Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

EnAVPClass: um Classificador Multifase de PAVs, Baseado em Aprendizado Profundo de Combinação

Caio Fontes de Castro

Monografia Final

mac 499 — Trabalho de Formatura Supervisionado

Supervisor: Dr. Milton Yutaka Nishiyama Jr.

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro do CNPq, através do programa PIBITI

São Paulo
2022

Agradecimentos

There is no value in anything until it is finished.
— Gengis Khan

Primeiramente quero agradecer aos meus pais, Róbison e Ana, sem os quais eu não teria nada. Os dois me permitiram a oportunidade de estudar fora da minha cidade natal, e sempre me apoiaram nessa jornada.

Agradeço ao meu orientador Dr. Milton Yutaka Jr. pela supervisão durante o desenvolvimento desse projeto. Também agradeço a Me. Fernanda Midori pelo auxílio direto na elaboração e na implementação de partes desse trabalho, pricipalmente na segunda fase do preditor.

Por fim agradeço aos meus colegas de curso, Caio Andrade, Leandro Rodrigues, Rogério Fernandes, Eduardo Laurentino e Artur Magalhães, dentre vários outros, pela companhia ao longo desses anos, pelos conselhos, pelas conversas e por compartilharem tantas experiências comigo.

Resumo

Caio Fontes de Castro. **EnAVPClass: um Classificador Multifase de PAVs, Baseado em Aprendizado Profundo de Combinação**. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2022.

Peptídeos Bioativos tem sido alvo de crescente interesse devido à sua presença em virtualmente todos os seres vivos e seu potencial para desenvolvimento de novos fármacos com uma probabilidade menor de desenvolvimento de resistência. Em particular peptídeos com ação antiviral tem sido estudados como alternativas as drogas antivirais tradicionais, que muitas vezes apresentam efeitos adversos severos aos pacientes.

No entanto, a grande variedade desses compostos também se revela um desafio para os pesquisadores, visto que a síntese e testagem de um único peptídeo pode apresentar um custo proibitivo, com alta probabilidade de que o peptídeo testado não apresente as características desejáveis.

Métodos computacionais tem sido cada vez mais utilizados para a identificação de novos peptídeos quanto a sua bioatividade e tipo funcional. Especialmente modelos baseados em Redes Neurais tem obtido resultados cada vez melhores.

Neste projeto, propomos o desenvolvimento de um classificador composto de duas fases. A primeira fase é baseada em um modelo composto de uma Rede Neural Recorrente Profunda e um modelo de Floresta Aleatória, capaz de predizer a ação antiviral de peptídeos desconhecidos, enquanto que a segunda fase é composta de um modelo de Máquina de Vetor Suporte, que é capaz de identifcar o tipo de atividade antiviral desses peptídeos.

O objetivo desse projeto foi identificar peptídeos nos venenos de aranhas, de escorpiões e na saliva de carrapatos com atividade antiviral, por meio de análise do transcriptoma por RNA-seq de aranhas, escorpiões e carrapatos, aplicando uma nova metodologia de predição de PAVs baseada em *Deep Learning*. Por meio desse projeto esperamos contribuir com descobertas em áreas como toxinologia/venômicas, bioinformática e trancriptômica de aracnídeos, auxiliando no desenvolvimento de novas ferramentas farmacológicas e terapêuticas para o combate a doenças virais.

Palavras-chave: Peptídeos Bioativos. Transcriptômica. Bioinformática. Aprendizado Profundo.

Abstract

Caio Fontes de Castro. **EnAVPClass: an Ensemble Two-Phase Deep Learning AVP Classifier**. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2022.

Bioactive Peptides have been the targets of growing interest due to their presence in virtually all living beings, and potential for the development of new drugs with a lower probability of resistance developing. In particular peptides with Antiviral activity have been studied as alternatives to traditional antiviral drugs, which commonly have severe adverse side effects.

However, the imense diversity of these compounds also reveals itself to be a challenge for researchers, given that the synthesis and testing of a single peptide can be prohibitively expensive, with a high chance that the tested peptide will not present the desired characteristics.

Computational methods have been used more and more in the identification of bioactive peptides of many functional types. Models based in Neural Networks specially, have achieved better and most proeminent results.

In this project we propose the development of a two phase classifier, in which the first phase is based on an Ensemble model of a Deep Recurrent Neural Network and a Random Forest module, capable of predicting the antiviral activity of unknown peptides, and the second phase utilizes a Support Vector Machine to classify the type of antiviral activity of these peptides.

The objective of this project is to identify peptides in the venom of spiders, scorpions and ticks saliva with antiviral activity, through the analysis of their Transcriptomes by RNA-seq, applying a new metodology based on Deep Learning for the prediction of AVPs. Through the findings of this project we expect to contribute to many areas, such as toxinology/venomics, bioinformatics and transcriptomics of arachnids, aiding in the development of new farmacologic tools and terapies for viral diseases.

Keywords: Bioactive Peptides. Transcriptomic. Bioinformatics. Deep Learning.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Venenos de Animais Peçonhentos tem se mostrado verdadeiros repositórios para descoberta de novos compostos farmacêuticos, devido a sua abundância e variedade de alvos e mecanismos de ação. Um componente importante dessas substâncias são os Peptídeos Bioativos, devido às suas ações de amplo espectro e possibilidade de associação a diversos tipos funcionais.

Os Peptídeos Antivirais em particular tem ganhado grande destaque e interesse, devido à pandemia por SARS-COV2 e a possibilidade de desenvolvimento de compostos alternativos sem os efeitos adversos extremamente severos das drogas antivirais tradicionais, a um custo final de tempo de prospecção para a indústria farmacêutica significativamente menor. No entanto a caracterização de peptídeos quanto a sua bioatividade, tipo funcional e possíveis alvos ainda é um processo muito custoso e trabalhoso, que requer diversas etapas de pesquisa, experimentação e validação.

Métodos computacionais baseados em dados biológicos tem se tornado cada vez mais comuns, e principalmente os métodos baseados em arquiteturas de *Deep e Machine Learning* os quais tem obtido resultados de estado da arte na resolução de diversos problemas de interesse em saúde pública. Esse métodos possibilitam o *High-Throughput Screening* de diversos compostos mas ainda não foram aplicados extensivamente ao problema da predição de peptídeos bioativos.

1.1 Animais Peçonhentos e seus Venenos

Existem mais de 100.000 espécies de organismos venenosos distribuídas em vários filos como os cordados (répteis, peixes, anfíbios e mamíferos), equinodermas (estrelas do mar, ouriços), moluscos (polvos, caracóis), anelídeos (sanguessugas), nemertea, cnidários (anêmonas, corais) e artrópodes (insetos, aracnídeos). O uso de venenos é uma das técnicas mais fascinantes usadas para sobrevivência e captura de presas na natureza. A produção de veneno representa uma vantagem adaptativa e um exemplo de convergência evolutiva (Fry, Scheib *et al.*, 2008).

Venenos são misturas heterogêneas de diversas moléculas, como sais inorgânicos, pequenas moléculas orgânicas, enzimas e peptídeos (Calvete *et al.*, 2009). São produzidos

em uma glândula especializada e quando são inoculados em outro organismo, interferem nos processos bioquímicos e fisiológicos normais que alteram os sistemas nervoso central e periférico, neuromuscular e cardiovascular e ainda, a coagulação sanguínea e a homeostase geral (Fry, Roelants *et al.*, 2009).

Os aracnídeos são a maior classe entre os artrópodes, constituindo mais de 6.000 espécies entre aranhas, escorpiões, caranguejos e carrapatos. Os venenos de artrópodes mais estudados são os de aracnídeos. São mais de 2.000 espécies de escorpiões, divididas em 7 famílias: Scorpionidae, Diplocentridae, Chactidae, Vaejovidae, Bothriuridae, Chaerilidae e Buthidae. Aranhas formam uma classe com mais de 48.000 espécies conhecidas (Platnick e Dupérré, 2011). Existem aproximadamente 899 espécies de carrapatos divididos em 3 famílias: Ixodidae, Argasidae e Nuttalliedae. Atualmente o NCBI possui disponível 148 transcriptomas por sequenciamento de nova geração de aracnídeos, sendo 88 de carrapatos e ácaros, 41 de aranhas, 13 de escorpiões.

1.2 Peptídeos Bioativos

Um dos principais componentes dos venenos além das proteínas são os peptídeos. Os peptídeos de veneno são muito específicos e potentes, devido a milhões de anos de pressão seletiva. São bastante estáveis e resistentes à degradação por proteases (Fry, Roelants *et al.*, 2009) e produzidos por virtualmente todos os seres vivos como parte do sistema inato imune de defesa (Zasloff, 2002). Além disso, possuem tamanho menor do que anticorpos, o que facilita a penetração nos tecidos e suas ligações com alvos específicos (Mahadevappa *et al.*, 2017). Devido a essas características, os peptídeos de venenos são cada vez mais utilizados como ferramentas farmacológicas e como protótipos para o desenvolvimento e síntese de novas drogas (Calvete *et al.*, 2009), devido ao seu escopo amplo de ação e a probabilidade reduzida de criação de resistência bacteriana (Fernebro, 2011). Esses peptídeos possuem um amplo espectro de propriedades como a anti-inflamatória e antimicrobiana (Moreno, 2015).

O uso exagerado de antibióticos, aliado ao ritmo desacelerado de desenvolvimento de novos medicamentos, levou ao desenvolvimento de resistência por parte dos patógenos a vários dos antibióticos químicos conhecidos. Essa situação ameaça as conquistas que a invenção dos antibióticos trouxeram para a medicina (Ventola, 2015). Essa crise tem motivado a busca por novos compostos com potencial ação antimicrobiana.

Em especial podemos ressaltar os Peptídeos Antimicrobianos (PAMs ou AMPs) (Fuente-Núñez *et al.*, 2017). A palavra 'antimicrobiano' abrange atividades antibacteriana, antiviral, antifúngica e antiparasitária entre outras. Um mesmo peptídeo bioativo pode ter mais de uma atividade confirmando seu grande valor como objeto de estudo científico e farmacêutico.

Os PAMs também possuem diversos mecanismos de ação e interação dentro de um organismo, além de apresentarem baixa toxicidade, com tamanho em geral entre 10 e 50 aminoácidos, mas podendo apresentar tamanhos menores ou maiores, porém com maior probabilidade de causar maior toxicidade ou baixa eficiência, e praticamente não apresentam homologia entre si (Jenssen *et al.*, 2006; Bahar e Ren, 2013).

Dentre os diversos tipos de PAMs, a alta taxa de mutação dos Vírus e as estratégias extremamente diversas que eles utilizam para auxiliar a infecção de células alvo tornam os Peptídeos com Ação Antiviral (PAVs ou AVPs) alvos frequentes de estudo na busca por novas famílias de drogas antivirais (Mata *et al.*, 2017).

A necessidade dessas novas drogas antivirais também se faz clara com a pandemia causada atualmente pelo vírus do SARS-COV2 (Hsu *et al.*, 2020), a qual exemplifica a rapidez com que esses organismos podem sofrer adaptações com um impacto profundo na sua taxa de infecção, causando sequelas na população e uma maior probabilidade de aumento na mortalidade.

1.3 *Deep Learning*

A área de *Deep Learning* (DL) ou Aprendizagem Profunda, está na intersecção entre a Estatística e a Ciência da Computação, e pode ser vista como uma extensão da área do *Machine Learning* (ML) ou Aprendizado de Máquina clássico. As duas áreas possuem um enfoque em modelos estatísticos capaz de reconhecer padrões em conjuntos de dados que podem não ser óbvios a primeira vista. O processo pelo qual isso é atingido é genericamente chamado de treinamento, e é normalmente apresentado como o modelo "aprendendo"algo sobre os dados.

A principal diferença entre as áreas é a quantidade de dados e a interpretabilidade dos modelos estudados, modelos de DL são normalmente menos interpretáveis e requerem uma quantidade maior de dados para seu treinamento, além de serem computacionalmente mais intensivos tanto em quantidade de recursos quanto em complexidade dos seus algoritmos de avaliação e treinamento.

Recentemente, modelos computacionais baseados em DL tem proporcionado resultados superiores a outras abordagens em diversas áreas, sendo uma das técnicas mais estudadas e empregadas em diversos contextos, principalmente na área de saúde. A maior disponibilidade de dados de larga escala sobre vários problemas biológicos, em especial na área de biotecnologia, tem tornado esses sistemas cada vez mais comuns nesse tipo de estudo (Min *et al.*, 2017), proporcionando o surgimento de novas metodologias e otimizações ampliando o conhecimento na área.

Esses modelos normalmente apresentam alguma variação sobre o modelo tradicional de uma Rede Neural Artificial (RNA), baseada em pequenas unidade lógicas chamadas de neurônios, que são ligados uns aos outros em camadas que lembram a estrutura dos neurônios no cérebro humano, de onde vem seu nome.

O grande benefício desse tipo de modelo é a capacidade de representar qualquer função, incluindo funções não-lineares (Goodfellow, Bengio *et al.*, 2016), em um espaço multidimensional arbitrário, porém ao contrário de modelos tradicionais de *Machine Learning* como Florestas Aleatórias (Belgiu e Drăguţ, 2016) e Máquinas de Vetor Suporte (Cortes e Vapnik, 1995), as *features* ou "características"dos dados não precisam ser explicitamente codificadas pelo arquiteto do modelo, pois são inferidas pela própria Rede durante o processo de treinamento. Isso confere as RNAs ampla aplicabilidade em diversas áreas do conhecimento.

Entre as variações mais comuns desse tipo de modelo podemos citar as Redes Neurais Convolucionais (W. Zhang *et al.*, 1990), que se mostraram muito efetivas no tratamento de dados multidimensionais, como imagens, devido a centralidade da operação de convolução em sua concepção, e as Redes Adversariais (Goodfellow, Pouget-Abadie *et al.*, 2014), que consistem em dois modelos que "competem"entre si no treinamento, um é desenhado para produzir exemplos que se assemelhem aos presentes nos dados reais, e o seu "adversário"é desenhado para diferenciar os exemplos produzidos pela outra rede dos exemplos reais.

Em especial, modelos baseados em Redes Neurais Recorrentes (RNR) (Goller e Kuchler, 1996), com sua capacidade de lidar com dados sequenciais de tamanho variável, tem sido muito utilizados em problemas como processamento de linguagem (Kiros *et al.*, 2015; Li *et al.*, 2015) e tradução automática (Luong *et al.*, 2015; Cho, Van Merriënboer, Gulcehre *et al.*, 2014). Uma das maiores vantagens dessa variação é a sua capacidade de integrar dados provenientes de diferentes -ômicas, tecidos e espécies, graças a capacidade de redes de aprendizagem profunda conseguirem identificar relações não-lineares nos dados e possuir robustez na validação do modelo gerado (Kim e Tagkopoulos, 2018).

Em particular RNRs que se utilizam de unidades lógicas mais complexas que um *perceptron* tradicional, como *Long Short-Term Memory* (LSTM) (Hochreiter e Schmidhuber, 1997) ou *Gated Recurrent Units* (GRU) (Cho, Van Merriënboer, Bahdanau *et al.*, 2014), que possuem uma "memória"interna associada à cada estado da rede, e tem se mostrado capazes de armazenar e manter informação sobre cada estado a várias passagens anteriores, possibilitando a identificação de padrões mais e mais complexos.

LSTM-RNRs tem se mostrado efetivas em problemas biológicos, com aplicação na predição da posição subcelular de proteínas (Sønderby *et al.*, 2015) , design *de novo* de PAMs (Nagarajan *et al.*, 2018; Müller *et al.*, 2018), e classificação de peptídeos quanto a sua bioatividade (Veltri *et al.*, 2018). Exploramos mais esse tipo de modelo na seção 1.3.1.

1.3.1 Redes Neurais Recorrentes

As arquiteturas exploradas foram variações do modelo de RNR, muito utilizado em problemas com uma entrada sequencial, que pode ser vista como uma série de passos ao longo do tempo. Esse modelo se adapta facilmente ao nosso problema, onde as proteínas são representadas como sequências de caracteres, principalmente pela possibilidade de processamento de sequências de tamanho variável, e da retenção de informação de diversos passos anteriores na sequência.

Nesse tipo de modelo, a cada passo, a rede produz uma saída y_t e um novo "estado interno"h_t, utilizando-se do estado anterior h_{t-1} e da entrada no momento t, chamada de x_t . É comum que a cada passo utilize-se a mesma matriz de pesos, que é treinada ao final se utilizando de todos os y_i.

A matriz de pesos, sendo a mesma em todos os passos, deve ser capaz de codificar e armazenar a informação sobre padrões ao longo de diversos passos da entrada, tornando a rede capaz de "aprender"padrões que podem ser vistos em qualquer posição. Numa rede neural tradicional por exemplo, um aminoácido, por exemplo uma Leucina, na primeira

posição da sequência seria interpretado de maneira diferente de uma mesma Leucina sendo vista na décima posição, pois suas representações seriam multiplicados por vetores de pesos distintos.

Usar a mesma matriz de pesos em todos os passos também permite a rede lidar com sequências de qualquer tamanho, o que em outras arquiteturas é impossível, forçando que se pré-processe as sequências de forma a padronizá-las. Normalmente essa padronização envolve definir um tamanho limite e colocar um caractere especial "nulo"ao final das sequências menores que esse limite.

Uma modificação comum em RNRs, assim como em outros modelos de DL, é a repetição ou adição de várias camadas colocadas uma "em cima"das outras, onde os estados internos são utilizados como entradas para outras RNRs. Essa modificação tem o propósito de aumentar a capacidade da rede de aprender representações internas ainda mais complexas dos dados de entrada, possibilitando várias camadas de abstração. Essa modificação pode ser vista na Figura 1.1.

Outra modificação presente na Figura 1.1 é a utilização de RNRs "bidirecionais", que consistem basicamente em duas RNRs que operam em direções opostas da sequência, muitas vezes chamadas de "rede para frente"ou *Forward* e "rede para trás"ou *Backward*. Em dados em que a direção da sequência não é importante, ou se conhece a sequência inteira da entrada, essa variação é comum, pois permite que a rede incorpore informações dos dois "lados"ao processar um passo específico. Para o nosso problema, onde as sequências representam proteínas que existem de maneira completa em um espaço tridimensional, a restrição de direção na sequência de aminoácidos pode acabar mascarando padrões que seriam descobertos sem esse impedimento.

No entanto, apesar da possibilidade teórica de RNRs incorporarem padrões entre passos da entrada distantes temporalmente, algumas limitações práticas (Hochreiter, 1998) fazem com que isso não se concretize com o modelo linear tradicional. Isso ocorre pois a mesma matriz de pesos tem que armazenar informações sobre quais partes da entrada ignorar, e quais manter relevantes para os próximos passos.

Sendo assim foram propostos modelos mais refinados, como as Redes de *Long-Short Term Memory* (LSTM) (Hochreiter e Schmidhuber, 1997) e *Gated Recurrent Units* (GRU) (Cho, Van Merriënboer, Bahdanau *et al.*, 2014), com fluxos informacionais separados para o esquecimento e atualização do estado interno da rede, os quais se mostraram capazes de concretizar esse reconhecimento de padrões temporalmente distantes. Essas redes funcionam de maneira similar as RNRs clássicas, apenas modificando o processo específico feito a cada passo.

Temos que o resultado dessas RNRs pode ser interpretado de várias formas em problemas de classificação, visto que como efeito colateral da sua adaptabilidade ao tamanho da sequência, elas acabam produzindo um "resultado"a cada passo. Esse resultado é na verdade um vetor com diversos valores, que precisa ser interpretado.

Utilizamos uma camada "Linear"(uma matriz de pesos que é multiplicada pela entrada), que recebia esse vetor y_T após o processamento do último aminoácido, e o convertia em um vetor com 2 valores, de acordo com o número de classes (positiva e negativa). Em redes bidirecionais, concatenamos os resultados do último aminoácido da rede para frente, com

o resultado do primeiro aminoácido da rede para trás, e enviamos esse vetor concatenado para a camada linear.

Não podemos no entanto, interpretar esses 2 valores produzidos pela camada Linear como a "confiança"da rede de que o peptídeo pertence a cada classe, mas ao aplicar a função de normalização *Softmax* convertemos esse resultado em uma distribuição de probabilidade sobre as classes. A classe com a maior probabilidade atribuída pelo modelo é interpretada como a predição da rede para aquele peptídeo. Esse processo também pode ser observado na Figura 1.1.

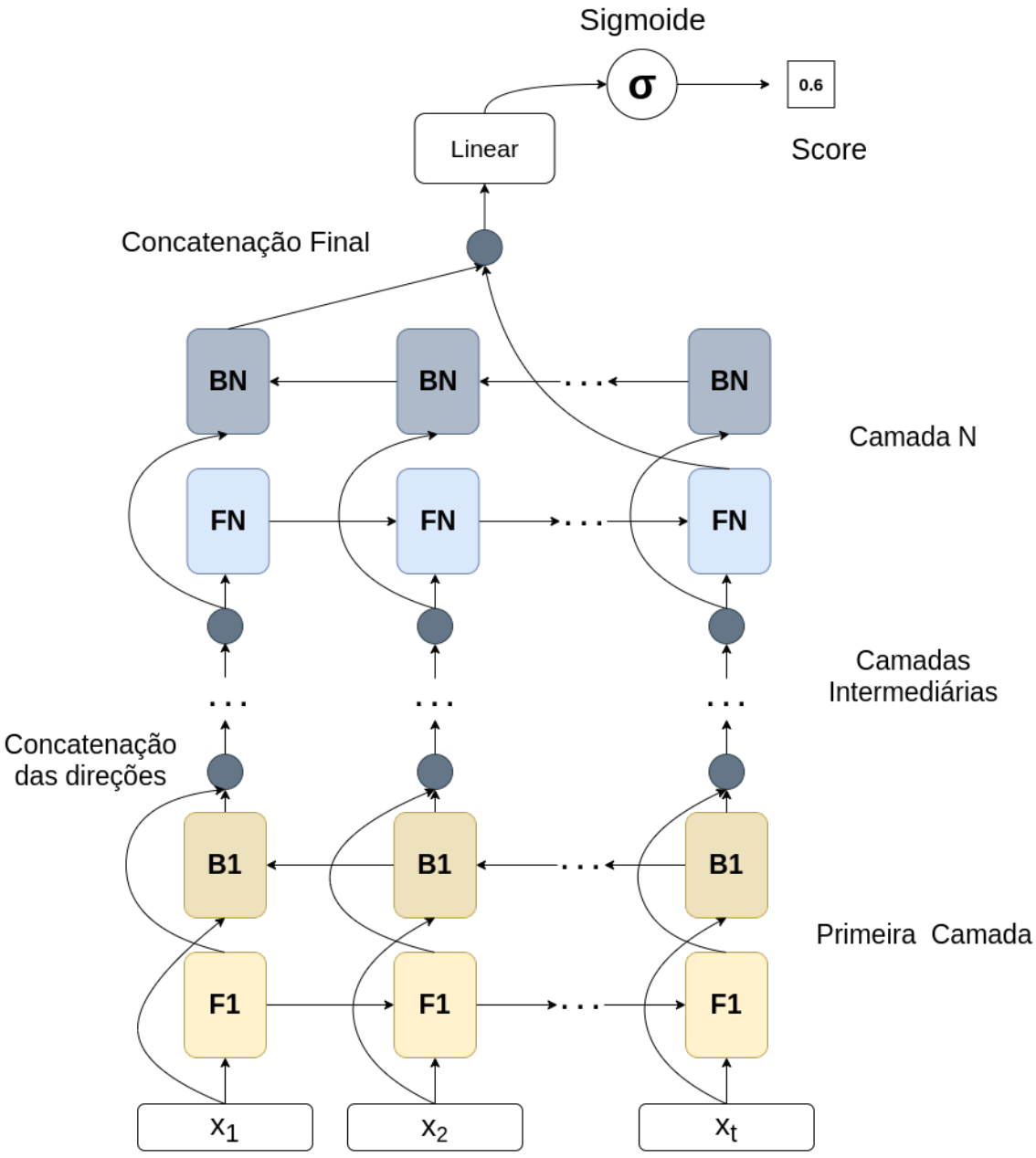

Figura 1.1: *Diagrama representando uma RNR Bidirecional com duas camadas, com uma finalização com camada Linear e função Sigmoide.*

1.3.2 Treinamento

De forma a estimar a performance do modelo em entradas não pertencentes ao conjunto de dados, realizamos a separação do banco de dados aleatoriamente em partições de Treinamento, Teste e Validação, contendo respectivamente 70%, 15% e 15% das sequências do conjunto de dados, mantendo o mesmo balanceamento entre as classes do banco original.

Primeiramente utilizamos as partições de Treinamento e Teste para configurar os híperparametros do modelo (número de *neurons* em cada camada, taxa de *dropout*, etc. Fazemos isso selecionando os parâmetros, treinando o modelo na partição de Treinamento e avaliando o modelo na partição de Teste. O melhor conjunto de hiperparâmetros é então selecionado.

O modelo utilizando os hiperparâmetros encontrados então é treinado nas partições de Treinamento e Teste, e depois seu resultado é calculado com suas predições na partição de Validação. Esse método torna simples avaliar problemas de *overfitting* com a arquitetura do modelo, visto que o modelo comumente tem um resultado melhor nos dados em que foi treinado, se a performance nos dados de Validação for muito baixa é indicativo de que o modelo ou o conjunto de treinamento possui algum problema.

Outro conceito relevante dentro do treinamento são as *epochs*. Uma *epoch* é definida como o processo de realizar o treinamento com todas as sequências do conjunto fornecido, e é comum repetir essa etapa algumas vezes, para se obter um ajuste mais refinado dos pesos internos do modelo. A ideia é que para diversos exemplos apenas uma etapa do ciclo avaliação/perda/correção do processo de treinamento não irá ter um impacto grande o suficiente nos pesos do modelo, mas penalizando o mesmo erro mais de uma vez eles poderão ser ajustados de maneira adequada.

1.3.3 Aprendizado com Combinação

Aprendizado com *Ensemble*, ou com "Combinação", são uma classe de estratégias em problemas de Aprendizado de Máquina na qual múltiplos modelos "base"são utilizados de maneira conjunta para resolver uma tarefa de aprendizado, seja supervisionado ou não-supervisionado (Dietterich, 2000).

Até recentemente, técnicas de combinação e de aprendizado profundo tem sido tratadas como metodologias independentes em aplicações de bioinformática. No entanto, a sinergia entre essas técnicas tem ocasionado o surgimento de diversos métodos que se utilizam dessas duas ferramentas (Cao *et al.*, 2020).

Embora se saiba que os modelos de aprendizado profundo por si só tem a capacidade de atingir uma acurácia preditiva superior a especialistas humanos, esses modelos também possuem limitações. Entre elas podemos citar a alta variabilidade de resultados e a possibilidade de se atingir um minímo local durante o treinamento. No entanto, resultados empíricos nos mostram que a junção desses modelos com essas técnicas de combinação podem levar a modelos com melhor generalização (Ju *et al.*, 2018).

1.3.4 Optimizadores

No treinamento de modelos de Redes Neurais, é comum a utilização de optimizadores, algoritmos mais sofisticados que informam ao modelo como atualizar seus pesos internos com base nos exemplos de treinamento que acabaram de ser avaliados. O mais comum, e mais simples, é o algoritmo de Descida do Gradiente Estocástica (SGD) (Bottou, 2010), onde tentamos minimizar a função de perda utilizada no treinamento através de uma estimação do seu gradiente.

Outros optimizadores mais complexos foram criados, e tem auxiliado na obtenção de melhores resultados com um tempo de treinamento mais curto. Entre eles, cabe destacar o ADAM (Z. Zhang, 2018), e o RMSprop (Sutskever *et al.*, 2013), dois métodos adaptativos, onde a taxa de aprendizagem é diferente para cada parâmetro, e vai variando durante o treinamento.

Capítulo 2

Dados Utilizados

2.1 Obtenção das Sequências de Peptídeos

Devido a sua grande importância biológica e potencial farmacológico, diversos PAMs foram descobertos ou sintetizados e experimentalmente testados ao longo dos anos. Com o advento da bioinformática, foram criados grandes repositórios de dados com a sequência de aminoácidos e diversas outras informações sobre esses compostos.

Essa seção descreve a obtenção dos dados de diversas bases e a consolidação em um banco de PAMs e PAVs experimentalmente validados. A Figura 2.3 descreve o processo resumidamente.

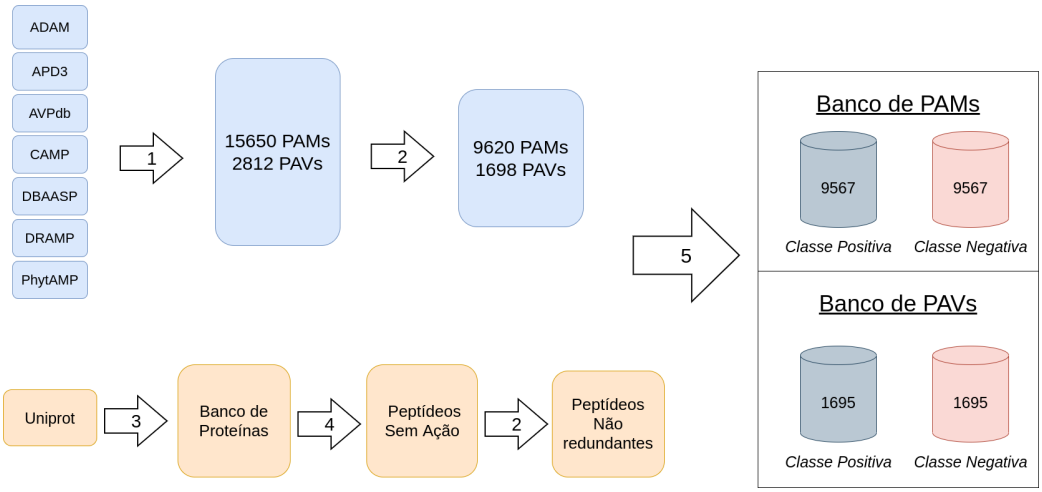

Figura 2.1: *Etapas do Processamento de Dados. 1: Coleta inicial de PAMs de repositórios abertos. 2: Busca por homologia entre as sequências de aminoácidos com a ferramenta cd-hit. 3: Coleta inicial de proteínas do UniprotSwissprot. 4: Digestão in silico das proteínas com a ferramenta pepdigest. 5 : Construção dos bancos de dados para o modelo, escolha dos peptídeos sem atividade ou negativos.*

2.1.1 Bancos Utilizados

Essa seção descreverá em detalhes sobre como as sequências de aminoácidos dos peptídeos foram extraídas de cada repositório utilizado. Os bancos de PAMs e PAVs utilizados para esse estudo estão listados nas seções abaixo.

ADAM

O banco ADAM: *A Database of Antimicrobial peptides* (Lee *et al.*, 2015) é um repositório de sequências de PAMs com diversas atividades, e é focado em fornecer informações e correlações estruturais entre os peptídeos disponiblizados.

Existe um link no site para se fazer o download das sequências de peptídeos presentes no banco em formato ".tsv", com outros campos indicando atividade Anti-bacteriana (além de colunas especificando bactérias Gram+ e Gram- quando essa informação é conhecida), Antifúngica, Antiviral e Antiparasítica.

APD3

O banco *APD3 : The Antimicrobial Peptide Database* (G. Wang *et al.*, 2016), reúne sequências de PAMs da literatura de vários tipos de atividade, todos experimentalmente validados. No site do banco é fornecido um arquivo ".fasta"com as sequências e algumas informações adicionais de todos os peptídeos, porém nesse arquivo não existem indicações do tipo de atividade. Cada peptídeo individual possui uma página própria no site, que é a fonte mais completa de informações. Criamos um *script* que fez o download de todas as páginas de peptídeos individuais utilizando a ferramenta *wget*, seguido pela extração das informações relevantes desse arquivo ".html", utilizando a biblioteca em python *BeatifullSoup*. Ao final foram mantidos apenas os peptídeos extraídos pelo *script* que também estavam presentes no arquivo fasta.

AVPdb

O banco *AVPdb: A Database of Antiviral Peptides* (Qureshi *et al.*, 2014) contém milhares de sequências de PAVs e seus organismos alvo, todos experimentalmente validados. São disponibilizados também peptídeos antivirais modificados com grupos adicionais conforme explicado no artigo, mas para nosso banco fizemos o download apenas do arquivo ".tsv"sob o título "*Natural Peptides*", que contém os PAVs validados.

CAMP

CAMP: Collection of Antimicrobial Peptides (Waghu *et al.*, 2016) é um banco de dados "criado para expandir e acelerar os estudos sobre PAMs baseados em suas famílias"(tradução livre do site oficial), e como tal possui informações detalhadas sobre a origem dos peptídeos que contém. Estão presentes tanto peptídeos validados experimentalmente, obtidos da literatura e de patentes de conhecimento público, e peptídeos preditos ou desenhados pelas ferramentas computacionais disponibilizadas.

Existe uma ferramenta de busca com diversos filtros, mas devido ao limite de sequências que podem ser baixadas de cada vez (25 por arquivo), uma estratégia semelhante a da

extração do banco *ADAM* precisou ser adotada. Foi feito o download das páginas web de todos os peptídeos da base, e mantivemos apenas aqueles que foram experimentalmente validados.

DBAASP

O banco *DBAASP: Database of Antimicrobial Activity and Structure of Peptides* (Pirtskhalava *et al.*, 2016) é o maior banco encontrado em quantidade de PAMs, além de possuir diversas informações sobre a composição estrutural de seus peptídeos.

Também é fornecido uma interface para acesso programático, o que facilitou a aquisição de todas as sequências presentes na base. Os dados continham alguns peptídeos multímeros, os quais foram descartados devido a limitação da arquitetura do modelo. Novamente foram mantidas apenas as sequências experimentalmente validadas.

DRAMP

DRAMP - Data Repository of Antimicrobial Peptides (Fan *et al.*, 2016) é um repositório de PAMs que não se limita apenas a compostos já testados, contendo diversas sequências que ainda estão em fases de testes e disponibilizando essas informações para pesquisadores.

Para nossos propósitos no entanto, fizemos o download apenas das sequências já validadas. Escolhemos também não utilizar os peptídeos provenientes de patentes, devido a dificuldade em extrair as sequências e informações dos documentos e carência de informações sobre sua efetividade presentes na base. Utilizamos o grupo classificado como "General AMPs"pelo site.

PhytAMP

PhytAMP - A Database Dedicated to Plant Antimicrobial Peptides (Hammami *et al.*, 2009) é, como o nome indica, uma coleção detalhada de PAMs já identificados em plantas, contendo diversas informações físico-químicas, biológicas e farmacológicas sobre os compostos. Todos os compostos foram experimentalmente validados, e foram baixados através do arquivo ".xml"fornecido, do qual os dados foram extraídos utilizando a biblioteca *Beatifull-Soup*.

2.1.2 Junção e Uniformização das Bases

Na Tabela 2.1 podemos observar o nome, a quantidade de AMPs e AVPs e a data de aquisição dos dados, que se torna relevante pelas constantes atualizações de vários desses bancos de dados.

Nome do Banco	PAMs (PAVs)	Data de Aquisição (d/m/a)
ADAM (Lee *et al.*, 2015)	3152 (183)	20/07/2020
APD3 (G. Wang *et al.*, 2016)	2654 (161)	02/10/2020
AVPdb (Qureshi *et al.*, 2014)	2059 (2059)	30/07/20220
CAMP (Waghu *et al.*, 2016)	2764 (98)	30/09/2020
DBAASP (Pirtskhalava *et al.*, 2016)	11736 (1059)	20/08/2020
DRAMP (Fan *et al.*, 2016)	3653 (148)	31/07/2020
PhytAMP (Hammami *et al.*, 2009)	273 (15)	30/07/2020

Tabela 2.1: *Bases de Dados Originais*

No intuito de montar um banco de dados de peptídeos bioativos validados experimentalmente não redundante, todos os peptídeos bioativos obtidos a partir dos sete bancos de dados distintos foram reunidos. Inicialmente as informações sobre sequências idênticas em bancos diferentes foram reunidas e padronizadas, em seguida removemos os peptídeos com aminoácidos que não estejam no conjunto dos 20 aminoácidos naturais, mais B,O,U,Z e que não sejam aminoácidos desconhecidos (representados por um X) ou 'gaps' integrais a sequência, representados pelo caractere '-'. Essas filtragens foram feitas com um script em linguagem *python* (versão 3.9), se utilizando principalmente da biblioteca *pandas*.

A aplicação de todos os filtros e critérios permitiu compor um conjunto de sequências experimentalmente validadas com 15.650 PAMs, dos quais 2.812 possuem atividade antiviral (PAVs).

Além das sequências de aminoácidos desses PAMs, foram recuperadas dos diferentes bancos de dados diversas informações descrevendo características funcionais, além das quantidades de cada tipo de peptídeo para as classificações mais comuns (Tabela 2.2). Estudos tem relatadodo que os peptídeos bioativos podem ter mais de um tipo de atividade, o que é corroborado a partir dos dados obtidos desses bancos públicos de peptídeos validados experimentalmente através do Diagrama de Venn (Figura 2.2) na interseção entre as classes mais comuns de bioativos.

Atividade	Quantidade de Peptídeos
Anti-bacteriana (Gram+/Gram-/Não especificado)	9696 / 9993 / 537
Anti-Células de Mamíferos	5602
Antifúngica	4747
Antiviral	2811
Anticâncer	1642
Anti-Candidíase	564
Antiparasítica	315
Atividade Hemolítica	276
Inseticida	154
Inibidor Enzimático	18

Tabela 2.2: *Quantidade de Peptídeos em Cada Grupo Funcional*

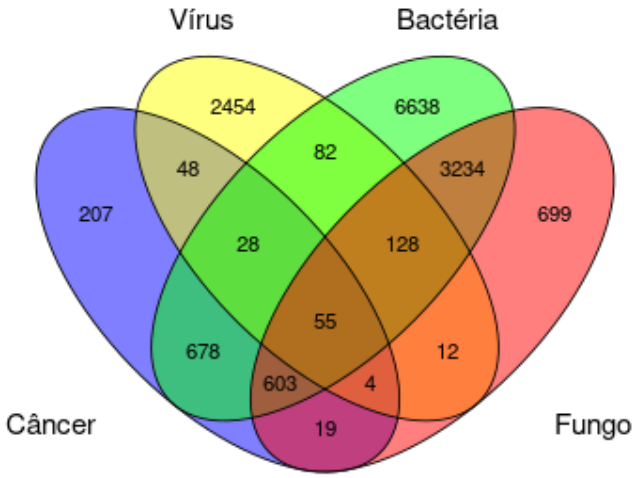

Figura 2.2: *Quantidade de PAMs por Tipo de Alvo*

2.1.3 Peptídeos sem Atividade

Para realizar o treinamento do modelo (explicado em mais detalhes na seção 1.3.2), é necessário um conjunto de peptídeos que não possua atividade antiviral ou antimicrobiana, tornando possível a identificação de padrões exclusivos aos PAMs e PAVs.

Uma dificuldade de trabalhos semelhantes é que não existem bancos de dados de peptídeos **sem** nenhum tipo de ação funcional experimentalmente validada (Veltri *et al.*, 2018; Torrent *et al.*, 2011). Seguimos então uma abordagem semelhante à desses trabalhos para a construção desse conjunto de dados. As etapas do processo podem ser visualizadas na Figura 2.1.

Primeiramente , obtivemos sequências de proteínas do banco de proteínas UniprotSwissprot (UniProt, 2021), com a seguinte consulta:

```
NOT antimicrobial NOT antibiotic
NOT antiviral NOT antifungal
NOT effector NOT excreted
locations:(location:"Cytoplasm [SL-0086]")
length:[10 TO 600] AND reviewed:yes
```

Essa busca pode ser traduzida para: "Proteínas sem ação antimicrobiana, antibiótica, antiviral ou antifúngica, que não sejam efectoras e que não sejam secretadas". Além disso também exigimos que elas estejam localizadas no citoplasma celular, que tenham entre 10 e 600 aminoácidos de comprimento, e cujas informações tenham sido manualmente revisadas.

Todos esses critérios são impostos para garantir com o mais alto nível de confiança

possível que não teremos proteínas com algum dos tipos de atividade presente no banco de PAMs construído na seção anterior.

Essa busca resultou em 135.497 sequências de proteínas. Para gerar os peptídeos, realizamos a fragmentação de todas essas sequências utilizando o programa *pepdigest* da suíte EMBOSS com os parâmetros padrão. Removemos desses peptídeos aqueles que estão presentes no conjunto de PAMs.

Ao final obtivemos 1.227.076 peptídeos únicos nesse banco, os quais podemos assumir que não tem ação contra os alvos biológicos que temos interesse.

Para os PAVs, realizamos um passo adicional, buscando na literatura informações sobre seu mecanismo de atividade, e categorizando os que possuíam essa informação disponível em 3 classes: **"Membrana"** (n PAVs), para peptídeos que tem interação com a membrana do capsídeo viral; **"Replicação"** (n PAVs), para peptídeos que impedem a replicação dos vírus dentro da célula; e **"Montagem"** (n PAVs), para peptídeos que impedem o processo de montagem das novas partículas virais dentro da célula. Esse processo pode ser visto na Figura 2.3.

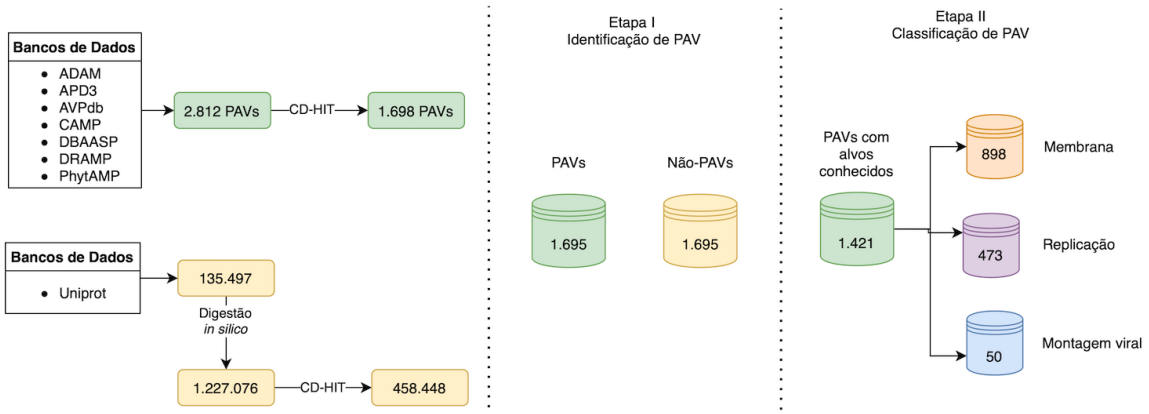

Figura 2.3: *Etapas do Processamento dos AVPs.*

2.1.4 Análises dos Peptídeos Coletados

Após a coleta e o processamento das sequências de peptídeos descrita nas seções anteriores, algumas análises iniciais foram realizadas. Os conjuntos foram analisados antes da busca de homologias entre as proteínas. Foram utilizados os 15.686 PAMs (AMPs nos gráficos), 2.812 PAVs (AVPs nos gráficos) e 1.279.077 peptídeos sem atividade (Não AMPs nos gráficos).

Primeiramente fizemos análises iniciais da composição média de Aminoácidos nas sequências de cada conjunto (Figura 2.4), e pudemos identificar as distribuições do comprimento das sequências de cada conjunto (Figura 2.5).

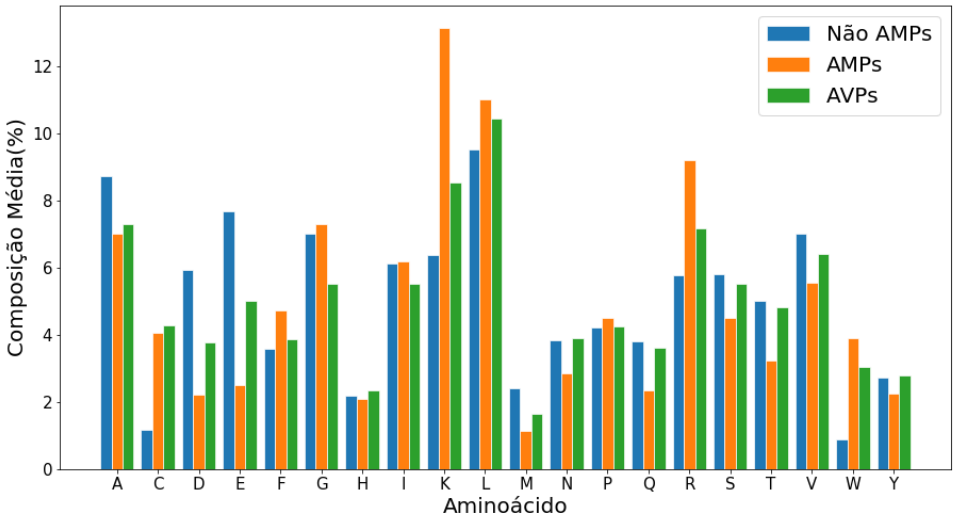

Figura 2.4: *Composições de AA nos conjuntos*

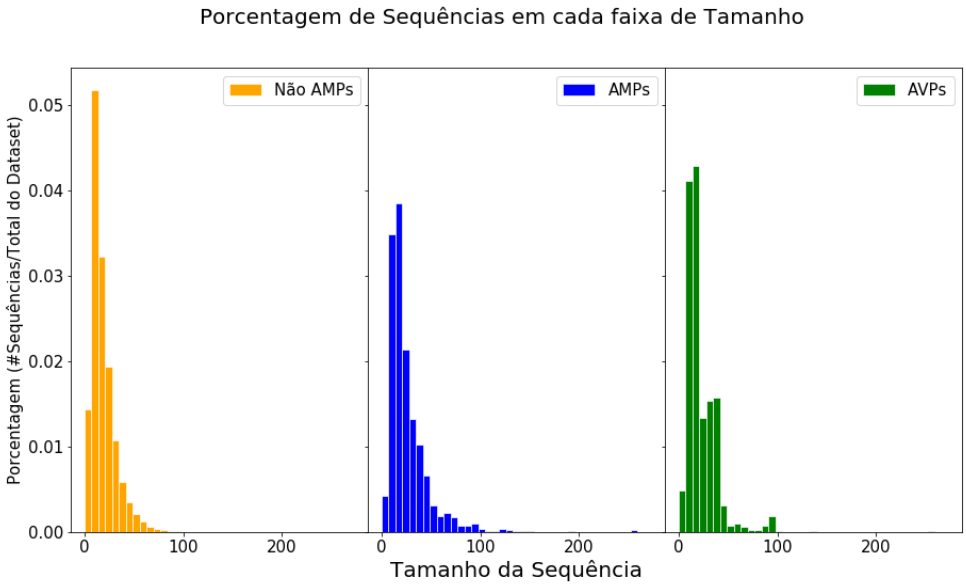

Figura 2.5: *Tamanho das sequências nos conjuntos*

A Figura 2.4 mostra evidências de diferenças na composição de PAMs, PAVs e peptídeos sem atividade, mas que essa informação por si só pode não ser suficiente para determinar a classe dos peptídeos. Podemos destacar as diferenças nas presenças de Lisina (K) e Cisteína (C) entre PAMs e PAVs quando comparados com peptídeos sem atividade.

Podemos observar também algumas diferenças entre os conjuntos de PAMs e PAVs, as diferenças nas quantidades de Lisina (K) e Ácido Glutâmico (E) se destacam. Isso era esperado, pois apesar do agrupamento sob o rótulo de PAMs e de alguns peptídeos possuírem vários tipos de atividade funcional, as interações reais entre cada sequência e seu alvo são extremamente específicas, criando diferenças significativas entre esses conjuntos.

Temos que a distribuição dos tamanhos (Figura 2.5) entre os conjuntos é similar, mas é importante ressaltar que o conjunto de "Não AMPs"contém uma quantidade muito maior de sequências. Isso é importante pois possibilita criar um cojunto de treinamento no qual o tamanho da sequência não influencie o resultado, como descrito na seção 2.2.

De forma complementar analizamos se n-gramas diferentes teriam alguma prevalência em algum desses conjuntos de dados. A fim de checar esse fato criamos os gráficos mostrados nas figuras 2.6 e 2.7, que mostram a composição média de alguns n-gramas.

A composição do n-grama g em cada sequência S foi calculada conforme a equação 2.1, onde n é o tamanho do n-grama (2 ou 3 no nosso caso), c é o número de vezes que o n-grama aparece na sequência nas duas direções, e t é o tamanho da sequência.

$$NGC(S, g) = \frac{c}{t - n} \tag{2.1}$$

Essa equação equivale à frequência(%) de vezes que o n-grama está presente na sequência considerando todas as posições possíveis. O valor nos gráficos é a média dessa compo-

sição para todas as sequências no banco para cada n-grama. Para facilitar a visualização foram selecionados os 4 n-gramas com maior composição média em cada banco de maneira separada, e depois foram mostradas os valores das composições para todos os bancos.

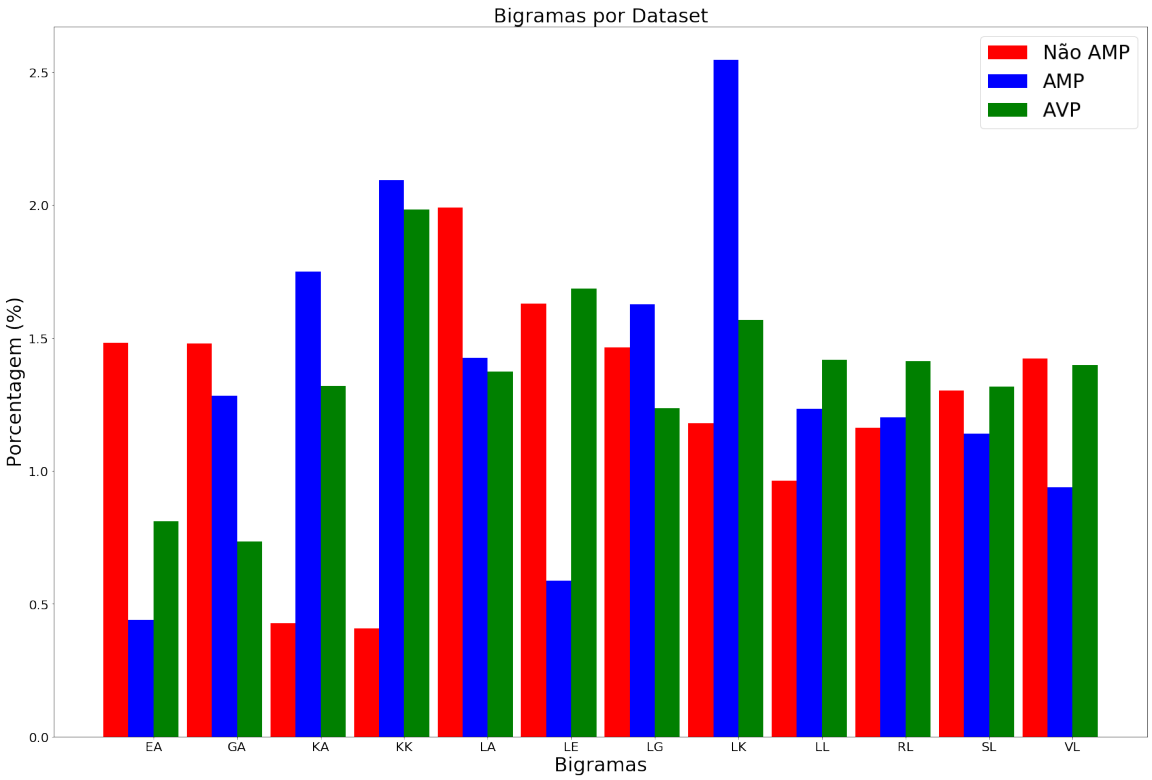

Figura 2.6: *Composições de bigramas nos Conjuntos*

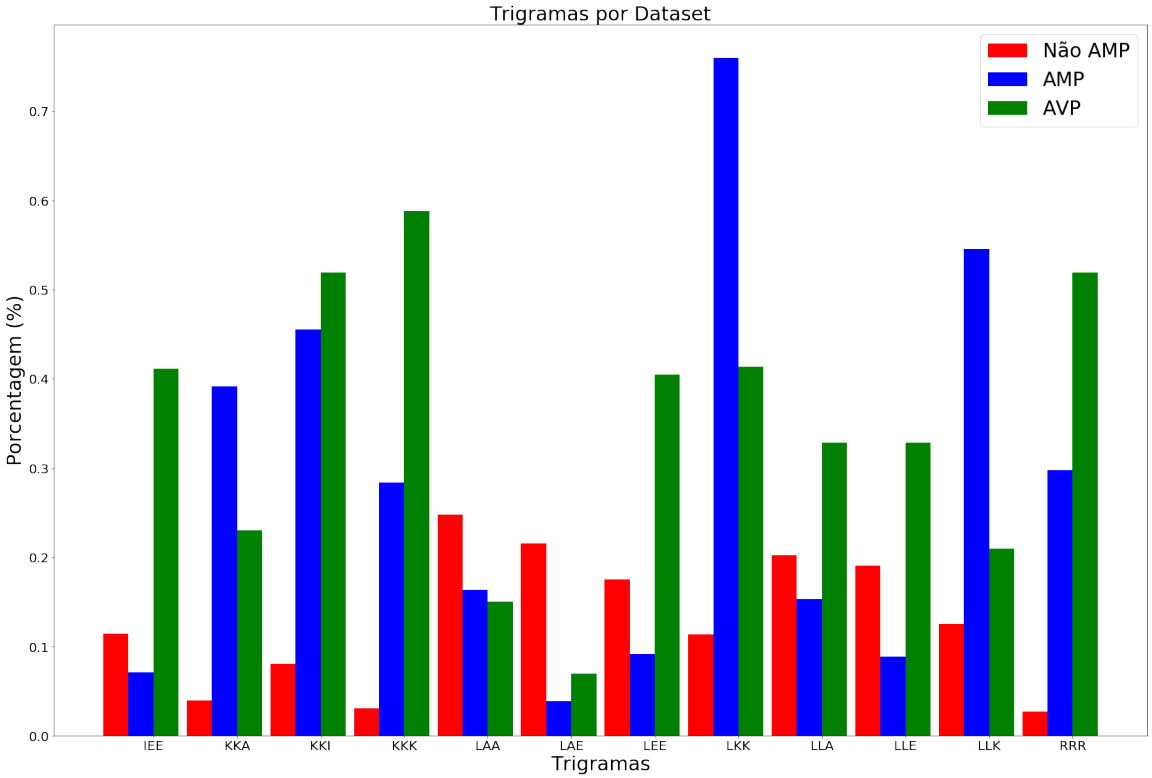

Figura 2.7: *Composições de trigramas nos Conjuntos*

Podemos observar nos gráficos 2.6 e 2.7 que exitem diferenças também na composição de n-gramas nas sequências, principalmente envolvendo o aminoácido Lisina (K). Vale notar a proeminência do bigrama LK e do trigrama LKK no conjunto de PAMs tanto em relação aos peptídeos sem atividade quanto aos PAVs, indicando novamente as diferenças entre os dois conjuntos utilizados como classe positiva.

2.2 Preparação dos Dados Para Treinamento

Após a aquisição inicial dos dados foram realizadas outras etapas de processamento em antecipação ao processo de treinamento. O modelo que iremos utilizar é classificado como um modelo de Aprendizado "Supervisionado", ou seja, já são fornecidos dados corretamente classificados, e a partir dessas informações o modelo aprende as diferenças entre essas classificações.

A primeira fase do modelo é baseada em duas classes ou classificações: Peptídeos com atividade Antiviral e Peptídeos sem atividade Antiviral, as classes "Positiva"e "Negativa"respectivamente, e é necessário fornecer exemplos de todas as categorias para o processo de aprendizagem. Caso o modelo fosse treinado apenas com PAVs, ele faria a predição e classificação de todos os peptídeo de entrada como PAV. O mesmo raciocínio se aplica para a segunda fase do modelo, que classifica compostos já identificados como PAVs em uma das 3 classes de atividade, "Membrana", "Replicação"ou "Montagem".

Uma das preocupações principais na área de *Deep Learning* é a capacidade do modelo

de manter o mesmo nível de performance em dados cujo modelo não tenha sido treinado, ou seja, que o modelo seja "generalizável".

Por isso é muito importante que os dados utilizados para o treinamento do modelo sejam "representativos"do espaço de possibilidades do problema, ou seja, tenham uma distribuição das suas diversas características similar a encontrada na população. Se os dados de treinamento possuírem um viés em relação a essa distribuição "real", esse viés irá se refletir no modelo e implicará na perda de eficiência do preditor e aumentar a taxa de falsos positivos.

Outro problema comum com esses modelos é o *overfitting*, onde o modelo acaba se tornando hiperespecializado para os dados de treinamento, e acaba não sendo aplicável em outras situações. Esse problema ocorre por vários motivos, porém é comum tentar reduzir a redundância interna do seu conjunto de dados como uma das precauções contra esse efeito colateral.

Busca por Homologias

Realizamos uma busca por homologias entre proteínas utilizando o programa *cd-hit* (Fu *et al.*, 2012) em 3 conjuntos de dados, a fim de eliminar redundâncias e evitar o *overfitting*. Os conjuntos foram: todos os PAMs, apenas os PAVs, e todos os peptídeos sem atividade. Esses peptídeos foram reunidos na seção 2.1, e em todos os conjuntos removemos as sequências com menos de 5 aminoácidos de comprimento.

Nos conjuntos de peptídeos Antimicrobianos e no de peptídeos Antivirais utilizamos os parâmetros `-c 0.90 -aS 0.90 -aL 0.90` (`-c` corresponde ao corte para a porcentagem de identidade entre os peptídeos, `-aS` corresponde à porcentagem de cobertura do peptídeo mais curto e `-aL` corresponde à porcentagem de cobertura do peptídeo mais longo), resultando em 9.620 e 1.698 peptídeos não redundantes, respectivamente. Para a redução da redundância no conjunto de peptídeos não bioativos utilizamos os parâmetros `-c 0.65 -aS 0.90 -aL 0.85`, resultando em um total de 458.448 peptídeos não redundantes.

Construção dos Bancos Negativo e Positivo

Após a eliminação das redundâncias, construímos dois conjuntos de dados, um para treinamento no problema de identificação de PAMs em geral, e um para o problema mais específico da identificação de PAVs. Os dois conjuntos de dados são divididos internamente em um banco Positivo, com as sequências não redundantes de PAMs e PAVs respectivamente, e um banco Negativo, com sequências de peptídeos sem atividade.

Como discutido antes, temos que o conjunto de possíveis peptídeos sem atividade, mesmo após uma eliminação de redundâncias mais estrita, é muito maior do que o conjunto de PAMs e PAVs, o que poderia potencialmente causar um viés no modelo. Isso ocorreria pois um modelo que identificasse todas as sequências como negativas ainda teria bons resultados, mas não seria útil. Sendo assim temos que escolher, dentre os possíveis peptídeos sem atividade, aqueles que irão compor os dados de treinamento.

Fizemos essa escolha da seguinte maneira: para cada peptídeo no banco Positivo de cada conjunto, um peptídeo de mesmo comprimento (mesma quantidade de aminoácidos)

no conjunto de peptídeos não bioativos. Esse processo foi estabelecido para evitar que o tamanho da sequência do peptídeo fosse utilizado pelo modelo como uma característica de diferenciação e evitar de causar algum viés ou perda de eficiência do modelo. PAMs e PAVs para os quais não havia um peptídeo sem atividade no conjunto negativo de mesmo comprimento foram excluídos do conjunto positivo.

Ao final obtivemos esses dois conjuntos: no conjunto para a identificação de PAMs temos 9.567 sequências positivas e 9.567 negativas, e no conjunto de identificação de PAVs temos 1.695 sequências positivas e 1.695 sequências negativas. O processo inteiro pode ser observado na Figura 2.1.

Construção do Banco de Classes de Atividade

A partir do conjunto de PAVs selecionado na seção anterior, buscamos nos bancos e na literatura pelo mecanismo de atividade desses compostos. A partir dessa informação, classificamos os PAVs que continham informações a respeito de seu mecanismo de ação ou alvo molecular, totalizando 1.399 peptídeos. Desses, foi feita uma separação manual em três classes: "Membrana", aqueles que possuem alvos na membrana da célula hospedeira ou que possuem ação virucida atuando na membrana; "Replicação", aqueles que possuem alvos relacionados com o processo de replicação viral e "Montagem", envolvidos no processo de montagem do vírus. O número total de PAVs para cada classe pode ser visto na Tabela 2.3 .

Classe	nº PAVs
Membrana	800
Replicação	552
Montagem	47
Sem Classe	296

Tabela 2.3: *Números totais de peptídeos para cada classe.*

2.3 Transcriptomas de Aracnídeos

O Centro National de Informação Biotecnológica (NCBI) é um dos principais bancos de dados referência mundial como repositório e repleto de funcionalidades para análise de dados biológicos, com o foco no avanço da ciência e saúde ao fornecer acesso às informações biomédicas e genômicas.

Baseado nesse banco foram baixados diversos transcriptomas resultantes de RNA-seq de aracnídeos, disponibilizados na base de dados do NCBI (Wheeler *et al.*, 2007), com interesse em organismos venenosos em particular aranhas, carrapatos e escorpiões. A Tabela 2.4 contém mais informações sobre como se caracterizam esses transcriptomas utilizados para cada grupo.

Grupo	Total de Transcriptomas para cada grupo	Total de Proteomas para cada grupo
Aranhas	58	13
Carrapatos	68	29
Escorpiões	14	1

Tabela 2.4: *Amostras de Transcriptomas de Aracnídeos*

Dentre essas amostras, foram identificadas 15 amostras de glândulas de veneno (6 de aranhas e 9 de escorpiões) e 23 amostras de glândulas salivares de carrapatos.

Selecionamos as amostras de 3 espécies diferentes, uma de cada grupo, para realizar uma busca por novos candidatos a peptídeos bioativos. A tabela 2.5 contém o número de transcritos por amostra.

Espécie	Nº de Transcritos	Proteínas Encontradas	Peptídeos Gerados
Hadrurus spadix (Escorpião)	744	203	4996
Hyalomma dromedarii (Carrapato)	142931	7116	106167
Phoneutria nigriventer (Aranha)	49992	1700	27747
Total	**193667**	**9019**	**138910**

Tabela 2.5: *Proteínas e peptídeos gerados insilico para transcriptomas de aracnídeos coletados.*

Esses transcritos foram traduzidos *in silico* pela ferramenta *Transdecoder*, que gera várias possíveis proteínas para cada transcrito. Filtramos esses candidatos através de buscas nas bases de dados PFAM, Uniprot-SwissProt e Uniprot Toxins. Selecionamos então as proteínas com os melhores resultados, limitando à uma proteína por transcrito. A quantidade de proteínas encontradas pode ser vista na tabela 2.5.

Realizamos então a digestão *in silico* de todas as proteínas encontradas, gerando um banco de 138910 peptídeos que foram investigados pelo preditor.

Capítulo 3

Modelo

3.1 Representação das Proteínas

Uma parte importante em qualquer problema de aprendizado é como são representados as entradas, mais especificamente como o modelo recebe os dados de entrada. É comum em dados de texto, como nossas sequências, a utilização de uma codificação vetorial de cada caractere, pois existem otimizações feitas na implementação dos modelos de *Deep Learning* que favorecem a utilização de operações matriciais.

Uma codificação comum é a *One-Hot encoding*, em que para cada aminoácido criamos um vetor de tamanho 26 (todos os valores possíveis incluindo caracteres especiais), com zeros em todas as posições exceto uma, que seria determinada de maneira única para cada tipo de aminoácido.

Outra representação são os chamados *embeddings*, muito comuns em modelos de Processamento de Linguagem Natural (Y. Wang *et al.*, 2016; Melamud *et al.*, 2016). Um *embedding* possibilita a criação de uma representação vetorial contínua dos diversos pontos discretos da entrada. No nosso caso específico, cada aminoácido é convertido em um vetor de tamanho fixo pré-definido, onde os valores específicos desse vetor são a representação desse aminoácido em um espaço multidimensional.

Embeddings são utilizadas pois possibilitam a representação de proximidades entre pontos da entrada, por meio da posição dos seus vetores, o que facilita o reconhecimento de padrões mas genéricos pela rede. Representações desse tipo de aminoácidos já foram propostas (Xu *et al.*, 2017; Asgari e Mofrad, 2015), e podem ser utilizadas em outros modelos, mas também é comum o treinamento de camadas de *embedding* em consonância com o modelo sendo utilizado (Veltri *et al.*, 2018).

Uma outra modificação possível é a utilização de conjuntos de aminoácidos como a "unidade básica"do modelo, ou seja, ao invés de ler a sequência do peptídeo com um único aminoácido por vez, podemos utilizar pares ou triplas de aminoácidos, criando essas representações a partir desses agrupamentos, que denominaremos de n-gramas.

3.1.1 Seleção das Features

Uma das considerações mais importantes na construção de um modelo de Aprendizado de Máquina é a escolha das *features*, ou seja, a seleção dos dados que serão efetivamente passados para o modelo. Para o módulo LSTM do modelo essa escolha é simplificada, pois a sua capacidade de lidar com entradas sequenciais de tamanho variável permite que ele receba uma representação direta da sequência de aminoácidos. Para o módulo RF, no entanto, temos que selecionar uma quantidade fixa de valores numéricos, que possam ser calculados sobre qualquer sequência. Nas próxima subseções descrevemos os conjuntos de valores candidatos utilizados, e na seção 4.5 descrevemos o processo de seleção que foi aplicado.

AAC

AAC se refere à *Amino Acid Composition*, Composição de Amino Ácidos. Para cada proteína, há um vetor de 21 posições, uma para cada amino ácido padrão e uma para resíduos desconhecidos (letra "X"), em que cada posição contém a quantidade de vezes que o resíduo designado aparece na proteína, dividida pela quantidade total de resíduos.

PCP

São as *Physico-Chemical Properties*, Propriedades Físico-químicas, definidas em (Lissabet *et al.*, 2019). São uma coleção de propriedades facilmente obtidas a partir da sequência que pontencialmente fornecem informações sobre o caráter funcional de uma proteína. São elas:

- A composição de amino ácidos (como definida em 3.1.1) (20 valores).
- A quantidade de hidrogênios livres nos resíduos.
- A carga total nos resíduos.
- O peso molecular da proteína.
- A composição (quantidade total dividida por tamanho da proteína) de aminoácidos das seguintes categorias, definidas em (Lissabet *et al.*, 2019): muito pequenos, pequenos, grandes, alifáticos, aromáticos, com carga, com carga positiva, com carga negativa, polares, apolares e hidrofóbicos (11 valores).

Totalizando um vetor com 34 valores para cada proteína.

Pseudo AAC

A Pseudo-Composição de Amino Ácidos (PseAAC) (Chou, 2001), é uma modificação da Composição de Amino Ácidos tradicional, que tenta incorporar informações sobre a ordem dos Amino Ácidos na Sequência. Considere uma sequência de amino ácidos de tamanho L, podemos representá-la como a sequência $R_1R_2R_3...R_L$, onde cada R_i representa o aminoácido na posição i.

Definimos o fator de correlação de ordem k da sequência como:

$$\theta_k = \frac{1}{L-k} \sum_{i=1}^{L-k} \Theta_i(R_i, R_{i+k}) \tag{3.1}$$

Onde $\Theta(.,.)$ é definido como:

$$\Theta(R_i, R_j) = \frac{1}{3}\left([H_1(R_j) - H_1(R_i)]^2 + [H_2(R_j) - H_2(R_i)]^2 + [M(R_j) - M(R_i)]^2 \right) \tag{3.2}$$

Onde $H_1(R_i)$, $H_2(R_i)$ e $M(R_i)$ são os valores normalizados da hidrofobicidade (Tanford, 1962), hidrofilicidade (Hopp e Woods, 1981) e massa da cadeia lateral.

A PseAAC pode então ser descrita como o vetor $\boldsymbol{X} = [x_1, ..., x_{20+\lambda}]^T$, onde cada componente x_u desse vetor é dada por:

$$x_u = \begin{cases} \dfrac{f_u}{\sum_{i=0}^{20} f_i + w \sum_{j=1}^{\lambda} \theta_j}, & (1 \leq u \leq 20) \\ \dfrac{w\theta_{u-20}}{\sum_{i=0}^{20} f_i + w \sum_{j=1}^{\lambda} \theta_j}, & (20+1 \leq u \leq 20+\lambda) \end{cases} \tag{3.3}$$

Onde θ_i é o fator de correlação de ordem i, f_i é a frequência normalizada de ocorrência dos 20 amino ácidos tradicionais na sequência, λ é o parâmetro que define quantos efeitos de correlação serão considerados, e w é o parâmetro que definie o peso a ser dado para os fatores de correlação.

O vetor $\boldsymbol{X}$ é então composto por $20 + \lambda$ componentes.

CTD

Outro de conjunto de descritores numéricos considerado foram os Descritores de Composição, Transição e Distribuição (CTD), descritos em (Dubchak, Muchnik, Holbrook *et al.*, 1995; Dubchak, Muchnik, Mayor *et al.*, 1999). Eles são uma tentativa de incorporar características sobre a ordenação dos aminoácidos na proteína, mas consideram divisões dos aminoácidos em diferentes grupos a depender das características biológicas sendo consideradas.

Vamos considerar como calcular os descritores para uma propriedade genérica, que divide os aminoácidos em dois grupos: Grupo A e Grupo B. Escrevemos então a Sequência de aminoácidos de uma proteína de exemplo, substituindo o aminoácido pelo seu grupo dentro dessa propriedade, temos então:

```
Sequência:                A B B A B B B B B A A A A B B B A B A B B B B B A A
Numeração da Sequência:   1       5       10      15      20      25
Numeração Grupo A:        1     2           3 4 5 6       7   8           9 10
Transições A-B:            |     |                 |      |   |
Transições B-A:                |           |             |   |           |
```

Temos que essa sequência possui 10 resíduos do tipo A(n_A), e 16 resíduos do tipo B (n_B). Calculamos os descritores de Composição (C) como $n_A \times 100/(n_A + n_B) = 38.5\%$ e $n_B \times 100/(n_A + n_B) = 61.5\%$.

Os descritores de Transição (T), são a porcentagem das vezes que um tipo de resíduo ocorre após o outro, no nosso exemplo temos 10 transições de A para B, ou de B para A. O valor para a transição AB seria de 10/25 = 40%.

Por fim temos os descritores de Distribuição, para cada grupo em uma propriedade, eles são compostos de 5 tamanhos da sequência (em porcentagem) dentro dos quais 0,25,50,75 e 100 % dos aminoácidos do grupo estão contidos. Por exemplo, para o grupo A, o primeiro valor seria 0, pois o primeiro aminoácido A está na primeira posição. O segundo valor seria $(4/26) \times 100\% = 15.4\%$, o terceiro valor seria $(12/26) \times 100\% = 46.1\%$ e assim por diante.

No nosso estudo utilizamos 7 propriedades para agrupar os aminoácidos, exibidas na tabela 3.1 com seus repectivos agrupamentos.

Propriedade	**Grupos**
Hidrofobicidade	(DEKNQR), (AGHPSTY), (CLVIMFW)
Volume VDW Normalizado	(GASTPDC), (NVEQIL), (MHKFRYW)
Polaridade	(LIFWCMVY), (PATGS), (HQRKNED)
Carga	(KR), (ANCQGHILMFPSTWYV), (DE)
Estrutura Secundária	(EALMQKRH), (VIYCWFT), (GNPSD)
Acessibilidade de Solventes	(ALFCGIVW), (RKQEND), (MPSTHY)
Polarizabilidade =	(GASDT), (CPNVEQIL), (KMHFRYW)

Tabela 3.1: *Propriedades Utilizadas no cálculo do CTD*

Cada Propriedade acima é representada por 21 valores CTD, criando um vetor de 147 valores entre 0 e 1.

3.2 Arquitetura

A arquitetura é separada em duas fases distintas, que funcionam de maneira independente. A primeira fase, composta por dois modelos de aprendizado que tem suas predições combinadas, é focada no problema de classificação binário dos peptídeos em "Positivo"e "Negativo". A segunda fase, composta por um único modelo de aprendizado, é treinada no problema de classificação do peptídeo em uma das 3 classes de atividade antiviral delimitadas pelo nosso banco de dados, "Membrana", "Replicação"e "Montagem". Nessa seção descreveremos a arquitetura em detalhes. A Arquitetura completa pode ser vista na Figura 3.1.

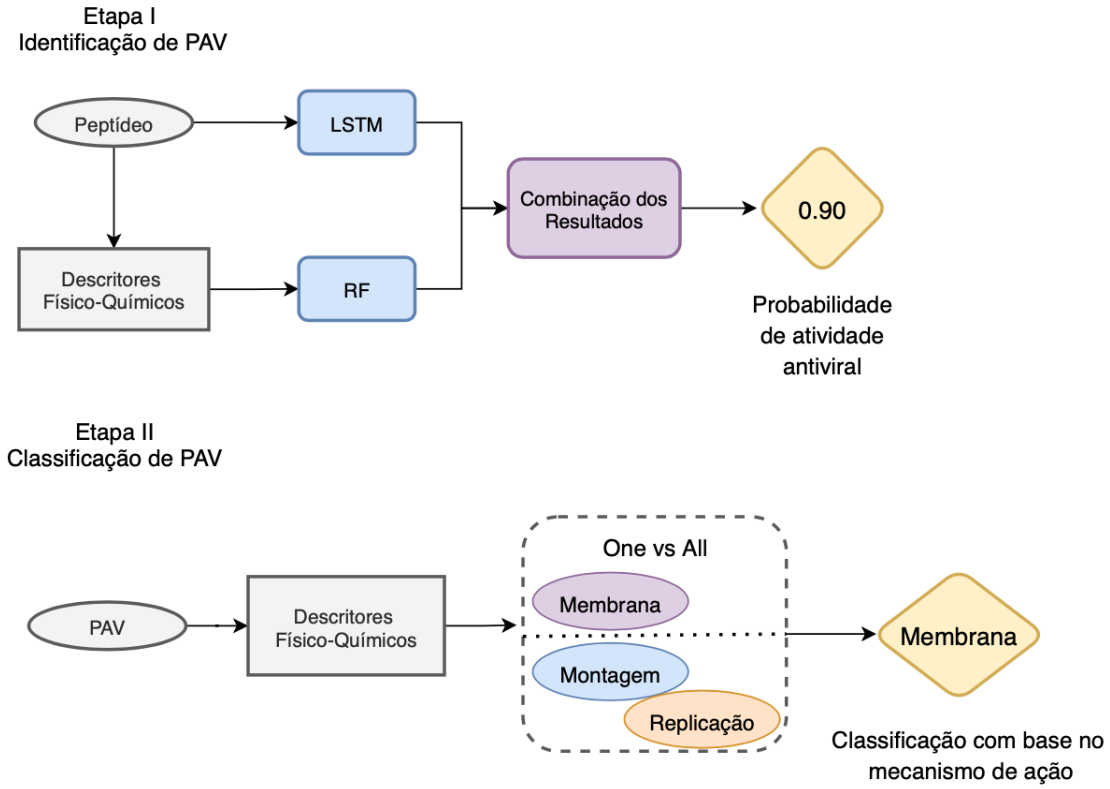

Figura 3.1: *Diagrama representando a arquitetura do preditor EnAVPClass*

3.2.1 Primeira Fase

A primeira fase do preditor é composto de dois módulos, os quais são em si modelos de aprendizado de máquina independentes. Os dois módulos recebem diferentes informações sobre a sequência sendo analisada e realizam suas predições individualmente sobre a bioatividade da sequência. Essas predições são então combinadas linearmente para gerar o resultado final. O processo de avaliação de uma sequência pode ser visto na figura 3.2.

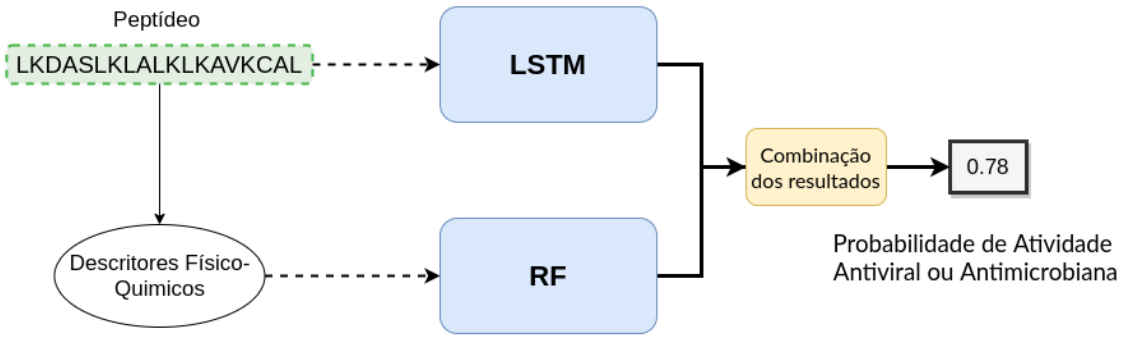

Figura 3.2: *Diagrama representando o modelo de Ensemble utilizado na primeira fase do EnAVPClass*

Um dos módulos é uma Rede Neural Recorrente Bidirecional Multicamada (módulo LSTM), que recebe como entrada a sequência de aminoácidos a ser avaliada, e produz

um número entre 0 e 1 que pode ser interpreado como a confiança da rede de que aquela sequência possui a atividade sendo investigada.

O outro módulo é um modelo de Floresta Aleatória ou *Random Forest* (módulo RF), que recebe um conjunto de descritores numéricos calculados sobre a sequência, as chamadas *features*, e não a sequência de aminoácidos em si. Esse modelo é composto de uma quantidade N de árvores de decisão simples, que implementam um plano que divide o espaço multidimensional das *features* em dois. Cada divisão desse plano é uma classificação dos pontos desse espaço (sendo que cada ponto representa uma proteína neste caso) em duas classes diferentes (que neste caso seriam "com atividade"e "sem atividade"). O modelo de Floresta aleatória combina as predições feita por todas as árvores individuais em um único valor final.

A combinação linear desses dois módulos ocorre de acordo com a equação 3.4, onde S é a sequência de amino ácidos sendo avaliada, $P_F(S)$ é a Probabilidade Final atribuída pelo modelo de que a sequência S possui a atividade desejada, $P_{\text{LSTM}}(S)$ e $P_{\text{RF}}(S)$ são as probabilidades atribuídas à sequência S pelos módulos LSTM e RF respectivamente, e ω é um parâmetro do modelo que pode assumir valores entre 0 e 1.

$$P_F(S) = \omega P_{\text{LSTM}}(S) + (1 - \omega) P_{\text{RF}}(S) \tag{3.4}$$

3.2.2 Segunda Fase

A Segunda fase do modelo é um preditor SVM, que também se baseia em características fisíco-químicas dos PAVs. Ele foi treinado com o banco separado em classes descrito na seção 2.2, porém realizamos um pré-processamento com a técnica de *Synthetic Minority Oversampling Technique* (SMOTE) (Chawla *et al.*, 2002) de forma a balancear a quantidade de exemplos em cada classe, terminando ao final com 600 exemplos para cada tipo de atividade (1800 PAVs no total).

Com essa técnica novos elementos são gerados com base nos dados disponíveis. Considerando uma amostra com valores x_i, um novo elemento x_{novo} será gerado com base em seus k vizinhos próximos. Um dos vizinhos é selecionado aleatoriamente (x_{zi}) e uma nova amostra é gerada pela fórmula na equação 3.5.

$$x_{novo} = x_i + \lambda(x_{zi} - x_i) \tag{3.5}$$

Onde λ é um número aleatório entre $[0, 1]$. Os dados após o SMOTE foram normalizados com o método de *Standard Scaler*, que normaliza as características removendo a média e escala por variância de unidade. O valor da amostra x normalizada (z) é calculado pela equação 3.6.

$$z = \frac{x - \bar{x}}{\sigma} \tag{3.6}$$

No qual $\bar{x}$ é a média e σ é o desvio padrão dos dados de treinamento.

A partir desse conjunto ampliado de dados, foi feita uma busca por hiperparâmetros de forma a otimizar o modelo de SVM, utilizando-se a função *Grid Search* do pacote *sklearn*.

Deste modo, a aplicação desenvolvida está organizada em duas principais etapas, sendo a primeira responsável pela predição dos peptídeos bioativos recebendo um score que pode variar entre [0, 1] e a segunda etapa a qual utilizará apenas os peptídeos classificados como PAVs com score acima de 0.5 para a classificação pelo mecanismo de ação. Ao final do pipeline será gerado um arquivo de resultado final contendo cinco colunas, sendo a primeira o ID do peptídeo, a segunda coluna contendo o score de predição do PAV, e as três últimas colunas contendo a probabilidade conjunta do PAV para cada um dos três mecanismos de acção, sendo a maior probabilidade o mecanismo mais provável.

Capítulo 4

Resultados

4.1 Métricas Utilizadas

As métricas utilizadas para a avaliação dos modelos foram a sensibilidade (SENS), especificidade (SPEC), precisão (PREC), acurácia (ACC), o coeficiente de correlação de Mathews (MCC) (Matthews, 1975), e o score F1 calculados da seguinte maneira:

$$
\begin{aligned}
&TP = \text{Verdadeiros Positivos}\\
&FP = \text{Falsos Positivos}\\
&TN = \text{Verdadeiros Negativos}\\
&FP = \text{Falsos Negativos}\\
&SENS = \frac{TP}{TP + FN}\\
&SPEC = \frac{TN}{TN + FP}\\
&PREC = \frac{TP}{TP + FP}\\
&ACC = \frac{TP + TN}{TP + TN + FP + FN}\\
&MCC = \frac{(TP * TN) - (FP * FN)}{\sqrt{(TP + FN)(TN + FP)(TP + FN)(TN + FN)}}\\
&F1 = \frac{2 * PREC * SENS}{PREC + SENS} = \frac{2 * TP}{2 * TP + FP + FN}
\end{aligned}
\tag{4.1}
$$

Onde Verdadeiros Positivos são PAMs ou PAVs classificados como positivos, Falsos Positivos são peptídeos sem atividade classificados como positivos, Verdadeiros Negativos são peptídeos sem atividade classificados como negativos e Falsos Negativos são PAMs ou PAVs classificados como negativos.

Essas métricas também podem ser visualizadas através de uma matriz de confusão,

onde o número de exemplos que se enquadra em cada tipo de resultado é colocado em uma grade com quadrantes determinados pelas classes reais do exemplo em um eixo, e pela classe predita pelo modelo no outro eixo. Essa visualização permite identificar visualmente qual o tipo de erro o modelo comete mais frequentemente.

Também utilizamos a Curva Característica do Operador Receptor (ROC) como uma métrica de avaliação do modelo. Essa curva é construída ao se plotar a Taxa de Verdadeiros Positivos (TPR) pela Taxa de Falsos Positivos (FPR) conforme se altera o valor de "corte"para a separação das classes pelo modelo. Comumente se utiliza a Área Sob a Curva ROC (AUC) como uma métrica de avaliação do modelo mais confiável que a acurácia.

4.2 Análise dos Transcriptomas de Aracnídeos

Os estudos do transcriptoma de aracnídeos foram conduzidos a partir da análise dos 38 transcriptomas de glândulas de veneno e salivas descritos na seção 2.3, numa tentativa de reduzir o escopo dos tipos de proteínas sendo analisadas para tecidos potencialmente mais enriquecidos para AMPs e AVPs.

Primeiramente realizamos a predição das proteínas geradas pelos transcritos obtidos do NCBI, com o software Transdecoder (Haas, Papanicolaou *et al.*, 2016). Em alguns poucos casos as proteínas também foram depositadas no NCBI para os transcriptomas correspondentes, porém a maior parte desses proteomas foi omitida ou apresentado um conjunto reduzido de proteínas, nos casos em que a quantidade de proteínas reportadas fossem menor que 70% da quantidade de transcritos, realizamos a predição das proteínas *in silico*.

Ao final desse processo obtivemos as proteínas preditas a partir dos 38 transcriptomas, pertencentes a 29 espécies diferentes de aracnídeos. Utilizamos 3 das ferramentas citadas anteriormente (Primeira Fase do Nosso Preditor, AntiVPP e Meta-iAVP) para identificar potenciais AVPs nos peptídeos resultantes do processo descrito na seção 2.3. Para todos os preditores, um valor maior ou igual à 0.5 é considerado uma predição positiva, ou seja, que o peptídeo possui atividade antiviral.

A tabela 4.1 mostra quantos peptídeos foram identificados como AVPs por cada preditor, e a Figura 4.1 revela quantos foram preditos por mais de uma ferramenta. Podemos ver que a nossa ferramenta sugere um número consideravelmente menor de peptídeos como potenciais AVPs. Esse conjunto de 2335 peptídeos (1.6% dos peptídeos de aracnídeos) identificado pelas 3 ferramentas é o conjunto de candidatos a AVPs que deve ser considerado para estudos futuros.

Podemos observar que nosso modelo identifica menos peptídeos como PAVs. Devido a raridade desses compostos, essa é na verdade uma métrica positiva, que possivelmente indica que o modelo comete menos erros do tipo "Falso Positivo", porém mais testes são necessários.

Preditor	Peptídeos Identificados como AVPs			
	H. spadix **(4996)**	***P. nigriventer*** **(27747)**	***H. dromedarii*** **(106167)**	**Total (138910)**
AntiVPP	1782 (36%)	9260 (33%)	33862 (32%)	**44904 (32%)**
Meta-iAVP	1748 (35%)	9686 (35%)	34622 (33%)	**46056 (33%)**
Nossa Ferramenta	343 (7%)	2316 (8%)	10001 (9%)	**12660 (9%)**

Tabela 4.1: *Predições feitas pelas diferentes ferramentas a partir dos peptídeos provenientes dos transcriptomas. As porcentagens se referem ao total de peptídeos preditos como positivos em cada espécie (coluna). Em média 30% dos peptídoes foram preditos como bioativos antivirais.*

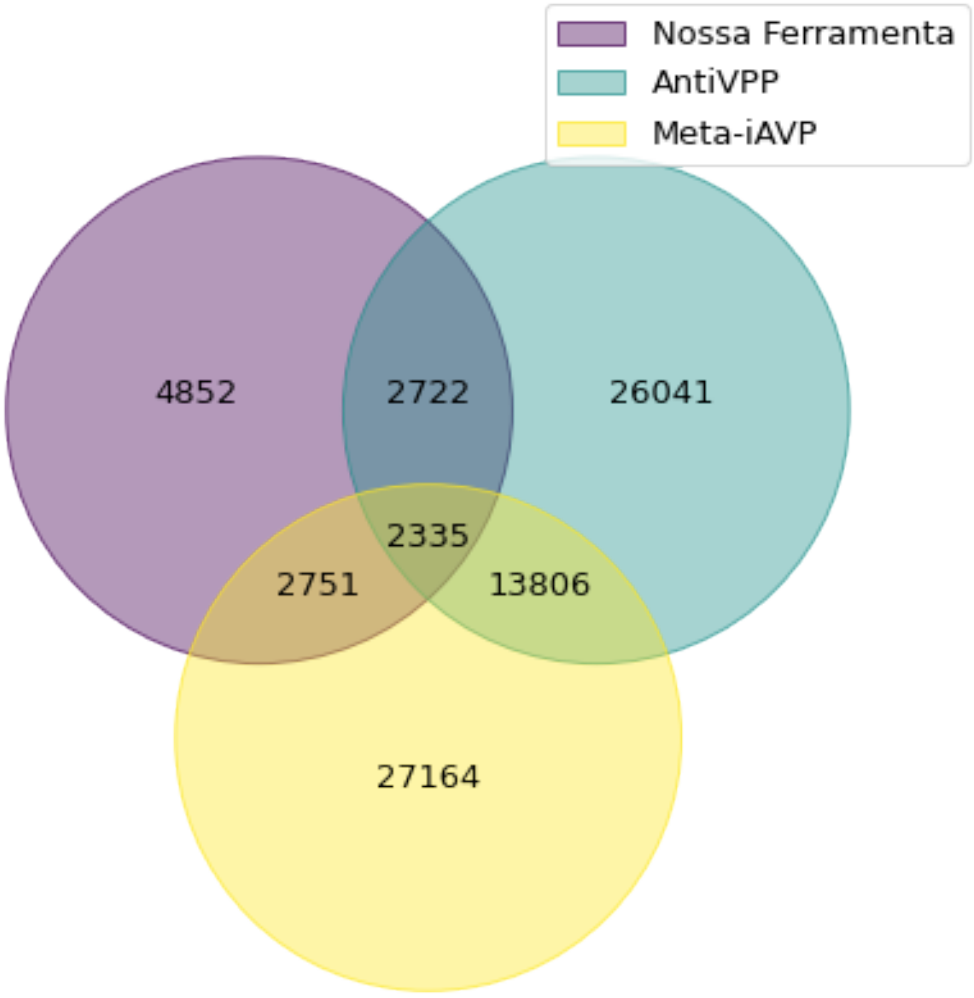

Figura 4.1: *Peptídeos identificados como AVPs por cada ferramenta.*

4.3 Experimentos com a Arquitetura Base da RNR

A primeira leva de experimentos teve como objetivo determinar a base da arquitetura da RNR que seria utilizada para a primeira fase preditor. Inicialmente foram implementados 4 modelos diferentes, cujas diferenças podem ser observadas na tabela 4.2.

Modelo	Tipo de RNR	Codificação dos AAs	Bidirecional
1	Linear	One-Hot	Não
2	LSTM	One-Hot	Não
3	LSTM	Embeddings	Não
4	LSTM	Embeddings	Sim

Tabela 4.2

Os testes foram realizados com os dois conjuntos de sequências de peptídeos, com os bancos de PAMs (9567 peptídeos de cada classe) e PAVs (1695 peptídeos de cada classe), construídos como detalhado na seção 2.2. Em ambos as partes de Treinamento e Teste também foram separadas como descrito nessa seção.

Cada modelo (nesse caso o modelo é somente a RNR) realizou o processo de avaliação 5 vezes. O processo consistia em: separação dos peptídeos nas partes de Treino e Teste aleatoriamente, treinamento no modelo na parte de Treino por uma *epoch*, e avaliação das predições na parte de Teste.

Os resultados podem ser vistos nas Tabelas 4.3 e 4.4. Para cada arquitetura foi selecionado o modelo com os valores de parâmetros que forneceram o melhor resultado.

Modelo	SENS	SPEC	ACC	MCC
1*	0.0	1	0.5	-
2*	0.0	1	0.5	-
3	0.68±0.3	0.86±0.3	0.77±0.1	0.61±0.17
4	**0.84±0.2**	**0.87±0.2**	**0.85±0.1**	**0.74±0.1**

Tabela 4.3: *Predição de AVPs*

Modelo	SENS	SPEC	ACC	MCC
1	0.0	1	0.5	-
2	0.0	1	0.5	-
3	0.96±0.04	0.95±0.05	**0.96±0.01**	0.92±0.03
4	**0.97±0.03**	0.95±0.05	0.96±0.02	0.92±0.03

Tabela 4.4: *Predição de AMPs*

Para os modelos 1 e 2, para todas as combinações de parâmetros o modelo treinado classificava todos os peptídeos da parte de teste como negativos ou todos como positivos. Nesse caso a média dos valores não seria uma ilustração correta dos resultados, então representamos o caso de classificação de todos os peptídeos como negativos nas tabelas 4.3 e 4.4.

A partir desses resultados, concluímos que a representação dos aminoácidos utilizando *One-Hot encoding* se mostrou menos útil para os modelos, visto que os modelos 1 e 2 não foram capazes de separar as classes de peptídeos.

Dentre os modelos que utilizavam a representação de *embeddings*, podemos ver que a variação bidirecional (modelo 4) obteve resultados melhores em todas as métricas no banco de PAVs, e equivalente em 3 das 4 no banco de PAMs. Baseados nesses resultados adotamos uma arquitetura de RNR-LSTM bidirecional com representações de *embeddings* como base para os próximos testes.

4.4 Representação das sequências de aminoácidos

Após testes da seção 4.3, concluímos a partir dos resultados obtidos que o modelo de RNR-LSTM Bidirecional seria o mais apropriado, o modelo utilizado no preditor final sendo composto por uma camada de *embedding* no início como uma arquitetura base para novas modificações.

Inicialmente, adotamos a "unidade básica"da sequência dos peptídeos como um único aminoácido. Modelos de Linguagem Natural comumente são treinados tanto utilizando palavras inteiras como caracteres individuais como a menor unidade para a rede, visto que essa abordagem permite que todos os níveis de estruturas entre os dados sejam capturadas pela rede. A partir do mesmo princípio criamos variantes do modelo ampliando sua capacidade, permitindo a utilização de dois e três aminoácidos como unidade básica para a rede, como descrito na seção 3.1.

Com as taxas de acurácia e eficácia obtidas com os resultados da seção 4.3 decidimos por ampliar o tempo de treinamento do modelo, então realizamos alguns testes aumentando o número de *epochs* (quantidade de vezes que o processo de treinamento do modelo é realizado). O número de *epochs* foi aumentado para 20 ao invés de apenas uma única vez, de modo a captar um espectro amplo de resultados.

Para avaliar os bancos de treinamento, avaliamos o impacto da seleção de peptídeos para criação do banco negativo na performance do modelo. Pois há uma diferença significativa da quantidade de peptídeos nos bancos negativos (9.567 para o banco de PAMs, 1.695 para o banco de PAVs) quando comparado com a quantidade de peptídeos sem atividade após a busca por homologias (458.444 peptídeos sem atividade).

Deste modo, foram realizados 5 sorteios diferentes para os peptídeos negativos, utilizando o mesmo processo descrito na seção 2.2, com conjuntos de tamanhos similares ao banco positivo e realizamos os testes e análise da precisão e acurácia para modelos treinados em cada um desses conjuntos individualmente e ao final calculando uma precisão aproximada relativa. Esse processo foi feito considerando apenas o banco de PAVs como classe positiva, então cada um desses 5 conjuntos de peptídeos negativos contém 1.695 peptídeos sem atividade.

Podemos ver a interseção entre esses conjuntos negativos na Figura 4.2, e observar que a quantidade de sequencias em comum é desprezível, possibilitando o teste do impacto dessas sequências na performance do modelo.

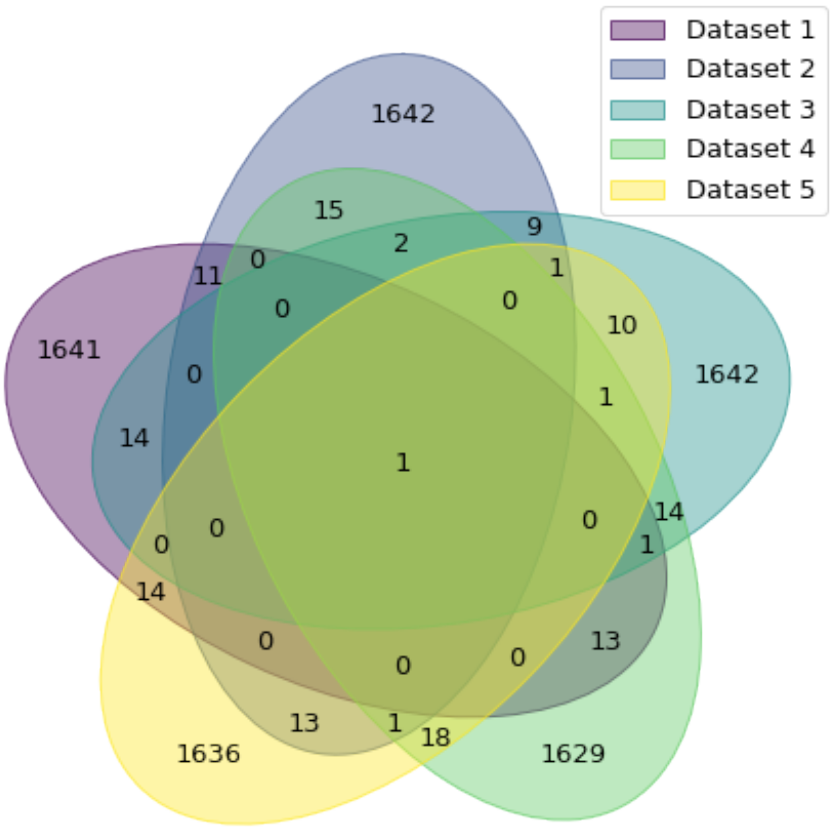

Figura 4.2: *Peptídeos em Comum entre os conjuntos negativos*

Essas três características - aumento do número de *epochs*, utilização de n-gramas como unidade básica da rede e utilização de diferentes sorteios do banco negativo foram testadas em consonância.

Os testes foram feitos da seguinte maneira. Para cada banco de dados, composto pelo conjunto de PAVs e por um dos conjuntos negativos sorteados, as sequências foram divididas em partes de treinamento e de teste como anteriormente (70% e 30% das sequências totais respectivamente). A cada *epoch* o modelo era treinado em todas as sequências da parte de treinamento, e ao final era avaliado em todas as sequências da parte de teste. Para cada conjunto esse processo era repetido por 20 epochs, 3 vezes.

Esses testes foram realizados com 3 modelos diferentes, todos se utilizando de uma arquitetura com uma camada de *embedding*, 5 camadas LSTM bidirecionais, uma camada linear e uma de *softmax* ao final. A diferença entre os modelos está na 'unidade básica', como discutida na seção 3.1, que eram n-gramas de 1, 2 e 3 aminoácidos respectivamente. Esse valor é chamado de 'Ngram' nos gráficos.

Os resultados dos testes com os diferentes bancos e variações estão representados nas Figuras 4.3 e 4.4. Nos gráficos cada linha representa a média do valor daquela variável nos 3 ciclos de um modelo em um conjunto. As linhas tem cores similares a depender do modelo testado.

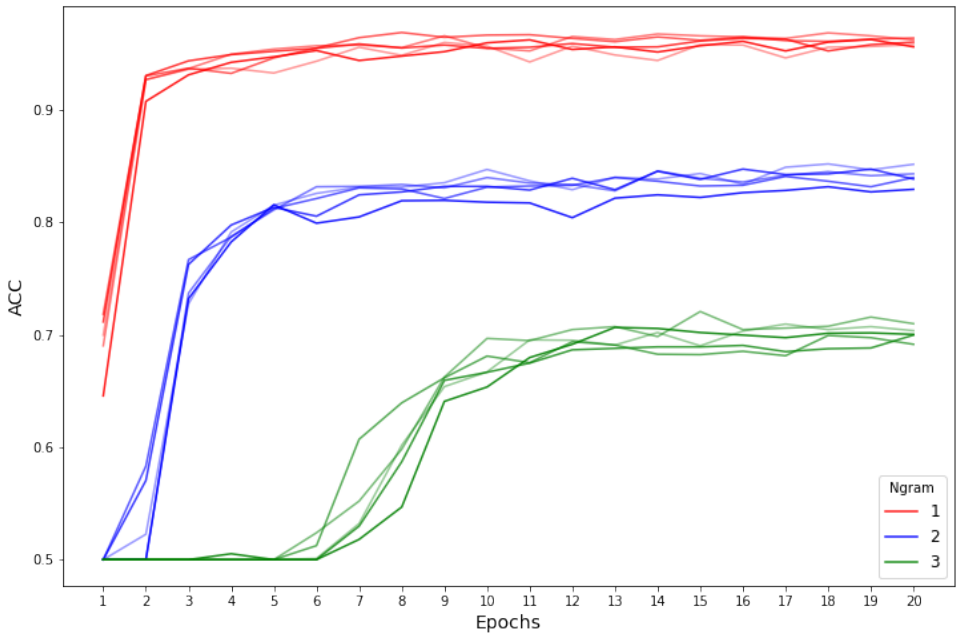

Figura 4.3: *ACC por epoch de treinamento dos modelos testados*

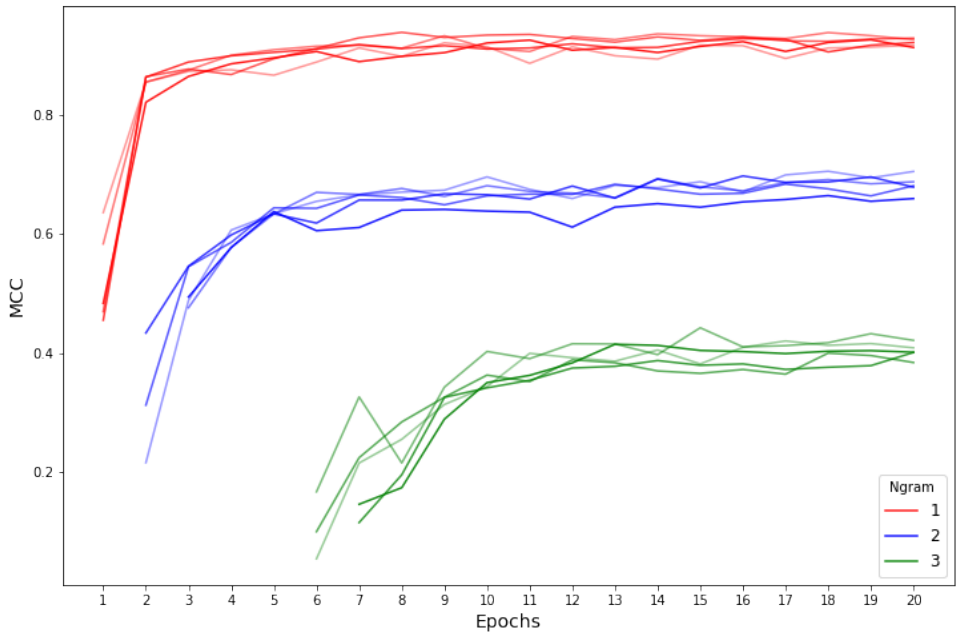

Figura 4.4: *MCC por epoch de treinamento dos modelos testados*

Na Figura 4.4 algumas linhas não possuem valores nas primeiras *epochs*, isso se dá pois os resultados dos modelos nesses pontos em especifico não permitiam calcular o MCC. Isso ocorria pois os modelos no estágio inicial de treinamento poderiam classificar todas as sequências de testes como positivas ou como negativas, o que tornaria o denominador em 4.1 = 0, suprimindo, deste modo, esses pontos iniciais.

Podemos concluir por esses testes que a utilização de n-gramas maiores que um aminoácido como unidade básica da rede não proporcionou um aumento na performance, pelo contrário, acabou por reduzir em aproximadamente 10% a acurácia para cada aminoácido adicionado.

Esse efeito pode ser potencialmente explicado pela quantidade pequena de vezes que cada combinação de mais aminoácidos está presente no banco de dados, que pode ser visualizada nas (Figuras 2.6 e 2.7) quando comparados com a composição de aminoácidos (Figura 2.4). Ressaltamos as diferenças nas escalas do eixo vertical dos 3 gráficos.

A "escassez"das unidades básicas dificulta o aprendizado da rede, o que resulta em poucos exemplos das combinações entre esses n-gramas, dificultando a internalização de padrões relevantes. Os experimentos realizados para estudo da representação das sequências sugerem que esse problema não é apenas uma questão de um período curto de treinamento, pois podemos observar que não há variação visível nos resultados dos modelos de bigramas e trigramas nas figuras 4.3 e 4.4 após as primeiras *epochs*.

A utilização de várias *epochs* de treinamento no entanto foi extremamente benéfica, podemos ver um ganho tremendo de performance em todos os modelos, principalmente nas primeiras iterações. Após esse primeiro crescimento no entanto, temos uma estabilização do desempenho.

Os resultados também nos mostraram que o sorteio de peptídeos negativos não possui um impacto significativos nos resultados obtidos, visto a proximidade dos resultados de cada arquitetura em diferentes conjuntos de dados.

4.5 Seleção dos Hiperparâmetros e Features

Como descrito na seção 3.2.1, a arquitetura utilizada para a primeira fase do modelo é composta por dois módulos principais, os quais também são modelos preditivos por si só. Cada um desses modelos pode ser entendido como uma família de modelos, visto que os seus parâmetros internos não treináveis podem assumir uma variedade de valores possíveis. Esses parâmetros são chamados de hiperparâmetros, e seu valor tem um impacto profundo no desempenho do modelo. Podemos considerar que a escolha do conjunto de features a serem utilizadas também é um hiperparâmetro do modelo, o que permite analisar essa escolha com as mesmas ferramentas para essa classe de problemas de optimização.

Investigamos os hiperparâmetros indicados na lista abaixo, limitando os valores testados aos intervalos ou conjuntos de valores indicados:

- Parâmetros do Módulo LSTM
  - **Camadas LSTM**: Número de Camadas Bidirecionais LSTM utilizadas em sequência. Foram testados valores de 1 a 7.

  - **Tamanho do Vetor de *Embedding***: Quantidade de valores do vetor de *Embedding* que representa cada amino ácido. Foram testados os valores: 64, 128, 256, 512.
  - **Tamanho das *Hidden Layers***: Quantidade de valores dos vetores internos das camadas de LSTM. Foram testados os valores: 64, 128, 256, 512.
  - ***Dropout***: Quantos valores são aleatoriamente descartados entre as camadas LSTM intermediárias durante o treinamento. Foram testados valores no intervalo $[0.1, 0.5]$.
  - **Optimizador**: Tipo de optimizador utilizado no treinamento do módulo LSTM. Foram testados os optimizadores: RMSprop, Adam e SGD, todos da biblioteca *pytorch*.
  - **Taxa de Aprendizagem Inicial**: Parâmetro inicial fornecido ao optimizador. Foram testados valores no intervalo $[10^{-5}, 0.1]$.

- Parâmetros do Módulo RF
  - **Número de Estimadores**: Número de árvores de decisão simples que foram utilizadas. Foram testados valores entre 50 e 150.
  - **Profundidade Máxima**: A profundidade se refere a quantas árvores de decisão são aplicadas conjuntamente, separando o espaço de features em mais regiões. Foram testados os valores 1, 2 e 3.
  - **Conjunto de Features**: Valores que são repassados para o Módulo RF, foram descritos de maneira detalhada na seção 3.1.1. Foram testados os conjuntos: PCP, AAC, CTD, PseudoAAC e um conjunto "controle", no qual consideramos apenas as predições do Módulo LSTM (indicado como "Pure LSTM"nos resultados).

- Parâmetro Geral da Fase do Modelo:
  - **Peso**: Valor ω da equação 3.4, pode ser interpretado como a prioridade que o modelo dá ao Módulo LSTM em relação ao Módulo RF. Foram testados os valores: 0.3 , 0.4, 0.5, 0.6, e 0.7.

Quando o Conjunto de *Features* PseudoAAC é selecionado, temos mais dois hiperparâmetros para o módulo de RF: λ e w, presentes na equação 3.3, que assumiram valores de 1 a 20 e $[0.1, 0.3]$ repectivamente.

O processo para seleção do Hiperparâmetros foi feito da seguinte forma: para cada um dos 5 conjuntos de *features* possíveis, realizamos 100 Tentativas ou *Trials*. Uma tentativa consiste dos seguintes passos: Cada hiperparâmetro é amostrado do conjunto de posíveis valores através da biblioteca *optuna* (Akiba *et al.*, 2019), que optimiza a seleção de valores para os parâmetros de acordo com o resultado das tentativas anteriores. Com os parâmetros selecionados, treinamos o modelo por uma *epoch* na parte de Treinamento dos dados, e avaliamos seus resultados na parte de Validação. Repetimos essa etapa de treinamento até a performance na parte de Validação piorar em relação com os resultados da última *epoch*.

Os resultados e parâmetros selecionados da melhor tentativa para cada conjunto de dados podem ser vistos na tabela 4.5. Um ponto a se destacar desses resultados é que nos dois bancos de dados o conjunto de *features* PCP resultou no melhor modelo, o que indica que fornecer as características fisicoquimicas do peptídeo de maneira mais direta contribuiu mais do que as tentativas de incorporar características de sequência também no módulo RF. Podemos também observar que os dois modelos utilizaram um ω de 0.7, indicando que o Módulo RF deve ocupar um papel secundário nas predições.

Parâmetro	**Preditor Final (AVP)**	**Preditor Final (AMP)**
Camadas LSTM	7	6
Tamanho do Vetor de *Embedding*	512	256
Tamanho das *Hidden Layers*	128	64
Dropout	0.11	0.13
Optimizador	RMSprop	RMSprop
Taxa de Aprendizagem Inicial	0.0005	0.0011
Número de Estimadores	50	96
Profundidade Máxima	2	2
Conjunto de Features	PCP	PCP
Peso (ω)	0.7	0.7
Epochs de Treinamento	9	3
ACC Final	0.978431	0.98675
Perda Final	32.6898	87.8276

Tabela 4.5: *Hiperparâmetros otimizados da primeira fase do modelo para classificação de AVPs e AMPs*

Para analizar melhor o impacto isolado do conjunto de *features* na performance do modelo, construímos os gráficos na figura 4.5. Eles mostram a Acurácia (ACC) de cada Tentativa na *epoch* antes do treinamento ser finalizado, separados em um *boxplot* por conjunto de *features*.

Esse gráfico nos mostra que, embora exista algum impacto na escolha do conjunto de *features*, esse impacto é minúsculo se comparado ao impacto da seleção de diferentes valores para os hiperparâmetros.

4.6 Avaliação da Primeira Fase do Modelo

Nos próximos passos utilizaremos o melhor modelo encontrado para cada conjunto de dados, ou seja, a primeira fase do modelo treinada com os hiperparametros elencados na tabela 4.5. Cada modelo foi treinado por 10 *epochs* em um banco de dados composto das partes de Treinamento e Validação (no total 85% dos dados), e avaliamos suas predições nas partes de Teste.

Temos que na tarefa de predição de AVPs, o modelo com os hiperparâmetros definidos na tabela 4.5 obteve 95% de Acurácia, e uma Área sob a curva ROC de 0.977 . A curva ROC do preditor pode ser vista na figura 4.6, enquanto na figura 4.7 podemos ver a matriz de confusão das predições.

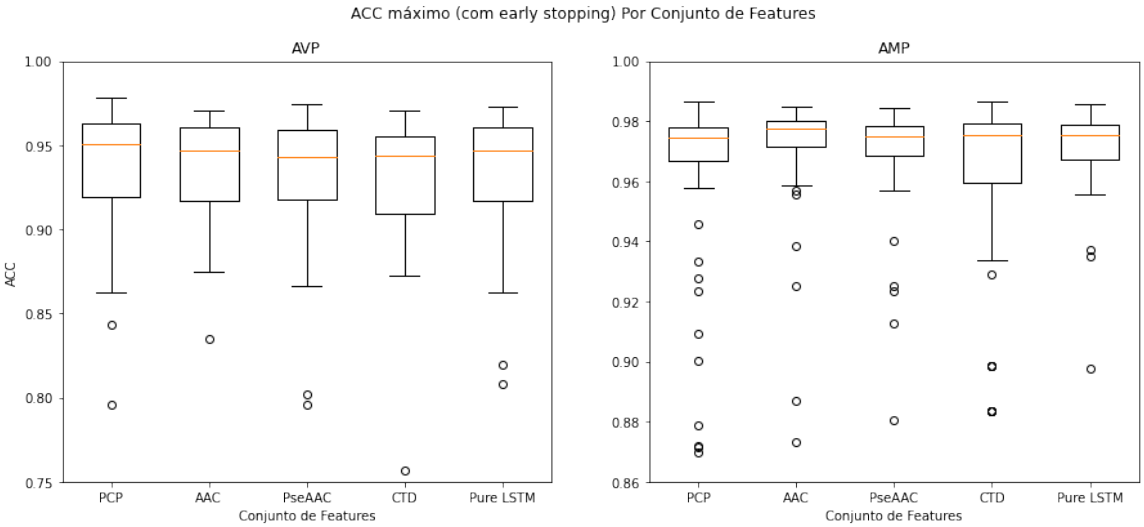

Figura 4.5: *Valor máximo de ACC atingido por cada modelo com o conjunto de features utilizando o método early stopping de treinamento nos bancos de AVPs e AMPs respectivamente.*

Na tarefa de predição de AMPs, o modelo com os hiperparâmetros definidos na tabela 4.5 obteve 92% de Acurácia, e uma Área sob a curva ROC de 0.976 . A curva ROC do preditor pode ser vista na figura 4.8, enquanto na figura 4.9 podemos ver a matriz de confusão das predições.

4.7 Avaliação da Segunda Fase do Modelo

Para a segunda fase do modelo, avaliamos apenas o banco de PAVs classificados previamente, descritos na seção 2.1. Abaixo podemos ver os resultados.

Classe	PREC	SENS	F1	ACC	Suporte
Membrana	**0.91**	0.91	**0.91**	0.89	200
Replicação	0.87	0.86	0.86	0.89	138
Montagem	0.73	**0.92**	0.81	**0.98**	12

Tabela 4.6: *Valores de Precisão, Sensibilidade, F1 score e o número de peptídeos no conjunto de teste (suporte) para cada classe.*

O modelo para a classe Membrana teve os maiores valores de Precisão (0.91) F1 (0.91) no conjunto de teste. E o modelo para classe de Montagem teve o maior valor de Recall (0.92) e ACC (0.98), porém como tem somente 12 exemplos da classe positiva, a métrica de acurácia não reflete a verdadeira performance do modelo, sendo a Precisão e o Recall medidas mais confiáveis. Os modelos para as três classes se mostraram eficientes para classificar os peptídeos antivirais, com todas as métricas maiores do que 0.85.

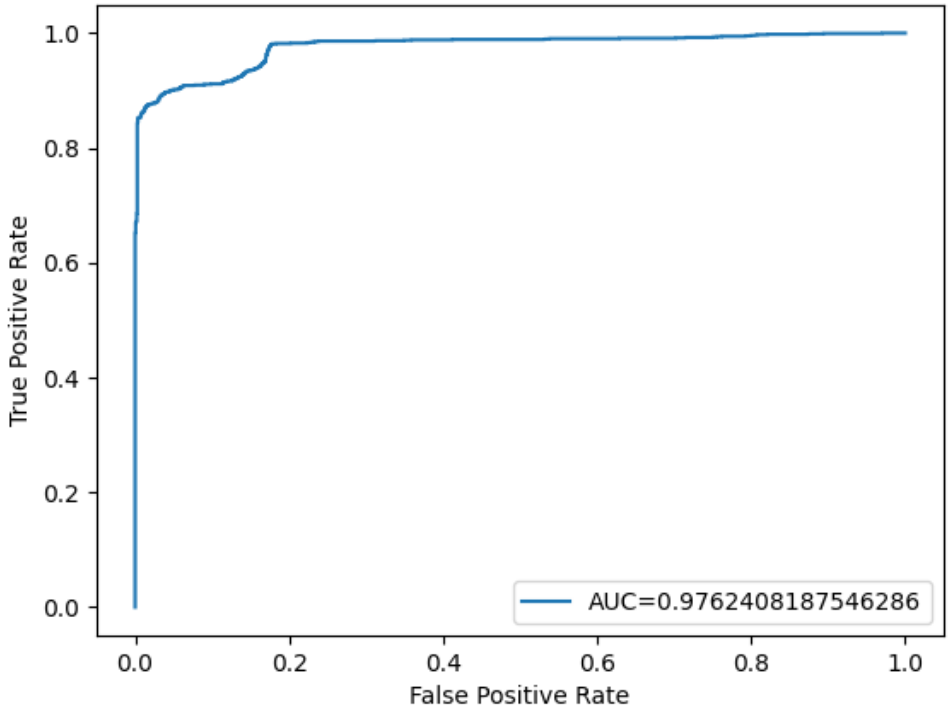

Figura 4.6: *Curva ROC na Predição de AVPs, utilizando a parte de Validação*

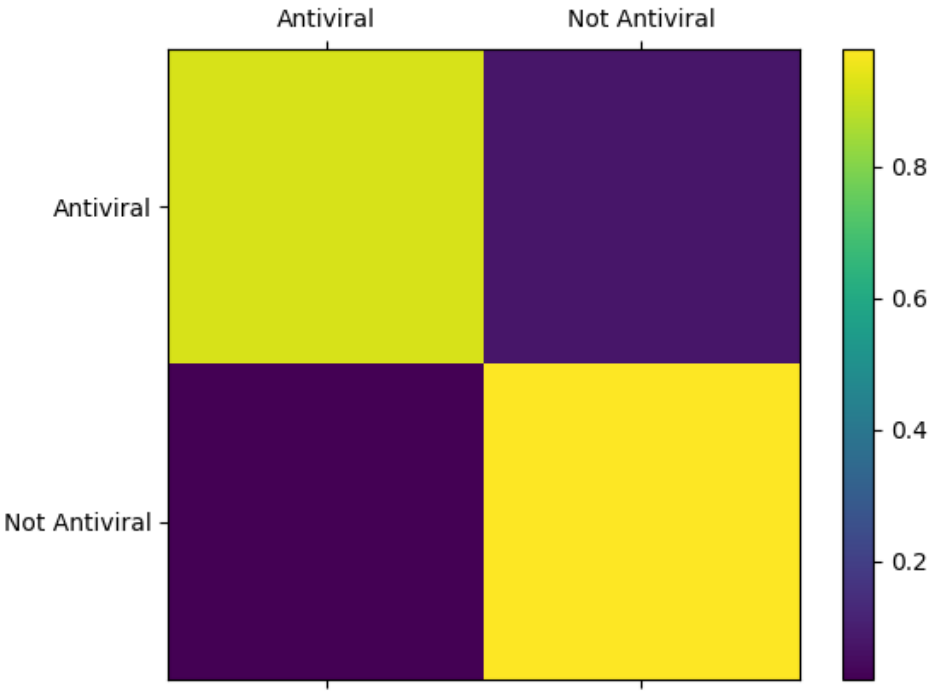

Figura 4.7: *Matriz de Confusão (AVPs)*

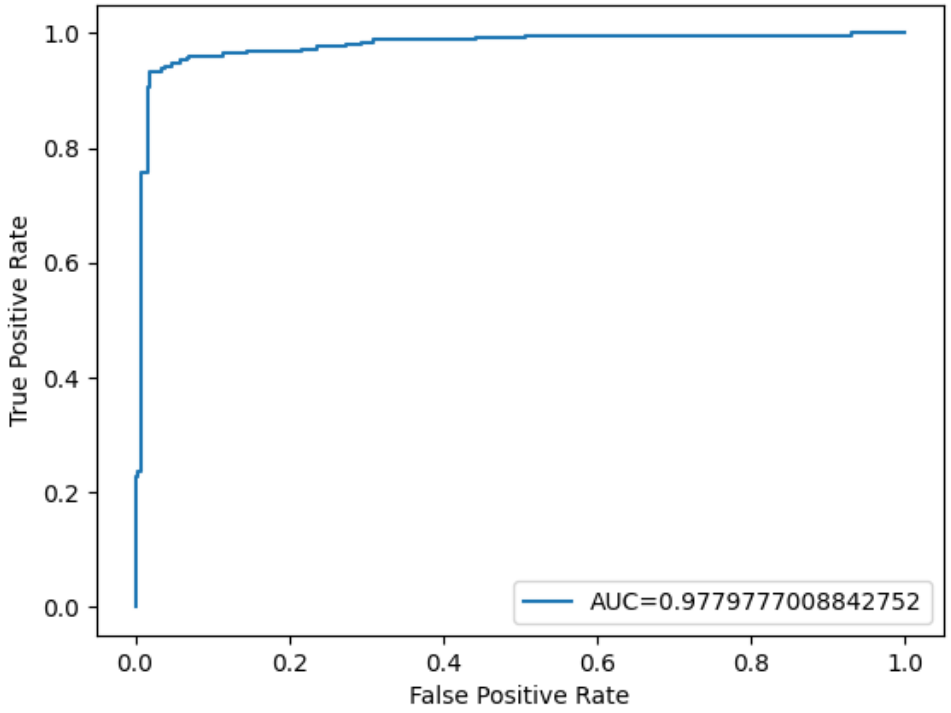

Figura 4.8: *Curva ROC na Predição de AMPs, utilizando a parte de Validação*

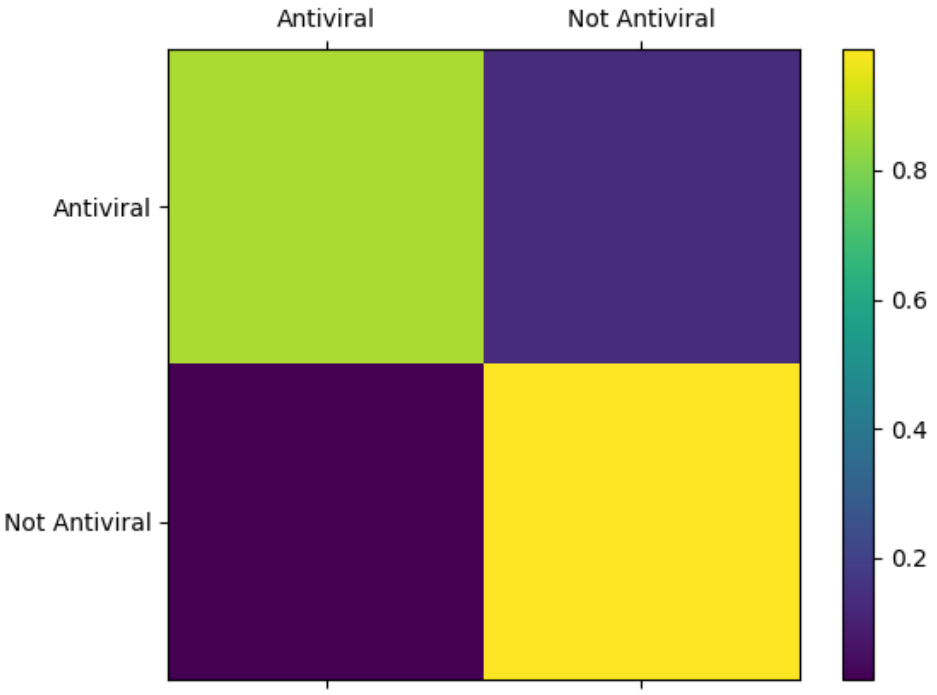

Figura 4.9: *Matriz de Confusão (AMPs)*

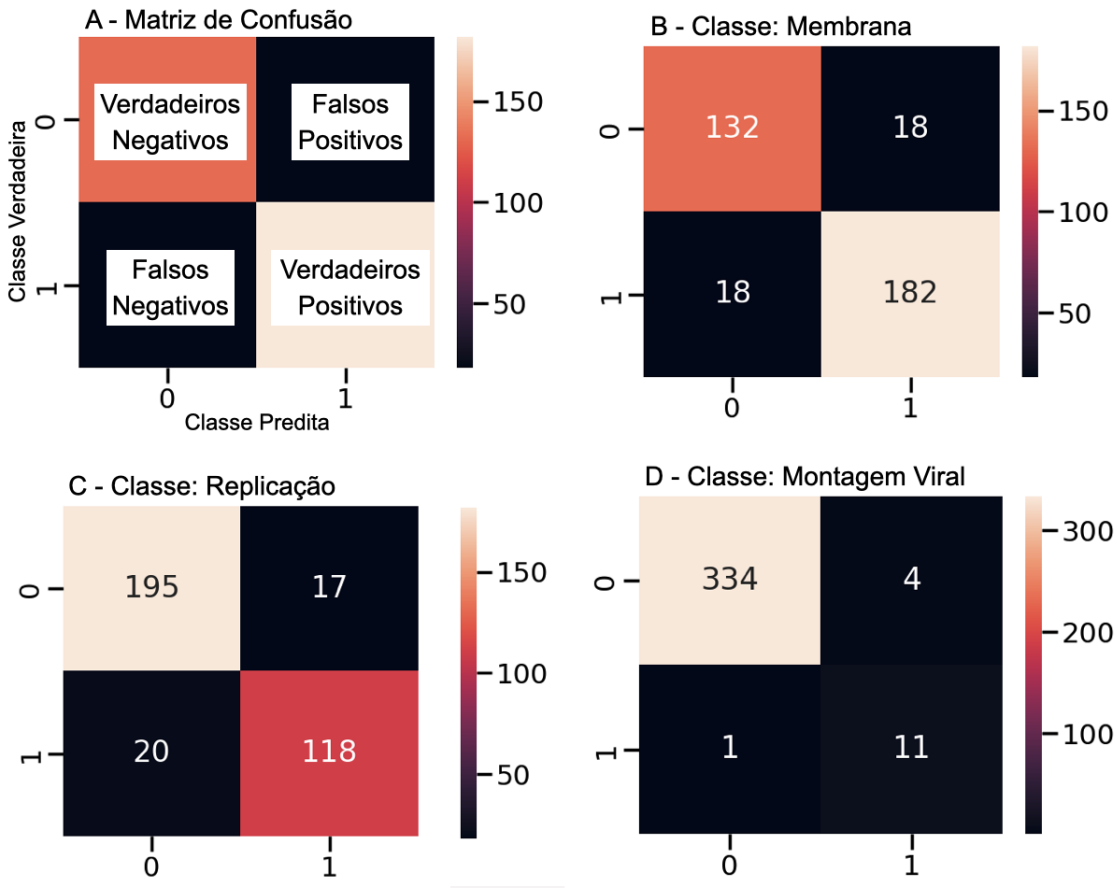

Figura 4.10: *Matrizes de Confusão para as três classes de mecanismo de ação dos PAVs. (A) Exemplo da matriz de confusão; Matrizes de confusão para classe membrana (B), Replicação (C) e Montagem viral (D).*

4.8 Comparação com Outras Ferramentas

Foram comparados os modelos existentes para predição de PAVs: DeepAVP, AntiVPP, MetaiAVP e AVPIden segundo a sensibilidade, a especificidade, acurácia e o valor de coeficiente de correlação de Matthews (Tabela 4.7). Os modelos foram treinados com o banco de (Thakur *et al.*, 2012) (Thakur) ou pelo conjunto de dados descrito anteriormente (Local), o qual foi utilizado por apenas dois preditores, pois os demais não permitiam um novo treinamento do modelo, apenas predição. Todos eles foram treinados conforme o método *4-fold* (Tabela 4.7).

Podemos observar que a primeira vista nosso modelo possui uma performance superior aos dois métodos disponíveis, destacando-se a baixíssima taxa de classificações incorretas nos Não AVPs, indicando que o modelo tem menor propensão para erros do tipo Falso Positivo. Esse tipo de erro é perigoso pois, pensando na aplicação da ferramenta como um filtro para identificação de compostos que serão analisados mais a fundo, cada composto sem ação classificadao erroneamente pode ocasionar um desperdício de recursos e/ou tempo.

Modelo	Treinamento	SubModelo	SENS	SPEC	ACC	MCC
DeepAVP	Thakur	-	0.967	0.9	0.933	0.87
AntiVPP	Thakur	-	0.87	0.97	0.93	0.87
Meta-iAVP	Thakur	-	0.917	**0.983**	0.949	0.9
AVPpred	Thakur	AVPphysico	0.933	0.917	0.925	0.85
Chang et al	Thakur	RFcompo	0.933	0.933	0.933	0.87
Chang et al	Thakur	RFcompo + agg	0.917	0.95	0.933	0.87
Chang et al	Thakur	RFphysico + structure + agg	0.95	0.867	0.908	0.82
EnAVPClass	Thakur	PseAAC+PCP	0.963	0.892	0.925	0.852
EnAVPClass	Local	PseAAC+PCP	**0.9711 ± 0.08**	0.9472 ± 0.22	**0.9584 ± 0.1**	**0.9175 ± 0.19**
AVPIden	Local	First-Stage	0.9243	0.9127	0.9185	-

Tabela 4.7: *Modelo, Conjunto de treinamento, Submodelo, Sensibilidade (SENS), Especificidade (SPEC), Acurácia (ACC) e o valor de coeficiente de correlação de Matthews (MCC) dos preditores existentes de PAVs.*

Para avaliar a segunda fase, como não existem, ao conhecimento do autor, modelos publicados que realizam o mesmo tipo de classificação por tipo de atividade, não comparamos a segunda fase com nenhum modelo estabelecido.

Capítulo 5

Conclusões

Nesse trabalho desenvolvemos um modelo com uma arquitetura inovadora (EnAVP-Class), sendo o primeiro preditor e classificador de Peptídeos Antivirais (PAVs) com foco no mecanismo de ação. O EnAVPClass é um dos primeiros a utilizar uma abordagem de Aprendizagem *Ensemble*, baseado em modelos de Aprendizagem Profunda (*Long Short Term Memory*) e Aprendizagem de Máquina Clássico (*Random Forest* e *Support Vector Machines*), que permite a combinação das vantagens de diversos modelos de aprendizado. Com isso atinigimos uma performance equivalente ou superior ao estado da arte na identificação de Peptídeos Antivirais (Tabela 4.7)), fornecendo assim uma ferramenta para futuros trabalhos nas áreas de Toxinologia, Farmacologia e Descoberta de novas Drogas.

Referências

[Akiba *et al.* 2019] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta e Masanori Koyama. "Optuna: a next-generation hyperparameter optimization framework". Em: *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019 (citado na pg. 37).

[Asgari e Mofrad 2015] Ehsaneddin Asgari e Mohammad RK Mofrad. "Continuous distributed representation of biological sequences for deep proteomics and genomics". Em: *PloS one* 10.11 (2015), e0141287 (citado na pg. 22).

[Bahar e Ren 2013] Ali Adem Bahar e Dacheng Ren. "Antimicrobial peptides". Em: *Pharmaceuticals* 6.12 (2013), pgs. 1543–1575 (citado na pg. 2).

[Belgiu e Drăguţ 2016] Mariana Belgiu e Lucian Drăguţ. "Random forest in remote sensing: a review of applications and future directions". Em: *ISPRS journal of photogrammetry and remote sensing* 114 (2016), pgs. 24–31 (citado na pg. 3).

[Bottou 2010] Léon Bottou. "Large-scale machine learning with stochastic gradient descent". Em: *Proceedings of COMPSTAT'2010*. Ed. por Yves Lechevallier e Gilbert Saporta. Heidelberg: Physica-Verlag HD, 2010, pgs. 177–186. isbn: 978-3-7908-2604-3 (citado na pg. 8).

[Calvete *et al.* 2009] Juan J Calvete, Libia Sanz, Yamileth Angulo, Bruno Lomonte e José María Gutiérrez. "Venoms, venomics, antivenomics". Em: *FEBS letters* 583.11 (2009), pgs. 1736–1743 (citado nas pgs. 1, 2).

[Cao *et al.* 2020] Yue Cao, Thomas Andrew Geddes, Jean Yee Hwa Yang e Pengyi Yang. "Ensemble deep learning in bioinformatics". Em: *Nature Machine Intelligence* 2.9 (2020), pgs. 500–508 (citado na pg. 7).

[Chawla *et al.* 2002] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall e W Philip Kegelmeyer. "Smote: synthetic minority over-sampling technique". Em: *Journal of artificial intelligence research* 16 (2002), pgs. 321–357 (citado na pg. 27).

[Cho, Van Merriënboer, Bahdanau *et al.* 2014] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau e Yoshua Bengio. "On the properties of neural machine translation: encoder-decoder approaches". Em: *arXiv preprint arXiv:1409.1259* (2014) (citado nas pgs. 4, 5).

[Cho, Van Merriënboer, Gulcehre *et al.* 2014] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre *et al.* "Learning phrase representations using rnn encoder-decoder for statistical machine translation". Em: *arXiv preprint arXiv:1406.1078* (2014) (citado na pg. 4).

[Chou 2001] Kuo-Chen Chou. "Prediction of protein cellular attributes using pseudo-amino acid composition". Em: *Proteins: Structure, Function, and Genetics* 43.3 (2001), pgs. 246–255. doi: 10.1002/prot.1035. url: https://doi.org/10.1002/prot.1035 (citado na pg. 23).

[Cortes e Vapnik 1995] Corinna Cortes e Vladimir Vapnik. "Support-vector networks". Em: *Machine learning* 20.3 (1995), pgs. 273–297 (citado na pg. 3).

[Dietterich 2000] Thomas G Dietterich. "Ensemble methods in machine learning". Em: *International workshop on multiple classifier systems*. Springer. 2000, pgs. 1–15 (citado na pg. 7).

[Dubchak, Muchnik, Holbrook *et al.* 1995] Inna Dubchak, Ilya Muchnik, S. R. Holbrook e Sung-Hou Kim. "Prediction of protein folding class using global description of amino acid sequence." Em: *Proceedings of the National Academy of Sciences* 92.19 (set. de 1995), pgs. 8700–8704. doi: 10.1073/pnas.92.19.8700. url: https://doi.org/10.1073/pnas.92.19.8700 (citado na pg. 24).

[Dubchak, Muchnik, Mayor *et al.* 1999] Inna Dubchak, Ilya Muchnik, Christopher Mayor, Igor Dralyuk e Sung-Hou Kim. "Recognition of a protein fold in the context of the SCOP classification". Em: *Proteins: Structure, Function, and Genetics* 35.4 (jun. de 1999), pgs. 401–407. doi: 10.1002/(sici)1097-0134(19990601)35:4<401::aid-prot3>3.0.co;2-k. url: https://doi.org/10.1002/(sici)1097-0134(19990601)35:4%3C401::aid-prot3%3E3.0.co;2-k (citado na pg. 24).

[Fan *et al.* 2016] Linlin Fan *et al.* "Dramp: a comprehensive data repository of antimicrobial peptides". Em: *Scientific reports* 6.1 (2016), pgs. 1–7 (citado nas pgs. 11, 12).

[Fernebro 2011] Jenny Fernebro. "Fighting bacterial infections—future treatment options". Em: *Drug Resistance Updates* 14.2 (2011), pgs. 125–139 (citado na pg. 2).

[Fry, Roelants *et al.* 2009] Bryan G Fry, Kim Roelants *et al.* "The toxicogenomic multiverse: convergent recruitment of proteins into animal venoms". Em: *Annual review of genomics and human genetics* 10 (2009), pgs. 483–511 (citado na pg. 2).

[Fry, Scheib *et al.* 2008] Bryan G Fry, Holger Scheib *et al.* "Evolution of an arsenal: structural and functional diversification of the venom system in the advanced snakes (caenophidia)". Em: *Molecular & Cellular Proteomics* 7.2 (2008), pgs. 215–246 (citado na pg. 1).

[Fu *et al.* 2012] Limin Fu, Beifang Niu, Zhengwei Zhu, Sitao Wu e Weizhong Li. "Cd-hit: accelerated for clustering the next-generation sequencing data". Em: *Bioinformatics* 28.23 (2012), pgs. 3150–3152 (citado na pg. 19).

[Fuente-Núñez *et al.* 2017] César de la Fuente-Núñez, Osmar N Silva, Timothy K Lu e Octavio Luiz Franco. "Antimicrobial peptides: role in human disease and potential as immunotherapies". Em: *Pharmacology & therapeutics* 178 (2017), pgs. 132–140 (citado na pg. 2).

[Goller e Kuchler 1996] Christoph Goller e Andreas Kuchler. "Learning task-dependent distributed representations by backpropagation through structure". Em: *Proceedings of International Conference on Neural Networks (ICNN'96)*. Vol. 1. IEEE. 1996, pgs. 347–352 (citado na pg. 4).

[Goodfellow, Bengio *et al.* 2016] Ian Goodfellow, Yoshua Bengio e Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016 (citado na pg. 3).

[Goodfellow, Pouget-Abadie *et al.* 2014] Ian Goodfellow, Jean Pouget-Abadie *et al.* "Generative adversarial nets". Em: *Advances in Neural Information Processing Systems 27*. Ed. por Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence e K. Q. Weinberger. Curran Associates, Inc., 2014, pgs. 2672–2680. url: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf (citado na pg. 4).

[Haas, Papanicolaou *et al.* 2016] B Haas, AJGS Papanicolaou *et al.* "Transdecoder (find coding regions within transcripts)". Em: *Google Scholar* (2016) (citado na pg. 30).

[Hammami *et al.* 2009] Riadh Hammami, Jeannette Ben Hamida, Gerard Vergoten e Ismail Fliss. "Phytamp: a database dedicated to antimicrobial plant peptides". Em: *Nucleic acids research* 37.suppl_1 (2009), pgs. D963–D968 (citado nas pgs. 11, 12).

[Hochreiter 1998] Sepp Hochreiter. "The vanishing gradient problem during learning recurrent neural nets and problem solutions". Em: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6.02 (1998), pgs. 107–116 (citado na pg. 5).

[Hochreiter e Schmidhuber 1997] Sepp Hochreiter e Jürgen Schmidhuber. "Long short-term memory". Em: *Neural computation* 9.8 (1997), pgs. 1735–1780 (citado nas pgs. 4, 5).

[Hopp e Woods 1981] T. P. Hopp e K. R. Woods. "Prediction of protein antigenic determinants from amino acid sequences." Em: *Proceedings of the National Academy of Sciences* 78.6 (jun. de 1981), pgs. 3824–3828. doi: 10.1073/pnas.78.6.3824. url: https://doi.org/10.1073/pnas.78.6.3824 (citado na pg. 24).

[Hsu *et al.* 2020] Li Yang Hsu, Po Ying Chia e JF Lim. "The novel coronavirus (sars-cov-2) pandemic". Em: *Ann als Academy of Medicine Singapore* 49.105 (2020), pgs. 105–107 (citado na pg. 3).

[Jenssen *et al.* 2006] Håvard Jenssen, Pamela Hamill e Robert EW Hancock. "Peptide antimicrobial agents". Em: *Clinical microbiology reviews* 19.3 (2006), pgs. 491–511 (citado na pg. 2).

[Ju *et al.* 2018] Cheng Ju, Aurélien Bibaut e Mark van der Laan. "The relative performance of ensemble methods with deep convolutional neural networks for image classification." Em: *Journal of Applied Statistics* 45.15 (2018) (citado na pg. 7).

[Kim e Tagkopoulos 2018] Minseung Kim e Ilias Tagkopoulos. "Data integration and predictive modeling methods for multi-omics datasets". Em: *Molecular omics* 14.1 (2018), pgs. 8–25 (citado na pg. 4).

[Kiros *et al.* 2015] Ryan Kiros *et al.* "Skip-thought vectors". Em: *Advances in Neural Information Processing Systems 28*. Ed. por C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama e R. Garnett. Curran Associates, Inc., 2015, pgs. 3294–3302. url: http://papers.nips.cc/paper/5950-skip-thought-vectors.pdf (citado na pg. 4).

[Lee *et al.* 2015] Hao-Ting Lee, Chen-Che Lee, Je-Ruei Yang, Jim ZC Lai e Kuan Y Chang. "A large-scale structural classification of antimicrobial peptides". Em: *BioMed research international* 2015 (2015) (citado nas pgs. 10, 12).

[Li *et al.* 2015] Jiwei Li, Minh-Thang Luong e Dan Jurafsky. "A hierarchical neural autoencoder for paragraphs and documents". Em: *arXiv preprint arXiv:1506.01057* (2015) (citado na pg. 4).

[Lissabet *et al.* 2019] Jorge Félix Beltrán Lissabet, Lisandra Herrera Belén e Jorge G. Farias. "AntiVPP 1.0: a portable tool for prediction of antiviral peptides". Em: *Computers in Biology and Medicine* 107 (abr. de 2019), pgs. 127–130. doi: 10.1016/j.compbiomed.2019.02.011. url: https://doi.org/10.1016/j.compbiomed.2019.02.011 (citado na pg. 23).

[Luong *et al.* 2015] Minh-Thang Luong, Hieu Pham e Christopher D Manning. "Effective approaches to attention-based neural machine translation". Em: *arXiv preprint arXiv:1508.04025* (2015) (citado na pg. 4).

[Mahadevappa *et al.* 2017] Ravikiran Mahadevappa, Rui Ma e Hang Fai Kwok. "Venom peptides: improving specificity in cancer therapy". Em: *Trends in cancer* 3.9 (2017), pgs. 611–614 (citado na pg. 2).

[Mata *et al.* 2017] Élida Cleyse Gomes da Mata, Caroline Barbosa Farias Mourão, Marisa Rangel e Elisabeth Ferroni Schwartz. "Antiviral activity of animal venom peptides and related compounds". Em: *Journal of Venomous Animals and Toxins including Tropical Diseases* 23.1 (2017), pg. 3 (citado na pg. 3).

[Matthews 1975] Brian W Matthews. "Comparison of the predicted and observed secondary structure of t4 phage lysozyme". Em: *Biochimica et Biophysica Acta (BBA)-Protein Structure* 405.2 (1975), pgs. 442–451 (citado na pg. 29).

[Melamud *et al.* 2016] Oren Melamud, Jacob Goldberger e Ido Dagan. "Context2vec: learning generic context embedding with bidirectional lstm". Em: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. 2016, pgs. 51–61 (citado na pg. 22).

[Min *et al.* 2017] Seonwoo Min, Byunghan Lee e Sungroh Yoon. "Deep learning in bioinformatics". Em: *Briefings in bioinformatics* 18.5 (2017), pgs. 851–869 (citado na pg. 3).

[Moreno 2015] José Moreno. "Prevotella copri and the microbial pathogenesis of rheumatoid arthritis". Em: *Reumatol Clin* 11.2 (2015), pgs. 61–3 (citado na pg. 2).

[Müller *et al.* 2018] Alex T Müller, Jan A Hiss e Gisbert Schneider. "Recurrent neural network model for constructive peptide design". Em: *Journal of chemical information and modeling* 58.2 (2018), pgs. 472–479 (citado na pg. 4).

[Nagarajan *et al.* 2018] Deepesh Nagarajan *et al.* "Computational antimicrobial peptide design and evaluation against multidrug-resistant clinical isolates of bacteria". Em: *Journal of Biological Chemistry* 293.10 (2018), pgs. 3492–3509 (citado na pg. 4).

[Pirtskhalava *et al.* 2016] Malak Pirtskhalava *et al.* "Dbaasp v. 2: an enhanced database of structure and antimicrobial/cytotoxic activity of natural and synthetic peptides". Em: *Nucleic acids research* 44.D1 (2016), pgs. D1104–D1112 (citado nas pgs. 11, 12).

[Platnick e Dupérré 2011] Norman I Platnick e Nadine Dupérré. "The andean goblin spiders of the new genus scaphidysderina (araneae, oonopidae), with notes on dysderina". Em: *American Museum Novitates* 2011.3712 (2011), pgs. 1–51 (citado na pg. 2).

[Qureshi *et al.* 2014] Abid Qureshi, Nishant Thakur, Himani Tandon e Manoj Kumar. "Avpdb: a database of experimentally validated antiviral peptides targeting medically important viruses". Em: *Nucleic acids research* 42.D1 (2014), pgs. D1147–D1153 (citado nas pgs. 10, 12).

[Sønderby *et al.* 2015] Søren Kaae Sønderby, Casper Kaae Sønderby, Henrik Nielsen e Ole Winther. "Convolutional lstm networks for subcellular localization of proteins". Em: *International Conference on Algorithms for Computational Biology*. Springer. 2015, pgs. 68–80 (citado na pg. 4).

[Sutskever *et al.* 2013] Ilya Sutskever, James Martens, George Dahl e Geoffrey Hinton. "On the importance of initialization and momentum in deep learning". Em: *International conference on machine learning*. PMLR. 2013, pgs. 1139–1147 (citado na pg. 8).

[Tanford 1962] Charles. Tanford. "Contribution of hydrophobic interactions to the stability of the globular conformation of proteins". Em: *Journal of the American Chemical Society* 84.22 (nov. de 1962), pgs. 4240–4247. doi: 10.1021/ja00881a009. url: https://doi.org/10.1021/ja00881a009 (citado na pg. 24).

[Thakur *et al.* 2012] Nishant Thakur, Abid Qureshi e Manoj Kumar. "Avppred: collection and prediction of highly effective antiviral peptides". Em: *Nucleic acids research* 40.W1 (2012), W199–W204 (citado na pg. 42).

[Torrent *et al.* 2011] Marc Torrent, David Andreu, Victòria M Nogués e Ester Boix. "Connecting peptide physicochemical and antimicrobial properties by a rational prediction model". Em: *PloS one* 6.2 (2011), e16968 (citado na pg. 13).

[UniProt 2021] Consortium UniProt. "Uniprot: the universal protein knowledgebase in 2021". Em: *Nucleic Acids Research* 49.D1 (2021), pgs. D480–D489 (citado na pg. 13).

[Veltri *et al.* 2018] Daniel Veltri, Uday Kamath e Amarda Shehu. "Deep learning improves antimicrobial peptide recognition". Em: *Bioinformatics* 34.16 (2018), pgs. 2740–2747 (citado nas pgs. 4, 13, 22).

[Ventola 2015] C Lee Ventola. "The antibiotic resistance crisis: part 1: causes and threats". Em: *Pharmacy and therapeutics* 40.4 (2015), pg. 277 (citado na pg. 2).

[Waghu *et al.* 2016] Faiza Hanif Waghu, Ram Shankar Barai, Pratima Gurung e Susan Idicula-Thomas. "Campr3: a database on sequences, structures and signatures of antimicrobial peptides". Em: *Nucleic acids research* 44.D1 (2016), pgs. D1094–D1097 (citado nas pgs. 10, 12).

[G. Wang *et al.* 2016] Guangshun Wang, Xia Li e Zhe Wang. "Apd3: the antimicrobial peptide database as a tool for research and education". Em: *Nucleic acids research* 44.D1 (2016), pgs. D1087–D1093 (citado nas pgs. 10, 12).

[Y. Wang *et al.* 2016] Yequan Wang, Minlie Huang, Xiaoyan Zhu e Li Zhao. "Attention-based lstm for aspect-level sentiment classification". Em: *Proceedings of the 2016 conference on empirical methods in natural language processing*. 2016, pgs. 606–615 (citado na pg. 22).

[Wheeler *et al.* 2007] David L Wheeler *et al.* "Database resources of the national center for biotechnology information". Em: *Nucleic acids research* 36.suppl_1 (2007), pgs. D13–D21 (citado na pg. 20).

[Xu *et al.* 2017] Zheng Xu, Sheng Wang, Feiyun Zhu e Junzhou Huang. "Seq2seq fingerprint: an unsupervised deep molecular embedding for drug discovery". Em: *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology,and Health Informatics*. ACM-BCB '17. Boston, Massachusetts, USA: Association for Computing Machinery, 2017, pgs. 285–294. ISBN: 9781450347228. DOI: 10.1145/3107411.3107424. URL: https://doi.org/10.1145/3107411.3107424 (citado na pg. 22).

[Zasloff 2002] Michael Zasloff. "Antimicrobial peptides of multicellular organisms". Em: *nature* 415.6870 (2002), pgs. 389–395 (citado na pg. 2).

[W. Zhang *et al.* 1990] Wei Zhang, Kazuyoshi Itoh, Jun Tanida e Yoshiki Ichioka. "Parallel distributed processing model with local space-invariant interconnections and its optical architecture". Em: *Applied optics* 29.32 (1990), pgs. 4790–4797 (citado na pg. 4).

[Z. Zhang 2018] Zijun Zhang. "Improved adam optimizer for deep neural networks". Em: *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. IEEE. 2018, pgs. 1–2 (citado na pg. 8).