Universidade Estadual de Campinas
Instituto de Computação

Amaury Mausbach Filho

Aplicação de redes neurais recorrentes ao projeto de anticorpos a partir da sequência de aminoácidos de epítopos

CAMPINAS
2021

Amaury Mausbach Filho

Aplicação de redes neurais recorrentes ao projeto de anticorpos a partir da sequência de aminoácidos de epítopos

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. João Meidanis

Este exemplar corresponde à versão final da Dissertação defendida por Amaury Mausbach Filho e orientada pelo Prof. Dr. João Meidanis.

CAMPINAS
2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

M448a

Mausbach Filho, Amaury, 1964-
Aplicação de redes neurais recorrentes ao projeto de anticorpos a partir da sequência de aminoácidos de epítopos / Amaury Mausbach Filho. – Campinas, SP : [s.n.], 2021.

Orientador: João Meidanis.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Redes neurais recorrentes. 2. Epítopos. 3. Paratopo. 4. Anticorpos - Projetos. I. Meidanis, João, 1960-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Application of recurrent neural networks to the design of antibodies from the amino acid sequence of epitopes
Palavras-chave em inglês:
Recurrent neural networks
Epitopes
Paratope
Antibodies - Design
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
João Meidanis [Orientador]
Hélio Pedrini
Pedro Otavio de Campos Lima
Data de defesa: 19-11-2021
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0003-3749-2371
- Currículo Lattes do autor: http://lattes.cnpq.br/8116732817948621

Universidade Estadual de Campinas
Instituto de Computação

Amaury Mausbach Filho

Aplicação de redes neurais recorrentes ao projeto de anticorpos a partir da sequência de aminoácidos de epítopos

Banca Examinadora:

- Prof. Dr. João Meidanis
  Instituto de Computação - Unicamp
- Prof. Dr. Hélio Pedrini
  Instituto de Computação - Unicamp
- Prof. Dr. Pedro Otavio de Campos Lima
  Centro de Pesquisa Boldrini

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 19 de novembro de 2021

Dedicatória

Ao meu pai,
por ter despertado minha curiosidade infindável, me ensinado a aprender, pelo incentivo aos estudos e por todo conhecimento de vida.

Agradecimentos

Agradeço à minha esposa Helena, e aos meus filhos Lucas e Paulo, que cederam muito do nosso tempo de convívio e também sempre me incentivaram a continuar. Agradeço especialmente à minha mãe, que sempre esteve ao meu lado e me incentivou nos estudos.

Agradeço muito ao meu orientador Meidanis, por ter me aceito e por toda dedicação e paciência nesta longa jornada. Um grande mestre!

Agradeço ao Dr. Pedro Lima, do Centro de Pesquisa Boldrini, criador da semente desta pesquisa. Que esta semente continue a se desenvolver e que, num futuro próximo, dê os frutos tão esperados.

Agradeço aos professores Célio Guimarães, Hélio Pedrini e Paulo Geus que, após muitos anos, me auxiliaram neste retorno à UNICAMP.

Agradeço ao Henrique, pela revisão e sugestões. Agradeço ao Guilherme e Diego, do Centro de Pesquisas Boldrini, por ampliar meus conhecimento em Imunologia.

Agradeço a todos que direta ou indiretamente me incentivaram e me apoiaram neste pesquisa.

Resumo

Anticorpos são elementos criados pelo sistema imunológico adaptativo dos mamíferos, a partir da identificação de um epítopo (segmento de uma parte externa de um antígeno) estranho ao corpo hospedeiro, com a finalidade de combate específico ao epítopo identificado. O desenho de anticorpos a partir de epítopos tem sido muito pesquisado no tratamento de certas doenças virais, e de alguns tipos de câncer que não respondem de maneira eficaz aos tratamentos convencionais. Nesta pesquisa, procuramos obter dados para treino e teste de preditores computacionais de anticorpos a partir de epítopos, e desenhamos uma arquitetura para avaliação. Assim, alguns bancos de dados públicos de epítopos e anticorpos foram avaliados. Foi possível verificar que a escassez de registros para treino e teste diminui a riqueza de informações. Métricas consideradas mais relevantes para análise de proteínas foram revistas e adaptadas para funcionamento em sequências com preenchimento (*padding*), necessárias no caso do treino por lote. Um modelo de aprendizado profundo, baseado em processamento de linguagem natural e utilizando Redes Neurais Recorrentes para a predição de CDRs(*Complementarity Determining Regions*), as principais partes dos anticorpos, a partir de epítopos foi implementado e avaliado. Apesar do bom desempenho nos treinos, experimentamos uma situação persistente de *overfitting* com baixa generalização, mesmo após a aplicação de técnicas como *weight decay*, *data augmentation* e *dropout*. Por fim, lançamos mão de *grid search*, o que permitiu uma pequena melhora dos resultados quando da aplicação de *weight decay*.

Abstract

Antibodies are elements created by the adaptive immune system of mammals, from the identification of an epitope (segment of an external part of an antigen) foreign to the human body, with the purpose of specifically combating the identified epitope. The design of antibodies from epitopes has been extensively researched in the treatment of certain viral diseases, and of some types of cancers that do not respond effectively to conventional treatments. Thus, some public databases of epitopes and antibodies were evaluated. It was possible to verify that the scarcity of records for training and testing reduces the wealth of information. Metrics considered most relevant for protein analysis were reviewed and adapted to function in padding sequences, necessary in the case of batch training. A deep learning model, based on natural language processing and using Recurrent Neural Networks to predict CDRs(*Complementarity Determining Regions*), the main parts of antibodies, from epitopes was implemented and evaluated. Despite the good performance in training, we experienced a persistent situation of overfitting with low generalization, even after applying techniques such as weight decay, data augmentation and dropout. Finally, we used grid search, which allowed a small improvement in results when applying weight decay.

Lista de Figuras

1.1 Estrutura de um anticorpo 19

3.1 Relacionamento entre as bases 30

4.1 Resultados de treino utilizando escore similaridade com banda limitada . . 47

5.1 Arquitetura do modelo proposto 52
5.2 Resultados do modelo para base `IEDB` e predição de **CDR** 53
5.3 Resultados do modelo para base `IEDB` e predição de **H3** 54
5.4 Resultados do modelo para base `IEDB` e predição de **L3** 54
5.5 Resultados do modelo para base `IEDB` e predição de **EPÍTOPO** 55
5.6 Resultados do modelo para base `IEDB2` e predição de **H3** 55
5.7 Resultados do modelo para base `IEDB3` e predição de **H3** 56
5.8 Resultados do modelo para base `IEDB4` e predição de **H3** 57
5.9 Resultados do modelo para base `IEDB5` e predição de **H3** 58
5.10 Resultados do modelo para base `IEDB6` e predição de **H3** 59
5.11 Resultados do modelo para base `IEDB7` e predição de **H3** 60
5.12 Regularização com parâmetro *weight decay* = 0.002 61
5.13 Comparativo da regularização com *data augmentation* 63
5.14 Comparativo da regularização com *data augmentation* 64
5.15 Resultado *grid search* - erro médio de teste 65
5.16 Resultado *grid search* - erro médio por resíduo no teste 66

A.1 Resultados do modelo para base `IEDB` e predição de **CDR** 73
A.2 Resultados do modelo para base `IEDB` e predição de **H3** 73
A.3 Resultados do modelo para base `IEDB` e predição de **L3** 74
A.4 Resultados do modelo para base `IEDB` e predição de **EPÍTOPO** 74
A.5 Resultados do modelo para base `IEDB2` e predição de **CDR** 74
A.6 Resultados do modelo para base `IEDB2` e predição de **H3**) 75
A.7 Resultados do modelo para base `IEDB2` e predição de **L3** 75
A.8 Resultados do modelo para base `IEDB2` e predição de **EPÍTOPO** 75
A.9 Resultados do modelo para base `IEDB3` e predição de **CDR** 76
A.10 Resultados do modelo para base `IEDB3` e predição de **H3** 76
A.11 Resultados do modelo para base `IEDB3` e predição de **L3** 76
A.12 Resultados do modelo para base `IEDB3` e predição de **EPÍTOPO** 77
A.13 Resultados do modelo para base `IEDB4` e predição de **CDR**) 77
A.14 Resultados do modelo para base `IEDB4` e predição de **H3** 77
A.15 Resultados do modelo para base `IEDB4` e predição de **L3** 78
A.16 Resultados do modelo para base `IEDB4` e predição de **EPÍTOPO** 78
A.17 Resultados do modelo para base `IEDB5` e predição de **CDR** 78

A.18 Resultados do modelo para base IEDB5 e predição de **H3** 79
A.19 Resultados do modelo para base IEDB5 e predição de **L3** 79
A.20 Resultados do modelo para base IEDB5 e predição de **EPÍTOPO** 79
A.21 Resultados do modelo para base IEDB6 e predição de **CDR** 80
A.22 Resultados do modelo para base IEDB6 e predição de **H3** 80
A.23 Resultados do modelo para base IEDB6 e predição de **L3** 80
A.24 Resultados do modelo para base IEDB6 e predição de **EPÍTOPO** 81
A.25 Resultados do modelo para base IEDB7 e predição de **CDR** 81
A.26 Resultados do modelo para base IEDB7 e predição de **H3** 81
A.27 Resultados do modelo para base IEDB7 e predição de **L3** 82
A.28 Resultados do modelo para base IEDB7 e predição de **EPÍTOPO** 82

B.1 Resultados da regularização com *weight decay* $= 0.002$ 83
B.2 Resultados da regularização com *weight decay* $= 0.005$ 84
B.3 Resultados da regularização com *weight decay* $= 0.008$ 84
B.4 Resultados da regularização com *weight decay* $= 0.01$ 84
B.5 Resultados da regularização com *data augmentation* = (0, 1x) 85
B.6 Resultados da regularização com *data augmentation* = (0, 5x) 85
B.7 Resultados da regularização com *data augmentation* = (0.2, 5x) 86
B.8 Resultados da regularização com *data augmentation* = (0.4, 5x) 86
B.9 Resultados da regularização com *data augmentation* = (0.5, 5x) 86
B.10 Resultados da regularização com *dropout* $(in = 0.3, rec = 0.0)$ 87
B.11 Resultados da regularização com *dropout* $(in = 0.5, rec = 0.0)$ 87
B.12 Resultados da regularização com *dropout* $(in = 0.0, rec = 0.3)$ 88
B.13 Resultados da regularização com *dropout* $(in = 0.0, rec = 0.5)$ 88
B.14 Resultados da regularização com *dropout* $(in = 0.3, rec = 0.3)$ 88
B.15 Resultados da regularização com *dropout* $(in = 0.5, rec = 0.3)$ 89
B.16 Resultados da regularização com *dropout* $(in = 0.3, rec = 0.5)$ 89
B.17 Resultados da regularização com *dropout* $(in = 0.5, rec = 0.5)$ 89
B.18 Resultados da regularização com *dropout* $(in = 0.6, rec = 0.0)$ 90
B.19 Resultados da regularização com *dropout* $(in = 0.0, rec = 0.6)$ 90
B.20 Resultados da regularização com *dropout* $(in = 0.6, rec = 0.6)$ 90

Lista de Tabelas

1.1 Vinte aminoácidos básicos e suas abreviações 17

3.1 Bases de Treino e Teste 31
3.2 Cinco primeiros registros de treino de `IEDB_1` 31
3.3 Exemplo de repetição de epítopo 32
3.4 Exemplo de diferenciação de epítopo com inserção de resíduo 32
3.5 Exemplo de registro do epítopo `NANPNANPNANPNANPNA` 33
3.6 Exemplo de registro do epítopo `RCNPNMEPPRCWAAEGD` 33
3.7 Exemplo com uma mutação aleatória e fator x10 34
3.8 Exemplo com uma mutação aleatória e fator x2 34
3.9 Exemplo com mutação aleatória em 25% dos resíduos e fator x2 34
3.10 Exemplo com mutação aleatória em 25% dos resíduos e fator x5 35

4.1 Comparação entre epítopo esperado e epítopo inferido - amostra 48 37
4.2 Comparação entre epítopo esperado e epítopo inferido - amostra 60 37
4.3 Comparação entre epítopo esperado e epítopo inferido - amostra 19 37
4.4 Valores para cálculo de similaridade 38
4.5 Valores para cálculo de similaridade com preenchimento 41

5.1 Exemplo de inferência epítopo - CDR 51
5.2 Parâmetros para o *grid search* 65
5.3 Predições perfeitas H3 66

Lista de Algoritmos

1 Cálculo da similaridade entre duas sequências 25
2 Cálculo da similaridade em banda limitada 26
3 Cálculo da distância de Levenshtein entre duas sequências 27

4 Cálculo da similaridade com suporte a preenchimento 39
5 Rotinas Auxiliares 40
6 Cálculo da similaridade em banda limitada com preenchimento 46
7 Cálculo da distância de Levenshtein entre duas sequências em banda limitada 49

Lista de Abreviações e Siglas

AgAbDb	*the Antigen-Antibody Interaction Database*
BERT	*Bidirectional Encoder Representations from Transformers*
BLOSUM	*Blocks of Amino Acid Substitution Matrix*
CDR	*Complementarity-Determining Regions*
CNN	*Convolutional Neural Networks*
CSV	*comma-separated values*
GRU	*Gated Recurrent Unit*
IEDB	*the Immune Epitope Database*
IMGT	*the International Immunogenetics Information System*
LSTM	*Long Short-Term Memory*
NLP	*Natural Language Processing*
PAM	*Point Accepted Mutation*
PDB	*Protein Data Bank*
RNN	*Recurrent Neural Networks*
SVM	*Support Vector Machine*

Sumário

1 Introdução **16**
1.1 Conceitos de biologia molecular e imunologia . . . 16
1.2 Motivação . . . 19
1.3 Objetivos . . . 20
1.4 Contribuições . . . 21
1.5 Organização deste documento . . . 21

2 Revisão da literatura **22**

3 Descrição da fonte de dados **28**
3.1 Introdução . . . 28
3.2 Métodos e materiais . . . 29
3.2.1 Escolha da base . . . 29
3.2.2 Obtenção e análise de dados . . . 29
3.2.3 Estrutura das bases . . . 30
3.2.4 Base `IEDB2` . . . 31
3.2.5 Base `IEDB3` . . . 32
3.2.6 Bases `IEDB4` e `IEDB5` . . . 33
3.2.7 Bases `IEDB6` e `IEDB7` . . . 33
3.2.8 Caso adicional de *data augmentation* . . . 33

4 Descrição das métricas **36**
4.1 Introdução . . . 36
4.2 Comparação de sequências . . . 36
4.2.1 Preenchimento (*padding*) . . . 39
4.2.2 Escore similar . . . 43
4.2.3 Similaridade em banda limitada . . . 45
4.2.4 Similaridade em banda limitada com preenchimento . . . 45
4.2.5 Escore similaridade com banda limitada . . . 47
4.2.6 Levenshtein, Levenshtein em banda limitada e erros por resíduo . . . 47

5 Modelo e avaliação **50**
5.1 Introdução . . . 50
5.2 Abordagem do modelo proposto . . . 50
5.3 Modelo e ensaios . . . 51
5.3.1 Modelo . . . 51
5.3.2 Infraestrutura de treino e teste . . . 53
5.3.3 Avaliação inicial do modelo com a base `IEDB` . . . 53
5.3.4 Avaliação modelo com a base `IEDB2` . . . 54

5.3.5 Avaliação modelo com a base IEDB3 . . . 56
5.3.6 Avaliação modelo com a base IEDB4 . . . 57
5.3.7 Avaliação modelo com a base IEDB5 . . . 58
5.3.8 Avaliação modelo com a base IEDB6 . . . 59
5.3.9 Avaliação modelo com a base IEDB7 . . . 60
5.3.10 Tentativa de eliminação do *overfitting* . . . 61
5.3.11 *Grid search* dos hiper-parâmetros . . . 64
5.3.12 Análise quantitativa e comparativa dos resultados . . . 65

6 Contribuições e trabalhos futuros **67**
6.1 Contribuições . . . 67
6.2 Trabalhos Futuros . . . 68

Referências Bibliográficas **70**

A Resultados dos testes iniciais para CDR, L3 e EPITOPE **73**

B Resultados dos testes com regularização **83**
B.1 Regularização com *weight decay* . . . 83
B.2 Regularização com *data augmentation* . . . 85
B.3 Regularização com *dropout* . . . 87

C Infraestrutura de treino e teste **91**
C.1 Hardware . . . 91
C.2 Software . . . 92

Capítulo 1

Introdução

Avanços recentes em aprendizado de máquina têm se mostrado muito promissores quando aplicados em biologia molecular, por permitirem uma abordagem menos complexa e melhor acurácia nos resultados. Destacam-se entre as tecnologias as redes neurais de aprendizado profundo, em especial, arquiteturas como RNN (do inglês, *Recurrent Neural Network*, CNN (do inglês, *Convolutional Neural Network*) e Autoencoders, que aprendem as características necessárias ao objetivo da previsão, extraindo-as de maneira automática. Esta pesquisa avalia a aplicação inicial destas tecnologias no campo da imunologia, para o desenvolvimento de anticorpos, dada uma necessidade específica.

1.1 Conceitos de biologia molecular e imunologia

Aminoácidos são compostos constituídos, em sua grande maioria, por carbono (C), oxigênio (O), hidrogênio (H) e nitrogênio (N), sendo que alguns também contêm enxofre (S) ou selênio (Se). Sua estrutura é composta por um carbono central, chamado carbono α, onde estão ligados: um grupo amina (NH_2), um grupo carboxila (COOH), um átomo de hidrogênio e uma cadeia lateral, que define seu nome e suas propriedades [8]. Nos humanos, existem dezenas de aminoácidos sendo, os 20 chamados básicos ou principais, apresentados na Tabela 1.1 com suas respectivas abreviaturas.

Os aminoácidos podem ser classificados em: polares ou apolares, hidrofóbicos ou hidrofílicos, e neutros, ácidos ou básicos. Aminoácidos podem formar cadeias chamadas **peptídeos**, através de ligações peptídicas, quando o grupo amina de um aminoácido (que perde um hidrogênio), liga-se a outro aminoácido no grupo carboxila (que perde um radical OH), liberando uma molécula de água.

As **proteínas** são cadeias com várias dezenas, centenas ou mais aminoácidos produzidos pelos seres vivos. Proteínas ou segmentos de proteínas podem ser representadas por sequências das abreviações citadas na Tabela 1.1. Por exemplo, o segmento de um anticorpo formado pela sequência de aminoácidos Alanina, Valina, Glicina, Isoleucina, Glicina, Alanina, Valina e Fenilalanina, pode ser representado como

```
Ala-Val-Gly-Ile-Gly-Ala-Val-Phe
```

usando as abreviações de três letras (Abrev.3), ou

Tabela 1.1: Vinte aminoácidos básicos e suas abreviações

Nome	Abrev.3	Abrev.1
Alanina	Ala	A
Cisteína	Cys	C
Aspartato	Asp	D
Glutamato	Glu	E
Fenilalanina	Phe	F
Glicina	Gly	G
Histidina	His	H
Isoleucina	Ile	I
Lisina	Lys	K
Leucina	Leu	L
Metionina	Met	M
Asparagina	Asn	N
Prolina	Pro	P
Glutamina	Gln	Q
Arginina	Arg	R
Serina	Ser	S
Treonina	Thr	T
Valina	Val	V
Triptofano	Trp	W
Tirosina	Tyr	Y

```
AVGIGAVF
```

usando as abreviações de uma letra (Abrev.1).

As proteínas podem ser representadas por sua estrutura primária, secundária, terciária e, em alguns casos, quaternária. A **estrutura primária** corresponde simplesmente à sequência de aminoácidos que a constitui. Alguns segmentos desta estrutura primária podem se organizar em **estruturas secundárias**, sendo as principais denominadas (em inglês) α ***hélices*** e β ***sheets***. As α *hélices* são partes da estrutura da proteína que se estabilizam em formato de mola, através de pontes de hidrogênio. De maneira semelhante, β *sheets* são estruturas compostas por duas ou mais β ***strands*** (sequências alinhadas de aminoácidos), estabilizadas entre si por pontes de hidrogênio. As sequências curtas de aminoácidos que ligam as estruturas secundárias são chamadas **alças** (em inglês, *loops*). Padrões de estruturas secundárias ou sequências primárias são ocasionalmente chamadas de **motivos** (em inglês, *motifs*). As **estruturas terciárias** das proteínas correspondem à sua estrutura 3D(tridimensional). Esta estrutura depende principalmente das características polares, apolares, hidrófilas e hidrofóbicas dos aminoácidos e do meio onde se encontra a proteína. **Estruturas quaternárias** ocorrem quando proteínas se ligam formando complexos proteicos maiores.

A conformação da estrutura terciária e sua constituição primária são as responsáveis pelas propriedades da proteína no organismo. As principais funções no organismo são:

- comunicação, por exemplo, como é o caso da insulina no controle do nível de açúcar do sangue;

- enzimática, por exemplo, a ação da α-amilase que auxilia na quebra das moléculas de amido no processo digestivo;
- transporte, por exemplo, a bomba de cálcio durante a contração muscular;
- armazenamento, por exemplo, no caso da ferritina que permite o armazenamento de ferro em uma forma não tóxica;
- estrutural, como por exemplo, o colágeno; e
- defesa, como no caso dos anticorpos produzidos pelas células B do sistema imunológico.

O **sistema imunológico** é um dos sistemas mais complexos existentes nos seres humanos. O sistema tem a capacidade de identificar e proteger o corpo de agentes estranhos chamados **patógenos** que podem ser micro-organismos como, por exemplo, vírus, bactérias e parasitas, e também substâncias nocivas.

Esta proteção é executada por um vasto conjunto de elementos (células e proteínas) do sistema imunológico, alguns destes nativos, de ação mais ampla e rápida, como as células *natural killer* (**células NK**), e outros adaptativos e de ação mais lenta, letal e específica como as **células T**, **células B** e **anticorpos (Ab)** [8, p.4].

A ativação do sistema imunológico nativo se dá de maneira espontânea, através da identificação de sequências repetitivas. No caso do sistema imunológico adaptativo, o agente estranho é identificado por macromoléculas liberadas ou em sua superfície chamadas **antígenos (Ag)**, sendo que um patógeno pode conter vários antígenos. Os antígenos possuem regiões chamadas de **epítopos** que são as menores regiões identificáveis a partir das quais uma **resposta imunológica adaptativa** pode ser ativada. A resposta imunológica consiste na produção envio de elementos do sistema imunológico para combate ao antígeno, causando, inicialmente, sua desativação ou bloqueio, através do acoplamento adequado de um **anticorpo** a um epítopo do antígeno. Após acoplamento do anticorpo ao antígeno, o anticorpo atrai outros elementos do sistema imune para auxiliarem na destruição completa do patógeno. As regiões de acoplamento entre o antígeno e o anticorpo são de especial importância para o presente trabalho, e serão apresentadas em mais detalhes a seguir.

Os anticorpos são proteínas chamadas **imunoglobulinas (Ig)** produzidas pelas células B e podem possuir uma ou mais estruturas em formato de Y. Cada uma destas estruturas é composta por um par de proteínas constituídas por duas cadeias: uma pesada, conhecida como **Heavy Chain (H)**, e uma leve, chamada de **Light Chain (L)**. As cadeias leve e pesada possuem duas regiões: uma constante **(c)** e outra variável **(v)**. Assim, o anticorpo pode ser dividido em um fragmento constante **(Fc)** e um fragmento variável **(Fv)**, como pode ser visto na Figura 1.1.

Cada uma das extremidades dos braços do Y, compostas por parte da região constante e pela região variável também é chamada de fragmento ab **(Fab)**. No fragmento Fv existe uma região de hiper-variação, onde residem as **CDRs (Complementarity-determining regions)**, sendo o conjunto de CDRs chamado de **paratopo**. O paratopo é composto por três regiões de hiper-variação: **CDR1, CDR2 e CDR3**. Cada uma destas regiões

possui uma parte na cadeia pesada **(H1, H2 e H3)** e uma parte na cadeia leve **(L1, L2 e L3)**. Assim, cada estrutura em Y possui um par de paratopos, cada um com três CDRs, totalizando 12 regiões de hiper-variação [8, p.15].

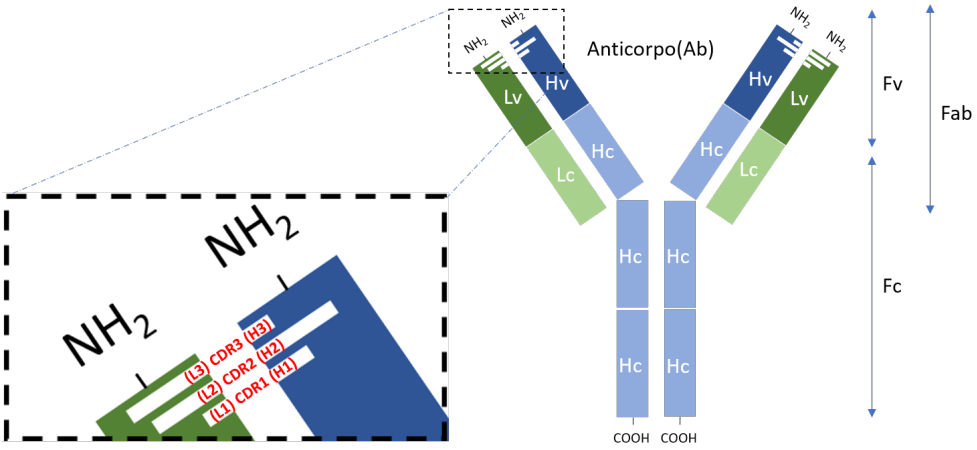


Figura 1.1: Estrutura de um anticorpo

As CDRs são as regiões dos anticorpos responsáveis pela ligação com o epítopo do antígeno, causando sua desativação e posteriormente, sua destruição. Podem existir várias CDRs diferentes que permitem ligação com um mesmo epítopo. A eficácia de um anticorpo está diretamente associada à qualidade do acoplamento (em inglês, ***docking***) entre o paratopo e o epítopo correspondente. Por sua vez, esta qualidade e estabilidade estão diretamente relacionadas a estrutura 3D dos elementos envolvidos e a força de ligação. Observou-se que esta força e especificidade é determinada, principalmente, pelas CDR3.

1.2 Motivação

A evolução propiciou aos humanos o sistema imunológico adaptativo, que funciona de maneira semelhante à capacidade de adaptação, através de mutação, dos micro organismos, que geram novas cepas resistentes às defesas existentes. Este sistema possui filtros que procuram evitar a criação de anticorpos que possam atacar as próprias células do corpo (filtragem contra o próprio). Falhas na ação destes filtros podem gerar doenças auto-imunes [8, p.8-11]. Estes filtros, ao mesmo tempo que nos protegem de um processo autoimune, também limitam a produção de anticorpos contra células cancerígenas, pois estas são muito semelhantes às células normais.

Há algum tempo são estudadas técnicas para auxílio ao sistema de defesa humano através, por exemplo, da criação de anticorpos monoclonais (**mAb**) e técnicas de enxerto, onde partes funcionais de anticorpos de outros mamíferos são enxertadas em estruturas de anticorpos humanos, permitindo assim o transporte da resposta imunológica entre espécies [9]. Infelizmente, tais técnicas são custosas e com resultados limitados a aplicações específicas.

Outros métodos envolvendo bioinformática criam anticorpos através da reorganização de partes de estruturas secundárias de anticorpos conhecidos e, em alguns casos, a substituição de partes das sequências de aminoácidos das CDRs para a criação de novos

anticorpos [4, 15]. Estas técnicas envolvem avaliações da estrutura 3D das proteínas, cálculos de entropia e energia de ligação, além de estudos de acoplamento espacial (*docking*) que normalmente exigem grande capacidade computacional, e ainda gerando informações aproximadas e nem sempre completamente conclusivas.

Como qualidade do acoplamento entre epítopo e paratopo depende fortemente da estrutura 3D destes, e esta de suas constituições primárias, é razoável considerar a possibilidade de previsão do paratopo (CDRs) a partir da estrutura primária do epítopo. Mais especificamente, estudos mostram que a CDR3 é a principal responsável pela qualidade deste acoplamento. Pretendemos, nesta dissertação, examinar a factibilidade de aprendizado por máquina da relação entre epítopos e paratopos.

A determinação das CDRs para um epítopo específico utilizando-se sua estrutura primária é de grande interesse, pois, por não envolver estruturas 3D, pode reduzir a complexidade do trabalho e consequentemente os recursos computacionais necessários.

Uma vez encontradas sugestões de CDR3 para epítopos de interesse *in silico*, a intenção é confirmar a qualidade destas sugestões *in vitro*. Como os anticorpos são proteínas, esta confirmação pode ser feita através da alteração genética de uma célula eucarionte ou bactéria, de modo a fazê-la produzir proteínas com as características desejadas, no nosso caso, anticorpos com a CDR3 sugerida. Após a produção, estes anticorpos podem, então, ser testados no laboratório contra os epítopos de interesse inicialmente escolhidos, por exemplo utilizando teste **ELISA (*Enzyme Linked Immuno Sorbent Assay*)**.

A relativa facilidade de criação de anticorpos com CDR3 desejada, a menor complexidade e recursos associados aos estudos a partir de estruturas primárias e a capacidade de evitar a filtragem negativa contra o próprio são os principais fatores motivantes desta pesquisa.

Acredita-se que, através desta pesquisa, a criação de anticorpos diretamente a partir dos epítopos e evitando a filtragem contra o próprio, possa vir a sugerir anticorpos que nunca seriam produzidos pelo corpo, e que, mesmo com efeitos colaterais, em condições controladas, podem ser úteis no combate ao câncer.

1.3 Objetivos

A possibilidade de desenvolvimento de anticorpos específicos a partir de epítopos pode ser considerada como um dos maiores avanços que a tecnologia poderá trazer nos próximos anos. Esta pesquisa tem como objetivo iniciar esta busca, tendo como base as técnicas de aprendizado de máquina desenvolvidas nos últimos anos. Neste sentido, tem-se como elementos essenciais para este desenvolvimento:

- encontrar fontes de dados de qualidade e com tamanho suficiente para o treinamento dos modelos;
- identificar e entender métodos estado-da-arte disponíveis atualmente;
- definir métricas apropriadas para avaliação da performance e resultado dos modelos;
- avaliar os métodos disponíveis, permitindo a seleção das técnicas mais promissoras a serem aplicadas.

1.4 Contribuições

O desenvolvimento de anticorpos é uma área de grande interesse e investimentos atualmente, tanto na busca da cura para doenças transmissíveis quanto para tratamento do câncer. Várias técnicas focadas nas estruturas 3D continuam sendo avaliadas e aprimoradas neste sentido, porém, normalmente demandam grandes recursos humanos e computacionais.

Uma das contribuições esperadas deste trabalho é a avaliação das bases disponíveis para estudos utilizando aprendizado de máquina, visto que cada uma fornece dados em um formato específico e identificando as CDRs diferentemente.

A principal contribuição é a avaliação de algoritmos focados nas estruturas primárias para previsão de CDR3 a partir de epítopos, tendo-se como foco o quanto que estes algoritmos podem aprender das características necessárias para esta previsão. Acreditamos que a identificação ou criação de métricas específicas que possam auxiliar o aprendizado seja um dos pontos que possam melhorar o desempenho destes algoritmos.

1.5 Organização deste documento

O restante deste documento está organizado como segue. No Capítulo 2 é apresentada a revisão da literatura. No Capítulo 3 apresentamos uma avaliação de algumas das bases de dados públicas de interesse e, detalhamos a estrutura da base escolhida e as bases geradas para teste do modelo. No Capítulo 4 são apresentados os algoritmos desenvolvidos e utilizados na pesquisa. No Capítulo 5 são apresentados a arquitetura proposta para estudo e os resultados obtidos. No Capítulo 6, os resultados são comentados frente aos problemas e contribuições e, por fim, são propostas algumas novas linhas de pesquisa. Resultados de todos os testes executados são apresentados nos Apêndices A e B. A configuração da infraestrutura computacional utilizada é apresentada no Apêndice C.

Capítulo 2

Revisão da literatura

Os avanços recentes em aprendizado de máquina têm se mostrado muito promissores quando aplicados em biologia molecular por permitirem uma abordagem menos complexa e com acurácia nos resultados. Destacam-se entre as tecnologias as redes neurais de aprendizado profundo, em especial arquiteturas como RNN (do ingles, *Recurrent Neural Network*, CNN (do inglês, *Convolutional Neural Network*) e *Autoencoders*, que aprendem as características necessárias ao objetivo da previsão, extraindo-as de maneira automática [2, 7, 19]. Ofer e colegas [21] mostraram que a similaridade entre a representação das sequências de aminoácidos nas estruturas primárias e a linguagem natural permitem o uso de técnicas e arquiteturas de NLP (*Natural Language Processing*) no entendimento das proteínas em suas diversas representações estruturais.

Em pesquisas recentes, algumas abordagens utilizando aprendizado profundo e estruturas primárias foram relatadas. Akbar e colegas [1] encontraram um vocabulário de motivos estruturais, derivados de sequências de resíduos das regiões de contato entre o epítopo e do paratopo. Estes motivos estruturais, convertidos em sequências e utilizando um algoritmo baseado em tradução através de redes neurais (*Neural Machine Translation*) [17], obtiveram bons resultados na previsão de motivos do paratopo a partir de motivos do epítopo e vice-versa. Já o mesmo teste, usando apenas as sequências de resíduos de epítopos e paratopos, e comparando resultados (paratopo previsto x paratopo original) utilizando distância normalizada de Levenshtein, não teve resultados significativos.

Os resultados obtidos a partir de motivos estruturais mostram uma capacidade de previsibilidade dos motivos de 60% a 70% (erro médio de previsão de 40% a 30%). Assim, considerando que os motivos mantêm um bom nível de relacionamento com as respectivas sequências de resíduos, as regiões associadas aos motivos podem ser previstas com, praticamente, o mesmo nível de previsibilidade. Este resultado, apesar de interessante, não permite a previsão das sequências completas das CDRs, pois os motivos não cobrem completamente estas regiões, que é justamente um dos objetivos de nossa pesquisa.

Em alguns casos, a sequência do epítopo não é conhecida. Técnicas de biologia molecular, como *phage display*, podem ser utilizadas nestes casos para aprimorar a afinidade de anticorpos, através da criação de novos anticorpos derivados de um anticorpo conhecido, por meio da randomização da CDRH3. Apesar de resultados satisfatórios, este processo é muito custoso. Liu e colegas [15], desenvolveram um método de seleção, utilizando redes

neurais, que permite a seleção *in silico*, melhorando a especificidade e afinidade dos anticorpos. A seleção é feita através do treino de uma rede neural para rejeição de anticorpos que tenham afinidade com outros alvos. Este método difere de nossa proposta por utilizar anticorpos conhecidos e melhorá-los, não envolvendo as características do epítopo como entrada.

Mason e colegas [20] utilizaram RNN e CNN em outra abordagem. A partir do CDRH3 de um anticorpo conhecido, específico para um determinado antígeno, foram geradas combinações associadas a dez aminoácidos. Uma RNN e uma CNN foram treinadas para selecionar sequências com capacidade de acoplamento ao antígeno desejado. Para ambos os casos, foram feitos ajustes nos hiper-parâmetros utilizando *grid search*. Por fim, após uma pré-filtragem, as sequências originalmente geradas foram aplicadas às redes treinadas, sendo que a seleção final é dada pelo consenso entre os resultados das duas redes. Com esta abordagem, foi possível uma nova seleção da ordem de 10% das sequências com maior probabilidade e qualidade de acoplamento. Para aferir o resultado, outros métodos de avaliação de acoplamento foram utilizados, mostrando que, em uma amostra de 30 sequências, dentre as selecionadas, todas tinham boa capacidade de acoplamento. Esta abordagem, como a anterior de Liu e colegas, consiste em otimização de anticorpos, pois partem de anticorpos conhecidos e dos epítopos.

Algumas abordagens utilizando a estrutura 3D foram estudadas, porém também não partem do epítopo e visam à construção, otimização e avaliação de anticorpos. Baran e colegas [4] propuseram o algoritmo *AbDesign*, que cria anticorpos a partir de *backbones* e segmentos de outros anticorpos conhecidos. Estes então são otimizados através de ajustes nas sequências e avaliação de estabilidade e atividade e força de ligação ao alvo escolhido.

Outro estudo utilizando estrutura 3D propõe um método computacional genérico para desenho de anticorpos com alta especificidade e afinidade para um epítopo de um antígeno [23]. O método parte de uma estrutura canônica de CDR no qual aminoácidos são agregados utilizando-se de funções de energia, tabela de rotâmeros e processos cíclicos de otimização através de programação linear inteira-mista. O preenchimento é executado para cada CDR na seguinte ordem: H3, L3, H2, H1, L1 e L2, de maior aproximação das CDRs com o epítopo. Neste caso, o método tem foco no desenho das CDRs que, uma vez desenhados e otimizados pelo método proposto, podem ser enxertados em uma estrutura genérica de um anticorpo. O processo foi aplicado em três casos onde tinha-se epítopo e um anticorpo de referência, gerando anticorpos com características de grande afinidade com os respectivos epítopos.

Em todos os trabalhos citados até o momento, o foco de desenvolvimento está nos anticorpos, utilizando os epítopos, de maneira geral, apenas para avaliação de afinidade e qualidade do resultado. Existem alguns estudos onde o foco é a identificação ou previsão de epítopos [25, 16, 12, 26, 28]. A exatidão no sequenciamento de um antígeno é a informação de partida para se definir precisamente qual alvo da CDR deseja-se prever.

Bowman e colegas [6] exploraram uma técnica chamada Vacinologia Reversa. Esta técnica busca identificar potenciais epítopos a partir do antígeno, utilizando técnicas de aprendizado de máquina, como por exemplo, SVM (*Support Vector Machine*) entre outras. A identificação de epítopos específicos permite o desenvolvimento de vacina de maneira mais rápida e eficaz.

Em relação a técnicas específicas para o previsão e desenvolvimento de CDR, várias abordagens são possíveis e alguns conhecimentos são essenciais como: sequência do epítopo de interesse, arquitetura de referência do anticorpo, entendimento da natureza da interação Ag-Ab e conhecimento das características das CDRs.

O conhecimento das características das CDRs e interações Ag-Ab foram estudadas por Ofran e colegas [22]. A partir de dados estruturais de complexos Proteína-Ab disponíveis no PDB (*Protein Data Bank*) e SCOP (*Structural Classification of Proteins*), foram feitos estudos para a determinação dos pontos de contato e, a partir destes, as posições de início e fim das CDRs. Resíduos nas CDRs estão em contato com resíduos dos epítopos em 98% dos casos. Foi verificado que a frequência de ocorrência de aminoácidos é diferente quando comparadas às CDRs, epítopos e o restante da estrutura dos anticorpos, havendo também diferença na frequência entre as cadeia H e L.

Alguns modelos mais recentes utilizam-se de aprendizado não supervisado buscando um entendimento da linguagem das proteínas, isto é, as regras que regem as sequências de aminoácidos. Vig e colegas [30] avaliaram profundamente os mecanismos de atenção da arquitetura BERT, treinado apenas com a estrutura primária das de proteínas, e mostraram que esta é capaz de aprender diversas propriedades como: a atenção sobre cada tipo de aminoácido tem grande correlação com regras de substituição de proteínas sugeridas pela tabela BLOSUM; outras camadas de atenção conseguem aprender propriedades de contato entre aminoácidos que produzem as estruturas secundárias e terciárias; e camadas mais profundas mostram relacionamento entre a atenção e sítios funcionais. Rao e colegas [24] foram mais a fundo e mostraram que é possível extrair as propriedades de contato das camadas de atenção utilizando-se de um novo modelo obtido através de *transfer learning* com a adição de uma camada final e um pequeno retreino para ajuste fino (*fine tunning*).

Vários métodos de numeração destas posições foram propostos e são utilizados, dependendo das aplicações e dos diversos métodos de engenharia de anticorpos desenvolvidas [9]. Os esquemas de numeração são baseados em diferentes alinhamentos das sequências de aminoácidos e consideram, em alguns casos, a existência de inserções e espaços vagos para não alterem a sequência da numeração. Os esquemas de numeração Kabat e IMGT são baseados em alinhamentos nas sequências da estrutura dos anticorpos, enquanto outros como Chotia têm maior relação com as estruturas das alças.

Abordagens nas pesquisas sobre proteínas podem envolver foco em suas estruturas primárias, secundárias ou terciárias. No caso de pesquisas envolvendo estruturas primárias, a comparação entre sequências é a operação mais importante e envolve vários conceitos como, por exemplo, alinhamento e similaridade. Setubal e Meidanis [27, Cap 3] apresentam alguns algoritmos de grande importância para esta pesquisa, como o Algoritmo 1 de cálculo de similaridade e o Algoritmo 2 de cálculo de similaridade em banda, que é mais otimizado apesar de envolver algumas restrições de uso.

Algoritmo 1: Algoritmo de cálculo da similaridade entre duas sequências como proposto por Setubal e Meidanis [27, Sec 3.2]. Os valores das constantes *GAP, MATCH* e *MISMATCH* estão definidas na Tabela 4.4. Por convenção, foi considerado que a primeira posição das sequências possui índice zero ([0]).

```
Result: similaridade entre as sequências s e t
Function sim(s,t)
    m ← |s|;
    n ← |t|;
    for i de 0 até m do
        a[i, 0] ← i * GAP
    end
    for j de 0 até n do
        a[0, j] ← j * GAP
    end
    for i de 1 até m do
        for j de 1 até n do
            if s[i-1] = t[j-1] then
                d ← MATCH
            else
                d ← MISMATCH
            end
            a[i, j] ← max(a[i − 1, j] + GAP, a[i − 1, j − 1] + d, a[i, j − 1] + GAP)
        end
    end
    return a[m,n]
end
```

Outra métrica interessante para comparação de sequências é a distância de edição [32], também chamada distância de Levenshtein, que pode ser calculada utilizando o Algoritmo 3.

Algoritmo 2: Algoritmo de cálculo da similaridade entre as sequência s e t, limitando a k a diferença no número de gaps nos alinhamentos entre s e t, adaptado de Setubal e Meidanis [27, Sec 3.3.4], para sequências de tamanhos distintos. Os valores das constantes *GAP, MATCH* e *MISMATCH* são definidas na Tabela 4.5. Por convenção, foi considerado que a primeira posição das sequências possui índice zero ([0]).

Result: melhor avaliação de similaridade entre as sequências s e t, limitando a diferença de *gaps* nos alinhamentos entre s e t a k.

```
Function InsideStrip(i,j,k)
    return  -k ≤ i - j ≤ k
end
Function sim_ banda(s,t,k)
    m ← |s|;
    n ← |t|;
    for i de 0 até k do
        a[i, 0] ← i * GAP
    end
    for j de 0 até k do
        a[0, j] ← j * GAP
    end
    for i de 1 até m do
        for d de -k até k do
            j ← i + d;
            if 1 ≤ j ≤ n then
                if s[i-1] = t[j-1] then
                    a[i, j] ← a[i - 1, j - 1] + MATCH
                else
                    a[i, j] ← a[i - 1, j - 1] + MISMATCH
                end
                if InsideStrip(i-1,j,k) then
                    a[i, j] ← max(a[i, j], a[i - 1, j] + GAP)
                end
                if InsideStrip(i,j-1,k) then
                    a[i, j] ← max(a[i, j], a[i, j - 1] + GAP)
                end
            end
        end
    end
    /* trata os casos quando a diferença entre o tamanho das
       sequências é maior que k                                  */
    if  n > m + k then
        a[m, n] ← a[m, m + k] + (n - m - k) * GAP
    end
    if  m > n + k then
        a[m, n] ← a[n + k, m] + (m - n - k) * GAP
    end
    return a[m,n]
end
```

Algoritmo 3: Cálculo da distância de Levenshtein entre duas sequências. Por convenção, foi considerado que a primeira posição das sequências possui índice zero ([0]).

```
Result: distância de Levenshtein entre as sequências s e t
Function dist_leven(s,t)
    m ← |s|;
    n ← |t|;
    for i de 0 até m do
        a[i, 0] ← i
    end
    for j de 0 até n do
        a[0, j] ← j
    end
    for i de 1 até m do
        for j de 1 até n do
            if s[i-1] = t[j-1] then
                a[i, j] ← a[i − 1, j − 1]
            else
                a[i, j] ← a[i − 1, j − 1] + 1
            end
            a[i, j] ← min(a[i − 1, j] + 1, a[i, j], a[i, j − 1] + 1)
        end
    end
    return a[m,n]
end
```

Outro fator importante, considerando a possibilidade de previsão executada por modelos de aprendizado de máquina, é a existência de dados precisos, completos e curados para treinamento dos modelos. Neste sentido, a *Immune Epitope Database* - IEDB é uma alternativa atraente, por fornecer acesso público e livre à base [18].

Capítulo 3

Descrição da fonte de dados

O treino, o teste e a validação de modelos de aprendizado profundo necessitam, normalmente, de uma amostra grande de dados que representem adequadamente o espaço de possibilidades onde o modelo irá ser utilizado. Em nosso caso, em especial, tais informações são obtidas por processos de sequenciamento e análise de afinidade efetuados durante vários anos e registrados em bases de dados, de modo geral, com objetivos específicos.

Este capítulo detalha e avalia a obtenção dos dados utilizados para treino e avaliação do modelo proposto.

3.1 Introdução

Dados são elementos essenciais para o treinamento, validação e teste de modelos de aprendizado de máquina. A disponibilidade de dados públicos sobre proteínas é bem ampla, sendo a *Protein Data Bank* (PDB), disponibilizada pela Worldwide PDB (wwPDB) [5], a principal base aberta disponível. Entretanto, para nosso objetivo que envolve proteínas específicas (anticorpos) e seus epítopos correspondentes, a PDB não nos atende, e bases públicas são mais restritas. Além disso, para esta pesquisa é importante que a base tenha informações complementares sobre a qualidade da ligação Ag-Ab, se as informações existentes envolvem sequências curadas ou não, se as ligações foram efetivas e realmente provocaram a eliminação do antígeno, se as sequências correspondem a pares de CDRs (um de cadeia leve e um de cadeia pesada) e ainda que estejam disponíveis as sequências primárias das CDRs e epítopos, pois estas estão no formato adequado, no caso sequências, para utilização no modelo proposto.

Inicialmente três bases foram consideradas promissoras:

- **IEDB** [31]. *The Immune Epitope Database* é uma base de acesso livre disponibilizado pela NIAID (*National Institute of Allergy and Infectious Diseases*). Contém um catálogo de dados experimentais de epítopos de anticorpos e células T relativos a estudos em humanos, outros primatas e outras espécies de animais no contexto de doenças infecciosas, alergias, doenças autoimunes e transplantes. O site também fornece ferramentas para auxílio na predição e análise de epítopos. A IEDB possuía em 26/12/2019, 618.745 epítopos peptídicos, 2.886 não peptídicos, 492.022 ensaios com células B, e epítopos de 3.788 organismos, associados a 20.791 referências.

- **IMGT®** [14, 13]. IMGT® é uma base criada em 1989 por Marie-Paule Lefranc (Université de Montpellier e CNRS) especializada em imunoglobulinas (Ig), células T (TR) e complexo principal de histocompatibilidades (MHC). A base contém informações sobre sequência, genoma, estruturas imunogenéticas e ferramentas de suporte.

- **AgAbDb** [10]. *The Antigen-Antibody Interaction Database (AgAbDb)* é uma base de imunoinformática desenvolvida pelo Centro de Bioinformática da Universidade de Pune, na Índia, e contém informações sobre a interação molecular entre antígenos proteicos e peptídicos caracterizados por estruturas de co-cristais. Além das informações de interação, o banco também possui informações sobre a descrição dos antígenos e anticorpos, e a definição dos resíduos das cadeias H3 e L3, seguindo a numeração de Wu e Kabat [9]. A base AgAbDb possuía, em 27/12/2019, 551 complexos Ag-Ab, sendo 389 complexos Proteína-Ab e 162 complexos Peptídeo-Ab.

3.2 Métodos e materiais

Nesta seção é apresentado o resultado da análise das bases públicas e o processo de construção dos conjuntos de treino e teste.

3.2.1 Escolha da base

O primeiro passo seria a escolha da base ou bases para obtenção dos dados. Esta escolha deveria garantir a disponibilidade, isto é, ser uma base de acesso estável e em crescimento, e de fácil seleção de sequências de pares epítopo-CDR curados. Considerando estes dois fatores de facilidade e disponibilidade, foi possível descartar quase que de imediato a base **AgAbDb**, pois, em três tentativas durante o período de estudo e consulta de dados a mesma não estava disponível. A base **IMGT®** também foi descartada como fonte de dados, pois não permitia consultas flexíveis para avaliação de seu conteúdo o que não permitiu uma análise de sequências com as características necessárias para a pesquisa. Por fim, a base **IEDB**, além de informar a quantidade total de elementos na base, fornece logo em sua primeira página, um mecanismo flexível de busca e seleção, mostrando-se portanto uma base viável para análise e utilização.

3.2.2 Obtenção e análise de dados

Uma vez selecionada a base **IEDB** para avaliação, prosseguimos com o processo de extração de dados e análise. Assim, em 6/7/2020 efetuamos uma seleção no site com as opções: *Epitope* = `Linear`, *Assays* = `B-cell` + `Positive Only`. Estes parâmetros permitem selecionar apenas epítopos com sequências contíguas e que tenham células-B com CDR correspondente. A seleção resultou em 955 registros na aba Receptor os quais foram exportados em formato CSV.

Avaliando os dados exportados, notou-se que havia entre os epítopos casos não desejados, como produtos químicos, sequências modificadas e epítopos não lineares. Estes

casos foram então excluídos, restando 751 registros de epítopos sequenciais. Novos casos não desejados foram identificados e removidos envolvendo anticorpos com duas cadeias pesadas, ou duas cadeias leves, ou com apenas uma cadeia, resultando num total de 583 registros. Por fim, verificou-se que existiam triplas repetidas (Epítopo, CDRH3, CDRL3) devido a ensaios distintos. As repetições foram removidas resultando em 472 registros distintos, os quais constituem a base inicial para treinamento e teste do modelo. Esta base é referenciada neste texto como base Baseline Pesquisa IEDB.

A partir desta base inicial, duas outras etapas de processamento são aplicadas para geração das bases: um para geração dos conjuntos de treino e teste e outro para geração de outras bases através de diferentes processos de melhorias e *data augmentation*.

A partir da base Baseline Pesquisa IEDB, outras bases foram geradas, conforme certas especificidades dos dados foram identificadas, e tentativas de manutenção ou ampliação da base (*data augmentation*) foram utilizadas. No total, oito bases foram geradas para testes dos modelos, conforme detalhado na Tabela 3.1. O relacionamento entre as bases está ilustrado na Figura 3.1.

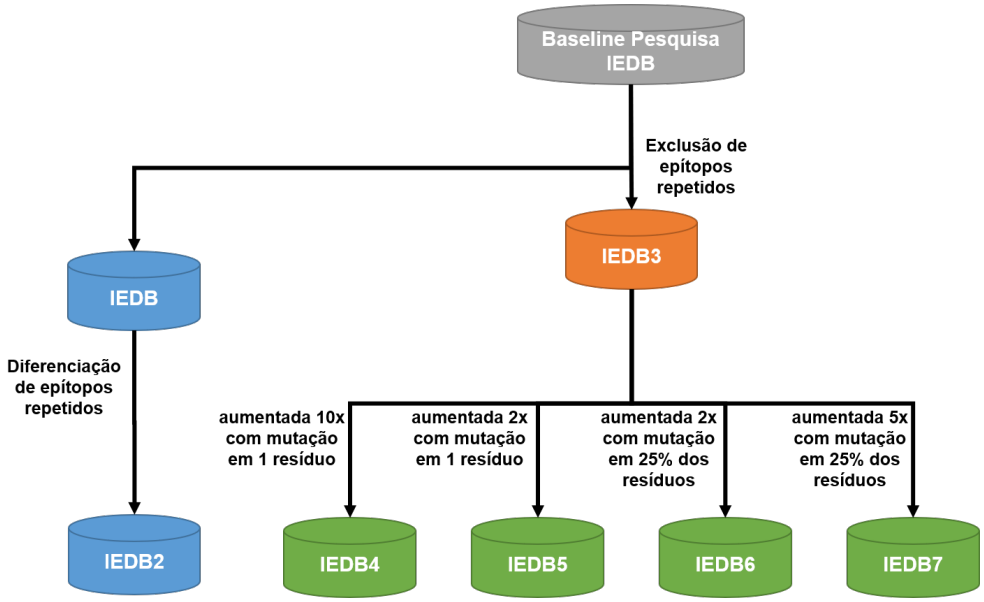


Figura 3.1: Relacionamento entre as bases

Os resultados obtidos com as bases e uma explicação mais detalhada da motivação de sua criação pode ser encontrada no Capítulo 5. A seguir, a estrutura de cada uma das bases é apresentada.

3.2.3 Estrutura das bases

As bases são compostas de dez conjuntos de registros de treino e registros de teste para execução de validação cruzada dos modelos. Cada conjunto corresponde à seleção aleatória de 80% da base inicial derivada da pesquisa na IEDB, para o conjunto de treino e 20% restantes para o conjunto de teste.

Tais conjuntos são nomeados adicionando-se um sufixo _n com n entre 1 e 10. Como exemplo, a base `IEDB`, derivada diretamente da pesquisa, após os expurgos citados anteri-

Tabela 3.1: Bases de Treino e Teste. IEDB3 é usada como baseline de avaliação. As bases aumentadas IEDB4 até IEDB7 são sem repetição de epítopos e foram aumentadas apenas no conjunto de treino. As mutações foram aplicadas apenas nos epítopos.

Nome	Treino	Teste	Descrição Resumida da Base
IEDB	377	95	Baseline Pesquisa IEDB
IEDB2	377	95	diferenciação de epítopos repetidos
IEDB3	288	72	sem repetição de epítopos
IEDB4	2880	72	aumentada 10x com mutação em um resíduo
IEDB5	576	72	aumentada 2x com mutação em um resíduo
IEDB6	576	72	aumentada 2x com mutação em 25% dos resíduos
IEDB7	1440	72	aumentada 5x com mutação em 25% dos resíduos

Tabela 3.2: Cinco primeiros registros de treino de IEDB_1

Epítopo	CDRH3	CDRL3
TKDTNNNL	SIIYFDYADFIMDY	SQSTHVPRT
NANPNANPNANPNAN	ARVYSYGVPFDY	CSYVGSDTWV
NWFDISKWLWYI	AREGTTGWGWLGKPIGAFAH	QQYGQSLST
VATALNPDPSQK	VRQMGDY	QHFWGTPFT
ARSNSTIARGGNSN	YYSDWDYAMDS	WQGTHFPL
DAEFRHD	VRYDHYSGSSDY	WQGTHFPRT

ormente, é composta dos seguintes conjuntos: IEDB_1, IEDB_2, IEDB_3, IEDB_4, IEDB_5, IEDB_6, IEDB_7, IEDB_8, IEDB_9 e IEDB_10.

Uma amostra do conjunto de treino de IEDB_1 pode ser encontrada na Tabela 3.2. Cada registro é composto por uma tupla (Epítopo, CDRH3 e CDRL3). Epítopo, CDRH3 e CDRL3 são compostos por sequências de símbolos diretamente associados as sequências de resíduos (aminoácidos) que os compõem, sendo cada simbolo correspondente a letra definida como Abrev.1 na Tabela 1.1. Para qualquer um dos modelos, é possível selecionar qual atributo se deseja inferir a partir do epítopo entre: Epítopo, CDR3 (H3,L3), somente H3 ou somente L3.

3.2.4 Base IEDB2

Esta base foi criada após a identificação da existência de registros com os mesmos epítopos na base IEDB. Dada a seleção aleatória, tais registros podem ocorrem nos conjuntos de treino, nos conjuntos de teste e em ambos.

A existência de anticorpos com pares (CDRH3, CDRL3) distintos para um mesmo epítopo é algo esperado e conhecido, sendo estas variações normalmente associadas a diferentes níveis de afinidade de ligação.

Para diferenciação entre os registros com o mesmo epítopo, nesta base, foi incluído um resíduo adicional, escolhido aleatoriamente, no final das sequências dos epítopos, reduzindo a probabilidade de existência de registros com epítopos iguais. Esta diferenciação foi aplicada nos registros com epítopos repetidos uma única vez antes da geração dos

Tabela 3.3: Exemplo de repetição de epítopo em registros de treino de IEDB_1

Epítopo	CDRH3	CDRL3
NANPNANPNANPNANPNA	TRGIIVGDTWHFDP	MQSIQLVT
NANPNANPNANPNANPNA	AACSNVGCYVY	MQGTHWPDT
NANPNANPNANPNANPNA	TVWYLETPDDGFDI	QAWDSGIWV
NANPNANPNANPNANPNA	AKSTGDNWFDP	QQRSNWYT
NANPNANPNANPNANPNA	ARNVLDSSGYPTYFDY	MQAVQTPLT
NANPNANPNANPNANPNA	VKVEDYVRGSSHGGAFHI	QQYSSYWT
NANPNANPNANPNANPNA	AKVGDGTVAGTIDY	MQGTHWWT
NANPNANPNANPNANPNA	VKVGANWGWRYFDL	MQGIHSWT

Tabela 3.4: Exemplo de diferenciação de epítopo com inserção de resíduo no conjunto de treino IEDB2_1

Epítopo	CDRH3	CDRL3
NANPNANPNANPNANPNAI	TRGIIVGDTWHFDP	MQSIQLVT
NANPNANPNANPNANPNAG	AACSNVGCYVY	MQGTHWPDT
NANPNANPNANPNANPNAN	TVWYLETPDDGFDI	QAWDSGIWV
NANPNANPNANPNANPNAE	AKSTGDNWFDP	QQRSNWYT
NANPNANPNANPNANPNAH	ARNVLDSSGYPTYFDY	MQAVQTPLT
NANPNANPNANPNANPNAD	VKVEDYVRGSSHGGAFHI	QQYSSYWT
NANPNANPNANPNANPNAF	AKVGDGTVAGTIDY	MQGTHWWT
NANPNANPNANPNANPNAA	VKVGANWGWRYFDL	MQGIHSWT

conjuntos de treino e teste. Assim, um registro no qual foi aplicada a diferenciação pode aparecer tanto no conjunto de treino, como no conjunto de teste.

São apresentados, na Tabela 3.3, um exemplo de epítopos repetidos no conjunto de treino de IEDB_1, e na Tabela 3.4, a diferenciação produzida no conjunto de treino de IEDB2_1.

3.2.5 Base IEDB3

Esta base foi criada com objetivo de eliminação de registros com os mesmos epítopos na base IEDB. Desta forma todos os registros da base possuem epítopos distintos entre si. Esta base é gerada a partir da base Baseline Pesquisa IEDB, mantendo-se apenas o registro com a primeira ocorrência do epítopo e eliminando os demais registros com epítopos repetidos.

Assim, após a execução deste processo na base Baseline Pesquisa IEDB, inicialmente com 472 registros, obtém-se 360 registros com epítopos distintos. A partir destes então é gerado base IEDB3 totalizando 288 registros para treino e 72 registros para teste.

Desta forma, por exemplo, os registros com epítopos repetidos exemplificados na Tabela 3.3 , no caso do conjunto IEDB3_1, reduz-se ao registro detalhado na Tabela 3.5.

Tabela 3.5: Exemplo do registro representativo do epítopo NANPNANPNANPNANPNA no conjunto de treino IEDB3_1

Epítopo	CDRH3	CDRL3
NANPNANPNANPNANPNA	VKVGANWGWRYFDL	MQGIHSWT

Tabela 3.6: Exemplo do registro representativo do epítopo RCNPNMEPPRCWAAEGD no conjunto de treino IEDB3_2

Epítopo	CDRH3	CDRL3
RCNPNMEPPRCWAAEGD	VSGMEY	SQSTHVPPLT

3.2.6 Bases IEDB4 e IEDB5

As bases IEDB4 e IEDB5 são conjuntos com *data augmentation* gerados a partir da criação de novos registros de treino efetuando-se mutações aleatórias de um resíduo do epítopo.

A base IEDB4 foi gerada com um fator x10 (mutação aleatória na geração de 9 registros), associados a um mesmo par (CDRH3, CHRL3). De maneira semelhante, para geração da base IEDB5 foi utilizado um fator x2.

Para exemplificar, pode ser visto na Tabela 3.6, um registro da base IEDB3_2, observando-se na Tabela 3.7 os registros correspondentes gerados com a aplicação da mutação aleatória e fator x10 para geração da base IEDB4_2, e na Tabela 3.8 os registros correspondentes gerados com a aplicação da mutação aleatória e fator x2 para geração da base IEDB5_2.

3.2.7 Bases IEDB6 e IEDB7

As basesIEDB6 e IEDB7 são conjuntos com *data augmentation* gerados a partir da criação de novos registros de treino efetuando-se mutações aleatórias de 25% dos resíduos do epítopo.

A base IEDB6 foi gerada com um fator x2, isto gerando o dobro de registros associados a um mesmo par (CDRH3, CHRL3) que a base IEDB3. De maneira semelhante, para geração da base IEDB7 foi utilizado um fator x5.

Para exemplificar, pode ser visto na Tabela 3.6, um registro da base IEDB3_2, observando-se na Tabela 3.9 os registros correspondentes gerados com a aplicação da mutação aleatória em 25% dos resíduos do epítopo e fator x2 para geração da base IEDB6_2, e na Tabela 3.10 os registros correspondentes gerados com a aplicação da mutação aleatória em 25% dos resíduos do epítopo e fator x5 para geração da base IEDB7_2.

3.2.8 Caso adicional de *data augmentation*

Além de todas opções de bases apresentadas para teste dos modelos, um outro processo de *data augmentation* também foi implementado de maneira dinâmica, durante o processo de treino.

Tabela 3.7: Exemplo dos registros resultantes do epítopo `RCNPNMEPPRCWAAEGD` no conjunto de treino `IEDB4_2` após o uso de *data augmentation* com uma mutação aleatória e fator x10. Temos um total de dez registros, sendo um original e nove gerados por mutações, indicados em vermelho.

Epítopo	CDRH3	CDRL3
LCNPNMEPPRCWAAEGD	VSGMEY	SQSTHVPPLT
RCNPNMEDPRCWAAEGD	VSGMEY	SQSTHVPPLT
RCNPNMEPPRSWAAEGD	VSGMEY	SQSTHVPPLT
RCNPNMEPPRCWAAEGD	VSGMEY	SQSTHVPPLT
RCNPNMEPPRCYAAEGD	VSGMEY	SQSTHVPPLT
RCNPNMEPPRCWAAVGD	VSGMEY	SQSTHVPPLT
RKNPNMEPPRCWAAEGD	VSGMEY	SQSTHVPPLT
RCNPSMEPPRCWAAEGD	VSGMEY	SQSTHVPPLT
RCNLNMEPPRCWAAEGD	VSGMEY	SQSTHVPPLT
RCNPNMEPPRCWYAEGD	VSGMEY	SQSTHVPPLT

Tabela 3.8: Exemplo dos registros resultantes do epítopo `RCNPNMEPPRCWAAEGD` no conjunto de treino `IEDB5_2` após o uso de *data augmentation* com uma mutação aleatória e fator x2. Temos um total de dois registros, sendo um original e outro gerado por mutação, indicada em vermelho.

Epítopo	CDRH3	CDRL3
RCNPNMEPPRCWAAEGD	VSGMEY	SQSTHVPPLT
RCNPNMEPPRCVAAEGD	VSGMEY	SQSTHVPPLT

Tabela 3.9: Exemplo dos registros resultantes do epítopo `RCNPNMEPPRCWAAEGD` no conjunto de treino `IEDB6_2` após o uso de *data augmentation* com mutação aleatória em 25% dos resíduos e fator x2. Temos um total de dois registros, sendo um original e outro gerado por mutações, indicadas em vermelho. O número de resíduos com mutação corresponde à aplicação do percentual sobre o comprimento do epítopo, arredondado para cima.

Epítopo	CDRH3	CDRL3
RCNPNMEPPRCWAAEGD	VSGMEY	SQSTHVPPLT
RCNPNMEPPRCNRAGKI	VSGMEY	SQSTHVPPLT

Tabela 3.10: Exemplo dos registros resultantes do epítopo `RCNPNMEPPRCWAAEGD` no conjunto de treino `IEDB7_2` após o uso de *data augmentation* com mutação aleatória em 25% dos resíduos e fator x5. Temos um total de cinco registros, sendo um original e quatro gerados por mutações, indicadas em vermelho. O número de resíduos com mutação corresponde à aplicação do percentual sobre o comprimento do epítopo, arredondado para cima.

Epítopo	CDRH3	CDRL3
WQNPNEEPPRCWAAENW	VSGMEY	SQSTHVPPLT
RCNPNMQPPRCWHTLGR	VSGMEY	SQSTHVPPLT
GCIENMAEPRCWAAEGD	VSGMEY	SQSTHVPPLT
FPNTNMEPTRCWAGEGD	VSGMEY	SQSTHVPPLT
RCNPNMEPPRCWAAEGD	VSGMEY	SQSTHVPPLT

Neste processo, quando selecionado, cada registro de entrada do modelo é replicado n vezes, sendo n correspondente ao valor do parâmetro opcional ***-a*** de treino do modelo.

O uso deste parâmetro deve ser feito em conjunto com o parâmetro ***-i***, o qual especifica o nível de ruído na codificação *one_hot* dos dados de entrada.

O uso conjunto dos parâmetros ***-a*** e ***-i*** permite a geração de vários registros de entrada com codificação *one_hot* distintas, porém associadas a um mesmo epítopo.

Maiores detalhes e motivação serão apresentados no Capítulo 5.

Capítulo 4

Descrição das métricas

Ensaios científicos requerem métricas bem definidas para registro e avaliação. Este capítulo detalha as métricas que foram utilizadas nesta pesquisa, tanto para o cálculo de perda durante os treinos como para a avaliação dos resultados.

4.1 Introdução

O objetivo desta pesquisa é encontrar sequências de CDR3 a partir do epítopo. Em biologia computacional, métricas que envolvem similaridade e alinhamento são as mais utilizadas na comparação de sequências e busca em bancos de dados [27]. Nos ensaios de avaliação do modelo foram utilizadas métricas baseadas em similaridade e distância de edição (Levenshtein), na tentativa de melhorar a inferência, e também, melhor avaliar os resultados obtidos.

O principal motivador, em se tratando de sequências de resíduos do tipo CDR3, é melhor avaliar as sequências que possuam inserções ou deleções mas que mantenham ainda boa semelhança com a sequência alvo. Esta característica, inclusive, levou-nos a experimentar a substituição da função de perda *SparseCategoricalCrossEntropy*, fornecida pelas bibliotecas de aprendizado de máquina, mas que apenas contempla substituições, por um escore baseado em similaridade.

Três exemplos podem ser vistos nas Tabelas 4.1, 4.2 e 4.3. É possível observar que nos exemplos detalhados nas Tabelas 4.1 e 4.3, métricas baseadas em similaridade são mais coerentes com os objetivos da pesquisa que a fornecida pela função *SparseCategoricalCrossEntropy*, a qual tem resultado melhor apenas no exemplo da Tabela 4.2.

4.2 Comparação de sequências

A comparação entre sequências é uma das operações de maior importância em biologia computacional. A semelhança entre duas sequências muito parecidas pode ser evidenciada alinhando-se as sequências, como no exemplo a seguir:

```
SELLSKINDMPITNDQKKLMSN        (4.1)
SELLSLINDMPITNDQKKLMS-
```

Tabela 4.1: Comparação entre epítopo esperado e epítopo inferido na amostra 48 do conjunto de teste `IEDB3_1`. Um espaço necessário para alinhamento é representado pelo símbolo '-' na coluna em vermelho. Apesar da semelhança das sequências, o erro calculado pela função *SparseCategoricalCrossEntropy* possui valor 12,062703 (relativamente alto), embora apenas uma coluna em doze está diferente, o que resulta em $1/12 = 0,0833$ erros por resíduo (relativamente baixo), isto é, menos que 10% de erro na sequência.

Sequências	Resíduos
Sequência Esperada	<lle-ldkwaslw>
Sequência Inferida	<lleeldkwaslw>

Tabela 4.2: Comparação entre epítopo esperado e epítopo inferido na amostra 60 do conjunto de teste `IEDB3_1`. Neste exemplo, *SparseCategoricalCrossEntropy* gera um bom resultado, pois os alinhamentos ótimos não têm espaços.

Sequências	Resíduos
Sequência Esperada	<pcrtct>
Sequência Inferida	<prrtct>

Tabela 4.3: Comparação entre epítopo esperado e epítopo inferido na amostra 19 do conjunto de teste `IEDB3_1`. O erro calculado pela função *SparseCategoricalCrossEntropy* possui valor 14,462871, enquanto que calculado por uma métrica baseada em erros por resíduo tem-se 0,818182, isto é, 81% de erro na sequência. É importante observar que o valor 0,818182 é maior que o valor real, no caso 0,40909, devido ao uso do algoritmo Levenshtein em Banda Limitada para otimização do cálculo de erros por resíduo.

Sequências	Resíduos
Sequência Esperada	<yqqyqdataeeeedfgeeae>
Sequência Inferida	<---------eeeedfgeeae>

Tabela 4.4: Valores para cálculo de similaridade, nas ocorrências de *MATCH* (resíduos iguais), *MISMATCH* (resíduos diferentes) e *GAP* (resíduo x espaço)

Constante	Valor
MATCH	+1
MISMATCH	0
GAP	−1

Observa-se que, neste caso, existem apenas duas diferenças (ressaltadas em vermelho): a substituição de um resíduo `K` por `L` e a existência de um **espaço** ao final da sequência, representado pelo símbolo '-'.

Em vários casos, a inserção de **espaços** é necessária para que se tenha sequências de um mesmo tamanho para serem alinhadas. Definimos um **alinhamento** como sendo a inserção de espaços em duas sequências a serem alinhadas de modo que fiquem do mesmo tamanho, porém, **espaços** não podem estar alinhados com **espaços**. Note que, esta restrição de não permitir alinhamentos de espaços com espaços faz com que o número total de alinhamentos entre duas sequências fixas seja finito.

Num alinhamento, podemos atribuir um valor para cada coluna, seguindo, por exemplo, as regras da Tabela 4.4. A interpretação desta tabela deve ser feita como segue. O valor *MATCH* deve ser usado para colunas que contém resíduos iguais. Observe que resíduos iguais é a situação mais desejada e ganham valor positivo. O valor *MISMATCH* deve ser usado para colunas que contém resíduos diferentes, que é uma situação razoável, mas não tão boa. Finalmente, o valor *GAP* deve ser usado para colunas que contém um espaço (necessariamente estas colunas deverão também conter uma símbolo pelas regras de alinhamento). Esta é a pior situação, e portanto é penalizada com valor negativo.

Os valores de *MATCH*, *MISMATCH* e *GAP* podem ser escolhidos com certa liberdade. Para melhores resultados é interessante escolher $MATCH > MISMATCH > GAP$. No nosso caso, decidimos fazer $MATCH = +1$ para que um alinhamento entre duas sequências iguais tenha pontuação igual ao tamanho destas sequências.

De posse de uma tabela como esta, podemos atribuir uma **pontuação** a cada alinhamento, somando os valores de todas as suas colunas. Por exemplo, para o Alinhamento 4.1, temos os valores abaixo totalizando uma pontuação de 19:

```
+1+1+1+1+1+1+0+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1-1
 S E L L S K I N D M P I T N D Q K K L M S N
 S E L L S L I N D M P I T N D Q K K L M S -
```

Um alinhamento entre duas sequências é chamado ótimo quando possui a maior pontuação possível entre todos os alinhamentos destas sequências. Como o número de alinhamentos entre duas sequências dadas é finito, sempre haverá pelo menos um alinhamento ótimo. Esta pontuação máxima é chamada de **similaridade** entre as sequências e denotada como $sim(s, t)$. Dependendo das sequências, podem existir vários alinhamentos ótimos. É importante lembrar que a **similaridade** é dependente dos valores definidos para *MATCH*, *MISMATCH* e *GAP*, como por exemplo, os definidos na Tabela 4.4.

A construção de alinhamentos ótimos é um problema largamente estudado em computação e de solução conhecida. Uma possível implementação do cálculo de similaridade está apresentada no Algoritmo 1.

4.2.1 Preenchimento (*padding*)

Durante o processo de treino dos modelos, especialmente quando é utilizado o treino por lotes (*batch*), as sequências de entrada devem ter um mesmo tamanho, sendo que, para isso, são completadas com um símbolo de preenchimento, em nosso caso, o símbolo '#' (*pad*), até atingir um certo tamanho pré-definido que deve ser maior do que o comprimento de todos os exemplos de treino e teste. Assim, novas combinações devem ser consideradas envolvendo o símbolo de preenchimento, como as apresentadas na Tabela 4.5.

Além disso, uma nova função *sim_pad* deve ser definida para tratar das novas combinações de modo que $sim(s,t) = sim_pad(s_pad, t_pad)$. Esta nova função de cálculo de similaridade, designada por $sim_pad(s_pad, t_pad)$, calcula a similaridade com suporte a símbolo *pad* ('#'), sendo s_pad e t_pad, respectivamente, as sequências originais s e t preenchidas com símbolos '#' até um tamanho pré-definido. Uma implementação desta nova função é apresentada no Algoritmo 4.

Algoritmo 4: Algoritmo de cálculo da similaridade entre duas sequências com suporte a preenchimento, baseado no algoritmo proposto por Setubal e Meidanis [27]. Por convenção, foi considerado que a primeira posição das sequências possui índice zero ([0]). As funções *gap*() e *match*() estão detalhadas no Algoritmo 5.

```
Result: similaridade entre as sequências s e t com preenchimento
Function sim_pad(s_pad,t_pad)
    m ← |s_pad|;
    n ← |t_pad|;
    a[0,0] ← 0;
    for i de 1 até m do
        a[i,0] ← a[i−1,0] + gap(s_pad[i−1])
    end
    for j de 1 até n do
        a[0,j] ← a[0,j−1] + gap(t_pad[j−1])
    end
    for i de 1 até m do
        for j de 1 até n do
            a[i,j] ← a[i−1,j−1] + match(s_pad[i−1], t_pad[j−1]);
            a[i,j] ← max(a[i,j], a[i−1,j] + gap(s_pad[i−1]);
            a[i,j] ← max(a[i,j], a[i,j−1] + gap(t_[j−1]));
        end
    end
    return a[m,n]
end
```

O conceito no desenvolvimento do Algoritmo 4 de cálculo de similaridade com preenchimento, utilizando a Tabela 4.5, considera o Algoritmo 1 de cálculo de similaridade originalmente proposto, reavaliando os valores de ***GAP***, ***MATCH*** e ***MISMATCH*** através das funções *gap*() e *match*(), de modo a atingir nosso objetivo de ter $sim(s,t) = sim_pad(s_pad, t_pad)$.

Algoritmo 5: Rotinas auxiliares para os Algoritmos 4 e 6 de cálculo de similaridade com suporte a preenchimento. Os valores das constantes *PAD, GAP, MATCH* e *MISMATCH* estão definidas na Tabela 4.5. Por convenção, foi considerado que a primeira posição das sequências possui índice zero ([0]).

```
Result: valores de gap e match considerando símbolo de preenchimento
Function gap(r)
    if r = '#' then
        return PAD
    else
        return GAP
    end
end
Function match(r,s)
    if r = s then
        if r = '#' then
            return PAD
        else
            return MATCH
        end
    else
        if r = '#' ou s = '#' then
            return GAP
        else
            return MISMATCH
        end
    end
end
```

As seguintes considerações nos convenceram de que a nova tabela proposta satisfaz nosso objetivo. A título de exemplo, consideremos duas sequências $s =$ `ATA` e $t =$ `TATAR`, e um alinhamento ótimo entre s_pad e t_pad, sendo s_pad e t_pad obtidos, respectivamente, com o preenchimento das sequências s e t com o símbolo '#', a partir do final das sequências até atingirem um tamanho pré-definido (n), neste caso, tamanho oito.

```
-1+1+1+1-1+0+0+0+0
 - A T A # # # # #
 T A T A R # - # #
```

Alinhamento ótimo entre s_pad **e** t_pad**, donde** $sim_pad(s_pad, t_pad) = +1$

Tabela 4.5: Possibilidades a considerar para cálculo de similaridade e valores para cálculo de similaridade, nas ocorrências de *match*, *mismatch*, *gap* e *pad*.

Possibilidade	Valor
resíduos iguais	*MATCH*
resíduos diferentes	*MISMATCH*
resíduo x espaço	*GAP*
resíduo x *pad*	*GAP*
espaço x *pad*	*PAD*
pad x *pad*	*PAD*

Constante	Valor
MATCH	+1
MISMATCH	0
GAP	−1
PAD	0

Removendo todos os '#' do alinhamento, completando a sequência mais curta com espaços e removendo possíveis colunas com espaço contra espaço, obtemos um alinhamento entre s e t com a mesma pontuação:

```
-1+1+1+1-1
 - A T A -
 T A T A R
```

Alinhamento entre s e t com mesma pontuação $= +1$

Consideremos agora um alinhamento ótimo entre as mesmas sequências s e t.

```
-1+1+1+1-1
 - A T A -
 T A T A R
```

Alinhamento ótimo entre s e t, donde $sim(s,t) = +1$

Preenchendo as sequências com $8 - |s|$ e $8 - |t|$ símbolos '#', respectivamente, e completando a sequência menor com '-', teremos um alinhamento entre s_pad e t_pad de igual pontuação:

```
-1+1+1+1-1+0+0+0+0+0
 - A T A - # # # # #
 T A T A R # # # - -
```

Alinhamento entre s_pad e t_pad com igual pontuação: $+1$

Estas ideias podem ser generalizadas, como mostra o teorema a seguir.

Teorema 1. *Sejam s e t duas sequências de aminoácidos, e s_pad e t_pad duas outras sequências construídas, respectivamente, a partir de s e t inserindo símbolos pad ('#'), ao final das sequências até um tamanho máximo n previamente fixado e maior que $|s|$ e $|t|$. Temos que:*

$$sim(s,t) = sim_pad(s_pad, t_pad) \tag{4.2}$$

para quaisquer s e t, com tamanhos menores que n, utilizando-se dos valores da Tabela 4.5.

Demonstração. Em primeiro lugar, considere um alinhamento ótimo A entre s_pad e t_pad, com o preenchimento das sequências s e t com o símbolo '#', no final até atingirem o tamanho pré-definido (n). A partir dele, podemos construir um alinhamento A' entre s e t como segue. Vamos remover todos os '#' do alinhamento A e, se alguma das duas sequências ficar mais curta que a outra após a remoção, completamos a mais curta com espaços até esta ficar com o tamanho da outra. Se, neste processo, acabarmos alinhando espaço contra espaço, removemos este tipo de coluna do alinhamento. Assim, obteremos um alinhamento A' entre s e t, pois todos os símbolos de pad_s e pad_t sobrevivem no alinhamento A', e na mesma ordem que têm em s_pad e t_pad, visto que apenas foram retirados '#' e talvez alguns espaços, mas nenhum resíduo.

Agora vejamos como se comparam as pontuações de A e A'. As colunas de A modificadas para formar A' podem ser dos seguintes tipos:

(1) *pad* vs *pad* : neste caso, a coluna foi retirada. O valor da coluna é zero, portanto sua retirada não modificou a pontuação.

(2) *pad* vs espaço: estas se transformaram em espaço vs espaço e foram eliminadas. Seu valor também é zero, de forma que não afetam a pontuação. Observe que aqui precisamos que o valor de *pad* vs espaço seja zero.

(3) *pad* vs resíduo: estas se transformaram em espaço vs resíduo. Neste caso, pela nossa escolha de pontuação envolvendo preenchimento , ambos os tipos de coluna têm o mesmo valor, GAP. Sendo assim, a pontuação do alinhamento não foi alterada.

A partir das análises destes tipos (1), (2) e (3), concluímos que A e A' têm a mesma pontuação. Como A é ótimo, sua pontuação corresponde a $sim_pad(s_pad, t_pad)$.

$$pontuacao(A) = sim_pad(s_pad, t_pad). \tag{4.3}$$

Por outro lado, A' é apenas um alinhamento entre os muitos que podem haver entre s e t, e a similaridade entre s e t é o máximo valor de alinhamento, portanto é maior ou igual à pontuação de A'.

$$pontuacao(A') \leq sim(s, t) \tag{4.4}$$

Chegamos então à conclusão de que:

$$sim(s, t) \geq sim_pad(s_pad, t_pad). \tag{4.5}$$

Em segundo lugar, consideremos agora um alinhamento ótimo B entre as sequências s e t, formado por duas sequências $s_o t$ e $t_o t$ obtidas de s e t, respectivamente, por inserção de espaços. Temos, pela definição de alinhamento que $|s_ot| = |t_ot|$. Agora, a partir de B, vamos construir um alinhamento entre B' de s_pad e t_pad como segue. Primeiramente, vamos eliminar do final da sequências s_ot e t_ot os símbolos '-' correspondentes a espaço. Note que isso ocorre apenas em uma sequência, pois colunas espaço vs espaço não são aceitas em alinhamentos. Então, preenchemos o final das sequências, respectivamente, com $n-|s|$ e $n-|t|$ símbolos '#'. Caso as sequências finais fiquem com tamanhos diferentes, preenchemos, a partir do final, a sequência menor com símbolos '-', correspondente a espaço, até que ambas fiquem com o mesmo tamanho. Assim, obteremos um alinhamento

B' entre s_pad e t_pad, pois todos os símbolos de s e t sobrevivem no alinhamento B', e na mesma ordem que têm em s e t, visto que apenas foram retirados espaços e inseridos '#' e talvez alguns espaços tenham retornado, mas nenhum resíduo foi alterado.

Neste processo, as colunas de B modificadas para formar B' podem ser dos seguintes tipos:

(1) resíduo vs espaço: neste caso, um espaço do final de uma sequência, ou foi substituído por *pad* ou voltou a ter valor de espaço que pode ter sido inserido no final do processo para deixar as sequências do mesmo tamanho. Como resíduo vs espaço e resíduo vs *pad* possui o mesmo valor, GAP, na Tabela 4.5, a pontuação de B' não se altera com relação a B.

(2) *pad* vs *pad* : neste caso, a coluna não existia e foi inserida devido ao preenchimento em ambas sequências. O valor da coluna é zero e não afeta a pontuação de B'.

(3) *pad* vs espaço: neste caso, a coluna não existia e foi inserida devido ao preenchimento e a complementação final com espaço. O valor da coluna é zero e também não afeta a pontuação de B'.

A partir de (1), (2) e (3), concluímos que B e B' têm a mesma pontuação. Como B é ótimo, temos que

$$pontuacao(B) = sim(s, t). \tag{4.6}$$

Sabemos que B' pode não ser ótimo, mas as sequências de B' possuem os mesmos resíduos de B e também os mesmos números de *pad* que ocorrem em s_pad e t_pad. Assim, podemos considerar que

$$pontuacao(B') \leq sim_pad(s_pad, t_pad) \tag{4.7}$$

o que nos leva a

$$sim(s, t) \leq sim_pad(s_pad, t_pad). \tag{4.8}$$

A partir das equações 4.5 e 4.8, podemos então concluir que

$$sim(s, t) = sim_pad(s_pad, t_pad), \tag{4.9}$$

validando os valores definidos na Tabela 4.5. □

As similaridades sim e sim_pad são calculadas pelos Algoritmos 1 e 4, respectivamente.

Todas as demais métricas de similaridade, detalhadas à frente, excetuando-se Levenshtein e Escore Levenshtein em Banda Limitada, sempre consideram a existência de preenchimento nas sequências de entrada.

4.2.2 Escore similar

A similaridade é um métrica interessante na comparação de sequência em termos absolutos, porém não é adequada para uso como função de perda, pois possui um valor proporcional ao comprimento da sequências além de não ser uma função decrescente com a similaridade entre as sequências.

Desta forma, definiu-se como função de perda, para uso nos treinos e validação, o $escore_similar_pad$ para uma sequência esperada t e uma sequência prevista p como:

$$escore_similar_pad(t, p) = 1 - \frac{sim_pad(t_pad, p_pad)}{|t|} \tag{4.10}$$

sendo t_pad a sequência esperada com preenchimento e p_pad a sequência inferida com preenchimento pelo modelo, ambas estendidas até um tamanho n.

Teorema 2. *Para quaisquer sequências t e p com tamanhos não ultrapassando n, temos que*

$$-||t| - |p|| \leq sim_pad(t_pat, p_pad) \leq min(|t|, |p|) - ||t| - |p||.$$

Demonstração. Sejam duas p_pad e t_pad, ambas de tamanho n. Um alinhamento simples que podemos considerar é emparelhar diretamente as duas sequências. Num pior caso, considerando que não temos nenhum caso de MATCH entre resíduos, a pontuação teria contribuição de $+0$ para as colunas resíduo vs resíduo e colunas *pad* x *pad*, e contribuição de -1 para as colunas resíduo vs *pad*, totalizando $-||t| - |p||$. Um alinhamento ótimo entre p_pad e t_pad terá pontuação maior a este valor.

No melhor caso, podemos considerar $p \subseteq t$ ou $t \subseteq p$. Neste caso o emparelhamento direto é ótimo e a pontuação formada por contribuições de $+1$ para resíduo vs resíduo, -1 resíduo vs *pad* e $+0$ para *pad* vs *pad*, totalizando $min(|t|, |p|) - ||t| - |p||$.

Consideremos um alinhamento genérico entre t_pad e p_pad. Neste alinhamento haverá três classes de colunas: resíduo vs resíduo iguais (classe A), resíduo vs espaço ou *pad* (classe B) e todas as demais (classe C). A pontuação do alinhamento é formada pela soma das pontuações calculadas em cada classe, pois as classes são disjuntas. Na classe A, cada coluna contribui com $+1$, que totalizam no máximo $min(|t|, |p|)$. Logo, $pontos(A) \leq min(|t|, |p|)$. Na classe B, cada coluna contribui com -1 e temos no mínimo $||t| - |p||$ colunas. Logo, $pontos(B) \leq -||t| - |p||$. Na classe C temos que $pontos(C) = 0$, pois cada coluna contribui com 0.

Isso é válido para todos os alinhamentos, inclusive os alinhamentos ótimos, o que resulta em:

$$\begin{aligned} sim_pad(t_pat, p_pad) &= pontos(A) + pontos(B) + pontos(C) \\ &\leq min(|t|, |p|) - ||t| - |p|| \end{aligned}$$

Assim, temos:

$$-||t| - |p|| \leq sim_pad(t_pat, p_pad) \leq min(|t|, |p|) - ||t| - |p||$$

□

Corolário 2.1.

$$1 + \frac{||t| - |p|| - min(|t|, |p|)}{|t|} \leq escore_similar_pad(t_pat, p_pad) \leq 1 + \frac{||t| - |p||}{|t|} \tag{4.11}$$

Demonstração. A partir de da equação 4.10 e do teorema 2, manipulações algébricas nos levam à conclusão desejada. □

Nossa experiência com o uso deste escore como função de perda não foi satisfatória, pois o tempo de treino aumentou demasiadamente, impedindo os estudos do modelo com 200 épocas. Por esse motivo, optamos por desenvolver e utilizar os algoritmos de similaridade e escore com banda limitada, detalhados nos itens seguintes.

4.2.3 Similaridade em banda limitada

A implementação de similaridade utilizando-se o Algoritmo 4 , apesar de adequada para a avaliação de sequências, tem uma desvantagem de consumir um tempo elevado, $O(|s|.|t|)$, principalmente quando utilizada no treino onde, devido ao treinamento em lotes, leva sempre à avaliação da similaridade considerando sequências de tamanho máximo devido ao uso de preenchimento (*padding*).

Como o resultado esperado do modelo é a inferência de uma sequência semelhante a uma sequência esperada, podemos considerar o uso de um algoritmo mais rápido, com abordagem heurística, como o de cálculo de similaridade em banda limitada, ou simplesmente $sim_banda(s, t, k)$, no contexto deste documento, sendo k um limite superior para a diferença de *gaps* nos alinhamentos entre s e t. Este algoritmo é baseado no Algoritmo **KBand** proposto por Setubal e Meidanis [27, Sec 3.3.4] e listado como Algoritmo 2. A abordagem heurística garante que $sim_banda(s, t, k) = sim(s, t)$ nos casos onde as sequências respeitam a banda k, isto é, todo o cálculo da similaridade é efetuada dentro da banda em relação a diagonal. Nos casos onde a banda seria violada, os resultados tendem a serem minimizados, isto é, $sim_banda(s, t, k) \leq sim(s, t)$.

Neste algoritmo, o cálculo exato está limitado a um número de espaços inseridos em uma das sequências, denominado k, mesmo que não consecutivos, até a inserção de uma espaço na outra sequência, e assim sucessivamente. Ultrapassando este limite, o valor calculado pelo algorítimo tende a reduzir o valor da similaridade. Por limitar os cálculos dentro de uma banda k de distância, a complexidade de execução é $O(k.|t|)$. Assim, para valores de k bem menores que $|t|$, o algoritmo é substancialmente mais rápido.

4.2.4 Similaridade em banda limitada com preenchimento

Como já explicado na Seção 4.2.1, o uso de algoritmos durante o treino deve considerar o suporte a símbolos de preenchimento (*pad*). Assim, o Algoritmo 2 também necessitou de adaptação para suporte a símbolos '#' e foi implementado conforme listado no Algoritmo 6.

O conceito utilizado foi o mesmo do Algoritmo 4, sendo reutilizadas as mesmas funções $gap()$ e $match()$, detalhadas no Algoritmo 5.

Pela abordagem heurística do algoritmos, temos que

$$sim_banda_pad(s, t, k) \leq sim_pad(s, t).$$

Algoritmo 6: Algoritmo de cálculo de similaridade entre as sequências *s_pad* e *t_pad* com preenchimento, limitando a k a diferença de *gaps* nos alinhamentos entre *s_pad* e *t_pad*. Os valores das constantes *PAD, GAP, MATCH* e *MISMATCH* estão definidas na Tabela 4.5 As funções *gap*() e *match*() estão detalhadas no Algoritmo 5. Por convenção, foi considerado que a primeira posição das sequências possui índice zero ([0]).

Result: melhor avaliação de similaridade entre as sequência s e t, considerando um limite de k para a diferença de *gaps* nos alinhamentos entre s e t.

Function *sim_banda_pad(s_pad,t_pad,k)*

```
m ← |s_pad|;
n ← |t_pad|;
/* Considera-se que os k primeiros elementos das sequências são
   diferentes de '#'                                              */
a[0,0] ← 0;
for i de 1 até m do
  a[i,0] ← a[i−1,0] + gap(s_pad[i−1])
end
for j de 1 até n do
  a[0,j] ← a[0,j−1] + gap(t_pad[j−1])
end
for i de 1 até m do
  for d de -k até k do
    j ← i + d;
    if 1 ≤ j ≤ n then
      a[i,j] ← a[i−1,j−1] + match(s[i−1],t[j−1]);
      if InsideStrip(i-1,j,k) then
        a[i,j] ← max(a[i,j], a[i−1,j] + gap(s[i−1]))
      end
      if InsideStrip(i,j-1,k) then
        a[i,j] ← max(a[i,j], a[i,j−1] + gap(t[j−1]))
      end
    end
  end
end
/* trata os casos quando a diferença entre o tamanho das
   sequências é maior que k                                       */
for j de m+k+1 até n do
  a[m,j] ← a[m,j−1] + gap(t[j])
end
for i de n+k+1 até m do
  a[i,n] ← a[i−1,n] + gap(s[j])
end
return a[m,n]
end
```

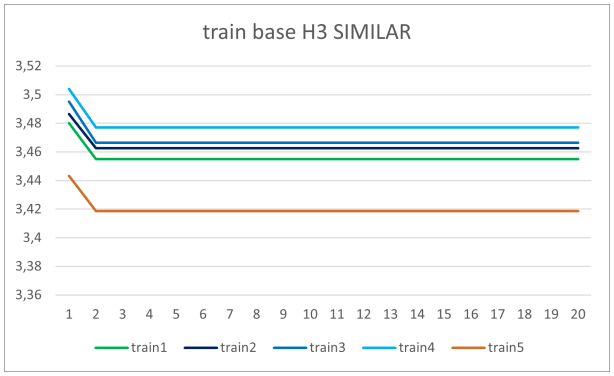
4.2.5 Escore similaridade com banda limitada

Seguindo os mesmos princípios e motivações da definição de Escore Similar, definiu-se:

$$escore_sim_banda_pad(t_pad, p_pad, k) = 1 - \frac{sim_banda_pad(t_pad, p_pad, k)}{sim_banda_pad(t_pad, t_pad, k)} \tag{4.12}$$

sendo t_pad a sequência esperada, p_pad a sequência inferida pelo modelo e k o limite superior para a diferença de *gaps* em alinhamentos entre s_pad e t_pad.

Os resultados do uso de $escore_sim_banda_pad(t_pad, p_pad, k)$ no treino estão apresentados na Figura 4.1. É possível observar que não ocorre aprendizado adequado pelo modelo com este escore, uma vez que o valor da função de perda estabiliza-se em um patamar alto, mesmo com a troca dos conjuntos de treino.



(a) treino

Figura 4.1: Resultados do treino utilizando escore similaridade com banda limitada, para base `IEDB3` e predição de **H3**. Os eixos X e Y referem-se, respectivamente, ao número de épocas e ao valor da função de erro (*SparseCategoricalCrossEntropy loss*)

Como os resultados não foram tão satisfatórios como esperávamos, voltamos a utilizar *SparseCategoricalCrossEntropy* como função de perda do modelo, e optamos por usar uma métrica baseada na distância de Levenshtein apenas para avaliação dos resultados, como está detalhado na próxima seção.

4.2.6 Levenshtein, Levenshtein em banda limitada e erros por resíduo

Como já explicado na Seção 4.1, o uso dos valores de *SparseCategoricalCrossEntropy* não são adequados para uma avaliação precisa das sequências, visto que uma única inserção poderia afetar significativamente o valor de comparativo a partir de *SparseCategorical-CrossEntropy* podendo gerar uma visão pessimista dos resultados.

Após as tentativas sem muito sucesso com as funções de perda baseadas em similaridade, buscou-se a identificação de um escore que, pelo menos, melhor avaliasse os resultados. Optou-se então por implementar um escore que representasse os erros por resíduo na inferência. Para isso foi desenvolvido o algoritmo de distância de Levenshtein em banda limitada, ou simplesmente $dist_leven_banda(s, t, k)$, e a partir deste um

escore que representa a proporção das diferenças sobre o comprimento da sequência, ou simplesmente $escore_leven_banda(s, t, k)$ definida como:

$$escore_leven_banda(t, p, k) = \frac{dist_leven_banda(t, p, k)}{|t|} \tag{4.13}$$

sendo t a sequência esperada, p a sequência inferida pelo modelo e k a maior diferença de *gaps* quando avaliado o melhor alinhamento entre t e p.

O algoritmo Levenshtein em banda limitada (Algoritmo 7) foi implementado usando a mesma abordagem utilizada por Setubal e Meidanis no Algoritmo 2 de similaridade em banda limitada, porém agora aplicada ao algoritmo de distância de Levenshtein (Algoritmo 3).

Os resultados do uso de $escore_leven_banda(t, p, k)$ são detalhados no Capítulo 5.

Algoritmo 7: Cálculo da distância de Levenshtein entre duas sequências em banda limitada. Por convenção, foi considerado que a primeira posição das sequências possui índice zero ([0]).

Result: distância de Levenshtein em banda limitada k entre as sequências s e t

Function *leven_ banda(s,t,k)*

```
m ← |s|;
n ← |t|;
for i de 0 até k do
    a[i, 0] ← i
end
for j de 0 até k do
    a[0, j] ← j
end
for i de 1 até m do
    for d de -k até k do
        j ← i + d;
        if 1 ≤ j ≤ n then
            if s[i-1] = t[j-1] then
                a[i, j] ← a[i − 1, j − 1]
            else
                a[i, j] ← a[i − 1, j − 1] + 1
            end
            if InsideStrip(i-1,j,k) then
                a[i, j] ← min(a[i, j], a[i − 1, j] + 1)
            end
            if InsideStrip(i,j-1,k) then
                a[i, j] ← min(a[i, j], a[i, j − 1] + 1)
            end
        end
    end
end
/* trata os casos quando a diferença entre o tamanho das
   sequências é maior que k                                  */
if n > m + k then
    a[m, n] ← a[m, m + k] + (n − m − k)
end
if m > n + k then
    a[m, n] ← a[n + k, m] + (m − n − k)
end
return a[m,n]
end
```

Capítulo 5

Modelo e avaliação

Neste capítulo são apresentados o modelo proposto e os ensaios realizados com os respectivos resultados.

5.1 Introdução

A tentativa de criação de anticorpos tem sido muito explorada, como pode ser visto nos diversos exemplos citados no Capítulo 2. Na quase totalidade dos casos analisados, as pesquisas envolvem modelos e manipulação de estruturas terciárias, isto é, estruturas 3D, o que aumenta significativamente a complexidade dos modelos e a exigência de recursos de processamento.

Nossa pesquisa buscou explorar uma abordagem pouco utilizada, baseada nas estruturas primárias, ou seja, apenas na sequência de aminoácidos. A estrutura primária é a base química da proteína, e uma das principais responsáveis pela formação de suas estruturas secundárias, e em grande parte por suas estruturas terciárias.

5.2 Abordagem do modelo proposto

Uma das principais características de modelos de aprendizado profundo é a capacidade de extrair as informações (características) relevantes dos dados, guiados pelo processo de treinamento. A hipótese da pesquisa é que seja possível extrair as informações relevantes exclusivamente da estrutura primária do epítopo, uma vez que estas são uma das principais responsáveis pelas características das estruturas secundárias e terciárias.

Assim, o modelo proposto deverá inferir, a partir da sequência de resíduos do epítopo, um par de sequências relativas ao CDR3 (CDRH3, CDRL3). A Tabela 5.1 mostra um exemplo.

Por se tratar de um modelo de sequência gerando sequência (`seq-2-seq`) [29], foram pesquisados modelos semelhantes, muitos dos quais estão relacionados ao processamento de linguagem natural. Alguns dos modelos que envolvem tradução (*NMT - Neural Machine Translation*), utilizam-se de uma arquitetura codificador-decodificador com atenção, como os modelos propostos por Luong [17] e Badhanau [3] e se mostram flexíveis e com aplicabilidade em nossa pesquisa, pois:

Tabela 5.1: Neste exemplo, o modelo deveria inferir as sequências CDRH3 e CDRL3 a partir do epítopo que pertence ao antígeno correspondente ao envelope de glicoproteína do Vírus 1 da Imunodeficiência Humana (HIV-1).

Epítopo	CDRH3	CDRL3
NWFDITNWLWYIR	AREGTTGWGWLGKPIGAFAH	QQYGQSLST

- relacionam uma sequência de palavras (frase em um idioma) a outra sequência de palavras (tradução da frase em outro idioma),
- um conjunto de palavras em um idioma mantém uma relação íntima, no caso de significado, com um conjunto de palavras no outro idioma,
- a relação entre palavras não necessariamente é única (palavras podem ter vários significados),
- a relação entre palavras não necessariamente é um-para-um, por exemplo, as duas palavras "turn off"se traduzem para uma única palavra "desligar", e
- palavras distantes podem manter influência, como as observadas em concordâncias de gênero ou número em português.

Uma abordagem semelhante foi avaliada por Akbar e colegas [1], com objetivo de previsão de resíduos e *motifs* (sequência de resíduos), tendo obtido resultados mais interessantes utilizando *motifs* que resíduos. Por estes resultados, avaliamos que valia a pena tentar o mesmo experimento com uma função de *loss* que entendesse distância de edição, pois, em casos de adição ou exclusão de um resíduo na inferência, a função de *loss* tipicamente utilizada, *Sparse Categorical Cross Entropy*, aumenta substancialmente apesar da distância de edição aumentar ou diminuir apenas uma unidade.

5.3 Modelo e ensaios

5.3.1 Modelo

Os modelos utilizados por Luong [17] e Bahdanau [3] para tradução, e Akbar e colegas [1] para inferência de resíduos ou *motifs* são modelos `seq-2-seq` baseados em RNNs com estrutura codificador-decodificador e camada de atenção. Nesta pesquisa, gostaríamos de prever uma sequência (CDR), a partir de outra sequência (epítopo), portanto, consideramos utilizar um modelo semelhante, isto é, um modelo `seq-2-seq` baseados em RNNs com estrutura codificador-decodificador e camada de atenção, cuja arquitetura está detalhada na Figura 5.1.

No modelo proposto, o codificador é formado por uma camada *embedding* e uma camada RNN. Durante a fase de treinamento, ambas as camadas sofrem aprendizado. O decodificador é formado por uma camada *embedding*, uma camada de atenção, uma camada RNN e uma camada *fully connected*. Durante o treino, a camada embedding do

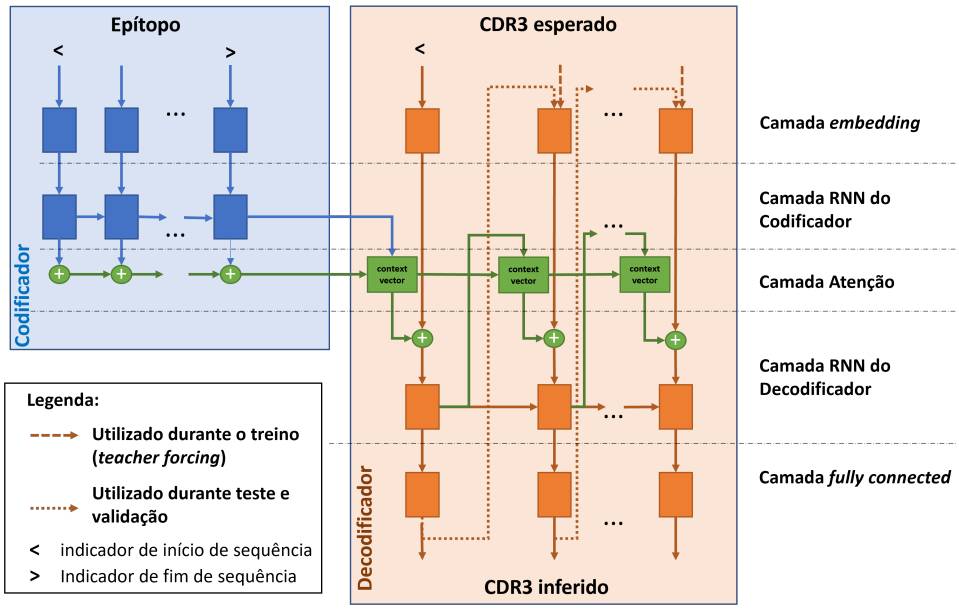


Figura 5.1: Arquitetura do modelo proposto

decodificador sempre recebe os valores esperados na saída correspondente. Este mecanismo é chamado de *teacher forcing* e acelera o processo de aprendizado da rede, e é utilizado apenas durante o treino do modelo. A camada de atenção é responsável pela geração do `vetor de contexto` que consolida uma representação dos dados de entrada para ser utilizada pela RNN como informação adicional para efetuar a inferência dos valores de saída do modelo.

Na implementação do modelo, a camada *embedding* pode ser escolhida entre uma camada também sujeita a aprendizado juntamente com o restante da rede, ou então uma codificação padrão *one-hot*. A camada de atenção também pode ser escolhida entre os modelos de Luong [17] ou Bahdanau [3].

A avaliação do modelo passou por quatro etapas:

- Avaliação inicial do modelo;
- Avaliação e aprimoramento do modelo e das bases;
- Tentativa de eliminação do *overfitting*
- Otimização dos hiper-parâmetros (*grid-search*)

Em todas as etapas, a avaliação foi efetuada utilizando-se a estratégia de *K-fold*, isto é, a partir de cada base definida, eram geradas dez bases de testes através da escolha aleatória de 80% dos dados para treino e 20% dos dados para teste. É importante observar que neste momento da avaliação, optou-se por não reservar parte dos dados para validação, pois gostaríamos primeiro de avaliar se o uso da arquitetura, inicialmente usada para tradução, seria aplicável e teria resultados satisfatórios na predição de CDRs ou de suas partes H3 e L3. Caso os dados se mostrassem suficientes e o modelo gerasse resultados satisfatórios, buscaríamos, então, resultados mais precisos através do treino e um ajuste fino de hiper-parâmetros com uma partição da base para treino e teste e então, efetuando a validação com a partição complementar.

5.3.2 Infraestrutura de treino e teste

Esta seção fornece informações gerais sobre o ambiente hardware e software utilizado no processamento dos ensaios de treino e teste do modelo.

O hardware utilizado possui 32 CPUs INTEL, 3 GPUs NVIDIA e 128G de memória RAM [1].

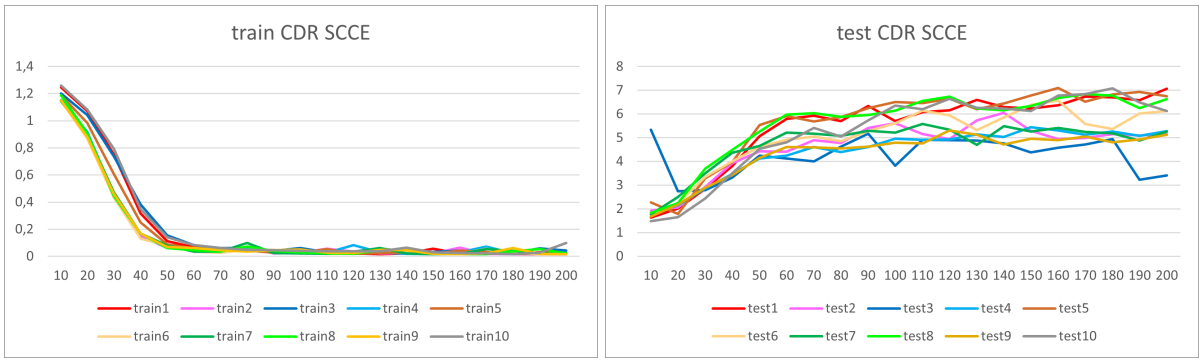
O modelo foi implementado em `Python` usando bibliotecas `Tensorflow` e `Keras`, em um ambiente `Docker` e sistema operacional `Linux`.

Maiores detalhes sobre o ambiente podem ser encontrados no Apêndice C.

5.3.3 Avaliação inicial do modelo com a base `IEDB`

Nesta etapa, o objetivo foi uma avaliação inicial do desempenho do modelo em termos de aprendizado e predição, considerando várias possibilidades de predição (CDR, H3, L3 e EPÍTOPO) e também com relação as características da base IEDB, avaliando se seriam suficientes para o aprendizado e predição satisfatória, e também se possuem alguma característica que poderia afetar o aprendizado e as predições.

Os resultados do treino e teste de cada caso são apresentados, respectivamente, para predição de CDR, Figura 5.2, para predição de H3, Figura 5.3, para predição de L3, Figura 5.4 e para predição de EPÍTOPO na Figura 5.5.



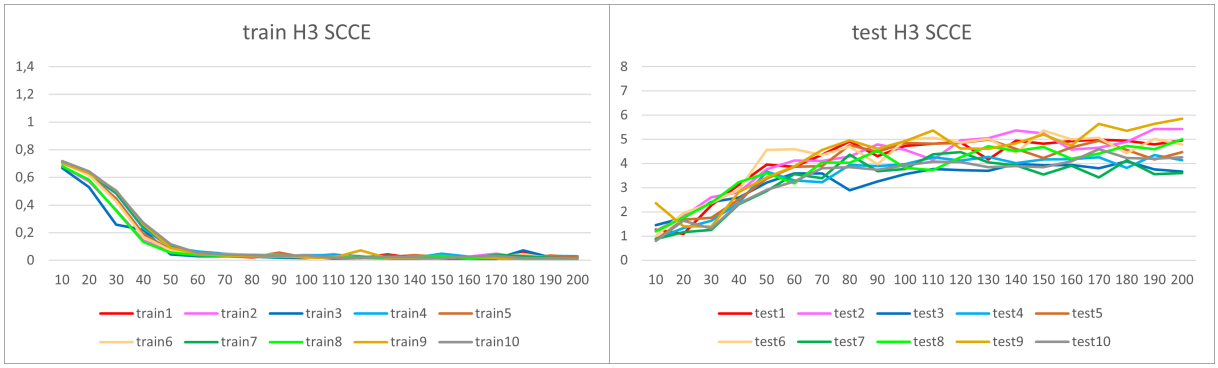
(a) treino (b) teste

Figura 5.2: Resultados do modelo para base `IEDB` e predição de **CDR**. O eixo X refere-se ao número de épocas e o eixo Y ao valor da função de erro (*SparseCategoricalCrossEntropy loss*)

Neste ensaio, os resultados de predição são melhores para L3, seguidos de H3 e CDR. L3 constitui-se, em geral, em sequências mais curtas, seguida por H3 e em por fim CDR que é composta do par H3 e L3. Por outro lado, do ponto de vista de aplicabilidade no contexto de imunologia, as preferências são, em ordem decrescente, CDR, H3 e L3, isto porque H3 é, reconhecidamente, a sequência determinante para o objetivo do anticorpo. Desta forma, no contexto desta base completa, a melhor escolha de predição seria H3 por possuir os melhores resultados com importância na determinação do anticorpo;

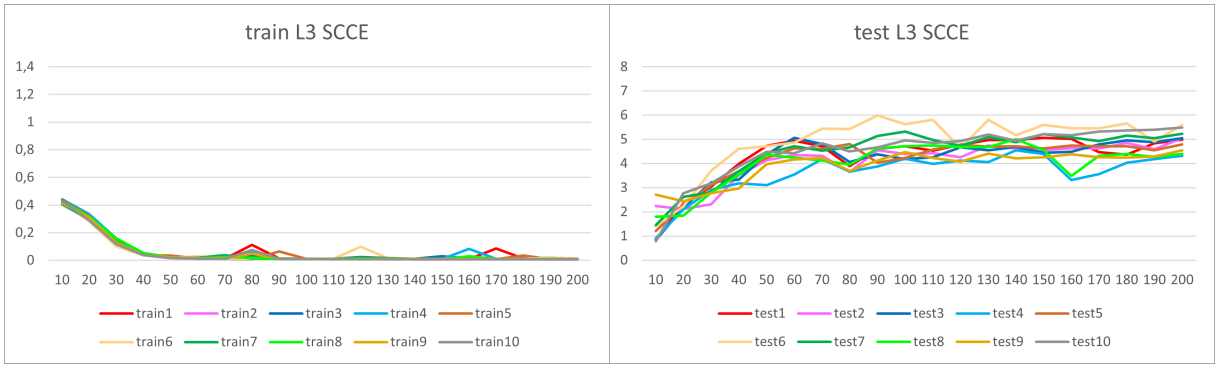
A partir dos resultados obtidos, e com a confirmação da existência de repetições de epítopos entre as bases de treino e testes, decidimos primeiramente melhorar os dados de

[1] Agradecimentos ao Projeto Fapesp 2018/00031–7 pela infraestrutura de processamento



(a) treino (b) teste

Figura 5.3: Resultados do modelo para base `IEDB` e predição de **H3**. O número de épocas está apresentado no eixo X e o valor da função de erro (*SparseCategoricalCrossEntropy loss*) no eixo Y



(a) treino (b) teste

Figura 5.4: Resultados do modelo para base `IEDB` e predição de **L3**. O número de épocas e o valor da função de erro (*SparseCategoricalCrossEntropy loss*) estão apresentados, respectivamente, nos eixos X e Y

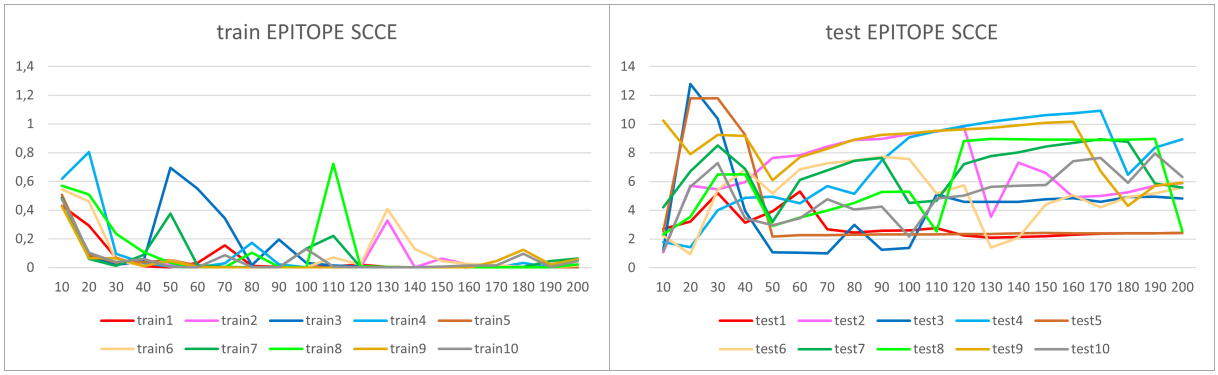
treino e teste, e posteriormente atacar questões como *overfitting*, se a generalização do modelo ainda permanecer baixa.

Assim, decidimos gerar uma nova nova base `IEBD2`, a partir de `IEDB`, inserindo um diferenciador nos epítopos repetidos. Isso busca evitar uma potencial degradação do treino evitando a existência de várias possibilidades de predição para uma mesma entrada (epítopo), ao mesmo tempo que evitamos reduzir a base que é considerada pequena. Os resultados desta tentativa são apresentados a seguir.

5.3.4 Avaliação modelo com a base `IEDB2`

A criação da base `IEDB2` tem como objetivo melhorar a predição com relação à base `IEDB`, mantendo a mesma quantidade de registros de entrada originais através da diferenciação dos registros com epítopos repetidos. Esta diferenciação, já explicada na Seção 3.2.4, visa evitar os problemas causados pelos registros com epítopos repetidos.

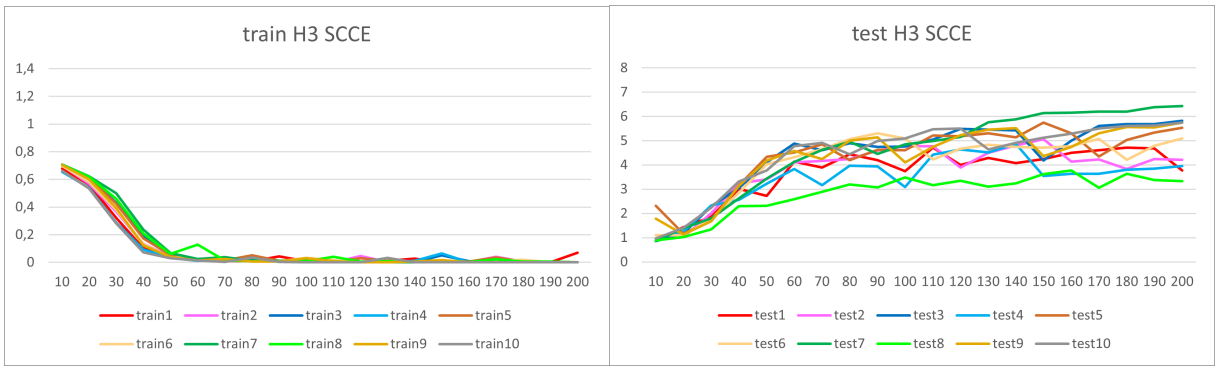
Os resultados do treino e teste de cada caso são apresentados, respectivamente, para predição de CDR, Figura A.5, para predição de H3, Figura 5.6, para predição de L3,



(a) treino (b) teste

Figura 5.5: Resultados do modelo para base `IEDB` e predição de **EPÍTOPO**. O eixo X refere-se ao número de épocas e o eixo Y ao valor da função de erro (*SparseCategoricalCrossEntropy loss*)

Figura A.7 e para predição de EPÍTOPO na Figura A.8.



(a) treino (b) teste

Figura 5.6: Resultados do modelo para base `IEDB2` e predição de **H3**. Os eixos X e Y referem-se, respectivamente, ao número de épocas e ao valor da função de erro (*SparseCategoricalCrossEntropy loss*)

A partir destes resultados e comparando-os com os resultados da base `IEDB` pode-se observar:

- As alterações inseridas na base `IEDB2` não produziram melhorias nos resultados. Os resultados de treino, um pouco melhores que no caso `IEDB`, são compensados por uma piora nos testes;
- Nota-se que, de maneira semelhante à base original, os registros oriundos de epítopos repetidos podem ser os responsáveis pela piora do resultado. Observe que eles podem aparecer em ambos os conjuntos de treino e de teste;
- Considerando o bom desempenho de treino e baixo desempenho de teste (baixa capacidade de generalização) temos caracterizado novamente *overfitting* no modelo.

Com base nestas observações, avaliamos ser melhor desconsiderar os registros com epítopos repetidos, evitando, tanto a potencial degradação causadas por registros com

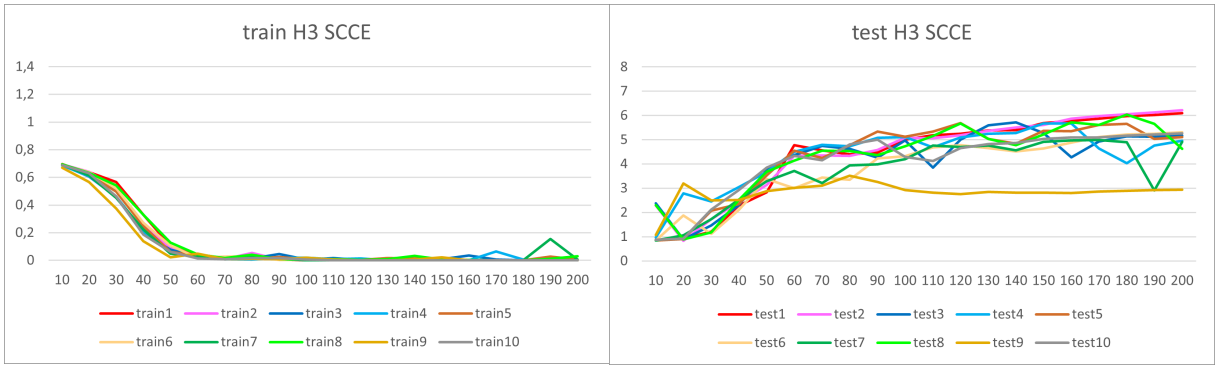
mesma entrada e saídas distintas (epítopos repetidos), quanto o impacto negativo nos resultados, quando se treina com uma sequência e espera-se outra nos testes (no caso de registros com epítopos repetidos que, mesmo com as alterações propostas pela `IEDB2`, ainda mantêm grande parte das sequências de entrada semelhantes).

Desta forma, uma nova base `IEDB3`, sem epítopos repetidos foi gerada para testes, com os resultados apresentados na próxima seção.

5.3.5 Avaliação modelo com a base `IEDB3`

Os testes com a base `IEDB3` visam avaliar se a remoção de epítopos repetidos pode auxiliar na melhoria do aprendizado e generalização do modelo.

Os resultados do treino e teste de cada caso são apresentados, respectivamente, para predição de CDR, Figura A.9, para predição de H3, Figura 5.7, para predição de L3, Figura A.11 e para predição de EPÍTOPO na Figura A.12.



(a) treino (b) teste

Figura 5.7: Resultados do modelo para base `IEDB3` e predição de **H3**. O número de épocas está apresentado no eixo X e o valor da função de erro (*SparseCategoricalCrossEntropy loss*) no eixo Y

A partir dos resultados de treino e teste com base na `IEDB3`, comparativamente com `IEDB` e `IEDB2`, podem ser observados:

- As alterações inseridas na base `IEDB3` também não produziram melhorias significativa nos resultados;
- Nota-se que, de maneira semelhante as bases `IEDB` e `IEDB2` foram obtidos bom desempenho de treino e baixo desempenho de teste (baixa capacidade de generalização) caracterizando novamente *overfitting* no modelo;
- Apesar da redução de registros, é possível notar uma pequena melhoria de desempenho do modelo, causada principalmente pela remoção do impacto negativo que registros baseados em epítopos repetidos tinham no resultado.

Com base nestas observações, a base `IEDB3` é, até o momento, a que apresenta melhores resultados e sem vícios (registros com epítopos repetidos). A ocorrência de *overfitting*

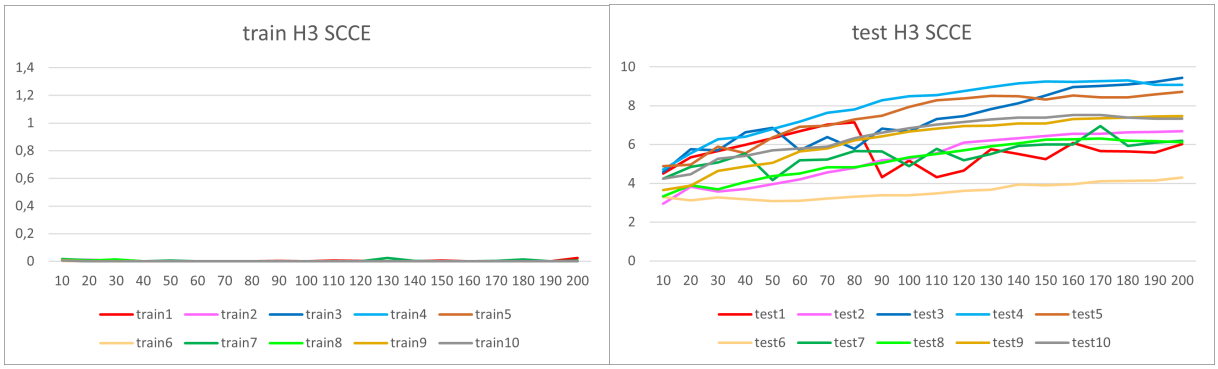
juntamente com o baixo número de registros sugere explorarmos a técnica *data augmentation* para melhorar os resultados.

Seguindo este caminho, foram propostos para avaliação, duas abordagens para *data augmentation*: inserção de mutação aleatória em um resíduo e inserção de mutação aleatória em vários resíduos. Para cada abordagem foram consideradas duas bases, alterando-se o número de registros gerados no processo de *data augmentation* para cada registro do conjunto de treino. É importante observar que estas abordagens foram aplicadas apenas nos conjuntos de treino, mantendo intactos os conjuntos de teste. Os resultados são analisados a seguir, na avaliação das bases `IEDB4`, `IEDB5`, `IEDB6` e `IEDB7`.

5.3.6 Avaliação modelo com a base `IEDB4`

Os testes com a base `IEDB4` visam avaliar a técnica de *data augmentation* sobre a base `IEDB3` gerando novos registros de treino a partir da aplicação de uma mutação aleatória em um resíduo escolhido aleatoriamente no epítopo. No caso deste ensaio, buscou-se um fator de aumento de 10 vezes, isto é, a geração de nove registros adicionais por registro de treino. Detalhes da geração da base `IEDB4` podem ser encontrados na Seção 3.2.6.

Os resultados do treino e teste de cada caso são apresentados, respectivamente, para predição de CDR, Figura A.13, para predição de H3, Figura 5.8, para predição de L3, Figura A.15 e para predição de EPÍTOPO na Figura A.16.



(a) treino (b) teste

Figura 5.8: Resultados do modelo para base `IEDB4` e predição de **H3**. O número de épocas e o valor da função de erro (*SparseCategoricalCrossEntropy loss*) estão apresentados, respectivamente, nos eixos X e Y

A partir dos resultados de treino e teste com base na `IEDB4`, comparativamente com `IEDB3`, podem ser observados:

- O fator de *augmentation* 10x utilizada na base `IEDB4` produziu um efeito de rápido aprendizado pelo modelo. Os valores de erro próximos de zero obtidos já a partir de dez épocas são compatíveis como os valores de cem épocas da base `IEDB3`;
- Nota-se piora significativa na generalização causada por acentuado *overfitting* no modelo;

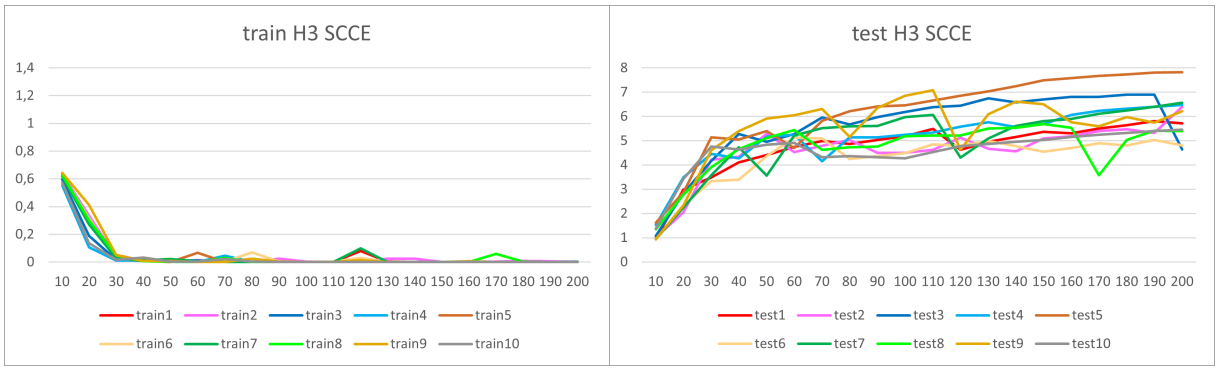
Com base nestas observações, a técnica de *data augmentation*, como aplicada, não gerou o efeito desejado, pois, apesar da melhora no treino houve significativa piora na generalização. Um dos motivos da piora pode ter sido o fator 10x utilizado. Desta forma, seria interessante um novo ensaio com um fator de *augmentation* menor.

Assim, uma nova base `IEDB5`, usando mutação em um único resíduo e fator 2x foi gerada e os resultados apresentados na próxima seção.

5.3.7 Avaliação modelo com a base `IEDB5`

Os testes com a base `IEDB5` visa avaliar a técnica de *data augmentation* sobre a base `IEDB3`, porém com um menor fator de aumento, devido a piora na generalização e *overfitting* avaliado no ensaio `IEDB4`. No caso deste ensaio, buscou-se um fator de aumento de apenas 2 vezes, isto é, a geração de um registro adicional, com mutação, por registro de treino. Detalhes da geração da base `IEDB5` pode ser encontrada na Seção 3.2.6.

Os resultados do treino e teste de cada caso são apresentados, respectivamente, para predição de CDR, Figura A.17, para predição de H3, Figura 5.9, para predição de L3, Figura A.19 e para predição de EPÍTOPO na Figura A.20.



(a) treino (b) teste

Figura 5.9: Resultados do modelo para base `IEDB5` e predição de **H3**. O eixo X refere-se ao número de épocas e o eixo Y ao valor da função de erro (*SparseCategoricalCrossEntropy loss*)

A partir dos resultados de treino e teste com base na `IEDB5`, comparativamente com as bases `IEDB3` e `IEDB4` , podem ser observados:

- O fator de *data augmentation* 2x utilizado na base `IEDB5` produziu um efeito de rápido aprendizado pelo modelo, compatível com fator aplicado, em relação a base `IEDB3`. Mais especificamente, o erro chega a aproximadamente zero, após 60 épocas com a base `IEDB3` e após 30 épocas coma base `IEDB5` ;
- Nota-se uma sensível piora na generalização e nenhum indício de redução do *overfitting* no modelo;

Com base nestas observações, conclui-se que a técnica de *data augmentation* utilizando-se uma mutação em um resíduo aleatório não produz melhoria na generalização do modelo.

Uma última alternativa a ser examinada é o uso de *data augmentation* aplicando-se um maior fator de mutação, por exemplo 25%. Assim, duas novas bases foram geradas para testes utilizando-se de mutação aleatória em 25% do resíduos do epítopo sendo os fatores de geração de registros 2x e 5x respectivamente para as bases `IEDB6` e `IEDB7`. Os resultados dos ensaios com estas duas novas bases são apresentados nas próximas seções.

5.3.8 Avaliação modelo com a base `IEDB6`

Este ensaio visa avaliar a técnica de *data augmentation* sobre a base `IEDB3`, porém com um maior fator de mutação, no caso 25%. O fator de aumento foi mantido baixo, no caso 2x, pois, até o momento, verifica-se que um maior fator tem efeito semelhante ao aumento de épocas e não uma real melhora do aprendizado. Detalhes da geração da base `IEDB6` podem ser encontrados na Seção 3.2.7.

Os resultados do treino e teste de cada caso são apresentados, respectivamente, para predição de CDR, Figura A.21, para predição de H3, Figura 5.10, para predição de L3, Figura A.23 e para predição de EPÍTOPO na Figura A.24.

(a) treino (b) teste

Figura 5.10: Resultados do modelo para base `IEDB6` e predição de **H3**. Os eixos X e Y referem-se, respectivamente, ao número de épocas e ao valor da função de erro (*SparseCategoricalCrossEntropy loss*)

A partir dos resultados de treino e teste com base na `IEDB6`, comparativamente com as bases `IEDB3` e `IEDB5` , podem ser observados:

- O fator de *data augmentation* 2x utilizado na base `IEDB5` produziu um efeito de rápido aprendizado pelo modelo, compatível com fator aplicado, em relação a base `IEDB3`. Mais especificamente, o erro chega a aproximadamente zero, após 60 épocas com a base `IEDB3` e após 30 épocas coma base `IEDB6`, de maneira semelhante ao ensaio com a base `IEDB5` ;
- Nota-se uma sensível piora na generalização e nenhum indício de redução do *overfitting* no modelo;

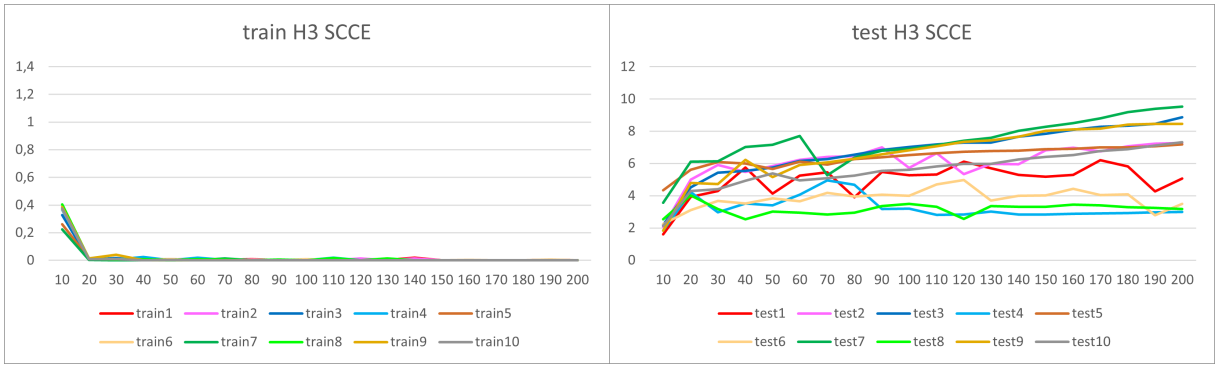
A partir destas observações, pode-se concluir que a técnica de *data augmentation* utilizando-se mutações aleatórias em 25% dos resíduos não conseguiu produzir melhoria na generalização do modelo.

Como um último ensaio ampliou-se o fator de geração de registros de x2 para x5 para avaliação. Os resultados estão discutidos na próxima seção.

5.3.9 Avaliação modelo com a base IEDB7

Os testes com a base IEDB7 visam avaliar a técnica de *data augmentation* com múltiplas mutações sobre a base IEDB3, com um maior fator de aumento, no caso 5x, para observamos se múltiplas mutações gera algum aprendizado adicional ou apenas é semelhante ao aumento de épocas de treino, algo já observado, entre as bases IEDB6 e IEDB7. Detalhes da geração da base IEDB7 pode ser encontrada na Seção 3.2.7.

Os resultados do treino e teste de cada caso são apresentados, respectivamente, para predição de CDR, Figura A.25, para predição de H3, Figura 5.11, para predição de L3, Figura A.27 e para predição de EPÍTOPO na Figura A.28.



(a) treino (b) teste

Figura 5.11: Resultados do modelo para base IEDB7 e predição de **H3**. O número de épocas está apresentado no eixo X e o valor da função de erro (*SparseCategoricalCrossEntropy loss*) no eixo Y

A partir dos resultados de treino e teste com base na IEDB7, comparativamente com as bases IEDB3 e IEDB6 , podem ser observados:

- O fator de *data augmentation* 5x utilizado na base IEDB7 produziu um efeito de rápido aprendizado pelo modelo, compatível com fator aplicado, em relação a base IEDB3, de maneira semelhante a base IEDB6 . Mais especificamente, o erro chega a aproximadamente zero, após 60 épocas com a base IEDB3 e após 20 épocas com a base IEDB7. Neste caso o fator de 5x não é observado diretamente porque os valores do gráfico são coletados apenas de dez em dez épocas;
- Nota-se uma sensível piora na generalização e, portanto, nenhum indício de redução do *overfitting* no modelo.

Com base nestas observações, conclui-se que a técnica de *data augmentation* utilizando-se mutações aleatórias em 25% dos resíduos não produz melhoria na generalização do modelo.

5.3.10 Tentativa de eliminação do *overfitting*

As seções anteriores avaliaram o modelo com parametrização padrão utilizando várias bases (`IEDB` a `IEDB7`) para treino sendo que, em todos os casos, o *overfitting* se mostrou presente. Nesta seção são apresentados algumas das tentativas de eliminação do *overfitting*, seguindo algumas das técnicas propostas por Goodfellow [11, Cap 11] e colegas.

Nesta avaliação, utilizamos a base `IEDB3`, pois foi a base mais simples (sem registros com epítopos repetidos) e que obteve os melhores resultados. Por outro lado, dentre as três possibilidades de predição, escolhemos H3 pois produzem melhores resultados que CDR e contém a porção de maior interesse do CDR.

As técnicas avaliadas foram:

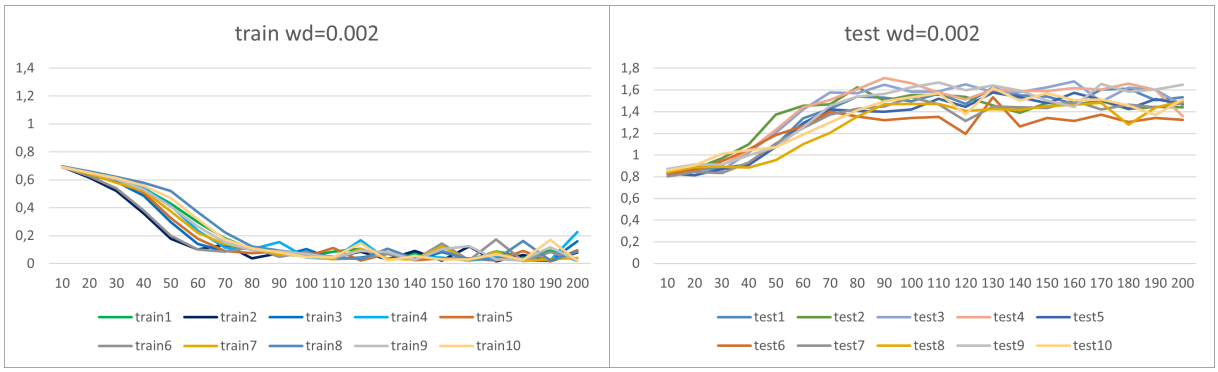
- *weight decay*;
- *data augmentation*;
- *dropout*;

A escolha das técnicas baseou-se na necessidade de eliminação do *overfitting* e também em técnicas que estivessem preferivelmente disponíveis para uso imediato na plataforma utilizada. Duas das técnicas escolhidas, *weight decay* e *dropout*, já existiam nativamente na plataforma, e *data augmentation* necessitou de implementação.

Tentativa de regularização com *weight decay*

A primeira técnica sugerida por Goodfellow [11, Cap 11] e colegas é *weight decay* e foi a primeira escolhida para ser testada. Cabe lembrar que *early stopping*, como já citado anteriormente, foi a primeira técnica avaliada, mas foi descartada, visto que os erros mínimos das inferências não se mostravam aceitáveis.

Utilizamos os valores 0.002, 0.005, 0.008 e 0.01 para o parâmetro *weight decay*. O resultado mais relevante foi obtido com valor 0.002, apresentado na Figura 5.12. Os demais resultados são apresentados no Apêndice B.1, Figuras B.2, B.3 e B.4.



(a) treino (b) teste

Figura 5.12: Resultados do modelo para base `IEDB3` com parâmetro *weight decay* = 0.002 e predição de **H3**. O número de épocas e o valor da função de erro (*SparseCategoricalCrossEntropy loss*) estão apresentados, respectivamente, nos eixos X e Y

Comprando-se os resultados dos ensaios com variação do parâmetro *weight decay*, em relação a base de referência `IEDB3`, podemos observar:

- o aumento no valor do parâmetro *weight decay* leva rapidamente a uma dificuldade de aprendizado do modelo, porém com certa sensível melhora nos resultados para os valores 0.002 e 0.005. Valores de 0.008 e 0.01 não permitem aprendizado do modelo dentro de 200 épocas;
- a melhora observada mostra-se ainda insuficiente, pois o erro nos testes ainda se mantém elevado.

Considerando a melhora obtida com parâmetro ajustado para 0.002, optou-se por usar este parâmetro como base para a avaliação dos outros métodos.

Tentativa de regularização com *data augmentation*

Uma outra técnica muito utilizada para regularização é *data augmentation*. Esta técnica já foi utilizada anteriormente, mas não com este objetivo, durante a definição e avaliação das bases de treino e validação nas Seções 3.2.6 e 3.2.7, como tentativa de aumento das bases de treino.

Nestes novos ensaios, a abordagem de *data augmentation* é diferente, e tem foco na avaliação da técnica como regularização, com objetivo de melhorar a generalização do modelo. Neste caso, a técnica foi aplicada, substituindo-se a camada *Embedding* do modelo por uma codificação *one-hot*, na qual aplicou-se um perturbação aleatória, para geração de diferentes codificações do epítopo e, consequentemente, aumentando os exemplos de treino.

A perturbação foi implementada adicionando-se uma parcela aleatória, para cada elemento da codificação *one-hot*, gerada através de distribuição uniforme entre, $-noise$ e $noise$, sendo $noise$ o parâmetro limite da perturbação. O fator de *data augmentation* utilizado foi cinco.

Os ensaios foram executados mantendo o parâmetro *weight decay* de melhor desempenho, em 0.002. Um ensaio de referência, a título comparativo, foi executado, sem perturbação ($noise = 0$) e fator de *data augmentation* $= 1$.

A avaliação comparativa dos erros de generalização dos ensaios com *data augmentation* é apresentada na Figura 5.13. Os resultados detalhados de cada ensaio são apresentados no Apêndice B.2, Figuras B.5 a B.9.

Pode-se observar que os resultados de referência ($noise = 0$ e fator de *data augmentation* $= 1$) são semelhantes ao modelo com camada *embedding* durante o treino, porém nota-se piora na generalização. Este resultado pode ser explicado parcialmente porque a camada *embedding* também sofre aprendizado no modelo original. O ensaio sem perturbação e com fator de *data augmentation* $= 5$, treinado até 100 épocas, também apresenta resultados semelhantes.

Para avaliar o modelo com *data augmentation* e perturbação, três valores para perturbação foram utilizados: 0.2, 0.4 e 0.5. Estes ensaios foram executados até 40 épocas. Nestes casos é possível observar que uma pequena melhora na generalização com o aumento da perturbação, porém com maior dificuldade no processo de treino.

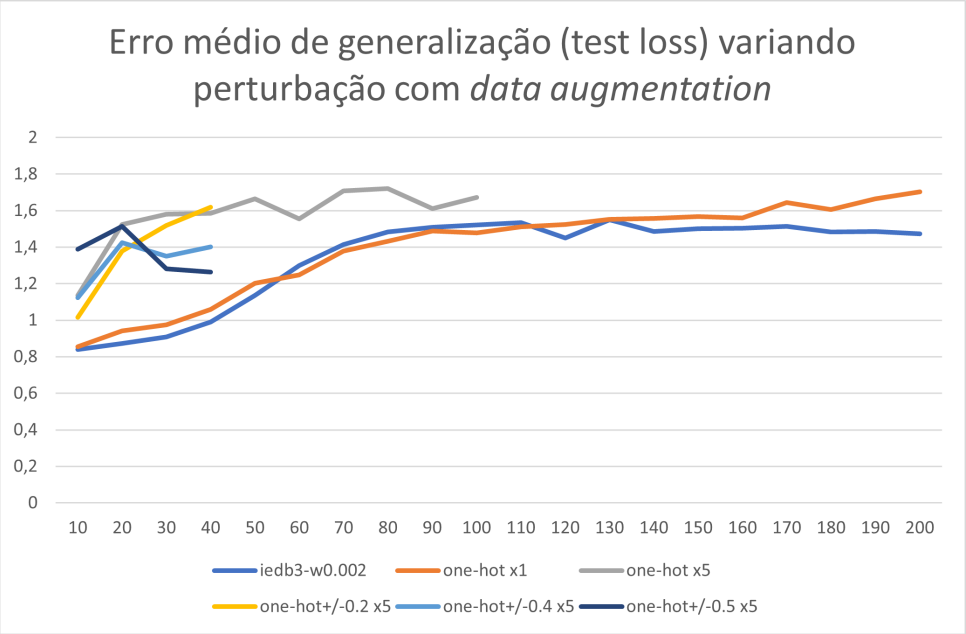


Figura 5.13: Comparativo dos resultados com uso de *data augmentation*, mantendo *weight decay* = 0.002. O eixo X refere-se ao número de épocas e o eixo Y ao valor da função de erro (*SparseCategoricalCrossEntropy loss*)

Por fim, a pequena melhoria na generalização não se mostra relevante para considerarmos esta técnica efetiva na melhoria dos resultados. Assim, como os resultados são praticamente semelhantes ao modelo com camada *embedding*, esta técnica não será utilizada nos próximos experimentos.

Tentativa de regularização com *dropout*

Uma nova tentativa de melhoria no modelo foi efetuada com a utilização de *dropout*. Por se tratar de um modelo com arquitetura de rede recorrente, existem dois pontos distintos para introdução de *dropout*: na entrada da rede, aqui denominado como *dropout in* e nas entradas dos elementos recorrentes, aqui chamado de *dropout rec*.

Inicialmente utilizou-se todas as combinações dos valores 0.0, 0.3 e 0.5 para os parâmetros *dropout in* e *dropout rec*, excetuando a combinação (0.0, 0.0), visto que esta corresponde à condição de referência já testada sem *dropout*. Por fim, foram utilizadas as combinações adicionais (0.0, 0.6), (0.6, 0.0) e (0.6, 0.6). Desta forma, um total de onze combinações foram avaliadas. Em todos os casos, o parâmetro *weight decay*, de melhor desempenho, também foi mantido em 0.002.

A avaliação comparativa dos erros de generalização dos ensaios com *dropout* é apresentada na Figura 5.14. Os resultados para cada combinação de parâmetros são apresentados no Apêndice B.3, Figuras B.12 a B.20.

Avaliando o gráfico comparativo na Figura 5.14 com os resultados dos ensaios, observa-se que a utilização de *weigth decay* e *dropout* reduzem consideravelmente o erro de generalização, quando comparado com o modelo de referência da Figura 5.7. Por outro lado, não consegue melhorar substancialmente a generalização quando comparada com o uso isolado de *weigth decay*.

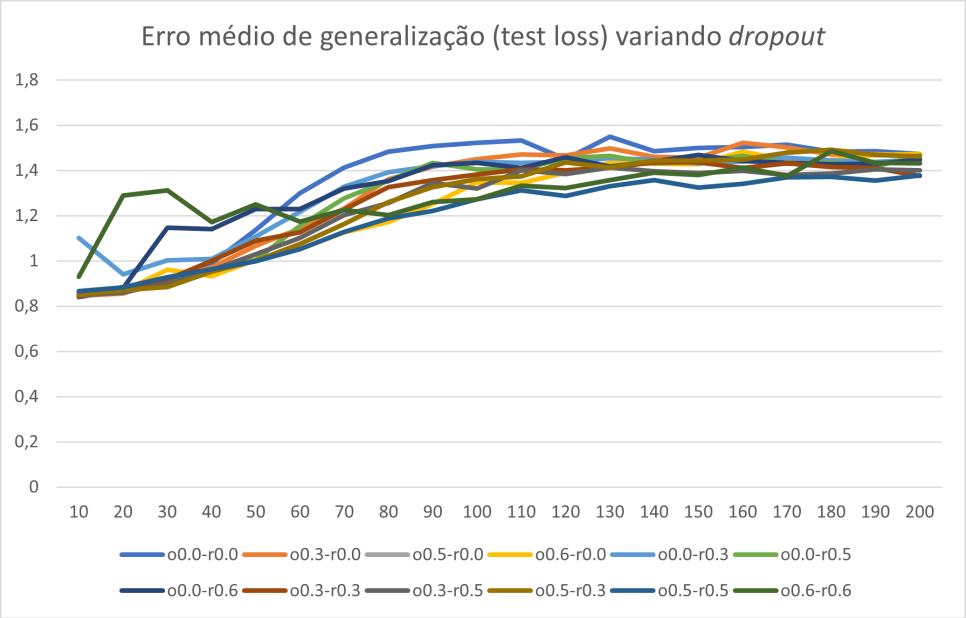


Figura 5.14: Comparativo dos resultados com a aplicação de *dropout*, mantendo *weight decay* = 0.002. Os eixos X e Y referem-se, respectivamente, ao número de épocas e ao valor da função de erro (*SparseCategoricalCrossEntropy loss*)

Assim, conclui-se que a aplicação de *dropout* não trouxe ganho significativo para os resultado. Observou-se também, como citado na documentação do Tensorflow, o uso de *dropout* nas entradas das unidades recorrentes (*dropout rec*) não permite o uso eficiente de paralelismo no processamento, o que aumenta substancialmente o tempo de treino dos modelos.

5.3.11 *Grid search* dos hiper-parâmetros

Nas seções anteriores o modelo foi avaliado em relação a várias bases (`IEDB` a `IEDB7`) e também algumas técnicas para eliminação de *overfitting*. Nesta seção, o objetivo será buscar uma combinação ótima de parâmetros através de um *grid search*. Serão utilizands dois dos parâmetros do modelo que mostraram algum grau de melhoria nas avaliações anteriores, e também alterações nos parâmetros de dimensões e unidades do modelo, parâmetros estes que alteram a sua capacidade, e que podem revelar caminhos para sua otimização. O estudo foi feito sobre a base `IEDB3` e com predição de H3, pelos mesmos motivos já relatados anteriormente.

O *grid search* será feito sobre as combinações dos parâmetros conforme Tabela 5.2.

No total são 108 combinações cujos resultados são apresentados nos gráficos das Figuras 5.15 e 5.16. Cada figura apresenta o erro (*SparseCategoricalCrossentropy*) para 100 e 200 épocas.

Os resultados do conjunto de testes apresentados na Figura 5.15 indicam uma redução na variação do valor do erro, calculado a partir da função *SparseCategoricalCrossentropy*, conforme os valores dos parâmetros *weight decay* e *drop out* são incrementados, porém o erro médio de generalização é reduzido para valores ligeiramente inferiores a um, o que é considerado um erro elevado, quando se observa os valores preditos em relação

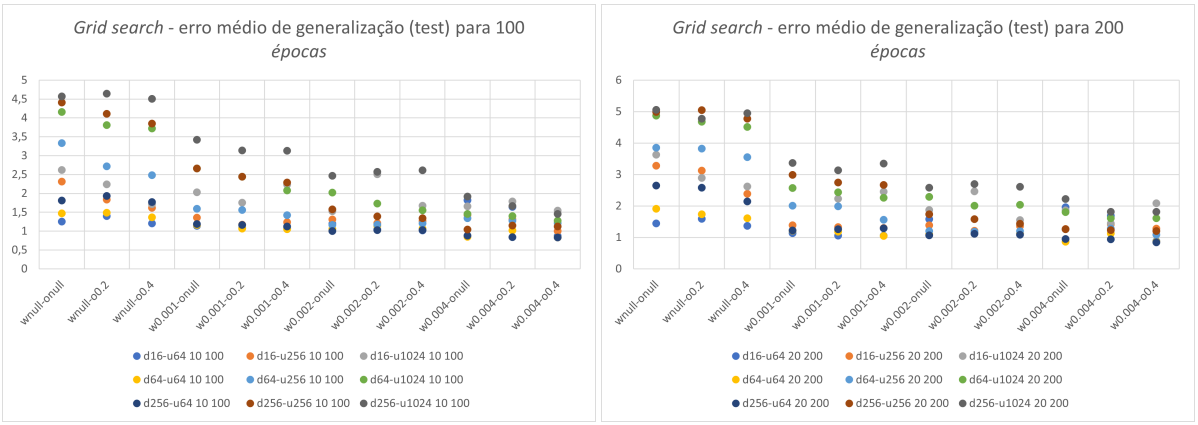
Tabela 5.2: Valores utilizados para cada parâmetro no *grid search*, totalizando $3 \times 3 \times 4 \times 3 = 108$ combinações.

Parâmetro	Valores
dimensões	256, 64, 16
unidades	1024, 256, 64
weight decay	nulo, 0.001, 0.002, 0.004
dropout in	nulo, 0.2, 0.4

aos esperados. Desta forma, com o *grid search* proposto não foi possível identificar um conjunto de parâmetros com resultados interessantes, dentro de um mínimo aceitável.

Na Figura 5.16 pode ser verificado que a diminuição do erro de generalização não resulta em real melhoria nos resultados preditos tendo, na verdade, um efeito contrário, visto que a média de erros por resíduo aumenta com o aumento no valor dos parâmetros *weight decay* e *drop out*.

Por fim, podemos concluir que as combinações propostas para *grid search* não foram suficientes para encontrar uma combinação com melhoria relevante da predição de H3.



(a) erro em 100 épocas

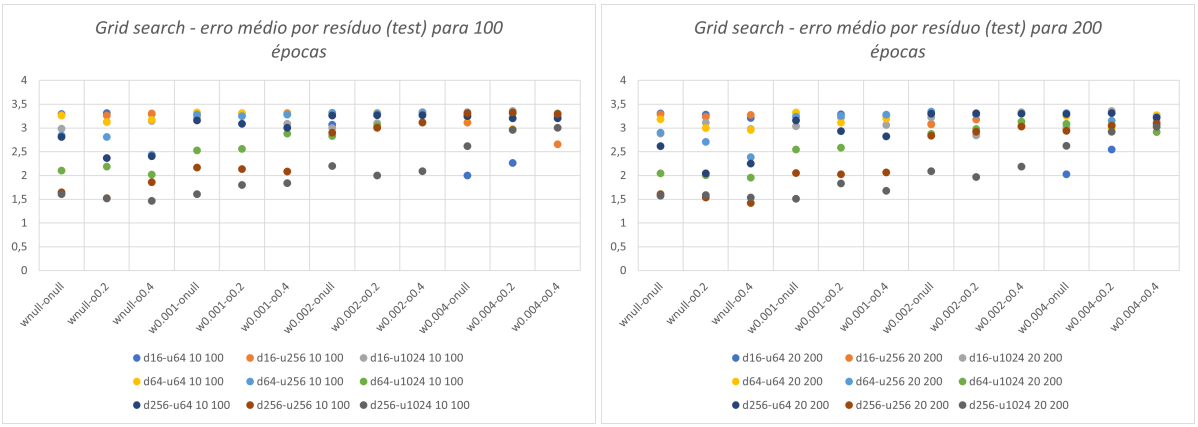
(b) erro em 200 épocas

Figura 5.15: Resultados do erro médio de generalização (test) durante *grid search*, modelo para base `IEDB3` e predição de H3

5.3.12 Análise quantitativa e comparativa dos resultados

Os resultados dos ensaios apresentados mostram que não foi possível alcançar um bom nível de generalização pelo modelo. Porém é fato que em vários dos casos de teste, o modelo fez a predição do H3 de maneira correta. A Tabela 5.3 mostra que para alguns dos conjuntos de dados utilizados, utilizando parametrização padrão, mais de 10% dos casos de teste foram previstos perfeitamente.

Em todos os casos onde a predição foi perfeita, existiam epítopos no grupo de treino associados ao mesmo H3. Isso mostra que o modelo foi capaz de avaliar a semelhança entre os epítopos nos grupos de treino e grupo de testes, associando o mesmo valor de H3. Existem casos onde uma única ocorrência de H3 no treino foi suficiente para a predição



(a) erro em 100 épocas (b) erro em 200 épocas

Figura 5.16: Erro médio por resíduo no teste durante *grid search*, modelo para base `IEDB3` e predição de H3

Tabela 5.3: Predições perfeitas H3 nos conjuntos de teste do `IEDB3` com 72 amostras cada.

Conjunto	Amostras	Percentual
`IEDB3_1`	9	12,5%
`IEDB3_2`	4	5,6%
`IEDB3_3`	5	6,9%
`IEDB3_4`	10	13,9%
`IEDB3_5`	3	4,2%
`IEDB3_6`	6	8,3%
`IEDB3_7`	3	4,2%
`IEDB3_8`	6	8,3%
`IEDB3_9`	0	0%
`IEDB3_10`	6	8,3%

perfeita. Por outro lado, nem todos os casos de epítopos semelhantes com mesmo H3 tiveram predição perfeita.

Akbar e colegas [1] conseguiram erros de 78% e 88% na predição de resíduos de contato no CDR, a partir dos resíduos de contato do epítopo. Em nosso modelo, o erro foi da mesma ordem de grandeza, variando entre 86.1% e 100% (complementos dos percentuais da Tabela 5.3), porém para predição da sequência completa H3 a partir da sequência completa do epítopo.

Capítulo 6

Contribuições e trabalhos futuros

6.1 Contribuições

A criação de anticorpos a partir de epítopos é de grande interesse e importância no tratamento de doenças como câncer e também, no caso de pandemias, como a COVID-19 que ainda estamos sofrendo. Uma das vantagens do uso de anticorpos é sua ação direta e mais rápida em casos graves de uma doença, onde, por exemplo, a vacina, caso exista, não tenha se mostrado eficaz.

Nesta pesquisa, buscamos avaliar a aplicação de aprendizado profundo, mais especificamente, RNNs, com objetivo de inferir a sequência do CDR de um anticorpo a partir da sequência de um epítopo, utilizando uma abordagem um pouco diferente da originalmente utilizada por Akbar e colegas [1]. Nosso trabalho teve o foco principal na avaliação da capacidade da rede em prever a sequência completa do CDRH3, sendo que foram experimentados vários recursos para melhoria na qualidade da inferência.

Inicialmente, buscou-se avaliar se a quantidade e qualidade dos dados eram suficientes. Neste sentido, verificamos que o número de registros de amostras disponíveis é pequena, suficiente para o treino, mas insuficiente para uma generalização. O número de pares (Epítopo, CDR) documentados é relativamente pequeno, sendo que o conjunto utilizado foi ainda mais reduzido pois optamos por utilizar apenas registros considerados positivos, isto é, com a confirmação de que o CDR é efetivo para o epítopo associado. Observamos ainda que vários dos registros possuíam epítopos, H3 e L3 repetidos. Os registros com epítopos repetidos foram removidos, pois a técnica de diferenciá-los com a inserção aleatória de um resíduo, piorou o resultado, o que reduziu um pouco mais o conjunto de dados de treino. O impacto da repetição de H3 e L3 nos registros não foi avaliada em profundidade, mas acreditamos que tem impacto negativo no aprendizado do modelo, pois não oferece diversidade para um melhor aprendizado, induzindo o modelo a não aprender aspectos relevantes da relação epítopo-CDR, mas apenas sugerir CDRs conhecidos a partir da semelhança dos epítopos, como foi observado, pois alguns epítopos do conjunto de teste permitiram a inferência do H3 correspondente com sucesso. Outro fator que pode ter atrapalhado o aprendizado é a existência de poucos casos de epítopos próximos (com grande similaridade) porém com os respectivos H3 e L3, com baixa similaridade.

Uma vez definido o conjunto de dados, passamos para uma avaliação e ajuste fino do modelo de rede neural proposto. Como resultado geral, identificamos uma situação

constante de *overfitting*. Avaliamos várias técnicas para melhorar a generalização do modelo, entre elas, *early stopping*, *weight decay*, *dropout* e *data augmentation*, sendo que uma pequena melhora foi observada com *weight decay*. Durante o uso de *data augmentation*, verificamos a dificuldade de uso desta técnica de maneira efetiva em nossos experimentos, pois é de conhecimento na área de imunologia que pequenas alterações no CDR podem afetar em muito a efetividade de acoplamento entre o epítopo e o CDR. Isso nos limitou a fazer experimentos de *data augmentation* apenas variando o epítopo, e que geraram resultados semelhantes ao aumento no número de épocas do treino. A não geração de novos CDRs com o processo de *data augmentation* limitava, consequentemente, a capacidade de aprendizado do modelo, pois não apresentava novos valores para o modelo aprender a essência que determina a sequência do CDR a partir do epítopo.

O uso de métricas como similaridade e Levenshtein é interessante, em casos como de nossa pesquisa, porque a inserção ou omissão de um resíduo em uma sequência, causando apenas um deslocamento de parte da sequência, pode ser considerada de menor relevância, comparativamente ao impacto que tal diferença causa, por exemplo, em funções do tipo *Sparse Categorical Cross Entropy*. A principal contribuição de nossos experimentos foi o desenvolvimento dos algoritmos de cálculo com preenchimento, com respectiva prova de equivalência com o algoritmo convencional, e o algoritmo de distância de Levenshtein em banda. Exemplos como os Algoritmos 4 e 6 foram desenvolvidos para uso como função de *loss* nos treinos em *batch* e também para avaliação dos resultados. Os algoritmos com cálculo em banda são mais rápidos, retornando o mesmo resultado que os algoritmos convencionais de cálculo, desde que os dados de entrada permitam que o cálculo fique restrito a um limite associado à banda utilizada. Caso este limite seja ultrapassado, os algoritmos de cálculo em banda não conseguem computar o valor otimizado, isto é, podem retornar, por exemplo, no caso de distância de edição, uma valor maior que a distância.

Os resultados apresentados parecem indicar que a base de dados utilizada é insuficiente para uma boa generalização. Além disso, verificamos que o uso de *data augmentation* não melhorou a generalização, o que indica que a técnica necessita de aprimoramento para que o uso seja efetivo neste contexto. Para finalizar, o uso da métrica baseada em similaridade, que inicialmente nos parecia mais interessante, não permitiu aprendizado adequado do modelo proposto, o que sugere que a arquitetura da rede necessita ser alterada para que o uso da métrica possa ser efetiva.

6.2 Trabalhos Futuros

Neste trabalho, podemos ressaltar algumas questões encontradas e que não foram tratadas, por questão de tempo e para não sairmos do objetivo inicialmente proposto.

O uso de *data augmentation* tem se mostrado de grande importância na melhoria dos resultados de modelos de aprendizado profundo. Em nossa tentativa de uso, foi possível verificar que uma abordagem diferente da convencional é necessária para que se consiga ampliar os dados de treino com casos realmente relevantes para o objetivo a ser aprendido pelo modelo, visto a quantidade relativamente reduzida e interdependente de exemplos para treino. Assim, abordagens utilizando matrizes de substituição PAM e BLOSUM,

ou até mesmo pequenas substituições entre aminoácidos muito semelhantes como, por exemplo, uma Alanina por uma Glicina, poderiam ser estudadas dentro do contexto de *data augmentation*.

Este conjunto de dados reduzido também sugere experimentos futuros com outros algoritmos voltados a poucos dados de treino como abordagens *one-shot learning* e *few-shot learning*. Outra opção interessante para avaliação futura é o uso de modelos baseados em *transformers* como, por exemplo, BERT, que são considerados estado da arte atualmente. Neste tipo de modelo, o aprendizado não supervisionado de proteínas em geral, cujos exemplos disponíveis são em maior quantidade que pares (epítopos-CDR) utilizados em nossa pesquisa, poderia ser especializado na aplicação de anticorpos, que são proteínas, e depois em sua relação com epítopos específicos. Modelos utilizando unidades LSTM e RNNs bidirecionais e outras abordagens de atenção também são opções que podem ser avaliadas futuramente.

Por fim, outro tema a ser abordado é a aplicação das matrizes de substituição PAM e BLOSUM no processo de *data augmentation*, nas unidades codificadoras de entrada do modelo e também nas métricas de avaliação dos aminoácidos inferidos pelo modelo. Isso daria, por exemplo, maior legitimidade às mutações utilizadas nos processos de *data augmentation* e maior flexibilidade na validação dos resíduos preditos pelo modelo.

Referências Bibliográficas

[1] Rahmad Akbar, Philippe A Robert, Milena Pavlović, Jeliazko R Jeliazkov, Igor Snapkov, Andrei Slabodkin, Cédric R Weber, Lonneke Scheffer, Enkelejda Miho, Ingrid Hobæk Haff, et al. A compact vocabulary of paratope-epitope interactions enables predictability of antibody-antigen binding. *Cell Reports*, 34(11):108856, 2021.

[2] Christof Angermueller, Tanel Pärnamaa, Leopold Parts, and Oliver Stegle. Deep learning for computational biology. *Molecular Systems Biology*, 12(7), 2016.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] Dror Baran, M Gabriele Pszolla, Gideon D Lapidoth, Christoffer Norn, Orly Dym, Tamar Unger, Shira Albeck, Michael D Tyka, and Sarel J Fleishman. Principles for computational design of binding antibodies. *Proceedings of the National Academy of Sciences*, 114(41):10900–10905, 2017.

[5] Helen Berman, Kim Henrick, and Haruki Nakamura. Announcing the worldwide protein data bank. *Nature Structural & Molecular Biology*, 10(12):980–980, 2003.

[6] Brett N Bowman, Paul R McAdam, Sandro Vivona, Jin X Zhang, Tiffany Luong, Richard K Belew, Harpal Sahota, Donald Guiney, Faramarz Valafar, Joshua Fierer, et al. Improving reverse vaccinology with a machine learning approach. *Vaccine*, 29(45):8156–8164, 2011.

[7] Chensi Cao, Feng Liu, Hai Tan, Deshou Song, Wenjie Shu, Weizhong Li, Yiming Zhou, Xiaochen Bo, and Zhi Xie. Deep learning and its applications in biomedicine. *Genomics, Proteomics & Bioinformatics*, 16(1):17–32, 2018.

[8] Thao T Doan, Roger Melvold, and Carl Waltenbaugh. *Imunologia Médica Essencial*. Guanabara Koogan, 2006.

[9] Mathieu Dondelinger, Patrice Filée, Eric Sauvage, Birgit Quinting, Serge Muyldermans, Moreno Galleni, and Marylène Stéphanie Vandevenne. Understanding the significance and implications of antibody numbering and antigen-binding surface/residue definition. *Frontiers in Immunology*, 9:2278, 2018.

[10] AD Ghate, BU Bhagwat, SG Bhosle, SM Gadepalli, and UD Kulkarni-Kale. Characterization of antibody-binding sites on proteins: development of a knowledgebase

and its applications in improving epitope prediction. *Protein and Peptide Letters*, 14(6):531–535, 2007. Acesso à `http://bioinfo.net.in/AgAbDb.htm` em 25-12-2019.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[12] Jian-Hua Huang, Ming Wen, Li-Juan Tang, Hua-Lin Xie, Liang Fu, Yi-Zeng Liang, and Hong-Mei Lu. Using random forest to classify linear B-cell epitopes based on amino acid properties and molecular features. *Biochimie*, 103:1–6, 2014.

[13] Marie-Paule Lefranc. Immunoglobulin and T cell receptor genes: IMGT® and the birth and rise of immunoinformatics. *Frontiers in Immunology*, 5:22, 2014.

[14] Marie-Paule Lefranc, Véronique Giudicelli, Patrice Duroux, Joumana Jabado-Michaloud, Géraldine Folch, Safa Aouinti, Emilie Carillon, Hugo Duvergey, Amélie Houles, Typhaine Paysan-Lafosse, et al. IMGT®, the international ImMunoGeneTics information system® 25 years on. *Nucleic Acids Research*, 43(D1):D413–D422, 2015. Acesso a `https://www.imgt.org` em 26-12-2019.

[15] Ge Liu, Haoyang Zeng, Jonas Mueller, Brandon Carter, Ziheng Wang, Jonas Schilz, Geraldine Horny, Michael E Birnbaum, Stefan Ewert, and David K Gifford. Antibody complementarity determining region design using high-capacity machine learning. *bioRxiv*, page 682880, 2019.

[16] Claus Lundegaard, Ole Lund, and Morten Nielsen. Prediction of epitopes using neural network based methods. *Journal of Immunological Methods*, 374(1-2):26–34, 2011.

[17] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[18] Swapnil Mahajan, Randi Vita, Deborah Shackelford, Jerome Lane, Veronique Schulten, Laura Zarebski, Martin Closter Jespersen, Paolo Marcatili, Morten Nielsen, Alessandro Sette, et al. Epitope specific antibodies and T cell receptors in the Immune Epitope Database. *Frontiers in Immunology*, 9, 2018.

[19] Polina Mamoshina, Armando Vieira, Evgeny Putin, and Alex Zhavoronkov. Applications of deep learning in biomedicine. *Molecular Pharmaceutics*, 13(5):1445–1454, 2016.

[20] Derek M Mason, Simon Friedensohn, Cédric R Weber, Christian Jordi, Bastian Wagner, Simon Meng, Pablo Gainza, Bruno Correia, and Sai T Reddy. Deep learning enables therapeutic antibody optimization in mammalian cells by deciphering high-dimensional protein sequence space. *bioRxiv*, page 617860, 2019.

[21] Dan Ofer, Nadav Brandes, and Michal Linial. The language of proteins: Nlp, machine learning & protein sequences. *Computational and Structural Biotechnology Journal*, 2021.

[22] Yanay Ofran, Avner Schlessinger, and Burkhard Rost. Automated identification of complementarity determining regions (CDRs) reveals peculiar characteristics of CDRs and B cell epitopes. *The Journal of Immunology*, 181(9):6230–6235, 2008.

[23] RJ Pantazes and Costas D Maranas. OptCDR: a general computational method for the design of antibody complementarity determining regions for targeted epitope binding. *Protein Engineering, Design & Selection*, 23(11):849–858, 2010.

[24] Roshan Rao, Joshua Meier, Tom Sercu, Sergey Ovchinnikov, and Alexander Rives. Transformer protein language models are unsupervised structure learners. In *International Conference on Learning Representations*, 2020.

[25] Nimrod D Rubinstein, Itay Mayrose, and Tal Pupko. A machine-learning approach for predicting B-cell epitopes. *Molecular Immunology*, 46(5):840–847, 2009.

[26] Inbal Sela-Culang, Yanay Ofran, and Bjoern Peters. Antibody specific epitope prediction — emergence of a new paradigm. *Current opinion in virology*, 11:98–102, 2015.

[27] Joao Carlos Setubal and Joao Meidanis. *Introduction to Computational Molecular Biology*. PWS Publishing Company, Boston, 1997.

[28] Ruth E Soria-Guerra, Ricardo Nieto-Gomez, Dania O Govea-Alonso, and Sergio Rosales-Mendoza. An overview of bioinformatics tools for epitope prediction: implications on vaccine development. *Journal of Biomedical Informatics*, 53:405–414, 2015.

[29] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*, 2014.

[30] Jesse Vig, Ali Madani, Lav R Varshney, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. Bertology meets biology: Interpreting attention in protein language models. *arXiv preprint arXiv:2006.15222*, 2020.

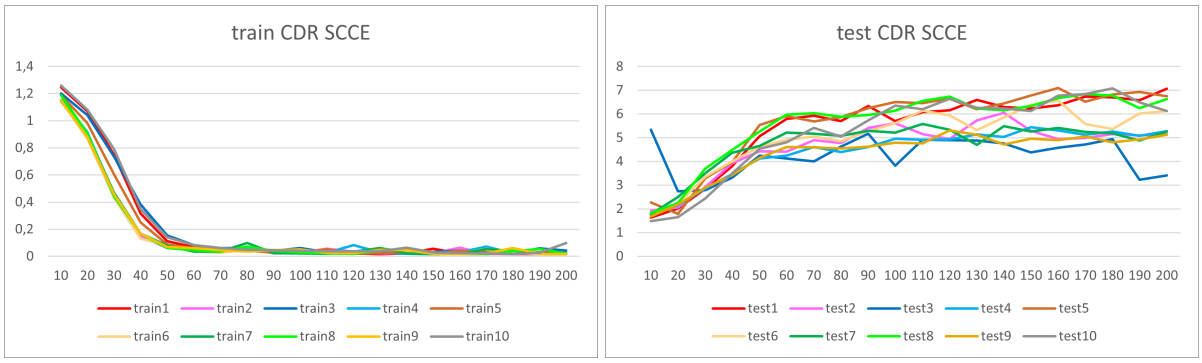
[31] Randi Vita, Swapnil Mahajan, James A Overton, Sandeep Kumar Dhanda, Sheridan Martini, Jason R Cantrell, Daniel K Wheeler, Alessandro Sette, and Bjoern Peters. The immune epitope database (iedb): 2018 update. *Nucleic Acids Research*, 47(D1):D339–D343, 2019. Acesso à `http://www.iedb.org/home_v3.php` em 27-08-2019.

[32] Robert A Wagner and Michael J Fischer. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173, 1974.

Apêndice A

Resultados dos testes iniciais para CDR, L3 e EPITOPE

NOTA: alguns resultados, já apresentados no texto, são reapresentados neste apêndice para facilitar a avaliação comparativa entre os diversos ensaios. Em todos os gráficos, o eixo X refere-se ao número de épocas e o eixo Y ao valor da função de erro (*SparseCategoricalCrossEntropy loss*).

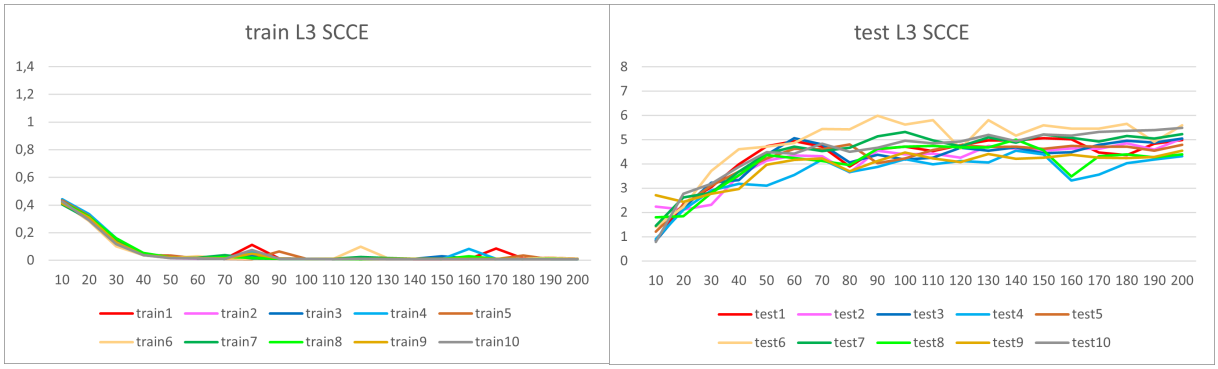


(a) treino (b) teste

Figura A.1: Resultados do modelo para base `IEDB` e predição de **CDR**

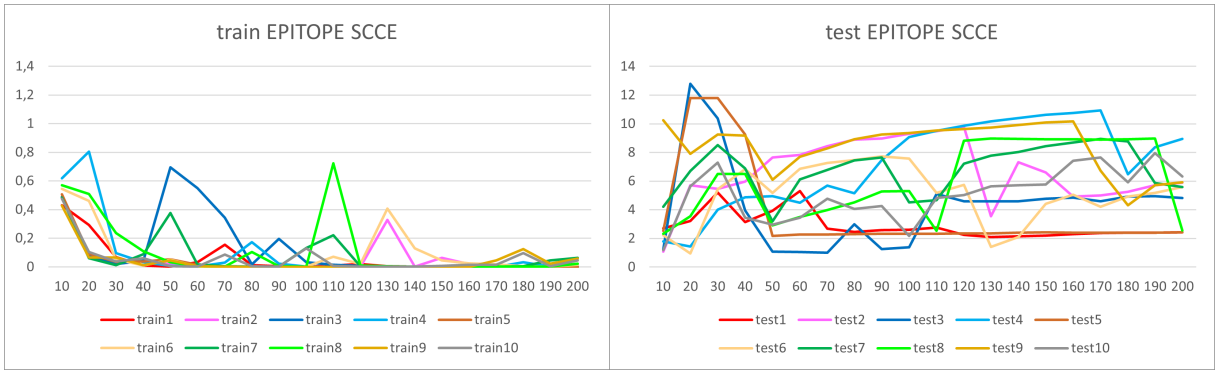
(a) treino (b) teste

Figura A.2: Resultados do modelo para base `IEDB` e predição de **H3**



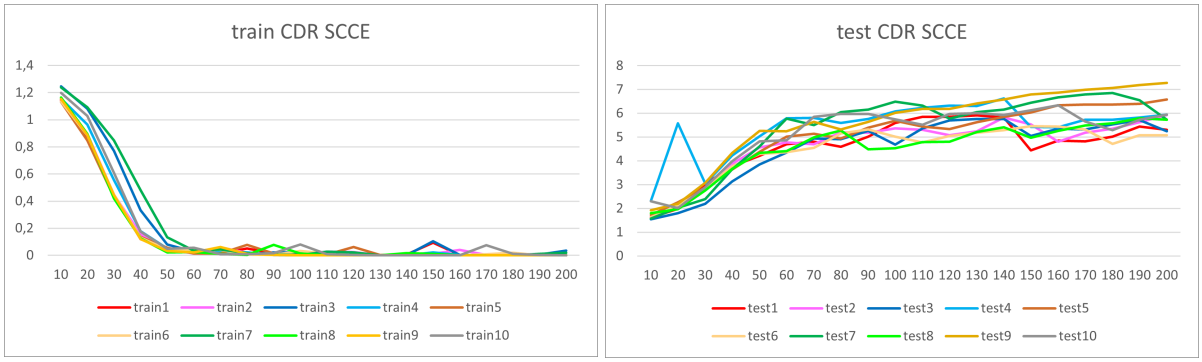
(a) treino (b) teste

Figura A.3: Resultados do modelo para base `IEDB` e predição de **L3**



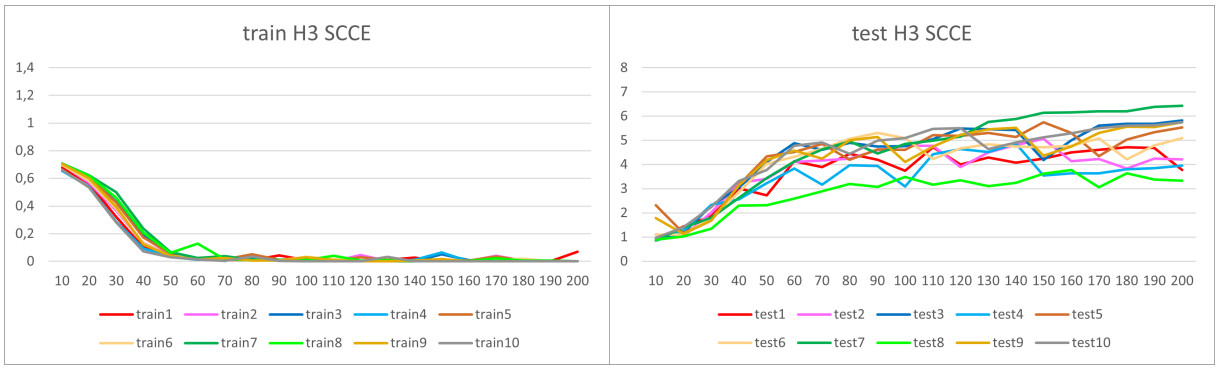
(a) treino (b) teste

Figura A.4: Resultados do modelo para base `IEDB` e predição de **EPÍTOPO**



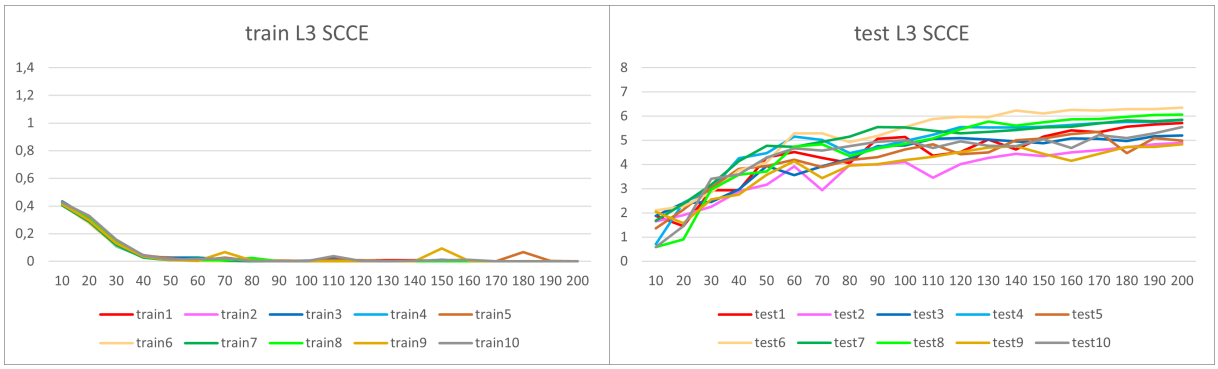
(a) treino (b) teste

Figura A.5: Resultados do modelo para base `IEDB2` e predição de **CDR**



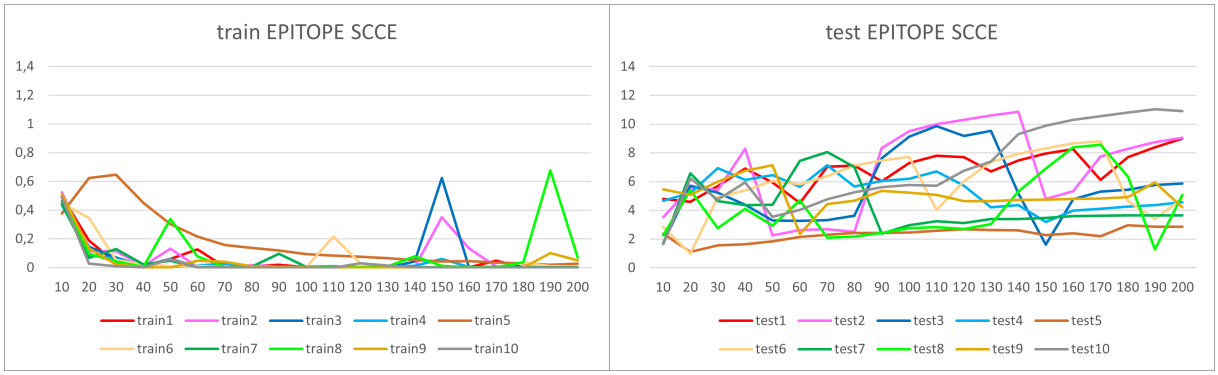
(a) treino (b) teste

Figura A.6: Resultados do modelo para base `IEDB2` e predição de **H3**)



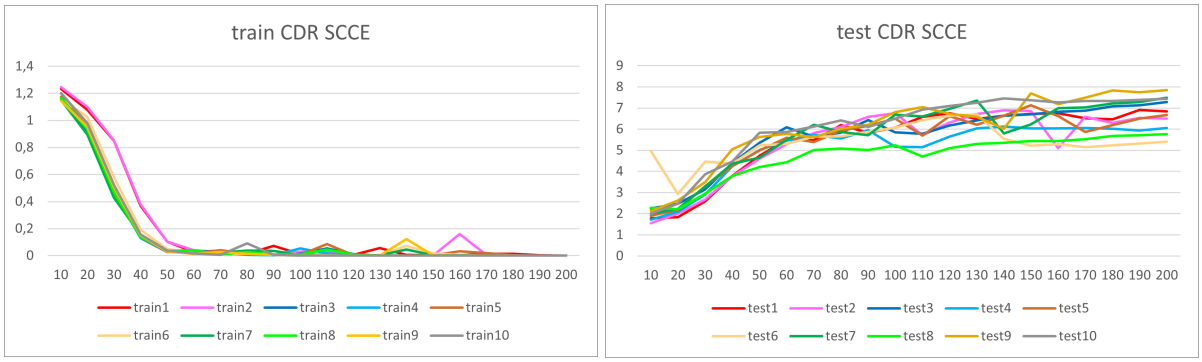
(a) treino (b) teste

Figura A.7: Resultados do modelo para base `IEDB2` e predição de **L3**



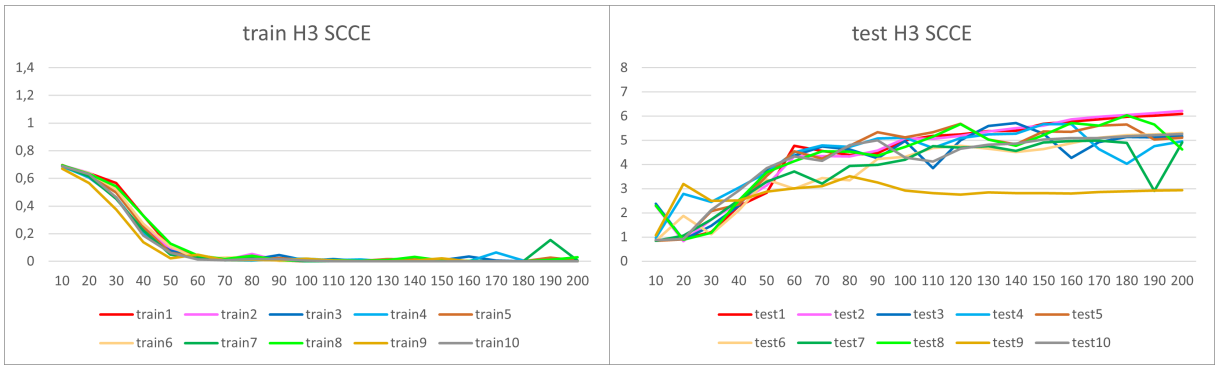
(a) treino (b) teste

Figura A.8: Resultados do modelo para base `IEDB2` e predição de **EPÍTOPO**



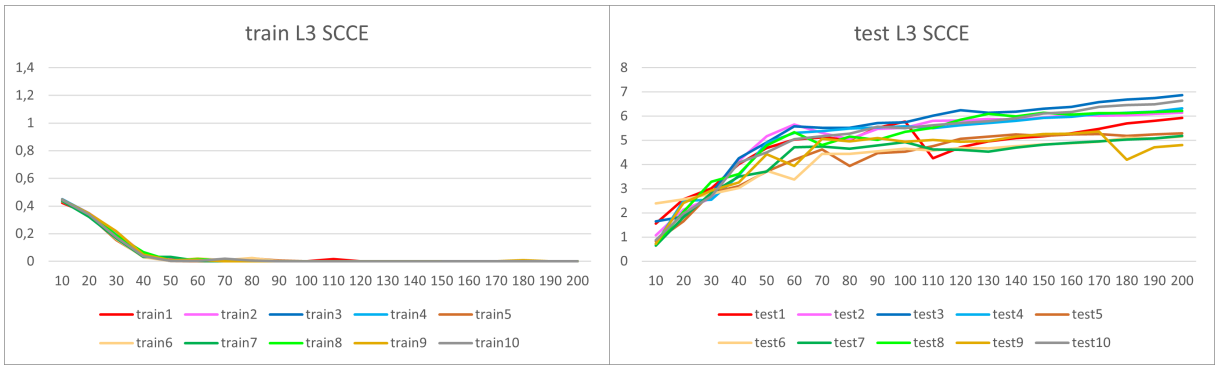
(a) treino (b) teste

Figura A.9: Resultados do modelo para base `IEDB3` e predição de **CDR**



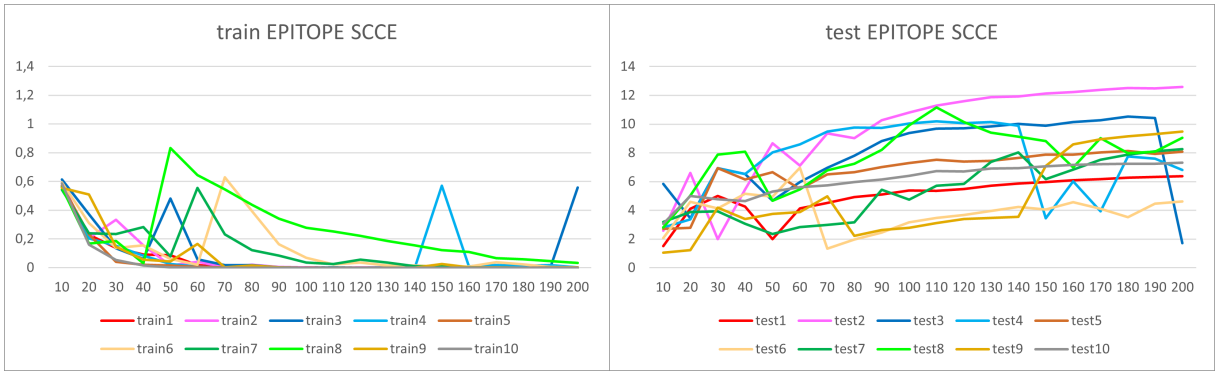
(a) treino (b) teste

Figura A.10: Resultados do modelo para base `IEDB3` e predição de **H3**



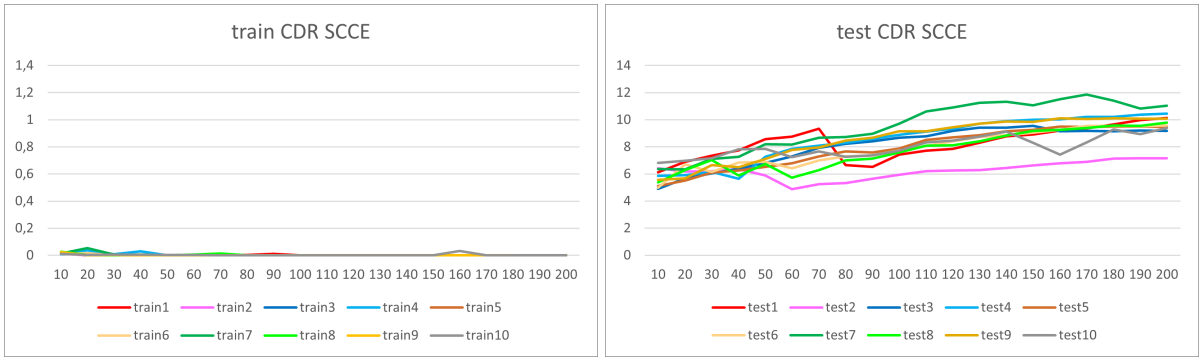
(a) treino (b) teste

Figura A.11: Resultados do modelo para base `IEDB3` e predição de **L3**



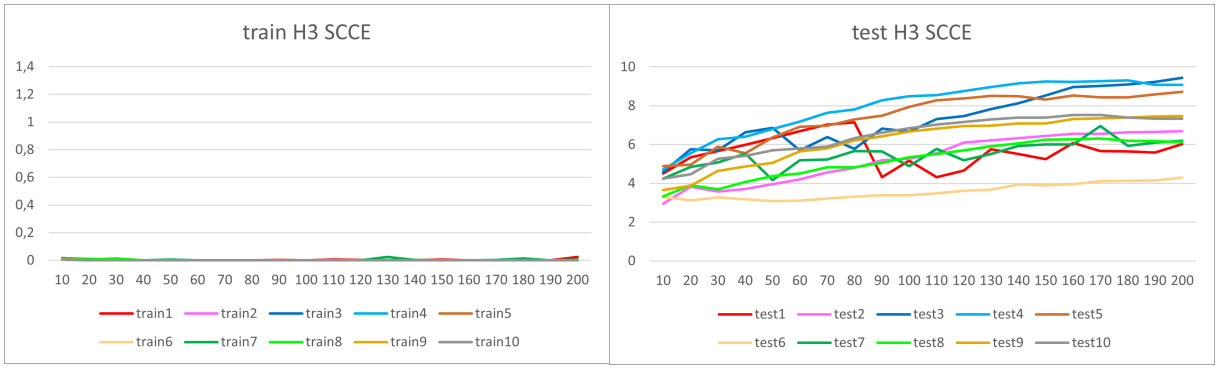
(a) treino (b) teste

Figura A.12: Resultados do modelo para base `IEDB3` e predição de **EPÍTOPO**



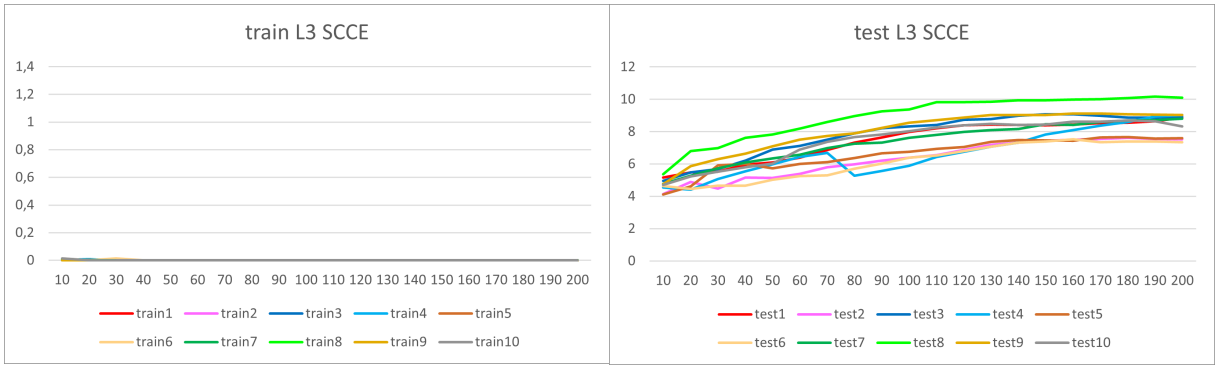
(a) treino (b) teste

Figura A.13: Resultados do modelo para base `IEDB4` e predição de **CDR**)



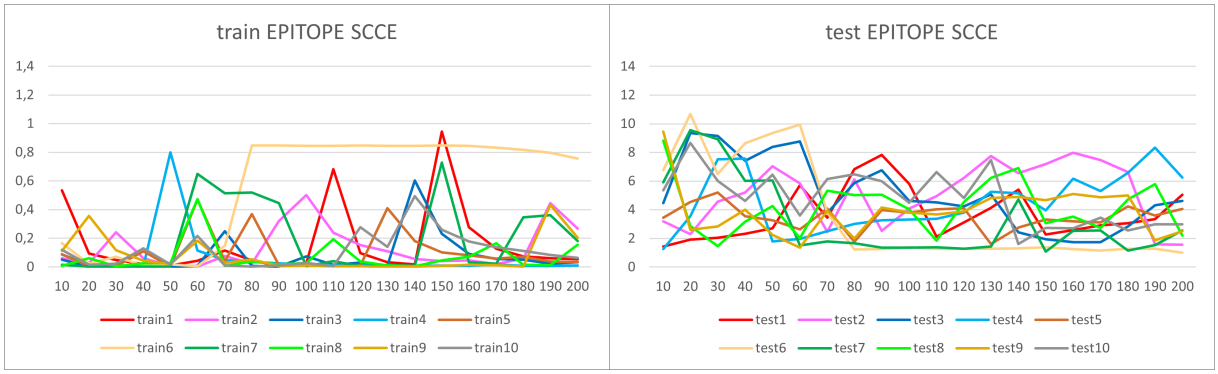
(a) treino (b) teste

Figura A.14: Resultados do modelo para base `IEDB4` e predição de **H3**



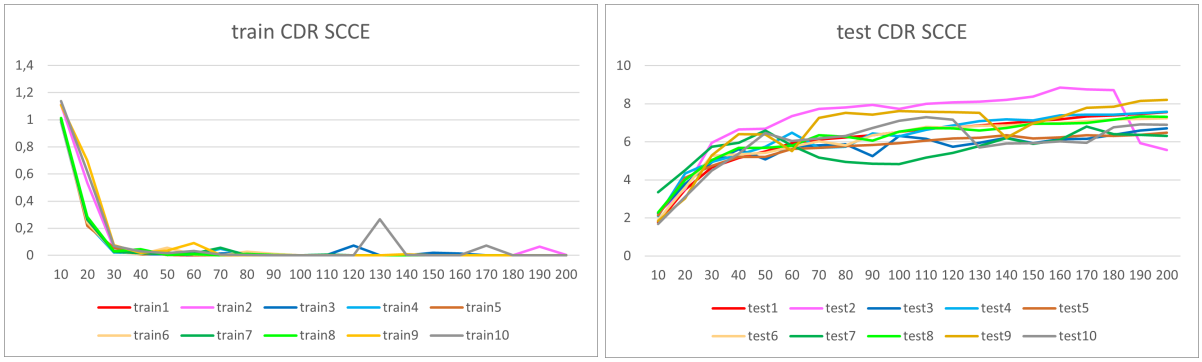
(a) treino (b) teste

Figura A.15: Resultados do modelo para base `IEDB4` e predição de **L3**



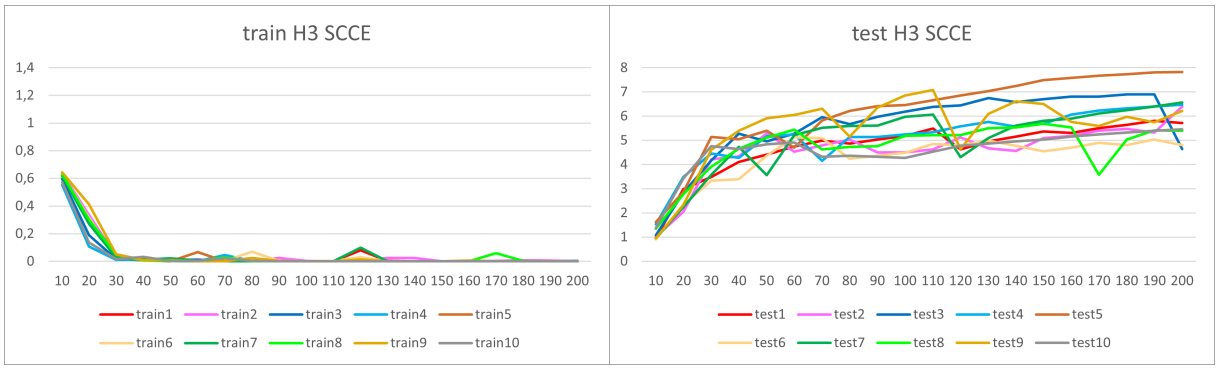
(a) treino (b) teste

Figura A.16: Resultados do modelo para base `IEDB4` e predição de **EPÍTOPO**



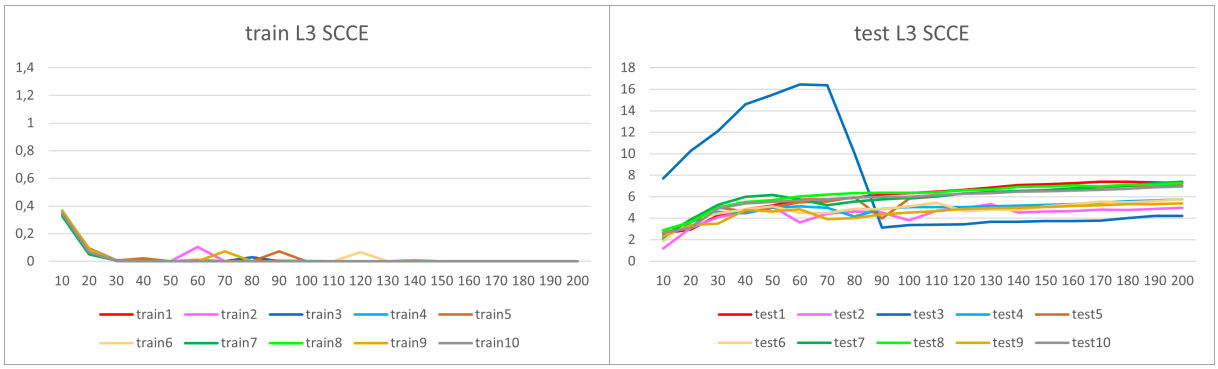
(a) treino (b) teste

Figura A.17: Resultados do modelo para base `IEDB5` e predição de **CDR**



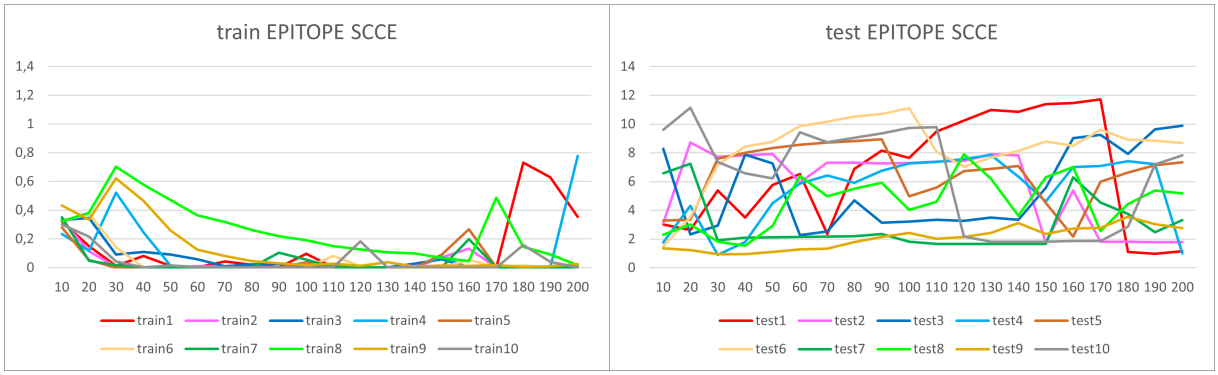
(a) treino (b) teste

Figura A.18: Resultados do modelo para base `IEDB5` e predição de **H3**



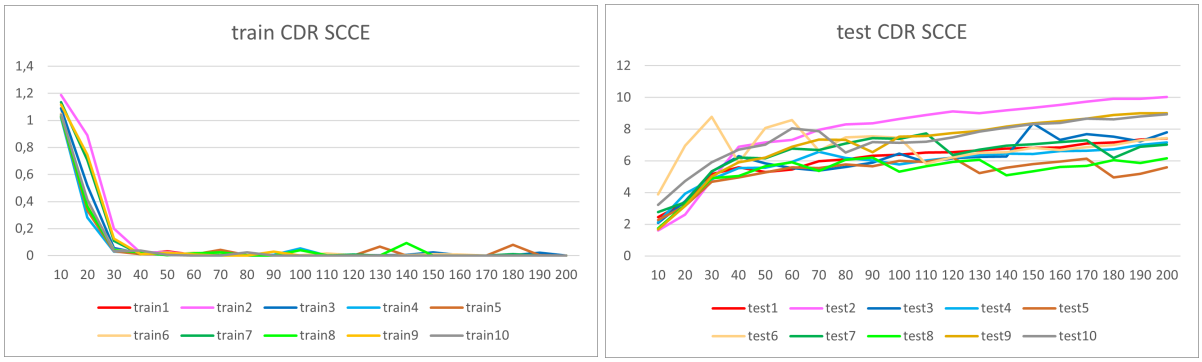
(a) treino (b) teste

Figura A.19: Resultados do modelo para base `IEDB5` e predição de **L3**



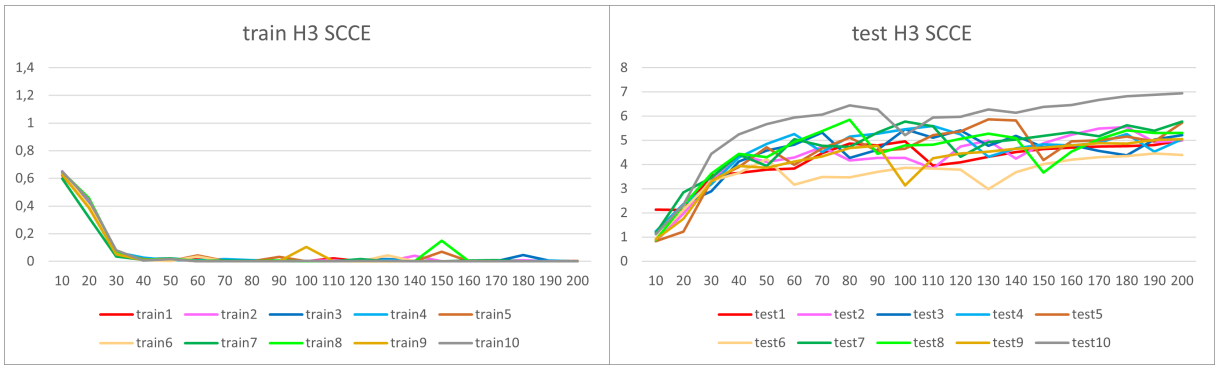
(a) treino (b) teste

Figura A.20: Resultados do modelo para base `IEDB5` e predição de **EPÍTOPO**



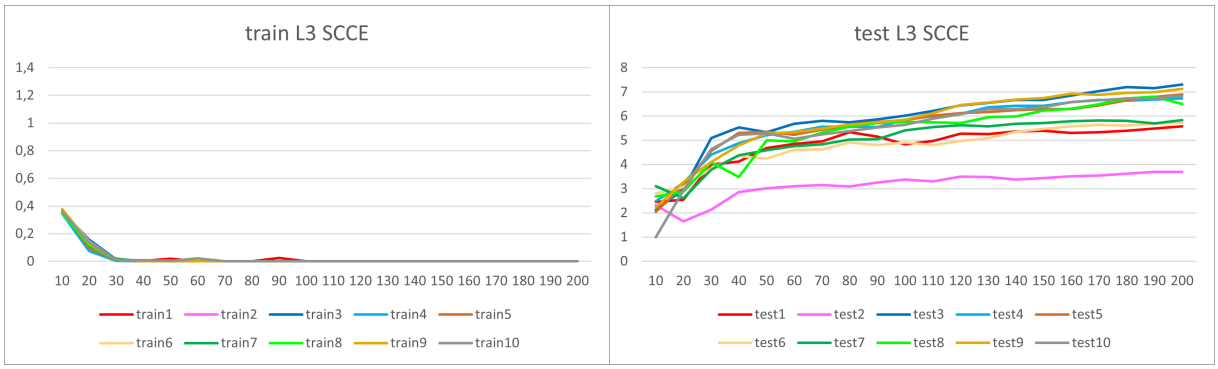
(a) treino (b) teste

Figura A.21: Resultados do modelo para base `IEDB6` e predição de **CDR**



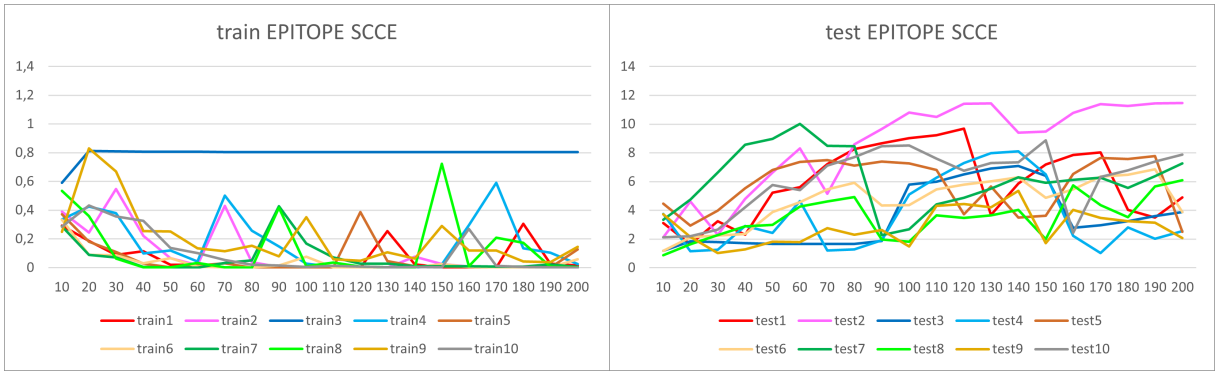
(a) treino (b) teste

Figura A.22: Resultados do modelo para base `IEDB6` e predição de **H3**



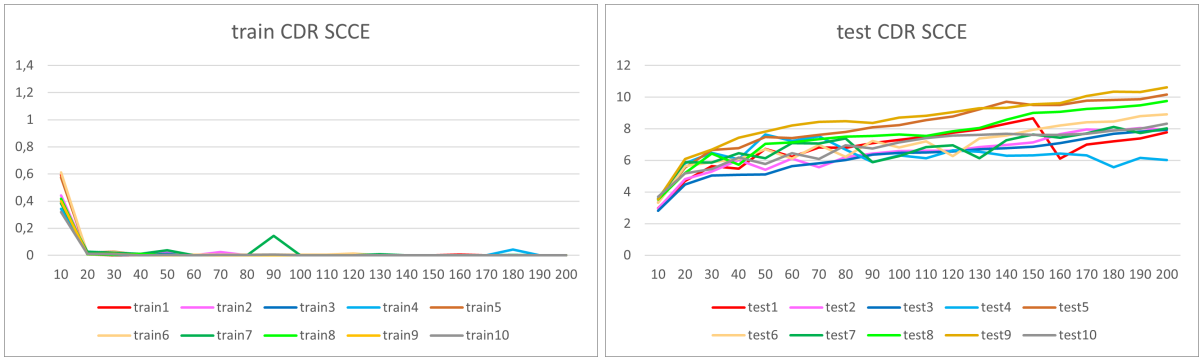
(a) treino (b) teste

Figura A.23: Resultados do modelo para base `IEDB6` e predição de **L3**



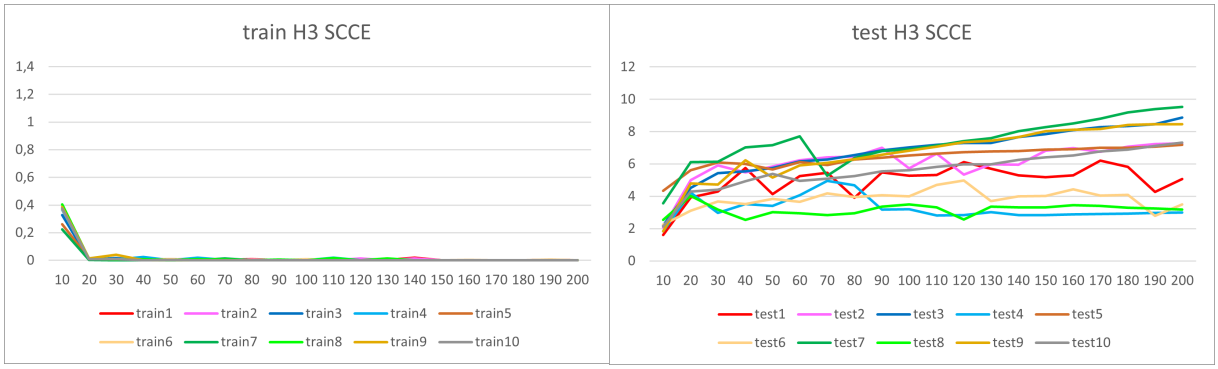
(a) treino (b) teste

Figura A.24: Resultados do modelo para base `IEDB6` e predição de **EPÍTOPO**



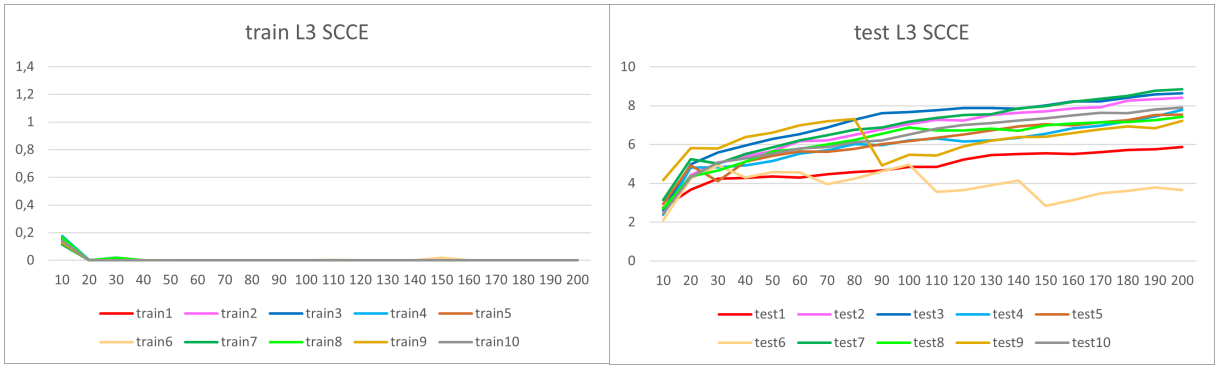
(a) treino (b) teste

Figura A.25: Resultados do modelo para base `IEDB7` e predição de **CDR**



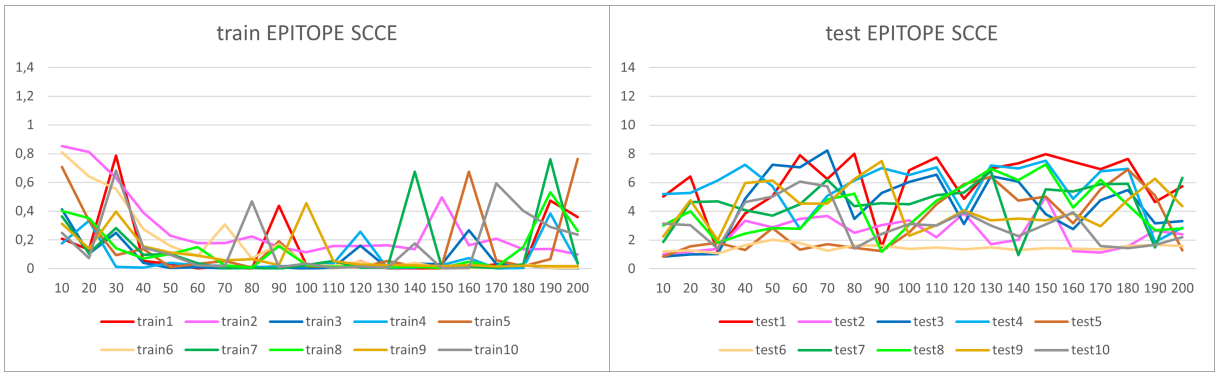
(a) treino (b) teste

Figura A.26: Resultados do modelo para base `IEDB7` e predição de **H3**



(a) treino (b) teste

Figura A.27: Resultados do modelo para base `IEDB7` e predição de **L3**



(a) treino (b) teste

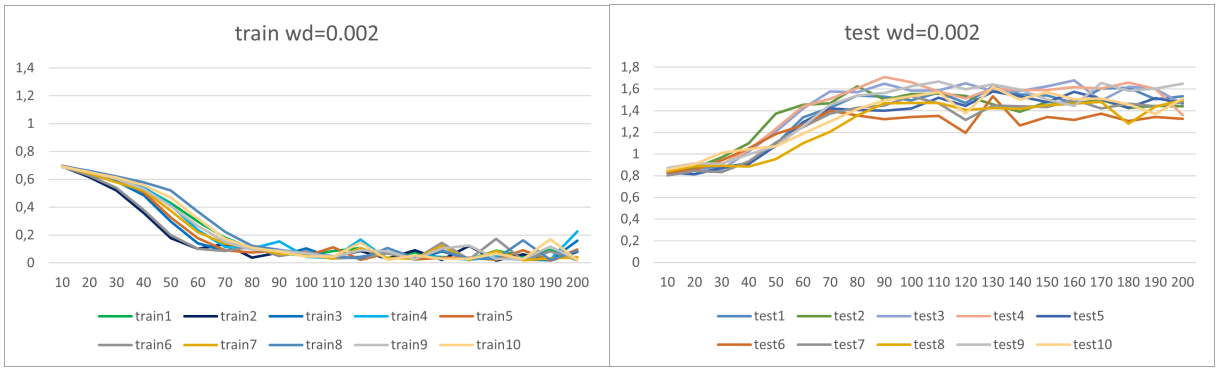
Figura A.28: Resultados do modelo para base `IEDB7` e predição de **EPÍTOPO**

Apêndice B

Resultados dos testes com regularização

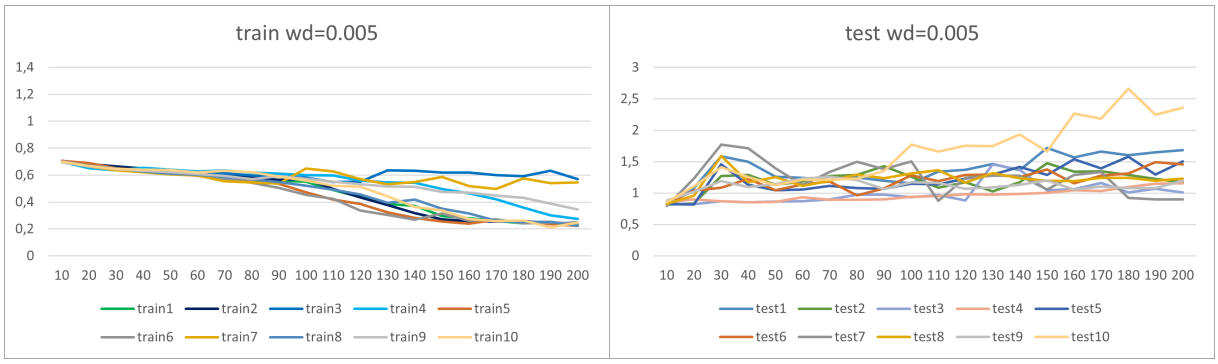
B.1 Regularização com *weight decay*

NOTA: alguns resultados, já apresentados no texto, são reapresentados neste apêndice para facilitar a avaliação comparativa entre os diversos ensaios. Em todos os gráficos, o eixo X refere-se ao número de épocas e o eixo Y ao valor da função de erro (*SparseCategoricalCrossEntropy loss*).



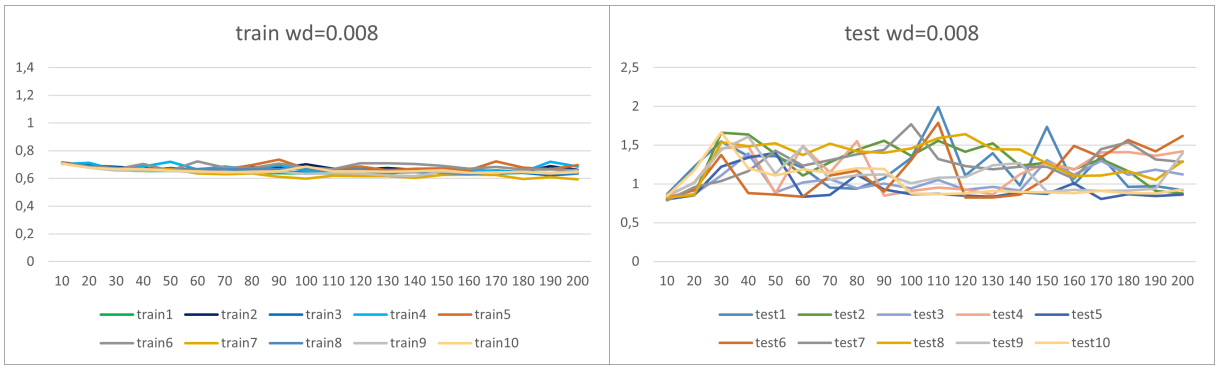
(a) treino (b) teste

Figura B.1: Resultados da regularização do modelo para base `IEDB3` com parâmetro *weight decay* = 0.002 e predição de **H3**



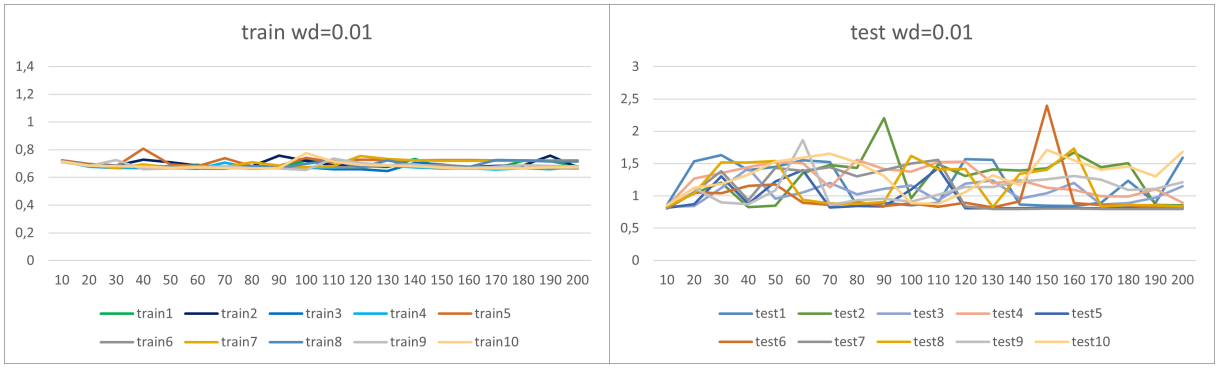
(a) treino (b) teste

Figura B.2: Resultados da regularização do modelo para base `IEDB3` com parâmetro *weight decay* $= 0.005$ e predição de **H3**



(a) treino (b) teste

Figura B.3: Resultados da regularização do modelo para base `IEDB3` com parâmetro *weight decay* $= 0.008$ e predição de **H3**

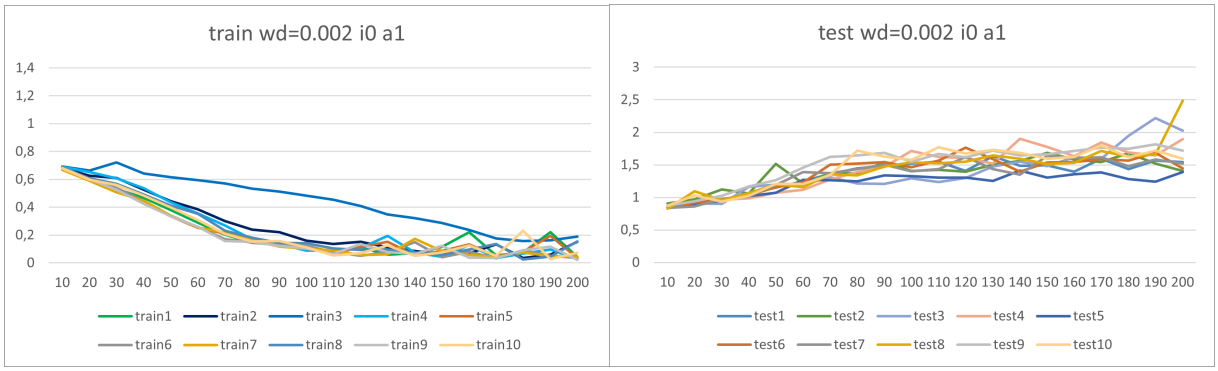


(a) treino (b) teste

Figura B.4: Resultados da regularização do modelo para base `IEDB3` com parâmetro *weight decay* $= 0.01$ e predição de **H3**

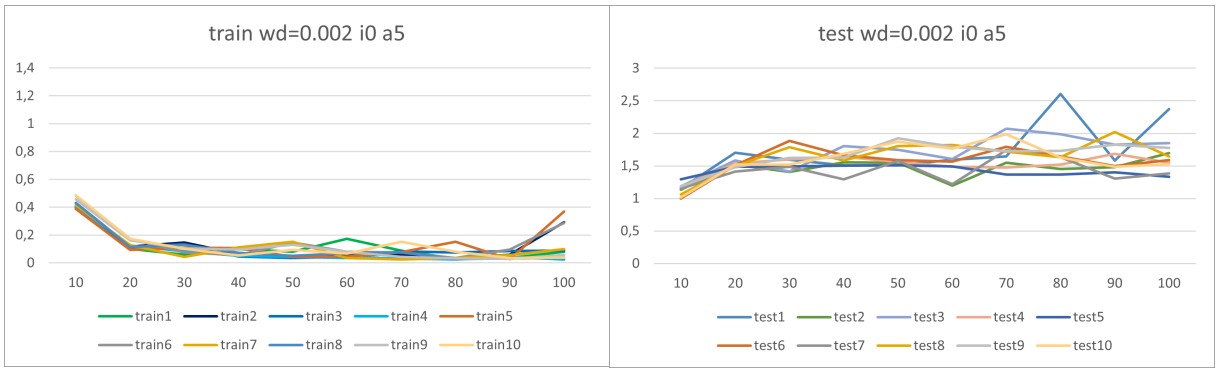
B.2 Regularização com *data augmentation*

NOTA: alguns resultados, já apresentados no texto, são reapresentados neste apêndice para facilitar a avaliação comparativa entre os diversos ensaios. Em todos os gráficos, o eixo X refere-se ao número de épocas e o eixo Y ao valor da função de erro (*SparseCategoricalCrossEntropy loss*).



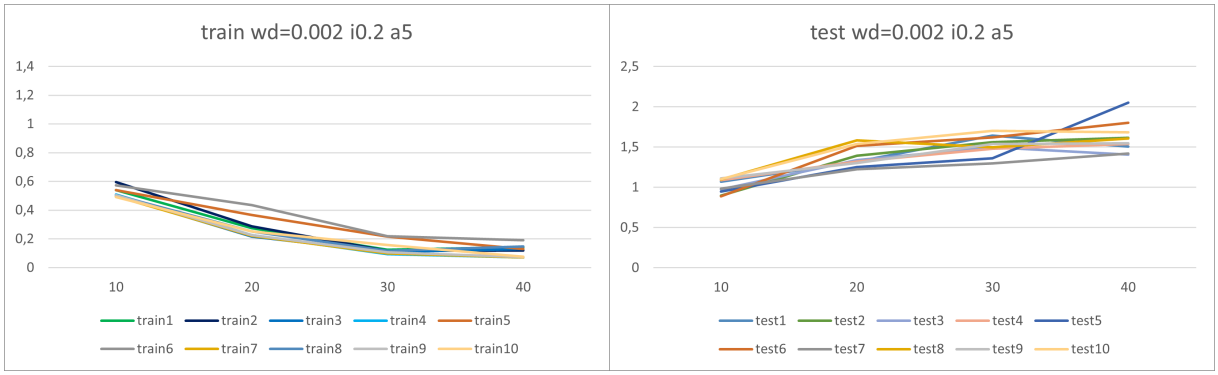
(a) treino (b) teste

Figura B.5: Resultados da regularização do modelo para base `IEDB3` com parâmetro *weight decay* $= 0.002$, *noise* $= 0$, fator de *augmentation* $= 1$ e predição de **H3**



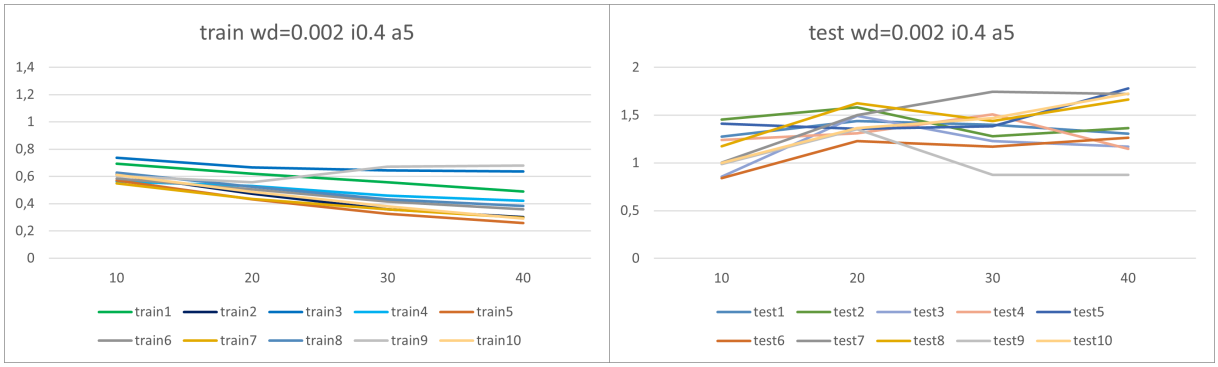
(a) treino (b) teste

Figura B.6: Resultados da regularização do modelo para base `IEDB3` com parâmetro *weight decay* $= 0.002$, *noise* $= 0$, fator de *augmentation* $= 5$ e predição de **H3**



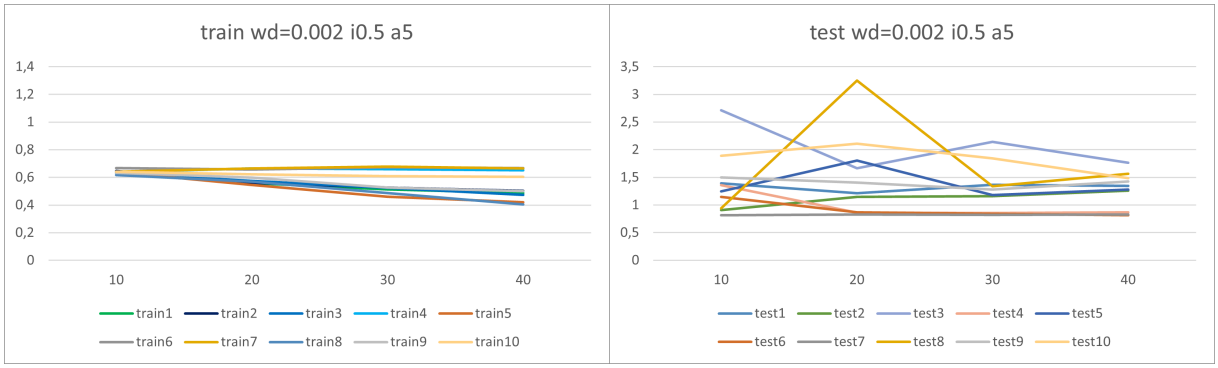
(a) treino (b) teste

Figura B.7: Resultados da regularização do modelo para base `IEDB3` com parâmetro *weight decay* $= 0.002$, *noise* $= 0.2$, fator de *augmentation* $= 5$ e predição de **H3**



(a) treino (b) teste

Figura B.8: Resultados da regularização do modelo para base `IEDB3` com parâmetro *weight decay* $= 0.002$, *noise* $= 0.4$, fator de *augmentation* $= 5$ e predição de **H3**

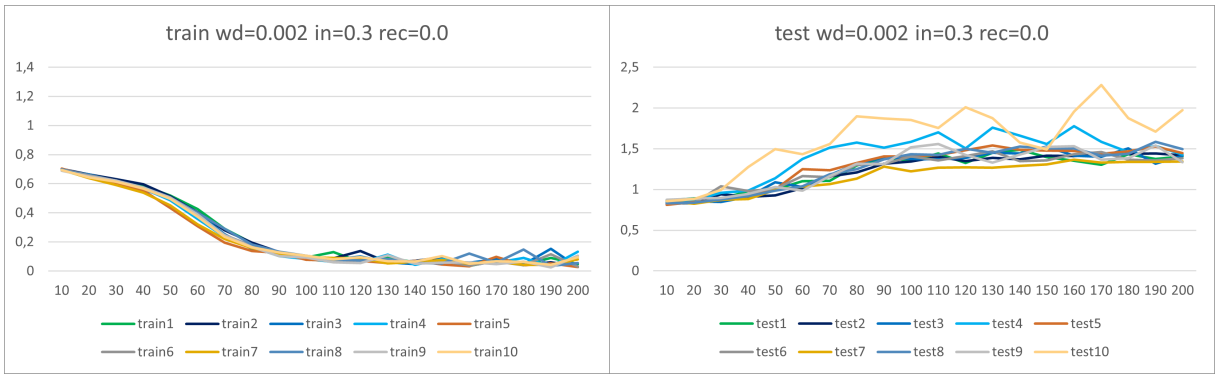


(a) treino (b) teste

Figura B.9: Resultados da regularização do modelo para base `IEDB3` com parâmetro *weight decay* $= 0.002$, *noise* $= 0.5$, fator de *augmentation* $= 5$ e predição de **H3**

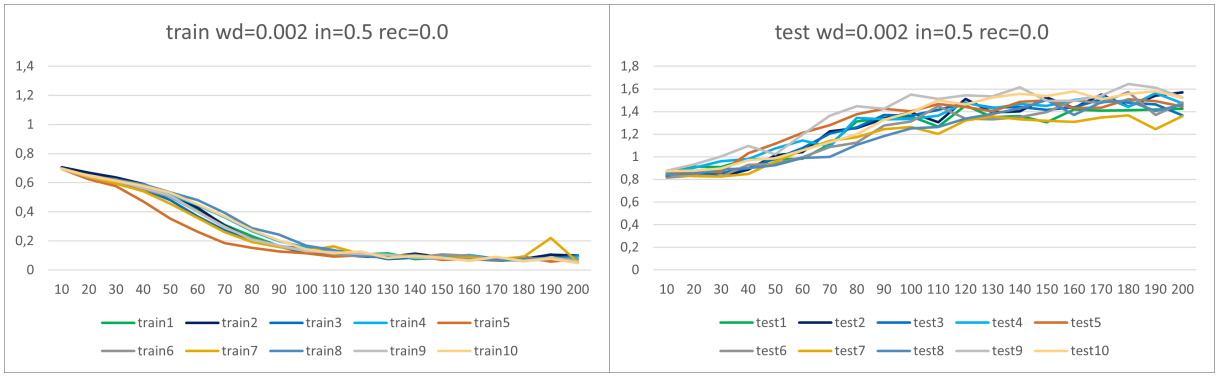
B.3 Regularização com *dropout*

NOTA: alguns resultados, já apresentados no texto, são reapresentados neste apêndice para facilitar a avaliação comparativa entre os diversos ensaios. Em todos os gráficos, o eixo X refere-se ao número de épocas e o eixo Y ao valor da função de erro (*SparseCategoricalCrossEntropy loss*).



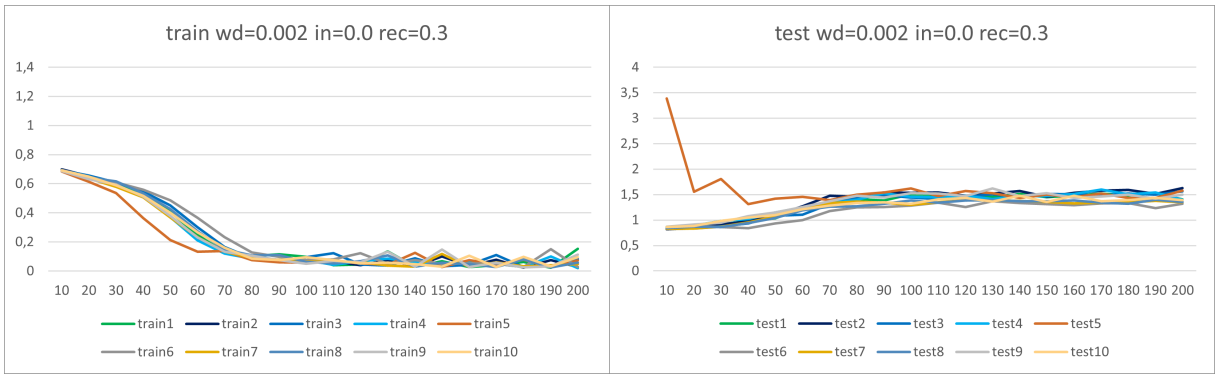
(a) treino (b) teste

Figura B.10: Resultados da regularização do modelo para base `IEDB3` com parâmetro *weight decay* = 0.002, *dropout in* = 0.3, *dropout rec* = 0.0 e predição de **H3**



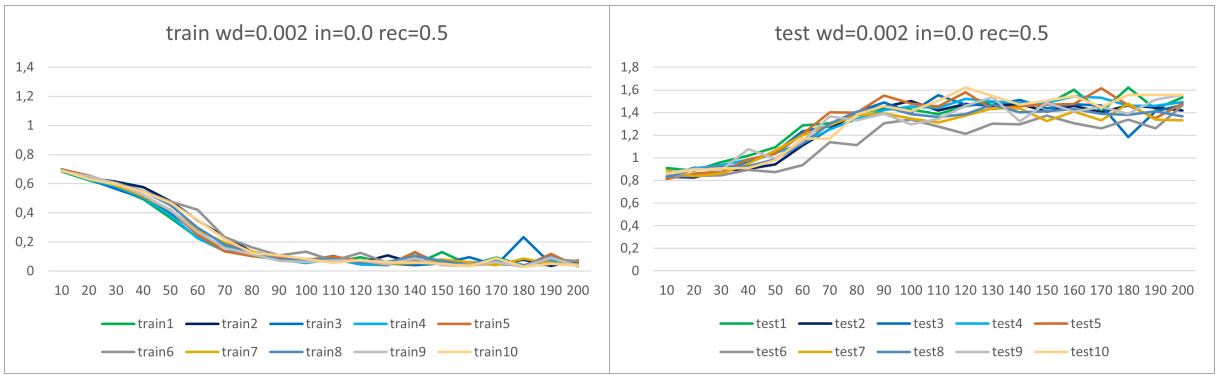
(a) treino (b) teste

Figura B.11: Resultados da regularização do modelo para base `IEDB3` com parâmetro *weight decay* = 0.002, *dropout in* = 0.5, *dropout rec* = 0.0 e predição de **H3**



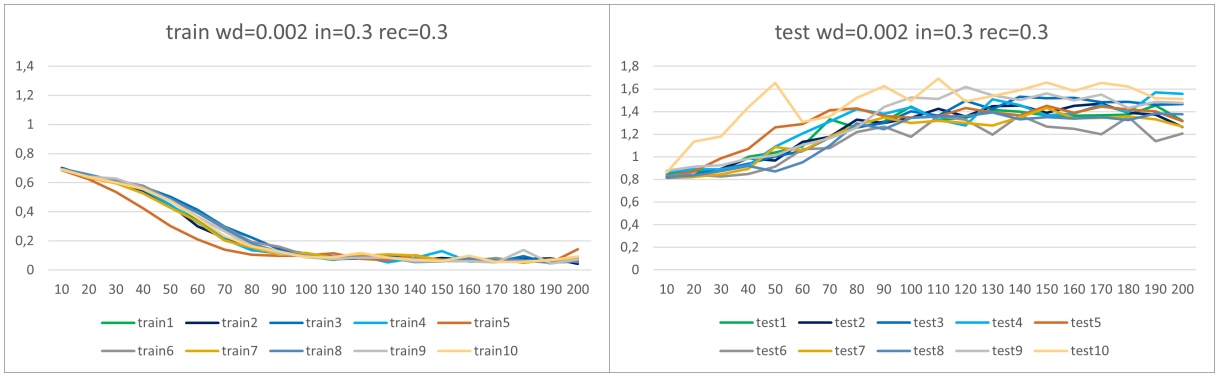
(a) treino (b) teste

Figura B.12: Resultados da regularização do modelo para base `IEDB3` com parâmetro *weight decay* = 0.002, *dropout in* = 0.0, *dropout rec* = 0.3 e predição de **H3**



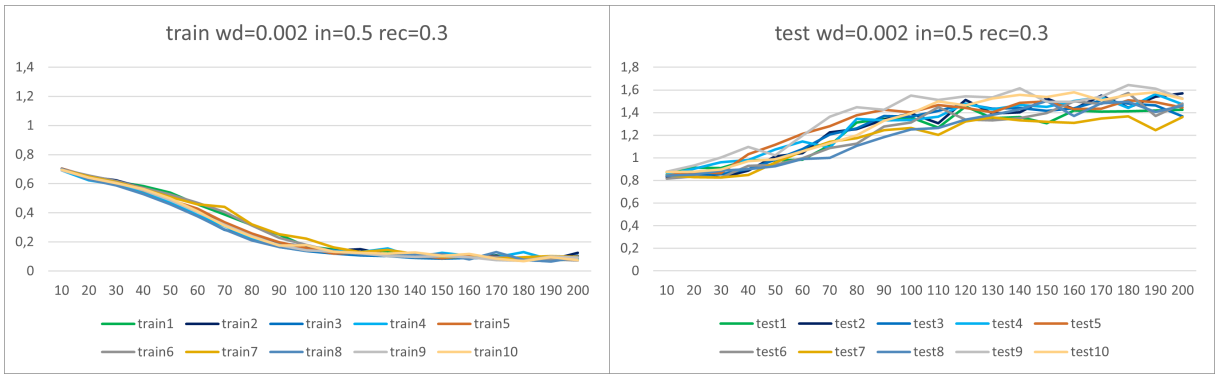
(a) treino (b) teste

Figura B.13: Resultados da regularização do modelo para base `IEDB3` com parâmetro *weight decay* = 0.002, *dropout in* = 0.0, *dropout rec* = 0.5 e predição de **H3**



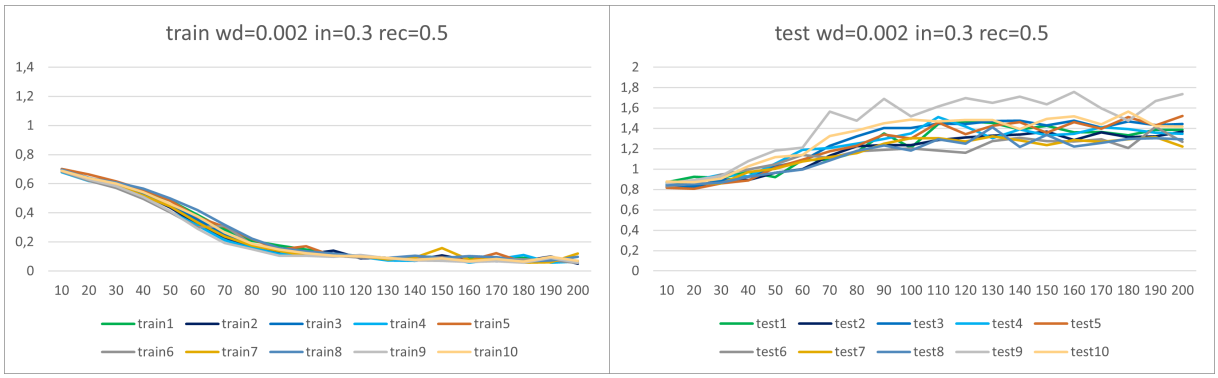
(a) treino (b) teste

Figura B.14: Resultados da regularização do modelo para base `IEDB3` com parâmetro *weight decay* = 0.002, *dropout in* = 0.3, *dropout rec* = 0.3 e predição de **H3**



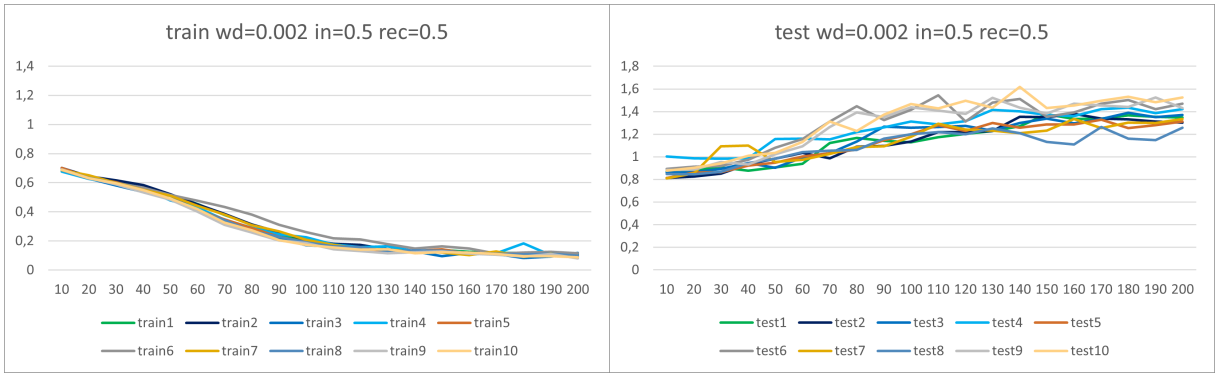
(a) treino (b) teste

Figura B.15: Resultados da regularização do modelo para base `IEDB3` com parâmetro *weight decay* = 0.002, *dropout in* = 0.5, *dropout rec* = 0.3 e predição de **H3**



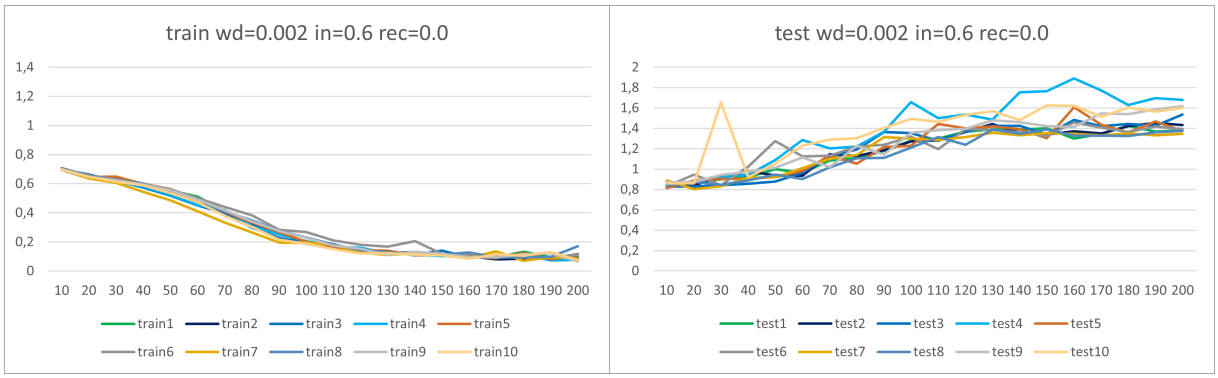
(a) treino (b) teste

Figura B.16: Resultados da regularização do modelo para base `IEDB3` com parâmetro *weight decay* = 0.002, *dropout in* = 0.3, *dropout rec* = 0.5 e predição de **H3**



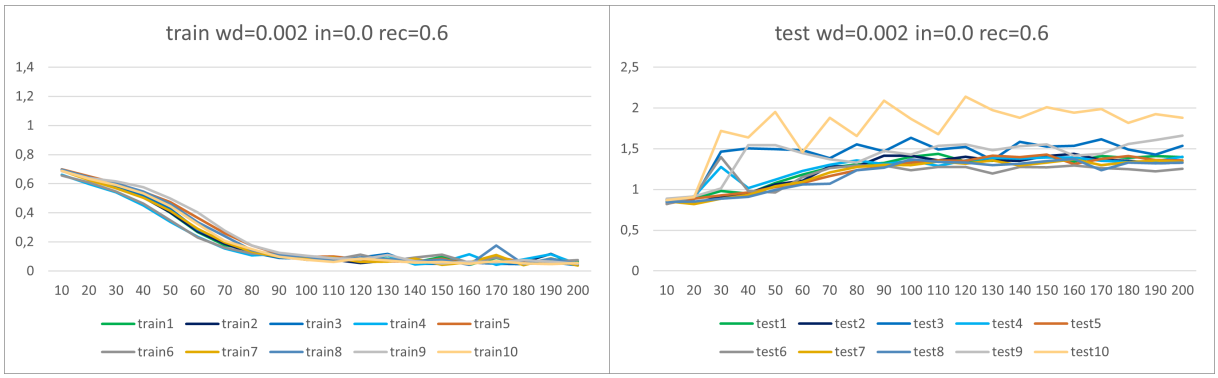
(a) treino (b) teste

Figura B.17: Resultados da regularização do modelo para base `IEDB3` com parâmetro *weight decay* = 0.002, *dropout in* = 0.5, *dropout rec* = 0.5 e predição de **H3**



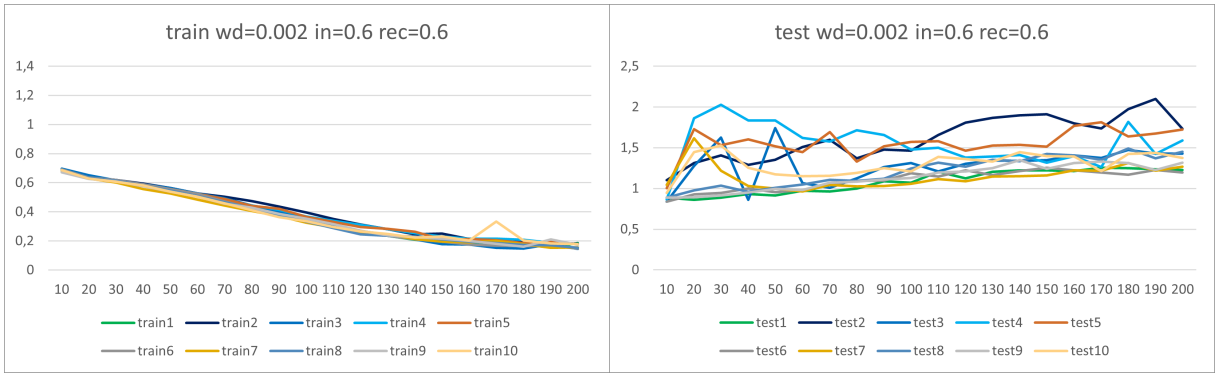
(a) treino (b) teste

Figura B.18: Resultados da regularização do modelo para base `IEDB3` com parâmetro *weight decay* = 0.002, *dropout in* = 0.6, *dropout rec* = 0.0 e predição de **H3**



(a) treino (b) teste

Figura B.19: Resultados da regularização do modelo para base `IEDB3` com parâmetro *weight decay* = 0.002, *dropout in* = 0.0, *dropout rec* = 0.6 e predição de **H3**



(a) treino (b) teste

Figura B.20: Resultados da regularização do modelo para base `IEDB3` com parâmetro *weight decay* = 0.002, *dropout in* = 0.6, *dropout rec* = 0.6 e predição de **H3**

Apêndice C

Infraestrutura de treino e teste

C.1 Hardware

O hardware utilizado possui 32 CPUs INTEL, 3 GPUs NVIDIA e 128G de memória RAM [1].

A infraestrutura INTEL possui as seguintes características:

```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                32
On-line CPU(s) list:   0-31
Thread(s) per core:    2
Core(s) per socket:    8
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 85
Model name:            Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz
Stepping:              4
CPU MHz:               2124.697
CPU max MHz:           2101.0000
CPU min MHz:           800.0000
BogoMIPS:              4200.00
Virtualization:        VT-x
L1d cache:             32K
vL1i cache:             32K
L2 cache:              1024K
L3 cache:              11264K
NUMA node0 CPU(s):     0-7,16-23
NUMA node1 CPU(s):     8-15,24-31
```

[1] Agradecimentos ao Projeto Fapesp 2018/00031–7 pela infraestrutura de processamento

A infraestrutura NVIDIA possui as seguintes características:

```
GPU 0: GeForce RTX 2080 Ti / 11019MiB
GPU 1: GeForce RTX 2080 Ti / 11019MiB
GPU 2: GeForce RTX 2080 Ti / 11019MiB
```

C.2 Software

O modelo foi implementado na linguagem `Python` versão 3.6.9 usando a biblioteca de aprendizado de máquina `Tensorflow` 2.0.0a0, em um ambiente `Docker` (*Client* 20.10.7 e *Server* 19.03.13) e sistema operacional `Linux` Ubuntu versão 18.04.5 LTS (*Bionic Beaver*).