

UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

PREDIÇÃO DE VAZÃO EM CONEXÕES TCP DEDICADAS AO
STREAMING DE VÍDEO DE CURTA DURAÇÃO

MARCOS PEREIRA DA SILVA

PREDIÇÃO DE VAZÃO EM CONEXÕES TCP DEDICADAS AO
STREAMING DE VÍDEO DE CURTA DURAÇÃO

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Instituto de Computação da Universidade Federal do Amazonas, Campus Universitário Senador Arthur Virgílio Filho, como requisito parcial para a obtenção do grau de Mestre em Informática.

Área de concentração: Redes de Computadores e Telecomunicações

Orientador: Prof. Dr. César Augusto Viana Melo

Manaus-AM
Abril/2022

Ficha Catalográfica

Elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

S586p Silva, Marcos Pereira da
 Predição de vazão em conexões tcp dedicadas ao streaming de vídeo de curta duração / Marcos Pereira da Silva. - 2022.
 86 f. : il., color. ; 31 cm.

 Orientador(a): César Augusto Viana Melo.
 Dissertação (mestrado) - Universidade Federal do Amazonas, Programa de Pós-Graduação em Informática, Manaus, 2022.

 1. Predição de vazão. 2. Sessão de vídeo curto. 3. Streaming de vídeo adaptativo. 4. Aprendizagem de máquina. 5. Rede neural recorrente. I. Melo, César Augusto Viana. II. Universidade Federal do Amazonas. Programa de Pós-Graduação em Informática. III. Título



PODER EXECUTIVO
MINISTÉRIO DA EDUCAÇÃO
INSTITUTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA



FOLHA DE APROVAÇÃO

**"PREDIÇÃO DE VAZÃO EM CONEXÕES TCP DEDICADAS AO
STREAMING DE VÍDEO DE CURTA DURAÇÃO."**

MARCOS PEREIRA DA SILVA

Dissertação de Mestrado defendida e aprovada pela banca examinadora constituída pelos Professores:

César Augusto Viana Melo

Prof. César Augusto Viana Melo - PRESIDENTE

Prof. Juan Gabriel Colonna - MEMBRO INTERNO

Moisés Gomes de Carvalho

Prof. Moisés Gomes de Carvalho - MEMBRO EXTERNO

Manaus, 14 de Abril de 2022

*Dedico este trabalho à minha família, minha
fonte de inspiração, força, motivação e
resiliência.*



Agradecimentos

Meus sinceros agradecimentos:

- Ao Criador do universo, pois me guiou e me fortaleceu nos momentos de escuridão. Consagro tudo a Ti;
- Ao meu orientador Cesár Melo pelo apoio incondicional, orientação, por nunca ter desistido de mim e influência para que fosse um estudante profissional. Obrigado, professor.
- à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo auxílio de bolsa, durante a jornada acadêmica;
- aos meus amigos Tonny Osaki, Maria Azevedo e João da Mata, que me ajudaram de alguma maneira na jornada;
- à mãe da minha filha, Laura Lima, que esteve comigo na última etapa da jornada ao cuidar da nossa preciosa filha.

Resumo

Um estudo recente de previsão da Cisco mostrou que a transmissão de vídeo responderá por 82% de todo o tráfego da Internet em 2022. Esse tipo de conteúdo é distribuído em diferentes formas e formatos, em razão do seu consumo não linear e da fragmentação da audiência. Nesse contexto, a oferta de vídeos de curta duração, inferior a um minuto, por exemplo, tem ganhado popularidade, sendo impulsionada por plataformas de *Streaming* de Vídeo como YouTube, Instagram e TikTok. A distribuição desse tipo de conteúdo apresenta desafios particulares que incluem a restrição das taxas de transferências, em conexão fim-a-fim, pelos mecanismos de controle de congestionamento; enquanto em sessões de mais longa duração, existe a possibilidade de se ofertar conteúdo com maior taxa de bits, isto é, com melhor qualidade visual, visto que, existe um certo tipo de estabilização daqueles mecanismos de controle de congestionamento, ao longo da sessão de vídeo. A mesma realidade na transmissão de vídeos de curta duração não é verificável. Os esforços da indústria para a retenção de audiências impõem a necessidade de se entregar sempre conteúdo com a melhor qualidade possível. Para isso, utiliza-se algoritmos e heurísticas com técnicas de adaptação da taxa de bits para a tomada de decisão, a qual possui o pilar, a predição precisa da vazão com a sessão de vídeo em andamento. O uso de dados das sessões em andamento pelos algoritmos de adaptação de taxa, como a única fonte balizadora das decisões sobre a qualidade visual oferecida, mostra-se insuficiente. Neste trabalho, apresenta-se um modelo de aprendizagem de máquina para prever a vazão em conexões dedicadas à transmissão de vídeos de curta duração de dados capturados em históricos de acessos. O modelo, em sua essência, caracteriza-se por uma rede neural recorrente profunda que introduz artefatos que capturam eventos de curta duração, ao aprender padrões representativos de sinais de série temporal. Os resultados mostram que o modelo proposto alcança uma maior precisão em cenários em que as conexões operam com velocidade média acima de 2,88 Mbits/s, quando comparado a modelos tradicionais, de média móvel e modelos similares de rede neural recorrente.

Palavras-chave: Predição de vazão, Sessão de vídeo curto, Aprendizagem de máquina, *Streaming* de vídeo Adaptativo, Rede neural recorrente.

Abstract

A recent forecasting study by Cisco showed that video transmission would account for 82% of all Internet traffic in 2022. This type of content is distributed in different forms and formats due to its non-linear consumption and audience fragmentation. In this context, the offering of short-form videos, shorter than one minute, for example, has gained popularity, driven by video streaming platforms such as YouTube, Instagram, and TikTok. The distribution of this type of content presents particular challenges, which include limitations on transfer rates in end-to-end connections imposed by congestion control mechanisms. In longer-duration sessions, it is possible to offer content with a higher bit rate—that is, better visual quality—since there is a certain stabilization of these congestion control mechanisms over the course of the video session. The same situation is not observed in the transmission of short-duration videos. Industry efforts to retain audiences impose the need to consistently deliver content with the highest possible quality. To this end, algorithms and heuristics employing bitrate adaptation techniques are used for decision-making, which rely on the accurate prediction of throughput during an ongoing video session. The use of data from ongoing sessions by bitrate adaptation algorithms as the sole guiding source for decisions regarding the offered visual quality proves to be insufficient. In this work, a machine learning model is presented to predict throughput in connections dedicated to the transmission of short-duration videos using data collected from access histories. The model is essentially characterized by a deep recurrent neural network that introduces artifacts to capture short-term events by learning representative patterns from time-series signals. The results show that the proposed model achieves higher accuracy in scenarios where connections operate at average speeds above 2.88 Mbit/s, when compared to traditional moving average models and similar recurrent neural network models.

Keywords: Throughput prediction, Short video session, Machine learning, Adaptive video streaming, Recurrent neural network.

Lista de ilustrações

Figura 1 – Rede de Distribuição de Conteúdo. Fonte: Autoria própria.	19
Figura 2 – Arquitetura cliente-servidor do DASH para criação de sessões de vídeo com taxa adaptável. Fonte: Autoria própria.	21
Figura 3 – Modelo matemático de neurônio artificial. Fonte: Autoria própria.	33
Figura 4 – Arquitetura de RNN. Fonte: Autoria própria.	40
Figura 5 – Distribuição geográfica de clientes (Neubot). Fonte: Retirado de [47].	46
Figura 6 – Medições de sessões. Fonte: Autoria própria.	48
Figura 7 – Correlação de Pearson	53
Figura 8 – Informação Mútua	54
Figura 9 – Arquitetura do modelo de rede neural recorrente proposto	58
Figura 10 – Divisão do conjunto de dados.	65
Figura 11 – Desempenho de GSTNet no primeiro ciclo.	69
Figura 12 – Desempenho de GSTNet no segundo ciclo.	71
Figura 13 – Desempenho de GSTNet no terceiro ciclo.	72
Figura 14 – Número de medições e throughput do cenário HFM	74
Figura 15 – Desempenho de GSTNet no cenário HFM com MAE	74
Figura 16 – Desempenho de GSTNet no cenário HFM com RSE	75
Figura 17 – A <i>throughput</i> prevista por GSTNet (vermelha) versus a real (azul) no cenário HFM.	75
Figura 18 – Desempenho de GSTNet no cenário MAM com MAE	76
Figura 19 – Desempenho de GSTNet no cenário MAM com RSE	76
Figura 20 – Predição de GSTNet no cenário MAM	76
Figura 21 – Dinâmica da flutuação da <i>throughput</i> no cenário MIM	77
Figura 22 – Desempenho de GSTNet no cenário MIM com MAE	77
Figura 23 – Desempenho de GSTNet no cenário MIM com RSE	78
Figura 24 – Predição de GSTNet no cenário MIM	78

Sumário

1	Introdução	12
1.1	O Problema	13
1.2	Motivação	15
1.3	Objetivo	16
1.4	Contribuições da dissertação	16
1.5	Estrutura do Trabalho	16
2	Fundamentação teórica	18
2.1	O funcionamento do <i>Streaming</i> de vídeo sobre HTTP	18
2.1.1	<i>Streaming</i> de vídeo Adaptativo Dinamicamente sobre HTTP	20
2.1.2	Modalidades de entrega de <i>Streaming</i> de vídeo	22
2.1.3	Qualidade de experiência em HAS	23
2.1.4	Algoritmos de adaptação de taxa de bits	25
2.1.5	Médias Móveis	26
2.1.5.1	Média Móvel Simples	27
2.1.5.2	Média Móvel Exponencial Ponderada (EWMA)	27
2.1.5.3	Média Harmônica (HM)	28
2.1.6	Predição de vazão em sistemas HAS	28
2.2	Aprendizagem de máquina	30
2.2.1	Random Forest	30
2.2.2	K-Vizinhos Mais Próximos (KNN)	31
2.2.3	Redes Neurais Artificiais	32
2.2.4	O processo de aprendizagem	34
2.2.5	Funções de ativação	36
2.2.6	Técnicas aprendizagem de máquina	37
2.2.7	Função de perda e método de otimização	37
2.2.8	Redes Neurais Recorrentes	39
2.2.8.1	Long Short-Term Memory	40
2.2.8.2	Gated Recurrent Unit	41
2.2.9	Convolutional Neural Network (CNN)	42
2.2.10	Considerações finais	44
3	A base de dados Neubot	45
3.1	O Projeto Neubot	45
3.2	Caracterizando o conjunto de dados	47
3.3	A Engenharia de atributos	50
3.3.1	Seleção de atributos	52
3.3.1.1	Correlação de Pearson	52

3.3.1.2	Informação Mútua	53
3.4	Considerações finais	54
4	O modelo GSTNet	56
4.1	O problema de predição	56
4.2	A arquitetura do modelo	57
4.2.1	A normalização das features	57
4.2.2	As camadas de GSNet	58
4.2.2.1	A camada CNN 1D	58
4.2.2.2	A camada BiGRU	60
4.2.2.3	A camada Flatten	60
4.2.2.4	A camada Densa	61
4.2.3	Função de perda	61
4.3	Métricas de erro	62
4.4	Considerações finais do capítulo	63
5	Resultados numéricos	64
5.1	Protocolo experimental	64
5.1.1	Ajuste de parâmetros	66
5.2	Resultados e discussão	67
5.2.1	Treinamento	67
5.2.1.1	O primeiro ciclo	67
5.2.1.2	O segundo ciclo	69
5.2.1.3	O terceiro ciclo	70
5.2.2	Teste: predição de GSTNet	72
5.2.2.1	Higher Flow Measurements	73
5.2.2.2	Majority Measurements	75
5.2.2.3	Minority Measurements	76
5.3	Considerações finais	78
6	Conclusões e considerações finais	79
	Referências	82

CAPÍTULO 1

Introdução

A *Throughput* e predição em canais fim-a-fim são temas bastante investigados na literatura de conectividade de redes de computadores. Um dos resultados mais referenciados dessa literatura é conhecido como: a guerra dos ratos e dos elefantes [19]. De acordo com esse estudo, aponta-se que poucas conexões transferem grandes quantidades de dados e perduram por longos períodos, isto é, a maior parte do tráfego é resultante de uma pequena fatia de todos os fluxos - caracterizando um fluxo elefante, segundo limiares predefinidos [19], e.g., grandes transações de transferência de dados e compartilhamento de arquivos ponto a ponto; enquanto que um número bem maior de conexões transferem poucos dados, e.g., o conteúdo de um formulário, perdurando por curtos períodos, e.g., leilões virtuais, transações bancárias e compras on-line. Todas essas conexões caracterizam um fluxo rato.

Desde a sua exposição, esforços têm sido empreendidos para minimizar os impactos dessa disputa por recursos de transmissão, à proporção que, fluxos elefantes podem impactar, em tráfego de fluxos menores, aqueles cujos limiares são significativamente menores, além de interferir na qualidade de experiência dos usuários das aplicações que geram tais.

Nesse espaço de disputa por recursos dentro do tráfego de rede, a transmissão de vídeos de curta duração, pela sua natureza em si, possui restrição temporal acentuada, exigindo uma fatia de demanda significativa de recursos da rede. Por conseguinte, as conexões criadas para a transmissão de vídeos de curta duração, ou seja, com menos de 60 segundos, não se encaixam perfeitamente naquele tipo de caracterização. Essa constatação propicia um ensejo para investigação de como os mecanismos de controle de conexão têm reagido a esse perfil de tráfego de rede emergente.

TikTok, Instagram e YouTube são exemplos de plataformas de alcance global que disponibilizam conteúdo no formato de vídeos de curta duração, com um alto grau de destaque para a primeira, nos últimos anos. As estratégias de distribuição dessas plataformas estão ancoradas em duas tecnologias: i) cacheamento/pre-carga, i.e., aproximação do conteúdo de suas audiências;

ii) fluxo com taxa adaptável, i.e., adequação da qualidade visual das imagens aos recursos de transmissão disponíveis. A otimização dessas estratégias demandam o conhecimento dos recursos que estarão disponíveis nas infraestruturas de acesso na última milha, permitindo a tomada de decisões mais otimizadas, a qual está relacionada diretamente à disponibilização de conteúdo à *Throughput* dos canais de acesso.

Aquelas plataformas de alcance global são notórias produtoras de sessões de vídeo com alta aderência. Uma característica presente em suas aplicações é o reconhecimento de que os recursos de transmissão são finitos e, por natureza, flutuam ao longo das sessões. Para construir, então, essas sessões, as aplicações ajustam a qualidade visual da sessão aos recursos disponíveis para transmissão. Esse ajuste está diretamente relacionado ao uso das tecnologias de taxa de bits adaptável, em que a predição de *Throughput* das conexões é um fator de rede indispensável e, ao mesmo tempo crítico, para a tomada de decisão.

A literatura produzida sobre a otimização da qualidade visual das sessões de vídeo, no contexto de *Streaming de Vídeo*, que se confunde com fluxo contínuo, tem apresentado algoritmos, que em sua maioria, utilizam dados da sessão em andamento para realizar a predição da *Throughput* [58] [41]. Em resumo, o esforço de otimização visa definir qual deve ser a qualidade visual dos próximos segmentos a serem acessados. Essa decisão tem como restrição: i) o conjunto de qualidades visuais, i.e., taxas de codificação, entregues pelos provedores de conteúdo nos servidores mais próximos da audiência; ii) o volume de vídeo já armazenado localmente, e iii) a *Throughput* percebida no histórico de acessos já realizados durante as sessões. A primeira e terceira restrição possuem forte correlação com a capacidade de transmissão que estará disponível para o *player* interagir com o servidor que armazena o vídeo.

Nesse contexto, o presente trabalho desenvolveu e validou um modelo de rede neural recorrente com recursos de aprendizagem profunda para a tarefa de predição de *Throughput* de canais fim a fim característicos do TCP. Como fonte de dados, utilizou-se centenas de dados históricos que foram observados de informações de logs de sessões de vídeo de curta duração, precisamente, vídeos de 30 segundos com taxa de bits adaptável variada. Os logs foram capturados por meio da coleta de medições de acesso na borda da rede de distribuição com a intervenção de mecanismos de bots¹, implantados em máquinas de usuários voluntários de várias regiões do Brasil.

1.1 O Problema

O Fluxo Contínuo com Taxa de Bit Adaptada Dinamicamente sobre HTTP², do inglês *Dynamic Adaptive bitrate Streaming over HTTP - (DASH)* [54], padroniza as diversas tecnologias viabilizadoras para transmissão de vídeos com qualidade visual adaptável. Os pilares dessa tecnologia são i) o versionamento do vídeo em várias taxas de bits, segmentadas e codificadas

¹ <https://www.measurementlab.net/tests/neubot/>

² *Hypertext Transfer Protocol*

com poucos segundos de tempo de reprodução; e ii) o acesso desse conteúdo a partir de uma aplicação cliente, i.e., *player* de vídeo, que reconhece as várias versões e que adapta a qualidade visual.

Visando maximizar a qualidade de experiência das audiências, algoritmos para adaptação da taxa de bits, do inglês, *Adaptive BitRate* - ABR, são utilizados para controlar as sessões de vídeo a partir do *player*. Esses algoritmos adaptam a qualidade visual das sessões, dinamicamente, às mudanças das condições de transmissão [34]. As tomadas de decisão têm como base a *Throughput* da conexão e/ou a ocupação de um *buffer* de reprodução, que armazena segmentos do vídeo já acessados, mas não reproduzidos. Se a *Throughput* for maior que a taxa de bits dos últimos segmentos, o algoritmo ABR pode selecionar segmentos de vídeo com uma taxa de bits mais alta, com o intuito de melhorar a qualidade visual e impactar na experiência da audiência. Por outro lado, um *Throughput* menor que as taxas de bits dos segmentos já acessados pode induzir o algoritmo ABR a acessar segmentos codificados com taxas de bits menores, i.e., qualidade visual menor, evitando o esvaziamento do *buffer* e, por conseguinte, interrupção do fluxo de dados [34].

A respeito das técnicas populares mencionadas anteriormente, ainda persiste-se o desafio da escolha da qualidade mais adequada para um certo momento da sessão. Esse desafio é resultante de uma combinação de fatores, mais especificamente i) a oscilação inata da *Throughput* do canal fim a fim; ii) a granularidade das taxas de bits usadas para definir o conjunto de versões do vídeo e iii) a imprecisão da medição de *Throughput*. Os dois primeiros fatores escapam ao escopo deste trabalho, mas o terceiro fator está diretamente ligado à proposta realizada: um mecanismo para estimar, de modo contínuo, a taxa de *Throughput*, dado que, a predição da *Throughput*, realizada a partir da dinâmica de requisição e resposta, resulta em medidas imprecisas.

Uma sessão de *Streaming* de vídeo é caracterizada pela existência de dois estados, ambos denominados de preenchimento de *buffer* (*buffer-filling*) e estado de equilíbrio dinâmico (*steady buffer*). O objetivo do estado *buffer-filling* consiste em preencher o nível do *buffer* com segmentos até um certo limite, desde que *downloads* de segmentos anteriores tenham sido finalizados e, dessa forma, a reprodução do vídeo pode ser iniciada ou retomada. Já o estado *steady buffer*, tenta manter um tamanho de *buffer* constante, acima de um limite mínimo, mesmo com a presença de flutuações de largura de banda com a finalidade de evitar eventos de interrupção e esvaziamento [8].

O *steady buffer* é caracterizado por dois períodos de atividade, denominados de *ON* e *OFF*, em que o *player* de vídeo está no estado *ON* quando acessa o segmento atual e registra a taxa de *Throughput* alcançada, que será utilizada posteriormente para a seleção de segmentos futuros, ou no estado *OFF*, quando fica ocioso. No período de ociosidade, o *player* mantém informações desatualizadas sobre a *Throughput* atual do canal [4], ocasionando em períodos de silêncio.

O período de silêncio é um fator que interfere no desempenho da estimação de *Throughput*

inerente à sessão de vídeo com taxa adaptável, em razão de ações relacionadas ao engajamento do usuário (tempo total de permanência do usuário no sistema) e da dinâmica da estratégia de adaptação. Em relação à estratégia de adaptação, a ocorrência de sobrecarga do *buffer*, a demanda de tempo para a formulação de novas requisições e a seleção de outro servidor são ações que também estimulam em períodos de silêncio no canal, gerando ruído no processo de decisão da qualidade visual do próximo segmento de vídeo a ser visualizado.

As questões levantadas acima revelam a importância da concepção de métodos precisos de predição de *Throughput* para a seleção adequada de taxa de bits durante a sessão, uma vez que esquemas tradicionais de adaptação de taxa [41, 58] fazem medições imprecisas nos períodos de silêncio [9], ocasionando degradação na qualidade de vídeo percebida pelo usuário final.

1.2 Motivação

Um dos principais desafios na transmissão de vídeos pela Internet é lidar com a flutuação da *Throughput*. Essa característica inata da Internet torna improvável a transmissão de vídeos, com uma taxa de bits constante, em sessões do tipo click e reproduz [8]. Por outro lado, a escala da Internet permite que grandes audiências se formem em torno de provedores de vídeos. Plataformas como TikTok e Kwai servem bilhões de acessos mensalmente, conforme estudo recente³. O conteúdo dessas plataformas é essencialmente formado por vídeos de curta duração.

O sucesso dessas plataformas está diretamente relacionado com a capacidade de transformar acessos eventuais em aderência permanente. Duas tecnologias chaves para aumentar essa aderência são: a aproximação antecipada dos conteúdos de suas audiências e a habilidade de adequar a qualidade visual das sessões aos recursos disponíveis para a transmissão. Em ambas as abordagens, a capacidade de transmissão efetiva das infraestruturas de acesso é uma informação chave para entregar o conteúdo com a melhor qualidade visual, sob a restrição do modelo click e reproduz. Ao mesmo tempo, a operação de aproximação do conteúdo da audiência pode ter seus custos reduzidos, ao evitar que o conteúdo com certa qualidade visual e baixa possibilidade de uso sejam incluídas na lista de candidatas à disponibilidade de uma determinada audiência.

As conexões utilizadas para transferir vídeos de curta duração apresentam características únicas, pois combinam restrições de atraso, i.e., um vídeo curto precisa ter a sua reprodução iniciada rapidamente, com a demanda por transmissão de um significativo volume de dados, i.e., 30 segundos de vídeo 4K demanda 175 MBytes para a sua codificação. A reação dos mecanismos de controle de tráfego, em particular do TCP, a essas características não foram ainda abordadas na literatura. Ao mesmo tempo, métodos para a predição de *Throughput*, considerando a transmissão daquele tipo de vídeo, em uma determinada infraestrutura de acesso à Internet - ISP, não estão disponíveis. A prevalência é por modelos um-para-todos, em que as conexões são tratadas indistintamente.

³ <https://www.conviva.com/pt/redacao/new-report-looks-at-tiktok-brand-performance-benchmarks-and-tactics/>

1.3 Objetivo

Este trabalho teve como objetivo a construção de um modelo para predição de *Throughput* de conexões na última milha, a partir de históricos de acesso de aplicações de *Streaming* de vídeo, atendidas por provedores de acesso a Internet situados em diversas regiões do Brasil. Para alcançar esse objetivo, definiu-se os seguintes objetivos específicos:

1. Analisar e avaliar o conjunto de dados em busca de características e especificações que estejam implicitamente relacionadas com estabilidade e *Throughput* dos canais de rede;
2. Desenvolver um modelo de rede neural recorrente profunda que se ajuste à distribuição temporal e espacial da sequência dos dados para a realização de predição de *Throughput* com maior precisão.
3. Avaliar modelos de predição de *Throughput* adequados para modelar sequências de dados com características de série temporal;

1.4 Contribuições da dissertação

As principais contribuições deste trabalho foram estas:

1. Estudo do impacto de técnicas de aprendizagem de máquina na modelagem do comportamento de *Throughput* de canais de aplicações de *Streaming* de Vídeo sob demanda;
2. Criação de uma arquitetura de rede recorrente neural profunda para a predição de *Throughput* de acordo com o histórico de acesso das aplicações;
3. Estudo comparativo entre redes neurais recorrentes profundas com algoritmos previsores de regressão na tarefa de predição de *Throughput*;
4. Identificação de diferentes perfis de canais fim a fim para os quais a predição de taxa de *Throughput* se realiza com maior precisão.

1.5 Estrutura do Trabalho

O capítulo 2 apresenta o contexto e desafios na área de distribuição de *Streaming* de vídeo sob demanda na Internet com ênfase para a transmissão de taxa de bits adaptável, baseada em métricas de ajustes de *Throughput* (vazão) da rede. Conceitos relacionados à aprendizagem de máquina como redes neurais recorrentes, processo de aprendizagem, funções de ativação e perda, dentre outros, são apresentados também no capítulo. O capítulo 3 apresenta o conjunto de dados utilizado para a realização deste estudo. Apresenta-se também um estudo de engenharia e seleção de atributos aplicado no conjunto de dados. No capítulo 4 apresenta-se a descrição

do modelo de rede neural recorrente e seus componentes. O capítulo 5, por fim, apresenta os resultados do processo de aprendizagem e inferência do modelo. Mostra-se sua complexidade matemática e seu desempenho no conjuntos de validação e teste, além de uma comparação justa diante de modelos rasos de regressão e modelos com arquiteturas profundas de rede neural recorrente.

CAPÍTULO 2

Fundamentação teórica

Este capítulo apresenta a fundamentação teórica para a compreensão deste trabalho. As principais tecnologias utilizadas na transmissão de vídeo desde a origem até o usuário final são apresentadas e explicadas, tais como a arquitetura cliente-servidor, serviços de entrega, protocolos de rede que são executados durante a entrega de vídeo e anomalias que interferem no fluxo contínuo de vídeo, em geral. Fatores de qualidade de serviço que interferem na qualidade de experiência percebida de vídeo são abordados, bem como perda de pacotes, atraso de entrega, jitter e eventos de paradas.

O capítulo também apresenta um exemplo de estratégia de adaptação de taxa utilizada por aplicações de *Streaming* de vídeo, a qual é executada no *player* de vídeo da audiência e a forma convencional de estimação de vazão. O capítulo finaliza apresentando a teoria do campo de aprendizagem de máquina e suas técnicas, com ênfase para redes neurais recorrentes profundas, que são investigadas e tratadas neste trabalho para a finalidade de estimação de vazão por meio de informações históricas de acesso provenientes de sessões de *Streaming* de vídeo.

2.1 O funcionamento do *Streaming* de vídeo sobre HTTP

O *Streaming* ou fluxo contínuo de vídeo é uma tecnologia utilizada em aplicações de entrega de conteúdo de vídeo na Internet que envolve tecnologias de preparação e distribuição de vídeos e aspectos de estímulo à aderência do usuário final [38]. O *Streaming* de vídeo possibilita que o conteúdo de vídeo seja reproduzido no dispositivo do usuário em pequenas partes, chamadas de segmento, sem a necessidade de carregá-lo completamente. O arquivo de vídeo é armazenado, remotamente, em diversos servidores, localizados próximos das audiências e transmitido pela Internet usando um modelo de transferência de melhor esforço. Os segmentos de vídeo são reproduzidos no player de vídeo do usuário em consonância à alguma lógica de adaptação que monitora os recursos locais e do canal de rede disponíveis. Essa forma de entrega e

reprodução de vídeo na Internet é utilizada por grandes empresas de entretenimento como Netflix, You Tube, Vimeo e Amazon Prime Vídeo, que possuem audiências de milhões de consumidores.

Um aspecto inerente a todas as tecnologias de *Streaming* de vídeo é a divisão do vídeo original em pedaços ou segmentos (chunks) para serem entregues de maneira contínua e adaptável ao usuário final. O conteúdo de vídeo é armazenado em diferentes sistemas de distribuição de conteúdo, como por exemplo, em *Content Delivery Network* (CDN). As CDNs [49] representam grupos de servidores distribuídos, geograficamente, com centenas de nós de borda para facilitar a entrega de conteúdo multimídia de maneira escalável e rápida. Essas redes de distribuição foram desenvolvidas para superar problemas de sobrecarga de servidor e congestionamento de rede, em consequência da distribuição em larga escala de dados de vídeo e crescimento de acesso de conteúdo popular.

Uma vez que o conteúdo não é distribuído diretamente do servidor de origem, este é enviado por meio de um canal de rede mais curto, tornando possível reduzir: i) a probabilidade de perda de pacotes; ii) o tempo de resposta e iii) o uso dos recursos de rede [49]. As CDNs replicam o conteúdo de vídeo para os servidores de borda; colocando-o mais próximo da sua audiência com o objetivo de melhorar a experiência de visualização [36]. Em termos de serviço, esses conjunto de servidores distribuídos são utilizados para melhorar a percepção da audiência quanto à interatividade da aplicação, uma vez que suas requisições são redirecionadas para o melhor servidor disponível baseado na proximidade geográfica. A Figura 1 mostra como uma CDN é estruturada, e como os serviços de *Streaming* aproveitam-se dessa estrutura para melhorar a percepção dos usuários de serviços de vídeo.

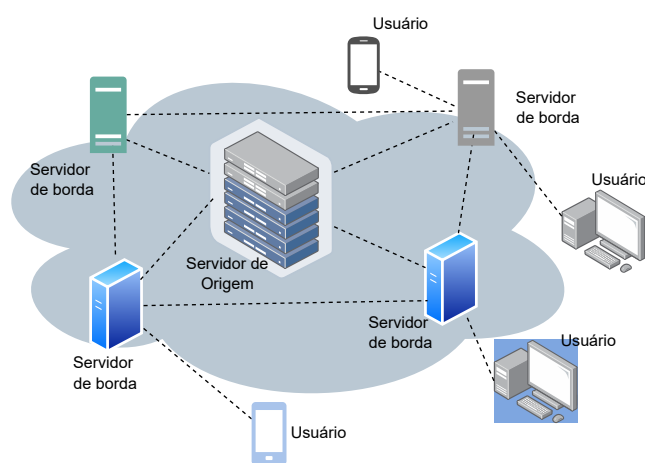


Figura 1: Rede de Distribuição de Conteúdo. Fonte: Autoria própria.

Em cada servidor de borda da CDN é replicado parte do conteúdo original, e essas réplicas são usadas para atender as requisições direcionadas inicialmente ao servidor de origem. Escolher o conteúdo que será replicado em cada servidor de borda é um problema reconhecidamente difícil. A abordagem usada é identificar padrões de acesso e replicar o conteúdo com base nesses padrões. Conteúdos relacionados são dispostos nos mesmos servidores. Além disso, arquivos que viabilizam as funcionalidades de dublagem e legendas também precisam ser replicados nesses

servidores. Novamente, os padrões de acesso são usados para determinar quais arquivos serão replicados e quais servidores de borda irão receber as réplicas.

A primeira tecnologia de *Streaming* de vídeo que atingiu a escala da Internet ficou conhecida como Download Progressivo. Com o Download Progressivo, as sessões de vídeo têm qualidade constante definida no início da sessão. Então, o player de vídeo (cliente), inicia a reprodução antes que o download de todo o vídeo seja concluído. No entanto, a mesma qualidade de vídeo pode ser transmitida para diferentes usuários que dispõe de recursos diversos, em termos de dispositivos e conexões de rede, causando com isso, interrupções na reprodução do vídeo [35].

Para permitir que as sessões tivessem qualidade distintas, o conteúdo original era transcodificado ou codificado em diferentes taxas de bits, tornando-o apropriado para criação daquelas sessões. Quanto maior a taxa de bits (*bitrate*), maior o nível de qualidade do vídeo. As taxas de bits de vídeo podem variar de qualidade 4K (cerca de 25 Mbps) à qualidade de definição padrão - *Standard Definition* (SD) - medidas em kbps. Quando a audiência reproduz um vídeo, o player de vídeo busca os segmentos em sequência, usando requisições HTTP no modo range [57], numa conexão persistente entre player e servidor.

A escolha da qualidade da sessão no seu início tinha consequência na experiência da audiência. O desalinhamento entre a taxa de transmissão da conexão e a taxa de codificação dos segmentos produzia períodos de interrupção da sessão ou obrigava que se experimentasse longos períodos de buferização no início da sessão. Esses distúrbios na sessão motivou provedores de conteúdo, e.g., Apple [2], Adobe [1] e Microsoft [3], a desenvolver tecnologias que ficaram conhecidas sob o nome guarda-chuva de *Streaming* de Taxa Adaptativa sobre HTTP, do inglês (HAS - HTTP Adaptive Bitrate *Streaming*). O processo de codificação do vídeo divide o arquivo em segmentos de duração fixa e cada segmento é codificado em diferentes taxas de bits. O Player da audiência é capaz de adaptar a taxa de bits da sessão, dentro do conjunto de taxas disponíveis no servidor. O algoritmo que é executado pelo player, realiza a adaptação daquela taxa (*Adaptive Bitrate*-ABR), considerando diversas medidas coletadas da sessão em andamento e/ou sessões anteriores. Na próxima seção, mostra-se o esforço de padronização das tecnologias geradas por grandes provedores, com vista na ampliação das audiências, redução de custos de operação e interoperabilidade de soluções.

2.1.1 *Streaming* de vídeo Adaptativo Dinamicamente sobre HTTP

Em 2010, o Fluxo Contínuo Dinamicamente Adaptável sobre HTTP (DASH), do inglês *Dinamic Adaptive Streaming over HTTP*, foi desenvolvido pelo MPEG (*Moving Picture Experts Group*), tendo como ponto de partida as melhores práticas produzidas sob o guarda-chuva HAS. Em 2012, a primeira versão do padrão foi publicada oferecendo a possibilidade de construir fluxo contínuo de conteúdo multimídia de alta qualidade na Internet [38]. Os pilares do padrão DASH são: i) agnóstico em relação a mídia; ii) aplicação cliente é quem controla a qualidade da mídia na sessão; iii) reconhece a heterogeneidade das audiências.

O padrão definido abarca todas as mídias, portanto é possível usar o padrão para criar canais de áudio, por exemplo, com qualidade adaptável. Toda sessão é iniciada e controlada pela aplicação cliente, que tem a responsabilidade de estabelecer a melhor taxa de bits, i.e., a qualidade do som e da imagem, em um certo período da sessão. Por fim, as audiências acessam o conteúdo de diferentes formas e com recursos diversos, i.e., plataformas de hardware e software diferentes, por isso o padrão permite que a mídia esteja codificado com diferentes qualidades e formatos. Com essas diretivas, o padrão garante que uma audiência heterogênea seja contemplada nas aplicações.

O cliente DASH, quem controla o player de vídeo, deve ser capaz de criar as sessões de vídeo com qualidade adaptável. As decisões que geram essa adaptação são baseadas em variáveis que capturam o estado da aplicação cliente e o estado do canal estabelecido. A avaliação de forma isolada ou combinada desses fatores produz uma lógica de adaptação da qualidade que é um dos pilares da aplicação cliente. A Figura 2 mostra os componentes da arquitetura de uma aplicação DASH, como destaque aos fatores que são usados para tomadas de decisão.

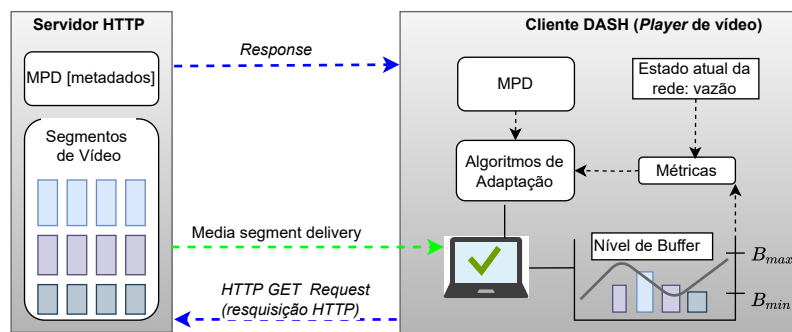


Figura 2: Arquitetura cliente-servidor do DASH para criação de sessões de vídeo com taxa adaptável. Fonte: Autoria própria.

A dinâmica de requisição e resposta envolvendo cliente e o servidor de vídeo permite monitorar a natureza instável do canal de comunicação, medindo-se, por exemplo, a flutuação da largura de banda (throughput) e do atrasos (jitter). Devido ao papel relevante que o estado do canal possui, é preciso compensar a sua instabilidade de alguma forma. A bufferização de conteúdo tem sido utilizada para compensar os efeitos dessa instabilidade na construção de sessões de melhor qualidade. O cliente mantém um *buffer* de reprodução que armazena alguns segundos de vídeo com o objetivo de compensar o descompasso entre a taxa de bits usada na codificação do vídeo e a vazão do canal de transmissão. O *buffer* registra do tempo total de vídeo armazenado nele, que é chamado de nível de buffer, e como é finito, possui uma capacidade máxima. Com o vídeo bufferizado, as lógicas de adaptação estabelecem limites, e.g., mínimo (B_{min}) e máximo (B_{max}), para tomar decisões que minimizem os efeitos da variabilidade da vazão, protegendo o fluxo de vídeo de interrupções e sobrecarga do cliente. Quanto maior o tamanho do *buffer*, menor a probabilidade de interrupção [36], por outro lado, maior será o custo de operação no caso de abandono de sessões.

Um servidor DASH armazena as representações do vídeo e um arquivo de descrição, i.e.,

MPD (*Media Presentation Description*), que possui os metadados que o descrevem. A apresentação dos metadados é feita em formato XML (*Extensible Markup Language*). As principais informações contidas no MPD são: i) URL (*Uniform Resource Locator*) das representações; ii) quais taxas de bits estão disponíveis; e iii) informação da largura de banda máxima e mínima esperada. As informações contidas no MPD, o estado da rede e da aplicação são as entradas que uma lógica de adaptação da qualidade precisa ajustar às requisições de segmentos de uma sessão em curso.

2.1.2 Modalidades de entrega de *Streaming* de vídeo

As aplicações multimídia da Internet podem ser entregues ao cliente em diferentes formatos, tamanhos, protocolos, modalidades e links de comunicação. Para as aplicações de *Streaming* de vídeo, o consumo pode ser realizado na modalidade de vídeo sob demanda (*Video-on-Demand*) ou ao vivo (*Live Streaming*). O vídeo sob demanda assume que o vídeo esteja armazenado em algum servidor e disponível para visualização a qualquer momento. Já para o *Streaming* de vídeo ao vivo, o conteúdo de vídeo é transmitido em tempo real à medida que os eventos acontecem.

Na aplicações *Video-on-Demand* (VoD), os vídeos pré-gravados (filmes, programas de TV e jogos) são armazenadas no servidor e consumidos pela audiência em seu próprio tempo. O YouTube, Hulu e Netflix são exemplos de provedores comerciais que oferecem a entrega de conteúdo de vídeo nessa modalidade [36]. Como foi explicado na seção 2.1.1, antes de iniciar-se a reprodução de vídeo no player do cliente, o arquivo MPD é baixado com as configurações de como e quando os segmentos devem ser executados. O conteúdo é gerado antecipadamente antes de ser enviado ao cliente. Ao ser baixado, os segmentos são alinhados e remontados de modo a reconstruir o arquivo original. Nesse tipo de serviço, o nível e capacidade do *buffer* de reprodução podem ser maiores e um único arquivo MPD pode ser baixado para a sessão. Dessa forma, aplicações VoD podem ser executadas com um conjunto mínimo de requisitos.

O vídeo de fluxo contínuo ao vivo necessita que a rede apresente uma vazão média superior a taxa de consumo do vídeo devido à restrição de tempo inerentes ao seu contexto. Como esse tipo de serviço abrange uma certa audiência ou grupo de visualizadores, o atraso é uma questão desafiadora [36]. Em outras palavras, manter em nível aceitável a sincronia entre todas as sessões criadas em torno de um vídeo é uma questão em aberto. A sincronia é afetada pelo atraso experimentado por cada participante da sessão, enquanto o atraso, por sua vez, é determinado por fatores não controlados pela aplicação. A consequência mais visível dessa não sincronização dos participantes pode gerar uma sensação de que a sessão em curso perdeu a característica de ser ao vivo. Uma diferença fundamental entre aplicações ao vivo e VoD é que a capacidade do *buffer* deve ter um limite reduzido para sessões ao vivo em razão da latência.

Em sessões de vídeo de ambos os serviços, VoD e ao vivo, as características naturais da rede impõe restrições no desempenho desses serviços, que afetam diretamente a qualidade de experiência da audiência. Esse último conceito leva em consideração como os consumidores

percebem o valor geral de um serviço. Esse princípio de HAS será explicado com mais detalhes na próxima seção.

2.1.3 Qualidade de experiência em HAS

A qualidade de experiência (*Quality of Experience*) é a avaliação de qualidade que representa a satisfação da percepção do usuário sobre a qualidade de um serviço, sistema ou aplicação [12]. A *Quality of Experience* (QoE) está relacionada ao nível de prazer ou aborrecimento de um serviço ou aplicação que o usuário utiliza [27]. Neste trabalho, o conceito de QoE, bem como os fatores que a influenciam e suas métricas são apresentados e explicados no contexto de entrega de vídeo sob o aspecto de HAS, de um ponto de vista da lógica de adaptação executada no player de vídeo do cliente por meio de heurísticas de adaptação de taxa. A medição da QoE, tal qual seu *feedback* pelo usuário final, é importante para os provedores de serviços de vídeos, pois, com isso, podem melhorar seus serviços e requisitos de *Streaming* de vídeo.

Correlacionado à QoE, destaca-se o aspecto de qualidade de serviço (*Quality of Service* - QoS) que contribui no impacto da qualidade dos serviços de vídeo. No ambiente de redes de telecomunicação, a QoS é designada como a configuração de um serviço de telecomunicação focado na satisfação do usuário, expressando, objetivamente, parâmetros de rede, como perda de pacote, jitter no envio de pacote e atraso [54]. Uma boa QoS nem sempre pode garantir um serviço de qualidade para a audiência [27]. Os parâmetros de QoS são mais concentrados na recepção de imagem e vídeo (métrica voltada para a rede) e, por isso, não são capazes de quantificar a satisfação do usuário. Por outro lado, a QoE centra-se na percepção do usuário, a qual envolve fatores humanos e contextuais. Dessa forma, a QoE é uma extensão da QoS [22].

Nos serviços de fluxo contínuo de vídeo, alguns fatores influenciam a qualidade percebida. Tais fatores estão ligados a qualquer característica expressa por parte do usuário, sistema, serviço, ou até mesmo em contextos específicos do usuário como localidade e hora [27]. Dependendo de seu estado atual, essas razões implicam, de alguma forma, na QoE. Tais fatores mantêm uma relação entre si e podem ser agrupados em: fatores de influência humana, sistema, contexto e conteúdo [12, 27]:

- Fator de influência humana considera as características humanas do usuário, como o estado psicológico, físico, mental e emocional, além do contexto demográfico, cultural, educacional;
- Fator de influência do sistema está relacionado às propriedades técnicas do sistema no caminho de transmissão de vídeo do *host*. São fatores relacionados ao dispositivo, conteúdo, mídia e ao processo de rede;
- Fator de influência de contexto refere-se ao contexto da situação do usuário, como contexto físico (característica local e espacial), contexto temporal (tempo de frequência e duração do vídeo) e contexto econômico (custo e marca do dispositivo, tipo de assinatura);

- Fator de influência de conteúdo representa as características do vídeo como taxa de codificação, resolução, formato, duração de reprodução, qualidade do vídeo, dentre outros.

Como pode ser observado, qualidade de experiência é uma medida subjetiva que depende do contexto e expectativa do usuário. Obter a experiência desse usuário, em relação à qualidade perceptual de vídeo é um componente crítico e, ao mesmo tempo, um indicador-chave para o aprimoramento de serviço de vídeo. Em razão da diversidade de fatores e componentes incontrolláveis da rede, como flutuação da largura de banda e interrupções, a previsão de QoE torna-se não-trivial e complexa [51]. Portanto, estimar a QoE de acordo com esses fatores é importante para atender à expectativa do usuário final [6].

O *Streaming* de vídeo consiste em uma mistura de *download* e reprodução simultânea. No entanto, o serviço de melhor esforço da rede não possibilita a transmissão do fluxo de vídeo com vazão constante e garantida. Em consequência, o cliente HAS experimenta eventos de interrupção (*stall*), *rebuffering*, congelamento de vídeo, *delay* inicial e perda de pacotes, apesar da existência de soluções baseadas em transcodificação de vídeo e codificação escalável [34]. Esses fatores apresentam-se em canais de rede heterogêneos, o que torna complexo o desenvolvimento um modelo de QoE de propósito geral para HAS [50].

A entrega de conteúdo multimídia por redes IP (*Internet Protocol*) é impactada pelo modelo de serviço implementado nessas redes. Em uma rede compartilhada, pontos de gargalos podem ocorrer em vários locais, como na extremidade do servidor, na interconexão de redes ISP (*Internet Service Provider*), no *link* de acesso, ou na conexão de última milha (conectividade do usuário final a um provedor de acesso por meio de um provedor intermediário). Quando o tráfego sobrecarrega esses gargalos, ocorrem *delay*, *rebuffering* e perdas de pacotes [34].

A perda de pacote é resultante do congestionamento da rede e em virtude da chegada atrasada de pacotes no *buffer* de reprodução. Quando tal evento ocorre, o decodificador de vídeo no *player* não consegue reconstruir o fluxo devidamente, resultando na degradação de qualidade do vídeo. Eventos de atraso podem ser definidos como a quantidade de tempo que o pacote leva para chegar ao destino. Quando o atraso excede um certo limiar, isso resulta em congelamento e perda de blocos de vídeo [46]. A taxa de atraso varia de acordo com a característica do serviço de vídeo. Já o atraso de inicialização representa o tempo necessário para preenchimento do *buffer* no início da sessão e, seu valor mínimo depende da taxa de dados de transmissão disponível [54].

No mesmo sentido, outra questão que afeta a qualidade da sessão de vídeo é a variação do tempo de chegada do pacote no *buffer* de reprodução, ou seja, o jitter, em razão das diferentes rotas seguida pelos pacotes até alcançar o mesmo destino. Tais eventos causam lentidão no preenchimento do *buffer* de reprodução e, em condições extremas, o seu esvaziamento, gerando *rebuffering* [34]. Esses são fatores de qualidade de serviço dependentes de elementos de rede e que influenciam a qualidade de experiência em diferentes níveis. Para atenuar esses fatores, estratégias ou heurísticas de adaptação são projetadas no player da audiência.

Com as diferentes estratégias de adaptação executadas no lado do cliente, pode-se coletar, dinamicamente, vários eventos e métricas que podem ser utilizados para medir a QoE da sessão

e a avaliação de desempenho de tais estratégias à medida que os segmentos são renderizados durante a sessão em curso, como a taxa de bits média (R_{ave}), oscilação da taxa de bits média (R_{osc}), tempo de download, tamanho de cada segmento, duração de cada evento de *rebuffering* e cada mudança de taxa. O número de *rebufferings* (N_{rebuf}) representa a fração do tempo de suspensão da execução do conteúdo de vídeo (*rebuffer*) até que o *playout buffer* esteja preenchido com uma qualidade suficiente de vídeo para retomar a reprodução. Logo, o número de *rebufferings* é igual ao tempo total de *rebuffer* dividido pela duração da sessão. A taxa de bits média é a média da taxa de bits de todos os segmentos renderizados. A oscilação da taxa de bits média é a diferença média nas taxas de bits de segmentos renderizados consecutivamente, sendo definida como segue:

$$R_{osc} = \frac{1}{N_t - 1} \sum_{i=1}^{N_t-1} |bitrate(i) - bitrate(i + 1)| \quad (2.1)$$

onde, *bitrate* é a taxa de bits do i -ésimo segmento renderizado e N_t é o número de segmentos usados na sessão até o instante t .

Durante a sessão, os algoritmos de adaptação de taxa de bits (ABR) visam a maximização da QoE percebida pela audiência ao minimizar as mudanças de qualidade do conteúdo, porém, esse processo dinâmico causa instabilidade na qualidade da taxa de codificação de vídeo. No aspecto dessa abordagem, o espaço de otimização considerado é definido pela taxa de bits dos segmentos, eventos de *rebuffering* e as oscilações da qualidade [8].

A próxima seção apresenta várias abordagens de algoritmos de adaptação de taxa, utilizadas em players de reprodução, por exemplo, aqueles utilizados no MPEG-DASH.

2.1.4 Algoritmos de adaptação de taxa de bits

Em uma sessão HAS, dois objetivos conflitantes precisam ser harmonizados: i) as trocas desnecessárias de qualidade; e ii) a redução do atraso inicial da reprodução. As trocas de qualidade mediante oscilações temporárias da vazão são impedidas pelo uso de conteúdo armazenado em um *buffer*. Esse *buffer* é formado no início da sessão, quando a reprodução do conteúdo acessado é postergada em alguns segundos, impondo um atraso inicial à sessão, e é mantido a partir de acessos aos segmentos com qualidade reduzida.

O acesso aos segmentos é guiado por um algoritmo que tenta compatibilizar a taxa de bits, usada na codificação do vídeo, com a taxa de transferência da conexão. Essa compatibilização é um problema complexo, e tem sido abordado sob diferentes perspectivas. O corpo de trabalhos que abordam esse problema é robusto e pode ser agrupado em três categorias: i) baseado em vazão, ii) baseado em *buffer* e iii) abordagens híbridas. Os algoritmos baseado em vazão selecionam a taxa de bits do próximo segmento, a ser acessado, avaliando uma estimativa de vazão futura. Os algoritmos baseados em ocupação do *buffer* decidem verificando a ocupação de *buffer*, sem utilizar informação sobre a vazão [61]. Os algoritmos híbridos utilizam tanto a

estimativa de vazão futura quanto a ocupação do *buffer*, na tentativa de explorar as vantagens de ambas as abordagens.

Em virtude deste trabalho investigar métodos de estimativa de vazão, no restante desta seção, apresentam-se exemplos de algoritmos de adaptação de taxa de bits que fazem uso dessa métrica para tomada de decisão sobre a qualidade dos próximos segmentos.

Um exemplo de algoritmo baseado em *buffer* é o *Buffer Occupancy-based Lyapunov Algorithm* (BOLA) proposto em [58]. No BOLA o problema de adaptação de taxa de bits às flutuações da vazão do canal é modelado como um problema de maximização de uma função de utilidade arbitrária, ou função de seleção de taxa de bits, que mapeia o nível de *buffer* atual para a taxa de bits (em kbps) do próximo segmento a ser baixado. Tal função está associada com a taxa de bits média e tempo de *rebuffering*, enquanto se adapta às flutuações do canal para quantificar a QoE. Então, um algoritmo é projetado utilizando a otimização de Lyapunov, que garante que a utilidade alcançada esteja dentro de um certo limite da solução *off-line* ótima.

O PANDA - (*Probe AND Adapt*) é um algoritmo baseado em vazão, que teve inspiração no controle de congestionamento do TCP. Ele trabalha para eliminar o problema causado pelos períodos de *ON-OFF*, durante o estado de equilíbrio da sessão. Além disso, o algoritmo reduz as variações de taxa de bits quando vários clientes compartilham o mesmo *link* de gargalo. Em uma sondagem de rede, uma taxa de dados é configurada pelo cliente, então, um segmento com taxa de bits é requisitado e o intervalo de tempo entre requisições consecutivas é calculada para se ajustar com esta meta. O nível de preenchimento de *buffer* é observado nessa dinâmica, como uma informação extra visando reduzir a ocorrência de eventos de parada.

Um algoritmo preditivo, chamado de MPC (*Model Predictive Control*), foi proposto em [62]. O MPC prever a vazão esperada do próximo par de segmentos de vídeo e a combina com sinais de retorno de ocupação do *buffer* para a tomada de decisão sobre taxa de bits que levaria a maximização da QoE. O MPC resolve o problema de seleção de taxa caracterizando-o como um problema de controle estocástico.

A concepção de um algoritmo ABR ideal ainda representa um desafio nos sistemas de vídeo de fluxo contínuo adaptável, pois estes precisam estimar, de modo preciso, condições de rede com a dinâmica da vazão disponível [34]. Variações das abordagens descritas acima são utilizadas em aplicações HAS do mundo real, mas não existe uma abordagem amplamente aceita [61]. A maioria das implementações emprega técnicas de predição simples e estimativa de vazão futura, como a média do número de taxa de vazões passadas da sessão em andamento [10], conduzindo à subestimação de recursos no servidor.

2.1.5 Médias Móveis

As médias móveis são técnicas estatísticas fundamentais para estimar a largura de banda em sistemas HAS, como DASH, HLS e Smooth Streaming. Elas servem para suavizar e interpretar as flutuações das medições de rede em tempo real, permitindo decisões de adaptação mais estáveis e eficientes. Ela filtra ruídos de curto prazo e revela a tendência de médio/longo

prazo da largura de banda. Os tipos comuns de médias móveis usadas são Média Móvel Simples (SMA), Média Móvel Exponencial (HM) e Média Móvel Exponencial Ponderada (EWMA).

2.1.5.1 Média Móvel Simples

A média móvel, do inglês *Moving Average* (MA), é um método usado no processo de suavização de dados, especialmente para dados de séries temporais que visam estimar a tendência dos dados [23], por ser vista como uma janela de tempo que se move ao longo do sinal de entrada, executando uma média local dos quadros vizinhos relacionados à continuidade da sequência do alvo. Seu valor é continuamente recalculado à medida que novos dados são disponibilizados, ela segue descartando o valor mais antigo e adicionando o mais recente. O limiar da janela pode ser entre curto e longo prazo dependendo da aplicação, e seus parâmetros são definidos em concordância.

Um modelo de média móvel é conceitualmente uma regressão linear do valor atual da série em relação à janela corrente e em termos do erro observado anteriormente. O método visa reduzir o efeito das variações temporárias nos dados, melhorar o ajuste dos dados com um processo chamado “suavização” para mostrar a tendência dos dados mais claramente e destacar qualquer valor acima ou abaixo da tendência. Uma média móvel simples (do inglês, *simple moving average* - SMA) é definida como segue:

$$SMA = \frac{M_1 + M_2 + \dots + M_{n-1}}{n} = \frac{1}{n} \sum_{i=1}^{n-1} M_i$$

onde os valores de M representam valores em uma janela de tempo e n o tamanho da janela.

A taxa de transferência média passada para prever horizontes futuros é utilizada em muitas aplicações. O modelo auto-regressivo ARIMA (do inglês, *Autoregressive Integrating Moving Average*), por exemplo, consiste em três componentes, os modelos $AR(p)$, $MA(q)$ e $ARMA(p, q)$, onde p é o número de termos autorregressivos e q é o número de previsão defasada na equação de previsão [63]. Embora os modelos como ARIMA e Média Harmônica, (HM) possam resolver muito bem o problema de previsão em séries temporais, a precisão da predição ainda não é muito alta para a série com grandes flutuações, principalmente quando há ausência de dados.

2.1.5.2 Média Móvel Exponencial Ponderada (EWMA)

A Média Móvel Exponencialmente Ponderada (do inglês, *Exponentially Weighted Moving Average* - EWMA) é uma média ponderada dos últimos n valores, em que a ponderação diminui exponencialmente a cada dado/período, isto é, dar mais importância aos dados mais recentes e à medida que os dados ficam antigos a significância da informação diminui exponencialmente. Essa média reage de maneira mais significativa às pequenas mudanças dos dados do que uma SMA, que aplica peso igual à todas as n observações.

Apesar do método responder mais rapidamente às flutuações dos valores do que uma SMA, o método é mais sensível aos ruídos nos dados. A EWMA é definida como segue:

$$\hat{S}_t = B [S_t + AS_{t-1} + A^2S_{t-2} + \cdots + A^nS_{t-n}] + A^{n+1}S_{t-(n+1)}$$

onde B é um fator de suavização constante entre 0 e 1, A é $(1 - B)$, os S 's são observações de períodos e o t indica a ordenação temporal das observações.

2.1.5.3 Média Harmônica (HM)

A média harmônica (do inglês, *Harmonic Mean* - HM) é uma média usada para fornecer uma agregação entre os operadores max e min, e é amplamente usada como uma ferramenta para agregar dados de tendência central [63]. Geralmente, é considerada uma técnica de fusão de informações numéricas de dados. É calculado dividindo o número de observações pelo inverso multiplicativo, definido como $\frac{1}{x}$ ou x^{-1} , de cada valor x na série, como segue:

$$\hat{H}_i = K / \left(\sum_{k=1}^K \frac{1}{H_{i,-k}} \right) z$$

onde $\hat{H}_i$ é a taxa de transferência futura prevista e $H_{i,-k}$ denota a vazão média do k^{th} ($k = 1, 2, \dots, K$) Δ – segundos antes da sessão i começar.

A HM é muito usada no contexto de problemas que envolvam grandezas inversamente proporcionais, tal como a vazão de redes. O método tende a mitigar o impacto de valores altos (como os *outliers*), e agravar o impacto dos valores baixos, fazendo com que cada um dos dados tenha peso igual. Diante disso, é utilizada para recuperar informações e estimar dados de séries temporais.

2.1.6 Predição de vazão em sistemas HAS

A vazão de uma conexão estabelecida durante a sessão de vídeo tem potencial para afetar o engajamento da audiência ao conteúdo. A principal razão para essa constatação é que a vazão tem potencial para produzir eventos de *rebuffering*, aumentar a duração do *delay* inicial e causar flutuações na qualidade das imagens apresentadas [40]. Explorar a dinâmica da vazão da rede diante de tais eventos e sobre largura de banda insuficiente é uma questão desafiadora. Nesta seção, destacam-se a relação entre os conceitos de largura de banda e vazão, e algumas formas de medição de vazão.

Existem diversos meios para transmissão de dados, cada um com uma largura de banda específica. Dada a suposição de que a largura de banda de um determinado canal é 10 Mbps, nem sempre se enviará ou receberá 10 Mbps, mas, a quantidade máxima de dados que pode ser transferida por esse canal é de 10 Mbps. Nesse sentido, a vazão assume um papel importante, pois representa a quantidade total de bits recebida por segundos, de acordo com a largura de banda do canal de rede [43].

O fluxo de vídeo pode ser modelado como um conjunto de segmentos $\{s_i\}$, onde $i \in \{1, 2, \dots, N\}$. Dado que, cada segmento de vídeo contém T segundos com diferentes taxas de bits, o tamanho total do fluxo de vídeo é definido como sendo $N \times T$ segundos. Um segmento s_i pode ser baixado com uma taxa de bits $q_i \in \{Q_{min}, \dots, Q_{max}\}$ [bit/s] no tempo t_i pelo *player* de vídeo, onde Q_{min} e Q_{max} representam os limiares do conjunto de níveis de taxa de bits disponíveis. O tempo para baixar um segmento depende da taxa de bits q_i selecionada e da vazão média da rede denotada como r_i [bit/s] [10]. No contexto de um serviço HAS, os algoritmos de adaptação baseados em vazão calculam a vazão utilizando o tamanho dos segmentos e o tempo de download:

$$r[i] = \frac{Q_i * D_i}{T_{down(i)}} \quad (2.2)$$

onde, T_{down} é o tempo de download e D é a duração do i -ésimo segmento.

Em um sessão de vídeo, o *player* ao selecionar a taxa de bits q_{i+1} no tempo t_i possui conhecimento da vazão passada $\{r_i, t \leq t_i\}$, enquanto a vazão futura, denotada como $\{r_i, t > t_i\}$, é desconhecida. Sendo assim, o *player* de vídeo pode utilizar um preditor de vazão $\{\hat{r}, t > t_i\}$. Então, considera-se, em uma abordagem híbrida de adaptação de taxa, o nível de ocupação do *buffer* e a estimativa de vazão futura, para selecionar a qualidade do próximo segmento s_{i+1} , como: $q_{i+1} = f(b_i, \{\hat{r}, t > t_i\})$, tal que $b_i \in [0, B_{full}]$ é o tamanho do vídeo armazenado no *buffer* B_{full} .

No problema de predição de vazão de uma conexão, medições passadas da vazão $\{r_t, r_{t-1}, \dots, r_{t-n}\}$ são levadas em consideração, uma vez que se deseja prever o valor de r_{t+m} , $m > 0$. O preditor $\hat{r}_{t+m}$ pode ser descrito como segue: $\hat{r}_{t+m} = \phi(r_t, r_{t-1}, \dots, r_{t-n})$. O maior problema consiste em escolher ϕ , uma vez que $\hat{r}_{t+m}$ é próximo de r_{t+m} [10], além das medições passadas apresentar algum tipo de viés. Então, o algoritmo que adapta a taxa de bits do fluxo de vídeo pode, de modo geral, utilizar a vazão média expressa da seguinte forma:

$$\hat{r}_{t+1} = \frac{1}{n} \sum_{k=1}^n r_{t+1-k}.$$

A utilização da vazão média é uma maneira simples de realizar a predição da vazão futura, sendo um método de estimação adotado em vários algoritmos ABR [61]. Outras formas de predição são baseadas em modelos estatísticos clássicos, e.g., Modelo Auto-Regressivo Integrado de Médias Móveis, do inglês, *AutoRegressive Integrated Moving Average* - ARIMA, uma vez que são ajustados aos dados da série temporal para entender melhor os dados ou para prever pontos futuros na série. Esse conceito é utilizado para estimar a vazão por meio da média móvel da vazão dos últimos segmentos através de uma janela deslizante. No entanto, a estimação de vazão baseada em média móvel é muito dependente da medição do canal atual (por exemplo, média móvel dos k segmentos baixados anteriormente) por onde trafega-se a vazão de rede, tornando-se sensível ao comportamento instantâneo de tal.

A próxima seção apresenta a teoria relacionada à aprendizagem de máquina, mostrando as técnicas e o processo de inferência estatística e matemática do processo de aprendizagem de modelos matemáticos construídos a partir de várias funções matemáticas simples. Além disso, a seção enfatiza o processo de aprendizagem de tais modelos cuja características envolvem a modelagem de dados sequenciais no tempo, tais como redes neurais recorrentes e redes neurais de convolução, adequadas ao problema de predição de vazão em questão.

2.2 Aprendizagem de máquina

Aprendizagem de Máquina é um subcampo da inteligência artificial capaz de criar sistemas robustos para realizar tarefas automáticas por meio da detecção de padrões na natureza dos dados. Esses sistemas inteligentes são algoritmos ou modelos matemáticos que aprendem o comportamento da distribuição estatística dos dados, apresentando, em sua saída, regras importantes para a tomada de decisão na resolução de problemas.

Muitos problemas e aplicações de negócios do mundo real de diversas natureza podem ser resolvidos com a intervenção de modelos de aprendizagem de máquina. Os problemas são explorados e estudados na tentativa de encontrar informações nos dados e *insights* valiosos que possam servir de entrada para modelos de aprendizagem de máquina.

Em sua maioria, os algoritmos de aprendizagem de máquina são treinados para a criação de modelos de classificação ou predição. A predição, que é foco deste trabalho, compreende dados qualificáveis com uma variável dependente e variáveis independentes, que são esboçadas para problemas de regressão ou série temporal. A classificação, por outro lado, exige que os dados sejam rotulados.

Alguns modelos, dependendo da regra de negócio, podem requerer razoável quantidade de dados e pouca complexidade no treinamento, que é o caso de algoritmos clássicos de aprendizagem de máquina.

Por outro lado, existem problemas para os quais é necessário centenas ou milhões de dados, exigindo uma complexidade matemática muito maior e tempo de processamento. É o caso das redes neurais artificiais, projetada para imitar a forma como o cérebro humano aprende.

As seções a seguir, apresenta dois algoritmos clássicos, frequentemente utilizados em tarefas de predição e classificação em aprendizagem de máquina, os quais foram estudados no presente trabalho a nível de comparação.

2.2.1 Random Forest

Random Forest (RF) ou Floresta aleatória é um meta classificador definido a partir de uma coleção de classificadores estruturados em árvore. Cada árvore é gerada a partir de um conjunto de vetores aleatórios, que pode ser obtido através de seleção aleatória de atributos (“característica”) e/ou seleção aleatória de amostra de árvore. Em sua implementação mais

simples, os vetores aleatórios são gerados independentemente dos vetores aleatórios anteriores, mas com a mesma quantidade de instâncias e atributos em cada sub-árvore.

O método pode ser usado tanto para classificação quanto para regressão. Sua saída no modo de classificação é construída sob votação de uma multidão de árvores de decisão no tempo de treinamento, que retorna a classe predominante. RF para regressão retorna a média das árvores de decisão da multidão de árvores de decisão para melhorar a precisão preditiva e controlar o viés. Esse processo consiste em dois passos:

- Amostrar k exemplos de treinamento de X (atributos do conjunto de treino), Y (classe); chamam estes X_k, Y_k ;
- Treinar uma árvore de regressão f_k em X_k, Y_k .

Após o treino das K árvores, as previsões podem ser calculadas usando a média das previsões de todas as árvores de regressão individuais em x' :

$$\hat{f} = \frac{1}{K} \sum_{k=1}^K f_k(x')$$

O RF é método variante do *bagging*, que cria n réplicas da base de treinamento através de seleção aleatória de amostras. A principal diferença entre os métodos é a escolha do tamanho de m “atributos” do subconjunto. Por exemplo, se um RF é construído usando $m = M$, total de “atributos” na base de dados, então isso é simplesmente equivalente ao *bagging*.

2.2.2 K-Vizinhos Mais Próximos (KNN)

O KNN (do inglês, *K Nearest Neighbors*) é um método baseado em observações de instâncias vizinhas e referido como um método de classificação clássico. O método assume que as instâncias podem ser representadas em um espaço, e.g. espaço euclidiano, em seguida, para rotular uma nova instância desconhecida x , as K instâncias mais próximas similarmente da nova instância x são identificadas para atribuição da classe à instância x . Por exemplo, usando k_i para representar o número de elementos na vizinhança de x que são da classe Y_i , temos:

$$P(Y_i|x) \approx \frac{k_i}{K}$$

Essa rotulação, após identificar as K instâncias mais próximas da nova instância x , acontece de forma diferente para ambos os tipos de modelo de KNN: para classificação usa-se a probabilidade condicional para a classe, ou seja, a saída é apoiada na classe mais comum entre os K vizinhos mais próximos, definida como segue [26]:

$$P(Y|X) = \frac{1}{K} \sum_{i \in N_0} I(y_i)$$

onde N_0 são os K pontos nos dados de treinamento que estão mais próximos; para regressão, a saída é o valor médio do alvo entre os k vizinhos mais próximos, definida como:

$$\hat{f}(x) = \frac{1}{K} \sum_{x_i \in N_0} y_i.$$

Ambos os tipos podem ser usados pondo pesos às contribuições dos vizinhos, de modo que os vizinhos mais próximos contribuam mais do que os mais distantes.

O método é sensível à localização dos dados no espaço. Um bom K pode ser selecionado usando várias técnicas heurísticas, por exemplo, escolher via método *bootstrap* [20]. Por esse motivo, a melhor escolha de K depende totalmente do dado, valores de K muito altos geralmente reduzem o efeito do ruído durante a rotulação, entretanto isso faz com que o método considere dados menos similares.

2.2.3 Redes Neurais Artificiais

Uma rede neural artificial é uma representação computacional simplificada inspirada na forma como o cérebro humano processa informação, retratada como uma coleção de neurônios artificiais interconectados. São formadas por funções matemáticas e otimizadores, dispostos em camadas, que aprendem sobre as representações dos dados. Neste trabalho, apresentam-se as redes neurais recorrentes e seus modelos variantes, utilizadas no problema de predição de vazão e, por fim, ressalta-se como uma rede neural neural convolutiva pode ser empregada na modelagem preditiva de dado sequencial- temporal.

No contexto de aprendizagem de máquina, um modelo aprende a tomar decisões a partir de amostras que lhe é apresentado. Esse processo envolve o treinamento do modelo com uma quantidade expressiva de amostras retiradas de um conjunto de dados representativo, isto é, dados constituídos de observações diversas. A modelagem preditiva precisa de dados e respostas para alimentar o modelo (*Input*), em seguida, realizar o processamento de dados (*Training*), e finalmente, apresentar, em sua saída (*Output*), as regras encontradas. Essas regras são aplicadas a novos dados na produção de respostas autênticas [14]. Um modelo de aprendizado de máquina, na verdade, faz uma inferência da estrutura estatística e matemática dos dados e respostas para, depois, criar suas próprias regras e automatizar uma dada tarefa [14].

Os modelos de aprendizagem de máquina vão desde algoritmos simples às redes neurais artificiais. As redes neurais artificiais são constituídas por unidades de processamento de dados, denominadas de neurônios artificiais, que representam funções matemáticas. Os neurônios são interconectados por conexões unidirecionais dispostas em uma ou mais camadas intermediárias diferentes, também chamadas de camadas ocultas (*hidden layer*). Esse arranjo de camadas e de neurônios interconectados constituem a arquitetura da rede neural. A Figura 3 apresenta os elementos-chave de um neurônio artificial.

A execução de um neurônio consiste na soma de $A = [x_1, \dots, x_n]$, que representa as informações ou sinais (*Inputs*) que chegam ao neurônio. Antes de serem somadas essas entradas

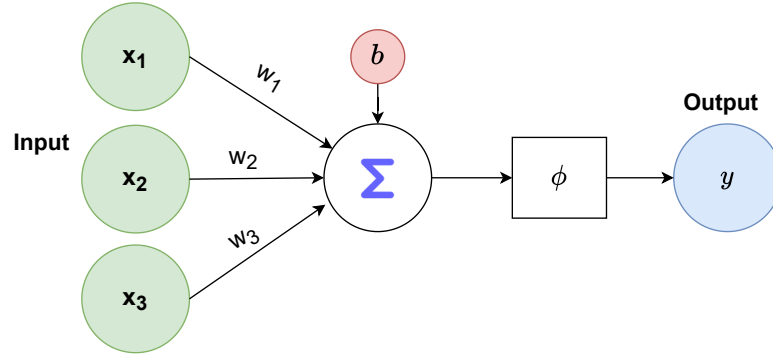


Figura 3: Modelo matemático de neurônio artificial. Fonte: Autoria própria.

são multiplicadas por seus pesos sinápticos (*Weights*) $W_k = [w_{k1}, \dots, w_{km}]$ e somadas com o viés b , definindo uma soma ponderada (*Weighted sum*). Em seguida, o resultado dessa operação passa por uma função de ativação (*Activation function*), que define a saída do neurônio. Na Figura 3, o viés b é dado por um vetor escalar e a entrada x e peso w são vetores de valores, onde, $x \in \mathbb{R}^n$ e $w \in \mathbb{R}^n$ com $n \in \mathbb{N}$. A fórmula, a seguir, descreve a função matemática que um neurônio representa.

$$y = \phi(z) = \phi(w^T x + b)$$

onde, $z = w^T x + b$ é a soma de vetores e o viés, que formam um argumento para a função de ativação ϕ , que por sua vez, realiza uma transformação não-linear de z , resultando na saída (*Output*) de informações do neurônio. Uma função de ativação decide se um neurônio deve ou não ser ativado, isto é, se a informação que o neurônio está recebendo é relevante para a saída do modelo ou deve ser ignorada.

A principal tarefa no campo de aprendizagem de máquina é a construção de algoritmos que possam aprender qualquer tipo de padrão presente nos dados. Um neurônio é bastante limitado em relação à inferência de padrões complexos dos fenômenos registrado nos dados. Dessa forma, não é possível aplicar um neurônio para aprender a natureza de dados com muita representatividade. Em razão disso, a solução de problemas complexos do mundo real demanda a construção de redes neurais com vários neurônios interconectados, produzindo assim, redes com diversas camadas, nesse caso, redes neurais profundas [14]. Um exemplo de rede com múltiplas camadas são as redes *Multilayer Perceptron* (MLP) ou *Feedforward Neural Network* (FNN).

O comportamento inteligente da FNN vem das interações entre suas unidades de processamento, dispostas em camadas ocultas que têm objetivos específicos. As camadas ocultas são extratoras de características e seus pesos designam uma codificação de características apresentadas nos padrões de entrada e permitem que a rede crie sua própria representação [21]. As informações são enviadas para uma camada totalmente conectada, depois, finalmente, enviados para a camada de saída, produzindo o padrão que será a resposta.

As FNNs são treinadas com um algoritmo chamado *Backpropagation*. A finalidade do *Backpropagation* é otimizar os pesos para que a rede neural possa aprender a mapear de

forma correta as entradas para as saídas. Esse algoritmo opera numa sequência de dois passos. Inicialmente, um padrão é apresentado à camada de entrada. Então, o nível de atividade na rede flui camada por camada, até que uma resposta seja produzida pela camada de saída. No segundo passo, a saída obtida é comparada à saída que se espera para o padrão que foi apresentado a rede. Se a saída esperada não estiver correta, o erro é calculado. Com isso, o erro é propagado da camada de saída até a camada de entrada para que o algoritmo corrija os pesos em todas as camadas. Então, os pesos dos neurônios das camadas ocultas são modificados, conforme o erro é retropropagado.

2.2.4 O processo de aprendizagem

O processo de aprendizagem de uma rede neural é iterativo e ajustável. A rede é treinada para realizar ajustes dos seus pesos e vieses. O aprendizado ocorre quando a rede consegue atingir uma solução generalizada para um determinado problema. O cálculo do erro de treinamento ocorre em todas as camadas da rede neural num processo no qual o objetivo é minimizar a diferença entre os valores de saída calculados - utilizando valores de entrada de dados de treinamento- e valores de saída de destino. Para isso, a rede neural precisa de uma combinação de parâmetros e técnicas de treinamento que envolve paradigma de aprendizado, quantidade de dados, otimizadores e função de custo.

Existem diversos modelos de redes neurais para diversas finalidades e cada um com suas particularidades específicas em relação ao problema que se deseja resolver. Para problemas em que deseja-se prever um alvo que representa uma ou mais variáveis categóricas, um problema de classificação pode ser definido. Quando o conjunto de dados apresenta o alvo como uma ou mais variáveis contínuas, tem-se um problema de regressão. A tarefa de previsão depende também do tipo de aprendizagem a qual o modelo de rede neural poder estar associado, que são:

- **Aprendizado supervisionado:** É um paradigma em que os padrões de entrada são fornecidos com a resposta para o modelo. Ou seja, os dados contém a variável dependente resultante das variáveis independentes observadas. Então, o modelo realizará uma previsão, conforme as respostas dos padrões de entrada desejadas.
- **Aprendizado não-supervisionado:** De acordo com esse paradigma, o modelo tem acesso apenas aos padrões de entrada e o modelo é forçado a aprender a distribuição dos dados.
- **Aprendizado por reforço:** nesse paradigma, o modelo aprende conforme uma política de penalidade e recompensa. O modelo precisa treinar bastante para aprender as características de um determinado ambiente dinâmico, fazendo interações do tipo “tentativa e erro”. Um agente avaliará a resposta produzida pelo modelo.

No processo de aprendizado de um modelo de rede neural, existe, para o conjunto de treinamento, um ciclo de X pares (entrada e saída). A correção de pesos pode ser feita de dois modos:

- **Modo Padrão:** a correção dos pesos acontece a cada apresentação ao modelo de um exemplo do conjunto de treinamento. Cada correção de pesos é baseada apenas no erro do exemplo apresentado naquela iteração. Assim, em cada ciclo, acontecem X correções.
- **Modo Batch:** Somente uma correção é feita por ciclo. Assim, todos os exemplos do conjunto de treinamento são apresentados ao modelo. Seu erro médio é calculado, e a partir deste erro, realizam-se as correções dos pesos.

Na aprendizagem supervisionada, deseja-se projetar um modelo sobre os dados de treinamento com a ideia de que tal modelo seja capaz de fazer previsões certas para dados nunca vistos antes, e que possuam as mesmas características do conjunto de treinamento. À capacidade de um modelo de realizar previsões precisas ou generalizar estão relacionados aos conceitos de viés e variância, uma vez que, o melhor compromisso tais significa que o modelo pode gerar uma solução suficientemente próxima da ideal.

O viés de um modelo expressa a incapacidade de um modelo de capturar a verdadeira relação entre as variáveis independentes e a variável dependente, assim quanto maior o viés menos o modelo aprende sobre aquela relação. Por outro lado, a variância de um modelo refere-se a quantidade de mudança esperada na variável dependente quando o treinamento é realizado em diferentes conjuntos de dado.

Por exemplo, em problemas de predição, um modelo com viés baixo e variância alta implica que tal modelo é capaz de reduzir o erro da predição no conjunto de treino, mas ao mesmo tempo o seu nível de generalização é baixo. Em outras palavras, no conjunto de treino, o modelo irá ter um erro de predição baixo, no entanto, ao ser utilizado em um novo conjunto de dados, i.e., o conjunto de teste, o erro aumenta consideravelmente, caracterizando o *overfitting*.

Um modelo com viés alto e variância baixa apresenta erro maior na fase de treinamento. Entretanto, o erro verificado na fase de teste/validação é próximo daquele identificado na fase de treinamento. Esse cenário caracteriza o fenômeno de *underfitting*. O objetivo, nesse caso, é reduzir o viés durante o treinamento, mantendo-se a variância controlada. Portanto, deve-se definir o melhor compromisso entre viés e variância do modelo durante o treinamento para que a capacidade de generalização seja maior.

Outro conceito importante em aprendizagem de máquina refere-se à capacidade do modelo. A capacidade está relacionada à complexidade do modelo. Se a complexidade do modelo for inferior à necessidade do problema, o erro de generalização (previsão) será elevado. Porém, se a capacidade do modelo for superior à necessidade do problema, o erro de generalização também será elevado. Portanto, deve-se definir a capacidade ideal do modelo durante o treinamento para que o erro de generalização também seja baixo, tanto quanto o erro de treinamento. No presente trabalho, utilizou-se uma quantidade significativa de dados para treinamento, tornando o modelo proposto complexo.

Na aprendizagem supervisionada, um algoritmo tem a responsabilidade de construir um modelo que deve examinar centenas ou milhares de exemplos com a ideia de encontrar uma

forma de minimizar a perda. Tal processo é chamado de minimização do risco empírico. O risco empírico representa erro de treinamento. Sua redução ocorre por meio da redução do erro de treinamento. Por exemplo, no contexto de um problema de classificação, ao tentar diminuir a quantidade de amostras de treinamento classificadas incorretamente, por meio de ajuste de parâmetros ou aumento do número de iterações de treinamento, busca-se minimizar o risco empírico.

2.2.5 Funções de ativação

A função de ativação quantifica as informações de *Input* do neurônio, e por conseguinte, transforma sua saída. Ou seja, essas funções possibilitam que pequenas mudanças nos pesos e vieses causem alguma alteração no *Output*. Esse ajuste refinado é importante para o desempenho de aprendizagem da rede neural. Nesta seção, são apresentados quatro funções de ativação: SELU, RELU (*Rectified Linear Unit*), Sigmoid e Tangente Hiperbólica.

A *Scaled exponential Linear Unit* - SELU é a função de ativação usada para construção de redes neurais auto-normalizadas. Ela garante estabilidade das entradas, mapeando-os de uma camada para outra, de forma que a sua média e a sua variância sempre estejam em torno de um ponto fixo, definido pelos valores originais da média e variância [32]. A SELU, denotada como γ , pode ser definida da seguinte forma:

$$\gamma(x) = \begin{cases} \lambda x & \text{se } x > 0 \\ \lambda(\alpha e^x - \alpha) & \text{se } x \leq 0 \end{cases}$$

onde α e e são constantes pré-definidas. Se x é maior do que 0, a saída resultante é x multiplicado por λ . Se o valor de x é menor ou igual a zero, logo, tem-se uma função que vai até 0, que é a saída y . α e λ são configuradas para 1,0507 e 1,6732, respectivamente, pois ajudam a tornar a distribuição de saída mais próxima do normal distribuição [32].

A função de ativação RELU [33], por sua vez, é utilizada em modelos de aprendizagem de máquina complexos, e tem como propriedade mapear os valores de saída no intervalo $[0, \infty)$. Esse mapeamento ocorre com a função atribuindo um valor zero para entradas com valor negativo, e mantendo o valor original das entradas com valor positivo. A seguir, mostra-se a definição da RELU.

$$RELU(x) = \begin{cases} 0 & \text{se } x < 0 \\ x & \text{se } x \geq 0 \end{cases}$$

Essa função ajuda a reduzir o custo de cálculo, ao acelerar o processo de treinamento. Além disso, promove a prevenção do desaparecimento de descida do gradiente, durante tal processo.

Já função de ativação *Sigmoid* possui como característica natural promover a não-linearidade dos dados. Os neurônios que atuam com tal função mantém seus valores de saída entre 0 e 1. Sua definição é como segue:

$$S(x) = \frac{1}{1 + e^{-x}}$$

Por fim, a função de ativação Tangente Hiperbólica (*Tanh*), assim como a *Sigmoid*, também é não-linear, com a diferença de que as saídas devem ser mapeadas para o intervalo $[-1, 1]$. Defini-se sua fórmula matemática da seguinte forma:

$$\text{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Ela é utilizada em camadas ocultas de redes neurais para ajudar a centralizar os dados, facilitando assim o aprendizado para camadas subsequentes.

Na condução do estudo realizado, utilizou-se as funções SELU e RELU, no processo de treinamento da rede neural do modelo proposto, em razão da performance de generalização obtida e a mitigação do desaparecimento do gradiente, como será apresentado no Capítulo 5.

Na próxima seção, apresentam-se as técnicas e métodos de otimização utilizados durante o processo de treinamento da rede neural como suporte para o ajuste de parâmetros.

2.2.6 Técnicas aprendizagem de máquina

Nesta seção, aborda-se as técnicas de aprendizagem de máquina utilizadas no presente estudo. As técnicas denominadas *Dropout* e *Batch Normalization* são aproveitadas para regularização e aceleração durante o processo de treinamento de rede neural, respectivamente.

2.2.7 Função de perda e método de otimização

A função de perda (*loss function*) ou função de custo é uma parte integrante de qualquer processo de aprendizagem de máquina supervisionado. Uma função de perda é usada para calcular o custo da diferença entre o valor previsto e o valor real. Não existe um consenso na literatura sobre a melhor função de perda [15], e por isso a escolha da função a ser utilizada depende do conjunto de dados e do problema que se deseja resolver. Em aprendizagem de máquina supervisionado, o objetivo é encontrar uma função de mapeamento ótimo que minimize a função de perda a partir das amostras de treinamento. A minimização da função de perda é feita por meio de algum método ou algoritmo de otimização, que por sua vez, tem um papel importante no treinamento de rede neural, pois tem implicações no seu desempenho.

A finalidade do algoritmo de otimização é encontrar valores de parâmetros que minimizam o erro da função de perda quando o modelo tenta mapear entradas para suas respectivas saídas. É um processo que visa encontrar um conjunto de parâmetros de entrada para a função de custo resultante na saída mínima ou máxima da função. A maioria dos algoritmos de otimização é baseada na descida do gradiente descendente. O gradiente descendente é uma função convexa cuja saída é uma derivada parcial de um conjunto de parâmetros de suas entradas. Quanto maior o gradiente, mais difícil é a inclinação. A ideia dos métodos baseados em gradiente descendente

é que as variáveis se atualizam iterativamente na direção (oposta) dos gradientes da função de perda. A atualização é realizada para convergir, gradualmente, para o valor ótimo de tal função. Neste trabalho, utilizou-se o algoritmo de descida estocástica do gradiente [52] para minimizar a função de perda.

Durante o processo de treinamento de uma rede neural, os neurônios da entrada da rede recebem informações ou sinais, como explicado na seção 2.2. As informações, assim, atravessam uma ou mais camadas e experimentam vários mapeamentos lineares e não lineares até a saída da rede; conhecida como saída de inferência. Este processo é chamado de propagação direta (*Forward Propagation*) que mostra uma direção da entrada para a saída. A diferença entre a inferência da saída e as informações de entrada é realizada pela função de perda, que é a responsável por medir o quão precisa é a saída gerada pelo modelo para uma determinada amostra de dados. Uma função de custo pode ser definida da seguinte forma:

$$J(\theta) = \frac{1}{N} \sum_{i=1}^N \frac{1}{2} (y^i - f_{\theta}(x^{(i)}))^2 = \frac{1}{N} \sum_{i=1}^N \text{cost}(\theta, (x^i, y^i))$$

onde, f_{θ} é uma função a ser aprendida, $J(\theta)$ é a função de custo, N a quantidade de amostras de treinamento, D é o número de entrada de características, x^i é a variável independente, tal que $x^i = (x_1^i, \dots, x_D^i)$ para $i = 1, \dots, N$, θ é o parâmetro a ser otimizado e y^i é a variável-alvo.

O algoritmo Gradiente Descendente Estocástico ou *Stochastic Gradient Descent* (SGD) é um dos métodos mais populares utilizados na tarefa de otimizar a função de custo de modelo aprendizagem profunda. A intuição por trás desse algoritmo consiste em minimizar a função de custo $J(\theta)$ pelos parâmetros $\theta \in \mathbb{R}^d$ de um modelo por meio da atualização de parâmetros na direção oposta do gradiente da função de custo $\nabla_{\theta} J(\theta)$ com relação aos parâmetros. O SGD pode ser calculado da seguinte forma:

$$\theta = \theta - \eta \nabla_{\theta} J(\theta; x^{(i)}; y^{(i)})$$

onde, ∇ representa uma derivada.

O SGD realiza a atualização para cada amostra aleatória i , a cada iteração do que para todas as amostras durante a fase de treinamento. Essa atualização é representada pela seguinte equação:

$$\theta' = \theta + \eta (y^i - f_{\theta}(x^i)) x^i$$

onde, a taxa de aprendizagem η determina o tamanho do passo em cada iteração e influencia o número de iterações até chegar ao valor ótimo. A complexidade computacional para cada iteração é $O(D)$, em que D é o número de características. Com o SGD, a taxa de atualização para cada iteração é muito mais rápida do que para outros métodos variantes da descida do gradiente, quando o número de amostras N é grande [60].

Portanto, o SGD realiza uma atualização de parâmetro para cada amostra de treinamento $x^{(i)}$ e rótulo $y^{(i)}$. A taxa de aprendizagem é um parâmetro que fornece ao modelo uma escala de quantos pesos devem ser atualizados. A ideia da descida estocástica do gradiente é utilizar uma amostra, aleatoriamente, para atualizar o gradiente por iteração, em vez de calcular diretamente o valor exato do gradiente. Uma característica inerente ao SGD é a redução do tempo de atualização para lidar com um grande número de amostras, além de remover uma certa quantidade de custo computacional, o que acelera significativamente o cálculo [60].

Neste trabalho, apresenta-se um problema que se caracteriza na abordagem de aprendizagem supervisionada com variáveis contínuas que envolvem sequências temporais, que pode ser definido da seguinte forma, do ponto de vista matemático: considere-se que z é um par de exemplo de treinamento (x, y) composto de uma entrada arbitrária x e uma saída escalar y , e a função de perda $J(\hat{y}, y)$ que mede o custo de prever $\hat{y}$, quando o valor atual é y , então, escolhe-se uma função $f_w(x)$ parametrizada por um vetor de peso w , para busca-se assim, a função f que minimiza a perda (loss) $Q(z, w) = J(f_w(w), y)$ sobre os exemplos. Na próxima seção, será visto a teoria de um tipo especial de rede neurais, chamadas de redes neurais recorrentes, empregadas para tarefas de modelagem de dados sequenciais.

2.2.8 Redes Neurais Recorrentes

As Redes Neurais Recorrentes, do inglês *Recurrent Neural Networks (RNNs)* caracterizam-se por modelar entradas de sequências de comprimento variável, principalmente, entradas de dados que exibem comportamento temporal. Essas redes têm sido usadas para resolver problemas que envolvem séries temporais, sequências de texto, genomas, caligrafia e palavra falada. A arquitetura de RNNs apresenta portões para armazenar as informações de entradas anteriores. O portão de uma RNN, na verdade, representa uma célula especial chamada de estado escondido recorrente, do inglês *recurrent hidden state*. Os estados escondidos possibilitam à RNN prever qual será a próxima entrada na sequência de dados de entrada. As RNNs possuem a habilidade de modelar sequências de dados de longo e curto prazo. Com isso, pode-se aproveitá-las para o desenvolvimento de aplicações de predição com aspecto e estrutura temporal.

Os estados de memória interna agem, de forma compacta, nas informações de entradas de sequências anteriores para aproveitá-las para as próximas entradas. A Figura 4 mostra um exemplo de arquitetura genérica de RNN. Nesse processo, o estado da memória é atualizado, recursivamente, com novas observações a cada passo de tempo (do inglês, *time step*). Por exemplo, ao considerar-se uma sequência $x = (x_1, x_2, x_3, \dots, x_T)$ de comprimento T , e uma célula de memória h_t no passo de tempo t , um modelo de RNN atualiza sua informação de memória da seguinte forma:

$$h_t = \sigma(W_x x_t + W_h h_{t-1} + b_t)$$

onde σ é uma função não-linear, e.g., a RELU ou Tanh, W_x e W_h são matrizes de peso e b_t é um

viés.

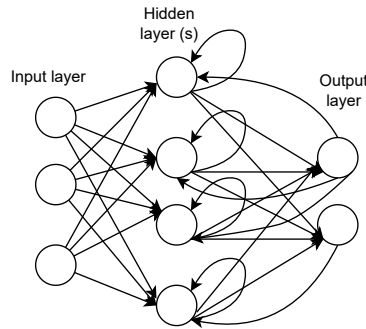


Figura 4: Arquitetura de RNN. Fonte: Autoria própria.

Ao utilizar h com entrada para outra RNN, pode-se desenvolver arquiteturas profundas [48] de RNNs, conforme a definição abaixo:

$$h_t^l = \sigma(W h_{t-1}^l + U h_{t1}^{l-1})$$

onde U é uma matriz de transição, semelhante a uma cadeia de Markov.

O treinamento das RNNs apresentam desafios que precisam ser observados para que a sua utilização seja efetiva quando o problema a ser resolvido envolve a captura de dependências de longo prazo [56]. Um desafio especialmente observado no treinamento das RNNs é o "desaparecimento do gradiente", do inglês *vanishing gradients*, e a "explosão do gradiente", do inglês *exploding gradients*. O desaparecimento do gradiente significa que a taxa de aprendizagem tende a zero rapidamente, pois as informações trazidas pelos gradientes, e que são usadas para atualização dos parâmetros da rede, tornam-se insignificantes. Por outro lado, o problema da explosão do gradiente acontece quando as informações de atualização dos parâmetros, os pesos, tornam-se muito grande a cada passo tempo. A consequência prática neste caso é a impossibilidade de representar o valor.

Variantes de RNNs foram projetadas para lidar com os problemas mencionados anteriormente. Incluem-se como variantes de RNNs a *Long Short-Term Memory* (LSTM), *Bidirectional LSTMs* e as *Gated Recurrent Unit* (GRUs). Os detalhes dessas arquiteturas de RNN serão explicados nas próximas seções.

2.2.8.1 Long Short-Term Memory

As redes neurais recorrentes de memória de longo prazo, do inglês *Long Short-Term Memory* (LSTM), são uma melhoria em relação às redes neurais recorrentes gerais pois tem a capacidade de aprender dependências temporais de longo prazo em sequências. As LSTMs resolvem tal problema, por meio da utilização de estados de células que armazenam informações de entrada de longo prazo, moduladas através de uma série de portões. Essa extensão de memória possui a característica de recuperar informações por um período maior de tempo, possibilitando leitura, escrita e exclusão de informações de suas memórias.

As LSTMs são constituídas de três portões: portão de esquecimento (*forget gate*), portão de entrada (*input gate*) e portão de saída (*output gate*). O portão de esquecimento toma a decisão de preservar ou remover as informações existentes, o portão de entrada especifica até que ponto as novas informações serão adicionadas à memória e o portão de saída controla se o valor existente na célula contribui para a saída. Matematicamente, os portões são definidos da seguinte forma:

$$\text{Input gate : } i_t = \sigma(W_{i_1}z_{t-1} + W_{i_2}y_t + W_{i_3}x_t + W_{i_4}s + b_i) \quad (2.3)$$

$$\text{Output gate : } o_t = \sigma(W_{o_1}z_{t-1} + W_{o_2}y_t + W_{o_3}x_t + W_{o_4}s + b_o) \quad (2.4)$$

$$\text{Forget gate : } f_t = \sigma(W_{f_1}z_{t-1} + W_{f_2}y_t + W_{f_3}x_t + W_{f_4}s + b_f) \quad (2.5)$$

onde z_{t-1} é o estado interno oculto da LSTM, σ é a função de ativação sigmoid, W é uma matriz de pesos e b vieses da rede, respectivamente. As matrizes de pesos são responsáveis em determinar a importância que deve ser dada tanto à entrada atual quanto ao estado oculto do passado. O erro que elas geram retornará por meio de retro-propagação e será usado para ajustar seus pesos até que o erro não diminua mais. Os portões modificam as células e estados escondidos da LSTM da seguinte forma:

$$z_t = o_t \odot \tanh(c_t),$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{c_1}z_{t-1} + W_{c_2}y_t + W_{c_3}x_t + W_{c_4}s + b_c)$$

onde $\odot$ é um produto elementar, do inglês *element-wise product*, $\tanh(\cdot)$ é a função de ativação tangente hiperbólica, c_t é uma célula de estado e z_t é o estado oculto.

Uma variante desse modelo é a LSTM bidirecional [53]. Nesse tipo de arquitetura, duas camadas de entrada de LSTMs são consideradas. Na primeira fase do treinamento, uma LSTM é aplicada na sequência de entrada-camada direta. Na segunda fase, a forma inversa da sequência de entrada é alimentada no modelo LSTM - camada inversa). Aplicar a LSTM duas vezes leva a melhorar o aprendizado de dependências de longo prazo e, conseqüentemente, melhorar a precisão do modelo [5].

2.2.8.2 Gated Recurrent Unit

Uma rede neural recorrente baseada em *Gated Recurrent Unit* (GRU) é semelhante a uma LSTM. A GRU [55] é um tipo de RNN utilizada para resolver o problema de explosão e desaparecimento de gradiente em RNNs tradicionais, ao ser utilizada para aprender dependências de longo prazo. As RNNs baseadas em GRU são compostas apenas por estados ocultos que transferem informações, por meio de dois portões: um portão de redefinição (*reset gate*) e um portão de atualização (*update gate*).

As GRUs podem ser treinadas para manter informações de muito tempo atrás, sem dissipá-las com o tempo ou remover informações irrelevantes para a previsão. Sua estrutura possibilita capturar, de modo, adaptativo, dependências de longa duração em sequências de dados por meio de suas unidades de portões. Seja r_t e u_t , respectivamente, o *reset gate* e *update gate*; as equações abaixo, mostram como as células nas camadas escondidas de uma GRU funcionam:

$$u_t = s(W_u \cdot [h_{t-1}, x_t]) \quad (2.6)$$

$$r_t = s(W_r \cdot [h_{t-1}, x_t]) \quad (2.7)$$

$$h'_t = (W \cdot [r_t * h_{t-1}, x_t]) \quad (2.8)$$

$$h_t = (1 - u_t) * h_{t-1} + u_t * h'_t \quad (2.9)$$

W é uma matriz de pesos, h_t é o estado escondido e h'_t é a saída da rede.

Nas configurações experimentais deste trabalho, explicadas no capítulo 4, utilizou-se essas três abordagens de RNNs para a predição de vazão com a finalidade de aproveitar o potencial de ambas. Além dessas redes, utilizou-se também uma camada de rede neural convolucional, visto que, tal rede também possui como característica a habilidade aprender características de dados sequenciais. A próxima seção apresenta os detalhes dessa rede.

2.2.9 Convolutional Neural Network (CNN)

Uma Rede Neural Convolucional [37], do inglês *Convolutional Neural Network* (CNN), é uma rede neural baseada em aprendizagem profunda que trabalha com o conceito de extração de características. Uma CNN reconhece objetos de uma imagem partindo de primitivas básicas, como por exemplo, patas e rabos de um cachorro. A construção desses conceitos mais abstratos ocorre à medida que os pesos da rede são ajustados, por meio de *Backpropagation*. A CNN é um tipo de rede neural *Feedforward* que adere a dados de duas dimensões (2D) e propriedade de conectividade local entre neurônios.

Uma rede baseada em CNN lida, basicamente, com seus pesos e vieses (bias), na detecção de objetos e aspectos de imagem, sendo capaz de diferenciá-los [18]. Uma das principais características desse tipo de rede é a convolução. A convolução é a aplicação simples de um filtro (campo receptivo) a uma imagem inteira de entrada (pode ser colorida ou multicanais), resultando em uma ativação. Os elementos de uma CNN apresentam-se num plano 2D, acrescidas de matrizes com largura ω e altura n . Essas matrizes são os *kernels* ou filtros.

A extração de características pelos filtros na imagem resulta em mapas de características. No processo de extração de características, os filtros são aplicados, repetidamente, formando-se mapa de ativações (resultado da convolução). O trabalho dessa rede consiste em aprender, por

meio dos filtros ao detectar mapas de características significativos, à medida que os pesos da rede são atualizados.

A camada de convolução é formada pelos componentes *Stride* (deslocamento) e *Padding* (preenchimento), seguida pela camada de *Pooling* (agrupamento) e uma camada totalmente conectada. O *Stride* é um parâmetro que modifica a quantidade de movimento sobre os pixels de uma imagem. Quanto maior o *Stride*, menos sobreposição entre campos receptivos e saída com menor dimensão. Uma rede neural pode ter sucessivas camadas convolutivas, sendo assim, pode detectar atributos de mais alto nível. Logo, os campos receptivos se tornam mais amplos, ou seja, com informações globais. *Padding* é o preenchimento realizado para deixar a imagem convoluída (características extraídas) com o mesmo tamanho da original.

As camadas de *Pooling* visam reduzir a dimensão da entrada com algum método pré-especificado, por exemplo, *maxpooling*, resultando em uma entrada menor, conservando o máximo de informações possível. Por fim, tem-se a camada totalmente conectada, que é uma unidade básica da camada oculta em FNN. Em CNNs, essa camada é frequentemente adicionada entre a penúltima camada e a camada de saída para modelar ainda mais as relações não lineares das características de entrada.

Uma modificação de CNN 2D foi desenvolvida para atuar com dados de uma dimensão (1D). As CNNs 1D [31] demonstraram um desempenho superior naquelas aplicações que possuem dados rotulados limitados e altas variações de sinal adquiridos de diferentes fontes [30]. Quanto a sua arquitetura básica, a camada de entrada de uma CNN 1D é uma camada passiva que recebe o sinal 1D bruto e a camada de saída pode ser uma camada MLP com o número de neurônios igual ao número de classes. Considerando-se o processo de aprendizagem com propagação de sinal à frente (*Forward*) e retropropagação *Backpropagation*, uma CNN 1D é definida da seguinte forma:

$$x_k^l = b_k^l + \sum_{i=1}^{N_l-1} conv1D(w_{ik}^{l-1}, s_i^{l-1}) \quad (2.10)$$

onde x_k^l representa a entrada, b_k^l é definido com um viés do k -ésimo neurônio da camada l , s_i^{l-1} é a saída do i^{th} neurônio na camada $l - 1$, w_{ik}^{l-1} é o kernel do i -ésimo neurônio l_1 para o k -ésimo na camada l .

No presente trabalho, utilizou-se uma camada de CNN 1D para modelar o comportamento das séries temporais de maneira sequencial dentro do espectro temporal. A princípio, a CNN 1D opera sobre os sinais de sequência 1D de cada série por meio de seus filtros (coleção de *kernels*) de convolução operando com vetores unidimensionais ao invés de matrizes. Feito isso, o sinal de saída da CNN 1D é transferido para a entrada da rede neural recorrente. No capítulo 4, apresenta-se o papel da camada CNN 1D foi utilizada no modelo proposto.

2.2.10 Considerações finais

Este capítulo apresentou o cenário do funcionamento do *Streaming* de vídeo sob HTTP com suas técnicas e protocolos utilizados para entregar vídeo com a melhor qualidade possível. Destacou-se o processo de preparação do servidor, transmissão até o player da audiência. Esse por sua vez controla a sessão a partir da medição de vazão do canal de rede e da verificação do *buffer* de reprodução. Além disso, apresentou-se alguns dos principais algoritmos de adaptação utilizados no protocolo DASH e as formas primitivas de estimação de vazão utilizadas por alguns mecanismos de adaptação com o propósito de manter a continuidade e melhorar a qualidade de experiência do usuário HAS. Em seguida, foi apresentado o funcionamento das redes neurais artificiais e seu processo de aprendizagem. Enfatizou-se as redes neurais recorrentes, um tipo de rede neural profunda que pode ser utilizada para realizar predição de dados com características temporais e sequenciais. E, por fim, destacou-se uma variante de rede neural profunda que utiliza *kernels* de convolução para filtrar dados de natureza sequencial, produzindo mapas de características que serão, posteriormente, combinados às redes neurais recorrentes.

CAPÍTULO 3

A base de dados Neubot

Este capítulo apresenta a análise da base de dados usada neste trabalho. Essa base é formado de *traces* de sessões de *Streaming* de vídeo adaptável de curta duração. O capítulo inicia enfatizando a forma como os dados foram coletados, sua origem, análise estatística do comportamento das diferentes variáveis que denotam um conjunto de séries temporais, por meio de um projeto de medição de desempenho de rede espalhado em várias regiões do mundo. Apresenta-se também o tratamento dos dados, bem como o processo de seleção de atributos do conjunto para escolher os atributos-chave que formarão o conjunto de dados que servirá para a modelagem preditiva com modelos de série temporal.

3.1 O Projeto Neubot

O conjunto de dados utilizado para realizar o estudo encaminhado neste trabalho foi coletado entre 2015 e 2018. Essa base apresenta 1100 horas de medições realizadas em várias regiões do Brasil, a partir de infraestruturas de operadoras de acesso, e.g. Claro, Telefônica e TIM. Os dados foram coletados no contexto do projeto Neubot¹, que realiza a medição de vazão a partir da borda da Internet com o apoio de milhões de usuários de vários países. Esse projeto, monitora os provedores de acesso à Internet, do Inglês *Internet Service Provider* - ISP, na finalidade de testar a neutralidade do serviço prestado. A realização do monitoramento é feita por meio de bots instalados em máquinas de participantes voluntários, que simulam o comportamento de aplicações de *Streaming* de vídeo. Esses bots enviam requisições para servidores instalados em diversos lugares e registram os dados e as dinâmicas observadas em cada sessão de acesso.

A Figura 5 mostra como os clientes do projeto estão distribuídos pelo planeta. O Neubot chegou a realizar 10.000 testes por dia envolvendo mais de 1.000 endereços de Internet Protocol

¹ <https://www.measurementlab.net/tests/neubot/>



Figura 5: Distribuição geográfica de clientes (Neubot). Fonte: Retirado de [47].

(IP) em cerca de 100 países e 1.000 sistemas autônomos. Esses sistemas são grupos de redes IP que estão sob o controle de uma gerência técnica e utilizam uma mesma política de roteamento.

Os bots de *Streaming* de vídeo estão programados para realizar dois testes, diariamente. Nesses testes, um serviço é executado em segundo plano e a cada 30 minutos realiza-se uma análise, que é dependente, de maneira direta, do padrão de conexão dos voluntários envolvidos. Os voluntários que permanecem mais tempo conectados possibilitam, dessa forma, um monitoramento maior de suas infraestruturas. Então, as informações associadas a cada sessão são enviadas para um repositório público.

As instâncias do bot de *Streaming* de vídeo, em cada sessão de acesso, geram 15 requisições para o servidor. Cada segmento requisitado contém 2 segundos de vídeo, disponibilizado nas seguintes taxas de bits: 100, 150, 200, 250, 300, 400, 500, 700, 900, 1.200, 1.500, 2.000, 2.500, 3.000, 4.000, 5.000, 6.000, 7.000, 10.000, 20.000 kbps. Dessa forma, uma sessão de vídeo dura em torno de 30 segundos. O algoritmo de adaptação da taxa de bit, implementado pelo bot de *Streaming* de vídeo, utiliza a vazão medida do último segmento requisitado para definir a taxa de bits do próximo segmento. O primeiro segmento é acessado sempre na menor taxa de bits (100 kbps).

As requisições são realizadas usando uma conexão HTTP persistente, com o servidor configurado para receber um número ilimitado de requisições, por um período de tempo indeterminado. Cada requisição de teste é descrita com propriedades que caracterizam o cliente Neubot, o servidor, a conexão e cada um dos segmentos acessados. Após cada teste, o cliente envia os resultados, em formato JSON (*JavaScript Object Notation*), ao servidor que os armazena e disponibiliza ao público. Aos dados coletados em cada requisição, acrescentou-se as seguintes informações: localização geográfica e propriedade de rede, especificamente o nome do país, cidade e Sistema Autônomo (AS). Essas informações foram obtidas usando serviços disponíveis publicamente². A próxima seção apresenta a análise realizada no conjunto de dados.

² <https://www.maxmind.com/en/home>

3.2 Caracterizando o conjunto de dados

Esta seção apresenta uma análise do conjunto de dados em relação à quantidade de atributos (*features*) e número de medições nas principais cidades do Brasil para efeitos de amostragem. Inicialmente, apresenta-se o padrão de popularidade dos bots nas regiões metropolitanas brasileiras e os ISPs mais monitorados pelo Neubot. Depois, apresenta-se os padrões de frequência de acesso dessas medições e a duração das sessões de vídeo. Por fim, apresenta-se os atributos utilizados para treinar os modelos de redes neurais recorrentes.

A Tabela 1 mostra o resumo de dados dos ISPs da Claro Telecom S.A., da BR (Claro), a Telefônica Brasil S.A. (Telefônica) e a Telemar Norte Leste S.A., da BR (Telemar), que foram os provedores mais monitorados nas três maiores regiões metropolitanas do Brasil, em termos de número de sessões e duração.

Cidade	ISP	2015-2016			2017-2018		
		Sessões (k)	Dur. (hrs)	Bots (n)	Sessões (K)	Dur. (hrs)	Bots (n)
São Paulo	Telefônica	18.9-15.2	114-92	66-63	13.3-15.4	74-89	67-267
	Claro	4.5-5.2	18-29	18-20	6.7-12.0	37-71	69-109
Rio de Janeiro	Claro	3.9-6.6	23-39	12-9	10.0-9.3	60-56	27-60
	Telemar	1.2-1.9	08-12	12-11	3.6-2.9	26-19	19-30
Belo Horizonte	Claro	4.2-5.2	25-28	9-11	6.7-7.3	40-40	19-79
	Telefônica	1.2-1.7	08-11	10-12	3.2-6.1	20-39	17-25

Tabela 1: Resumo de dados coletados por região e provedor.

Diante disso, constata-se que existe uma combinação de um grande número de voluntários e a frequência da coleta podem fornecer um retrato mais preciso dos serviços de rede fornecidos por um ISP. Para verificar tal suposição, os mapas de calor foram construídos com base na frequência das sessões coletadas em períodos da semana, conforme mostra a Figura 6. Delimitado por esse período de tempo, as medições foram agrupadas usando a hora do dia e os segundos em que as primeiras solicitações ocorreram e foram enviadas para análise. O calor expressa um padrão do número de sessões para cada uma hora/segundo. Os dados foram coletados dos provedores da Telefônica Brasil e Claro Telecom, com maior número de medições concentrado nas cidades de São Paulo, Rio de Janeiro e Belo Horizonte.

Diante desses dados, pode-se notar a existência de padrões de acesso em intervalos de tempo distintos: de 1 hora às 6 horas, e de 10 horas às 22 horas. Nesse último, existem para a duração e frequência das sessões exploradas, os maiores valores registrados sobre os dados coletados. A ação de exploração dos bots está fortemente relacionada ao tempo de conexão do dispositivo. As sessões de exploração são previamente agendadas [7], e os dados sugerem uma tendência de investigar infraestruturas durante períodos de sobrecarga, ou seja, quando os usuários estão envolvidos em atividades profissionais, à tarde e ao anoitecer, e/ou estão envolvidos com entretenimento com o avançar da noite.

À medida que o número de bots ativos aumenta, o padrão observado tende a ser mais

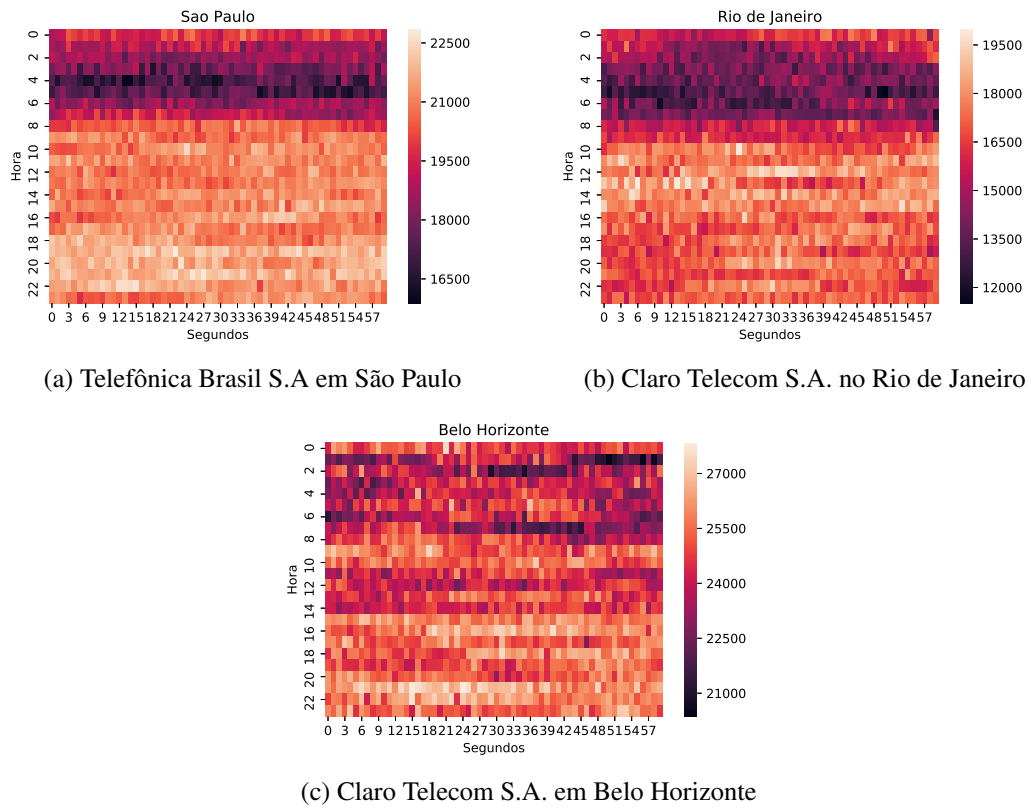


Figura 6: Medições de sessões. Fonte: Autoria própria.

correlacionado ao número de dispositivos conectados, como sugerido pelos dados coletados em São Paulo. No intervalo de 2 horas às 5 horas, o número de sessões coletadas diminuiu. Isso levanta uma incerteza em relação à duração da sessão, uma vez que uma infraestrutura de rede super dimensionada terá sessões com baixa latência e alta vazão.

Outra característica encontrada no conjunto de dados é a esparsidade das medições. A esparsidade do conjunto de dados foi calculada usando as informações das medições na forma dia-hora. Para realizar esse estudo, os dados foram filtrados usando sua marca de tempo e a taxa de *download*. As sessões de medição foram executadas em planos de 2,0 Mbps, realizadas durante os dias úteis e finais de semana. Além disso, a esparsidade foi calculada considerando os seguintes períodos: 24 horas e horário nobre (das 10 horas às 22 horas), conforme a Tabela 2.

Tabela 2: Avaliação de Esparsidade nos Dados de Planos de 2,0 Mbps

Cidade	ISP	Esparsidade			
		24 horas		Horário nobre	
		Sab-Dom	Seg-Sex	Sab-Dom	Seg-Sex
São Paulo	Telefônica	0.713478	0.702738	0.689947	0.678363
Belo Horizonte	Claro	0.882066	0.885063	0.862254	0.866709
Rio de Janeiro	Claro	0.88941	0.884299	0.873707	0.884299

A esparsidade tem os valores mais baixos durante o horário nobre, das 10 horas às 22 horas. O modelo oportunista de bots vincula sua ativação ao horário em que a máquina hospedeira está conectada. Embora o número de medições seja limitado ao longo do dia, os dados sugerem que os bots realizam simples coletas de forma independente. Essa independência de ações força o número de bots online a ser elevado para que se possa ter um retrato do serviço oferecido pelos ISPs. A partir dos dados coletados, pode-se ver que o dilema do dia *versus* hora diminui ao longo do tempo, o que sugere que a hora da coleta é mais importante do que o dia.

Analisou-se também o conjunto de dados no período do início, meio e fim do ano. Nessa análise, considerou-se a frequência das sessões de acesso, vazão máxima de cada sessão de acesso e a frequência dos diferentes planos (1,0 Mbps, 2,0 Mbps, 4,0 Mbps, 8,0 Mbps e acima de 8,0 Mbps). Assim, foi possível verificar um padrão de flutuação das vazões entre os meses do ano. A vazão observada nas medições dos meses iniciais do ano, de março até julho, tende a se repetir e apresenta uma tendência de aumento da vazão média nos meses seguintes, ou seja, de agosto até novembro. Além disso, verificou-se que a maioria das medições concentra-se nos planos de até 2,0 Mbps. Somando-se os demais planos identificados não se chega a 50% das medições dos planos de até 2,0 Mbps.

Os dados coletados que compõe o conjunto é bastante representativo, composto de milhares de sessões com diferentes perfis de rede. A análise de dados contou com um processo de limpeza e seleção de atributos relevantes, pois, de algum modo, a coleta de dados pode apresentar incompletudes, imprecisão e inconsistências. Diante disso, os dados do conjunto foram tratados, considerando-se a remoção de medições e atributos irrelevantes, bem como a atribuição de valores padrões. A Tabela 3 apresenta os dados e informações das medições que integram o conjunto de dados inicial.

Tabela 3: Informações dos dados coletados durante uma sessão de acesso dos bots de fluxo contínuo.

Atributo	Exemplo	Descrição
uuid	87c750ff-e5b2-4b40	ID do usuário.
real_address	216.158.92.121	IP do cliente Neubot, visto pelo servidor.
platform	linux2	O sistema operacional: “linux2”, “win32”.
internal_address	192.168.55.55	Endereço IP do cliente.
remote_address	184.105.23.83	Endereço IP do servidor.
version	0.004016008	Número da versão do Neubot.
timestamp	1496390549	Hora em que o download do segmento foi iniciado, em Unix.
connect_time	0.02710292226402089	Latência ou RTT (do inglês, <i>Round Trip Time</i>).
iteration	1	Número de sequência do segmento solicitado.
rate	1500	Taxa de bits do segmento solicitado, em kbits.
elapsed	0.24498810980003327	Duração do download, em segundos.
elapsed_target	2	Duração esperada do download, em segundos.
received	375130	Quantidade de bytes recebidos.
ISP	Claro Telecom S.A	Provedor de acesso
city	São Paulo	Região da medição

Removeu-se sessões que apresentavam medições de acesso incompletas, como por

exemplo, sessões que apresentavam quantidade bastante inferior às 15 medições padrão. Além disso, removeu-se clientes com poucas sessões de medição, durante os quatro anos coletados. Considerou-se atributos irrelevantes aqueles que não apresentavam informações explicativas das condições da conexão entre cliente e servidor, por exemplo *platform* e *version*. Removeu-se também os atributos com valores alfanumérico e dados categóricos, por exemplo, *uuid*, *real_address*, *city*, *ISP*. O atributo *elapsed_target* é uma constante por isso foi também removido. Essas remoções foram necessárias em razão da construção do modelo de predição, que deve ser genérica. A ideia é que modelo realize predição independente do tipo sessão e região onde foi gerada.

3.3 A Engenharia de atributos

O conjunto de dados utilizado neste trabalho possui uma quantidade expressiva de registros com informações representativas oriundas das conexões TCP de sessões de vídeo de curta duração obtidas no lado do usuário final. O estudo e análise desse conjunto de dados possibilitaram a criação de novos atributos. Dados brutos relacionados à medição da taxa de transferência do canal e ao tempo de medição da sessão foram designados para a expansão do conjunto. Depois, realizou-se outro processo com a finalidade de selecionar aqueles atributos que apresentavam algum tipo correlação e mutualidade com a vazão do canal.

A criação de novos atributos foi realizada a partir de atributos existentes com ideia de expandir o conjunto de dados e promover a transformação de dados brutos em atributos úteis para o aprendizado de máquina. Automaticamente, o conjunto de dados tornou-se mais representativo ao passo que novos atributos foram identificados.

O conjunto de dados original não tinha a estimativa de taxa de vazão do canal, mas esse atributo poderia ser derivado a partir dos atributos de taxa de download do i -ésimo segmento de vídeo da sessão. Denominado de (*downthrpt*), esse atributo foi definido pela relação da quantidade de bits do i -ésimo segmento de vídeo - representado pelo atributo *received*, e pelo seu tempo de download necessário - representado pelo atributo *elapsed*. Esse atributo representa a variação da vazão medida em cada segmento.

Com o conhecimento da vazão dos segmentos que compõe cada sessão, considerou-se a possibilidade de derivar outras formas de estimativa de vazão, incluindo a *Last Segment Throughput* (LST), que é vazão medida do último segmento, a *Session Average Throughput* (SAT), na qual mede a vazão média da sessão de vídeo inteira e a *Moving Window Average Throughput* (WAT), na qual mede a vazão média dos últimos k segmentos.

Além dessas estimativas de vazão, derivou-se a taxa de vazão máxima de cada sessão, a *max_download_rate*; e um atributo de conexão persistente da janela do TCP, o *tcp_mean_wind*; derivado a partir do atributo *connect_time*, o qual está relacionado ao tempo de ida e volta da solicitação de um segmento (RTT) e da vazão instantânea. O atributo em questão é originado da seguinte equação:

$$W = RTT * T$$

onde T é a vazão observada na requisição, RTT , conforme definido anteriormente, indica o tempo gasto para estabelecer a conexão *tcp_mean_wind*.

Um último atributo relacionado ao tempo de download de cada requisição foi originado de *elapsed*. Nomeado de *jitter*, esse atributo é gerado a partir do tempo de download, em segundos, de dois segmentos consecutivos, $i - 1$ e i , ou seja; $jitter = (elapsed_i - elapsed_{i-1})$.

Os seguintes atributos relacionados a tempo foram criados: *weekday* (dia da semana), *day* (dia do mês) *hour* (hora), *minute* (minuto) e *second* (segundos). Esses atributos foram extraídos de um atributo original chamado *timestamp*, que é uma época UNIX. O primeiro indica o dia da semana que ocorreu a medição, o terceiro, o quarto e quinto representam, respectivamente, a hora, minuto e o segundo em que a sessão de medição foi realizada.

A Tabela 4 lista os atributos que compõem o conjunto de dados final. Atribuiu-se "CAT" aos atributos categóricos e "DISC" a aqueles que se caracterizam como discretos. Aqueles destacados na cor marrom representam atributos derivados manualmente do timestamp com informações de hora e data. Os atributos em destaque na cor cinza designam aqueles derivados manualmente da dinâmica da largura de banda (bandwidth) do canal. Os atributos restantes são originários do conjunto, provenientes da dinâmica da vazão.

Tabela 4: Atributos do conjunto de dados após a limpeza e seleção de atributos.

Atributo	Exemplo	Descrição	Tipo
year	2018	Um valor inteiro entre 2015 e 2018	CAT
month	12	Um valor inteiro entre 1 e 12	CAT
weekday	6	Um valor inteiro entre 0 e 6	CAT
day	31	Um valor inteiro entre 1 e 31	CAT
hour	23	Um valor inteiro entre 0 e 23	CAT
minute	47	Um valor inteiro entre 0 e 60	CAT
second	20	Um valor inteiro entre 0 e 60	CAT
max_download_rate	1.49004	Taxa de download máxima da sessão	DISC
delay	0.156004	Variação do atraso entre segmentos consecutivos	DISC
tcp_mean_wind	0.116743	Dinâmica da janela do TCP	DISC
LST	0.726591	Vazão do último segmento baixado	DISC
SAT	0.598825	Vazão média da sessão inteira	DISC
WAT	0.568129	Vazão média dos últimos k segmentos	DISC
downthrpt	0.35030	Vazão do segmento em Mbit/s	DISC
connect_time	0.02710292226402089	Latência ou RTT (<i>Round Trip Time</i>)	DISC
iteration	1	Número de sequência do segmento solicitado	CAT
rate	1500	Taxa de bits do segmento solicitado, em kbits	DISC
elapsed	0.24498810980003327	Duração do download, em segundos	DISC
received	375130	Quantidade de bytes recebida do servidor para a solicitação atual, medida em bytes	DISC
delta_user_time	0.060000000000000005	Tempo acumulado do usuário durante uma solicitação, medido em segundos	DISC
delta_sys_time	0.060000000000000009	Tempo acumulado do sistema durante uma solicitação, medido em segundos	DISC
request_ticks	1382434858.103292	Hora em que a solicitação foi executada no servidor (hora Unix)	DISC

Após a limpeza, análise e transformação dos dados, o conjunto de dados final totalizou 803.221 sessões de vídeo. Sendo assim, cada sessão de vídeo totalizou um subconjunto padrão de 15 requisições, tal que cada requisição representou um segmento de vídeo baixado com duração de 2 segundos. Por último, converteu-se o conjunto numa matriz de dados bidimensional com 19 atributos (colunas) e 12.048.315 registros (linhas) disposta num arquivo em formato de texto. Esse processo possibilita a redução do tamanho do arquivo e facilita a transformação dos atributos em variáveis de *lag* para a implementação da entrada do modelo preditivo.

A seções seguintes apresentam duas abordagens para a seleção de atributos do conjunto de dados. A finalidade desse pré-processamento consiste em filtrar um subconjunto significativo de atributos, ao reduzir possíveis informações irrelevantes e redundantes, de tal maneira que ele possa contribuir, positivamente, no processo de aprendizado do modelo de rede neural.

3.3.1 Seleção de atributos

Esta seção enfatiza duas abordagens de seleção de atributos: Informação Mútua e Correlação de Pearson. Elas contribuíram significativamente no pré-processamento dos dados e para o processo de aprendizado do modelo. Justifica-se essas abordagens, pois, independentemente do algoritmo de aprendizagem de máquina a ser utilizado, os atributos influenciam diretamente no desempenho do modelo preditivo [13].

3.3.1.1 Correlação de Pearson

O conceito de correlação compreende o grau de análise da associação numérica entre duas variáveis. Precisamente, a correlação numérica entre as observações de duas variáveis X e R . A variável X representa um conjunto de exemplos, denotado por $X = \{x_1, x_2, x_3, \dots, x_N\}$ e R representa um conjunto de exemplos formado por variáveis contínuas, ou seja, a variável-alvo, neste caso, a vazão (*downthrpt*), tal que, $R = \{r_1, r_2, r_3, \dots, r_N\}$. A ideia é obter a correlação de cada par $(x_1, r_1), (x_2, r_2), \dots, (x_N, r_N)$. Para isso, utilizou-se o coeficiente de correlação linear de Pearson.

O coeficiente de correlação linear de Pearson (k) é um grau de correlação entre duas variáveis quantitativas observadas. Ao verificar a correlação em um conjunto de dados, o valor do coeficiente k estará no intervalo de -1 a 1. Os dados observados apresentam correlação forte quando k tende a ser tão próximo de 1 (ou -1). Quando $k = +1$ significa que existe uma correlação linear positiva. Por outro lado, quando $k = -1$, apresenta-se um correlação linear negativa. Um coeficiente de correlação próximo de zero indica que não existe correlação nos dados. A fórmula a seguir, apresenta o cálculo do coeficiente linear de Pearson:

$$k = \frac{n \sum (x_i \cdot r_i) - (\sum x_i)(\sum r_i)}{\sqrt{n \sum x_i^2 - (\sum x_i)^2} \cdot \sqrt{n \sum r_i^2 - (\sum r_i)^2}} \quad (3.1)$$

onde x_i e r_i denotam valores para as variáveis X e R , respectivamente.

A Figura 7 apresenta a correlação positiva e negativa resultante da aplicação da correlação de Pearson. Na realização dessa aplicação, utilizou-se 67 amostras de requisições de acesso. As features *request_ticks*, *tcp_mean_wind*, *delta_sys_time*, *delta_user_time*, *max_download_rate*, *rate* e *received* possuem uma correlação positiva com *downthrpt*, tal que, *rate* e *received* são as variáveis que mostram-se fortemente correlacionadas, cujo $k > 0,8$. Por outro lado, *delay*, *elapsed* e *connect_time* apresentam-se num subconjunto de correlação negativa diante

de *downthrpt*, tal que, *delay* é a variável que corresponde a uma correlação negativa maior cujo $k > -0,2$.

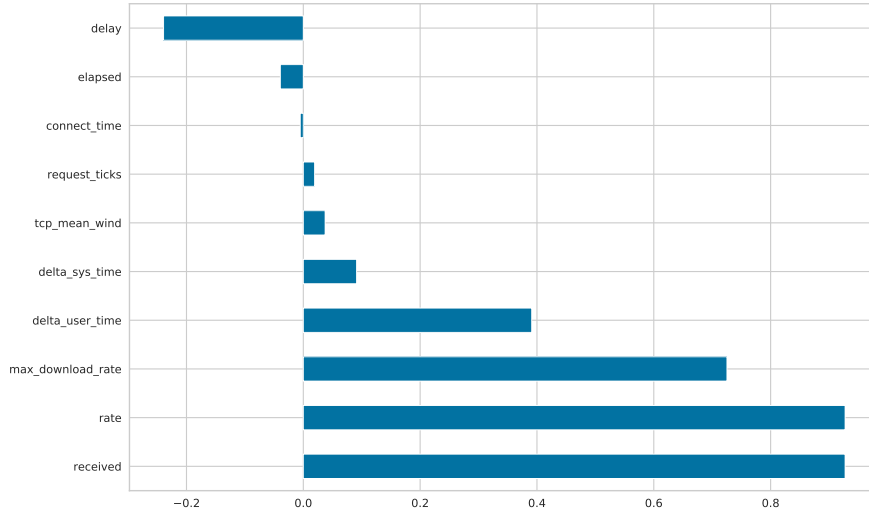


Figura 7: Correlação de Pearson

3.3.1.2 Informação Mútua

Essa abordagem de seleção é pertinente ao campo da Teoria da Informação e Probabilidades. Os métodos baseados em Informação Mútua exploram diferentes critérios de filtros heurísticos para mensurar a importância de variáveis, na estimativa do ganho de informação mútua para uma determinada variável-alvo contínua [39]. Um valor igual a zero entre as variáveis significa que são estatisticamente independentes, enquanto valores acima de zero representa dependência entre elas. A definição formal de Informação Mútua de duas variáveis aleatórias X e R , ao considerar a distribuição conjunta $P(X, R)$, é definida de acordo com a seguinte fórmula:

$$IM(X; R) = \sum_{x \in X} \sum_{r \in Y} P(x, y) \log \frac{P(x, r)}{P(x)P(r)} \quad (3.2)$$

onde $P(X)$ e $P(R)$ são as distribuições marginais de X e R . A Figura 8 apresenta o ranqueamento das variáveis em ordem crescente de escore. Utilizou-se também 67 medições de acesso nesse procedimento. Como pode ser visto, as *features year, day, weekday e month* não apresentam nenhum tipo de informação com *downthrpt*. A característica *second* (segundos) é a primeira característica com grau de informação menor e *tcp_mean_wind* corresponde ao maior grau de informação mútua em relação as demais. Observa-se que as características relacionadas com a vazão, ou seja, a taxa de transferência de dados, como *rate, elapsed* possuem um grau de informação muito maior do que características ligadas a tempo, por exemplo, *hour*.

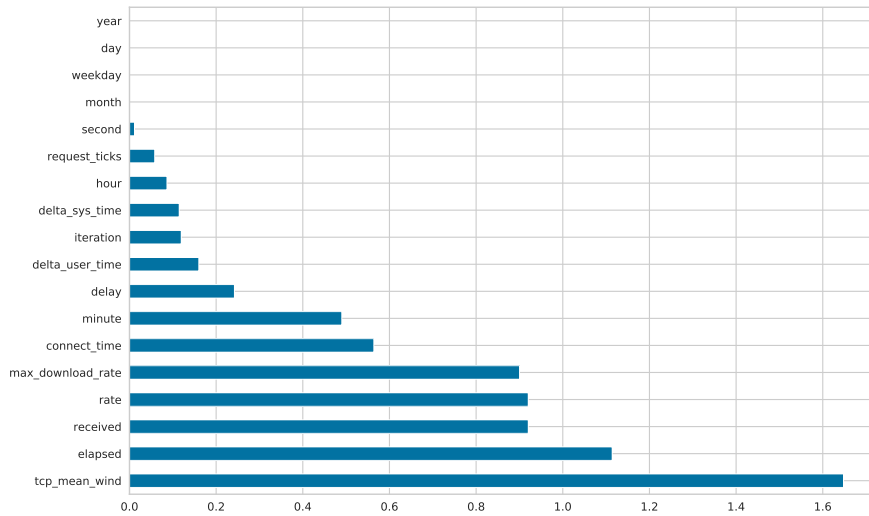


Figura 8: Informação Mútua

Em razão da correlação de Pearson atuar apenas com variáveis contínuas e o resultado de sua aplicação no conjunto de dados não ser significativo em termos de quantidade de características, no desenvolvimento do modelo de rede neural recorrente profunda, treinou-se-a com um subconjunto de características que possuíam um nível de representatividade mínima em termos de informação mútua, desprezando aquelas as quais não possuíam nenhum grau de informação com *downthrpt*. A princípio, as características estabelecidas, de acordo com o estudo de seleção de *features* para o processo de treinamento do modelo foram: *second*, *request_ticks*, *hour*, *delta_sys_time*, *iteration*, *delta_user_time*, *delay*, *minute*, *connec_time*, *max_download_rate*, *rate*, *received*, *elapsed* e *tcp_mean_wind*.

No entanto, na condução do desenvolvimento e testes do modelo, com essas características, obteve-se uma elevada taxa de overfitting e underfitting ao validar o modelo. Diante disso, realizou-se mais experimentos e testes para a melhoria de desempenho do modelo proposto. Por fim, após um extenso teste empírico, as características *delta_sys_time*, *delta_user_time*, *request_ticks*, *connec_time*, *rate*, *received*, *elapsed*, *max_download_rate*, *delay*, *tcp_mean_wind*, LST, SAT, WAT e *downthrpt*, viabilizaram melhorias significativas em termos de generalização e performance para o modelo proposto neste trabalho. O desempenho do modelo com as últimas características fornecidas serão apresentadas no próximo capítulo.

3.4 Considerações finais

Neste capítulo, apresentou-se uma análise exploratória do conjunto de dados utilizado no estudo em questão. Em razão da forma como os dados foram coletados, fez-se necessário a avaliação qualitativa e quantitativa dos dados e posterior tratamento de tais. O conjunto de dados

contém uma vasta quantidade de medições de conexões de *Streaming* de vídeo capturados no lado do cliente. São traces de tráfego de vídeo compostos de seções de vídeo com diferentes perfis de acesso e componentes de condições de rede, onde a vazão representa o reflexo do tipo de conexão medida em determinado contexto de acesso. O capítulo finaliza com a descrição do estudo de seleção de atributos do conjunto de dados, com o objetivo de encontrar, por meio de métodos tradicionais de seleção, os atributos mais importantes, como um passo inicial para processá-los através de uma rede neural recorrente profunda.

O próximo capítulo, apresenta o processo de construção do modelo de rede neural recorrente para definir um modelo que realize a previsão da vazão futura de conexões estabelecidas para transferir uma sessão de *Streaming* de vídeo, em que o conjunto de dados explorado, é utilizado no processo de aprendizagem do modelo proposto. Explicam-se os detalhes do processo de aprendizagem do modelo, tal qual, a forma como os dados são inseridos na entrada e o papel das regras aplicadas em sua estrutura por meio de vários processamentos.

CAPÍTULO 4

O modelo GSTNet

Este capítulo apresenta o modelo de rede neural recorrente proposto que foi desenvolvido com a finalidade de prever a *Throughput* futura com base em dados históricos de acesso de medições passadas. O modelo, denominado de GSTNet (um acrônimo para GRU *Short-Term Network*), processa dados dispostos na forma de série temporal, considerando as variáveis do conjunto de dados analisadas no capítulo anterior. O desenvolvimento do modelo preditor envolve desde o papel do pré-processamento das features selecionadas até a formação da estrutura de aprendizado interna das camadas intermediárias e as regras de saída. Além disso, apresenta-se também as métricas de erro identificadas para verificar o quão bem ou mal, em termos de desempenho, o modelo consegue realizar a tarefa para o qual foi treinado.

4.1 O problema de predição

Neste estudo, aborda-se o problema de predição da *Throughput* como uma série de vazões de canais de transmissão de *Streaming* de vídeo, cuja a análise do erro de predição é realizada em função das observações passadas das séries temporais, por meio do estabelecimento de janelas de tempo no ajuste do modelo.

Seja Z uma série temporal multivariada definida por $Z = \{z_1, z_2, \dots, z_T\}$, tal que, $z_t \in \mathbb{R}^n$, um modelo de predição para Z tem como objetivo a predição da variável dependente, i.e., a *Throughput* (r), no instante de tempo z_{T+k} , a partir de um conjunto de m pares de exemplos $(z_1, r_1), (z_2, r_2), \dots, (z_m, r_m) \subseteq Z \times R$ de treinamento de entrada e saída, considerando uma janela ajustável de tamanho q e o horizonte de predição definido como k , sendo este último o intervalo de tempo que separa a última observação realizada no ajuste do modelo e o valor futuro a ser previsto, ou seja, o número de instantes de tempo à frente para o qual é realizada a predição. Nesse sentido, a predição da *Throughput* no instante de tempo z_{T+k} é denotada por $\hat{z}_{T+k}$.

Durante o estudo, ajustou-se a entrada do modelo com a janela q igual a 20 segundos, isto é, o histórico, e o horizonte de predição k 10, igual a segundos. Seguindo essa configuração, o modelo GSNet foi designado para realizar a predição dos primeiros 10 segundos de vídeo, considerando a *Throughput* da sessão em andamento, quando os primeiros segmentos de vídeos já foram baixados.

4.2 A arquitetura do modelo

A presente seção apresenta o pré-processamento das features de entrada, bem como a normalização de tais, no projeto de arquitetura de rede neural do modelo proposto, e a definição do papel de todos os componentes que integram sua estrutura do ponto de vista de aprendizagem de máquina.

4.2.1 A normalização das features

As features para o modelo, descritas no Capítulo 3, tais quais aquelas do subconjunto selecionado, são decisivas para o desempenho de aprendizagem do modelo, uma vez que, podem contribuir para a sua capacidade de generalização, ao ponto de encontrar um possível alcance da *Throughput* esperada; quando são também submetidas ao processo de normalização, em que garante-se que estas tenham as mesmas ordem de grandeza, ao ajustá-las para um intervalo numérico comum, inclusive pela razão do modelo ser baseado em gradiente descendente.

O processo de normalização foi conduzido por meio de experimentação empírica para encontrar a normalização adequada às features fornecidas. Nesse sentido, padronizou-se os dados para uma distribuição normal. Para esse fim, considerou-se a média e o desvio padrão individual da distribuição de cada série do conjunto de dados. Essa abordagem de normalização, chamada de z-score, foi utilizada para assegurar que a distribuição dos dados pertençam, aproximadamente, a uma média igual 0 e desvio padrão igual 1, conforme a equação a seguir.

$$z_i = (z - \mu) / \sigma \quad (4.1)$$

onde μ é a média e σ representa o desvio padrão. O elemento z_i , por sua vez, representa a i^{th} feature da matriz multidimensional M com $i \times j$ representações de dados de séries temporais.

Com as features da matriz M normalizadas para a uma escala comum, a entrada do modelo de rede neural pode, portanto, ser alimentada e inicializar o fluxo de aprendizagem, ao longo de suas camadas, no treinamento. Constatou-se que, ao aderir essa normalização, GSTNet obteve um melhor ajuste no treinamento, no aspecto de convergência e estabilidade, como será visto nos resultados numéricos do Capítulo 5.

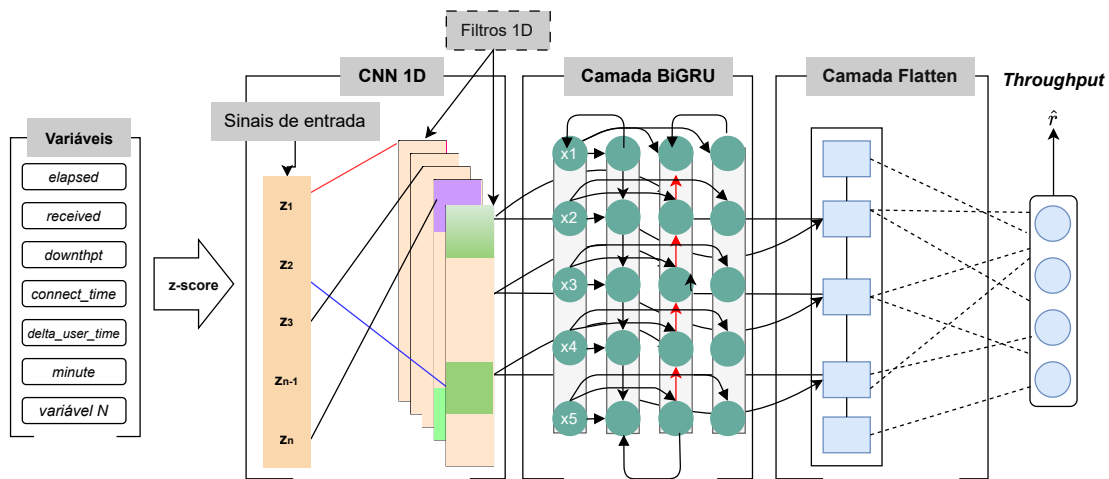
4.2.2 As camadas de GSNet

O modelo GSTNet é uma rede neural recorrente profunda integrada por várias camadas que executam regras específicas para a abstração do modelo de predição, cada uma. As camadas, com seus parâmetros e pesos foram esboçados, conforme o problema de predição em questão, considerando-se a natureza sequencial e temporal da disposição dos dados.

Pela disposição dos dados, na forma de séries temporais, definiu-se a composição da rede neural do modelo, de tal modo que, formou-se um empilhamento de camadas, originando-se uma rede neural profunda. Ela é formada por: i) CNN 1D, uma rede convolutiva que desempenha o papel de detectar padrões temporais; ii) uma camada de *Batch normalization* [25], cuja função é minimizar o tempo de treinamento; iii) uma camada de *Dropout* [42], que possui o papel de eliminar neurônios redundantes; iv) uma GRU bidirecional (BiGRU), a qual captura informações históricas; v) a função de ativação SELU, na forma de camada; vi) uma camada *Flatten* que unifica todos os sinais e, por fim, vii) duas camadas densas.

CNN 1D, BiGRU e *Flatten* formam os pilares da estrutura do modelo de rede neural. Elas são responsáveis pela essência do processo de aprendizagem. Já as camadas *Batch normalization*, *Dropout* e função de ativação são camadas intermediárias à rede. Essas camadas, com exceção da camada de ativação e *Flatten*, desempenham o papel de regularizar a saída das camadas. Conjuntamente com seus parâmetros e pesos, contribuem para que o modelo se torne uma rede neural totalmente hibridizada. A Figura 9 apresenta a arquitetura do modelo proposto com as principais camadas.

Figura 9: Arquitetura do modelo de rede neural recorrente proposto



Fonte: Autoria própria.

4.2.2.1 A camada CNN 1D

A camada de CNN 1D é uma variante de CNN a qual integra a primeira camada de rede do modelo. A principal razão pela sua utilização na modelagem dos dados de série temporal,

nesse estudo, consiste na sua típica tarefa de extração de informação temporal, quando os filtros de convolução unidimensionais são aplicados aos dados de entrada. O resultado da filtragem é detectar características em vetores unidimensionais, sequencialmente no tempo.

Os kernels, descritos na seção 2.2.9 do Capítulo 2, são filtros 1D. Eles representam matrizes de menor ordem (menor tamanho) com o finalidade de remover aqueles sinais providos de ruídos, extraindo somente os que possuem relação e características temporais entre as variáveis das séries temporais.

O tamanho do kernel, denotado por ϑ , representa o deslocamento da janela de convolução ao longo do vetor de entrada unidimensional $z[n]$, tal que, $n = 0, 1, \dots, N - 1$. O número e tamanho do kernel determinará a capacidade de extrair os padrões relevantes, resultando nos mapas de características.

Os mapas de características são a consequência da operação dos kernels, cuja saída de convolução $\zeta[n]$, é calculada da seguinte forma:

$$\zeta[n] = k[n] \circledast z[n] = \sum_{m=0}^{n-1} k[m].z[n-m], n = 0, 1, \dots, N - 1 \quad (4.2)$$

onde $\circledast$ representa a operação de convolução com o kernel denotado como $k[n]$. Com o kernel, as características filtradas pela operação de convolução são processadas pela função de ativação não-linear SELU e enviadas para a saída da camada. Essa função atua como um parâmetro da camada de convolução, que foi descrita na seção 2.2.5 do Capítulo 2. Denotada por ϕ , a SELU realiza a operação da seguinte forma:

$$h_i = \phi(W_i.b_i + z_i.k_i) \quad (4.3)$$

onde h_i representa o vetor de saída com a i^{th} feature filtrada, b_i denota o vetor de viés (bias) e W_i representa a matriz de pesos de uma determinada convolução.

Além da função de ativação SELU contribuir na aprendizagem de relações complexas e contínuas entre as variáveis pela rede, destaca-se também o impacto do tipo de inicialização da matriz de pesos da entrada utilizado. Os pesos da camada de convolução foram inicializados com uma distribuição normal. Isso significa que os valores de pesos e vieses foram gerados, aleatoriamente, com a média próxima de 0.0 e desvio padrão próximo de 1.0.

Ao final da realização do processamento de detecção de informação temporal pela camada de convolução, junto com seus parâmetros, a saída produzida foram vetores com filtros daquelas informações mais relevantes. Em seguida, essas informações são enviadas para as camadas intermediárias, que são *Batch normalization* e *Dropout*.

Dropout e *Batch normalization* são técnicas que podem ser adicionadas à estrutura da rede neural, em forma de camadas. A camada de *Dropout* foi utilizada com a finalidade de evitar *overfitting*, durante o treinamento. Como descrito no Capítulo 2, essa técnica exclui uma

parcela de neurônios da camada, propiciando menos complexidade. Após a camada de *Dropout*, adicionou-se *Batch normalization* como camada. Essa camada desempenha o papel de acelerar e estabilizar o treinamento do modelo. Com isso, reduziu-se o tempo de treinamento em cerca de 50%, possibilitando um treinamento em menos de 12 horas.

4.2.2.2 A camada BiGRU

A segunda camada integrante de GSNet é uma GRU Bidirecional, uma variante de rede neural recorrente com o acrônimo BiGRU. Uma justificativa para a utilização dessa rede, na modelagem dos dados, consiste no fato de que, a GRU requisita menos parâmetros, apresenta baixo custo de *overhead* computacional no treinamento e maior velocidade [45], além de ser adequada para a predição de curto prazo. BiGRU também apresentou um melhor ajuste, quanto ao tipo de dado apresentado. Todas essas condições e constatações foram determinantes para a sua integração à arquitetura do modelo.

A rede BiGRU caracteriza-se por levar em consideração a relação entre a saída anterior, a atual e as saídas subsequentes, ao processar a sequência temporal dos dados em duas formas. Essa rede é configurada para coletar informações dos estados nas sequências de ida (*forward*) e volta (*backward*), em vários pontos no tempo, unindo duas camadas GRU, além de encapsular informação de associação e sequência nos dados, ao combinar passado e futuro. Esse processo é expressado nas Equações (4.4)-(4.6) e a rede é ilustrada na Figura 9, junto com seus parâmetros.

$$\vec{h} = GRU_f(x_t, \vec{h}_{t-1}) \quad (4.4)$$

$$\overleftarrow{h} = GRU_b(x_t, \overleftarrow{h}_{t-1}) \quad (4.5)$$

$$h_t = \vec{h}_t \oplus \overleftarrow{h}_t \quad (4.6)$$

onde x_t representa a entrada do momento atual, $\vec{h}_t$ e $\overleftarrow{h}_t$ denotam os passos de ida e volta das redes GRU_f e GRU_b , no tempo t , respectivamente. A operação $\oplus$ combina, por sua vez, todas as informações de estado capturadas pelas duas camadas.

A rede BiGRU adota em sua estrutura interna os mesmos parâmetros de rede definidos para a camada de convolução, como a função de ativação não-linear SELU, expressa na Equação 4.3, e inicialização de pesos, conforme uma distribuição normal. Aqueles sinais resultantes desse processamento e aprendizado são transferidos para a próxima camada da rede, a camada *Flatten*.

4.2.2.3 A camada Flatten

A camada *Flatten*, tipicamente utilizada como parâmetro de rede CNN 1D, é aproveitada na arquitetura do modelo, isoladamente, como um processamento adicional para achatamento

dos sinais processados pelas camadas anteriores, transformando-as em um enorme vetor unidimensional. Essa camada faz parte de uma experimentação empírica, que possibilitou melhoria no desempenho da rede.

O vetor com os sinais de saída dessa camada são enviados para a entrada da camada densa, que os conecta com suas unidades de neurônios. Essa é a penúltima camada de rede a processar os sinais transmitidos.

4.2.2.4 A camada Densa

A camada densa é constituída por 128 unidades de neurônios. Essa camada possui a finalidade de conectar todos os sinais processados pelas camadas anteriores a todos seus neurônios. Como argumentos ou parâmetros, essa camada apresenta por padrão apenas uma função de ativação, que neste caso, é linear. Sua matriz de pesos é inicializada com uma distribuição uniforme [17].

Após o processamento da camada densa, a função de ativação SELU é adicionada antes na saída da rede. O objetivo da SELU, como apresentado em sua definição, é garantir a não linearidade dos dados ao transmitir essa condição para a última camada da rede, que também é uma densa, mas com 10 unidades de neurônios, denotando com isso, a regra de prever os primeiros segundos de vídeo para uma determinada sessão.

4.2.3 Função de perda

Ao realizar o processamento dos sinais, ao longo de suas camadas, a rede neural do modelo precisa conceber as regras contidas nos dados, em sua saída. Para isso, a função de perda (*loss*) e o algoritmo de otimização influenciam diretamente aquelas condições, durante o treinamento.

O treinamento do modelo consiste no procedimento de otimização que envolve determinar o conjunto de parâmetros de rede ajustáveis, θ , que minimizam a função de perda estabelecida para o problema de predição, em questão.

A função de perda $J(\theta)$, apresentada na Seção 2.2.7 do Capítulo 2, é uma medida que, por meio do algoritmo otimizador *Stochastic Gradient Descent* (SGD), apresentado no Capítulo 2, determina o quanto a rede aprendeu, durante o treinamento, na tentativa de diminuir o erro de generalização.

O cálculo utilizado na minimização da função de perda é realizado pela escolha de medida de erro apropriada para a predição de dados, que nesse caso, adotou-se o *Mean Absolute Error* (MAE) como medida padrão, conforme a Equação 4.8. Essa medida possibilitou um melhor desempenho da rede em comparação a outras, e.g, *Mean Squared Error* (MSE). A função de perda é calculada da seguinte forma:

$$\underset{\theta}{\text{minimize}} \sum_{t \in \Omega_{Train}} \frac{1}{n} \sum_{i=0}^{n-1} |Y_{t,i} - \hat{Y}_{t-h,i}| \quad (4.7)$$

onde θ representa o conjunto de parâmetros do modelo e Ω_{Train} é o conjunto de *time stamps*, i.e, observações de tempo, utilizados durante o treinamento. Dessa forma, deve-se minimizar a função de perda $J(\theta)$, que mede o custo de prever $\hat{Y}$ quando a resposta atual é Y .

4.3 Métricas de erro

As Métricas de erro consistem em verificar a diferença entre o valor previsto e o valor observado, no conjunto de dados validado, pelo modelo treinado. A medida de erro fornece a capacidade de precisão do modelo e, conseqüentemente, a interpretação do seu desempenho no processo de predição.

Na avaliação do modelo, durante o treinamento, validação e teste, utilizou-se métricas consideradas inerentes ao problema de predição investigado, como o MAE, o *Mean Squared Error* (MSE) e o *Root Relative Squared Error* (RSE). Essas métricas tornaram-se adequadas, durante a experimentação, conforme a evolução do desempenho de GSNet. As fórmulas das métricas são descritas, a seguir:

$$MAE = \frac{1}{n} \sum_{i=1}^n |Y_{it} - \hat{Y}_{it}| \quad (4.8)$$

$$MSE = \frac{1}{n} \sum_{i=1}^n |Y_{it} - \hat{Y}_{it}|^2 \quad (4.9)$$

$$RSE = \frac{\sqrt{\sum_{(i,t) \in \Omega_{Test}} (Y_{it} - \hat{Y}_{it})^2}}{\sqrt{\sum_{(i,t) \in \Omega_{Test}} (Y_i - \text{mean}(Z_{it}))^2}} \quad (4.10)$$

onde $\hat{Y}_i$ é o valor previsto da i^{th} amostra e Y_i é o valor original correspondente, no instante de tempo t , considerando-se n observações. A métrica RSE é uma versão do *Root Mean Square Error* (RMSE).

Com as métricas de erro definidas, treinou-se várias redes neurais recorrentes, modelos tradicionais de aprendizagem de máquina, como KNN e RF, e o próprio GSNet. Comparou-se o desempenho dos modelos com as métricas de erro apresentadas, em diferentes contextos de medições de Throughput. Os resultados do estudo numérico são apresentados e discutidos no próximo capítulo.

4.4 Considerações finais do capítulo

Este capítulo apresentou os detalhes envolvidos no processo de construção do modelo proposto de rede neural recorrente profunda e os componentes da sua arquitetura. Apresentou-se o processo de aprendizagem e como o modelo foi adequado à forma de série temporais. Dois outros aspectos foram destacados, o processo de otimização e a função de custo, como impactantes do processo de aprendizagem do modelo. Finalmente, apresentou-se uma função de aproximação que tenta se ajustar aos dados com o objetivo de alcançar um dado alvo, nesse caso, a *Throughput*. O próximo capítulo apresenta a evolução do modelo durante o treinamento e dos modelos derivantes de RNN, além do poder de generalização frente a dados nunca vistos.

CAPÍTULO 5

Resultados numéricos

Os resultados obtidos no estudo investigativo são apresentados neste capítulo, seguidos pela sua respectiva discussão, em resposta aos objetivos levantados, inicialmente. São fornecidos e discutidos resultados do estudo quantitativo realizado para avaliar o desempenho do modelo de predição de *Throughput* desenvolvido neste trabalho, o GSTNet. Apresentam-se os resultados dos estudos numéricos realizados no processo de treinamento e teste, onde uma comparação de desempenho de GSTNet foi realizada, em relação a outros modelos previsores, como aqueles baseados em RNN, modelos tradicionais e média móvel. A análise dos resultados revelou *insights* e respostas valiosos, que serão detalhados nas seções seguintes.

5.1 Protocolo experimental

Nesta seção, apresenta-se o protocolo dos experimentos realizados durante a condução do estudo do problema de previsão de *Throughput*. O protocolo refere-se aos detalhes das atividades de desenvolvimento e evolução do modelo de rede neural proposto. Durante esse processo, foram definidas as configurações do formato dos dados de entrada dos sinais de séries temporais, a divisão dos dados e o ajuste de parâmetros de treinamento.

As séries temporais, bem como as variáveis selecionadas, foram transformadas em *lags* em razão do problema de previsão investigado. As *lags* são observações usadas como insumos para o modelo preditor, configuradas através de uma janela deslizante (do inglês, *sliding window*) de tamanho fixa, de modo que o modelo explorasse o comportamento histórico dos dados, ao passo que previsse a tendência do comportamento futuro. A finalidade da janela é deslizar ao longo da série temporal, passo a passo, para criar pares de entrada e saída usados no treinamento e teste do modelo.

Nesse sentido, definiu-se a janela de entrada de dados do modelo com comprimento de histórico para k segundos e horizonte de predição futura configurado para h segundos de

vídeos da dinâmica da *Throughput*, no qual k foi configurado para 20 segundos, com a ideia de que o modelo fosse capaz de realizar a predição da *Throughput* do canal para os primeiros 10 segundos de vídeo, considerado um histórico daqueles 20 segundos. Com isso, a janela converte o problema de previsão de série temporal, que é sequencial por natureza, em um problema de aprendizado supervisionado tradicional.

Quanto ao desenvolvimento da rede, utilizou-se de técnicas de otimização de treinamento nas configurações de seus parâmetros. Em princípio, a condução do processo de desenvolvimento do modelo seguiu uma experimentação empírica por meio de ajustes de parâmetros, como épocas, filtros técnicas de otimização e regularização, para controlar o processo de treino e encontrar os pesos ideais para a convergência. Os parâmetros de rede foram ajustados como uma etapa de pré-treinamento na finalidade de garantir um melhor desempenho possível.

Dividiu-se o conjunto de dados em três subconjuntos, na ordem cronológica que as séries de tempo apresentavam-se: treinamento, validação e teste. A Figura 10 ilustra a divisão do conjunto de dados nos três subconjuntos. A justificativa para a divisão do conjunto de dados apresentar dois conjuntos de validação, ao invés de apenas um conjunto de teste, deve-se pela configuração do modelo em um espaço de parâmetros e hiperparâmetros de rede, uma vez que, por meio do conjunto de validação é possível fazer ajustes nas camadas e pesos da rede [14].

O processo de treinamento do modelo consistiu de 60% de amostras do conjunto de dados, totalizando 642.576 medições com diferentes perfis de sessões e requisições de acesso, posto que, cada sessão é provida de 15 requisições de segmentos de vídeo com duração de 02 segundos. Já as amostras do conjunto de validação e teste totalizaram 80.322 medições cada um, ou seja, 20% das amostras para cada conjunto. Após avaliado no conjunto de validação, validou-se uma última vez no conjunto de teste.

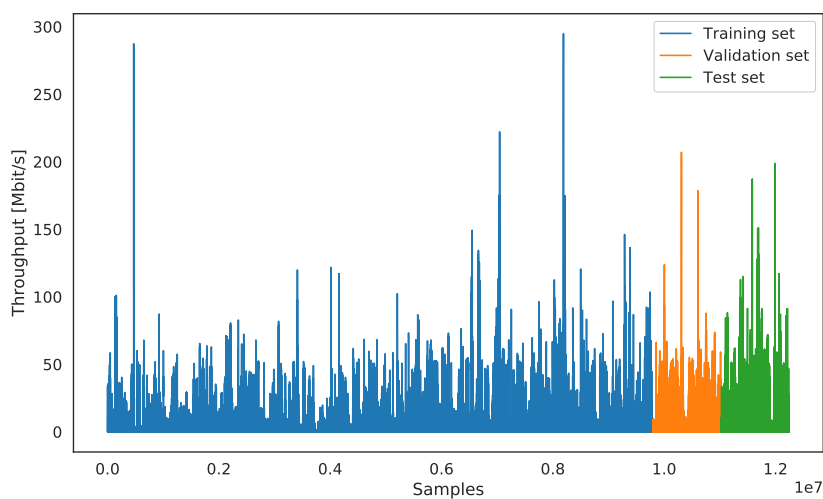


Figura 10: Divisão do conjunto de dados.

5.1.1 Ajuste de parâmetros

O objetivo do ajuste de parâmetros da rede neural de GSTNet, bem como sua otimização, visou promover um melhor desempenho, ao encontrar os pesos ideais que se ajustara no processo de aprendizagem. O ajuste envolve o número de épocas, quantidade de amostras de treinamento (lote), o tipo de algoritmo de otimização, taxa de aprendizagem e quantidade de unidades de camadas para iniciar os ciclos de treinamento. O processo de ajustes foi realizado, manualmente, durante a condução do treinamento. Os detalhes das configurações no processo de aprendizagem do modelo são descritos, a seguir.

Nos ciclos de treinamento do modelo foram concebidos com um limite máximo de 150 épocas. A descoberta desse limite de iterações foi possível por meio de sucessivas experimentações empíricas, as quais consistiram-se de ajuste de parâmetros e utilização de técnicas de otimização, como *Early stopping*, *Batch Normalization* e *Dropout*.

A utilização de *Early stopping* no treinamento justifica-se em razão da redução da probabilidade de *overfitting* e, conseqüentemente, o erro de generalização. O objetivo de se instaurar essa técnica consiste em interromper o treinamento quando uma dada métrica monitorada parou de melhorar antes do limite de épocas estabelecido, considerando que o objetivo do treinamento é reduzir os valores da função de perda (*loss*). Nesse caso, monitorou-se a função de perda na validação com um número de épocas igual a 10. Durante o treino, uma parte dos dados era utilizada para validação, i.e., a cada época de treinamento, a rede neural validava a saída prevista. Ao completar, sequencialmente, um número de épocas igual a 10 sem melhoria na redução do erro de generalização, a rede paralisava o treino.

Outra técnica instaurada para controlar o processo de treinamento, ao propiciar a descida do gradiente, foi a adição do algoritmo *Batch Normalization* (normalização de lote) como uma camada integrante do modelo. Justificou-se a adição dessa camada de normalização pelo em razão do aceleração do tempo de treinamento, além de possibilitar um melhor desempenho da taxa de aprendizagem dos algoritmos otimização *Stochastic Gradient Descent* (SGD) [11] e *Adaptive Moment Estimation* (ADAM) [16].

A respeito da otimização do modelo, em termos de erro de generalização, no treinamento, utilizou-se o algoritmo SGD. Nesse algoritmo, considerou-se, inicialmente, uma taxa de aprendizagem igual a 1×10^{-4} , isto é, $\alpha = 0.0001$. No entanto, essa configuração foi ajustada, de acordo com o resultado do desempenho do modelo nos ciclos de treinamento.

Na performance do tempo de treinamento, buscou-se a instauração da técnica de regularização *Dropout* [59]. Essa técnica foi inserida ao modelo como uma camada de rede com o objetivo de eliminar neurônios redundantes, ou seja, aqueles que extraíram características ocultas iguais ou semelhantes de uma determinada camada. No modelo proposto, a rede neural possui duas camadas de *Dropout*, uma após a camada de convolução e outra, depois da camada de GRU Bidirecional.

Buscou-se, também, ajustar a quantidade de amostras de dados em cada época de treinamento, por meio do parâmetro *Batch size* (tamanho de lote). O objetivo da utilização desse

parâmetro, é encontrar um número adequado de amostras que envolvesse o desempenho global do modelo, ao longo do tempo de treinamento, até a convergência [28]. Desse modo, as previsões são comparadas com as variáveis de saída (valores esperados) e um erro é calculado no final de cada lote, durante o treinamento.

Os valores de cada parâmetros foram ajustados, empiricamente, de modo que o modelo evoluísse no processo de treinamento, em termos de redução, pela medida de erro da função de perda. O resultado do ajuste dos parâmetros são apresentados, a seguir, nos ciclos de treinamento.

5.2 Resultados e discussão

Nesta seção, apresentam-se os dados numéricos do desempenho de GSTNet resultantes da aprendizagem do modelo no ciclo de treinamento. Os dados são provenientes da avaliação de GSTNet no conjunto de validação e conjunto de teste. Esse processo de validação, essencialmente, no conjunto de teste, consiste em verificar a precisão do modelo, ao realizar previsões de dados não observados, a partir daqueles observados no treinamento. A seção finaliza com a apresentação dos resultados da comparação de GSTNet com modelos tradicionais de previsão da *Throughput* auxiliados pela média móvel.

5.2.1 Treinamento

Conduziu-se o treinamento da rede do modelo GSTNet em três ciclos. Os ciclos representam a evolução do modelo, de acordo com o ajuste de parâmetros. Em dados ciclos, ajustou-se a taxa de aprendizagem, função de ativação, normalização de lote e o tipo de algoritmo de otimização. O ajuste de tais contribuíram para se alcançar um modelo refinado com elevada capacidade de generalização, em diversas amostras do conjunto de treinamento e validação.

5.2.1.1 O primeiro ciclo

No primeiro ciclo de treinamento, desenvolveu-se uma rede neural com sete camadas. As camadas integrantes são: camada convolucional (CNN), camada de *Dropout*, camada de normalização de lote, uma camada BiGRU, outra camada de *Dropout*, uma camada de função de ativação (RELU) e uma camada densa, que representa a saída da rede com cinco unidades de neurônios. A Tabela 5 apresenta as camadas e os parâmetros de rede.

Na camada de convolução, ajustou-se o número de filtros e tamanho do *stride* do *kernel* com tamanho 64 e dimensão 6 x 1, respectivamente, para capturar as informações temporais, sequencialmente no tempo. Como função de ativação dessa camada, utilizou-se a RELU em razão de sua capacidade de aprender características complexas nas amostras de dados [24].

Após a camada de CNN, adicionou-se uma camada de *Dropout*, estabelecendo-se uma frequência de até 20% de regularização para remover aquela quantidade de amostras de entrada de dados e ativações da camada anterior. Justifica-se a adição dessa camada em razão da prevenção

Tabela 5: Ciclo 1 - configurações de parâmetros.

Parâmetros	Quantidade	Camada
Filtro convolucional (Convolutional Filter)	64	CNN
Tamanho do kernel (Kernel size)	6x1	CNN
Função de ativação (Activation)	RELU	CNN
Dropout	20%	—
Normalização de lote (Batch Normalization)	64	—
Estado oculto (Hidden State)	64	BiGRU
Dropout	20%	—
Função de ativação (Activation)	RELU	—
Neurônios	5	Densa
Otimizador (Optimizer)	SGD	—
Taxa de aprendizagem (Learning Rate)	1×10^{-4}	—
Épocas (Epochs)	140	—

Fonte: Elaborado pelo autor.

de *overfitting* que ela fornece, além de ao diminuir o surgimento de *underfitting*, no início do treinamento [44]. Veja que a camada de *Dropout* é utilizada duas vezes na arquitetura da rede.

A camada seguinte é uma camada de normalização de lote. A finalidade dessa camada na arquitetura da rede consiste na possibilidade de acelerar o treinamento e utilizar uma taxa de aprendizagem maior no algoritmo de otimização, durante o treinamento do modelo. Nessa camada, estabeleceu-se uma configuração fixa de lote com 64 amostras de dados. Constatou-se que uma quantidade de lote maior ou menor afetava a precisão do modelo.

Na camada recorrente, uma BiGRU apresentou melhor desempenho, em contrapartida a outras redes da mesma natureza para o problema de previsão em questão. Nessa camada, uma quantidade de 64 estados escondidos foi estabelecida. Assim como a camada convolucional, BiGRU também adere à função de ativação RELU, em sua implementação.

Essa mesma função é adicionada, separadamente, como uma camada integrante à arquitetura da rede, na finalidade de introduzir a propriedade de não linearidade às informações de entrada das camadas anteriores. Descobriu-se, empiricamente, que tal função melhorava, significativamente, o aprendizado da rede.

Finalmente, a camada densa é a última camada integrante à arquitetura da rede. Essa camada é responsável pela saída das das informações processadas pelas rede. A saída é composta de cinco neurônios, os quais representam os sinais de *Throughput* a serem previstos.

Com as configurações estabelecidas para o Ciclo I, o desempenho do modelo no treinamento é ilustrado na Figura 11. Mostra-se o comportamento das curvas, ao longo da iterações, evidenciando um treino sem presença do fenômeno de *underfitting*, visto que elas apresentam uma relação quase equilibrada na taxa de erro. Nas primeiras épocas, as curvas de treino e validação mostram um rápido decaimento, estabilizando-se antes do limite de época definido. O valor da função de perda, no treino, resulta em um valor de MAE igual a 0.1651. Por outro lado, na validação, o valor de MAE é igual a 0.1543. Diante desse resultado, verifica-se que o risco empírico foi minimizado, ao buscar-se a função que melhor se ajusta aos dados de treinamento.

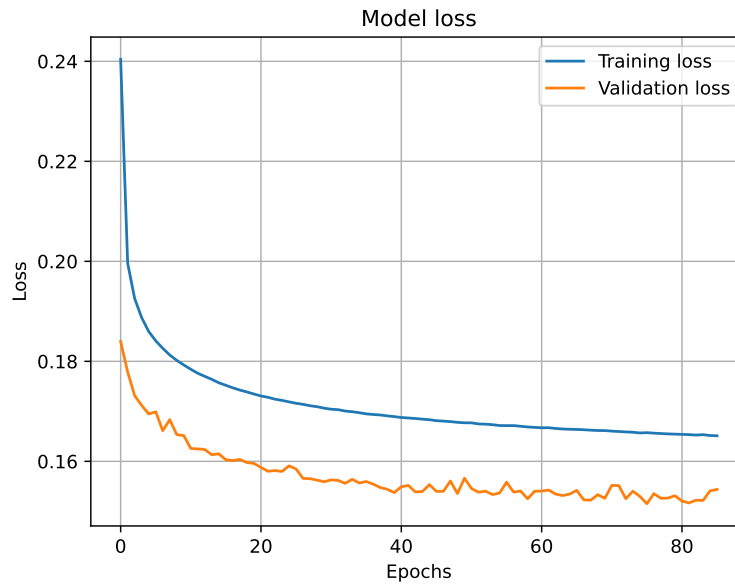


Figura 11: Desempenho de GSTNet no primeiro ciclo.

O desempenho apresentado revela que o modelo de rede neural desenvolvido no primeiro ciclo, convergiu, produzindo um modelo capaz de detectar padrões temporais nas variáveis de séries temporais apresentadas, ao passo que encontrou os pesos ideais, durante o treinamento, sob as configurações de parâmetros realizadas.

5.2.1.2 O segundo ciclo

No segundo ciclo de treinamento, tentou-se, melhorar o desempenho de GSTNet com o objetivo de desenvolver um modelo com maior capacidade de generalização que o modelo produzido no primeiro ciclo. Para alcançar esse objetivo, adicionou-se mais duas camadas à arquitetura da rede, mudança de algoritmo de otimização da descida do gradiente e mudança de função de ativação. O ajuste de parâmetros, como taxa de aprendizagem, filtros da camada convolutiva e estados da camada recorrente permaneceram inalterados, conforme o ciclo anterior.

Uma camada *Flatten*, responsável por dimensionar os dados num grande vetor, e uma camada densa com 128 unidades de neurônios foram adicionados a rede do modelo de maneira empírica. A Tabela 6 mostra os parâmetros de redes adicionados. Devido a essa nova configuração, o modelo GSTNet surge com uma versão de maior maior complexidade, em termos de ajuste de parâmetros.

Ao contrário da primeira versão do modelo, a iteração de treinamento perdurou, aproximadamente, 21 iterações, devido o valor da taxa de perda não apresentar melhoria na validação. A taxa de aprendizagem, permaneceu com $\alpha = 0.0001$. Essa taxa foi estabelecida para o algoritmo de otimização ADAM (*Adaptive Moment Estimation*) [29]. *Dropout* e *Batch Normalization* permaneceram inalterados.

A Figura 12 ilustra o desempenho resultante de GSTNet, no segundo ciclo de treinamento.

Tabela 6: Ciclo 2 - configurações de parâmetros.

Parâmetros	Quantidade	Camada
Filtro convolucional (Convolutional Filter)	64	CNN
Tamanho do kernel (Kernel size)	6x1	CNN
Função de ativação (Activation)	RELU	CNN
Dropout	20%	—
Normalização de lote (Batch Normalization)	64	—
Estado oculto (Hidden State)	64	BiGRU
Função de ativação (Activation)	SELU	BiGRU
Dropout	20%	—
Flatten	—	—
Neurônios	128	Densa
Neurônios	5	Densa
Função de ativação (Activation)	SELU	—
Otimizador (Optimizer)	ADAM	—
Taxa de aprendizagem (Learning Rate)	1×10^{-4}	—
Épocas (Epochs)	140	—

Fonte: Elaborado pelo autor.

A perda do modelo decai, rapidamente nas primeiras épocas, estabilizando-se próximo do limite de época definido, de acordo com as curvas. Esse desempenho revela que o modelo apresenta evidência de ocorrência de *underfitting* durante o processo de aprendizagem, constituindo, dessa forma, um modelo não desempenho aceitável, apesar da perda no treinamento e validação apresentar MAE igual a 0.12029 e 0.14062, respectivamente.

Contata-se que o desempenho apresentado na segunda versão de GSTNet, conforme as curvas de perda, não apresentou um equilíbrio durante as iterações de treinamento, indicando um modelo característico de *underfitting*, uma vez que ele permaneceu viciado nos dados de treinamento e não minimizou o risco empírico. Uma das principais causas para esse comportamento do modelo, atribuí-se ao algoritmo de otimização ADAM, pois não foi capaz de chegar ao um ponto mínimo na descida do gradiente. Por esse motivo, implementou-se outra versão GSTNet, esta sendo a última.

5.2.1.3 O terceiro ciclo

O terceiro ciclo de treinamento produziu a última versão de GSTNet, um modelo que alcançou maior capacidade de generalização no conjunto de validação, em relação as suas versões anteriores. Essa capacidade foi possível por meio, novamente, de ajustes nos parâmetros da rede neural. As configurações foram realizados na camada de convolução e no algoritmo de otimização, ao observar-se que esses parâmetros não ocasionavam mudanças significativas nos treinamentos da rede.

Na camada de convolução, removeu-se sua função de ativação, a RELU. No algoritmo de otimização, alterou-se para SGD e diminuiu-se a taxa de aprendizagem α , como pode ser visto na Tabela 7, em relação àquelas utilizadas nas versões anteriores. Os demais parâmetros e

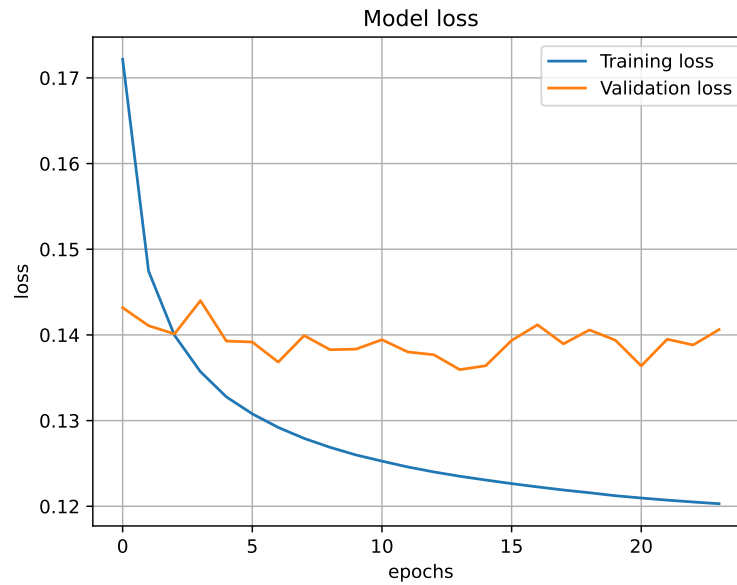


Figura 12: Desempenho de GSTNet no segundo ciclo.

componentes da arquitetura da rede da versão anterior permaneceram inalterados.

Tabela 7: Ciclo 3 - configurações de parâmetros.

Parâmetros	Quantidade	Camada
Filtro convolucional (Convolutional Filter)	64	CNN
Tamanho do kernel (Kernel size)	6x1	CNN
Dropout	20%	—
Normalização de lote (Batch Normalization)	64	—
Estado oculto (Hidden State)	64	BiGRU
Dropout	20%	—
Flatten	—	—
Neurônios	128	Densa
Função de ativação (Activation)	SELU	—
Neurônios	5	Densa
Otimizador (Optimizer)	SGD	—
Taxa de aprendizagem (Learning Rate)	1×10^{-2}	—
Épocas (Epochs)	140	—

Fonte: Elaborado pelo autor.

O alinhamento e ajustes dos parâmetros de rede durante o ciclo de treinamento, em questão, proporcionou que a rede estabilizasse sua aprendizagem, ao diminuir o decaimento das curvas de aprendizagem, como pode ser visto na Figura 13, denotando que o acréscimo de épocas de treinamento deixara de ser necessário.

No início das primeiras épocas, os valores da curva de treinamento decaem, rapidamente. Essa curva estabiliza-se quando o treinamento atinge a época número 81, onde o erro de treinamento é representado com um MAE igual a 0.13. Em contrapartida, a curva da validação iniciou-se com valores mais baixos, em termos de erro, estabilizando-se com MAE igual a 0.11.

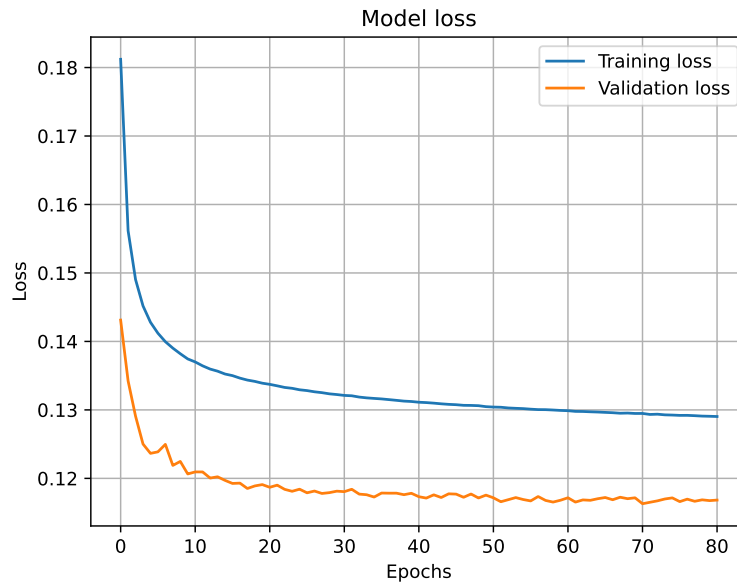


Figura 13: Desempenho de GSTNet no terceiro ciclo.

Constata-se que o desempenho de GSTNet no ciclo final de treinamento atingiu o melhor resultado possível, após vários ajustes de parâmetros em sua arquitetura. Um dos fatores determinantes, nesse processo, foi a definição da taxa de aprendizagem, a qual contribuiu para o projeto de um modelo predictor com baixo erro de generalização, no conjunto de validação.

Realizada a primeira avaliação de GSTNet durante o processo de treinamento, para ajustes de parâmetros no conjunto de validação, a próxima etapa consistiu da segunda avaliação do modelo, mas agora, no conjunto de teste, um subconjunto com observações de dados desconhecidas. Os resultados dessa validação são apresentados na próxima seção.

5.2.2 Teste: predição de GSTNet

A validação do modelo GSTNet foi realizada em subconjuntos do conjunto de teste para avaliar sua capacidade de predição, em relação a outros modelos precursores esboçados para a mesma finalidade. O desempenho de predição foi comparado com modelos de redes neurais recorrentes e com modelos tradicionais de aprendizagem de máquina. Realizou-se também a validação com modelos primitivos de estimativa de largura de banda baseados em média móvel, visto que essas abordagens são comumente empregadas em algoritmos de adaptação de taxa para ajustar dinamicamente a qualidade do vídeo à capacidade da rede em tempo real.

A condução das avaliações para a validação de GSTNet incluiu três cenários de condições de medição da *Throughput* originados do conjunto de teste, onde buscou-se por medições em que um subconjunto particular de clientes concentrou os maiores volumes de conexões, relativamente aos outros (*Higher Flow Measurements* - HFM); medições com maior predominância de volume (*Majority Measurements* - MAM) e, por fim, medições de *Throughput* oriundas de conexões de clientes com altas taxas de largura de banda, porém, com menor predominância no conjunto de

teste, as quais foram intituladas de *Minority Measurements* (MIM).

A razão para a validação do modelo proposto utilizando fracções do conjunto de teste se deve pela irregularidade dos intervalos de medição e à característica esparsa do conjunto de dados, de acordo com a análise realizada no Capítulo 3.

Os modelos profundos considerados na comparação com GSTNet incluem-se: GRU, CNNGRU, LSTM, CNNLSTM, BiLSTM e CNNBiLSTM. Alguns desses modelos são combinações de redes, e.g., rede convolucional com adição de uma rede recorrente. As redes testadas seguiram as mesmas configurações de parâmetros apresentados na Tabela 7.

Em relação aos modelos de aprendizagem de máquina, implementou-se os algoritmos *Random Forest* (RF) e *K-Nearest Neighbors* (KNN) da biblioteca Scikit-learn¹, em modo regressor, reajustando-se apenas as configurações mínimas em seus parâmetros, conforme o perfil de cada um visto na seção 2.2 do Capítulo 2. O RF foi configurado com 100 árvores aleatórias e profundidade máxima igual a 6. KNN foi configurado com k vizinhos mais próximos, cujo k foi igual a 10. A distância considerada no treinamento foi a distância de Minkowski. Os dois modelos foram treinados com 60% de amostras do conjunto de dados e 40% foram utilizadas para fazer previsões.

Os modelos baseados em média móvel utilizados incluem-se, a média móvel simples - *Simple Moving Average* (SMA), a média móvel ponderada - *Weighted Moving Average* (WSA) e a média móvel exponencial - *Exponential Moving Average* (EMA). Essas métricas foram empregadas para calcular o histórico de medições da *throughput*, em que cada sessão foi dividida em uma janela de tempo, onde q representa o histórico e k , o horizonte de predição, ambos configurados com 20 e 10 segundos, respectivamente, conforme apresentado no Capítulo 4.

5.2.2.1 Higher Flow Measurements

A primeira avaliação foi realizada no cenário (*trace*) *Higher Flow Measurements* (HFM), um subconjunto composto de 2849 medições observadas. Essa fração do conjunto de teste caracteriza-se pela sua heterogeneidade no espectro da disposição da dinâmica da largura de banda entre as sessões de cada cliente. As observações incluem um cenário que apresenta média igual a 2.88 e máxima de 41.9 Mbit/s para a *throughput* dos canais.

A Figura 14a apresenta o histograma do quantitativo das medições e a Figura 14b apresenta a distribuição da *throughput* para cada cliente. Na primeira figura, apresenta-se o conjunto dos clientes com maior número de medições em relação ao demais, no conjunto de teste. Pode-se observar que o cliente 01 contém o maior volume de medições, e por conseguinte, maior número de sessões. Com isso, a característica inerente a esse cenário são os clientes deterem o maior número de sessões de vídeo.

A análise do conjunto foi realizada de maneira empírica com o pressuposto de que esse conjunto poderia fornecer padrões de medições regulares, em dados instantes de tempo, o que pode ser útil na precisão da predição. A segunda figura, por outro lado, mostra a *throughput*

¹ <https://scikit-learn.org/stable/>

das conexões de cada cliente. Observa-se a identificação de medições com inúmeras taxas discrepantes de *throughput* e a identificação expressiva de taxas abaixo de 10 Mbit/s.

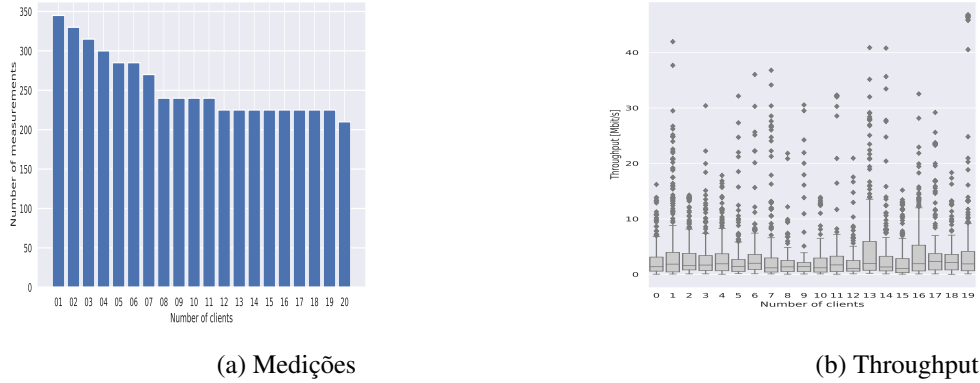


Figura 14: Número de medições e throughput do cenário HFM

Na avaliação da predição, GSTNet apresentou melhor desempenho, juntamente com a rede BiLSTM, em relação aos modelos convencionais de aprendizagem de máquina, modelos profundos e médias móveis, em termos de MAE e RSE, como pode ser visto na Figura 15. GSTNet alcança excelente desempenho com MAE e RSE igual a 0.0236 e 0.0028, respectivamente. BiLSTM foi o modelo pelo qual obteve menor erro no MAE com valor igual a 0.0088, mas com RSE igual a 0.0082. Observe que os valores de MAE e RSE foram transformados para a escala logarítmica na base 10, pois torna mais fácil visualizar tanto valores muito pequenos quanto maiores no mesmo gráfico, em razão da assimetria apresentada.

Os resultados da avaliação de GSTNet no conjunto HFM, mostram que a predição do modelo proposto é superior em comparação aos preditores convencionais, profundos e aqueles baseados em média móvel na métrica RSE. Constata-se que tal desempenho é resultante não apenas do baixo erro de generalização do modelo, mas também do padrão praticamente regular da *throughput* presente nas medições das sessões de cada cliente, o que confirma o pressuposto inicial sobre a característica desse cenário, conforme ilustra a Figura 17. Esse fragmento do cenário HFM mostra, além do padrão de flutuação regular do canal (entre 0.2 e 0.6 Mbit/s), a predição de maneira precisa que GSTNet realiza, na qual não sobrestima a largura de banda, principalmente quando a largura de banda aumenta (próximo de 30 segundos, no exemplo), onde a curva azul representa os dados reais e a curva vermelha representa a *throughput* prevista.

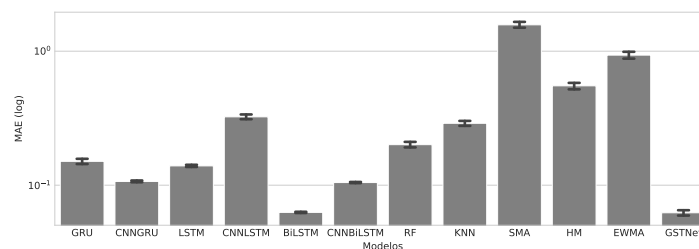


Figura 15: Desempenho de GSTNet no cenário HFM com MAE

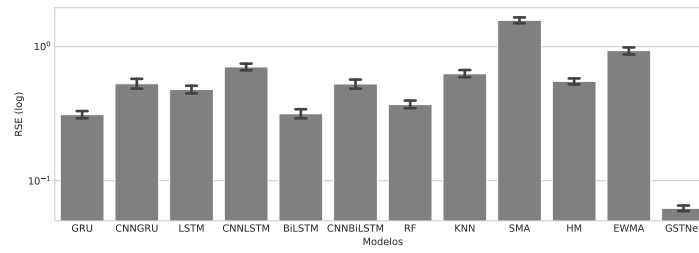


Figura 16: Desempenho de GSTNet no cenário HFM com RSE

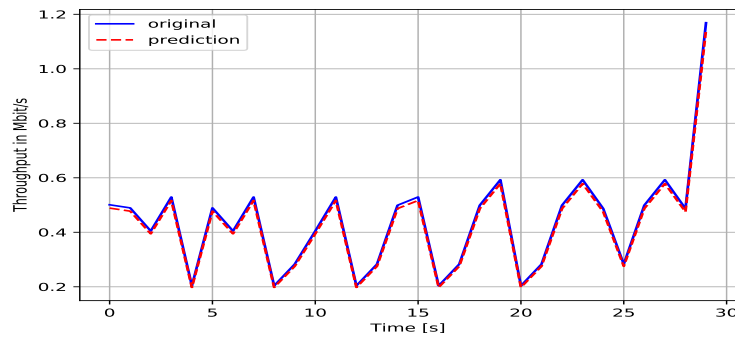


Figura 17: A *throughput* prevista por GSTNet (vermelha) versus a real (azul) no cenário HFM.

5.2.2.2 Majority Measurements

A segunda avaliação foi realizada no conjunto MAM, que representa a maior fração de medições do conjunto de teste. O conjunto contém uma quantidade de 1 109 489 medições, agrupadas em 73 966 sessões. Com uma *throughput* média de 2.87 e máxima de 29.0 Mbit/s dos canais, esse cenário foi utilizado na avaliação de GSTNet pela expressiva quantidade de dados apresentada e por considerar-se que sua representatividade, é formada, majoritariamente, por condições irregulares de flutuação da largura de banda da rede.

Os resultados de desempenho de GSTNet, nesse cenário, mostram que o modelo apresenta maior erro na métrica MAE com valor igual a 0.1995, em relação ao modelo CNNGRU, no qual alcançou um valor de 0.0883, conforme pode ser visto na Figura 18. Essa diferença de desempenho pode estar relacionada ao tipo de arquitetura implementada nos modelos de GRU, pois, tanto CNNGRU quanto GSTNet são baseados em tal rede. Outra constatação seria a presença de tráfego regular e irregular de flutuação do canal, embora exista uma maior tendência de irregularidade nas oscilações.

No entanto, o modelo proposto mantém superioridade na métrica RSE, em relação a todos os modelos comparados com RSE igual 0.0929, como pode ser visto na Figura 19. Observa-se que GSTNet consegue prever a tendência de *throughput*, de modo preciso, apesar da flutuação abrupta da largura de banda em vários instantes da sessão, em razão da instabilidade do canal, uma vez que é uma característica inerente ao cenário analisado.

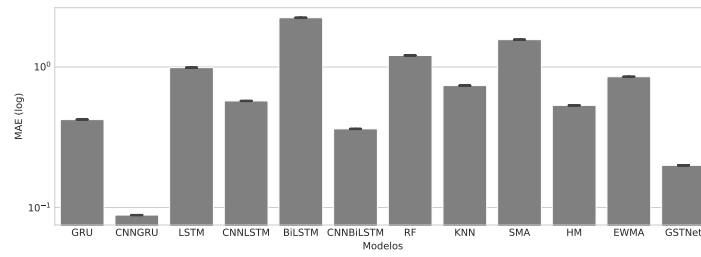


Figura 18: Desempenho de GSTNet no cenário MAM com MAE

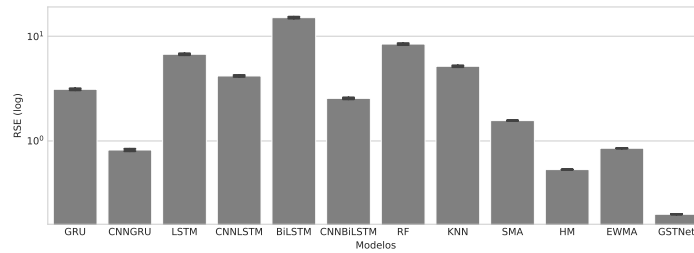


Figura 19: Desempenho de GSTNet no cenário MAM com RSE

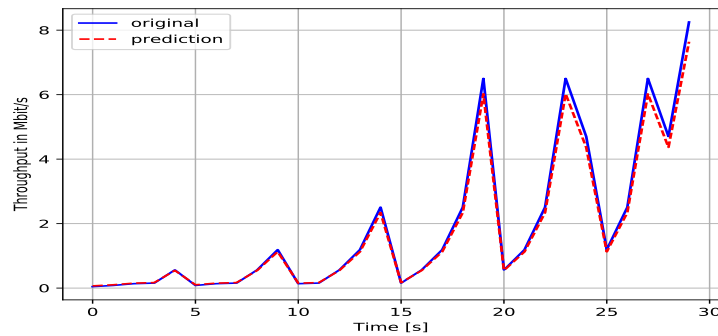
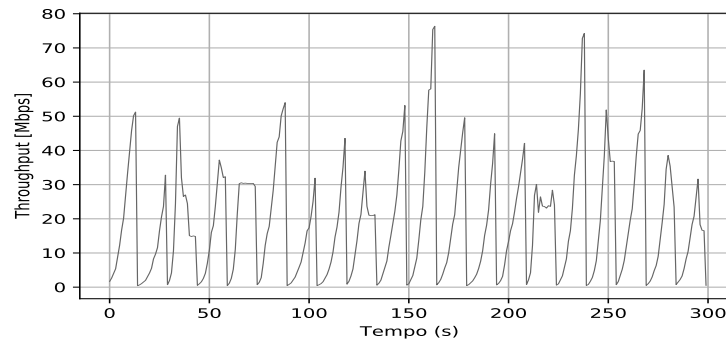


Figura 20: Predição de GSTNet no cenário MAM

5.2.2.3 Minority Measurements

O cenário MIM é um subconjunto de teste composto de 85319 medições, originado de 5687 observações de sessões de vídeos. Essas observações estão dispostas numa fração menor em relação a outros cenários. Além disso, possui uma variedade de perfis de clientes com as maiores velocidades de conexões de tráfego de rede, em que a *throughput* dos canais alcança taxa média igual a 19,59 Mbit/s e máxima de 295,0 Mbit/s. O cenário também é identificado pela forte presença de tráfego com alta taxa de largura de banda, revelando que a qualidade média da transmissão de vídeo deveria ser melhor.

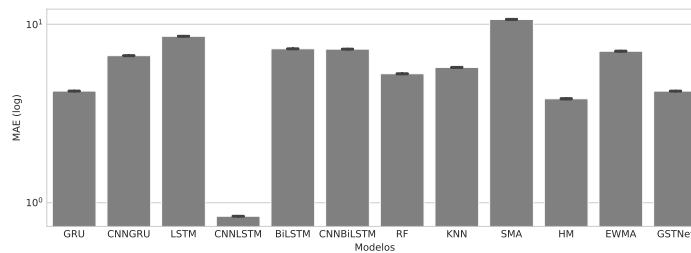
A Figura 21 apresenta um fragmento do comportamento de centenas de medições do cenário, as quais somam cerca de 300 segundos de vídeo. Observa-se que existe a presença de flutuações contínuas durante todas as sessões com picos e vales (períodos de baixa taxa de *throughput*) que ocorrem em curtos intervalos de tempo. Ou seja, ocorrem degradações repentinas nas condições da rede.

Figura 21: Dinâmica da flutuação da *throughput* no cenário MIM

A avaliação realizada nesse cenário mostrou que GSTNet obteve desempenho superior em relação à maioria dos modelos preditores comparados, exceto em relação aos modelos profundos, tais como GRU e CNNLSTM, pois obteve desempenho inferior nas duas métricas utilizadas, ao atingir um valor igual a 4,22 para MAE e 0,3051 para o RSE, enquanto que o modelo CNNLSTM alcançou 0,8403 e 0,0582 para MAE e RSE, respectivamente, superando, consideravelmente, o modelo nesse cenário. Referente aos modelos de médias móveis, HM foi o modelo que obteve um melhor desempenho em relação ao modelo prospoto. As Figuras 22 e 23 apresentam o desempenho do modelo proposto.

O principal motivo verificado para o alto erro de generalização de GSTNet em relação aos modelos de destaque na comparação, pode ser à característica do canal, em que o modelo falha na tentativa de prever o sinal de largura de banda, que por sua vez, representa uma série de tempo com dependência de longo prazo. Com isso, o aspecto do canal possui as condições favoráveis para a predição de CNNLSTM com baixo erro, pois a arquitetura de rede desse modelo é, por essência, desenvolvida para capturar sinais com esse tipo de característica.

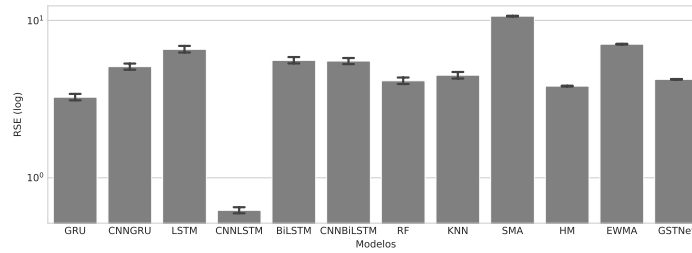
Figura 22: Desempenho de GSTNet no cenário MIM com MAE



Fonte: Autoria própria.

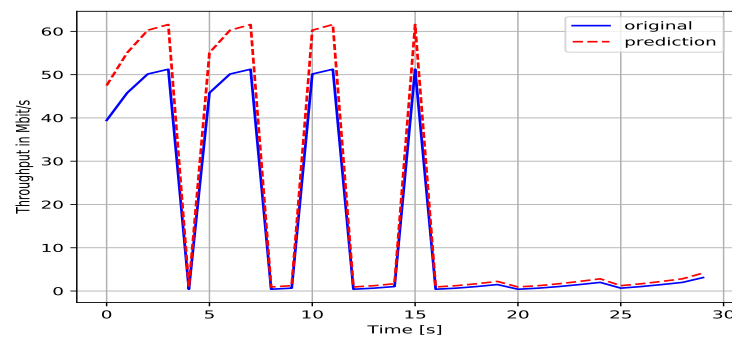
Constatou-se que o modelo proposto falha em prever com precisão a *throughput* disponível, principalmente, em momentos de grande picos, onde o fluxo da dinâmica da largura de banda aumenta repentinamente, como pode ser visto na Figura 24. Observa-se que o modelo subestima a largura de banda nesses picos, evidenciando sua baixa capacidade de generalização no cenário MIM, onde a taxa de largura de banda é absolutamente maior em relação aos outros cenários, assim como a taxa média de *throughput*.

Figura 23: Desempenho de GSTNet no cenário MIM com RSE



Fonte: Autoria própria.

Figura 24: Predição de GSTNet no cenário MIM



Fonte: Autoria própria.

Uma abordagem para mitigar a imprecisão de GSTNet diante de cenários com padrões de acesso de alta velocidade seria a realização de captura de medições semelhantes àquelas com velocidades mais baixas, em que muitos clientes possuem padrões de acesso representativos e em grande volume, ou seja, os dados são desprovidos de esparsidade. Portanto, essa é a questão-chave que reflete no desempenho do modelo proposto neste trabalho.

5.3 Considerações finais

Apresentou-se, neste capítulo, as sequências de desenvolvimento do modelo GSTNet e os resultados numéricos de desempenho no processo de predição, em comparação a outros modelos preditores, em particular, aqueles baseados em média móvel. O desenvolvimento consistiu na implementação de três ciclos de treinamentos com o objetivo de alcançar uma arquitetura de rede neural que se ajustasse à natureza dos dados fornecidos. Com o modelo aprimorado após os sucessivos treinamentos, realizou-se avaliações em três cenários representativos do conjunto de teste para validar seu desempenho em observações de medições de *throughput* desconhecidas. Constatou-se que o modelo proposto apresentou superioridade de desempenho na maioria das medições do conjunto de teste para canais com *throughput* média de 2.88 Mbit/s, principalmente em cenários onde existem as maiores concentrações de medições para usuários com planos de dados com tais velocidades de conexão.

CAPÍTULO 6

Conclusões e considerações finais

As conexões de aplicações de *Streaming* de vídeo que trafegam pela rede de melhor esforço encontram dificuldades de chegar ao destino em razão da natureza dinâmica do TCP/IP, das condições de rede e das tecnologias subjacentes envolvidas no processo de distribuição de vídeo. Por conta dessa dinâmica inata, a modelagem do tráfego dos canais de redes acompanham conexões heterogêneas fim a fim. Com isso, a *Throughput*, que consiste na capacidade do canal, sofre oscilações sob intervalos de tempo desconhecidos num formato de "dente de serra", o qual contribui para degradação da qualidade das aplicações quando chegam no usuário final, na qual corresponde à audiência dos provedores de serviço de vídeo.

Nesse cenário, a maximização da qualidade visual de vídeo é um grande desafio para plataformas de serviços populares de vídeo, como por exemplo, TikTok, Instagram e Kwai, nas quais o engajamento da audiência é um fator decisivo para a promoção de anúncios e publicidade de modo geral. Essas plataformas são dependentes de tecnologias de empacotamento e distribuição de vídeo. No entanto, muitas aplicações de vídeo sofrem com o estado dinâmico e instável do canal da rede durante a conexão. Muitas dessas aplicações, utilizam algoritmos de adaptação de taxa de bit para proporcionar uma experiência de visualização da audiência adequada aos recursos disponíveis da rede e da audiência. Esse processo envolve a captura de medidas atuais da rede como a *Throughput*, que por sua vez, é estimada com base na sessão em andamento, gerando um subaproveitamento da largura de banda, com a presença natural de períodos de atividades OF/OFF do canal, refletindo no download de segmentos seguintes de vídeo com taxas impróprias para o andamento da sessão.

Diante dessa problemática, este trabalho realizou uma análise minuciosa num conjunto de dados com milhares de medições de rede oriundas de conexões de usuários voluntários, no Brasil, por meio um projeto chamado Neubot, coletadas entre 2015 e 2018. Durante o estudo, buscou-se fatores de rede e evidências que estivessem alguma relação com a estabilidade da vazão do canal por meio da análise de características relacionadas diretamente com a sessão.

Após a análise, que incluiu um processo de seleção de características relevantes, implementou um modelo preditivo de *Throughput* para ser treinado com tais características - histórico de acesso - em contrapartida a aqueles preditores baseados em média móvel. Essa abordagem foi a grande contribuição deste trabalho ao construir um modelo de estimação de vazão com técnicas de aprendizagem profunda.

O desenvolvimento do modelo, sob o nome GSTNet, empregou técnicas de aprendizagem de máquina na implementação de uma rede neural recorrente profunda com propriedades de uma GRU para processamento bidirecional usando dados dispostos na forma de séries temporais. Durante as sequências de implementação, três ciclos de treinamentos foram necessários para que se pudesse refinar a rede neural, de forma que seu desempenho atingisse o menor baixo erro de generalização possível, ao ajustar parâmetros importantes na arquitetura do modelo.

Após o último ciclo, o modelo treinado foi comparado e avaliado com modelos previsores baseados em rede neural recorrente e em média móvel, em cenários (também chamados de traces) de medições do conjunto de teste. Os modelos previsores incluíram modelos convencionais de aprendizagem de máquina, como por exemplo, *Random Forest* e *K-Nearest Neighbors*. Já os modelos profundos incluíram redes LSTM, GRU, BiLSTM e combinações de cada uma com rede CNN para redes mais complexas em termos de parâmetros de arquitetura.

Na avaliação do modelo, que consistiu na validação do modelo, três cenários de medições foram analisados e selecionados para testar o desempenho de GSTNet. Os cenários caracterizavam diferentes condições de rede, como acentuada flutuação da largura de banda e instabilidade.

Apesar da instabilidade dos canais, a validação do modelo no teste revelou que o desempenho de predição foi superior em comparação aos modelos previsores profundos e baseados em média móvel para cenários específicos de medições de rede. A superioridade de GSTNet prevaleceu em cenários onde as condições do tráfego de rede, tais como o padrão de acesso, caracterizaram conexões com taxas médias de *Throughput* igual a 2,88 Mbit/s e máxima de 41,9 Mbit/s, formando a maioria das observações do conjunto de teste, onde a estabilidade das conexões era majoritariamente estável.

Em contrapartida, o modelo proposto obteve desempenho inferior para cenários de medições com velocidades maiores de conexões, isto é, alta largura de banda com taxa média de *Throughput* igual a 19,59 Mbit/s e máxima de 295.0 Mbit/s. Isso revela que, pelo fato do cenário, em questão, apresentar menor representatividade no conjunto de teste e por integrar clientes com medições únicas, evidenciando a esparsidade do conjunto de dados com padrões irregulares de medições, o modelo proposto não se ajustou ao padrão de acesso fornecido, apresentando maior erro de generalização, em relação a outros previsores.

Portanto, é um desafio estimar com precisão a *Throughput* futura guiada por dados históricos com diferentes características e espectros de infraestrutura de rede, principalmente de sessões com 30 segundos de reprodução de vídeo. Foi visto neste trabalho que vários esforços foram realizados com a finalidade de desenvolver um preditor baseado em redes neurais profundas

para prever a vazão futura para uma dada sessão. Dessa forma, a ideia fundamental é estimar a *Throughput* futura, a partir de medições de rede que estejam vinculadas ao um padrão constante de acesso por parte da audiência, num determinado contexto e perfil de rede. Nesse contexto,

Como lições aprendidas, o presente trabalho destaca: i) o treinamento de modelos com um conjunto representativo nem sempre é suficiente para se alcançar uma predição precisa, devido à natureza esparsa do conjunto, ii) a normalização dos dados é um fator que influencia no desempenho do modelo e que precisa ser realizada de várias maneiras, iii) o reconhecimento de padrões espaço/temporais no conjunto de dados é desafiador devido às características da sessão oriundas da dinâmica do protocolo TCP.

Os trabalhos futuros incluem a investigação em outras fontes de dados para cenários de redes 4G e 5G na finalidade de aprimorar o modelo de rede neural, além de implementar e adicionar técnicas de aprendizagem profunda contemporânea, como mecanismos de atenção temporal em sua arquitetura. Por último, um estudo de caso será conduzido por meio de um simulador realista de sessões de *Streaming* de vídeo com os principais algoritmos de adaptação de taxa.

Referências

- [1] Adobe HTTP Dynamic Streaming. Disponível em: <<https://www.adobe.com/br/products/hds-dynamic-streaming.html>>. Acessado em: out. 2019.
- [2] HTTP Live Streaming. Disponível em: <<https://developer.apple.com/streaming/>>. Acessado em: out. 2019.
- [3] Microsoft Silverlight Smooth Streaming. Disponível em: <<https://www.microsoft.com/silverlight/smoothstreaming/>>. Acessado em: out. 2019.
- [4] Akhshabi, S., Anantakrishnan, L., Dovrolis, C., and Begen, A. C. Server-based traffic shaping for stabilizing oscillating adaptive streaming players. In *Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video* (New York, NY, USA, 2013), NOSSDAV '13, ACM, pp. 19–24.
- [5] Baldi, P., Brunak, S., Frasconi, P., Soda, G., and Pollastri, G. Exploiting the past and the future in protein secondary structure prediction. *Bioinformatics* 15, 11 (1999), 937–946.
- [6] Bampis, C. G., Li, Z., Katsavounidis, I., Huang, T.-Y., Ekanadham, C., and Bovik, A. C. Towards perceptually optimized end-to-end adaptive video streaming. *arXiv preprint arXiv:1808.03898* (2018).
- [7] Basso, S., Servetti, A., Masala, E., and De Martin, J. C. Measuring dash streaming performance from the end users perspective using neubot. In *Proceedings of the 5th ACM Multimedia Systems Conference* (New York, NY, USA, 2014), MMSys '14, ACM, pp. 1–6.
- [8] Bentaleb, A., Taani, B., Begen, A. C., Timmerer, C., and Zimmermann, R. A survey on bitrate adaptation schemes for streaming media over http. *IEEE Communications Surveys & Tutorials* 21, 1 (2018), 562–585.

- [9] Bentaleb, A., Timmerer, C., Begen, A. C., and Zimmermann, R. Bandwidth prediction in low-latency chunked streaming. In *Proceedings of the 29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video* (New York, NY, USA, 2019), NOSSDAV '19, ACM, pp. 7–13.
- [10] Biernacki, A. Improving quality of adaptive video by traffic prediction with (f) arima models. *Journal of Communications and Networks* 19, 5 (2017), 521–530.
- [11] Bjorck, N., Gomes, C. P., Selman, B., and Weinberger, K. Q. Understanding batch normalization. *Advances in neural information processing systems* 31 (2018).
- [12] Brunnström, K., Beker, S. A., De Moor, K., Dooms, A., Egger, S., Garcia, M.-N., Hossfeld, T., Jumisko-Pyykkö, S., Keimel, C., Larabi, M.-C., et al. Qualinet white paper on definitions of quality of experience.
- [13] Casari, A., and Zheng, A. Feature engineering for machine learning. *O'Reilly Media, Inc.* (2018), 218.
- [14] Chollet, F. Deep learning with python. *Deep Learning With Python* (2018).
- [15] Clements, M. P., and Hendry, D. F. On the limitations of comparing mean square forecast errors. *Journal of Forecasting* 12, 8 (1993), 617–637.
- [16] Diederik, P. K. Adam: A method for stochastic optimization. (*No Title*) (2014).
- [17] Glorot, X., and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (2010), JMLR Workshop and Conference Proceedings, pp. 249–256.
- [18] Guessoum, S., Belda, S., Ferrandiz, J. M., Modiri, S., Raut, S., Dhar, S., Heinkelmann, R., and Schuh, H. The short-term prediction of length of day using 1d convolutional neural networks (1d cnn). *Sensors* 22, 23 (2022), 9517.
- [19] Guo, L., and Matta, I. The war between mice and elephants. In *Proceedings Ninth International Conference on Network Protocols. ICNP 2001* (2001), pp. 180–188.
- [20] Hall, P., Park, B. U., Samworth, R. J., et al. Choice of neighbor order in nearest-neighbor classification. *The Annals of Statistics* 36, 5 (2008), 2135–2152.
- [21] Haykin, S. *Redes neurais: princípios e prática*. Bookman Editora, 2007.
- [22] Hossain, M. F. M., Sarkar, M., and Ahmed, S. H. Quality of experience for video streaming: A contemporary survey. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2017 13th International* (2017), IEEE, pp. 80–84.

- [23] Hyndman, R. J. *Moving Averages*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 866–869.
- [24] Ide, H., and Kurita, T. Improvement of learning for cnn with relu activation by sparse regularization. In *2017 International Joint Conference on Neural Networks (IJCNN)* (2017), pp. 2684–2691.
- [25] Ioffe, S., and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (2015), pmlr, pp. 448–456.
- [26] James, G., Witten, D., Hastie, T., and Tibshirani, R. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014.
- [27] Juluri, P., Tamarapalli, V., and Medhi, D. Measurement of quality of experience of video-on-demand services: A survey. *IEEE Communications Surveys & Tutorials* 18, 1 (2016), 401–418.
- [28] Kandel, I., and Castelli, M. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT express* 6, 4 (2020), 312–315.
- [29] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
- [30] Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., and Inman, D. J. 1d convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing* 151 (2021), 107398.
- [31] Kiranyaz, S., Gastli, A., Ben-Brahim, L., Al-Emadi, N., and Gabbouj, M. Real-time fault detection and identification for mmc using 1-d convolutional neural networks. *IEEE Transactions on Industrial Electronics* 66, 11 (2019), 8760–8771.
- [32] Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. Self-normalizing neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Red Hook, NY, USA, 2017), NIPS’17, Curran Associates Inc., p. 972–981.
- [33] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Commun. ACM* 60, 6 (may 2017), 84–90.
- [34] Kua, J., Armitage, G., and Branch, P. A survey of rate adaptation techniques for dynamic adaptive streaming over http. *IEEE Communications Surveys & Tutorials* 19, 3 (2017), 1842–1866.
- [35] Kua, J., Armitage, G., and Branch, P. A survey of rate adaptation techniques for dynamic adaptive streaming over http. *IEEE Communications Surveys Tutorials* 19, 3 (2017), 1842–1866.

- [36] Kurose, J. F., and Ross, K. W. Redes de computadores e a internet. *Uma nova* (2013).
- [37] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural Computation* 1, 4 (1989), 541–551.
- [38] Li, B., Wang, Z., Liu, J., and Zhu, W. Two decades of internet video streaming: A retrospective view. *ACM transactions on multimedia computing, communications, and applications (TOMM)* 9, 1s (2013), 33.
- [39] Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., and Liu, H. Feature selection: A data perspective. *ACM Comput. Surv.* 50, 6 (Dec. 2017).
- [40] Li, Z., Kaafar, M. A., and Xie, G. Session throughput prediction for internet videos. *IEEE Communications Magazine* 54, 12 (2016), 152–157.
- [41] Li, Z., Zhu, X., Gahm, J., Pan, R., Hu, H., Begen, A. C., and Oran, D. Probe and adapt: Rate adaptation for http video streaming at scale. *IEEE Journal on Selected Areas in Communications* 32, 4 (April 2014), 719–733.
- [42] Liang, J., and Liu, R. Stacked denoising autoencoder and dropout together to prevent overfitting in deep neural network. In *2015 8th international congress on image and signal processing (CISP)* (2015), IEEE, pp. 697–701.
- [43] Liu, L., Wang, R., Guo, D., and Fan, X. Message dissemination for throughput optimization in storage-limited opportunistic underwater sensor networks. In *2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)* (June 2016), pp. 1–9.
- [44] Liu, Z., Xu, Z., Jin, J., Shen, Z., and Darrell, T. Dropout reduces underfitting. In *International Conference on Machine Learning* (2023), PMLR, pp. 22233–22248.
- [45] Mahjoub, S., Chrifi-Alaoui, L., Marhic, B., and Delahoche, L. Predicting energy consumption using lstm, multi-layer gru and drop-gru neural networks. *Sensors* 22, 11 (2022), 4062.
- [46] Mushtaq, M.-S., and Mellouk, A. 2 - methodologies for subjective video streaming qoe assessment. In *Quality of Experience Paradigm in Multimedia Services*, M.-S. Mushtaq and A. Mellouk, Eds. Elsevier, 2017, pp. 27 – 57.
- [47] NEUBOT. Neubot data analysis. Disponível em: <<http://streaming.polito.it/neubot/world.html>>. Acessado em: ago. 2019.
- [48] Pascanu, R., Gulcehre, C., Cho, K., and Bengio, Y. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026* (2013).

- [49] Peng, G. Cdn: Content distribution network. *arXiv preprint cs/0411069* (2004).
- [50] Petrangeli, S., Hooft, J. V. D., Wauters, T., and Turck, F. D. Quality of experience-centric management of adaptive video streaming services: Status and challenges. *ACM Trans. Multimedia Comput. Commun. Appl.* 14, 2s (May 2018), 31:1–31:29.
- [51] Pokhrel, J. *Intelligent quality of experience (QoE) analysis of network served multimedia and web contents*. PhD thesis, Institut National des Télécommunications, 2014.
- [52] Robbins, H., and Monro, S. A stochastic approximation method. *The annals of mathematical statistics* (1951), 400–407.
- [53] Schuster, M., and Paliwal, K. K. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing* 45, 11 (1997), 2673–2681.
- [54] Seufert, M., Egger, S., Slanina, M., Zinner, T., Hossfeld, T., and Tran-Gia, P. A survey on quality of experience of http adaptive streaming. *IEEE Communications Surveys & Tutorials* 17, 1 (2015), 469–492.
- [55] Shen, G., Tan, Q., Zhang, H., Zeng, P., and Xu, J. Deep learning with gated recurrent unit networks for financial sequence predictions. *Procedia computer science* 131 (2018), 895–903.
- [56] Siami-Namini, S., Tavakoli, N., and Namin, A. S. The performance of lstm and bilstm in forecasting time series. In *2019 IEEE International Conference on Big Data (Big Data)* (2019), IEEE, pp. 3285–3292.
- [57] Spiteri, K., Sitaraman, R., and Sparacio, D. From theory to practice: Improving bitrate adaptation in the dash reference player. *ACM Trans. Multimedia Comput. Commun. Appl.* 15, 2s (July 2019).
- [58] Spiteri, K., Urgaonkar, R., and Sitaraman, R. K. Bola: Near-optimal bitrate adaptation for online videos. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications* (April 2016), pp. 1–9.
- [59] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.
- [60] Sun, S., Cao, Z., Zhu, H., and Zhao, J. A survey of optimization methods from a machine learning perspective. *IEEE Transactions on Cybernetics* 50, 8 (2020), 3668–3681.
- [61] Wei, B., Song, H., Wang, S., Kanai, K., and Katto, J. Evaluation of throughput prediction for adaptive bitrate control using trace-based emulation. *IEEE Access* 7 (2019), 51346–51356.

-
- [62] Yin, X., Jindal, A., Sekar, V., and Sinopoli, B. A control-theoretic approach for dynamic adaptive video streaming over http. In *ACM SIGCOMM Computer Communication Review* (2015), vol. 45, ACM, pp. 325–338.
 - [63] Zhao, H., Xu, Z., and Cui, F. Generalized hesitant fuzzy harmonic mean operators and their applications in group decision making. *International Journal of Fuzzy Systems* 18, 4 (Aug 2016), 685–696.