Estimativa de Qualidade de Serviço em Estações Wi-Fi Utilizando Redes Neurais Recorrentes

Gilson Miranda Jr.[1], **Henrique D. Moura**[1], **Matheus H. N. Nunes**[1], **Luiz H. A. Correia**[2], **Daniel F. Macedo**[1]

[1] Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte – MG – Brazil

[2]Departamento de Ciência da Computação
Universidade Federal de Lavras (UFLA) – Lavras – MG – Brazil

{gilsonmiranda, henriquemoura, mhnnunes, damacedo}@dcc.ufmg.br,
lcorreia@dcc.ufla.br

Abstract. *Wireless networks are the most common way to access the Internet, with more than 10 billion Wi-Fi devices already sold. Wireless connections suffer from spectrum overuse, transmission errors, and loss of information. Intelligent control systems can be used for network management and to improve Quality of Service (QoS). However, the first step towards such systems is a way to relate current Wi-Fi settings to a QoS value. This work proposes a model using Recurrent Neural Networks (RNN) to infer such a relation, based on real Wi-Fi data, and compares two RNN types: Gated Recurrent Unit (GRU) and Long Short Term Memory (LSTM). The model predicts four network metrics: throughput, loss, delay, and jitter, based only in traffic data obtained at the AP. The average Root Mean Square Error (RMSE) is of the order of* 10^{-2} *for throughput,* 10^{-4} *for delay, and* 10^{-5} *for jitter and packet loss using both methods.*

Resumo. *As redes sem fio são a maneira mais comum de acessar a Internet, com mais de 10 bilhões de dispositivos Wi-Fi já vendidos. As conexões sem fio sofrem problemas relacionados ao uso excessivo do espectro, a erros de transmissão e a perda de informações. Sistemas inteligentes de controle são uma alternativa para o gerenciamento eficiente da rede, visando à melhoria da Qualidade de Serviço (QoS). No entanto, o primeiro passo é obter uma função que relaciona as configurações de Wi-Fi atuais a um valor de QoS esperado. Este trabalho propõe um modelo que utiliza RNN para inferir tal relação, baseado em dados reais de Wi-Fi. Ele compara duas RNNs: GRU e LSTM. O modelo prevê quatro métricas de rede: vazão, perda, atraso e variação do atraso (jitter), com base apenas nos dados de tráfego obtidos no AP. O RMSE médio é da ordem de* 10^{-2} *para vazão, de* 10^{-4} *para o atraso, e de* 10^{-5} *para a jitter e para a perda de pacotes para ambos os métodos.*

1. Introdução

As redes sem fio IEEE 802.11 (conhecidas como Wi-Fi) se tornaram comuns nas duas últimas décadas em ambientes como escritórios e universidades, e são a forma mais utilizada para acesso à Internet em redes domésticas [DiCioccio et al. 2013]. Estudos recentes

mostram que mais de dez bilhões de dispositivos Wi-Fi já foram vendidos, e mais de 4,5 bilhões deles estão em uso atualmente [Biswas et al. 2015].

Apesar da praticidade de uso, as redes sem fio estão sujeitas a problemas de desempenho como atrasos e perdas de informação, geralmente causados por interferências no sinal, concorrência no meio ou baixa potência de transmissão. A análise de métricas de desempenho dos dispositivos e da rede pode indicar uma degradação do desempenho, o que pode ser usado para refinar a configuração da rede.

Um dos desafios para realizar o controle de uma rede sem fio é obter um modelo para prever o desempenho da rede [Moura et al. 2019]. Tais modelos são importantes, pois fornecem ao administrador uma ferramenta para validar novas configurações antes que elas sejam aplicadas na rede alvo [Fazio et al. 2013]. Assim, evita-se o inconveniente de um desempenho degradado devido a uma decisão incorreta. Por exemplo, em [Moura et al. 2019], os autores utilizam um modelo de predição que indica qual seria o desempenho esperado para cada ação possível. A partir desta informação, a ação escolhida é aquela que maximiza o desempenho previsto. Já [Fazio et al. 2013] usam um modelo de previsão como parte de um algoritmo de *handover*.

Neste trabalho utilizamos Redes Neurais Recorrentes (RNN - *Recurrent Neural Networks*) para a construção de preditores de QoS para clientes Wi-Fi. Os preditores propostos utilizam somente dados de tráfego que podem ser obtidos no AP, evitando a instalação de ferramentas de monitoramento nos clientes. Foram avaliados preditores baseados em LSTM (*Long Short-Term Memory*) e GRU (*Gated Recurrent Unit*), comparando os resultados obtidos utilizando RMSE (*Root Mean Square Error*). Este é um primeiro passo para a construção de um sistema de controle para APs que monitore a QoS de fluxos individuais, e possa ajustar as configurações para melhorar a QoS. As principais contribuições deste trabalho são: (1) Uso de RNN (empregando GNU e LSTM) para a inferência de métricas de QoS (vazão, latência, *jitter* e perda) em estações Wi-Fi; e (2) Avaliação da precisão da predição e generalização dos modelos, treinados com diferentes hiperparâmetros e configurações. Este é um primeiro passo para a construção de um sistema de controle que monitora a QoS para fluxos individuais e ajusta as configurações de rede nos APs para melhorar a QoS dos clientes. Os modelos propostos podem ser usados em aplicações de controle que precisam prever o valor futuro da métrica de QoS, por exemplo, em *traffic shaping* e *traffic throttling*, ou alterar parâmetros de configuração da rede sem fio de uma determinada classe de serviço para atender a requisitos de SLA (*Service Level Agreement*).

Foram realizadas predições sobre um subconjunto do *dataset* que não fez parte da fase de treinamento dos preditores. O RMSE médio é da ordem de 10^{-2} para vazão, de 10^{-4} para o atraso, e de 10^{-5} para *jitter* e para a perda de pacotes com ambos os métodos. Os resultados mostram que os modelos são resistentes à remoção de dados de estações do conjunto de treino, sendo capazes de generalizar os resultados para estações que não fizeram parte do treinamento, mantendo a mesma ordem de grandeza do erro.

O restante deste trabalho está organizado como segue. A Seção 2 apresenta o referencial teórico. A seção seguinte mostra os trabalhos relacionados. A Seção 4 descreve a metodologia. Os resultados obtidos são mostrados na Seção 5. Na Seção 6 apresentamos as conclusões e trabalhos futuros.

2. Referencial Teórico

Esta seção apresenta alguns conceitos fundamentais: Qualidade de Serviço e Redes Neurais Recorrentes.

2.1. Qualidade de Serviço – QoS

A RFC 2386 define QoS como um conjunto de requisitos de serviço a serem atendidos durante a transmissão de um fluxo de pacotes da origem ao destino [Crawley et al. 1998]. Desta forma, QoS refere-se à capacidade da rede em garantir um certo nível de desempenho para um fluxo sob uma perspectiva técnica, baseada em parâmetros da rede. Por exemplo, aplicações multimídia diferentes podem possuir diferentes requisitos de QoS, como atraso, largura de banda, *jitter* e confiabilidade [Ehsan and Hamdaoui 2012]. Segundo [Gomes et al. 2009], existem diferentes mecanismos e algoritmos que fornecem QoS, e que atuam principalmente em duas camadas do modelo OSI: aplicação e rede.

De acordo com [Gozdecki et al. 2003], a QoS em redes IP deve considerar pelo menos os seguintes parâmetros: i) vazão (ou *bitrate*) da comunicação; ii) atraso (ou *delay*) dos pacotes, podendo considerar tanto o caminho da origem ao destino, quanto por subseções da rede; iii) *jitter*, ou variação de atraso, também podendo considerar origem ao destino ou subseções; iv) taxa de perda de pacotes.

2.2. Redes Neurais Recorrentes

Uma rede neural recorrente (RNN) é uma rede neural capaz de capturar dinâmicas de sequências através de ciclos na rede de nós. Tais ciclos, ou estruturas recorrentes, permitem aos nós ocultos observarem suas próprias saídas em tempos prévios, o que fornece um tipo de "memória" para a rede [Elman 1990]. A entrada para a RNN geralmente é uma sequência de comprimento variável $x = \{x_1, ..., x_T\}$ e a rede é treinada para predizer o próximo símbolo da sequência, ao aprender uma distribuição de probabilidade sobre os símbolos, dados os valores anteriores $p(x_t|x_{t-1}, ..., x_1)$. [Cho et al. 2014] Neste trabalho foram utilizados dois tipos de RNN: LSTM [Gers et al. 1999] e GRU [Cho et al. 2014], e cada uma destas arquiteturas é detalhada nas próximas subseções.

2.2.1. GRU - *Gated Recurrent Unit*

A GRU é uma RNN em que unidades recorrentes capturam de forma adaptativa as dependências de diferentes escalas de tempo [Skansi 2018]. Uma rede GRU é composta por unidades GRU encadeadas, que tratam uma sequência de entradas consecutivas correspondentes à extensão da rede GRU. A Figura 1a apresenta uma unidade GRU (ou bloco de memória). A figura identifica as portas ("*gates*") de entrada e saída, e como estas são relacionadas pelas funções. As setas indicam a direção de propagação dos dados.

Cada unidade GRU avalia o conteúdo da memória da unidade anterior ($\mathbf{h}_{t-1}$) e o novo dado de entrada ($\mathbf{x}_t$). Uma GRU redefine (utilizando a porta de redefinição $\mathbf{r}_t^j$) ou atualiza (utilizando a porta de atualização $\mathbf{z}_t^j$) de forma adaptável seu conteúdo de memória. A porta de redefinição define a forma em que a nova entrada será combinada com o conteúdo da GRU, enquanto a porta de atualização define o quanto manter do estado oculto da unidade anterior $\mathbf{h}_{t-1}$. Ambas utilizam a função sigmoide $\sigma(\cdot)$, limitada

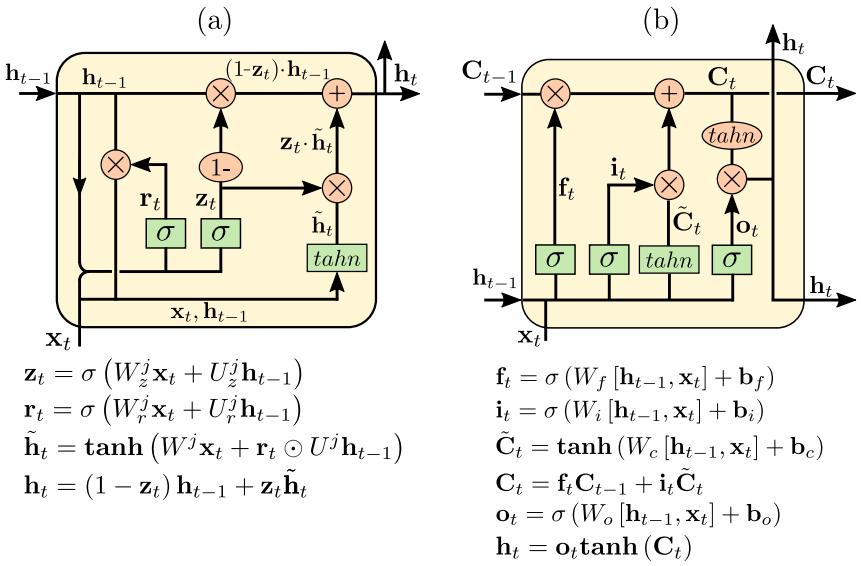

Figura 1. (a) Estrutura de uma unidade GRU; (b) Estrutura de uma unidade LSTM.

entre 0 e 1. Assim, $\mathbf{r}_t$ e $\mathbf{z}_t$ definem quais entradas terão maior relevância (i.e., valores mais próximos de 1) e quais terão menor relevância (i.e, valores mais próximos de 0).

O valor de $\tilde{\mathbf{h}}_t$, que é o novo valor (candidato) de memória, é calculado à partir do estado da unidade anterior $\mathbf{h}_{t-1}$, combinado à saída da porta de redefinição ($\mathbf{r}_t$), e aos dados de entrada $\mathbf{x}_t$. É utilizada a função de ativação **tanh** para normalizar os resultados entre $[-1, 1]$. A operação $\odot$ indica uma multiplicação elemento a elemento dos vetores. A cada passo t, é calculado o estado $\mathbf{h}_t^j$ da j-ésima GRU, onde $\mathbf{z}_t$ controla qual será o peso do estado anterior $\mathbf{h}_{t-1}$, e qual será o peso da entrada atual $\tilde{\mathbf{h}}_t$ na composição do novo estado $\mathbf{h}_t^j$. As matrizes de pesos são dadas por W e U, e seus valores são aprendidos durante o treinamento.

2.2.2. LSTM - *Long Short-Term Memory*

LSTM é uma variação de rede recorrente que resolve o problema do desaparecimento de gradiente ([Hochreiter and Schmidhuber 1997, Gers et al. 1999]). Uma unidade LSTM opera como uma célula de memória, onde seu conteúdo é alterado por meio de portas. As portas controlam o funcionamento da unidade, e seus comportamentos são determinados pelos pesos, que são ajustados no treinamento. Neste processo, as unidades LSTM de cada *timestep* (janela de treinamento) aprendem quando devem permitir que os dados entrem, saiam ou sejam excluídos da unidade.

Uma rede LSTM possui uma ou mais unidades conectadas recorrentemente. A Figura 1b mostra uma unidade LSTM com suas três portas: entrada, saída e redefinição, que fornecem operações de gravação (dada pela função $\mathbf{i}_t$), saída (dada pela função $\mathbf{o}_t$) e redefinição (dada pela função $\mathbf{f}_t$). Começando pela esquerda da figura, as setas mostram que a informação flui para dentro da unidade em três pontos. A entrada é formada pela combinação da entrada atual ($\mathbf{x}_t$), o estado oculto (conteúdo de memória) da unidade anterior ($\mathbf{h}_{t-1}$) e a saída da unidade anterior ($\mathbf{C}_{t-1}$). Aqui há uma distinção entre o estado $\mathbf{C}_t$ e o estado oculto $\mathbf{h}_t$. A operação das portas define em quanto o estado $\mathbf{C}_t$ será alterado

na unidade LSTM no *timestep* t, sendo possível que a rede mantenha informações inalteradas durante vários *timesteps*. Já o estado oculto da unidade atual, também considerado a saída da unidade, é baseado no novo estado de $\mathbf{C}_t$, e na operação da porta de saída $\mathbf{o}_t$, que define quais dados do estado serão retornados e com qual relevância.

3. Trabalhos Relacionados

O trabalho de [Bernaille et al. 2006] classifica fluxos TCP utilizando *clusters*. Apesar de não tratarem de QoS e não considerarem a correlação temporal dos fluxos, os autores apresentam alguns pressupostos que servem de guia para o desenvolvimento deste trabalho: (1) acesso a dados da camada de física, enlace, rede e transporte; e (2) acesso *online* aos cabeçalhos dos quadros.

Os autores [Gelenbe et al. 2002] propõem técnicas adaptativas para alterar uma condição de rede em função de um critério de QoS, como atraso, *jitter* e perda de pacotes. A proposta reconfigura rotas em uma rede cabeada utilizando aprendizado por reforço, visando a melhoria do desempenho para pacotes de voz. Este artigo se diferencia ao focar nas redes sem fio. [Pedras et al. 2018] propuseram um modelo de Qualidade de Experiência (QoE) para redes celulares que obteve valores de RMSE da ordem de 10%, contudo o modelo não considera que existe uma correlação temporal entre as medições, como proposto neste trabalho.

[White et al. 2018] empregam LSTM ou GRU para prever valores futuros de QoS em dispositivos IoT de baixa potência. Um *middleware* monitora o acesso de aplicações a *web-services*, prevê quando a QoS será degradada a ponto de não mais satisfazer um acordo de nível de serviço (SLA - *Service Level Agreement*). Com base nessa informação, o *middleware* determina ações para melhorar o QoS. A solução proposta demanda a colaboração dos nós participantes, por meio da transmissão de informações de QoS pelas aplicações para um monitor de desempenho.

A influência dos parâmetros de configuração de Wi-Fi na QoE de vídeo foram avaliados por [Paudel et al. 2014]. QoE é uma medida do prazer ou incômodo das experiências de um usuário com um serviço, e se concentra em toda a experiência de serviço com base nos requisitos de qualidade humana. Os autores fizeram testes controlados, com voluntários, para a classificação da QoE em *streaming* de vídeo, e em seguida construíram um modelo baseado em *Random Neural Network*. Por fim, utilizando simulações variaram parâmetros da camada MAC e parâmetros relacionados à interferência do sinal, e utilizaram o classificador para verificar a influência de tais parâmetros na avaliação média de QoE. Além de QoE, os autores também avaliaram a influência dos parâmetros sobre QoS, considerando as métricas de vazão, atraso, *jitter* e perda de pacotes. O trabalho de [Paudel et al. 2014] se assemelha à nossa proposta, dado que ele também trata da estimativa de métricas de qualidade em estações Wi-Fi, com base apenas em dados conhecidos pelo AP. No entanto, os trabalhos se diferem tanto na forma de aplicação de aprendizado de máquina, quanto na quantidade de parâmetros avaliados para realizar a estimativa de QoS das estações. A proposta destes autores considera 5 atributos e obtém um erro máximo de 11 %, enquanto este trabalho realiza seleção de atributos e obtém erros menores. Além disso, não utilizamos simulações, aumentando a aplicabilidade do trabalho.

Um preditor baseado em lógica *fuzzy* foi proposto em [Pokhrel et al. 2013]. Ele estima as distribuições de probabilidade normalizadas que correlacionam as métricas de

QoS (taxa de perda de pacotes, intermitência de perda de pacotes e *jitter*) com QoE. Entretanto, a proposta também demanda acesso a dados da aplicação no cliente e não há como afirmar que as funções do sistema *fuzzy* podem ser generalizadas.

Tabela 1. Comparação entre os trabalhos relacionados

Trabalho	Camada					Método	Correlação Temporal	Usa dados do cliente	Foco em redes sem fio
	Física	Enlace	Rede	Transporte	Aplicação				
[Gelenbe et al. 2002]		✓	✓	✓	✓		Não	Sim	Não
[Bernaille et al. 2006]	✓	✓	✓	✓		Clusterização	Não	Não	Não
[Pokhrel et al. 2013]	✓	✓				Lógica Fuzzy	Não	Sim	Sim
[Paudel et al. 2014]	✓	✓			✓	Random NN	Não	Sim	Sim (Wi-Fi)
[Pedras et al. 2018]		✓			✓	SVR	Não	Sim	Sim (4G)
[White et al. 2018]			✓	✓	✓	LSTM/GRU	Sim	Sim	Sim (IoT)
Este trabalho	✓	✓	✓	✓		LSTM/GRU	Sim	Não	Sim (Wi-Fi)

A Tabela 1 apresenta uma comparação entre os trabalhos relacionados e este trabalho. São identificadas as camadas da pilha de protocolos utilizadas para compor as características da previsão, o método de obtenção da métrica de desempenho e se são necessários dados coletados no cliente ou não. É indicado, ainda, se a métrica é obtida em redes sem fio e se o método considera a correlação temporal entre as medições. Nosso trabalho se destaca por utilizar métricas obtidas somente no AP e considerar a correlação temporal das medidas em uma rede sem fios.

4. Metodologia

Esta seção apresenta a metodologia para a obtenção do *dataset* em hardware real, bem como a construção e o treinamento dos preditores.

4.1. Construção do *Dataset*

Utilizamos um computador como ponto de acesso sem fio empregando uma placa de rede Ethernet gigabit, e placa de rede sem fio Qualcomm Atheros AR9485 (ath9k). Os clientes são nós Raspberry Pi versão 3, alocados no *testbed* FUTEBOL (http://futebol.dcc.ufmg.br). A disposição destes equipamentos no laboratório é mostrada na Figura 2. Foi utilizada uma rede cabeada independente para controle de execução e transferência de resultados, de modo que a rede sem fio entre os clientes e o AP fosse usada exclusivamente para o tráfego de teste. Essa separação é necessária para evitar a sobrecarga da conexão sem fio com tráfego de controle que não existiria em casos de uso reais. As estações são todas iguais, dessa forma as variações nos resultados ocorrem por fatores como distância e interferência. Isto facilita a análise, porém reduz a capacidade de generalização dos resultados apresentados neste trabalho. A utilização de uma rede mais heterogênea, incluindo estações móveis, será avaliada em trabalhos futuros. O ambiente de testes está na vizinhança de cerca de 50 ESSIDs (*Extended Service Set Identifier*) desta forma as coletas das sequências de dados é influenciada pelo tráfego destas redes.

A geração de tráfego entre estações e AP foi feita com o *iperf* (https://iperf.fr/) versão 3. Foram necessárias modificações na ferramenta para fazer a medição de *jitter* com tráfego TCP. Para simular tráfego de download das estações, o servidor *iperf* foi executado nas estações, e o cliente no AP. Cada rodada de coletas teve duração de 2

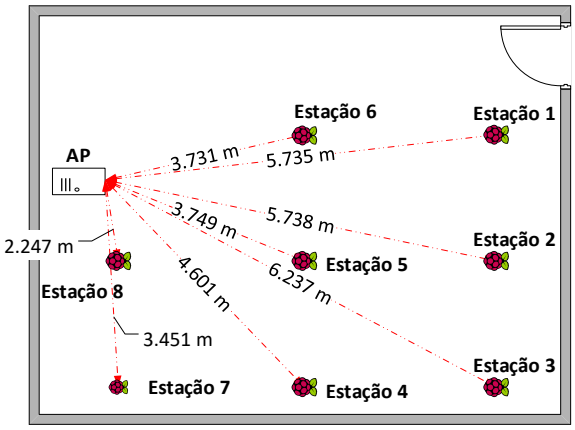

Figura 2. Disposição dos equipamentos no *testbed*.

minutos, com uma configuração diferente de canal (1 a 11), potência (1, 7 ou 15 dBm), tipo de tráfego (TCP ou UDP) e número de estações (1 a 8). Características adicionais dos fluxos foram obtidos no AP com o *tcpdump* (https://www.tcpdump.org/tcpdump_man.html) e com o comando *iw* (https://wireless.wiki.kernel.org/en/users/documentation/iw).

Antes de cada rodada de testes os relógios das estações e do AP eram sincronizados usando NTP. Os dados coletados com cada ferramenta foram combinados de acordo com o *timestamp* e o endereço MAC das estações. Os dados do *iperf* e do *iw* foram coletados em intervalos de 100 milissegundos, enquanto os dados do *tcpdump* foram coletados com intervalos de microssegundos. Os três conjuntos de dados foram agrupados em períodos de um segundo. Para cada tipo de dado coletado foi utilizado um critério para o agrupamento, sendo alguns agrupados pelo valor (e.g., canal, potência), pela média (e.g., vazão) ou pela soma dos valores no intervalo (e.g., número de pacotes perdidos). Os valores para a variável dependente y foram coletados nas estações, sendo quatro variáveis obtidas nos mesmos períodos de um segundo: vazão média, *jitter* médio, atraso médio e total de pacotes perdidos no período. As coletas de dados foram realizadas de 30 de Maio a 15 de Junho de 2017, totalizando 1.991.440 amostras, com 114 atributos.

4.2. Seleção Inicial de Atributos e *Feature Scaling*

O objetivo dessa seleção é reduzir o tempo de treinamento dos modelos sem afetar a qualidade do preditor. Dos 114 atributos coletados, alguns não apresentaram variação durante as coletas, como por exemplo o padrão de criptografia utilizado. Isso reduziu o *dataset* para 66 atributos, sobre os quais realizamos uma análise de correlação múltipla e uma Análise de Componentes Principais (ou PCA - *Principal Component Analysis*). Os atributos que apresentavam alta correlação (acima de 90%) foram agrupados, e aquele no grupo que apresentava maior importância no PCA foi selecionado, descartando-se os demais. Com essa análise reduzimos os atributos no *dataset* para 23. Após a análise da distribuição de cada variável independente (atributos, ou matriz $\mathbf{x}$), optou-se por utilizar uma normalização no intervalo $[0, 1]$ – quando a distribuição era mais uniforme entre os valores – ou uma normalização utilizando a distribuição normal – quando a distribuição apresentava um formato similar à distribuição normal. A lista completa dos atributos, a indicação de quais foram selecionados e o tipo de normalização pode ser obtida online em https://tinyurl.com/yyepptfk.

4.3. Formalização do Modelo

Foram utilizadas as implementações da biblioteca Keras versão 2.2.4. As arquiteturas foram montadas com uma ou mais camadas recorrentes, utilizando uma rede neural com duas camadas ocultas (MLP - *MultiLayer Perceptron*) na saída da última unidade recorrente. Foram testados, ainda, modelos de regressão multivariada não temporais, como por exemplo, *Support Vector Regression* e *MLP Regression*, bem como regressão multivariada usando GRU/LSTM. Estas abordagens apresentaram erros de predição muitas ordens de grandeza superiores, e por isso seus resultados não serão apresentados. Para cada métrica de QoS foi selecionado o modelo que apresentou melhores resultados com regressão univariada temporal, logo, cada métrica neste trabalho utiliza um modelo diferente.

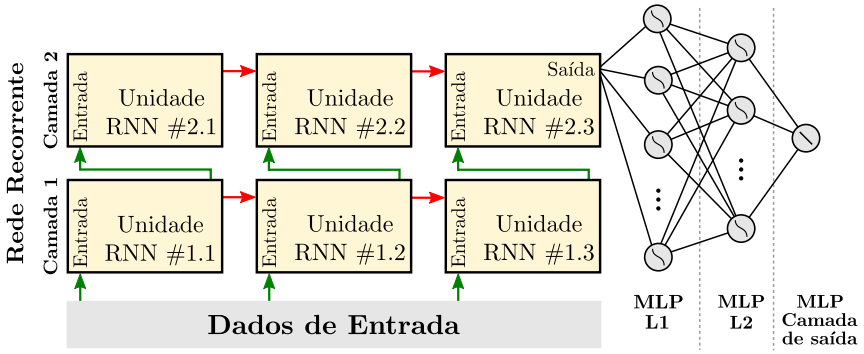

Figura 3. Forma geral dos modelos utilizados.

A Figura 3 ilustra a forma geral do modelo, com uma rede com duas camadas recorrentes. As camadas recorrentes são compostas por unidades recorrentes homogêneas, isto é, são todas GRU ou LSTM. A quantidade de unidades RNN na horizontal depende da quantidade de passos de tempo (*timesteps*) que serão analisados pela rede. No exemplo da Figura 3 são indicados três *timesteps* de entrada. Foram avaliadas redes com $n = \{1; 3; 5\}$ *timesteps*. A saída da unidade RNN inferior alimenta a unidade imediatamente superior, como mostrado na figura. A camada MLP é alimentada pela saída da última unidade recorrente. As saídas das demais unidades da camada superior das RNN não são utilizadas. As unidades recorrentes na mesma camada comunicam-se da esquerda para a direita. As unidades recorrentes foram configuradas com $\{4; 16; 32; 64; 128\}$ neurônios, e foram avaliadas configurações com $\{1; 2\}$ camadas recorrentes.

No diagrama apresentado na Figura 3, *Neurônios MLP L1* e *Neurônios MLP L2* indicam, respectivamente, a quantidade de neurônios na primeira e segunda camadas ocultas da MLP. Nestas camadas foram utilizadas funções de ativação *tanh*. Foi acrescentada uma camada de saída com apenas um neurônio, utilizando uma função de ativação *linear*, para combinar os resultados das camadas ocultas. Foram executados treinamentos com a primeira camada oculta da MLP contendo $\{16; 32; 64; 128; 256\}$ neurônios e a segunda camada contendo $\{4; 16; 32; 64; 128\}$ neurônios. Dessa forma, a rede final processa n amostras (linhas do *dataset*) de $\mathbf{x}$ e retorna o valor de $\mathbf{y}$, referente a um dos atributos mencionados anteriormente: vazão, latência, *jitter* ou perda de pacotes na estação.

Para reduzir a possibilidade de *overfitting*, foram configurados dois parâmetros de *dropout*: (1) o *dropout* – utilizado na transformação da entrada para a saída, e (2) *dropout recorrente* – na transformação do estado recorrente (de uma unidade para outra). Dropout é um método de regularização em redes neurais [Skansi 2018]. Este processo

consiste em ignorar conjuntos aleatórios de neurônios durante a fase de treino, i.e., os pesos relacionados a esses neurônios não são atualizados em um passo do treinamento. Foram testados valores entre $\{0,25; 0,5; 0,8\}$ para ambos os tipos. O treinamento foi feito em lotes de tamanho $\{10; 50; 100\}$. Foram avaliados modelos com as combinações de todas as configurações apresentadas acima, contudo, em função do espaço, discutiremos apenas os modelos que obtiveram os melhores resultados.

5. Resultados

Os resultados apresentados foram obtidos com o *dataset* dividido em 3 partes: treino, validação e teste. A porção de teste foi utilizada exclusivamente para obter o erro final de predição. Os dados devem ser apresentados para as redes recorrentes como sequências temporais, por isso, foram agrupados de acordo com a coleta, resultando em 16.554 conjuntos com dados de aproximadamente 120 segundos. A qualidade das respostas é medida pela raiz do erro médio quadrático (RMSE). Como os valores foram normalizados entre $[-1, 1]$, a ordem de grandeza do erro reflete o percentual de erro, por exemplo, um RMSE de 0,001 indica um erro da ordem de 0,05%.

A Tabela 2 apresenta as configurações dos melhores preditores obtidos para cada variável dependente utilizando GRU e LSTM. As colunas *Neurônios*, *Camadas* e *Timesteps* indicam as configurações da rede recorrente. Em seguida são apresentadas as quantidades de neurônios nas camadas da MLP de saída. Por fim, são relacionados os parâmetros de *dropout* recorrente e tamanho de lote utilizados no treinamento dos preditores. Em todos os casos foi utilizado *dropout* de 25%.

Tabela 2. Configurações de GRU e LSTM para cada preditor

	Neurônios	**Camadas**	***Timesteps***	**MLP L1**	**MLP L2**	***Dropout* Rec.**	**Tam. Lote**
Preditor				**GRU**			
Vazão	16	1	3	32	16	25%	10
Atraso	128	2	1	256	128	80%	100
Jitter	128	1	3	128	64	25%	100
Perda	32	2	1	128	32	25%	100
Preditor				**LSTM**			
Vazão	64	1	5	64	32	80%	50
Atraso	128	2	3	128	64	80%	100
Jitter	32	2	5	128	32	80%	100
Perda	32	2	1	128	32	80%	100

As configurações dos melhores preditores, observadas na Tabela 2, corroboram os resultados obtidos com testes preliminares utilizando regressão multivariada, dada a diferença de complexidade dos modelos necessários para a regressão sobre cada métrica. Os preditores de atraso com configurações mais complexas, com mais camadas recorrentes e mais neurônios, obtiveram os melhores resultados para esta métrica. Por outro lado, para o preditor de vazão utilizando GRU foram necessários 16 neurônios em cada unidade recorrente para se obter o melhor preditor. Também é possível observar que os modelos baseados em GRU requerem menos *timesteps* do que os baseados em LSTM. Isso é vantajoso em uma implementação de controle de uma rede real, pois são necessários menos dados históricos para que se tenha uma predição, além de demandar menos recursos de memória para armazenar estes dados.

5.1. Avaliação do tempo de predição

Para permitir sua aplicação em ambientes reais, o tempo necessário para processar os dados e dar uma predição sobre as métricas de QoS deve ser suficientemente baixo. A Tabela 3 apresenta o tempo médio necessário para que cada preditor retorne uma previsão sobre a métrica de QoS em uma estação e seu intervalo de confiança (IC). Estes testes foram realizados em um computador com processador Intel(R) Core(TM) i5-7400 @ 3.00 GHz com 16 GB de memória RAM. A execução das predições não necessariamente precisa ocorrer no dispositivo Wi-Fi, podendo este apenas coletar os dados de fluxo e repassar para um dispositivo com maior poder computacional executar as predições e avaliar a QoS das estações.

Tabela 3. Tempos de execução dos preditores (em μs)

	GRU		LSTM	
Preditor	**Médio** $\pm$ IC (95%)	**Máximo**	**Médio** $\pm$ IC (95%)	**Máximo**
Vazão	**11,0** $\pm$ 0,13	715	18,2 $\pm$ 0,16	896
Atraso	**20,8** $\pm$ 0,21	1.020	39,9 $\pm$ 0,22	1.160
Jitter	**18,0** $\pm$ 0,13	777	25,5 $\pm$ 0,30	1.800
Perda	**13,8** $\pm$ 0,21	1.290	**13,8** $\pm$ 0,21	1.250

Os preditores baseados em GRU apresentaram tempos de execução inferiores aos baseados em LSTM. Isso pode ser atribuído à maior complexidade dos preditores baseados em LSTM selecionados, que em geral utilizam mais *timesteps* que os preditores baseados em GRU. Os preditores de perda, que possuem configurações de camadas, *timesteps* e neurônios semelhantes para GRU e LSTM retornaram predições com tempos de execução semelhantes.

Considerando uma rede com controle central, como por exemplo uma rede SDN onde o controlador possui uma grande capacidade de processamento e muitas vezes o controlador usa mecanismos de processamento paralelo (e.g. ONIX), o tempo obtido para a predição nos parece razoável. [Tootoonchian et al. 2012] mostram que o tempo de resposta médio de um controlador NOX-MT é 9,92 ms para 32 switches e 2^{12} requisições, contudo, para a mesma situação o controlador Beacon responde em 30,5 ms. Este resultado é semelhante à [Dixit et al. 2013] que indica que o tempo de resposta de um controlador é da ordem de 5 ms para uma taxa de chegada de 1500 packet-in por segundo.

5.2. Avaliação da qualidade das previsões

A Tabela 4 mostra o valor médio do RMSE para cada uma das métricas de QoS. São mostradas três colunas: (1) o valor médio entre os resultados, (2) o valor mediano, e (3) o valor máximo obtido (i.e., o pior erro obtido). Observa-se que não há predominância de um método em relação ao outro. Os valores em negrito na tabela indicam os menores erros obtidos entre os dois métodos para cada uma das métricas. A última coluna da tabela mostra a diferença percentual do valor médio do melhor método em relação ao pior método. A GRU obtém melhores resultados para a vazão e *jitter*, enquanto LSTM obtém melhores resultados para atraso e perda de pacotes.

A Figura 4 mostra os valores previstos e os valores reais obtidos do conjunto de testes para uma amostra de execução de 120 segundos para a vazão. A curva cheia corresponde ao valor real obtido na coleta de dados, a curva pontilhada corresponde ao valor previsto pela GRU (*Y GRU*) e a curva tracejada indica o valor previsto por LSTM (*Y*

Tabela 4. Valores de RMSE obtidos para GRU e LSTM

Preditor	GRU			LSTM			Dif. Percentual
	Médio	Mediano	Máximo	Médio	Mediano	Máximo	Média
Vazão	**1,48e-02**	8,66e-03	7,71e-01	1,73e-02	1,40e-02	7,77e-01	16,9%
Atraso	1,57e-04	1,70e-05	8,70e-03	**1,53e-04**	3,69e-05	8,11e-03	2,5%
Jitter	**1,93e-05**	1,55e-06	2,19e-03	3,00e-05	1,71e-05	1,89e-03	55,4%
Perda	1,47e-05	1,18e-05	4,63e-04	**1,02e-05**	4,89e-06	4,44e-04	44,1%

LSTM). Os dados de (*Y medido*) não foram utilizados na fase de treinamento do modelo. A figura permite visualizar como a RNN é capaz de prever a evolução de cada métrica. Estes são exemplos típicos de previsão, não uma amostra selecionada. É possível observar na Tabela 4 que o preditor, seja ele obtido utilizando GRU ou LSTM, é capaz estimar a variação para estes casos com erros médios na ordem de 10^{-2} para a vazão, 10^{-4} para o atraso, e 10^{-5} para *jitter* e para a perda.

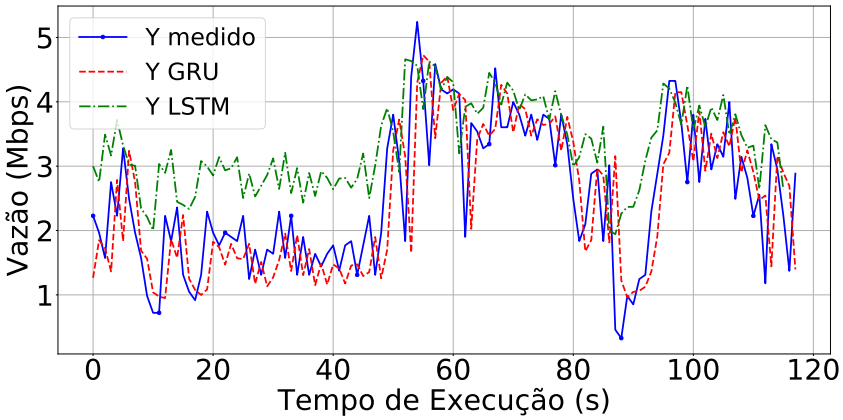

Figura 4. Previsão da Vazão usando dados de teste com GRU e LSTM.

Ao analisar o erro da predição, constata-se que esta proposta poderia ser utilizada em comunicações de missão crítica e em comunicação de baixa latência ultra confiável, onde a latência na rede é um fator importante. Por exemplo, aplicações como sistemas de transporte inteligentes e *smart grid* demandam uma latência fim-a-fim de até 20 ms [Parvez et al. 2018]. Como o erro máximo de predição da latência é da ordem de 10 ms, o método possui precisão suficiente para o controle de tais tipos de aplicação. Contudo, para algumas aplicações de automação de fábrica, com latências da ordem de 250 μs, a previsão pode apresentar erros em dígitos significativos.

5.3. Avaliação da capacidade de generalização

A Tabela 5 mostra o RMSE médio obtido para amostras com mais estações do que as utilizadas para treinamento para GRU e LSTM. Inicialmente foi removida apenas a estação 8 e o treinamento foi realizado com as estações 1 a 7. Em seguida as estações 7 e 8 foram removidas e o treinamento foi feito com as estações restantes. Depois foram removidas as estações 6, 7 e 8 e assim por diante, até que o caso onde o treinamento foi realizado somente com a estação 1. Os maiores valores para cada caso estão destacados em negrito.

Esse teste permite verificar a capacidade de generalização do modelo para estações que não fazem parte do conjunto de treino, para o cenário descrito na seção 4.1, onde as

Tabela 5. RMSE removendo estações do treinamento na GRU e LSTM

ID Estações Removidas	GRU				LSTM			
	Vazão	**Atraso**	***Jitter***	**Perda**	**Vazão**	**Atraso**	***Jitter***	**Perda**
8	**2,49e-02**	3,54e-04	1,03e-04	**2,14e-05**	3,00e-02	**3,51e-04**	**6,38e-05**	2,22e-05
7 e 8	**2,64e-02**	3,74e-04	**3,00e-04**	**7,75e-04**	2,69e-02	**3,49e-04**	3,02e-04	**7,75e-04**
6 a 8	2,55e-02	3,50e-04	2,64e-04	**6,48e-04**	**2,12e-02**	**3,41e-04**	**2,59e-04**	**6,48e-04**
5 a 8	**2,45e-02**	3,38e-04	2,38e-04	**7,52e-04**	2,57e-02	**3,32e-04**	**2,29e-04**	**7,52e-04**
4 a 8	2,26e-02	3,41e-04	2,10e-04	**1,79e-03**	**2,13e-02**	**3,34e-04**	**2,06e-04**	**1,79e-03**
3 a 8	**1,96e-02**	1,01e-03	4,56e-04	1,57e-03	3,36e-02	**6,57e-04**	**2,35e-04**	**1,56e-03**
2 a 8	**2,33e-02**	7,75e-04	**2,05e-04**	**2,19e-03**	3,44e-02	**6,07e-04**	2,09e-04	**2,19e-03**

estações são homogêneas e com perfil de tráfego semelhante. Além disso, verifica se um preditor treinado para uma determinada quantidade de estações também é capaz de fazer predições para uma rede com mais estações. As tabelas apresentam o RMSE sobre o conjunto de testes completo, tanto com dados de estações presentes no treinamento, quanto dados daquelas que foram removidas. Observa-se que, apesar de haver variações (esperadas) no RMSE, a ordem de grandeza dos erros não é afetada significativamente, exceto para a previsão de perda. Comparando os resultados obtidos com os dois métodos para vazão, por exemplo, observa-se que não há predominância entre os métodos.

5.4. Discussão

Os resultados obtidos neste trabalho não corroboram os resultados de [Jozefowicz et al. 2015]. Esses autores mostraram que a GRU proporciona um desempenho competitivo em relação à LSTM na maioria dos experimentos realizados. No presente trabalho, a GRU apresentou melhores resultados para a vazão (um erro cerca de 16,9% menor) e *jitter* (55,4%), enquanto LSTM apresentou melhores resultados nas outras duas métricas. LSTM obteve uma redução do RMSE de 2,5% frente ao valor obtido pela GRU no preditor de atraso e de 44,1% no preditor de perda de pacotes.

Apesar dos resultados utilizando LSTM e GRU apresentarem as mesmas ordens de grandeza para o RMSE, os resultados com GRU foram obtidos com redes menores e com valores de *dropout* menores. A menor quantidade de *timesteps* necessária para os melhores preditores com GRU também reduz a quantidade de dados históricos necessários para se obter uma predição, o que reduz a demanda por espaço em memória. Além disso, como os modelos são mais simples, os preditores que usam GRU executaram as previsões em menor tempo do que aqueles baseados em LSTM.

6. Conclusões

Neste trabalho mostramos a viabilidade de construção de uma função de regressão para os principais componentes de QoS em estações Wi-Fi, utilizando métricas obtidas exclusivamente no AP. Essa abordagem permite estimar a QoS percebida em clientes Wi-Fi, o que pode ser usado para guiar decisões de gerenciamento. Basta que o operador da rede tenha acesso aos dados do AP, pois o método não precisa ter acesso ou controle sobre as estações. Pelo conhecimento dos autores, nenhuma abordagem na literatura atual faz este tipo de inferência utilizando métodos de aprendizado de máquina.

Foram construídos modelos de regressão utilizando GRU e LSTM, empregando um *dataset* composto de 1.991.440 amostras (de 1 segundo de coleta), com 114 características, utilizando dispositivos de hardware em ambiente real. Estes dados são rotulados

com 4 características relacionadas a QoS, (vazão, atraso, perda e *jitter*), medidas nas estações Wi-Fi. O modelo apresentou um erro médio da ordem de 10^{-2} para vazão, de 10^{-4} para o atraso, e de 10^{-5} para *jitter* e para a perda de pacotes para ambos os métodos. Os resultados mostram, ainda, que o modelo é resistente, pois é capaz de generalizar resultados de predição para estações que não estão no conjunto de treinamento.

Como sequência deste trabalho, pretendemos utilizar a função de regressão obtida em um algoritmo de controle automático executando em um controlador SDN, como o Ethanol [Moura et al. 2015], de modo que ele possa definir a configuração dos APs que garanta a melhor QoS para as estações cliente. Também pretendemos realizar uma avaliação em um ambiente mais amplo e mais complexo, por exemplo considerando a mobilidade de estações e dispositivos com hardware heterogêneo.

Agradecimentos

Os autores agradecem às agências de pesquisa CNPq, FAPEMIG, e RNP (via MCTIC, sob acordo no. 688941 - FUTEBOL) pelo apoio financeiro ao presente trabalho. O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Referências

[Bernaille et al. 2006] Bernaille, L., Teixeira, R., Akodkenou, I., Soule, A., and Salamatian, K. (2006). Traffic classification on the fly. *ACM SIGCOMM CCR*, 36(2):23–26.

[Biswas et al. 2015] Biswas, S., Bicket, J., Wong, E., Musaloiu-e, R., Bhartia, A., and Aguayo, D. (2015). Large-scale measurements of wireless network behavior. In *ACM SIGCOMM CCR*, pages 153–165.

[Cho et al. 2014] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Empirical Methods in Natural Language Processing*.

[Crawley et al. 1998] Crawley, E. S., Nair, R., Rajagopalan, B., and Sandick, H. (1998). A Framework for QoS-based Routing in the Internet. RFC 2386, RFC Editor. http://www.rfc-editor.org/rfc/rfc2386.txt.

[DiCioccio et al. 2013] DiCioccio, L., Teixeira, R., and Rosenberg, C. (2013). Measuring Home Networks with Homenet Profiler . In *14th PAM*, pages 176–186.

[Dixit et al. 2013] Dixit, A., Hao, F., Mukherjee, S., Lakshman, T., and Kompella, R. (2013). Towards an elastic distributed SDN controller. In *ACM SIGCOMM CCR*, volume 43. ACM.

[Ehsan and Hamdaoui 2012] Ehsan, S. and Hamdaoui, B. (2012). A survey on energy-efficient routing techniques with QoS assurances for wireless multimedia sensor networks. *IEEE COMST*, 14(2):265–278.

[Elman 1990] Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

[Fazio et al. 2013] Fazio, P., Tropea, M., Marano, S., and Sottile, C. (2013). Pattern prediction in infrastructured wireless networks: Directional vs temporal statistical approach. In *Wireless Days, IEEE/IFIP IM*, pages 1–5. IEEE.

[Gelenbe et al. 2002] Gelenbe, E., Lent, R., Montuori, A., and Xu, Z. (2002). Cognitive packet networks: QoS and performance. In *10th MASCOTS*, pages 3–9. IEEE.

[Gers et al. 1999] Gers, F. A., Schmidhuber, J., and Cummins, F. (1999). Learning to forget: Continual prediction with LSTM. In *9th ICANN*. IET.

[Gomes et al. 2009] Gomes, R. L., Júnior, J. J., Abelém, A. G., and Júnior, W. M. (2009). QoE and QoS support on Wireless Mesh Networks. In *Proceedings of the XV Brazilian Symposium on Multimedia and the Web*, page 2. ACM.

[Gozdecki et al. 2003] Gozdecki, J., Jajszczyk, A., and Stankiewicz, R. (2003). Quality of service terminology in IP networks. *IEEE Communications Magazine*, 41(3):153–159.

[Hochreiter and Schmidhuber 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural computation*, 9(8):1735–1780.

[Jozefowicz et al. 2015] Jozefowicz, R., Zaremba, W., and Sutskever, I. (2015). An Empirical Exploration of Recurrent Network Architectures. In *Proceedings of the 32nd ICML*, pages 2342–2350.

[Moura et al. 2015] Moura, H., Bessa, G. V. C., Vieira, M. A. M., and Macedo, D. F. (2015). Ethanol: Software Defined Networking for 802.11 Wireless Networks . In *14th IEEE/IFIP IM*, pages 388–396.

[Moura et al. 2019] Moura, H. D., Macedo, D. F., and Vieira, M. A. M. (2019). Automatic Quality of Experience Management for WLAN Networks using Multi-Armed Bandit. *IEEE/IFIP IM*, pages 1–8.

[Parvez et al. 2018] Parvez, I., Rahmati, A., Guvenc, I., Sarwat, A. I., and Dai, H. (2018). A survey on low latency towards 5G: RAN, core network and caching solutions. *IEEE COMST*.

[Paudel et al. 2014] Paudel, I., Pokhrel, J., Wehbi, B., Cavalli, A., and Jouaber, B. (2014). Estimation of video QoE from MAC parameters in wireless network: A Random Neural Network approach. In *2014 14th ISCIT*, pages 51–55.

[Pedras et al. 2018] Pedras, V., Sousa, M., Vieira, P., Queluz, M. P., and Rodrigues, A. (2018). A no-reference user centric QoE model for voice and web browsing based on 3G/4G radio measurements. In *2018 IEEE WCNC*, pages 1–6. IEEE.

[Pokhrel et al. 2013] Pokhrel, J., Wehbi, B., Morais, A., Cavalli, A., and Allilaire, E. (2013). Estimation of QoE of video traffic using a fuzzy expert system. In *2013 IEEE 10th CCNC*, pages 224–229.

[Skansi 2018] Skansi, S. (2018). *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*. Springer.

[Tootoonchian et al. 2012] Tootoonchian, A., Gorbunov, S., Ganjali, Y., Casado, M., and Sherwood, R. (2012). On Controller Performance in Software-Defined Networks. *Hot-ICE*, 12:1–6.

[White et al. 2018] White, G., Palade, A., and Clarke, S. (2018). Forecasting QoS Attributes Using LSTM Networks. In *2018 IJCNN*.