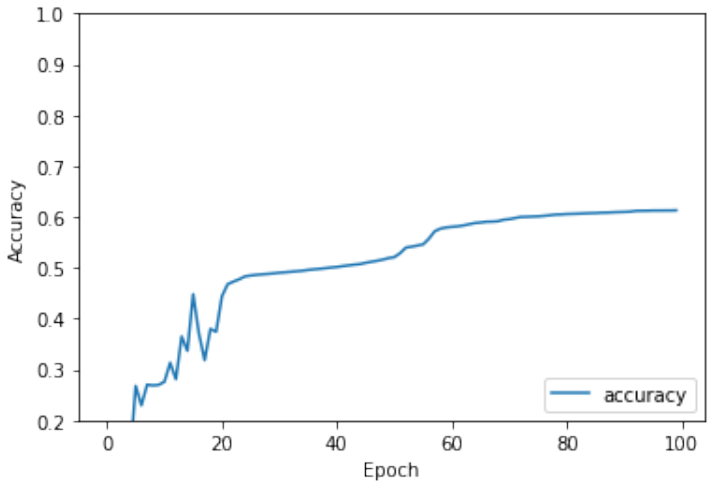
Já para o fungo *Diaporthe*, que contava com 3878 sequências na proteína beta tubulina, os resultados obtidos são apresentados na Tabela 4.2:

Tabela 4.2 – Medidas obtidas pelo modelo durante o treinamento de 100 épocas para o *Diaporthe*.

Rótulo	Quantidade	% exemplos na base	Medida F1
Intron	8199	0.44	0.9976
Not-Intron	10302	0.56	0.9976
Total	18501	1.00	

A acurácia final do modelo para o *Diaporthe* em 100 épocas de treinamento foi de 0.6143. A curva da acurácia gerada pelo modelo ao longo do treinamento é ilustrada na Figura 4.2

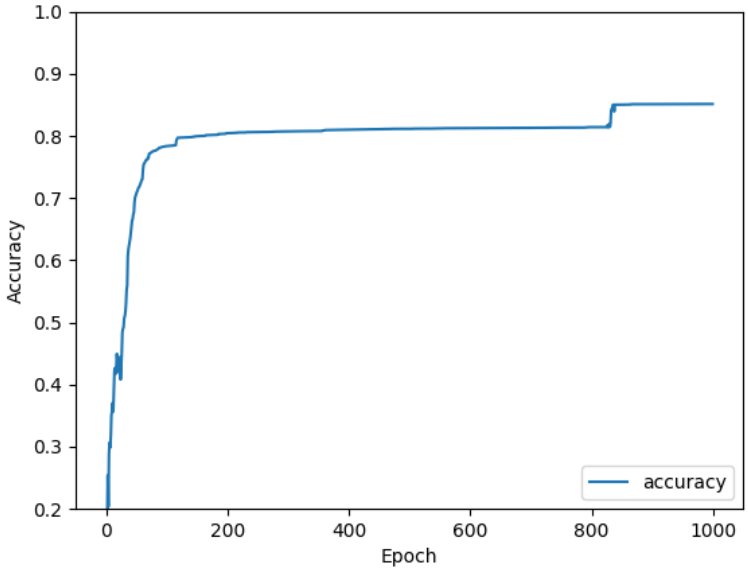
Figura 4.2 – Gráfico ilustrativo da acurácia do modelo durante o treinamento de 100 épocas para o *Diaporthe*.



Fonte: Próprio Autor.

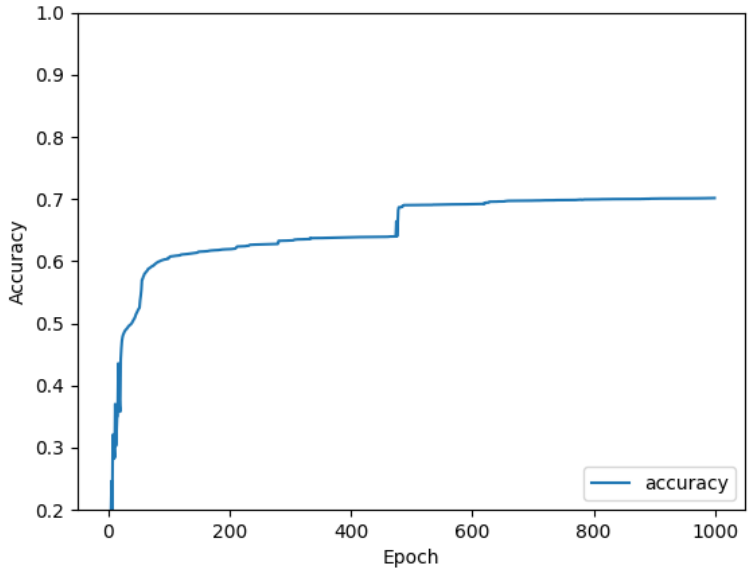
Além disso, foi feito o treinamento para 1000 épocas, para ambos os fungos, o que gerou os resultados mostrados nos gráficos 4.3 e 4.4.

Figura 4.3 – Gráfico ilustrativo da acurácia do modelo durante o treinamento de 1000 épocas para o *Colletotrichum*.



Fonte: Próprio Autor.

Figura 4.4 – Gráfico ilustrativo da acurácia do modelo durante o treinamento de 1000 épocas para o *Diaporthe*.



Fonte: Próprio Autor.

Houve uma melhora significativa na acurácia no modelo para ambas as sequências ao aumentar a quantidade de épocas para 1000, como é evidenciado na Tabela 4.3.

Tabela 4.3 – Comparação da acurácia obtida para 100 e 1000 épocas de treinamento.

Fungo	100 épocas	1000 épocas
Colletotrichum	0.8122	0.8513
Diaporthe	0.6143	0.6987

Por fim, usando a métrica de remontagem das sequências, foram obtidos os seguintes resultados (foram utilizados os modelos com 1000 épocas de treinamento para as predições):

Tabela 4.4 – Medidas obtidas na métrica de remontagem de sequência.

Fungo	Proteína	Acurácia
Colletotrichum	Actina	0.1628
Diaporthe	Beta Tubulina	0.0000

4.2 Discussão dos resultados

Os resultados obtidos para o *Colletotrichum* foram mais promissores que os obtidos para o *Diaporthe*, porém, quando comparados a resultados obtidos de abordagens que usam o CRF

e trabalham com a base de dados parecida, como os resultados de Cruz e Menossi (2021) ou (SANTOS, 2021), os resultados se mostram inferiores, porém, em todos os trabalhos com estes mesmos fungos, as sequências do *Diaporthe* na proteína beta tubulina apresentou padrões mais difíceis de serem aprendidos.

Quanto à métrica de remontagem, os resultados não foram promissores, com precisões muito abaixo do esperado, mesmo para os valores de acurácia obtidos. O comportamento padrão das predições foi de prever os íntrons reais corretamente, porém, íntrons falsos também foram dados como verdadeiros, o que na hora de remontar a sequência final resulta em uma sequência que não está de acordo de maneira integral a esperada, e, portanto, incorreta para a métrica. Este comportamento é evidenciado no resultado para o *Colletotrichum* e pode ser visto com um impacto ainda maior no *Diaporthe* - o que é um resultado esperado, uma vez que as sequências do *Diaporthe* para a proteína da beta tubulina tendem a possuir o dobro do tamanho do *Colletotrichum*, resultando em uma quantidade maior de íntrons para serem localizados com êxito, o que impacta diretamente na probabilidade de acerto.

Um dos possíveis motivos para a acurácia mais baixa do que a gerada por modelos como o CRF, é o fato do redimensionamento das imagens. As sequências consenso podem ser distorcidas ao longo do processo, o que faz com que a detecção de íntrons seja afetada. Se uma imagem é menor que a média, ela vai ser aumentada de forma a perder o padrão das sequências consenso. Por outro lado, se ela é maior que a média, ela deve ser comprimida, o que pode acarretar tanto na perda parcial quanto total de características (uma vez que um *splicing site* pode ser completamente comprimido, dependendo do tamanho da imagem e do redimensionamento sendo aplicado a ela). Outro ponto é que uma imagem que é menor e é redimensionada de forma que fique maior, pode coincidir com uma imagem que já era maior por padrão, causando ainda mais inconsistências.

5 Considerações Finais

Neste capítulo é feita a discussão dos objetivos alcançados, bem como a apresentação da conclusão final com base nos resultados obtidos. Por fim, são comentados possíveis trabalhos futuros.

5.1 Conclusão

Este trabalhou buscou criar um *software* capaz de identificar as regiões codantes em sequência de DNA, com o uso de redes neurais convolucionais.

As bibliotecas Tensorflow e Keras tiveram um grande impacto para o desenvolvimento do trabalho, fornecendo o ferramental básico para que o treinamento fosse realizado.

Com base nos resultados obtidos, podemos concluir que o modelo de CNN é apto a reconhecer padrões de regiões codantes no DNA, porém, com resultados inferiores a de outras abordagens. Quando se trata do *Colletotrichum* na proteína actina, os resultados ainda são promissores e podem servir de embasamento para os profissionais da área identificarem possíveis íntrons, porém, os resultados obtidos para o *Diaporthe* apontam para uma dificuldade maior para a identificação.

Apesar da acurácia acima de 60% para ambos os fungos, a remontagem de sequências que é um processo no qual permite a visualização do problema conforme os especialistas da área mais sofrem atualmente, apresentou resultados não tão promissores. Porém, com a criação da arquitetura da rede em módulos, o algoritmo criado pode ser facilmente adaptado para novas abordagens e/ou redes neurais diferentes.

Outra análise é que os dados da base recuperada só estão em abundância se forem consideradas sequências cortadas, que podem conter íntrons quebrados, prejudicando no padrão. Além disso, o GenBank conta com poucos exemplos do *Diaporthe* para a proteína Actina.

Conclui-se também que, tanto o fungo, quanto a proteína possuem grande impacto nas precisões obtidas, o que pode evidenciar a necessidade de uma base mais diversificada para uma boa generalização de uma proteína inteira ou de um fungo por completo. Uma discussão com profissionais remete à possibilidade do uso do *software* não para identificar exatamente as regiões codantes no DNA, mas para predizer os mais prováveis íntrons em uma sequência - o que já seria de grande auxílio para os profissionais da área.

Uma vez que o trabalho se deu tanto em biologia e biotecnologia como na ciência da computação, uma das dificuldades foi o entendimento dos conceitos necessários.

Outra dificuldade encontrada foi na entrada dos dados para o treinamento do modelo, onde

o processo de conversão para imagens não é simples, por se tratar de uma adaptação de sequências de bases nitrogenadas para imagem. Este é um processo que dependendo da biblioteca utilizada, ou da maneira que é realizada, pode gerar resultados inconclusivos e dificultar o treino. Os objetivos do trabalho, conforme especificados, foram atendidos. Outro aspecto que é importante ressaltar é o tempo que leva para treinamentos com mais épocas, uma vez que quanto mais épocas, mais processamento é necessário. Talvez um treinamento com ainda mais épocas elevasse ainda mais a acurácia.

5.2 Trabalhos Futuros

Como trabalhos futuros, podem ser elencadas algumas possibilidades:

- Implementação utilizando diferentes maneiras de gerar a imagem, como por exemplo, um agregado unidimensional com valores numéricos. Além de abordagens com diferentes combinações de camadas ocultas, otimizadores e funções de ativação.
- Diferentes abordagens para a identificação de regiões codantes em imagens, como a utilização de extratores de textura.
- Treino a partir de uma rede neural recorrente.
- Expandir as análises para mais de uma proteína por vez.
- Expandir as análises para mais fungos.
- Implementação de uma interface amigável para que profissionais da área possam utilizar a ferramenta. Bem como a implementação de um módulo de tradução para proteína que leve em consideração bases degeneradas.

Referências

ACADEMY, D. S. *Deep Learning Book*. London: Bantam, 2022. <https://www.deeplearningbook.com.br>(Acesso em: 16-03-2022).

ALBARADEI, S.; MAGANA-MORA, A.; THAFAR, M.; ULUDAG, M.; BAJIC, V. B.; GOJOBORI, T.; ESSACK, M.; JANKOVIC, B. R. Splice2deep: An ensemble of deep convolutional neural networks for improved splice site prediction in genomic dna. *Gene: X*, v. 5, 2020. ISSN 25901583.

ALBERTS, B.; JOHNSON, A.; LEWIS, J.; MORGAN, D.; RAFF, M.; ROBERTS, K.; WALTER, P.; WILSON, J.; HUNT, T. *Biologia Molecular da Célula*. [S.l.]: Artmed, 2017. v. 6. 1464 p. ISBN 9788582714225.

CHOW, L. T.; ROBERTS, J. M.; LEWIS, J. B.; BROKER, T. R. A map of cytoplasmic rna transcripts from lytic adenovirus type 2, determined by electron microscopy of rna:dna hybrids. *Cell*, v. 11, 1977. ISSN 00928674.

CHOWDHURY, G. G. Natural language processing. *Annual Review of Information Science and Technology*, v. 37, n. 1, p. 51–89, 2003. Disponível em: <https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/aris.1440370103>.

CRUZ, G. H. F.; MENOSSI, V. Um software para identificação de regiões codantes em genes de fungos filamentosos e sua tradução para os polipeptídeos correspondentes. *Universidade Estadual de Maringá*, 2021. <https://drive.google.com/file/d/1l-F1uZYAmYeaCdm0YDLDZz757rEHdk-y/view>(Acesso em: 17-04-2022).

FERREIRA, G. L. F. Identificando regiões não codantes em sequências de dna utilizando técnicas de aprendizado de máquina. *Universidade Estadual de Maringá*, 2019. <https://drive.google.com/file/d/1WPK0pdM5wopBtYJdqn7B3N_LFcSW3L0s/view>(Acesso em: 17-04-2022).

GéRON, A. *Mãos à obra: aprendizado de máquina com Scikit-Learn & TensorFlow*. [S.l.]: Alta Books, 2019. v. 1. 576 p. ISBN 8550803812.

KOHAVI, R.; PROVOST, F. Glossary of terms - journal of machine learning. *Machine Learning*, v. 30, 1998.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, 1998. ISSN 00189219.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, v. 5, 1943. ISSN 00074985.

MCGREGOR, A.; HALL, M.; LORIER, P.; BRUNSKILL, J. Flow clustering using machine learning techniques. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, v. 3015, 2004. ISSN 16113349.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. In: *Sistemas Inteligentes Fundamentos e Aplicações*. 1. ed. Barueri-SP: Manole Ltda, 2003. p. 89–114. ISBN 85-204-168.

NAITO, T. Human splice-site prediction with deep neural networks. In: . [S.l.: s.n.], 2018. v. 25. ISSN 10665277.

PASHAEI, E.; AYDIN, N. Markovian encoding models in human splice site recognition using svm. *Computational Biology and Chemistry*, v. 73, 2018. ISSN 14769271.

SANTOS, A. R. Estudo e aplicação do algoritmo conditional random fields para identificação de íntrons e éxons em sequências de dna. *Universidade Estadual de Maringá*, 2021. <https://drive.google.com/file/d/1khAVgn6S_0qA9GV6E7Xb3NwVkK8G445e/view>(Acesso em: 17-04-2022).

SHARP, P. A. The discovery of split genes and rna splicing. In: . [S.l.: s.n.], 2005. v. 30. ISSN 09680004.

SONNENBURG, S.; RäTSCH, G.; JAGOTA, A.; MüLLER, K. R. New methods for splice site recognition. In: . [S.l.: s.n.], 2002. v. 2415 LNCS. ISSN 16113349.

SONNENBURG, S.; SCHWEIKERT, G.; PHILIPS, P.; BEHR, J.; RäTSCH, G. Accurate splice site prediction using support vector machines. In: . [S.l.: s.n.], 2007. v. 8. ISSN 14712105.

STUBBLEFIELD, W. A.; LUGER, G. F. *An introduction to Machine Learning*. [S.l.: s.n.], 1992. 68 p.

WATSON, J. D.; BAKER, T. A.; BELL, S. P.; GANN, A.; LEVINE, M.; LOSICK, R. *Biologia Molecular do Gene*. Porto Alegre: Artmed, 2015. v. 7. 912 p. ISBN 9788582712085.