Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Modificando algoritmos de adaptação do MPEG-DASH para utilização de informações extras providas por redes neurais

Thiago Rocha de Paiva

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientador
Prof. Dr. Jacir Luiz Bordim

Brasília
2019

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Modificando algoritmos de adaptação do MPEG-DASH para utilização de informações extras providas por redes neurais

Thiago Rocha de Paiva

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof. Dr. Jacir Luiz Bordim (Orientador)
Universidade de Brasília

Prof. Dr. Jacir Luiz Bordim
Universidade de Brasília

Prof. Luis Alberto Belem Pacheco
Universidade de Brasília

Prof. Dr. Marcus Vinícius Lamar
Universidade de Brasília

Prof. Dr. José Edil Guimarães de Medeiros
Coordenador do Curso de Engenharia da Computação

Brasília, 08 de julho de 2019

Dedicatória

Dedico esse trabalho ao meu irmão, que em vários momentos me ouviu e importunou da maneira certa.

Agradecimentos

Chegado o momento de escrever os agradecimentos e percebo que há um grande número de pessoas a agradecer desde o início deste trabalho, e não farei um bom trabalho aqui porque não há como citar todos.

Primeiramente devo agradecer àqueles com que divido a maior parte da minha vida, meus pais e irmão. Eles que tiveram principalmente a paciência e a calma até aqui para me guiar um pouco no caminho, seja por meio de puxões e empurrões ou só na espera do meu ritmo para seguir em frente.

Agradeço aos professores e colegas de UnB, em especial o orientador Jacir Bordim, que me apoiou e incentivou independente do dia, horário ou espaço de tempo em que nos vimos. Também quero deixar explícito o agradecimento ao professor Marcos Caetano pelas discussões e incentivos em encontros ocasionais pelo departamento.

Minha gratidão se estende à banca, formada pelo orientador, e também os professores Marcus Lamar e Luis Pacheco, que dispuseram o tempo para ler, avaliar, e prover comentários que certamente contribuíram para a melhora do documento.

E, ao fim, o meu carinho aos amigos que um dia sentaram, escutaram e/ou falaram sobre suas próprias dificuldades quando viram as minhas. Em especial: João Fellipe, Estevão, Luis, Daniella, Bruno, Theus, e Matheus da escalada.

Resumo

O *MPEG Dynamic Adaptive Streaming over HTTP* (MPEG-DASH) surge da necessidade de especificar um formato aberto e flexível de transmissão de vídeos que faça uso das pilhas de protocolos e infra-estruturas já existentes da *web*, e assim ser capaz de ser utilizado independente de plataformas proprietárias. Porém um fator que não é padronizado no MPEG-DASH é como ocorrem as decisões dos programas clientes sobre quais qualidades requerer, e a partir disso há o desafio de implementar um algoritmo de adaptação que equilibre métricas e obtenha uma melhor experiência para o usuário. Como forma de incrementar algoritmos já existentes, neste trabalho foi feito um estudo por meio de simulações com o ns-3 da modificação de algoritmos para que utilizem informações de tamanhos dos segmentos de vídeos geradas por meio de redes neurais. As redes neurais foram escolhidas por serem capazes de abstrair, a partir de dados de tamanhos já existentes, tamanhos de segmentos em outras taxas de codificação de vídeo. Os resultados mostraram que, apesar de não haver clara vantagem no uso das redes neurais para um escopo de vídeos limitado, há espaço para melhoras na abstração dos dados, e que os algoritmos FESTIVE e o PANDA tem uma divisão da banda entre múltiplos clientes mais justa.

Palavras-chave: MPEG-DASH, algoritmos de adaptação, redes neurais

Abstract

The *MPEG Dynamic Adaptive Streaming over HTTP* (MPEG-DASH) emerge from the necessity of speficifying an open and flexible video transmission format which makes use of the already existing protocol stack and associated web infrastructure, and thus it is capable of running regardless of proprietary platforms. Despite that, a undocumented factor in MPEG-DASH is the decision process of client software regarding which quality to choose, and from that arises the challenge of implementing an adaptation algorithm which balances metrics and seeks a better user experience. As a way to improve existing algorithms, in this work a study was done through simulations in ns-3 about changes in algorithms in order to harness knowledge over segment sizes through neural networks. Neural networks were chosen due to being capable of abstracting segment sizes in other bitrates from existent data. The results have shown that, despite not having a clear advantagen when using neural networks faced with a limited video scope, there is room for improvement in data abstraction and that FESTIVE and PANDA have show more fairness in the bandwidth utilization.

Keywords: MPEG-DASH, adaptation algorithms, neural networks

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

3GPP *Third Generation Partnership Project.*

AAC *Advanced Audio Encoding.*

Adam *Adaptative Moment Estimation.*

AHS *Adaptive HTTP Streaming.*

CNAME *Canonical Name.*

CPU *Central Processing Unit.*

F4F *F4V Fragment.*

FESTIVE *Fair, Efficient, and Stable adaptive Algorithm.*

HAS *HTTP Adaptive Streaming.*

HDS *HTTP Dynamic Streaming.*

HLS *HTTP Live Streaming.*

HTTP *Hypertext Transfer Protocol.*

IETF *Internet Engineering Task Force.*

IIS *Internet Information Services.*

IP *Internet Protocol.*

ISO *International Organization for Standardization.*

ISOBMFF *ISO Base Media File Format.*

LSTM *Long Short-Term Memory.*

M3U8 *Extended M3U.*

MLP *Multilayer Perceptron.*

MP3 *MPEG-2 Audio Layer 3.*

MPD *Media Presentation Description.*

MPEG *Moving Picture Experts Group.*

MPEG-DASH *MPEG Dynamic Adaptive Streaming over HTTP.*

MPEG-TS *MPEG Transport Stream.*

MSE *Mean Square Error.*

Nadam *Nesterov-accelerated Adaptive Moment Estimation.*

NAG *Nesterov's Accelerated Gradient.*

PANDA *Probe And Adapt.*

QoE *Quality of Experience.*

RMSE *Root Mean Square Error.*

RNA Rede Neural Artificial.

RNRE Rede Neural Recorrente de Elman.

rRMSE *relative Root Mean Square Error.*

RTCP *RTP Control Protocol.*

RTMP *Real-time Messaging Protocol.*

RTP *Real-time Transport Protocol.*

RTSP *Real-time Streaming Protocol.*

SGD *Stochastic Gradient Descent.*

TCP *Transmission Control Protocol.*

TOBASCO *Threshold-Based Adaptation Scheme for On-demand streaming.*

UDP *User Datagram Protocol.*

URL *Uniform Resource Locator.*

UTF-8 *Unicode Transformation Format - 8-bit.*

VBPM *Video Bitrate Prediction Model.*

WMA *Windows Media Audio.*

WWDC *Worldwide Developers Conference.*

XML *Extensible Markup Language.*

Capítulo 1

Introdução

A entrega de conteúdo multimídia por meio da Internet vem crescendo nos últimos anos, em grande parte por avanços nas tecnologias de rede, capacidades dos dispositivos, e técnicas de compressão. Segundo pesquisas da Cisco [8], em 2016, 67% do tráfego total da Internet vinha do consumo de vídeo, e até 2021 esse número chegará aos 80%, com serviços como Netflix e Youtube contabilizando cerca de 26% do tráfego, sendo que a Netflix chegou a registrar picos de 40% [9]. Como os protocolos que formam a base da Internet foram projetados para transmissões de dados inconstantes e discretas, com transmissões de pequenos arquivos por vez, esse cenário apresenta alguns desafios, já que, desde 2007, com o surgimento do *streaming* sobre o HTTP, esta tem sido a maneira escolhida pelos distribuidores de conteúdo [10].

Apesar da inconstância com que o HTTP *streaming* tem que lidar, com riscos de perdas e atrasos de pacotes que são críticos a uma aplicação que depende do tempo e ordem de recebimento como o vídeo, a escolha se dá pelo custo da infraestrutura mantida pelos distribuidores, já que com o uso do HTTP e do TCP, programas clientes apenas tem de fazer requisições para servidores *web* comuns. Essa técnica também adquiriu popularidade por meio de soluções comerciais como o *Smooth Streaming*, o *HTTP Live Streaming* (HLS) [11], e o *HTTP Dynamic Streaming* (HDS) [12]. Para evitar fragmentação de mercado, o *Moving Picture Experts Group* (MPEG) junto ao *Third Generation Partnership Project* (3GPP) protagonizou o desenvolvimento do padrão aberto a ser discutido e utilizado neste trabalho, o *MPEG Dynamic Adaptive Streaming over HTTP* (MPEG-DASH) [13].

O MPEG-DASH define principalmente as possíveis funcionalidades a serem suportadas por um servidor, e como deve ser a estrutura para que haja o funcionamento das sessões de transmissões de vídeo. Um dos aspectos mais importantes do padrão é o fato de que um arquivo de mídia deve ser preparado antes de ser disponibilizado, sendo dividido em diversos segmentos com durações de tempo de reprodução iguais. Múltiplas versões dos segmentos de um vídeo são geradas com variações em taxa de codificação e resolução.

Além disso, o servidor gera um arquivo índice para que o cliente possa identificar os segmentos através de URLs únicas. O MPEG-DASH porém não diz como um programa cliente deve escolher qual qualidade a ser requerida entre suas requisições discretas, essa tarefa fica a cargo da implementação de um algoritmo de adaptação.

Soluções de HTTP *streaming* empregam a escolha de quais requisições fazer de forma dinâmica de acordo com variações das condições da rede, de modo a prover uma experiência mais agradável quanto possível. Durante uma sessão o cliente mede frequentemente certos parâmetros como banda disponível, ocupação de *buffer*, bateria, utilização de CPU, e outras. Essas medidas são utilizadas para decidir qual o próximo segmento a ser requerido entre os disponíveis no servidor e também podem servir como métricas para avaliação do desempenho do algoritmo.

1.1 Objetivo

Alguns trabalhos utilizaram de técnicas de inteligência artificial para otimizar o processo de decisão dos algoritmos de adaptação [14][15], ou mesmo na avaliação de métricas de QoE [16][17]. Dentre as técnicas de inteligência artificial, uma que tem se destacado bastante na área é o uso de redes neurais para resolver problemas não-lineares. Uma Rede Neural Artificial (RNA) possui baixa complexidade de implementação, por ser construídas a partir de camadas de neurônios e conexões entre estes e as diferentes maneiras com que pode ser organizada e obter valores abstraídos a partir desses componentes.

O objetivo inicial do trabalho foi de utilizar uma rede neural para inferir informações a partir de um próximo segmento de menor qualidade quanto possível e variações de banda. De posse dessas informações a rede poderia tomar decisão de qual qualidade seria a melhor a ser utilizada a partir de um ponto na reprodução do vídeo.

Porém este se provou um trabalho extenso, e, como parte do caminho até obter essa conclusão, começou-se com redes neurais capazes de estimar uma qualidade com base em outra, para eventualmente testar modelos de redes neurais que forneça informações extras dos tamanhos dos segmentos a algoritmos de adaptação existentes. Objetivos específicos incluem a seleção de parâmetros que formem um modelo de uma rede neural, o treinamento das RNAs de forma a abstrair os tamanhos dos segmentos, e a simulação e comparação entre algoritmos de adaptação com e sem o conhecimento dos tamanhos providos pelas etapas anteriores.

1.2 Metodologia

Expandindo os objetivos específicos citados acima, a metodologia utilizada para as redes neurais em grande parte do trabalho foi empírica, devido à não existência de método formal para a seleção dos hiper-parâmetros que formam o modelo de uma RNA. Porém como modo de obter resultados entre diferentes modelos, utiliza-se aqui um método de validação de 10 dobras conhecido dentro da estatística, a ser explicado na Seção 4.3.

Na etapa do treinamento das RNAs, buscou-se expandir o número de iterações com o modelo selecionado anteriormente, de modo a obter maior equivalência aos tamanhos de segmentos originais. Aqui o caso ótimo para a rede neural seria ter resultados exatos, iguais aos tamanhos de segmentos originais utilizados durante o trabalho, porém como o objetivo inicial era dar um passo além, não utilizou-se de arquivos tabela com os tamanhos originais para ver se as RNAs pareciam viáveis.

Como última etapa, realizaram-se simulações com algoritmos de adaptação já existentes. Os algoritmos selecionados buscaram abranger três modos de decisões diferentes: distribuídos entre os baseados em taxa de transmissão, em ocupação de *buffer*, e os híbridos. Com os registros das diversas simulações extraiu-se métricas definidas de qualidade de serviço, e, a partir das métricas, a conclusão.

1.3 Estrutura do documento

Este documento é composto por 6 capítulos, incluindo este. O Capítulo 2 detalha como as redes neurais geram seus resultados. O Capítulo 3 apresenta o surgimento do *streaming* de vídeo, com protocolos e o estado da arte envolvido. O Capítulo 4 enumera os passos para o teste e comparação dos algoritmos de adaptação modificados. O Capítulo 5 expõe os dados obtidos pelos experimentos realizados. O Capítulo 6 traz as conclusões a partir da análise dos resultados, apontando possíveis espaços de melhorias.

Capítulo 2

Redes Neurais

As redes neurais fazem parte de um conjunto de ferramentas desenvolvidas na área de inteligência artificial. A inteligência artificial teve seu início como área de estudo após a Segunda Guerra Mundial e abrange uma série de campos e aplicações, desde jogos e sistemas financeiros ao auxílio na detecção de doenças [18]. Dentro dela estão todos os algoritmos que buscam se aproximar do pensar humano, geralmente criados para problemas específicos. As redes neurais se destacam pela generalidade em seu uso para resolver problemas não-lineares (que não podem ser modelados apenas com equações lineares) e pela baixa complexidade para implementação.

Na Seção 2.1 é apresentada a origem das redes neurais e seu conceito de forma geral. A Seção 2.2 descreve um neurônio e suas sinapses, e o modelo matemático que rege os cálculos da rede. A Seção 2.3 discorre sobre as funções de ativação que delimitam a saída de um neurônio. Já na Seção 2.4, é apresentado uma classificação para o aprendizado de uma RNA, e depois diferentes algoritmos para otimizar ajuste de pesos associados às sinapses. Por último, a Seção 2.5 provê um resumo sobre os diferentes modos como uma rede pode ser organizada.

2.1 Introdução

A Rede Neural Artificial (RNA) surge como objetivo comum entre pesquisadores de várias áreas, como química, física, biologia e computação, para aprender a partir de exemplos e chegar a respostas. A tecnologia foi concebida por meio do desejo de emular o aprendizado de um cérebro ao formular matematicamente as menores unidades desse, um neurônio e suas conexões. Segundo Braga [19], uma RNA é constituída por unidades paralelas de processamento simples em um sistema que realiza cálculo de uma função matemática, usualmente não-linear.

Uma rede tem seus neurônios distribuídos em diferentes camadas, que podem ser ditas como camadas de entrada, de saída, ou escondidas. As camadas de entrada e saída lidam com os dados que definem o escopo do problema a ser resolvido, e possuem quantidades de neurônio equivalente ao tamanho destes dados. Já as camadas escondidas ficam entre a entrada e a saída, expandindo a capacidade de lidar com problemas de maior complexidade, e podem conter uma quantidade livre de camadas e neurônios.

Pesos são associados às conexões entre neurônios e por meio delas é possível fazer ajustes em um processo de aprendizado, e guardar um conhecimento que pode ser aplicado para classificação, ajuste de dados, reconhecimento de padrões, e predições de séries temporais. Pela maneira em que os neurônios e suas conexões são distribuídos e os processamentos das informações feitos, as redes neurais encontram aplicação de natureza não-linear em reconhecimento de escrita e fala, e até mesmo modelagem de próteses. Tudo isso de forma flexível, ajustando valores associados aos diversos neurônios de uma rede, tentando generalizar em cima de exemplos dados e minimizando os erros.

Na próxima seção será descrito o modelo de um neurônio.

2.2 Neurônio

O neurônio artificial foi primeiro proposto como um modelo matemático por McCulloch e Pitts [20], baseado no conhecimento simplificado sobre a biologia de sistemas neurológicos existente na época. Segundo Haykin [21], um neurônio artificial j, adjacente a um neurônio i, pode ser definido em três partes fundamentais:

- Um conjunto de conexões entre os neurônios, chamado de sinapses. Cada sinapse individual possui um peso w_{ij} a ser aplicado sobre um sinal x_{ij} que o neurônio recebe.
- Um potencial de ativação v_j que é o somatório dos sinais ponderados nas sinapses mais um *bias* b_j; este servindo de ajuste para aumentar ou diminuir o potencial de ativação.
- Uma função de ativação ϕ que tem o papel de limitar a amplitude da saída do neurônio a determinado intervalo, tipicamente utilizando valores dos intervalos [0, 1] ou [-1, 1].

Desta forma, um neurônio com n sinapses, ou seja, que é adjacente a n neurônios de uma camada anterior, como ilustrado na Figura 2.1, pode ter sua saída y_j calculada por

$$y_j = \phi(v_j), \tag{2.1}$$

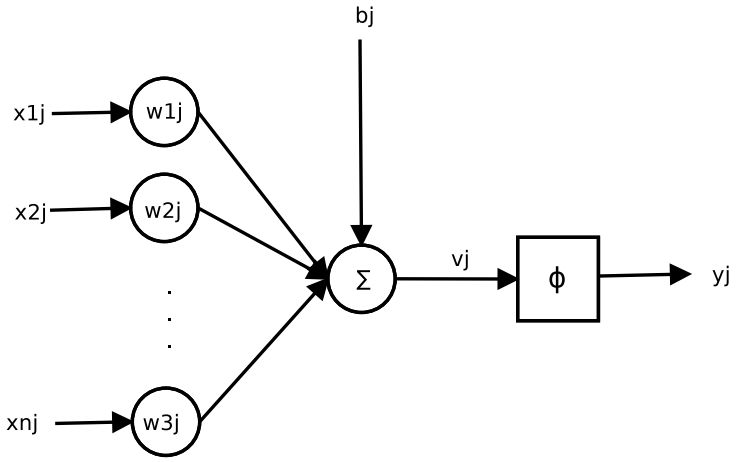


Figura 2.1: Modelo de neurônio.

que pode ser expandida para

$$y_j = \phi(\sum_{i=1}^{n} w_{ij}x_{ij} + b_j). \tag{2.2}$$

2.3 Funções de ativação

As funções de ativação podem também ser chamadas de funções de limiar ou de transferência de sinais, e servem para determinar a intensidade da saída da informação de um neurônio dentro de um determinado intervalo. As principais funções serão descritas a seguir:

Linear É exemplificada pela Figura 2.2 e descrita matematicamente por

$$\phi(v_j) = av_j, \tag{2.3}$$

sendo a um termo de ajuste da inclinação da reta e v_j o sinal de ativação de um neurônio.

Degrau Também chamada popularmente no inglês de *threshold*, *step*, ou Modelo de McCulloch-Pitts. É exemplificada pela Figura 2.3 e descrita matematicamente por

$$\phi(v_j) = \begin{cases} 0, & se\ v_j < 0, \\ 1, & se\ v_j \geq 0. \end{cases} \tag{2.4}$$

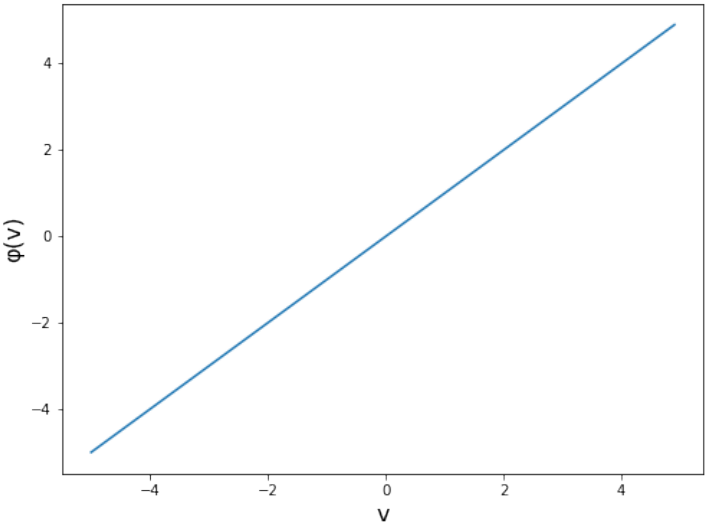


Figura 2.2: Função de ativação linear.

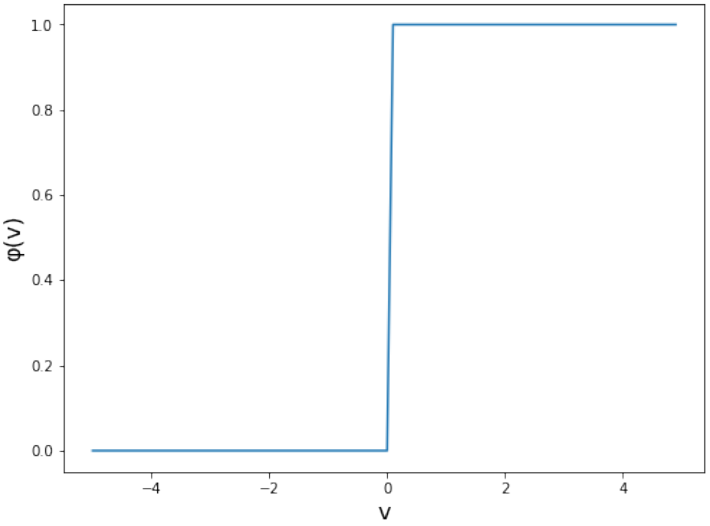


Figura 2.3: Função de ativação degrau.

Em rampa Essa função permite a delimitação de uma região de transição no limiar de saída do neurônio. É exemplificada pela Figura 2.4 e descrita matematicamente por

$$\phi(v_j) = \begin{cases} -1, & se\ v_j < -1, \\ v_j, & se\ -1 \leq v_j \leq 1, \\ 1, & se\ v_j > 1. \end{cases} \tag{2.5}$$

Sigmoide Essa função garante transições mais graduais e detalhadas, sem conter descontinuidades, e por isso é a função de ativação mais utilizada. É exemplificada

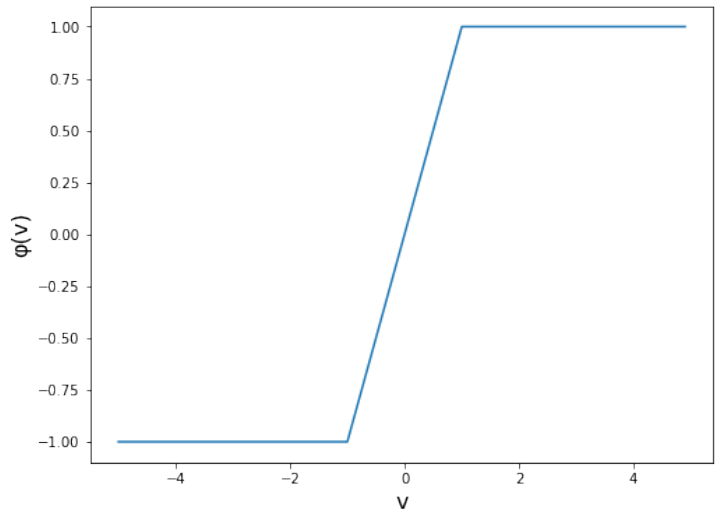


Figura 2.4: Função de ativação rampa.

pela Figura 2.5, contendo em sua descrição matemática o parâmetro a para ajustar a inclinação da curva, resultando na equação

$$\phi(v_j) = \frac{1}{1 + e^{-av_j}}. \tag{2.6}$$

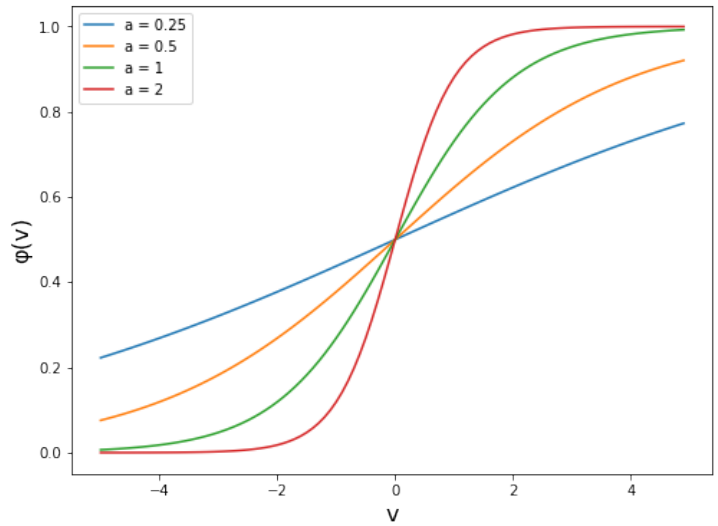


Figura 2.5: Função de ativação sigmoide.

Sigmoide Bipolar Possui as mesmas propriedades que a função de ativação sigmoide, porém é utilizada quando se deseja um resultado dentro de um limiar bipolar (in-

tervalo negativo e positivo), e por isso possui esse nome. É exemplificada pela Figura 2.6 e descrita matematicamente por

$$\phi(v_j) = \frac{1 - e^{-v_j}}{1 + e^{-v_j}}. \tag{2.7}$$

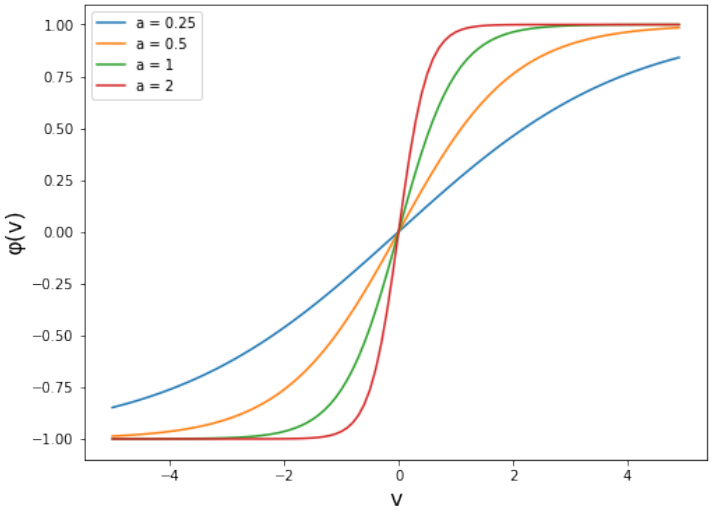


Figura 2.6: Função de ativação sigmoide bipolar.

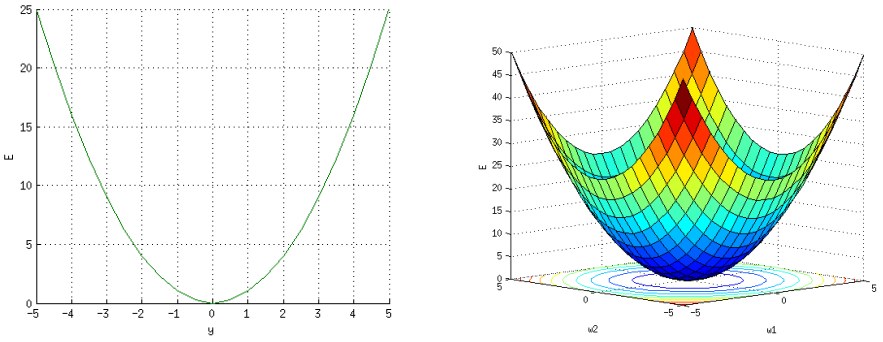
2.4 Aprendizado

Segundo Haykin [21], aprendizado é um processo pelo qual parâmetros de uma Rede Neural Artificial são adaptados por meio de um processo de simulação do ambiente no qual a rede está inserida. Como já dito anteriormente, esse processo de adaptação dos parâmetros é iterativo, e uma RNA utiliza o conhecimento obtido para realizar generalizações e interpolações sobre os dados apresentados.

A seguir serão apresentadas as diferentes classificações gerais dos tipos de aprendizados:

Aprendizado não-supervisionado A rede recebe apenas dados de entrada, e a partir destes busca padrões para estabelecer agrupamentos.

Aprendizado supervisionado Neste tipo de aprendizado, é fornecido à rede um conjunto de dados com entradas e saídas desejadas, se busca uma maneira de correlacionar estes pares de entrada-saída. Para isso, a cada iteração do treinamento da rede, serão feitos ajustes na rede depois de comparar a saída obtida a partir daquele estado da rede com a saída desejada.



(a) Curva de erro (b) Superfície de erro

Figura 2.7: Curva e superfície da função de erro (Fonte: [1])

O aprendizado supervisionado pode ser *offline*, em que o treinamento só é feito com aqueles dados iniciais e após isso a rede não se altera; ou *online*, em que a rede realiza seu treinamento de forma contínua, em tempo de execução, a cada novo dado ofertado.

Aprendizado por reforço É um caso particular do aprendizado supervisionado. Neste aprendizado um supervisor indica para um conjunto de saídas desejadas se a saída obtida da rede está correta ou não.

Na próxima seção são apresentados os algoritmos para o aprendizado supervisionado que buscam garantir a correlação entre os resultados obtidos e os dados de entrada e saída que são fornecidos a uma RNA.

2.4.1 Algoritmo de otimização

Um treinamento de rede neural é feito por meio de uma série de passos que visa ajustar os pesos das sinapses iterativamente, chamada de algoritmo de otimização. Esses algoritmos são assim chamados pois são utilizados para otimizar a busca por um mínimo de uma curva ou superfície formada pela função de erro, como exemplificadas pelas Figura 2.7a e Figura 2.7b. Tais funções de erro servem como base para a otimização de uma rede, sendo que em geral a função escolhida é a função de erro quadrática (também conhecida como função *Mean Square Error* (MSE)), como na Equação 2.8 abaixo, onde t é a saída desejada, y a obtida, e a fração é apenas uma constante para facilitar na derivação

$$E = \frac{1}{2}(t - y)^2. \tag{2.8}$$

O algoritmo mais conhecido, e que é utilizado como base para muitos outros mais atuais, é chamado de gradiente descendente. Ele é caracterizado por duas fases de execução: a propagação e a atualização das sinapses.

Na fase de propagação, primeiro são gerados pesos de forma aleatória para que o sinal de entrada possa ser transmitido pela rede, camada a camada, e gerar uma saída. Logo após surge uma etapa essencial nesta fase, a obtenção dos gradientes de erro nos pesos das sinapses dos neurônios. Como esses gradientes são obtidos será descrito a seguir, na retro-propagação de erros.

A fase de atualização das sinapses é tomada pelo ajuste do peso w_{ij} a partir da soma de um termo Δw_{ij}, como descrito na equação

$$w_{ij} = w_{ij} + \Delta w_{ij}. \tag{2.9}$$

O delta que representa o quanto o peso será atualizado, é definido pelo ajuste do gradiente do erro ∇E por meio de uma taxa de aprendizagem η, como exemplificado por

$$\Delta w_{ij} = -\eta \nabla E. \tag{2.10}$$

A negativação do termo $\eta \nabla E$ é feita para que o ponto da superfície de erro a ser utilizado vá em direção oposta a indicada pelo gradiente, e o nome ao gradiente descendente é dado por esse motivo.

Retro-propagação de erros

A retro-propagação de erros (do Inglês, *error backpropagation*) é como serão obtidos os gradientes de erro da RNA. Foi um método inicialmente proposto por Werbos [22] em 1974, mas teve amplo uso no contexto de redes neurais a partir de 1986, devido ao trabalho de Rumelhart *et al.* [23]. A retro-propagação tem esse nome pois os gradientes dos pesos das sinapses, usados para indicar a inclinação de uma curva de erro, são calculados a começar pela saída da rede e os resultados de cada cálculo propagam até os primeiros neurônios da rede.

Para cada neurônio j é preciso resolver um gradiente de cada entrada advinda de um neurônio i, e, por consequência, as derivadas obtidas a partir da regra da cadeia, como descritas em

$$\nabla E = \frac{\delta E}{\delta w_{ij}} = \frac{\delta E}{\delta y_j} \frac{\delta y_j}{\delta v_j} \frac{\delta v_j}{\delta w_{ij}}. \tag{2.11}$$

A partir da expansão das derivas parciais, o primeiro termo, a derivação da função de erro em relação a saída do neurônio $\frac{\delta E}{\delta y_j}$, somente poderá ser obtido tendo resolvido as derivações dos neurônios adjacentes a frente, ou seja, os neurônios vizinhos que se encon-

tram mais próximos a saída da rede. Importante notar que para isso, tanto a função de erro (E, exemplificada em Equação 2.8) quanto a função de ativação (y_j, exemplificada em Equação 2.1) devem ser deriváveis.

Nadam

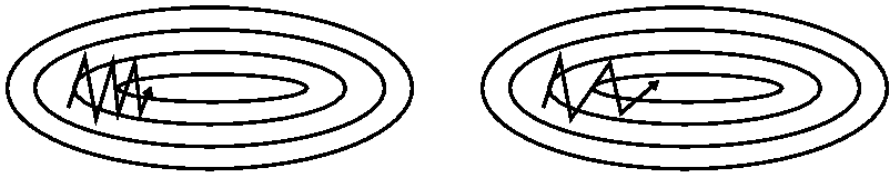
O gradiente descendente pode ser um dos mais populares algoritmos para fazer otimizações, mas está longe de ser o único. Diversas bibliotecas de redes neurais, como a Lasagne [24], a Caffe [25], e, a utilizada neste trabalho, Keras [26], implementam outros algoritmos que expandem a capacidade de treinamento de uma rede neural com algoritmos de otimização mais recentes. Nesta seção iremos discutir o *Nesterov-accelerated Adaptive Moment Estimation* (Nadam), junto com algumas propostas que o precederam e ajudaram a compô-lo, como o *Adaptative Moment Estimation* (Adam).

Originalmente, o gradiente descendente possui uma convergência lenta para o erro mínimo, o que ocorre devido a ele ter que iterar sobre todos os valores passados como dados de entrada na etapa de propagação antes de atualizar os pesos. Tal prática consome tempo e memória, tendo sido o primeiro alvo para as novas propostas. Visando melhorar isso, surge o *Stochastic Gradient Descent* (SGD) e o gradiente em mini-lotes, sendo que o primeiro atualizará sinapses a cada entrada individual e o segundo que poderá atualizar sinapses em uma quantidade configurável de valores de entrada. Na prática, o algoritmo por mini-lote também é chamado de SGD.

O SGD porém tem problemas com partes da função de erro em que há uma grande variação, como em áreas da curva de erro com grandes depressões, pois o gradiente adquire valores de módulo significativo, porém que ficam sucessivamente indicando direções opostas; ou em regiões que formam planaltos, pois o gradiente tem seu módulo com valor pequeno demais para fazer progresso suficiente. Para resolver essa questão, Qian [27] adapta a ideia de momento aos algoritmos de otimização. Assim como na física, a ideia do momento é demonstrar o acúmulo de movimento, nesse caso do gradiente em uma certa direção, tal que aumente o gradiente caso este continue no mesmo sentido, ou diminua caso vá mudar, diminuindo oscilações; como mostra a Figura 2.8. Isso é feito por meio da inclusão de uma dependência do gradiente de um tempo anterior, assim como demonstrado em

$$\Delta w_{ij}(t) = -(\alpha \Delta w_{ij}(t-1) + \eta \nabla E). \tag{2.12}$$

A partir do conceito de momento, podemos melhorar ainda mais a velocidade de um treinamento ao utilizar o *Nesterov's Accelerated Gradient* (NAG), definido em [28]. O NAG decorre da observação de que $w_{ij}(t) - \alpha \Delta w_{ij}(t-1)$ serve como boa estimativa para os próximos pesos, aqui representados por $w_{ij}(t+1)$, e que o termo $\Delta w_{ij}(t-1)$,



(a) SGD sem momento (b) SGD com momento

Figura 2.8: Efeito do momento no SGD (Fonte: [2])

da Equação 2.12, não depende do gradiente atual. Tais observações são estendidas ao cálculo do gradiente atual; isto é, em vez do gradiente ser obtido em relação aos pesos, ele é obtido por meio da estimativa dos próximos pesos fazendo uso do momento, resultando na equação

$$\Delta w_{ij}(t) = -(\alpha \Delta w_{ij}(t-1) + \eta \nabla E(w_{ij}(t+1)). \tag{2.13}$$

Outra parte fundamental e bem utilizada que antecede a proposta do Nadam é o Adam, porém para explicá-lo, deve-se antes falar do RMSprop. O RMSprop é um método adaptativo, proposto por Geoff Hinton durante o ensino de suas aulas [29], que altera a taxa de aprendizagem para que dados menos frequentes possuam maior peso no treino. Tomando um novo termo como gradiente por brevidade,

$$g_t = \nabla E(w_{ij}(t)), \tag{2.14}$$

a proposta é concretizada por meio de ajustes na taxa de aprendizagem com um denominador que depende da média das variâncias dos gradientes de erro passados, o que faz com que dados que variem mais em relação a média sejam menos significativos. A média das variâncias é aqui representada por $M(g_t^2)$, como pode ser observado em

$$\Delta w_{ij}(t) = -\frac{\eta}{\sqrt{M(g_t^2) + \epsilon}} g_t. \tag{2.15}$$

O termo das médias de variâncias decai com o passar das iterações ao ter seus valores passados ajustados pelo termo de decaimento *gamma*, o que resulta na equação

$$M(g_t^2) = \gamma M(g_{t-1}^2) + (1-\gamma) g_t^2. \tag{2.16}$$

Já o *Adaptative Moment Estimation* (Adam), publicado por Kingma *et al.* [30], é um algoritmo que, junto a ideia do RMSprop, traz também o momento através de uma média decrescente dos gradientes passados, representada por $M(g_t)$. Logo, o algoritmo tem duas médias como sua base para atualizar os pesos duas médias: a $M(g_t)$ e a $M(g_t^2)$. Essas

médias, chamadas de primeiro momento e segundo momento, dão o nome ao Adam, e podem ser observadas com seus respectivos termos de decaimento em

$$\begin{aligned} M(g_t) &= \beta_1 M(g_{t-1}) + (1-\beta_1) g_t \\ M(g_t^2) &= \beta_2 M(g_{t-1}^2) + (1-\beta_2) g_t^2. \end{aligned} \tag{2.17}$$

Com a adição dos dois termos a modificação do peso das sinapses fica descrita por

$$\Delta w_{ij}(t) = -\frac{\eta}{\sqrt{M(g_t^2) + \epsilon}} M(g_t). \tag{2.18}$$

Como as médias são inicializadas como vetores de zeros, os autores também fazem uma correção delas ao assumir os termos de decaimento com valores grandes (como 0.99) e aplicando à Equação 2.17. As correções resultam nos termos $\hat{M}(g_t)$ e a $\hat{M}(g_t^2)$, como descritos em

$$\begin{aligned} \hat{M}(g_t) &= \frac{M(g_{t-1})}{1-\beta_1} \\ \hat{M}(g_t^2) &= \frac{M(g_{t-1}^2)}{1-\beta_2}, \end{aligned} \tag{2.19}$$

e resultam na equação final do Adam:

$$\Delta w_{ij}(t) = -\frac{\eta}{\sqrt{\hat{M}(g_t^2) + \epsilon}} \hat{M}(g_t). \tag{2.20}$$

Por fim, o último trabalho referenciado é o do algoritmo de otimização *Nesterov-accelerated Adaptive Moment Estimation* (Nadam), definido por Dozat [31], que aplica a ideia do NAG ao Adam. Para alcançar isso de forma mais otimizada, Dozat propõe que o delta seja atualizado, não em relação ao momento passado (de $t-1$), mas sim utilizando o valor do momento t atual. Logo, expandindo os termos da Equação 2.20 e adotando as médias de t no lugar de $t-1$, obtem-se:

$$\Delta w_{ij}(t) = -\frac{\eta}{\sqrt{\hat{M}(g_t^2) + \epsilon}} \left(\beta_1 \hat{M}(g_t) + \frac{(1-\beta_1) g_t}{1-\beta_1} \right), \tag{2.21}$$

que herda as características de pouco uso de memória do Adam, mas encontra o mínimo da função de erro mais rápido para a maior parte dos casos, segundo o autor.

2.5 Arquiteturas

Outro fator que influencia diretamente no aprendizado de uma RNA é a sua estrutura. A estrutura, ou arquitetura, dita o modo como os neurônios e suas sinapses estão dispostos

numa rede. Nesta seção serão discutidas três arquiteturas: a MLP, a Elman, e a LSTM.

2.5.1 Rede *perceptron* multicamadas

Este tipo de rede neural, além de camadas de entrada e saída em quantidade equivalente a quantidade de dados de entrada/saída, possui uma ou mais camadas escondidas. Por possuir múltiplas camadas, esse tipo de RNA é chamado de rede Perceptron Multicamada (do Inglês, *Multilayer Perceptron* (MLP)). Uma MLP é totalmente conectada a nível de camadas, ou seja, todos neurônios de uma camada estão conectados a todos neurônios de camadas vizinhas, como pode ser observado na Figura 2.9. O número de camadas escondidas depende da complexidade do problema, com uma camada já sendo o suficiente para funções contínuas, e duas bastando para qualquer tipo de função segundo Cybenko [32][33].

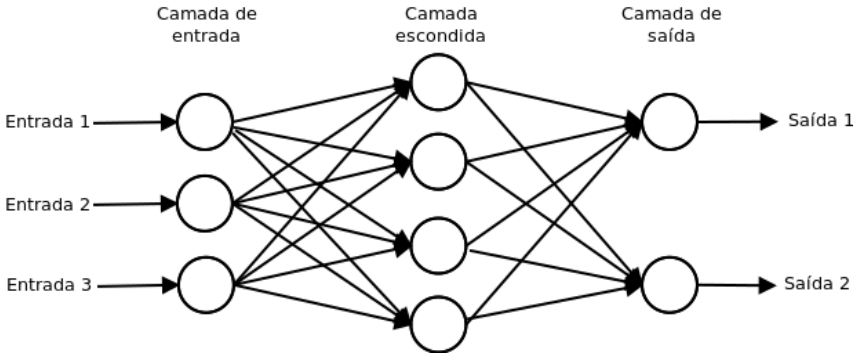


Figura 2.9: MLP com uma camada escondida.

2.5.2 Rede de Elman

Desenvolvida visando algoritmos de processamento de linguagem natural, Jeffrey Elman desenvolve a Rede Neural Recorrente de Elman (RNRE) [34]. Ao contrário de uma rede neural não-recorrente onde as entradas são assumidas como independentes entre si, uma rede recorrente tem como princípio o armazenamento de curto-termo de sinais propagados pela rede anteriormente.

Uma RNRE, como ilustrada pela Figura 2.10, é um tipo específico de rede recorrente onde camadas escondidas tem um retorno das suas saídas através de uma camada de contexto. As sinapses entre uma camada escondida e sua camada de contexto são passadas diretamente sem mudanças, com peso fixado como 1, tornando assim a camada de contexto uma memória do valor de saída de um neurônio. Já entre uma camada de contexto e uma escondida, há uma conexão com ajuste de pesos, assim como entre camadas de uma MLP.

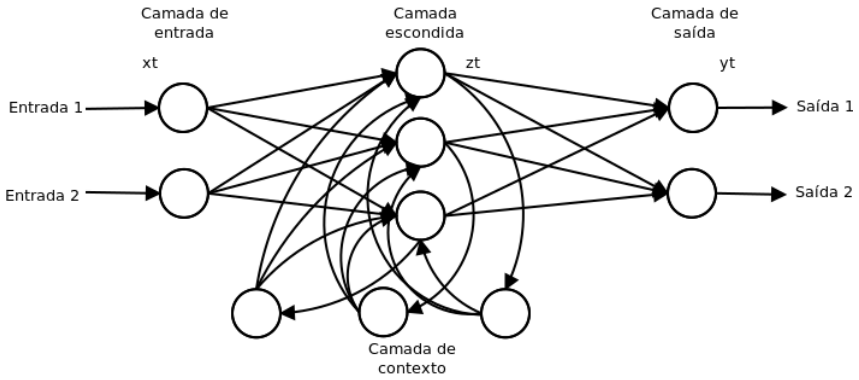


Figura 2.10: Estrutura de uma RNRE explicitando a camada de contexto.

A saída de um neurônio da camada escondida de uma Elman pode ser descrita por

$$z_t = \phi(wx_t + w_c z_{t-1} + b). \tag{2.22}$$

Aqui é importante atentar ao termo y_{t-1}, que é o valor armazenado pela camada de contexto da saída anterior de um neurônio da camada escondida, e é o que mantém a relevância de dados anteriores durante o treinamento da rede.

2.5.3 LSTM

Mesmo as RNREs têm suas limitações no aprendizado. Segundo Bengio *et al.* [35], redes recorrentes possuem problemas para representar dependências de maior significância devido ao decaimento dos pesos passados com os parâmetros do gradiente descendente. Dessa problemática surge a inspiração para a arquitetura utilizada aqui, a de redes *Long Short-Term Memory* (LSTM).

A arquitetura LSTM foi criada por Hochreiter e Schmidhuber em 1997 [36] para que uma rede pudesse recordar números mesmo com um sinal de entrada com bastante ruído. Desde então, redes LSTM encontraram diversas aplicações em tarefas como predição de estruturas de proteínas, reconhecimento de fala, e até geração de música.

Os neurônios da camada escondida aqui passam a ser chamados de blocos. Cada bloco de uma rede possui uma ou mais células de memória auto-conectadas e três unidades chamadas de portões (do Inglês, *gates*), cada qual com seu papel - de entrada, saída e esquecimento - provendo um controle de informações do bloco. Assim como exemplificado pela Figura 2.11, os três portões proporcionam uma soma das suas entradas com uma função de ativação aplicada a saída. Essas funções de ativação irão filtrar (multiplicar) as saídas dos portões para que cada um exerça uma função diferente sobre a célula do bloco: o portão de entrada checa se sua memória irá ser alterada pela entrada do bloco;

o portão de saída checa se a saída será propagada; e o portão de esquecimento checa se uma informação deve ser guardada ou esquecida.

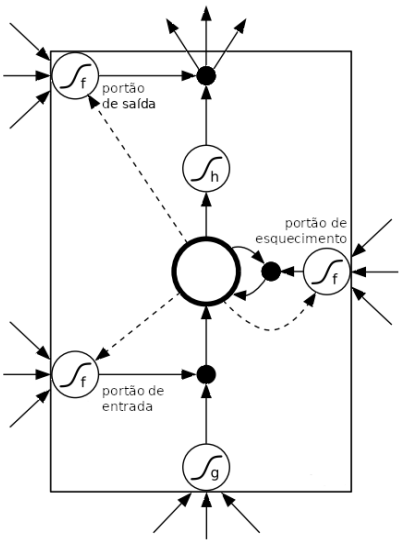


Figura 2.11: Bloco de uma rede LSTM com uma célula. Adaptado de [3].

O tipo de estrutura do LSTM comentado aqui chegou a ter modificações, mas ainda tem sido capaz de alcançar os melhores resultados segundo Greff [37]. Analogamente a outras redes recorrentes, os blocos são totalmente conectados entre camadas adjacentes e, além disso, possuem conexões de sua própria saída para seus próprios portões internos, como demonstrado pela Figura 2.12. As equações aqui adquirem maior extensão devido ao número de conexões, mas seguem a mesma lógica de aplicar pesos às conexões individuais e ter funções de ativação agindo nos portões, e podem ser vistas no Apêndice A.

Este capítulo tratou dos conceitos envolvidos em uma Rede Neural Artificial (RNA). A abstração de um neurônio foi brevemente apresentada junto às três características que formam sua descrição matemática. Também discorreu-se sobre as funções de ativação que controlam os limiares da saída de um neurônio. A partir dessa unidade básica de uma RNA e da função de ativação pode-se estudar o processo de aprendizagem, com sua classificação, base e incrementos de algoritmos de otimização. Por último, também influenciando na saída de uma rede neural, foi discorrido sobre os tipos de arquitetura.

No capítulo seguinte é discutido os conceitos e classificações de transmissão de vídeo pela Internet, além de também conter apresentações de padrões e protocolos do estado da arte.

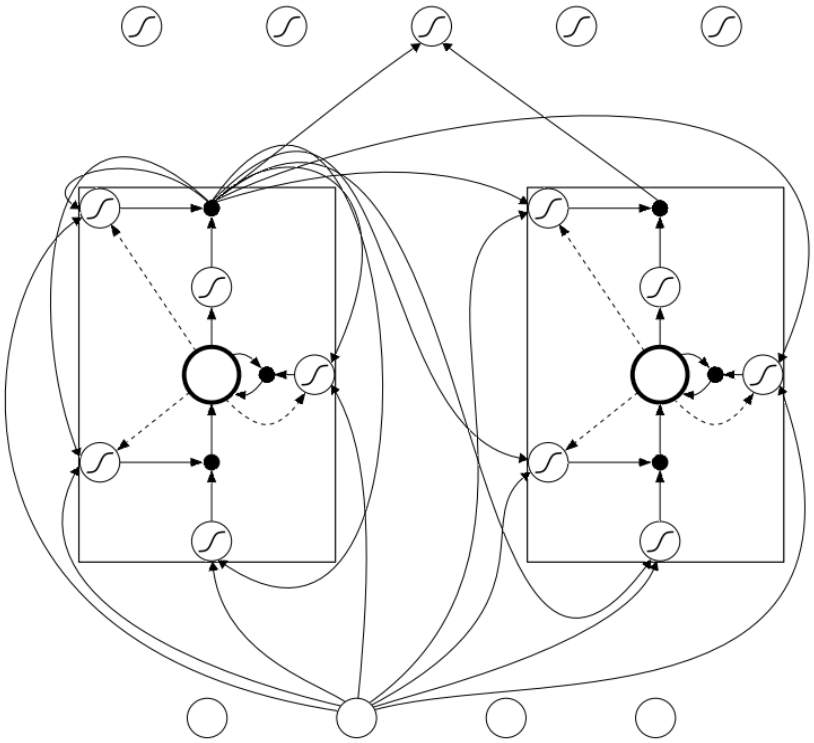


Figura 2.12: Rede LSTM exemplificando as conexões entre camadas adjacentes. Adaptado de [3].

Capítulo 3

Transmissão de vídeos pela Internet

Neste capítulo será discutido o processo da transmissão de vídeos através da rede, junto a uma classificação das possíveis maneiras disso ser feito, até chegar a apresentação dos principais protocolos voltados para o conteúdo em vídeo da Internet.

Na Seção 3.1 é introduzido o conceito de *streaming* e suas maneiras de utilização. A Seção 3.2 reporta a primeira técnica desenvolvida exclusivamente para transmissão multimídia. A Seção 3.3 revisa o HTTP e apresenta técnicas de transmissão que o utilizam. Na Seção 3.4 são elencadas as principais características de três padrões proprietários de *streaming*. E a Seção 3.5 detalha o MPEG-DASH, um padrão aberto para vídeo adaptativo. Por último, a Seção 3.7 expõe parte do estado da arte para o MPEG-DASH no lado do cliente.

3.1 *Streaming*

Um fluxo contínuo (*streaming*) de mídia é definido como uma transmissão que está acontecendo de algum conteúdo de mídia, e aqui "fazer o *stream*", ou *streaming* simplesmente, referirá à técnica utilizada para programas clientes poderem visualizar arquivos multimídia segundos após começar o recebimento vindo do servidor. Isso significa que o cliente estará reproduzindo parte de uma mídia, enquanto ainda é feito *download* de outra parte do arquivo que está mais a frente, também eliminando a necessidade de armazenamento local de toda a extensão de um conteúdo. Segundo Kurose e Ross [38], o termo pode ser aplicado para mídias além de áudio e vídeo, como legendas ou outras aplicações de texto em tempo real. O *streaming* é definido como um processo em tempo real, tal como ocorria continuamente nas televisões que reproduziam o sinal analógico a medida que esse ia sendo capturado e transmitido. Para que ocorra um processo fluido desta maneira, deve existir uma taxa de transmissão na conexão entre servidor e cliente maior ou igual à taxa de bits em que o conteúdo está codificado (também chamada de taxa de vídeo);

independente das variações na banda causadas por diferentes interferências. E, se interferências no sinal ocorrerem tal que a taxa de transmissão fique menor que a taxa de bits da mídia, ocorrerão interrupções frequentes na reprodução.

Segundo Ozer [10], existem diversas opções para realizar o fornecimento de vídeos pela internet, e escolher dentre os tipos de técnicas é uma decisão importante a ser feita pois pode impactar no desempenho da entrega e na forma de produção dos arquivos. Forouzan [39] cita três maneiras para realizar *streaming*: sob demanda, tempo real e interativo. O método em tempo real é utilizado para transmissões de acontecimentos ao vivo. Já o método interativo diferencia-se por trabalhar com dois canais de comunicação entre cliente e servidor, com um canal servindo conteúdo para o cliente e um segundo canal estando a disposição para o cliente enviar vídeo e som, como seria no caso de uma vídeo conferência.

Por último, destas distinções a mais relevante, a entrega sob demanda é a maneira mais popular pela natureza do compartilhamento de conteúdo da Internet. Similar em princípio ao *download* na forma que o conteúdo deve ser requerido por um usuário utilizando um programa cliente, porém com a distinção de que o material solicitado só permanece armazenado temporariamente. E a partir desta maneira de transmissão sob demanda, Pfeiffer [40] cita três técnicas utilizadas, *streaming* tradicional, *download* progressivo e HTTP *streaming* adaptativo, a serem diferenciadas abaixo.

3.2 O *streaming* tradicional

A técnica de *streaming* tradicional, também conhecida apenas por *streaming* de mídia, utiliza servidores *web* especializados para o fornecimento de arquivos multimídia por meio de protocolos como *Real-time Transport Protocol* (RTP) [41] e/ou *Real-time Streaming Protocol* (RTSP) [42]. Neste modo tradicional de transmitir vídeo, os dados não permanecem armazenados nem em *cache*, nem no cliente, de modo que o usuário irá receber trechos dos vídeos, decodificar, exibir, e apagá-los logo em seguida, sem armazenamento de estado de nenhum dos lados da transmissão.

As vantagens oferecidas pelo serviço de *streaming* tradicional advém da busca por suporte às transmissões ao vivo. E, de acordo com Pfeiffer [40], um serviço pensado deste modo também oferece um nível extra de segurança por oferecer transmissão dos seus arquivos criptografados, sendo assim, por consequência, uma boa forma de combate a pirataria.

Segundo Wijering [43], a maior desvantagem do *streaming* tradicional ocorre devido a falta de apoio estrutural existente. Os protocolos RTP e RTSP comumente enfrentam bloqueios por *firewalls* corporativos e a falta de interesse por parte das empresas de

hospedagem em fornecerem servidores dedicados a transmissão de vídeos com a utilização desses.

3.2.1 *Real-time transport protocol* e *Real-time control protocol*

Apesar da perda de espaço para exibição de vídeos através da Internet, o *Real-time Transport Protocol* (RTP), definido na RFC 3550 [41], ainda é utilizado em cenários onde ocorrem o fluxo de dados em tempo real, como em rádios pela Internet e videoconferências. Esse protocolo é implementado no espaço de usuário e utiliza o *User Datagram Protocol* (UDP) para transporte do seu conteúdo. E, assim como o UDP, ele não garante a entrega ou ordem de chegada dos pacotes, ainda que possua mecanismos que possam ajudar nisso caso a aplicação que venha a utilizá-lo desejar implementar algo do tipo.

De acordo com Tanenbaum [44], a principal função do RTP é a de multiplexar diversos *streams* de dados (áudio, vídeo ou texto) para uma única sessão de pacotes UDP. Depois, no cliente, ocorre a demultiplexação utilizando números de sequência para garantir a ordem correta durante a reprodução, e marcas de tempo para garantir a reprodução síncrona das diferentes mídias.

No cabeçalho do RTP, adicionado aos fragmentos do conteúdo, há também informação sobre o tipo da codificação utilizada, que segue definições encontradas na RFC 3551 [45] e também indica o tamanho do fragmento a ser transmitido. Além disso, ao fim do cabeçalho, há possibilidade de adicionar campos personalizados ao servidor/cliente sendo utilizado, bastando para isso que a extensão esteja sinalizada e o tamanho extra seja informado com uma palavra (32 bits) no início do novo campo.

A RFC 3550 [41] também define o *RTP Control Protocol* (RTCP) que confere controle extra ao RTP. Normalmente a porta utilizada pelo RTCP é uma unidade maior do que a utilizada pelo RTP, criando um canal diferente e assim tornando esse um protocolo auxiliar fora de banda (que utiliza outra porta para enviar dados extras à atividade principal). Tem como principal função prover informação sobre a qualidade das sessões, como contagem de pacotes e atrasos; porém também define um CNAME (um apelido) a ser afixado aos participantes de transmissões, e é uma forma de fazer uma informação ser capaz de alcançar todos participantes de um *stream* (já que o RTP só é transmitido pela origem das mídias).

3.2.2 *Real-time streaming protocol*

Comumente usado em conjunto com o RTP, o *Real-time Streaming Protocol* (RTSP), descrito na RFC 2326 [42], também é um protocolo fora de banda, assim como o RTCP apresentado na seção anterior. Segundo Tanenbaum [44], o funcionamento do RTSP é

Tabela 3.1: Os métodos RTSP.

Método	**Descrição**
OPTIONS	Solicita quais são os métodos disponíveis
DESCRIBE	Solicita informações sobre a mídia da URL
SETUP	Configura protocolos para transmissão da URL
PLAY	Solicita reprodução de mídia do servidor
PAUSE	Solicita pausa temporária na reprodução
TEARDOWN	Solicita fim do *stream*, liberando recursos
RECORD	Solicita gravação conforme período e URL indicadas

análogo ao de um controle remoto, com o cliente solicitando ao servidor a reprodução, pausa, avanço ou retrocesso de um conteúdo servido a ele.

O formato das mensagens utilizadas é bastante similar às utilizadas pelo HTTP, esse a ser definido na seção 3.3.1, com requisições feitas pelo cliente e respostas advindas do servidor. As requisições seguem o formato de apresentar na primeira linha um método, *Uniform Resource Locator* (URL), e versão do RTSP; sendo os métodos listados na Tabela 3.1. A principal diferença aqui sendo que o RTSP mantém o estado durante a sessão.

A interação utilizando este protocolo começa a partir da solicitação dos comandos suportados pelo servidor. Logo após, comumente é solicitado um arquivo de descrição que conterá uma lista dos conteúdos multimídia disponíveis, e com base nessa lista o cliente fará a escolha entre tipos de áudio e qualidades diferentes do vídeo disponíveis. Porém, antes do comando de reprodução em si, o cliente informa como será feito o transporte da sessão (como as portas a serem utilizadas), e o servidor responde confirmando os parâmetros e os complementando com algumas informações próprias. A partir desse ponto as solicitações passam a ser utilizadas para o controle de mídia, como demonstrado na Figura 3.1.

3.3 HTTP *streaming*

Na seção anterior foi visto que no *streaming* tradicional a informação é transportada em pequenos pacotes por meio do RTP e RTSP. Muitas vezes *firewalls* e *proxies* acabam bloqueando esses protocolos, por não serem tão populares e utilizarem portas incomuns. E a impopularidade destes protocolos levou à necessidade de criação de alternativas adaptadas à realidade da Internet, tornando assim os servidores *web* atrativos também para a distribuição multimídia.

Utilizando um servidor *web* de *Hypertext Transfer Protocol* (HTTP), os serviços distribuidores de conteúdo adquirem a vantagem quando comparamos ao uso de servidores exclusivos para o *streaming* com protocolos dedicados. Servidores do tipo são populares

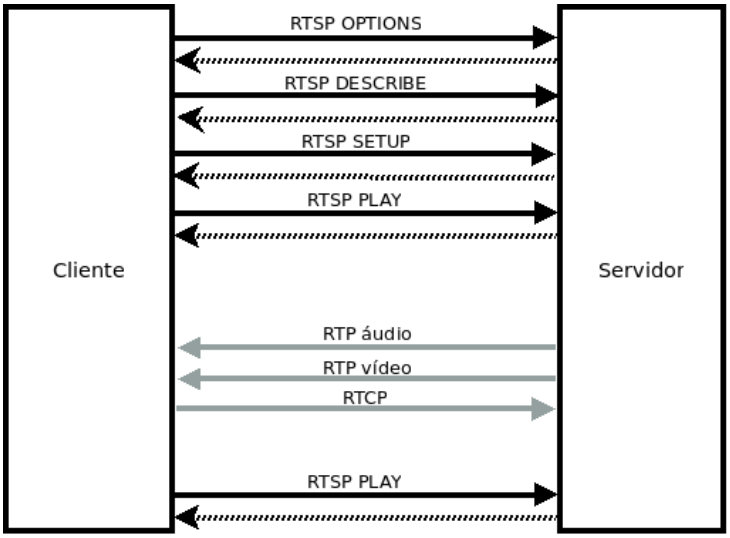


Figura 3.1: Operação RTSP com RTP/RTCP.

devido a sua responsabilidade em entregar páginas *web*, que são a finalidade original do HTTP; e, portanto, o uso do HTTP com foco em *streaming* é uma adaptação das requições de páginas discretas, fugindo da ideia de um fluxo contínuo de dados. Stockhammer [46] destaca que o *streaming* sobre o HTTP deve então considerar as particularidades deste protocolo de aplicação para aprimorar o serviço.

3.3.1 HTTP

O *Hypertext Transfer Protocol* (HTTP) pertence à camada de aplicação e é implementado em dois programas: um cliente e outro servidor. Os programas, enquanto estão sendo executados, podem trocar diversos tipos de informação entre si ao utilizarem o protocolo.

O HTTP surge para definir como ocorrem trocas de arquivos acessados por meio de URL únicos. O protocolo define a maneira com que arquivos, tipicamente páginas *web*, são requisitados e entregues por meio de diferentes métodos. Estes métodos (também referidos como requisições) servem para indicar a ação desejada a ser tomada em relação a um arquivo especificado. Normalmente, um navegador utiliza o método *GET*, e o servidor deve obrigatoriamente tratar ao menos esse e o método *HEAD*.

Os métodos foram concebidos para que o HTTP trabalhe de forma mais geral do que o necessário, indo além da simples solicitação de uma página *web*, e visando futuras aplicações. Cada solicitação HTTP consiste em uma ou mais linhas de texto, tendo como a primeira informação o nome do método.

O método *GET*, munido da representação do caminho de um determinado recurso, serve para retornar dados, sem nenhum outro efeito. O *HEAD* é bem semelhante ao

Tabela 3.2: Os métodos HTTP.

Método	**Descrição**
GET	Solicita a leitura de uma página *Web*
HEAD	Solicita a leitura de um cabeçalho de página *Web*
PUT	Solicita o armazenamento de página *Web*
POST	Acrescenta a um recurso (como uma página *Web*)
DELETE	Remove a página *Web*
TRACE	Ecoa a solicitação recebida
CONNECT	Estabelece um túnel TCP/IP
OPTIONS	Consulta certas opções

GET, mas sem retornar o recurso em si, somente informações sobre o recurso informado. Já o *PUT* e o *POST* servem para enviar informações, com a principal diferença de que, em vez de substituir dados existentes, o segundo anexa estes dados ao fim do recurso. Os métodos citados são os principais (a grande maioria das requisições utilizadas sendo *GET*), porém há outros métodos disponíveis que podem ser vistos na tabela Tabela 3.2.

As transferências feitas pelo HTTP utilizam o *Transmission Control Protocol* (TCP) como protocolo de transporte subjacente, e este fica encarregado de entregar os objetos dentro de uma conexão entre o cliente e o servidor, a cada requisição. A vantagem que o uso do TCP traz é que mensagens perdidas, mensagens duplicadas, mensagens longas e confirmações de entregas não estarão dentro do escopo do protocolo de aplicação.

No HTTP/1.0, depois que uma conexão TCP é estabelecida, ocorre uma única troca de requisição e resposta antes da conexão ser finalizada. Quando os conteúdos baixados são formados apenas por texto, esse método funciona bem. Porém com o passar do tempo, as páginas *web* gradualmente foram aumentando de tamanho, com o incremento do número de ícones, imagens e outros subterfúgios visuais, e, dessa forma, uma conexão TCP para cada arquivo se tornou algo custoso.

A partir dessa limitação trazida com o tempo, é especificado o HTTP/1.1 na RFC 2616 [47], desta vez admitindo conexões persistentes. Ou seja, nessa nova versão do protocolo, ao se estabelecer uma conexão TCP, passa a ser possível enviar mais de um conjunto de requisição e resposta, diminuindo o custo envolvido para se estabelecer conexões. Também são implementadas as requisições de intervalo, que permitem um cliente solicitar por fatias específicas de um arquivo. Ambos recursos veem a ser úteis dentro do contexto de *streaming*.

3.3.2 *Download* progressivo

O método de entrega que é mais utilizado nos dias atuais caracteriza-se pela transferência de dados multimídia utilizando um servidor *web* que disponibiliza o protocolo HTTP,

entregue do servidor para um cliente. Isso torna possível visualizar o conteúdo assim que se tenha dados suficientes, antes mesmo da cópia completa do arquivo.

Vantagens desse método se encontram, além do fato de que a espera do usuário é diminuída pois este não precisa esperar baixar todo conteúdo da Internet, também no fato do *download* progressivo disponibilizar a interrupção do *download* para continuar posteriormente, graças ao armazenamento dos arquivos localmente no usuário, permitindo aproveitar momentos de melhor banda. Outra vantagem é poder interromper e retroceder a exibição do vídeo enquanto se aguarda o carregamento do resto dele, ou até escolher outro ponto de exibição mais a frente, utilizando uma interface gráfica de barra do tempo de reprodução, demonstrada na Figura 3.2.

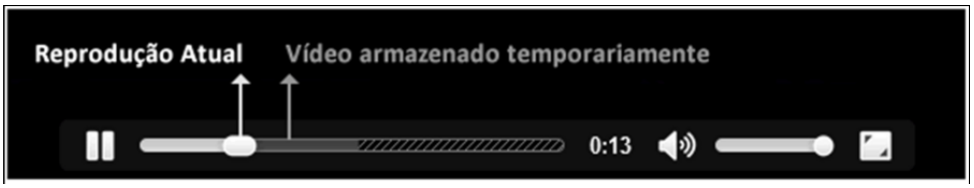


Figura 3.2: Barra de reprodução de vídeo no navegador *Chrome*.

Como desvantagens, há um menor grau de proteção contra a pirataria por parte dos distribuidores de conteúdo, já que os mesmos estarão salvos no disco do cliente, e também há o fato de que podem ocorrerem maiores atrasos para visualizar um vídeo, caso este esteja codificado a uma taxa que excede a capacidade da banda disponível. Logo, segundo Ozer [10], para que haja equilíbrio entre a qualidade e o tempo de espera, a taxa do vídeo deve ser escolhida pelo usuário com o devido cuidado antes de iniciar a transmissão.

3.3.3 *Streaming* adaptativo

Pfeiffer [40] descreve o *streaming* adaptativo como um processo que permite a entrega do vídeo com ajustes a fim de melhorar a qualidade do serviço oferecido. É também ofertada a possibilidade de suporte a fragmentos de arquivos em diferentes formatos, com uma ou mais taxas de vídeos a serem utilizadas com base em análises de banda disponível, situação do *buffer*, processamento da CPU e tamanho da tela de visualização.

As vantagens em relação a outros métodos está em situações onde há diminuição na taxa de transmissão da conexão de rede. Nestas condições, o servidor irá começar a enviar arquivos codificados em uma taxa menor, com qualidade inferior, para assegurar uma reprodução contínua. Para Wijering [43], o método oferece o melhor dentre os cenários discutidos, por ser possível garantir alta qualidade para aqueles com alta taxa de transmissão, e um fluxo razoável para aqueles com conexão lenta.

3.4 Protocolos proprietários

Ozer [10] descreve a história de como as principais alternativas de *streaming* existentes começaram a obter popularidade graças ao trabalho pioneiro da Move Networks que começou seu serviço de *streaming* em 2007. A Move chegou a ter importantes clientes e inicialmente se destacou, porém com o tempo deixou de ser um dos competidores no mercado de *streaming* por necessitarem da instalação de um *plugin* e não terem compatibilidade com propagandas. Porém, a partir desse momento, outras empresas começam a se conscientizarem quanto as possibilidades do *stream*, como a Adobe, a Apple, e a Microsoft. Esta seção tratará dos protocolos destas empresas, surgidos após esse espaço no mercado deixado pela Move.

3.4.1 Adobe *HTTP dynamic streaming*

Em 2008, a Adobe criou seu protocolo *Dynamic Streaming RTMP* que requeria a instalação de um servidor *Flash Media*, já popular na época, mas que conseguiu ampliar seu alcance para 99% dos computadores conectados a Internet [12]. Em 2010, de acordo com Ozer [10], ao perceber uma grande adoção do HTTP pelos concorrentes, a Adobe resolve lançar o *HTTP Dynamic Streaming* (HDS) que tornou o *Flash Player* compatível ao HTTP, permitindo-o receber conteúdos também de um servidor *web*, neste caso, o Apache [48] que também é comumente utilizado.

O Adobe HDS, definido em documento escrito por Hassoun [49], suporta tanto o uso de *streaming* sob demanda quanto a utilização de transmissões ao vivo, com a diferença entre esses no processo de codificação dos arquivos a serem enviados, com o fato de que o serviço em tempo real precisa de um servidor de empacotamento. Conforme a Adobe em [12], a codificação de vídeo pode ser H.264 ou VP6 (um formato de compressão e codificação de vídeos, anunciado em [50], e proprietário da On2 Technologies), combinadas respectivamente às codificações de áudio MP3 e AAC, com fragmentos de 2 ou 5 segundos no formato *F4V Fragment* (F4F), descrito em [51], que define a organização dos dados e metadados dentro de um arquivo multimídia, e é compatível com o *ISO Base Media File Format* (ISOBMFF) por usá-lo como base.

O pedido de fragmentos utiliza uma URL e numeração sequencial, para evitar repetidas requisições do arquivo manifesto, segundo Figueroa [52]. Manifesto esse que utiliza *Extensible Markup Language* (XML) e aponta as diferentes opções de áudio, legendas, e qualidades de vídeo; como pode ser visto no Código 3.1. Este padrão suporta tanto transmissões ao vivo quanto sob demanda, e adapta-se a variações de rede baseando-se na banda disponível e capacidade de processamento da máquina cliente.

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<manifest xmlns="http://ns.adobe.com/f4m/1.0" version="3.0">
 <id>myvideo</id>
  <duration>253</duration>
  <mimeType>video/x-flv</mimeType>
  <streamType>recorded</streamType>
  <baseURL>http://example.com</baseURL>
  <drmAdditionalHeader url="http://mydrmserver.com/
      mydrmadditionalheader"/>
  <bootstrapInfo profile="named" url="/mybootstrapinfo"
                 fragmentDuration="4"/>
  <media url="/myvideo/low" bitrate="408" width="640" height="
      480"/>
  <media url="/myvideo/med" bitrate="908" width="800" height="
      600"/>
  <media url="/myvideo/hi" bitrate="1708" width="1920" height="
      1080"/>
</manifest>
```

Código 3.1: Exemplo de arquivo manifesto do HDS

O HDS funciona via instalações de um módulo do Apache em conjunto ao *Adobe Media Server* no lado do servidor; e requer o *Adobe Flash Player 10.1* ou *Adobe Air 2*, ou versões mais recentes, instalados no cliente, o que traz passos extras para o usuário. A última atualização desse protocolo foi feita em 2015, de acordo com informações de lançamento da Adobe [53].

3.4.2 Microsoft *Smooth streaming*

Ao fim de 2008, ocorreu o anúncio do protocolo *Smooth Streaming* que utiliza o HTTP em constraste ao RTMP. A Microsoft demonstra a capacidade da nova tecnologia em diversos eventos esportivos, como as Olimpíadas de 2008, o campeonato de Wimbledon de 2009, e as Olimpíadas de inverno de 2010; com este último chegando a gerar mais de 6 *petabytes* de tráfego, como reportado em [54].

Para obter o funcionamento do *Smooth Streaming* é necessário que haja uma instalação do *Internet Information Services* (IIS), um servidor *web* da Microsoft com diversas funcionalidades via a instalação de extensões, como no caso do *Smooth Streaming* que funciona através da extensão *Media Services*. As requisições vindas do cliente podem seguir um formato único de URL ditado pela Microsoft, tornando o uso do IIS obrigatório para a tradução do endereço, ou via requisições de intervalo do HTTP/1.1.

O *Smooth Streaming* periodicamente detecta condições da rede para evitar as flutuações. E para otimizar o uso dessa informação, utiliza como métrica de decisão a banda disponível, janelas de tempo no *buffer* de reprodução, e a carga de processamento em que se encontra o cliente. Também distribui a carga do conteúdo entre duas conexões TCP, uma para áudio, outra para o vídeo.

Ao contrário dos outros protocolos, faz uso de dois arquivos manifestos, ambos em formato XML: um manifesto de cliente e outro manifesto de servidor. O arquivo de cliente deve ser requerido no início de uma sessão pelo programa cliente e apresentará as diferentes representações do conteúdo com marcas de tempo. Enquanto o arquivo manifesto do servidor é utilizado somente pelo próprio IIS para mostrar as relações entre taxas de vídeo, caminhos de arquivos e outros metadados, e assim a tradução de endereços poder ser feita ao se receber uma nova requisição. Segundo Mueller [4], caso as requisições pelo cliente em um sistema sejam feitas com intervalos do HTTP/1.1, o arquivo manifesto do servidor pode ser dispensado.

O último caso citado, da dispensa do manifesto do servidor, somente ocorre devido ao formato com que são armazenados os arquivos de mídia. O *Smooth Streaming* utiliza o mesmo conceito de enviar pequenos segmentos do conteúdo (tipicamente com 2 segundos de duração). Seus arquivos de mídia seguem o formato especificado pelo ISOBMFF, e podem conter mídias codificadas em VC-1, WMA, H.264, e AAC.

Os arquivos são organizados através do encapsulamento de metadados e conteúdo em um único bloco por fragmento, sendo que diversos fragmentos estarão presentes em um único arquivo de mídia, como demonstrado na Figura 3.3. Além disso, há no começo do arquivo metadados para todo o conteúdo, e ao fim do arquivo uma seção dedicada ao acesso direto dos diferentes fragmentos. De acordo com Deutscher [55], o acesso direto é o que permite o armazenamento de só um arquivo de mídia no servidor para cada representação do conteúdo existente, e, com isso, o envio de segmentos virtuais criados somente após a aquisição de uma nova requisição.

3.4.3 Apple *HTTP live streaming*

A Apple lançou em 2009 seu novo protocolo de *streaming*, o *HTTP Live Streaming* (HLS), através da inclusão de sua especificação como um IETF Internet Draft, até ser formalizado como a RFC 8216 [11]. Inicialmente o desenvolvimento do HLS foca nos seus dispositivos móveis, como o iPhone e o iPad, e nos navegadores Safari, sendo a única alternativa suportada nativamente segundo Ozer [10]. O protocolo continua em amplo uso devido a utilização dos dispositivos e programas mantidos pela Apple.

O HLS segue o mesmo princípio de divisão dos arquivos a serem transmitidos via *downloads* HTTP, e para isso define uma arquitetura com três partes fundamentais, de acordo

Figura 3.3: Formato de arquivo ISOBMFF usado no *Smooth Streaming*. Adaptado de Mueller [4].

com Figueroa [52]: servidor, distribuidor, e cliente, como exemplificado na Figura 3.4. O servidor codifica e encapsula o vídeo no formato apropriado, e também segmenta o arquivo gerando fragmentos e um arquivo manifesto. O distribuidor é um programa servidor *web* comum, que aceitará requisições e enviará os recursos desejados pelo cliente.

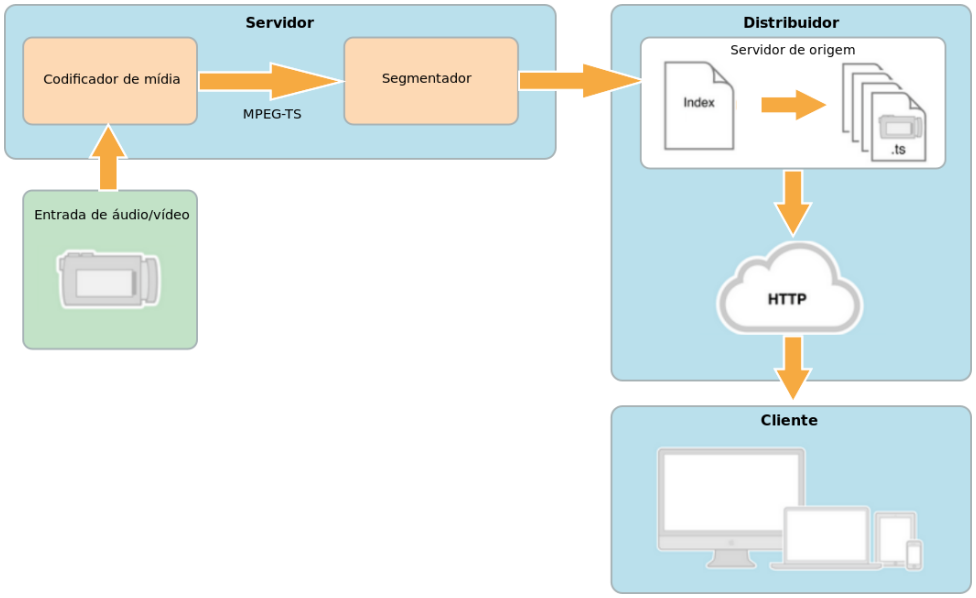


Figura 3.4: Arquitetura do HLS. Adaptado de Apple [5].

O cliente no início de uma sessão irá requerer o arquivo manifesto, uma lista de reprodução no formato *Extended M3U* (M3U8), que contem URLs e *tags* descritivas de outras listas de reprodução ou de segmentos de mídia, codificados em UTF-8. No caso mais simples, a lista de reprodução contém uma lista de segmentos que quando tocados sequencialmente irão exibir o conteúdo. O caso mais complexo ocorre quando se tem

Tabela 3.3: Resumo das principais características dos protocolos proprietários

	HDS	Smooth Streaming	HLS
Tipos de transmissão suportados	ao vivo e sob demanda	ao vivo e sob demanda	ao vivo e sob demanda
Formato de manifesto	XML	XML	M3U8
Formato de mídias	F4V Fragment (proprietário)	MP4f (ISOBMFF)	MPEG-TS, e MP4f (ISOBMFF)
Codecs de vídeo	H.264, VP6	VC-1, H.264	H.264
Codecs de áudio	MP3, AAC	WMA, AAC	AAC, MP3, AC-3, EC-3
Duração dos segmentos	2 ou 5 segundos	2 segundos	10 segundos
Heurísticas de adaptação	banda disponível, capacidade de processamento	banda disponível, buffer de reprodução, carga de processamento	taxa de transmissão atual, capacidade de processamento, resolução suportada, memória disponível

diversas codificações da mídia, o que requer uma hierarquia de listas de reprodução tendo uma lista mestre com um conjunto de variantes, que representam codificações em qualidades específicas. Cada variante ainda poderá ter diferentes representações, que são versões alternativas do conteúdo com, por exemplo, áudio em outros idiomas.

O conteúdo em si pode estar codificado como H.264 para o caso dos vídeos, e AAC, MP3, AC-3, EC-3 para o caso dos áudios. Já os segmentos no HLS inicialmente só podiam utilizar o formato *MPEG Transport Stream* (MPEG-TS), porém este adiciona uma sobrecarga significativa aos fragmentos e a Apple decide adotar o ISOBMFF, anunciando a decisão no *Worldwide Developers Conference* (WWDC) em 2016 [56], pensando em uma convergência com os clientes do MPEG-DASH.

O HLS foi projetado para atender tanto transmissões sob demanda quanto ao vivo. E para isso utiliza como heurística de adaptação a taxa de transmissão da rede mais as capacidades do dispositivo (dados do processador, resolução suportada, memória); suportando requisições de múltiplos segmentos ao mesmo tempo.

A duração dos segmentos é definida como um máximo, geralmente de 10 segundos, com a divisão do arquivo de mídia sendo feita de modo a conter ao menos um quadro chave com informações suficientes para uma decodificação eficiente. Além disso, o valor maior da duração em relação aos outros protocolos serve para o cliente diminuir a frequência com que são feitas requisições das listas de reproduções, que devem ser requeridas cada vez que houver uma mudança na variante ou representação utilizada.

A Tabela 3.3 apresenta um sumário sobre os protocolos proprietários discutidos anteriormente.

3.5 Padrão de *streaming* de vídeo adaptativo

Apesar de aparentar uma conversão entre as diferentes plataformas, a culpa disso é somente devido ao uso do HTTP, já que as plataformas privadas das empresas possuem realidades em que servidores e clientes, na maior parte dos casos, oferecem compatibilidade

restrita aos produtos das próprias empresas, sem que haja interoperabilidade. Porém, segundo Sodagar [6], a partir do cenário das novas propostas de empresas privadas, surge também o ímpeto para a padronização com um grupo no *Third Generation Partnership Project* (3GPP), que já pensava na convergência de tecnologias da Internet com a chegada das redes móveis.

O 3GPP foi a primeira organização a redigir um documento para atingir a padronização, e assim foi lançado o *Release 9* contendo o *Adaptive HTTP Streaming* (AHS). Após esse primeiro lançamento o *Open IPTV Forum* o tomou como base para descrever seus requerimentos para o *streaming* no seu *HTTP Adaptive Streaming* (HAS). Baseando-se nas especificações dos grupos anteriores e com forte colaboração com eles, o *Moving Picture Experts Group* (MPEG) estreou seu documento que viria a ser o estado da arte para a padronização de plataformas de *streaming* adaptativo. O *MPEG Dynamic Adaptive Streaming over HTTP* (MPEG-DASH) é um fruto dos esforços e contribuições para uma padronização independente de plataformas proprietárias, e ratificado internacionalmente pela MPEG e a ISO na ISO/IEC 23009-1 [13].

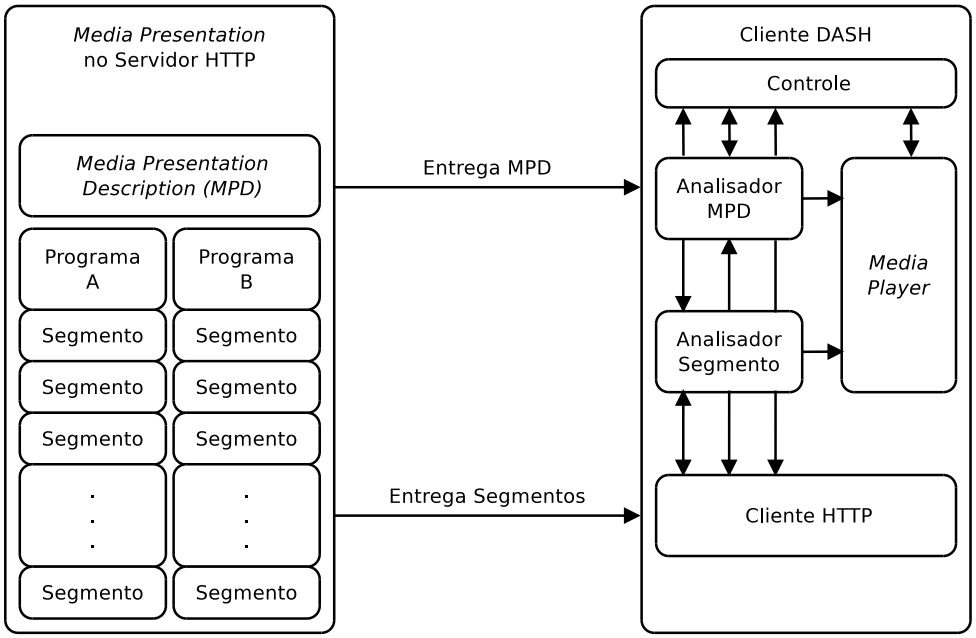


Figura 3.5: Escopo do MPEG-DASH. Adaptado de Sodagar [6].

Uma transmissão utilizando MPEG-DASH é estruturada como demonstrado na Figura 3.5. O conteúdo multimídia é capturado e armazenado em um servidor HTTP para posteriormente ser entregue. O conteúdo existe em duas partes: um arquivo descritor do conteúdo e suas alternativas disponíveis, chamado de *Media Presentation Description* (MPD), e os segmentos dos arquivos de mídia, podendo ser um único arquivo ou vários. E para que a reprodução comece, um cliente deve primeiro obter o MPD, depois reali-

zar a leitura do mesmo e ceder o controle a um algoritmo de adaptação, que servirá para selecionar qual será o segmento mais apropriado para requerer via comandos HTTP GET.

O algoritmo de adaptação não é definido na ISO/IEC 23009-1, ficando a cargo da implementação do *software* cliente. Os algoritmos que realizam um controle heurístico, ou seja, que tomam a decisão baseado em um subconjunto das informações, são discutidos na Seção 3.7. Nas próximas subseções serão detalhados os componentes do conteúdo disponibilizados pelo MPEG-DASH.

3.5.1 *Media Presentation Description*

O *Media Presentation Description* (MPD) [13] é um arquivo no formato XML que serve para descrever o conteúdo disponível e seus diversos formatos armazenados, endereços URLs, e outras características. Este arquivo descritor deve ser o primeiro recurso obtido pelo cliente, para que o mesmo possa depois fazer requisições HTTP GET para o *download* dos segmentos.

Um MPD é estruturado hierarquicamente como na Figura 3.6, mantendo a apresentação de um conteúdo dividida em períodos, grupos de adaptação, representações e segmentos, aumentando as possibilidades da descrição da mídia armazenada de acordo com as preferências do distribuidor. Os períodos delimitam temporalmente a mídia de forma relevante, podendo servir para contextualizar as cenas ou mesmo para a inserção de anúncios publicitários em determinadas marcas de tempo. Dentro dos períodos, há como classificar logicamente partes de um conteúdo através de grupos de adaptação, por exemplo agrupando por tipos de codificação, resolução, ou quantidade de canais de áudio. Como última hierarquia acima dos segmentos e seus endereços URL, temos as representações, que servem para abrigar a lista de segmentos sequenciais de mesma característica, como segmentos de vídeo de determinada qualidade.

Para realizar a descrição dos segmentos e seus endereços, o padrão do MPEG-DASH [13] descreve três principais *tags* XML a serem utilizadas dentro de uma representação, *SegmentBase*, *SegmentList*, e *SegmentTemplate*. Ao utilizar a *SegmentBase* é possível listar somente um segmento por representação, já as outras duas *tags* permitem listar diversos segmentos, com a diferença de que a *SegmentTemplate* oferece uma maneira de abstrair o padrão de nomes utilizado, assim não tendo que listar cada segmento da representação individualmente.

3.5.2 Segmentos

Segmentos são a menor e mais fundamental unidade do padrão, já que dentro deles que se encontra o conteúdo da mídia em si. São descritos através de URLs e outros atributos nas

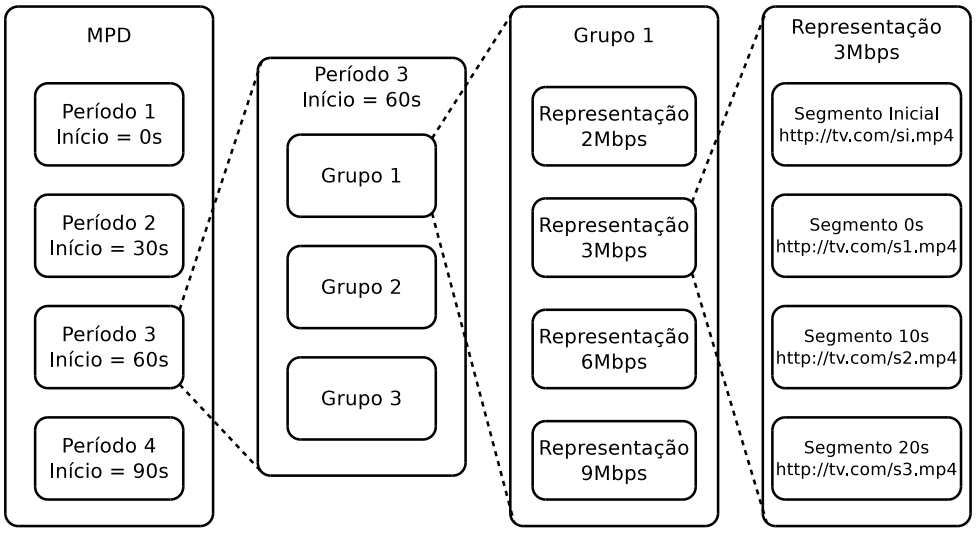


Figura 3.6: Hierarquia do MPEG-DASH. Adaptado de Sodagar [6].

representações, e são os segmentos que o cliente irá buscar ao iniciar uma sessão através de requisições HTTP GET ou requisições HTTP GET parciais.

Há três tipos principais de segmentos, são eles o segmento de inicialização, de mídia, e índice. Os segmentos de inicialização sempre serão os primeiros listados em uma representação por carregarem informações para a decodificação dos seguintes. Os segmentos de mídia são os que carregam o conteúdo em si, e também podem ser divididos em sub-segmentos para aproveitar as requisições parciais do HTTP/1.1. No caso de existirem sub-segmentos, cada segmento deverá ter no início de seu arquivo um campo que serve de índice pela informação dos tempos de reprodução e em que *byte* se encontram os sub-segmentos. Há também a opção de um arquivo próprio para estes índices existirem, e este é o segmento índice.

O MPEG-DASH não restringe a escolha da codificação de mídia, nem do tipo de formato que abrigará os segmentos, relegando a escolha aos desenvolvedores do programa cliente se usarão o MPEG-TS, o ISOBMFF, ou até outro formato ainda a ser adaptado. Um aspecto que o padrão deixa claro é que todos segmentos não podem se sobrepor no tempo de reprodução, e devem possuir ao menos um ponto de acesso ao *stream*, ou seja, um segmento deve possuir um quadro-chave, que contem informação suficiente de forma a ser possível decodificar todo o segmento independentemente dos outros. Isso para que a mudança entre diferentes representações possa ser arbitrária a qualquer mudança de segmento.

A próxima seção introduz uma parte fora do padrão, mas na prática importante para que o *streaming* ocorra, os algoritmos de adaptação.

3.6 Algoritmos de Adaptação

A entrega de conteúdo multimídia possui restrições firmes para que seja possível desfrutar dos dados entregues. Uma compressão torna ainda mais importante a relação entre a ordem dos pacotes para que possam ser decodificados, tornando perdas e atrasos bastante prejudiciais. Combinando isso com as redes de computadores atuais com natureza heterogênea em relação a diferentes bandas disponíveis e a números de clientes competindo por estas, *streaming* de vídeo apresenta diversos desafios.

Ainda que o padrão do MPEG-DASH seja bem robusto, ele também é flexível em sua implementação. Devido a essa flexibilidade, há bastante espaço para estudos em algumas áreas para oferecer assistência à adaptação de taxa de vídeos entregues. Algumas dessas áreas de estudo são sobre a arquitetura de entrega, técnicas de aplicação ao lado do servidor, auxílios das camadas de transporte e rede, e, o tópico de escolha deste trabalho, algoritmos de adaptação dos programas clientes.

Os algoritmos de adaptação não fazem parte do MPEG-DASH, portanto podem assumir tipos diferentes de funcionamento, ainda que o objetivo de todos seja aumentar a qualidade da experiência. A maneira com que eles buscam isso porém é distinta, já que há diversas métricas que podem ser utilizadas para avaliar seus desempenhos, e por consequência disso, diferentes algoritmos podem buscar otimizar diferentes aspectos numa sessão de *streaming* para um ou mais clientes. Algumas heurísticas porém se destacaram com o tempo e, tomando como principal distinção qual é o parâmetro de decisão utilizado, pode-se realizar a seguinte classificação para os algoritmos de adaptação: algoritmos baseados na taxa de transmissão, baseados na ocupação do *buffer*, ou híbridos. E todos tentam otimizar os mesmos ciclos de atividade ou ausência de atividade, além dos ciclos de controle, que são princípios gerais a serem discutidos nas subseções a seguir.

3.6.1 Ciclo ligado/desligado

Uma sessão de *streaming* em geral consiste em dois estados: o estado de preenchimento de *buffer* e o estado estável [57]. O estado de preenchimento de *buffer* ocorre logo no início de uma sessão, após a requisição do MPD, com o cliente requerindo um segmento após o outro, logo que o anterior tiver seu *download* completo. Quando um percentual alvo de ocupação do *buffer* é atingido (o que pode variar de acordo com o algoritmo), o cliente entra no estado estável.

No estado estável, ocorrem requisições periódicas de novos segmentos com a representação de acordo com o algoritmo de adaptação. Neste estado, o cliente se encontra ligado quando está requerendo ou baixando um segmento, e desligado quando está sem tráfego

de rede, resultando no ciclo de padrões alternados demonstrado na Figura 3.7. O tempo entre estados ligados é o tempo de ciclo e é tipicamente igual à duração de segmento.

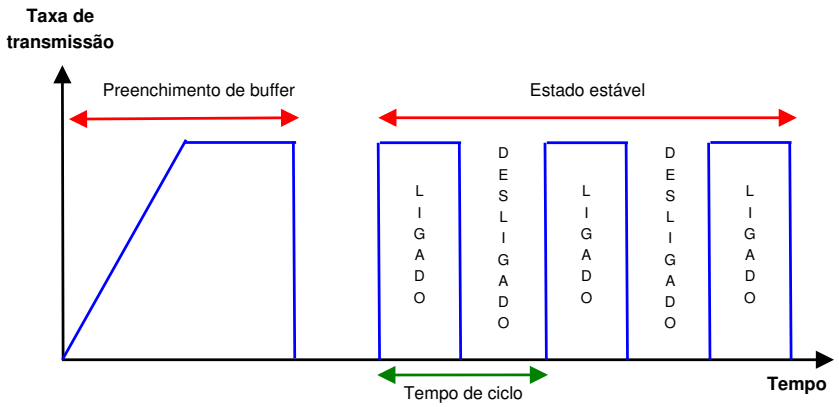


Figura 3.7: Comportamento cíclico de algoritmos de adaptação DASH.

Quando há competição por banda, os ciclos de atividades dos clientes geralmente tem seus inícios e tempos de ciclo diferentes. Caso vários clientes tenham seus estados ligados sobrepostos uns aos outros, como demonstrado na Figura 3.8, a visão de banda disponível varia. Isso causa os potenciais problemas como oscilações de qualidade, mudanças de taxas de vídeo, esvaziamento de *buffer*, injustiça e subutilização.

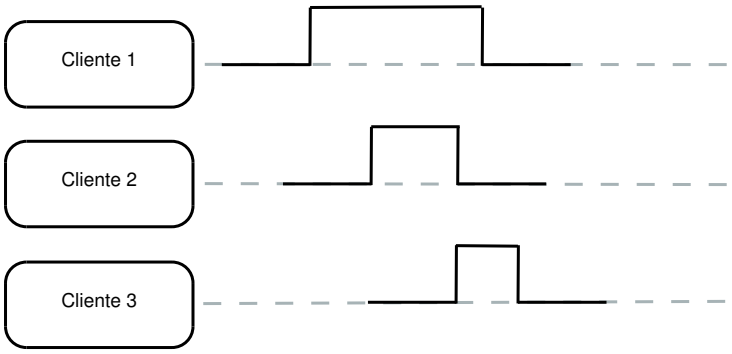


Figura 3.8: Ciclos de algoritmos de adaptação sobrepostos.

3.6.2 Controles dos algoritmos

Devido a utilização do TCP, o MPEG-DASH não controla as taxas de transmissões diretamente, dependendo do protocolo subjacente para regular esse aspecto. Por isso, para selecionar a taxa de vídeo apropriada em relação à rede, utiliza dois controles [58], como demonstrado na Figura 3.9.

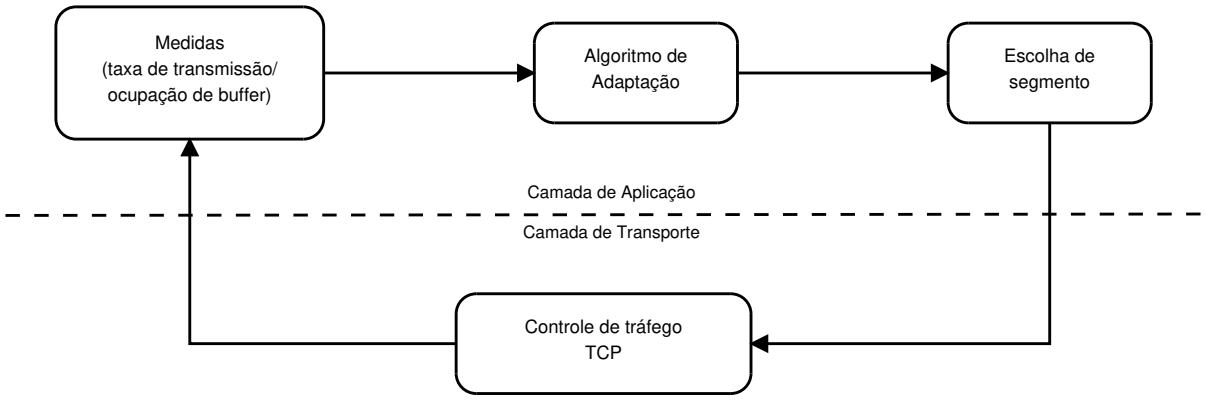


Figura 3.9: Controles de uma aplicação cliente do MPEG-DASH.

O controle interno equivale ao controle de congestionamento de rede do TCP e serve para reagir às mudanças nas condições da rede ao enviar dados em uma taxa de transmissão apropriada. Já o controle externo é equivalente à heurística utilizada pelo algoritmo de adaptação, reagindo às taxas de transmissão que o TCP impõe ao sistema. Assim, o controle do algoritmo de adaptação não precisa reagir às pequenas variações de rede, deixando isso para o controle interno e utilizando estimativas de taxa de transmissão nas escalas de segundos.

Os objetivos gerais que buscam ser atingidos com isso são:

- Evitar interrupções causadas por *buffers* vazios.
- Maximizar a qualidade de vídeo.
- Minimizar o número de mudanças na qualidade durante uma sessão.
- Minimizar o atraso para que ocorra o início da reprodução do conteúdo.

O desafio está em balancear a prioridade de cada objetivo individual, já que muitas vezes as escolhas estão em conflito uma com a outra.

Na próxima sessão serão apresentados como alguns algoritmos de adaptação existentes tentam atingir os objetivos, além de mencionar trabalhos relevantes ao *streaming* multimídia que fazem uso de redes neurais.

3.7 Revisão do Estado da arte

Os algoritmos de adaptação são os objetos de estudo do trabalho, e nesta seção serão discutidos três representantes destes, cada qual priorizando um parâmetro para sua decisão quanto a taxa de vídeo a ser utilizada. Ainda, dentre as várias partes que formam o todo

da entrega de conteúdo, buscou-se também as técnicas relevantes ao *streaming* de vídeo que empregam redes neurais, discutidas no Capítulo 2.

Para começar expondo os algoritmos de adaptação baseados na taxa de transmissão, apresenta-se o FESTIVE. *Fair, Efficient, and Stable adaptive Algorithm* (FESTIVE), por Jiang *et al.* [59], é um algoritmo multi-usuários que, a partir de observações de distribuidores de vídeos comerciais disputando o uso de mesmos canais de comunicação, busca focar em três métricas: justiça, eficiência e estabilidade. Os autores desenvolveram um conjunto de técnicas para adaptação de vídeo de modo a balancear os aspectos contraditórios das métricas. A injustiça, surgida quando há nível de uso muito díspar entre clientes; a ineficiência, quando não se consegue escolher a maior taxa de vídeo possível; e a instabilidade, que indica o número de alterações na taxa de vídeo, são resultados dos algoritmos estarem sobre a pilha de protocolos de rede e não conseguirem uma imagem da rede a tempo.

Os autores do FESTIVE então, com uma visão geral dos problemas, propõe um processo com três passos. Primeiro passo deve ser estimar a taxa de transmissão entre cliente e servidor, no artigo utilizando uma média harmônica entre valores obtidos durante os últimos vinte segmentos requeridos. Depois selecionando uma taxa de vídeo apropriada a estimativa anterior, mas com passos de incremento/decremento discretos, com uma política de aumento de taxa feito a cada determinado número de segmentos, e diminuição de taxa a cada segmento. E, por último, um agendamento de quando será feito a requisição, que pode ser imediata, caso o *buffer* esteja abaixo de um valor alvo ou de um valor aleatório para evitar sincronização entre clientes.

Representando os algoritmos de adaptação baseados na ocupação do *buffer*, o *Threshold-Based Adaptation Scheme for On-demand streaming* (TOBASCO), que foi desenvolvido por Miller [60]. Um grande objetivo no desenvolvimento do protocolo foi oferecer parâmetros flexíveis, para que pudesse manter uma baixa complexidade, e ser capaz de cobrir diferentes requerimentos de provedores e usuários. Em oposição ao FESTIVE, o TOBASCO preza mais pelo usuário individual, deixando o controle de alocação dos recursos para multi-usuários para o TCP, ainda que possua algumas salvaguardas. O estudo deste foca em quatro métricas: atraso de inicialização, *rebuffering*, qualidade do vídeo, e estabilidade; sendo o *rebuffering* a métrica com maior prioridade.

O TOBASCO busca melhorar as métricas com uma operação em duas fases: início rápido e adaptação. A fase de início rápido serve para obter o menor atraso possível ao iniciar a reprodução de um vídeo. Essa fase faz isso ao iniciar a reprodução sempre com a menor taxa de vídeo disponível e, para cada *download* de segmento seguinte, uma qualidade maior é selecionada até que a taxa de vídeo alcance uma determinada fração da média de taxas de transmissões passadas. Na adaptação, são definidos três níveis de

Tabela 3.4: Comparação entre os algoritmos de adaptação abordados no trabalho

Algoritmo	Heurística(s) principal(is)	Justiça	QoE	Número de clientes	Tipo de conteúdo
FESTIVE	Taxa de Transmissão	✓	✗	Vários	Sob Demanda
TOBASCO	Ocupação de buffer	✗	✗	Único	Sob Demanda
PANDA	Híbrido	✓	✓	Vários	Sob Demanda/Ao vivo

ocupação do *buffer*, B_{min}, B_{low}, e B_{high}. Através da comparação do nível de ocupação atual com esses limites definidos, é que se define a ação a ser tomada sobre a taxa de vídeo, controlando-a para que a ocupação possa ser mantida no intervalo alvo entre B_{low} e B_{high}.

Como último dentre os algoritmos de adaptação, temos o PANDA [61] que é uma abreviação de *Probe and Adapt*. Este é um algoritmo híbrido que parte do questionamento dos autores em relação aos limites existentes no uso de estimativas de taxas de transmissão, e então propõe um sistema que busca a banda disponível de forma análoga ao controle de congestionamento do TCP. O PANDA detecta congestão e a taxa de transmissão real disponível pela conexão através de uma sonda inicial com maior taxa de transmissão e posteriores reduções devido a congestionamento.

Através do uso dessa informação mais precisa, a adaptação pode ser feita ao determinar valores da qualidade de vídeo e intervalos entre requisições de forma a refletir o valor da taxa de transmissão. Por último, o *PANDA* tem uma mudança de taxa assimétrica, de forma que é mais conservador para subir a taxa de vídeo e mais permissivo na decisão para baixá-la, assim como no FESTIVE, porém sem a limitação de passos individuais na diminuição da taxa, o que o torna melhor para situações de queda na taxa de transmissão. Além disso, a técnica de sondar a rede frequentemente garante que múltiplos clientes convirjam para o uso justo da banda disponível.

A Tabela 3.4 resume alguns dos principais pontos ao projetar um algoritmo de adaptação, e se e como os algoritmos discutidos aqui abordam tais pontos. A começar pelo tipo de informação principal a ser usada na tomada de decisão, a tabela também informa se o algoritmo busca dividir o meio de transmissão de forma justa, se utiliza algum modelo de *Quality of Experience* (QoE) (no caso do PANDA, isso é feito através de uma maior modelagem do problema e uso de parâmetros), e para quantos clientes e para qual tipo de conteúdo foi pensado.

Para além dos estudos de algoritmos de adaptação, assim como o presente trabalho também faz uso de RNAs, encontra-se autores com focos diversos que implementam a tecnologia de aprendizado de máquina relacionando a questões abertas na transmissões de vídeos. Na parte de métricas, destaca-se Xue *et al.* [16] e Valderrama *et al.* [17] que buscam correlacionar métricas objetivas, que são mensuráveis por computadores, a

métricas subjetivas, vindas de avaliações de usuários. Por sua vez, Covell *et al.* [62] utiliza redes neurais para estimar valores de parâmetros que devem ser configurados para atingir uma taxa de vídeo desejada na etapa de codificação por parte de alguns distribuidores de conteúdo.

Por último, há a proposta de um algoritmo de adaptação feita por Lekharu *et al.* em [15], que na verdade é um modelo utilizando uma rede neural LSTM. O seu modelo, chamado de *Video Bitrate Prediction Model* (VBPM) possui a ideia simples de predizer a taxa de vídeo do próximo segmento baseando-se num conjunto de métricas. O sistema funciona com três fases: identificação de taxas, treinamento da RNA, e predição. Basicamente, as fases correspondem a uma anotação de uma tabela com as métricas (nível e tempo de *buffer*, e instabilidade) e taxas que sejam suficientes para obtê-las, treinamento em cima desses valores, e posterior uso da rede como função para dizer a qualidade desejada.

A busca e estudo dos trabalhos acima, fornecem as bases para compreender e propor uma mudança visando melhorar a qualidade de experiência do usuário que assiste a um vídeo pela Internet. Todos os trabalhos dos algoritmos de adaptação analisados visam essa melhora através da seleção de uma representação a ser exibida.

Este capítulo descreveu as bases para a transmissão de conteúdo multimídia através das redes de computadores, passando pela definição de *streaming* e a classificação de diferentes métodos que foram utilizados como estado da arte. Atualmente o foco dos estudos encontra-se em padrões que utilizam o HTTP, descritos na Seção 3.3, com alguns protocolos proprietários ainda relevantes devido às plataformas que atendem. Porém o MPEG-DASH, um padrão internacional rico em recursos, vem se popularizando, e foi discutido na Seção 3.5. Fora da definição oficial do padrão, mas utilizando-o para realizar as transmissões de vídeo, encontram-se os algoritmos de adaptação que são implementados em um *software* cliente; estes foram apresentados na Seção 3.7.

O próximo capítulo descreverá a série de passos para avaliar os algoritmos de adaptação mencionados, empregando as redes neurais para gerar informações dos segmentos.

Capítulo 4

Modificando algoritmos de adaptação para teste com saídas de RNAs

Os algoritmos de adaptação de *streaming* do estado da arte com origem no MPEG-DASH em sua maioria apenas inferem o tamanho dos segmentos de vídeo a partir da taxa usada na codificação. A proposta do trabalho é que por meio de redes neurais seja possível predizer tamanhos de segmentos de um conteúdo para então os algoritmos poderem fazer uso destas informações.

Neste capítulo é delineada uma metodologia a ser seguida para discutir a viabilidade de um algoritmo de adaptação que tenha a seu dispor o conhecimento sobre os tamanhos de segmentos. A Seção 4.1 fornece uma visão geral dos passos tomados, passando pelo problema a ser resolvido, a concepção da proposta e sua estrutura. Na Seção 4.2 é descrito como ocorre a utilização dos dados, detalhando sua formatação. A Seção 4.3 explica o processo de seleção do modelo de RNA a ser utilizado. Depois o processo de treino e teste do modelo selecionado é fornecido na Seção 4.1. Por último, a Seção 4.5 introduz o simulador e seus módulos utilizados para avaliar o sistema proposto.

4.1 Visão geral

Como visto no Capítulo 3, Seção 3.5, o MPEG-DASH é um padrão que busca tornar a visualização de conteúdos multimídia atra vés da Internet mais transparente e independente de protocolos privados com suas restrições de plataforma. Porém o padrão em si não é suficiente para realizar o *streaming* de vídeo, dado que ele deixa em aberto as questões de implementações. De especial interesse está a implementação da escolha de quais segmentos um programa cliente deve obter.

A ideia inicial é de que um programa cliente possuia dentro de sua mecânica de heurística o conhecimento dos tamanhos em *bytes* de um segmento de mídia. E com isso

chegou-se a proposta de utilizar uma rede neural para que esse tamanho pudesse ser estimado a partir de um segmento de menor qualidade, diminuindo a necessidade do uso de banda. O conhecimento que a estimativa de tamanho traz, mais a informação de banda utilizada, então seriam suficientes para que houvesse uma tomada de decisão também por meio de uma RNA.

Porém o trabalho começa a partir de testes para estudo da primeira questão de verificar se as redes neurais são capazes de estimar os tamanhos de forma satisfatória, o que inicialmente não estava claro. Depois também, como passos em direção à ideia original, aplica-se as informações estimadas em algoritmos reais para aferir se haveria algum ganho.

Tendo em vista este escopo a ser apresentado, é delineada uma divisão dos afazeres em dois componentes principais: redes neurais, e os algoritmos de adaptação. O objetivo é de, a partir de dados de entradas obtidos com vídeos, testar a utilização de redes neurais para estimar a informação extra dos tamanhos de segmentos com menor quantidade de entrada quanto possível, e também a implementação de um algoritmo que venha a interfacear com uma rede neural através da utilização das informações. Os resultados desse processo serão então utilizados para realizar comparações com algoritmos existentes.

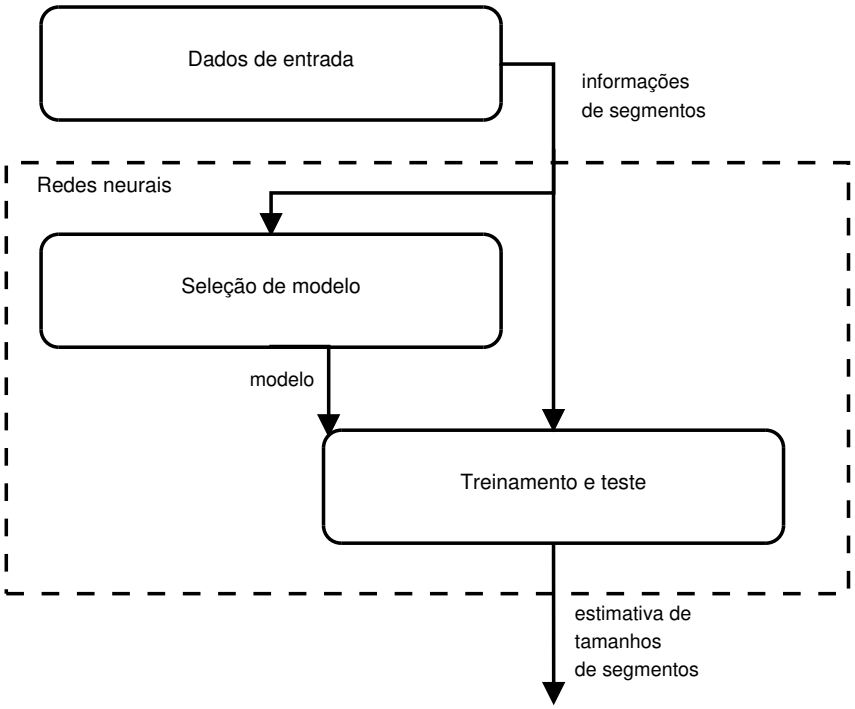


Figura 4.1: Principais etapas propostas do trabalho.

Para o uso das redes neurais para extração de informações dos segmentos de vídeo, toma-se três etapas exemplificadas pela Figura 4.1. Primeiro um pré-processamento dos segmentos disponíveis para o trabalho, a fim de tornar os dados aptos a serem fornecidos

de entrada aos programas desenvolvidos, ajustando formatos das estruturas de dados e divisão destes dados entre treinamento, teste e validação. Depois as etapas que envolverão os algoritmos de otimização, como descrito na Seção 2.4.1, contendo a seleção de parâmetros que melhor se apliquem a tarefa e, por último, o uso das redes neurais selecionadas para gerar tamanhos dos segmentos para as diversas taxas de vídeo disponíveis.

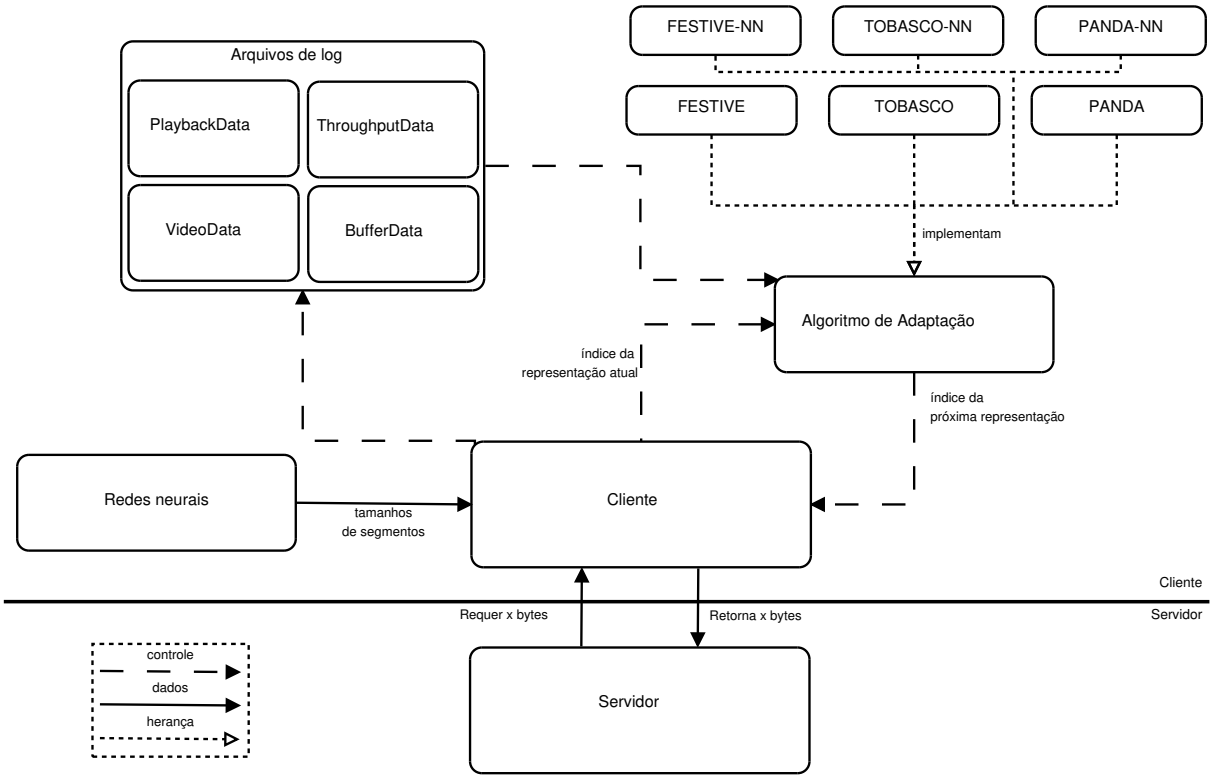


Figura 4.2: Diagrama funcional do modelo implementado. Adaptado de Ott [7].

O componente final do trabalho será a realização de simulações de algoritmos de adaptação. As simulações serão feitas de modo a comparar os algoritmos do estado da arte FESTIVE, TOBASCO e PANDA com versões modificadas dos mesmos que utilizarão de insumo para a tomada de decisão a saída das redes neurais. Na Figura 4.2 é possível verificar a organização das simulações, com os dados de saída das redes neurais podendo ser lidos pelo cliente, caso os algoritmos de adaptação estejam implementados através de classes que contem com a informação dos tamanhos individuais dos segmentos (na figura exemplificadas por FESTIVE-NN, TOBASCO-NN, e PANDA-NN). Mais detalhes das simulações serão discutidos na Seção 4.5.

Tabela 4.1: Representações para segmentos de 10 segundos.

Resoluções	Índices das representações	Taxas de vídeo
320 x 240	1 - 3	45, 88, 127 kbps
480 x 360	4 - 8	177, 217, 253, 317, 369 kbps
854 x 480	9 - 10	503, 569 kbps
1280 x 720	11 - 14	0.8, 1.0, 1.2, 1.4 Mbps
1920 x 1080	15 - 20	2.1, 2.4, 2.9, 3.2, 3.5, 3.8 Mbps

4.2 Dados de entrada

Para validar esta proposta, o primeiro passo é obter a informação dos vídeos e dispô-la de forma mais acessível para servir de dados de entrada ao trabalho desenvolvido. A fim de realizar a extração de informações que irão servir de dados de entrada, dispõe-se de quatro vídeos, popularmente utilizados nos estudos do MPEG-DASH e demais técnicas relacionadas ao *streaming*, apresentadas no Capítulo 3: o *Big Buck Bunny* [63], o *Elephant's Dream* [64], o *Tears of Steel* [65], e o *Of Forests and Men* [66]. Sendo que os três primeiros vídeos citados foram produzidos através do Blender [67] (programa de modelagem 3D, animação e edição de vídeos) como projetos de *software* livre, e o último foi encomendado pelas Nações Unidas e disponibilizado gratuitamente.

Previamente, os arquivos de mídia são codificados com diferentes taxas de vídeo (em torno de 13 a 20 taxas a depender do vídeo), e subsequentemente divididos em segmentos de diferentes durações (1, 2, 4, 6, 10, e 15 segundos). Na Tabela 4.1 encontra-se uma base de como as diversas representações foram obtidas, considerando segmentos com durações de 10 segundos, que é a duração a ser utilizada nas etapas deste trabalho, por ser um valor mediano entre os disponíveis. E, por consequência, obtém-se o conteúdo estruturado em diretórios, organizados de acordo com as durações dos segmentos e as diferentes taxas de codificação, como pode ser visto na Figura 4.3.

```
drwxr-x--- 22 thiroc www-data      4096 Mar 19 07:18 ./10sec
-rwxr-x---  1 thiroc www-data      8882 Out 16  2014 ./10sec/BigBuckBunny_10s_onDemand_2014_05_09.mpd
-rwxr-x---  1 thiroc www-data      4436 Out 16  2014 ./10sec/BigBuckBunny_10s_simple_2014_05_09.mpd
drwxr-x---  2 thiroc www-data      4096 Mar 19 07:18 ./10sec/bunny_1174238bps
-rwxr-x---  1 thiroc www-data    823348 Set 10  2014 ./10sec/bunny_1174238bps/BigBuckBunny_10s10.m4s
-rwxr-x---  1 thiroc www-data   1197960 Set 10  2014 ./10sec/bunny_1174238bps/BigBuckBunny_10s11.m4s
-rwxr-x---  1 thiroc www-data   1550369 Set 10  2014 ./10sec/bunny_1174238bps/BigBuckBunny_10s12.m4s
-rwxr-x---  1 thiroc www-data   1563038 Set 10  2014 ./10sec/bunny_1174238bps/BigBuckBunny_10s13.m4s
-rwxr-x---  1 thiroc www-data   1777391 Set 10  2014 ./10sec/bunny_1174238bps/BigBuckBunny_10s14.m4s
-rwxr-x---  1 thiroc www-data   1561549 Set 10  2014 ./10sec/bunny_1174238bps/BigBuckBunny_10s15.m4s
-rwxr-x---  1 thiroc www-data   1389232 Set 10  2014 ./10sec/bunny_1174238bps/BigBuckBunny_10s16.m4s
-rwxr-x---  1 thiroc www-data   1430428 Set 10  2014 ./10sec/bunny_1174238bps/BigBuckBunny_10s17.m4s
-rwxr-x---  1 thiroc www-data   1158045 Set 10  2014 ./10sec/bunny_1174238bps/BigBuckBunny_10s18.m4s
-rwxr-x---  1 thiroc www-data   1677747 Set 10  2014 ./10sec/bunny_1174238bps/BigBuckBunny_10s19.m4s
-rwxr-x---  1 thiroc www-data   1663239 Set 10  2014 ./10sec/bunny_1174238bps/BigBuckBunny_10s1.m4s
-rwxr-x---  1 thiroc www-data   1256371 Set 10  2014 ./10sec/bunny_1174238bps/BigBuckBunny_10s20.m4s
```

Figura 4.3: Exemplo de segmentos de vídeos armazenados.

Após ter essa listagem dos segmentos, realiza-se a leitura dos arquivos, montando

tabelas com as taxas de codificação dos vídeos, suas durações de segmentos, e os tamanhos individuais do segmentos. Com isso pode-se partir para uma divisão dos dados entre conjuntos de treinamento e conjuntos de teste; divisão essa que será por meio das taxas de codificação do vídeo. A razão entre as fatias da divisão será de acordo com o planejado para as etapas seguintes (detalhadas nas Seções 4.3 e 4.4), sendo feita de modo que o conjunto de teste represente 10% do todo na seleção de modelo e 20% para a realização dos treinamentos finais das redes neurais.

4.3 Seleção de modelo de RNA

Após ter disponível vetores dos tamanhos de segmentos e as taxas de vídeos aos quais os segmentos pertencem, classificados através do título do vídeo e duração dos segmentos, os melhores modelos de redes neurais serão selecionados para utilização no trabalho. Esse filtro do melhor modelo ocorrerá de forma empírica, iterando sobre os diversos parâmetros que formam um modelo, chamados de hiper-parâmetros, pois não há método formal para a escolha dos mesmos. Utiliza-se um ambiente de desenvolvimento Python, e com ele busca-se fazer uma validação cruzada de dez dobras (do Inglês, *ten-fold cross validation*) de forma a mensurar quão preciso são os modelos propostos.

Para a construção das redes neurais, é utilizado o Python e com ele a principal ferramenta para as redes neurais do trabalho, o Keras [68]. O Keras é uma biblioteca de alto-nível para a criação de redes neurais com foco em possibilitar uma rápida experimentação. Como demonstrado na Figura 4.4, a biblioteca serve como frente para do Tensorflow, que é a tecnologia central na declaração e cálculo de expressões matemáticas lineares com uso de múltiplos vetores e matrizes. A principal abstração utilizada é o objeto *Model*, contendo as informações do modelo e dos treinamentos entre seus atributos. Nesse objeto são adicionadas camadas, expostas primeiramente na Seção 2.1, sendo que cada uma possui um número de neurônios e sua arquitetura definida. Ao fim, também é definido o algoritmo de otimização a ser utilizado, tornando assim possível a atualização dos pesos das sinapses.

Para selecionar os hiper-parâmetros do modelo de RNA listados acima, será utilizado o procedimento de validação cruzada de dez dobras. Originalmente vindo da estatística para poder validar dados de forma mais justa, o procedimento funciona ao dividir a amostra de dados em dez e pegar uma parte dessa divisão dos dados para servir de teste. Essa divisão, chamada de dobra, é repetida também dez vezes com cada trecho dos dados sendo escolhidos como teste, como demonstrado pela Figura 4.5, para ao fim fazer uma média das métricas que se quer analisar.

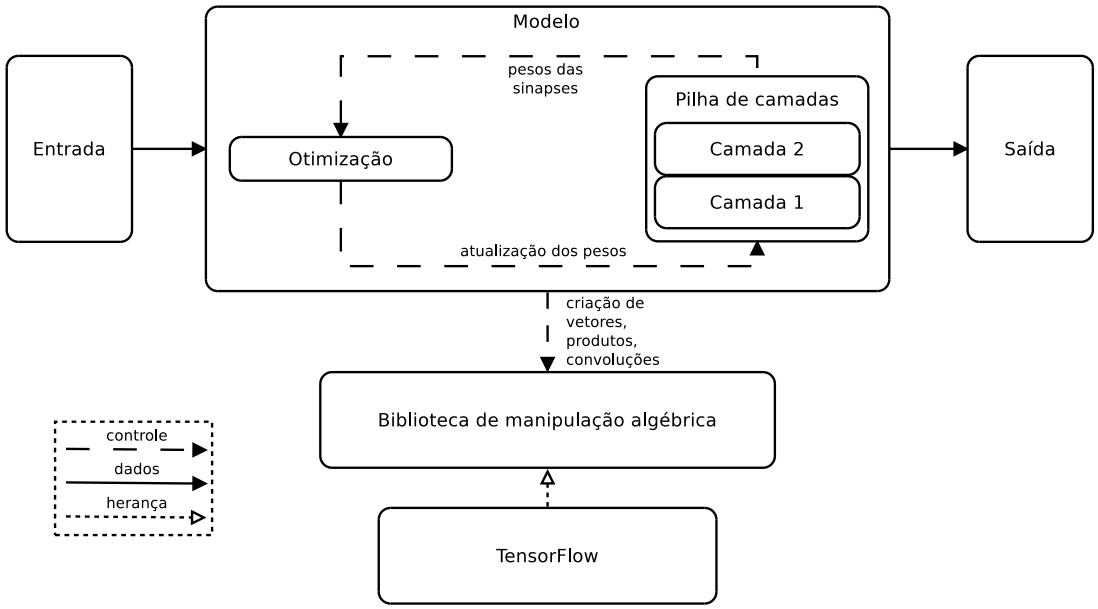


Figura 4.4: Diagrama de funcionamento de um modelo Keras.

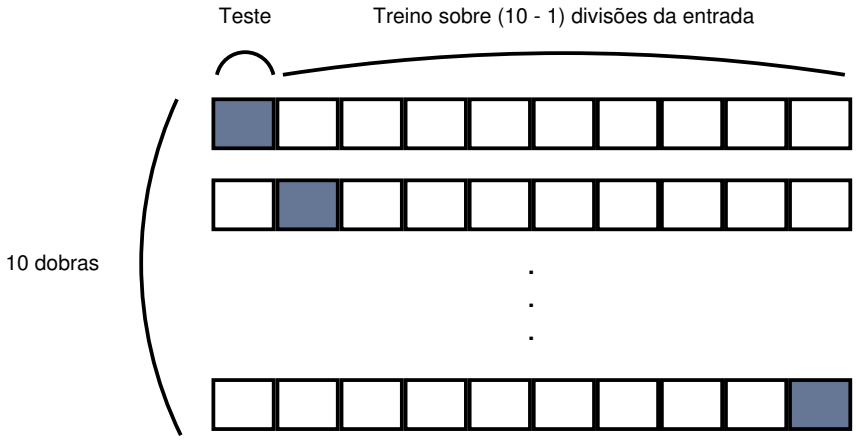


Figura 4.5: Ilustração do funcionamento da validação de dez dobras.

Com esse procedimento de validação, a seleção dos hiper-parâmetros busca reduzir o erro em relação aos valores desejados, que é medido como a raiz do erro médio quadrático (RMSE), como exposto na Seção 2.4.1, e sua versão percentual em relação aos valores desejados (rRMSE).

Aqui já serão realizados treinamentos e predições, manipulando os pesos do objeto Modelo do Keras e em cima dos dados de treinamento e testes das dobras, de forma a obter as métricas citadas acima. Durante o treinamento, também há uma divisão extra nos dados de forma que 10% dos dados serão utilizados como validação a fim de verificar a cada iteração do treinamento a métrica avaliada e poder interromper o processo mais cedo, quando o valor do RMSE apresentar algum crescimento. Logo, os dados iniciais,

citados em 4.2, são divididos em dados de treinamento, validação e testes; sendo que como entrada da rede serão utilizados um vetor grande o suficiente para caber tuplas com os índices dos segmentos e uma razão de proporção entre a qualidade mínima e a utilizada, e na saída um vetor com os tamanhos dos segmentos a serem obtidos.

Para encontrar o modelo que apresenta maior desempenho, os hiper-parâmetros sobre os quais ocorrerão as iterações serão as arquiteturas das camadas (RNRE, e LSTM), os algoritmos de otimização (*SGD*, *Adam*, e *Nadam*), os números de camadas e números de neurônios por camada; de acordo com o exposto no Capítulo 2. Todas as escolhas foram feitas de acordo com o que é mais conhecido no estado da arte e, tratando principalmente da arquitetura, o que é aplicado a contextos em que dados passados sejam levados em consideração no ajuste de pesos da rede.

4.4 Treinamento e teste

Na seção anterior, e com uma abordagem iterativa sobre diversos hiper-parâmetros, treinamentos foram realizados buscando satisfazer a proposta de uma RNA que obtivesse de saída tamanhos de segmentos em uma qualidade específica, e que isso fosse feito com menor erro possível. A partir desse ponto, será necessário selecionar os modelos de rede neural que obtiveram o menor erro e utilizá-los para realizar um treinamento e teste de maior extensão.

A maior diferença desta etapa do estudo para a etapa anterior (descrita em 4.3) será na extensão em que os treinamentos serão realizados. Em vez de 10 iterações, teremos uma única iteração para o treinamento da rede utilizada, com um maior número de épocas, 5000 épocas no caso, de forma a obter maior adequação dos pesos das sinapses. Além disso, os dados aqui serão divididos entre treinamento e teste, sendo que os dados de treinamentos representarão 80% dos segmentos de uma mídia, e os de testes representarão os 20% restantes.

Como saída desta etapa, serão colhidas todas as previsões obtidas tanto alimentando os dados do próprio treinamento quanto do teste, gerando assim duas novas tabelas por iteração de treinamento e teste, e viabilizando a próxima seção.

4.5 Simulações dos Algoritmos de Adaptação

Como última etapa do trabalho é desenvolvida uma maneira de simular os efeitos que o conhecimento do tamanho dos segmentos pode ter sobre um algoritmo de adaptação do MPEG-DASH.

Para começar essa etapa, além da pesquisa teórica sobre os algoritmos do estado da arte, utiliza-se uma implementação de um cenário de *streaming* com o ns-3 [69] por Ott *et al.* [7]. Os autores avaliaram que os estudos de algoritmos de adaptação frequentemente obtêm seus resultados sem o rigor necessário à avaliação de performance; seja por uso de cenários de rede não realistas, por avaliação de poucos parâmetros, ou por falta de clareza e compatibilidade nos métodos. E, tendo em vista esse cenário, visaram proporcionar um meio modular e que provê extensa capacidade de recordar o que se passa por meio de um *framework*, que nada mais é do que um grupo de código para um objetivo específico, neste caso para simulações.

A utilização de um simulador já estabelecido como o ns-3 provê um ambiente realista de rede com menor esforço. Já que constam todos modelos necessários para a pilha de protocolos TCP/IP, redes sem-fio, ambientes físicos e movimento dos participantes da rede, diversos cenários podem ser reproduzidos com maior controle sobre as variáveis envolvidas. E, além de tudo, um simulador permite obtenção dos resultados mais rapidamente.

O *framework* é organizado a partir de dois módulos que estendem a classe *Application* do ns-3. O módulo principal é o do cliente, que contêm toda a funcionalidade do algoritmo. O cliente também guarda as informações de taxas de transmissão, ocupação de *buffer*, e demais informações sobre o conteúdo a ser recebido por ele. Outro módulo filho da classe *Application* é o do servidor, que aqui cumpre o papel mais simples de receber uma requisição e enviar aquela quantia de *bytes* requerida. Por último, é importante citar a classe que contem o algoritmo de adaptação em si, que é uma classe filho da *AdaptationAlgorithm* e deve disponibilizar o método *GetNextRep()*, responsável pela seleção de qual a próxima representação a ser requerida, e com isso a próxima taxa de codificação. A Figura 4.2 apresenta os relacionamentos entre as diferentes partes do modelo de simulações empregado.

O repositório em que o *framework* é disponibilizado já contem alguns protocolos do estado-da-arte como o FESTIVE, TOBASCO e o PANDA, demonstrados em [7] e discutidos na Seção 3.7.

Além das implementações já disponíveis, busca-se criar novas classes de algoritmos de adaptação, que simplesmente adaptarão os mencionados acima para que possam utilizar informações discretas de tamanhos dos segmentos de um conteúdo, em vez da utilização de estimativas. Tais modificações foram simples pelo próprio *framework* oferecer métodos de leitura de tabela em arquivo texto do disco, permitindo o uso das novas classes para comparação entre os arquivos dos tamanhos de segmentos originais dos vídeos e dos que foram gerados através de RNAs como discutido na Seção 4.4; sendo que bastou atentar ao formato especificado para escrita e subsequente leitura durante simulações.

Também se fez necessário a alteração de parâmetros utilizados no FESTIVE, como a

janela que indica o número de iterações entre os segmentos utilizada para fazer estimativas das taxas de transmissão, e o valor de iterações com que o algoritmo fazia a mudança entre taxas de vídeo. Isso devido a utilização de duração de segmentos diferentes do proposto originalmente para o FESTIVE (segmentos de 10 segundos em vez de 2).

Por fim, as simulações são feitas em um ambiente reproduzindo um prédio de 30x18x18 metros, com um número de clientes variável a cada simulação tentando realizar uma transmissão de vídeo. Os arquivos tabela que contêm os tamanhos dos segmentos por vídeo serão ao todo oito, sendo quatro com os tamanhos originais e quatro com as saídas das redes neurais; todos utilizando somente os arquivos com segmentos de dez segundos. As combinações de parâmetros necessárias na simulação serão reproduzidas dez vezes de forma a obter um resultado estatisticamente relevante ao se analisar as métricas desejadas.

Ao fim desta etapa, o objetivo é extrair gráficos com as métricas para cada tipo de algoritmo de adaptação, com e sem o uso das saídas de redes neurais. As métricas desejadas serão indicativas de QoE. Quatro das métricas serão extraídas diretamente dos arquivos de registro feitos por clientes das simulações, como médias acompanhadas de seus respectivos desvios padrão; sendo elas: nível de *buffer*, tempo relativo de *buffer* vazio, índices das taxas de vídeo, e número de transições de taxas de vídeo. Além destas, já presentes em [7], serão adicionadas duas mais presentes em nos artigos do FESTIVE [59] e PANDA [61], indicativas de justiça na divisão da banda disponível e eficiência na utilização da mesma, como descritas nas equações:

$$justiça = \frac{(\sum_{i=1}^{n} bt_i)^2}{n \sum_{i=1}^{n} bt_i^2}, \tag{4.1}$$

e

$$eficiência = 1 - \frac{max(0, BW - \sum_{i=1}^{n} bt_i)}{BW}, \tag{4.2}$$

sendo n o número de clientes da simulação, BW a banda disponível, e bt_i a taxa de transmissão média do cliente i.

Este capítulo expôs as etapas pelo qual o trabalho passou para que obtivesse um resultado final sobre a possível otimização de algoritmos de adaptação ao utilizar informações de tamanhos de segmentos. As etapas foram divididas em quatro, sendo uma inicial para manipulação dos dados codificados de diferentes vídeos, duas outras dedicadas aos passos para o uso de RNAs como modo pelo qual obter as informações dos segmentos, e com a etapa final realizando simulações de algoritmos que possuíssem essas informações.

No próximo capítulo serão exibidos os resultados obtidos a partir dos treinamentos das redes neurais empregadas e as simulações realizadas.

Capítulo 5

Resultados Obtidos

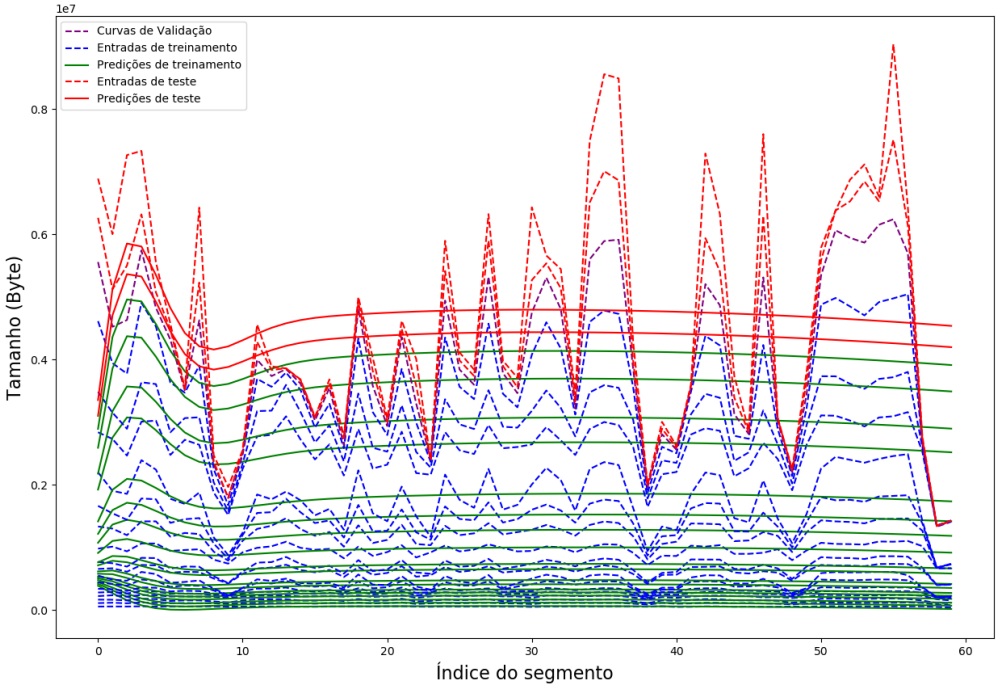
Os resultados observados nos experimentos são expostos neste capítulo. Primeiramente, são apresentados os resultados do processo de seleção entre diferentes modelos de redes neurais na Seção 5.1. Em seguida, é reforçado os resultados dos melhores modelos com um treinamento mais extenso na Seção 5.2. Por fim, na Seção 5.3, realiza-se a exposição dos frutos das simulações contendo três algoritmos de adaptação, sem e com o conhecimento dos tamanhos dos segmentos.

5.1 Seleção de modelo de RNA

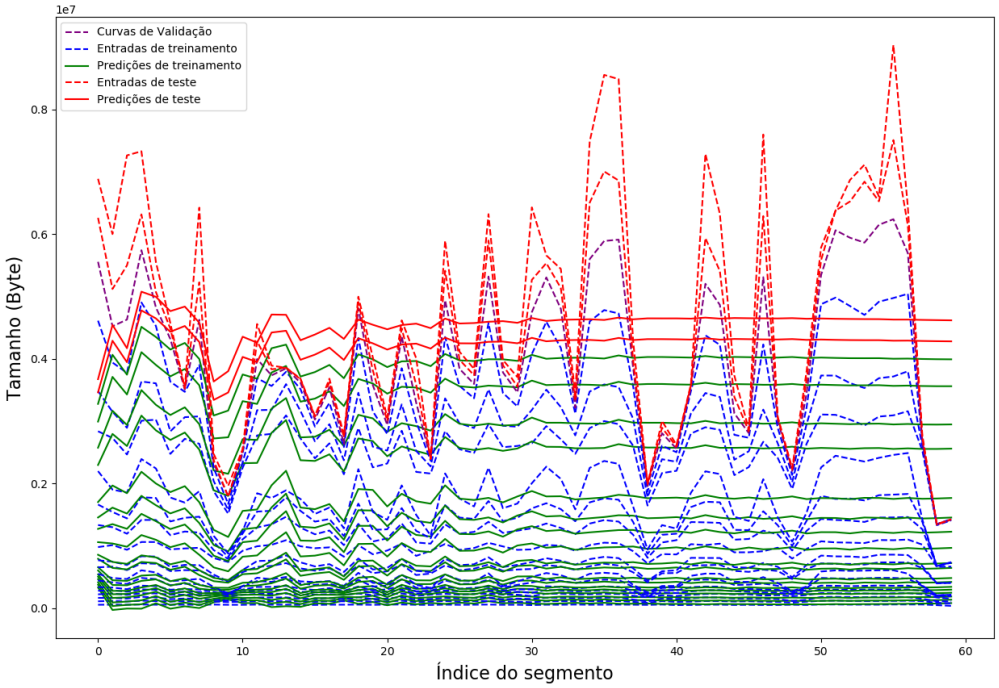
Como dito na Seção 4.3, nesta etapa foram realizados diversos treinamentos utilizando o método de validação cruzada de 10 dobras para selecionar hiper-parâmetros ótimos para uma RNA que possa gerar de saída vetores com tamanhos de segmentos de um vídeo.

Os hiper-parâmetros a serem testados foram selecionados de acordo com o encontrado na literatura do estado da arte, contendo as arquiteturas das camadas (RNRE, e LSTM), os algoritmos de otimização (SGD, Adam, e Nadam), os números de camadas e números de neurônios por camada; de acordo com o exposto no Capítulo 2. Os testes foram feitos ao utilizar quatro vídeos diferentes, todos com segmentos de duração de 10 segundos.

A métrica utilizada para a validação e escolha entre os diversos resultados dos parâmetros foi a *Root Mean Square Error* (RMSE), que é a raiz quadrada do erros médios quadrados entre os valores obtidos na saída do treinamento e os dados passados para que o treinamento fosse realizado (para realização do aprendizado supervisionado). Como exemplo, os resultados dos tamanhos de segmentos gerados pela rede com melhor métrica relativos a utilização dos dados do vídeo *Big Buck Bunny* [63] como entrada para a seleção podem ser visto na Figura 5.1; sendo que a Figura 5.1a representa uma RNA cujos hiper-parâmetros resultaram no melhor RMSE sobre os dados de treinamento, e a Figura 5.1b sobre os dados de teste.



(a) 58,5% rRMSE com dados de treinamento



(b) 54,29% rRMSE com dados de teste

Figura 5.1: Melhores redes identificadas através da validação cruzada sobre o Big Buck Bunny.

Tabela 5.1: Resumo dos melhores 5 resultados das validações cruzadas.

		Treino	Teste
Arquitetura	RNRE	09	13
	LSTM	11	07
Otimização	SGD	00	00
	Adam	14	12
	Nadam	06	08
Número de Camadas	1	17	17
	2	03	03
Número de Neurônios	15	06	04
	45	01	04
	50	04	04

Os hiper-parâmetros testados mais uma contagem dos mesmos nos melhores resultados de cada vídeo podem ser vistos na Tabela 5.1. A contagem inclui os melhores cinco resultados de todos os vídeos para cada cenário de treino e de teste, ou seja, há nas colunas da tabela vinte dos melhores resultados ao se considerar os valores RMSE a partir de dados de treinamento e vinte ao se considerar os valores dos dados de teste.

Ao observar os modelos com melhores RMSEs, nota-se que, ao se tratar das arquiteturas escolhidas, não há uma única que apresenta melhores resultados para avaliação sobre os dados de treinamento e dados de teste. Para a avaliação sobre os dados de treinamento, a arquitetura LSTM se sai melhor, já sobre os dados de teste, a RNRE se mostra na frente. Isso deve-se pela capacidade de memória das redes LSTM, provendo um maior contexto para a obtenção da saída quando já viu aqueles dados de entrada. Já no caso das redes RNRE, observa-se que obtiveram com a mesma quantidade de treinamento uma capacidade de abstração melhor quando os dados de entrada apresentados as redes eram novos.

Fora a arquitetura, o algoritmo de otimização que se destacou aqui foi o mais consagrado Adam, quebrando a expectativa inicial de que o Nadam obteria melhor desempenho devido a uma convergência mais rápida. Para a camada escondida, uma única camada se mostrou melhor, com a quantidade de quinze neurônios sendo escolhida ao fim, porém sem demonstrar vantagem tão clara como nos casos dos hiper-parâmetros anteriores.

5.2 Treinamento e teste

O objetivo dos treinamentos das redes neurais é o de prover informação dos tamanhos de segmentos dos vídeos. Sendo assim, todas rodadas de treinamento feitas almejam um resultado com menor erro possível, de forma que as informações de uma RNA possam ser

Tabela 5.2: Desvio padrão dos tamanhos de segmentos em relação a taxa de vídeo

	Tamanhos de segmentos	Big Buck Bunny	Elephant's Dream	Of Forest And Men	Tears of Steel
Q1	Originais	3.574,83	8.446,93	9.960,22	4.541,32
	Predições	93.372,78	169.146,81	37.135,87	627.407,36
Q2	Originais	41.085,74	73.265,26	69.413,14	112.838,02
	Predições	93.273,26	170.675,41	36.569,00	627.431,27
Q3	Originais	101.774,58	208.692,45	216.217,67	728.850,98
	Predições	92.050,96	174.729,42	36.347,51	628.810,56
Q4	Originais	603.994,10	901.942,56	797.132,06	1.878.509,51
	Predições	99.671,91	189.599,74	56.531,09	636.406,46

utilizadas pelo cliente de modo eficiente. Nesta seção serão apresentados os resultados obtidos com aqueles melhores hiper-parâmetros selecionados anteriormente.

Os modelos utilizados eram formados pela arquitetura LSTM e otimização Adam, com uma camada e quinze neurônios. Como foi ressaltado na seção anterior, as redes neurais que utilizaram da arquitetura LSTM obtiveram os melhores resultados em relação aos dados de entrada já utilizados durante o treinamento. Por isso, aqui serão a única arquitetura utilizada, já que para estes experimentos os dados dos tamanhos de segmentos estão disponíveis por todo o procedimento e é assumido que assim será em um caso de um servidor que possa vir a utilizar de técnica similar de uma rede por vídeo.

Os treinamentos foram feitos para os quatro vídeos buscando um maior número de iterações para verificar quais valores de erros obtidos e repetindo algumas vezes para que toda a extensão dos dados pudessem ser utilizadas ao menos uma vez como teste. Como os tamanhos de segmentos foram divididos por qualidade, de forma que os dados de teste equivalessem a 20% do todo, os treinamentos e predições foram realizados cinco vezes para cada vídeo. Ao fim, cada treinamento demonstrado aqui foi feito com 5000 épocas devido a observações empíricas da convergência dos gráficos que eram obtidos ao fim de cada treinamento. Destes treinamentos finais, também salvaram-se suas predições e erros para depois realizar cálculos das médias das saídas e erros.

As médias de erros citadas acima foram feitas inicialmente em relação a taxa de codificação dos vídeos, tanto dos tamanhos dos segmentos originais de cada vídeo quanto dos obtidos na saída das RNAs, como pode ser visto na Tabela 5.2. O objetivo foi de poder comparar estes desvios e assim estabelecer a vantagem ou desvantagem quanto ao uso das saídas das redes neurais para os algoritmos de adaptação sem nenhuma modificação.

Como ficou claro que haveria uma vantagem na utilização das saídas preditas pelas redes neurais, decide-se continuar com a ideia motriz do trabalho. Esta tabela apresenta os resultados não são sempre favoráveis (9 dos 16 casos ilustrados), porém, ao mesmo tempo, para a maior parte dos vídeos, quanto maior a qualidade de uma representação de vídeo, maior parece ser a vantagem que as redes neurais obtidas apresentam.

Nas Figuras 5.2 a 5.5 estão representadas amostras de uma das iterações de treinamento e teste das redes neurais. Em contraste às figuras relativas ao *Big Buck Bunny* apresentadas na Sessão 5.1, observa-se um avanço em termos de aproximação da saída geradas em relação aos gráficos originais, tanto de treinamento quanto de teste. A melhora nas predições com mais épocas indica espaço para melhorias, tanto nas saídas com os dados existentes quanto em casos em que dados dos tamanhos tenham de ser realmente previstos pelas redes neurais; sendo que o segundo caso indica a possibilidade de um estudo em que isso seja feito para que uma rede com maior capacidade generalista venha a prover informações para mais de um vídeo por modelo.

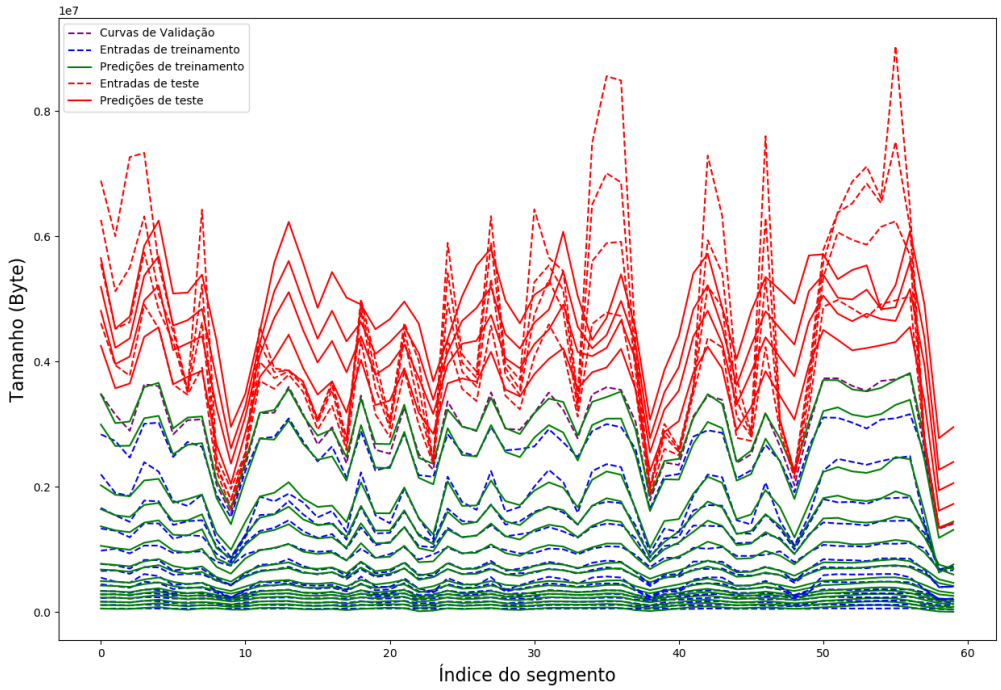


Figura 5.2: Tamanhos de segmentos gerados por melhor modelo com 5000 épocas para o Big Buck Bunny.

5.3 Simulações dos Algoritmos de Adaptação

Devido à combinação de fatores envolvidos na simulação (número de arquivos de entrada, algoritmos e quantidade de repetições de um cenário) é possível dividir o trabalho entre diversas máquinas individuais. Esta natureza independente das simulações tornou atrativo o uso de máquinas virtuais que podem ser replicadas a partir de uma única assim que esteja

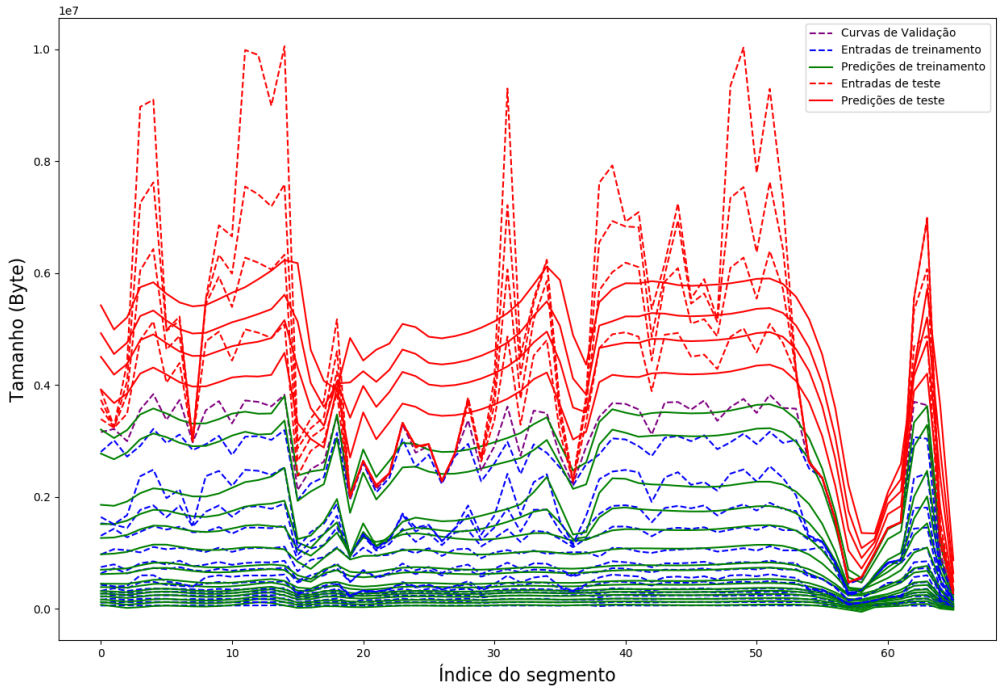


Figura 5.3: Tamanhos de segmentos gerados por melhor modelo com 5000 épocas para o Elephant's Dream.

configurada, ou fazer com que *scripts* sejam responsáveis por preparar os pré-requisitos em máquinas já existentes. Esses dois aspectos foram utilizados durante a execução do trabalho em diferentes momentos, aproveitando tanto do uso de virtualização local quanto da disponibilidade de servidores em nuvem para acelerar a obtenção dos resultados (16 máquinas por meio do Google Cloud no caso).

Após as simulações foram obtidos diversos arquivos de registro do tempo em que os algoritmos de adaptação dividiram um canal de rede sem fio, e a partir desses registros foram extraídas informações relevantes ao cálculo de métricas que ajudam a identificar a qualidade da experiência (QoE) de um usuário. Nesta seção serão apresentados os resultados de cada métrica obtida, sintetizados por alguns gráficos.

Ao todo são seis métricas distintas, de acordo com o exposto em [59] e [61]. As simulações foram executadas 10 vezes de acordo com cada combinação de parâmetros do estudo, sendo estes parâmetros: 4 possíveis vídeos com seus tamanhos de segmentos originais e preditos por redes neurais, 3 protocolos de acordo com o especificado originalmente e estes mesmos 3 protocolos adaptados ao uso da informações de tamanho de segmentos preditos, e 4 diferentes quantidades de clientes (5, 10, 15 e 20 clientes). Nesta Seção

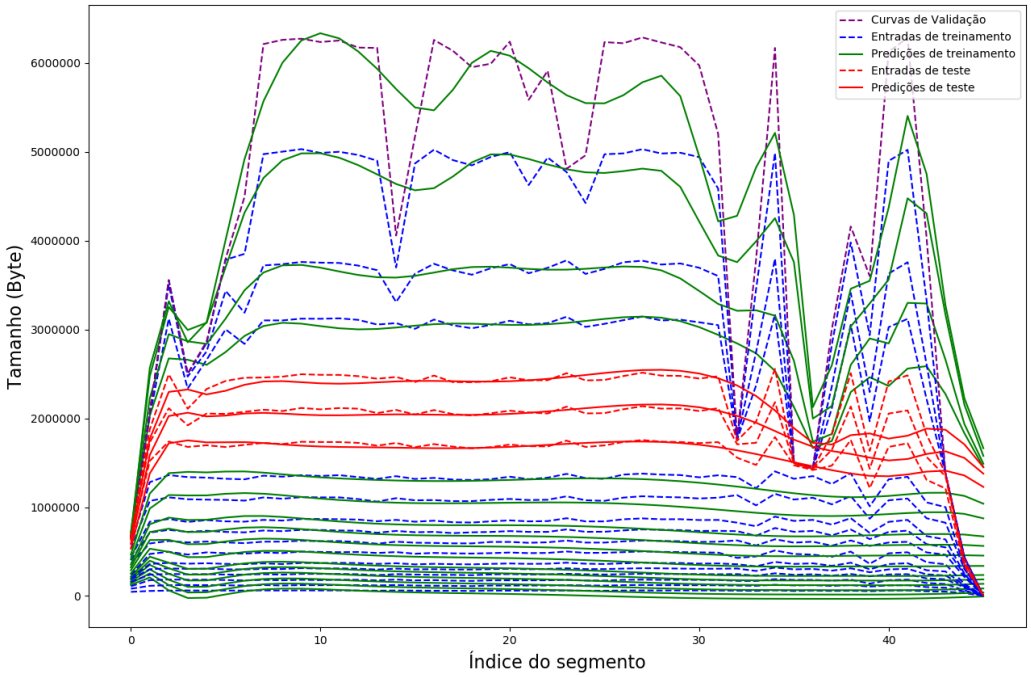


Figura 5.4: Tamanhos de segmentos gerados por melhor modelo com 5000 épocas para o Of Forest And Men.

serão exibidas figuras relativas somente às métricas de 20 clientes, por serem mais fáceis de visualizar e porque em sua maioria não há tanta diferença no grau de melhora com a variação do número de clientes; nos casos em que houver, será utilizado um gráfico de curvas apropriado.

Os protocolos utilizados possuem alguns parâmetros envolvidos em seus funcionamentos, e onde estes tiveram de ser especificados, seguiu-se as especificações dos artigos originais, sendo o FESTIVE descrito por Jiang *et al.* [59], o TOBASCO por Miller [60], e o PANDA por Li *et al.* [61]. Com a observação de que foram utilizadas duas versões do FESTIVE, uma original e outra otimizada, mais responsiva para o caso do trabalho em que foram utilizados segmentos de 10 segundos; os parâmetros alterados são mostrados na Tabela 5.3.

5.3.1 Índices de taxas de vídeo

A primeira métrica a ser observada em relação aos algoritmos de adaptação é a média dos índices de taxas de vídeos, que variam entre 13 e 20 a depender do vídeo, como mostrado

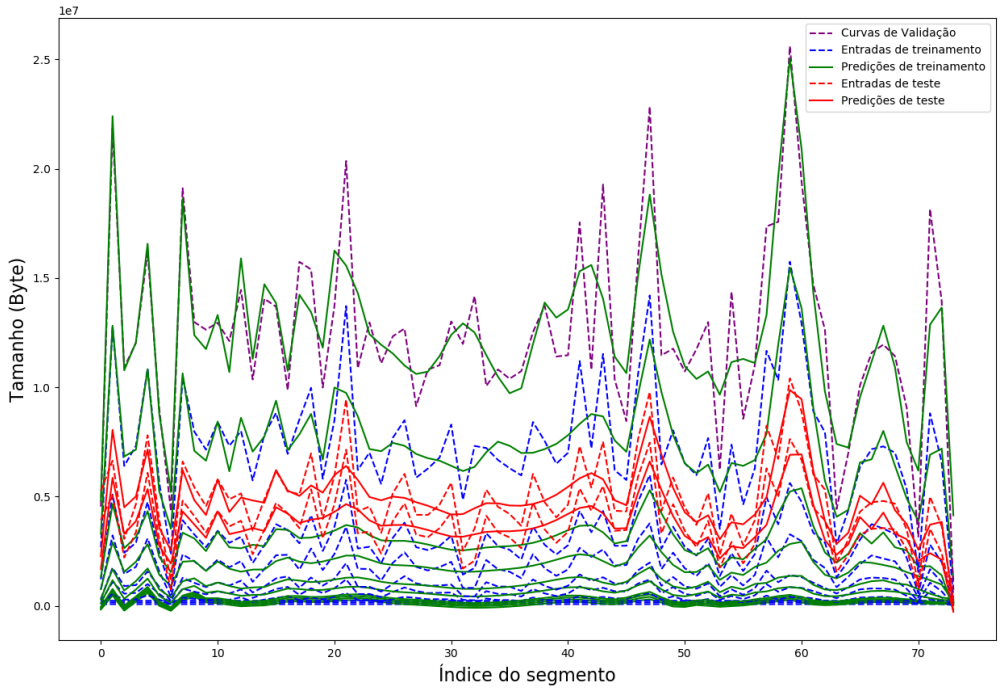


Figura 5.5: Tamanhos de segmentos gerados por melhor modelo com 5000 épocas para o Tears Of Steel.

na Seção 4.2. Aqui porém é feita uma média desconsiderando a diferenciação do vídeo, por brevidade.

Através da Figura 5.6, pode-se verificar que para todos os casos os índices obtidos pelos algoritmos que utilizaram as saídas das RNAs foram menores, independente do número de clientes.

O algoritmo de adaptação menos afetado pela mudança de heurística foi o TOBASCO, com os valores dos índices se mantendo bem próximo aos originais, possivelmente devido ao nível de seu *buffer* não ser afetado diretamente pelo valor dos segmentos preditos, que servirão somente para a consulta na hora da tomada de decisão. Já o mais afetado foi o PANDA, que é inicialmente o algoritmo cuja estimativa da taxa de transmissão de um

Tabela 5.3: Tabela de parâmetros alterados do FESTIVE.

Algoritmo	Parâmetro	Valor
FESTIVE	κ (janela de amortização)	20
	k (atraso de incremento)	5
FESTIVE otimizado	κ (janela de amortização)	5
	k (atraso de incremento)	3

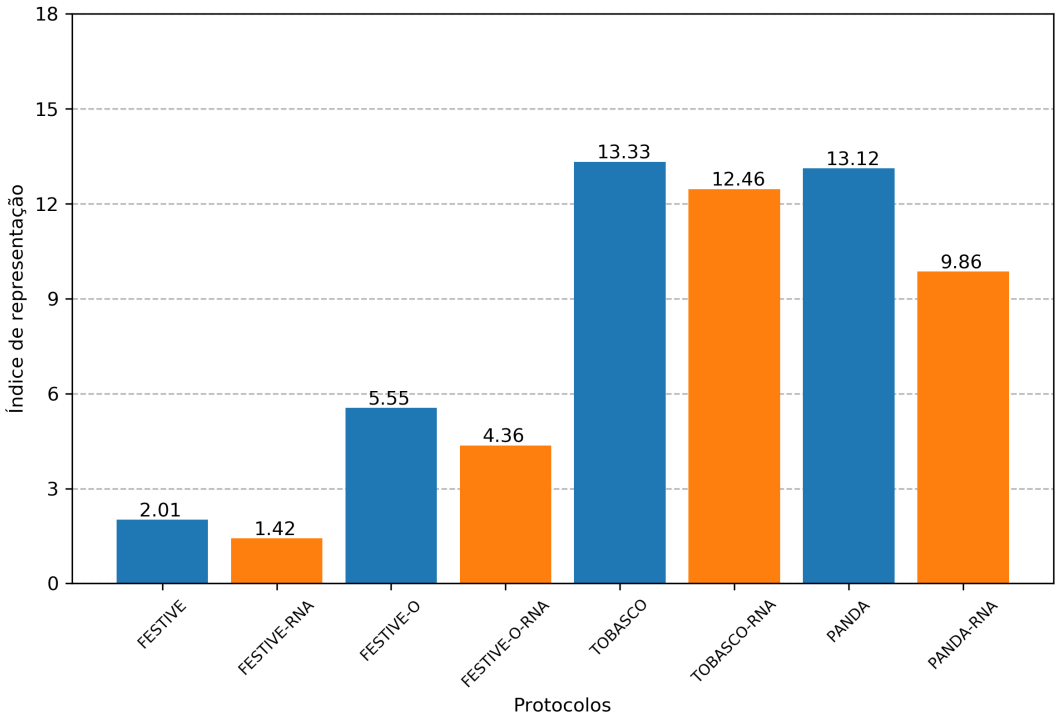


Figura 5.6: Médias dos índices de representação das simulações dos algoritmos de adaptação.

segmento possui mais peso.

5.3.2 Número de transições de taxas

A medida do número de transições de representação por qual um cliente passa durante uma sessão de *streaming* é uma métrica importante pois define a percepção de continuidade de um usuário que esteja assistindo a certa qualidade. Por si só, esta medida traz informação equivalente à estabilidade utilizada pelos trabalhos de Jiang *et al.* [59] e Li *et al.* [61].

Como observado na Figura 5.7, o FESTIVE é o algoritmo de adaptação que apresenta uma melhora se tratando do algoritmo que utiliza as saídas das RNAs em relação ao algoritmo original, independente do número de clientes. Já para o TOBASCO e o PANDA ocorrem pioras com a modificação do algoritmo, sendo que o PANDA é o mais afetado com uma perda de estabilidade.

5.3.3 Nível de *buffer*

A informação sobre a média da quantidade de tempo de vídeo armazenada em *buffer* não é traduzida imediatamente em uma melhora ou piora na qualidade do serviço percebida,

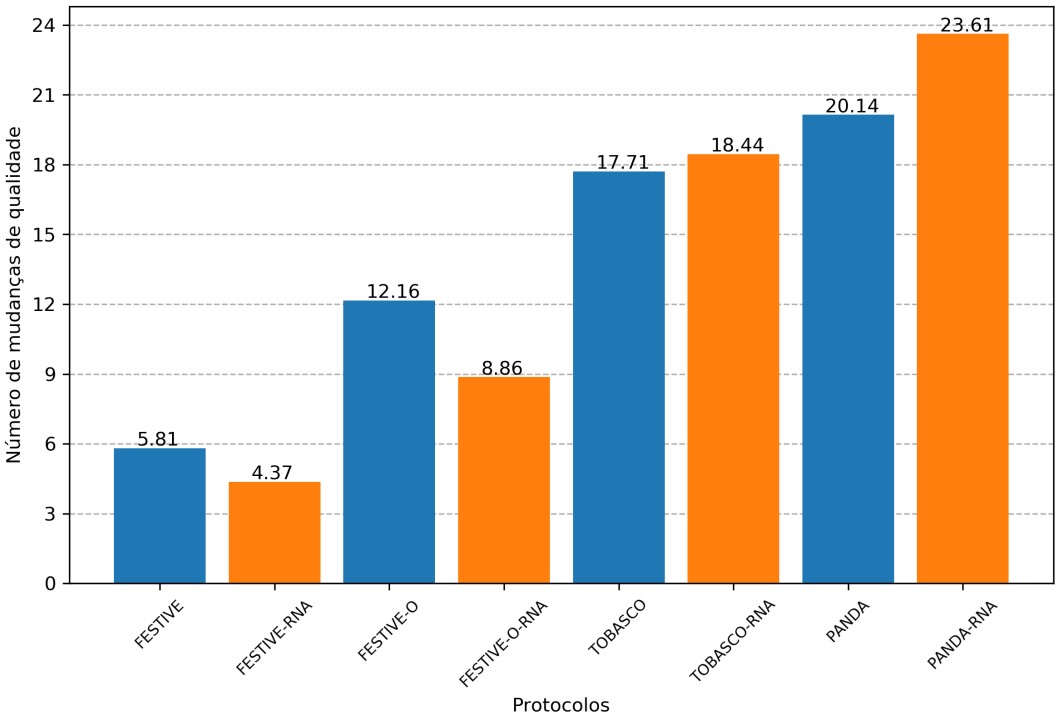


Figura 5.7: Médias da quantidade de mudanças nas representações das simulações dos algoritmos de adaptação.

porém é importante que este nível seja mantido por prevenir interrupções na reprodução. Sendo assim, um aumento do nível de *buffer* é positivo por diminuir as chances das interrupções.

Como resultado das modificações nos algoritmos de adaptação para uso das previsões das RNAs, observa-se na Figura 5.8 que aqueles algoritmos cuja taxa de transmissão tem influência maior na decisão, como o FESTIVE e PANDA apresentam uma melhora, enquanto para o TOBASCO, que tem uma heurística baseada em *buffer*, há uma leve piora.

5.3.4 Tempo de *buffer* vazio

Ainda relacionadas aos níveis de utilização de *buffer*, temos as medidas do tempo proporcional em que o *buffer* ficou vazio em relação ao tempo de reprodução do vídeo, demonstradas na Figura 5.9. Essa é uma medida que tem bastante peso na qualidade de um *streaming*, visto que é a causa de interrupções na sessão.

Pelas simulações, porém, não observou-se muitas interrupções por parte dos algoritmos avaliados, sendo o PANDA o único algoritmo que demonstrou maiores problemas. Sendo que a versão modificada do PANDA para uso das predições das redes neurais mostrou-

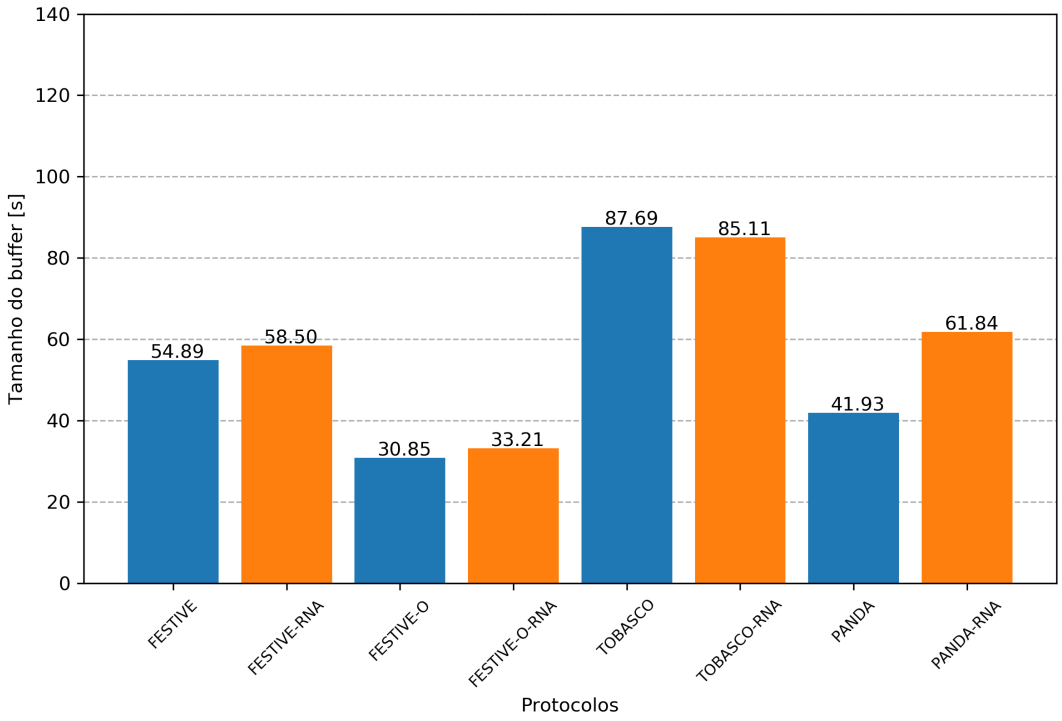


Figura 5.8: Médias do tempo de vídeo armazenado em *buffer* nas simulações dos algoritmos de adaptação.

se pior até a quantidade de 15 clientes disputando a banda, mas com uma melhora de desempenho com um maior número de clientes, de acordo com a curva na Figura 5.10. A explicação na melhora com maior número de clientes se daria a natureza mais ambiciosa (*greedy*) do algoritmo, que reduz a qualidade, e as comparações entre a taxa de codificação do vídeo e a banda disponível para um cliente seriam feitas mais eficientemente, com menores folgas entre os dois valores.

Em contraste, temos o FESTIVE sem interrupções em qualquer de suas versões, e o TOBASCO que até teve interrupção, mas mantém esse aspecto sob controle com uma política de redução de qualidade agressiva quando nota o *buffer* esvaziar.

5.3.5 Eficiência

Aqui começa-se a discorrer sobre métricas que não são diretamente extraídas dos arquivos de registro da simulação, e precisaram ser calculadas para indicar aspectos importantes da utilização da rede disponível entre os clientes. A primeira dessas métricas é a eficiência, que teve seu cálculo exposto na Equação 4.2, como a média das somas das taxas de transmissão de todos os clientes sobre a banda, para cada tempo t da simulação.

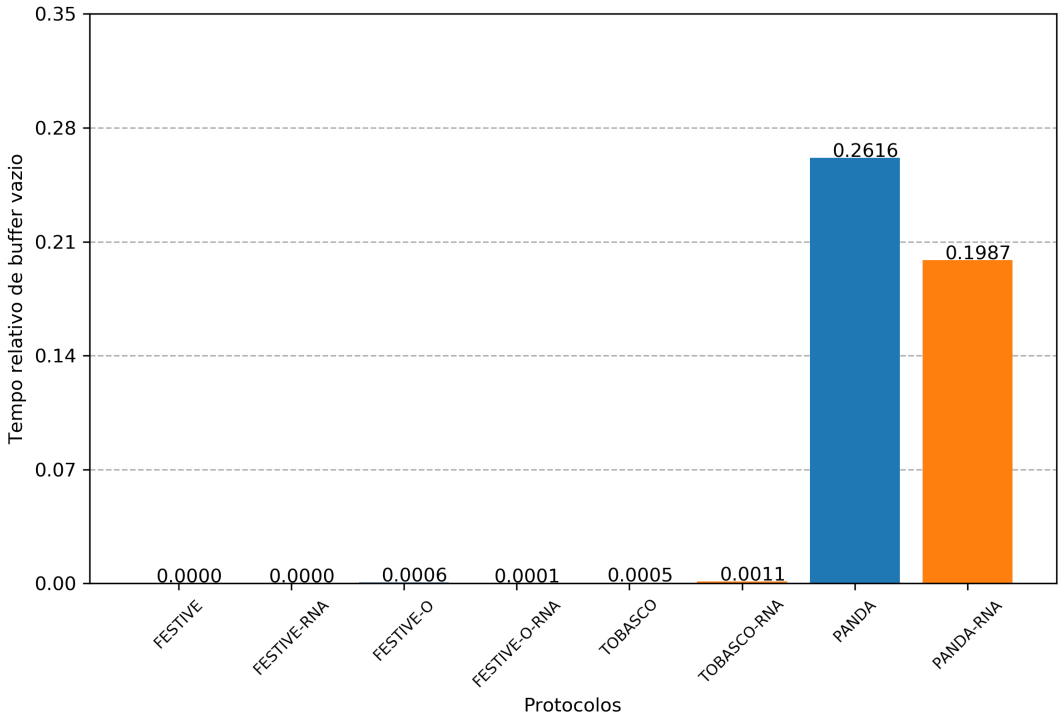


Figura 5.9: Tempo relativo de *buffer vazio* das simulações dos algoritmos de adaptação.

Para a eficiência de utilização da banda disponível, observa-se por meio da Figura 5.11 que os algoritmos TOBASCO e PANDA apresentam melhoras quando modificados para utilizar os tamanhos de segmentos preditos, enquanto o FESTIVE apresenta uma piora. O FESTIVE também foi o mais afetado pela modificação, mas como este possui um foco maior em otimizar a justiça e estabilidade na utilização da banda por múltiplos clientes, este resultado não é inesperado.

5.3.6 Justiça

Como última métrica, apresenta-se a justiça na divisão dos recursos da rede entre os clientes, a qual obteve em geral uma melhora com as modificações dos algoritmos. O destaque nesta métrica vai para os algoritmos de adaptação que são mais influenciados pelas taxas de transmissão, como pode ser visto na Figura 5.12.

No caso dos algoritmos FESTIVE e TOBASCO, estes somente obtiveram ganhos com o uso das saídas das RNAs; o que é especialmente positivo para o caso de uso do FESTIVE, já que a justiça é uma métrica central no desenvolvimento deste algoritmo. Porém, ao observar a Figura 5.13, fica claro que os resultados para o PANDA também merecem ser discutidos, já que inicialmente não foram bons, mas com o crescimento do número de

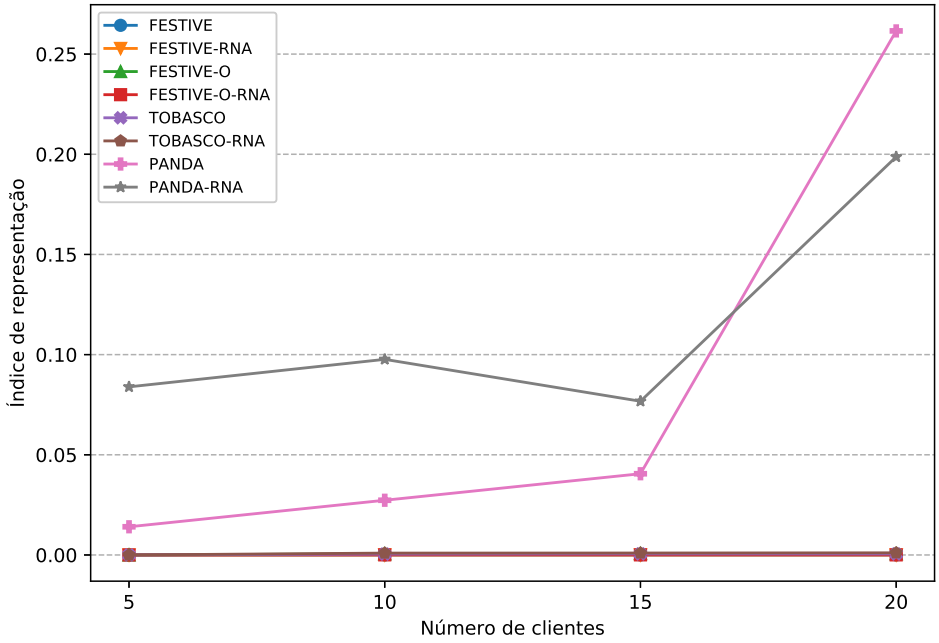


Figura 5.10: Tempo relativo de *buffer vazio* das simulações dos algoritmos de adaptação para diferentes quantidades de clientes.

clientes essa tendência se inverte e há um ganho significativo na métrica mensurada.

Este capítulo apresentou os resultados coletados em cada etapa do estudo desenvolvido para aferir a influência do uso de previsões de tamanho de segmentos de um vídeo obtidas por meio de redes neurais em algoritmos de adaptação estabelecidos no estado da arte. Primeiramente, foram mostrados como aconteceu a seleção dos hiper-parâmetros das redes neurais a serem utilizadas na Seção 5.1, depois uma comparação dos erros obtidos com os treinamentos utilizando tais redes na Seção 5.2, para enfim poder utilizar a saída das RNAs como fator de decisão nos algoritmos de adaptação e extrair as métricas apresentadas na Seção 5.3.

O Capítulo 6 discutirá as conclusões que podem ser traçadas.

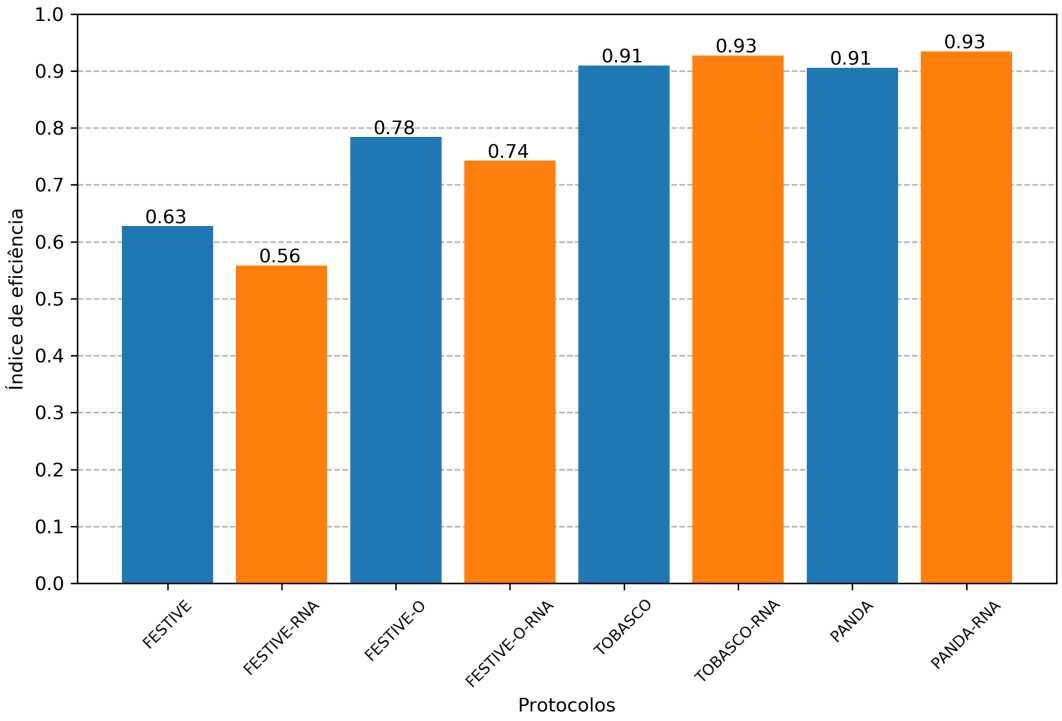


Figura 5.11: Médias da medida de eficiência das simulações dos algoritmos de adaptação.

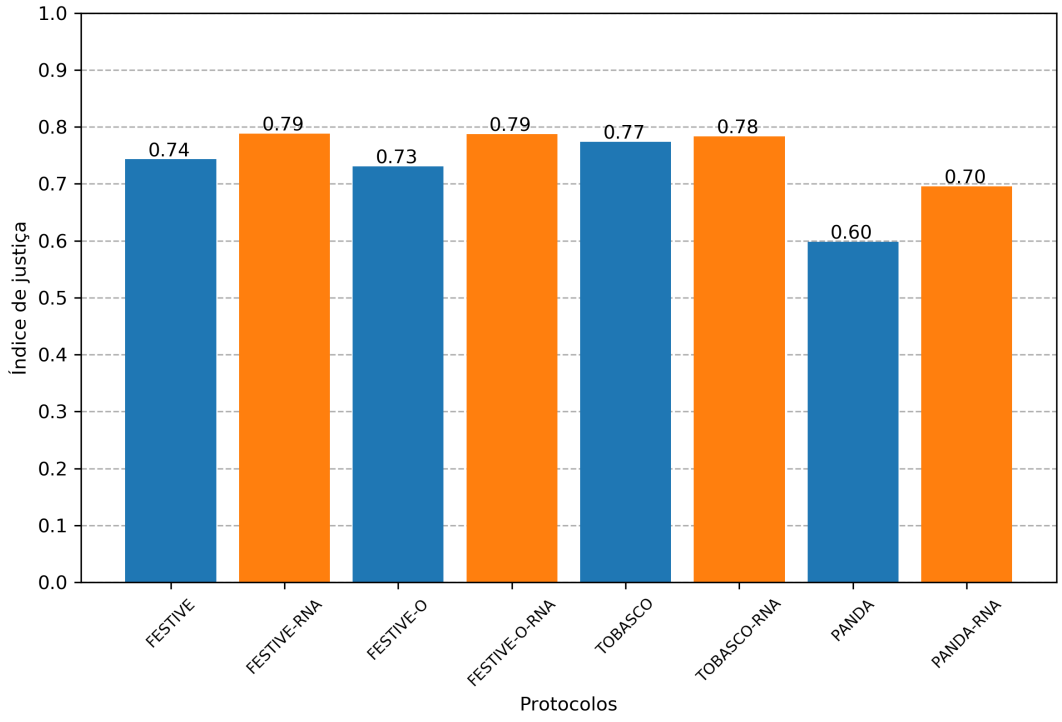


Figura 5.12: Médias da medida de justiça das simulações dos algoritmos de adaptação.

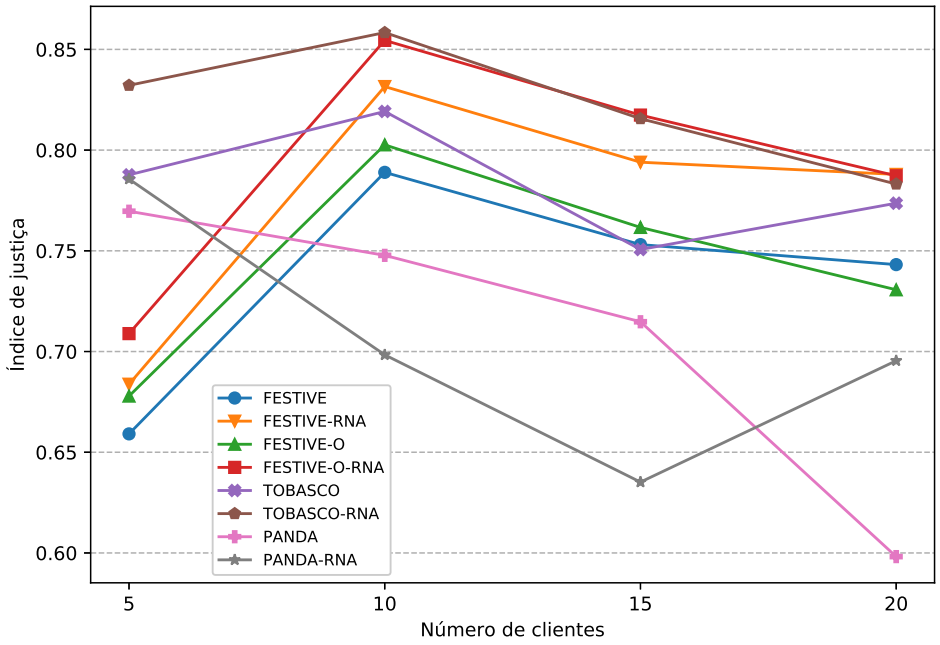


Figura 5.13: Medidas de justiça das simulações dos algoritmos de adaptação para diferentes quantidades de clientes.

Capítulo 6

Conclusão

Este trabalho visou explorar a ideia de uma sessão de *streaming* por meio do padrão aberto MPEG-DASH que pudesse estar consciente do tamanho dos segmentos individuais no seu processo de escolha de qual qualidade de vídeo a ser requerida por um programa cliente, independente do grau de variação de uma rede. A partir da ideia inicial de utilizar uma rede neural para fazer essa tomada de decisão com estimativas dos tamanhos de segmentos e banda disponível, foi-se decidido por testes quanto a capacidade de estimar os tamanhos e qual o impacto disso em algoritmos de adaptação existentes.

Para alcançar o objetivo, o trabalho propôs a utilização de redes neurais que poderiam prover a informação dos tamanhos de forma que os envolvidos no processo (servidor ou cliente) não precisassem ter todos os dados dos segmentos disponíveis inicialmente. E a partir das saídas geradas pelas redes neurais, um algoritmo de adaptação (que toma as decisões de um programa cliente) seria possibilitado de oferecer uma melhor experiência para um usuário em uma sessão de *streaming*, adaptando-se a taxa de transmissão variante de uma rede.

Os resultados obtidos ao se analisar diversas rodadas de treinamentos e testes das RNAs mostraram que a arquitetura LSTM se tornou mais útil para gerar saídas com dados já disponíveis (no caso dos dados de treinamento) do que uma rede Elman (RNRE) que conseguiu abstrair e obter menor erro na fatia menor de dados de teste. Isso significa que, apesar do estudo ter utilizado a LSTM, há potencial para utilização da Elman para casos onde exista maior quantidade de dados e seja necessário abstrair dados de outras qualidades para um vídeo que ainda não tenha sido utilizado no treinamento da rede. Além disso, o fato do Nadam utilizar o tempo t presente para atualizar os deltas das sinapses não mostrou vantagem em relação a otimização já estabelecida do Adam.

Quanto a comparação entre os algoritmos de adaptação originais e os modificados, pode-se dizer que não houveram vantagens tão grandes que valeram o uso das redes neurais limitadas aos dados disponíveis, em vez de somente extrair as informações de

tamanhos diretamente dos segmentos e disponibilizar para os clientes. Isso porque a qualidade média selecionada encontrou uma piora, as melhoras que ocorreram por outros métricas não foram tão significativas, e, além de tudo, o treinamento das RNAs adicionam um maior custo de recursos.

Porém, ainda assim, pode-se observar que os algoritmos FESTIVE mostraram melhoras quanto a justiça e estabilidade, seus principais pontos de apelo; enquanto o PANDA demonstrou potencial para mais justiça no caso em que ainda mais clientes estejam dividindo a banda. O potencial dos algoritmos de adaptação com conhecimento dos tamanhos de segmentos não pode ser descartado. E, além disso, PANDA tendo esse conhecimento em ambientes com muitos clientes mostrou-se como uma possível base de um futuro estudo sobre seu potencial de superar o FESTIVE em termos de justiça ao dividir a banda, com ainda mais eficiência, o que pode ser útil para uso em transmissões de eventos ao vivo com grande público, especialmente com a adoção de maiores resoluções.

Referências

[1] *Backpropagation*. https://en.wikipedia.org/wiki/Backpropagation, acesso em 2018-08-02. ix, 10

[2] Orr, Genevieve, Nici Schraudolph e Fred Cummins: *CS-449: Neural Networks; Momentum and Learning Rate Adaptation*, 1999. https://www.willamette.edu/{~}gorr/classes/cs449/momrate.html. ix, 13

[3] Graves, Alex: *Supervised Sequence Labelling*. páginas 5–13, 2012, ISSN 01406736. http://link.springer.com/10.1007/978-3-642-24797-2{_}2. ix, 17, 18, 72

[4] (Bitmovin), Christopher Mueller: *Microsoft Smooth Streaming*, 2015. https://bitmovin.com/microsoft-smooth-streaming-mss/, acesso em 2018-06-06. ix, 28, 29

[5] (Apple): *HTTP Live Stream Overview*. https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/Introduction/Introduction.html. ix, 29

[6] Sodagar, Iraj: *The MPEG-dash standard for multimedia streaming over the internet*. IEEE Multimedia, 18(4):62–67, 2011, ISSN 1070986X. http://ieeexplore.ieee.org/xpl/login.jsp?tp={&}arnumber=6077864. ix, 31, 33

[7] Ott, Harald, Konstantin Miller e Adam Wolisz: *Simulation Framework for HTTP-Based Adaptive Streaming Applications*. Workshop on ns-3 WNS3 2017, páginas 95–102, 2017. ix, 42, 47, 48

[8] (Cisco): *Cisco Visual Networking Index : Forecast and*. Relatório Técnico, Cisco, 2017. 1

[9] (Sandvine): *Sandvine Global Internet Phenomena Report*. Relatório Técnico, Sandvine, Waterloo,ON, Canada, 2018. https://www.sandvine.com/downloads/general/global-internet-phenomena/2015/global-internet-phenomena-report-latin-america-and-north-america.pdf. 1

[10] Ozer, Jan: *Video Compression for Flash, Apple Devices and HTML5*. Doceo Publishing, Galax, VA, 1st edição, 2011, ISBN 9780976259503. 1, 20, 25, 26, 28

[11] May, W.: *HTTP Live Streaming*. Relatório Técnico, RFC Editor, aug 2017. https://www.rfc-editor.org/info/rfc8216. 1, 28

[12] (Adobe): *FAQ HTTP Dynamic Streaming*, 2018. https://www.adobe.com/products/hds-dynamic-streaming/faq.html, acesso em 2018-06-06. 1, 26

[13] ISO/IEC: *23009-1:2014 - Dynamic adaptive streaming over HTTP (DASH): Media presentation description and segment formats*. Relatório Técnico, ISO/IEC, apr 2014. https://www.iso.org/standard/65274.html. 1, 31, 32

[14] Gadaleta, Matteo, Federico Chiariotti, Michele Rossi e Andrea Zanella: *D-DASH: a Deep Q-learning Framework for DASH Video Streaming*. IEEE Transactions on Cognitive Communications and Networking, 3(4):1–1, 2017, ISSN 2332-7731. http://ieeexplore.ieee.org/document/8048013/. 2

[15] Lekharu, Anirban, Satish Kumar, Arijit Sur e Arnab Sarkar: *A QoE Aware LSTM based Bit-Rate Prediction Model for DASH Video*. (i):392–395, 2018. 2, 39

[16] Xue, Yuanyi, Beril Erkin e Yao Wang: *A Novel No-Reference Video Quality Metric for Evaluating Temporal Jerkiness due to Frame Freezing*. IEEE Transactions on Multimedia, 17(1):134–139, 2015, ISSN 15209210. 2, 38

[17] Valderrama, Diego José Luis Botia e Natalia Gaviria Gómez: *Non-Intrusive Method Based on Neural Networks for Video Quality of Experience (QoE) Assessment*. 2016(iii), 2016, ISSN 16875699. 2, 38

[18] Russell, Stuart J e Peter Norvig: *Artificial Intelligence: A Modern Approach*. Prentice Hall, New Jersey, 3rd edição, 2010, ISBN 9780131038059. http://portal.acm.org/citation.cfm?id=773294. 4

[19] Braga, Antônio Pádua: *Redes neurais artificiais: teoria e aplicações*. LTC Editora, 2007, ISBN 9788521615644. https://books.google.com.br/books?id=R-p1GwAACAAJ. 4

[20] Mcculloch, Warren S e Walter Pitts: *a Logical Calculus of the Ideas Immanent in Nervous Activity**. Bulletin of Mathematical Biology, 52(l2):99–115, 1990. http://www.cs.cmu.edu/{~}./epxing/Class/10715/reading/McCulloch.and.Pitts.pdf. 5

[21] Haykin, S: *Neural Networks: A Comprehensive Foundation*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edição, 1998, ISBN 0132733501. 5, 9

[22] Werbos, P. J.: *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Tese de Doutoramento, Harvard University, 1974. 11

[23] Rumelhart, David E, Geoffrey E Hinton e Ronald J Williams: *Learning representations by back-propagating errors*. Nature, 323(6088):533–536, 1986, ISSN 1476-4687. https://doi.org/10.1038/323533a0. 11

[24] *lasagne.updates*, 2015. http://lasagne.readthedocs.io/en/latest/modules/updates.html, acesso em 2018-08-02. 12

[25] *Solver/Model Optimization*. http://caffe.berkeleyvision.org/tutorial/solver.html, acesso em 2018-08-02. 12

[26] *Optimizers*. https://keras.io/optimizers/, acesso em 2018-08-02. 12

[27] Qian, Ning: *On the Momentum Term in Gradient Descent Learning Algorithms The Momentum Term in Gradient Descent*. Neural Networks: The Official Journal of the International Neural Network Society, 5213(12(1)):145–151, 1999. 12

[28] Nesterov, Yurii: *A Method of Solving A Convex Programming Problem With Convergence rate O(1/kˆ2)*. Soviet Mathematics Doklady, 1983. 12

[29] Hinton, Geoff: *Neural Networks for Machine Learning, Lecture 6a: Overview of mini-batch gradient descent*, 2012. http://www.cs.toronto.edu/{~}tijmen/csc321/slides/lecture{_}slides{_}lec6.pdf. 13

[30] Kingma, Diederik P. e Jimmy Ba: *Adam: A Method for Stochastic Optimization*. páginas 1–15, 2014, ISSN 09252312. http://arxiv.org/abs/1412.6980. 13

[31] Dozat, Timothy: *Incorporating Nesterov Momentum into Adam*. ICLR Workshop, (1):2013–2016, 2016. 14

[32] Cybenko, George: *Approximations by superpositions of sigmoidal functions*. Approximation Theory and its Applications, 1989, ISSN 10009221. 15

[33] Cybenko, George: *Continuous Valued Neural Networks with Two Hidden Layers are Sufficient*. Relatório Técnico, University of Illinois. Center for Supercomputing Research and Development, 1988. 15

[34] Elman, Jeffrey L.: *Finding structure in time*. Cognitive Science, 1990, ISSN 03640213. 15

[35] Bengio, Yoshua, Patrice Simard e Paolo Frasconi: *Learning Long-Term Dependencies with Gradient Descent is Difficult*. IEEE Transactions on Neural Networks, 1994, ISSN 19410093. 16

[36] Hochreiter, Sepp e J Urgen Schmidhuber: *Long Short-Term Memory*. Neural Computation, 9(8):1735–1780, 1997, ISSN 0899-7667. http://www7.informatik.tu-muenchen.de/{~}hochreit{%}5Cnhttp://www.idsia.ch/{~}juergen. 16

[37] Greff, Klaus, Rupesh K. Srivastava, Jan Koutnik, Bas R. Steunebrink e Jurgen Schmidhuber: *LSTM: A Search Space Odyssey*. IEEE Transactions on Neural Networks and Learning Systems, 28(10):2222–2232, 2017, ISSN 21622388. 17

[38] Kurose, James F. e Keith W. Ross: *Redes de computadores e a Internet: uma abordagem top-down*. Pearson Addison Wesley, 5th edição, 2010. 19

[39] Forouzan, Behrouz A.: *Data communications and networking*. McGraw-Hill, New York, NY, 5th ed. edição, 2013, ISBN 978-0-07-337622-6. 20

[40] Pfeiffer, Silvia: *The Definitive Guide to HTML5 Video*. Apress, New York, NY, 1st edição, 2010, ISBN 9781430230908. 20, 25

[41] Schulzrinne, H, S Casner, R Frederick e V Jacobson: *RTP: A Transport Protocol for Real-Time Applications*. Std 64, RFC Editor, jul 2003. http://www.rfc-editor.org/rfc/rfc3550.txt. 20, 21

[42] Schulzrinne, H., A. Rao e R. Lanphier: *Real Time Streaming Protocol (RTSP)*. Relatório Técnico, RFC Editor, apr 1998, ISBN 9788578110796. https://www.rfc-editor.org/info/rfc2326. 20, 21

[43] Wijering, Jeroen: *What is Video Streaming?*, 2011. http://www.longtailvideo.com/blog/19578/what-is-video-streaming, acesso em 2018-06-06. 20, 25

[44] Tanenbaum, Andrew S: *Redes de Computadores*. Editora Campus, 4th edição, 2003, ISBN 8535211853. 21

[45] Schulzrinne, H. e S. Casner: *RTP Profile for Audio and Video Conferences with Minimal Control*. Relatório Técnico, RFC Editor, jul 2003. http://www.rfc-editor.org/rfc/rfc3551.txt. 21

[46] Stockhammer, Thomas: *Dynamic Adaptive Streaming over HTTP: Standards and Design Principles*. Proceedings of the Second Annual ACM Conference on Multimedia Systems, 2014(i):133–144, 2011, ISSN 1450305180. http://doi.acm.org/10.1145/1943552.1943572. 23

[47] Fielding, Roy T, James Gettys, Jeffrey C Mogul, Henrik Frystyk Nielsen, Larry Masinter, Paul J Leach e Tim Berners-Lee: *Hypertext Transfer Protocol – HTTP/1.1*. Rfc 2616, RFC Editor, jun 1999. http://www.rfc-editor.org/rfc/rfc2616.txt. 24

[48] Foundation), (The Apache Software: *Apache*. https://www.apache.org/, acesso em 2018-08-15. 26

[49] Hassoun, David (Adobe): *Dynamic streaming in Flash Media Server 3.5 – Part 1: Overview of the new capabilities | Adobe Developer Connection*. Developer Connection, página 5, 2009. https://www.adobe.com/devnet/adobe-media-server/articles/dynstream{_}advanced{_}pt1.html{#}articlecontentAdobe{_}numberedheader. 26

[50] (On2): *Advantages of TrueMotion VP6 Technology*. 2004. 26

[51] (Adobe): *Adobe Flash Video File Format Specification Version 10.1*. Relatório Técnico, 2010. http://download.macromedia.com/f4v/video{_}file{_}format{_}spec{_}v10{_}1.pdf. 26

[52] Figueroa, Amparito Alexandra Morales: *Pré-Busca de Conteúdo em Apresentações Multimídia*. Dissertação de mestrado, Pontifícia Universidade Católica do Rio de Janeiro, 2014. 26, 29

[53] (Adobe): *What's New in Adobe Media Server*, 2015. https://helpx.adobe.com/adobe-media-server/tech-overview/whats-new-ams.html, acesso em 2018-06-06. 27

[54] (Microsoft), Silverlight team: *2010 Winter Olympics Online hits new highs for engagement times , experience quality and monetization*, 2010. https://blogs.msdn.microsoft.com/silverlight/2010/05/17/2010-winter-olympics-online-hits-new-highs-for-engagement-times-experience-qualit acesso em 2018-06-06. 27

[55] Deutscher, John (Microsoft): *Smooth Streaming Primer*. Relatório Técnico, Microsoft, aug 2010. https://docs.microsoft.com/en-us/iis/media/smooth-streaming/smooth-streaming-primer. 28

[56] Roger Pantos (Apple): *What's new in HTTP Live Streaming*, 2016. https://developer.apple.com/videos/play/wwdc2016/504/, acesso em 2018-06-06. 30

[57] Akhshabi, Saamer, Sethumadhavan Narayanaswamy, Ali C. Begen e Constantine Dovrolis: *An experimental evaluation of rate-adaptive video players over HTTP*. Signal Processing: Image Communication, 27(4):271–287, 2012, ISSN 09235965. http://dx.doi.org/10.1016/j.image.2011.10.003. 34

[58] Varma, Subir: *Flow control in video applications*. Em Kaufmann, Morgan (editor): *Internet Congestion Control*, capítulo 6, páginas 173–202. Elsevier, Waltham, MA, 2015, ISBN 9780128036006. 35

[59] Jiang, Junchen, Vyas Sekar e Hui Zhang: *Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with festive*. IEEE/ACM Transactions on Networking, 22(1):326–340, 2012, ISSN 10636692. 37, 48, 54, 55, 57

[60] Miller, Konstantin: *Adaptation Algorithms for HTTP-Based Video Streaming*. (November 2016), 2016. 37, 55

[61] Li, Zhi, Xiaoqing Zhu, Joshua Gahm, Rong Pan, Hao Hu, Ali C. Begen e David Oran: *Probe and adapt: Rate adaptation for HTTP video streaming at scale*. IEEE Journal on Selected Areas in Communications, 32(4):719–733, 2014, ISSN 07338716. 38, 48, 54, 55, 57

[62] Covell, Michele, Martín Arjovsky, Yao chung Lin e Anil Kokaram: *Optimizing Transcoder Quality Targets Using a Neural Network with an Embedded Bitrate Model*. Electronic Imaging, 2016(2):1–7, 2016, ISSN 2470-1173. http://www.ingentaconnect.com/content/10.2352/ISSN.2470-1173.2016.2.VIPC-237. 39

[63] (Blender): *Big Buck Bunny*. https://peach.blender.org/, acesso em 2018-08-13. 43, 49

[64] (Blender): *Elephant's Dream*. https://orange.blender.org/, acesso em 2018-08-13. 43

[65] (Blender): *Tears of Steel*. https://mango.blender.org/, acesso em 2018-08-13. 43

[66] Nations), (United: *Of Forests and Men*, 2011. http://www.offorestsandmen.org/, acesso em 2018-08-13. 43

[67] (Blender): *Home of the Blender project - Free and Open 3D creation software*, 2019. `https://www.blender.org/`, acesso em 2019-05-09. 43

[68] *Keras*. `https://keras.io`, acesso em 2018-08-06. 44

[69] *ns-3*, 2011. `https://www.nsnam.org/`, acesso em 2018-08-20. 47

Apêndice A

Equações de uma rede LSTM

Este apêndice provê as equações utilizadas em uma rede *Long Short-Term Memory* (LSTM) divididas nas duas fases de aprendizado, analogamente às expostas por Graves [3]: o cálculo das saídas e a retropropagação dos erros.

Para um bloco j no tempo t, sua entrada é definida como a_j^t e sua saída como b_j^t. Os subscritos ι, ϕ e ω referem-se aos portões de entrada, de esquecimento e de saída. O subscrito c referem-se a célula de memória do bloco, com s_c^t definindo o valor mantido por ela. As funções f são as que definem as funções de ativação dos portões de um bloco, já as g e h são, respectivamente, funções de ativação da entrada e da saída do bloco.

I, K e H representam o número de entradas, saídas e células da camada escondida; com seus índices análogos i, k e h. E o termo G representará o número total de entradas na camada escondida, tendo o índice g para referir às entradas de uma maneira mais genérica.

A.1 Cálculo de saídas

Portões de entrada

$$a_\iota^t = \sum_{i=1}^{I} w_{i\iota} x_i^t + \sum_{h=1}^{H} w_{h\iota} b_h^{t-1} + \sum_{c=1}^{C} w_{c\iota} s_c^{t-1} \tag{A.1}$$

$$b_\iota^t = f(a_\iota^t) \tag{A.2}$$

Portões de esquecimento

$$a_\phi^t = \sum_{i=1}^{I} w_{i\phi} x_i^t + \sum_{h=1}^{H} w_{h\phi} b_h^{t-1} + \sum_{c=1}^{C} w_{c\phi} s_c^{t-1} \tag{A.3}$$

$$b_\phi^t = f(a_\phi^t) \tag{A.4}$$

Células

$$a_c^t = \sum_{i=1}^{I} w_{i\phi} x_i^t + \sum_{h=1}^{H} w_{h\phi} b_h^{t-1} \tag{A.5}$$

$$s_c^t = b_\phi^t s_c^{t-1} + b_\iota^t g(a_c^t) \tag{A.6}$$

Portões de saída

$$a_\omega^t = \sum_{i=1}^{I} w_{i\phi} x_i^t + \sum_{h=1}^{H} w_{h\phi} b_h^{t-1} + \sum_{c=1}^{C} w_{c\phi} s_c^t \tag{A.7}$$

$$b_\omega^t = f(a_\omega^t) \tag{A.8}$$

Saídas

$$b_c^t = b_\omega^t h(s_c^t) \tag{A.9}$$

A.2 Retropropagação dos erros

$$\epsilon_c^t \stackrel{\text{def}}{=} \frac{\partial E}{\partial b_c^t} \tag{A.10}$$

$$\epsilon_s^t \stackrel{\text{def}}{=} \frac{\partial E}{\partial s_c^t} \tag{A.11}$$

Saídas das células

$$\epsilon_c^t = \sum_{k=1}^{K} w_{ck} \delta_k^t + \sum_{g=1}^{G} w_{cg} \delta_c^{t+1} \tag{A.12}$$

Portões de saída

$$\delta_\omega^t = f'(a_\omega^t) \sum_{c=1}^{C} h(s_c^t) \epsilon_c^t \tag{A.13}$$

Estados

$$\epsilon_s^t = b_\omega^t h'(s_c^t) \epsilon_c^t + b_\phi^{t+1} \epsilon_s^{t+1} + w_{c\iota}^t \delta_\iota^{t+1} + w_{c\phi}^t \delta_\phi^{t+1} + w_{c\omega}^t \delta_\omega^t \tag{A.14}$$

Células

$$\delta_c^t = b_\iota^t g'(a_c^t) \epsilon_s^t \tag{A.15}$$

Portões de saída

$$\delta_\phi^t = f'(a_\phi^t) \sum_{c=1}^{C} s_c^{t-1} \epsilon_s^t \tag{A.16}$$

Portões de entrada

$$\delta_\iota^t = f'(a_\iota^t) \sum_{c=1}^{C} g(a_c^t) \epsilon_s^t \tag{A.17}$$