Estudo do comportamento de canal em redes 5G: uma análise preditiva

Patrick Luiz de Araújo

Universidade Federal de Uberlândia
Faculdade de Engenharia Elétrica

Uberlândia
2019

Patrick Luiz de Araújo

Estudo do comportamento de canal em redes 5G: uma análise preditiva

Trabalho de Conclusão de Curso apresentado a Faculdade de Engenharia Elétrica da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do diploma de graduação em Engenharia Eletrônica e de Telecomunicações.

Área de concentração: Engenharia Elétrica

Orientador: Éderson Rosa da Silva

Uberlândia
2019

Este trabalho é dedicado aos meus amigos, que sempre estiveram ao meu lado e à minha família, que jamais desistiu de mim.

Agradecimentos

Gostaria de agradecer aos meus pais, Luiz Gonzaga e Maria Isabel, duas figuras importantes na minha formação como cidadão e profissional.

Gostaria de agradecer à minha companheira Lorrayne por todas as conversas, incentivos e apoio moral pelo meu desenvolvimento técnico, profissional e humano.

Gostaria de Agradecer ao professor Éderson Rosa pela tutoria deste projeto. O profissionalismo, a ética e a atenção do professor foram indispensáveis no desenvolvimento dessa pesquisa.

Gostaria de agradecer acima de tudo à Universidade Federal de Uberlândia e à Faculdade de Engenharia Elétrica. Sem o fomento ao ensino público gratuito e universal, cidadãos como eu jamais teriam acesso ao ensino superior e jamais teriam a oportunidade de produzir ciência no Brasil.

"A year from now you may wish you had started today."
(Karen Lamb)

Resumo

Um parâmetro que passa a ser particularmente importante no dimensionamento das redes 5G é o estado do canal de comunicação, principalmente ao considerar as novas frequências usadas, o número crescente de dispositivos conectados e a necessidade de alocar recursos de forma eficiente. Para dimensionar os efeitos do canal, a *Base Station (BS)* recebe do *User Equipment (UE)* uma medição da qualidade do canal, chamada *Channel Quality Information/Indicator (CQI)*, a partir da qual faz o escalonamento dos recursos para melhor atender os usuários. O estudo aqui desenvolvido visa gerar uma base de dados confiável de um fluxo de CQI ao longo de uma transmissão em uma rede móvel *Long Term Evolution Advanced (LTE-A)* com alguns parâmetros da *Release 15* das *Technical Specification (TS)* do *3rd Generation Partnership Project (3GPP)* para o 5G usando o Matlab. Após isso, propõe-se a aplicação de uma rede neural temporal para prever o CQI ao longo de uma transmissão. Por fim, avalia-se a confiabilidade da rede, comparando os valores preditos pelo sistema e os reais.

Palavras-chave: 5G. 3GPP. CQI. Machine Learning. Redes móveis.

Abstract

A paramether that begins to be particularly important on the dimensionament of the 5G networks is the communication channel state, mainly considering the new frequencies used, the growing number of connected devices and the necessity of allocate resources efficiently. To dimension the channel effects, the Base Station (BS) receives from the User Equipment (UE) a measurement of the channel quality, called Channel Quality Information/Indicator (CQI), from wich makes the scheduling of the resources to better attend the users. The study developed here wants to generate a reliable data base of a CQI flow along a transmission into a mobile network Long Term Evolution Advanced (LTE-A) with some parameters of the Release 15 of the Technical Specification (TS) from 3rd Generation Partnership Project (3GPP) of 5G using Matlab. After this, it is purposed the aplication of a temporal neural network to predict the CQI along a transmission. Lastly, evaluates the reability of the network, comparing the predicted values by the system and the real ones.

Keywords: 5G. 3GPP. CQI. Machine Learning. Mobile Network.

Lista de ilustrações

Figura 1 – Consumo de banda ano a ano em dispositivos móveis. 22
Figura 2 – Diagrama do escalonamento de recursos. 22
Figura 3 – Reporte dos CSI em uma rede móvel. 26
Figura 4 – Representação gráfica de um neurônio. 27
Figura 5 – Representação de uma rede neural genérica. 28
Figura 6 – Representação de uma *Recurrent Neural Network* (RNN). 29
Figura 7 – Célula *Long Short-Term Memory* (LSTM) convencional. 30
Figura 8 – Estrutura da rede. 34
Figura 9 – Configuração de canal RMC R.3. 36
Figura 10 – SINR e CQI obtidos ao longo da simulação. 38
Figura 11 – SINR e CQI obtidos ao longo de um intervalo de tempo selecionado. . . 39
Figura 12 – Erro quadrático médio a cada 100 iterações durante o treinamento da rede. 41
Figura 13 – Valores reais e valores preditos para uma rede composta de células GRU. 42
Figura 14 – Valores reais e valores preditos para uma rede composta de células LSTM. 43
Figura 15 – SINR para diferentes *seeds*. 44
Figura 16 – 200 Primeiros registros da SINR para diferentes *seeds*. 45
Figura 17 – Convergência do erro quadrático médio no treinamento da rede para o *seed*=7. 47
Figura 18 – Teste de previsão para o *seed*=7. 48
Figura 19 – Convergência do erro quadrático médio no treinamento da rede para o *seed*=8. 49
Figura 20 – Teste de previsão para o *seed*=8. 50
Figura 21 – Convergência do erro quadrático médio no treinamento da rede para o *seed*=9. 51
Figura 22 – Teste de previsão para o *seed*=9. 51

Lista de siglas

3GPP *3rd Generation Partnership Project*

ACR *Average Difference Ratio*

AWGN *Additive White Gaussian Noise*

BS *Base Station*

CFI *Control Format Indicator*

CQI *Channel Quality Information/Indicator*

CSI *Channel State Information*

CSI-RS *Channel State Information Reference Signal*

GMEDS *Generalized Method of Exact Doppler Spread*

GRU *Gated Recurrent Unit*

HARQ *Hybrid Automatic Repeat Request*

LSTM *Long Short-Term Memory*

LTE-A *Long Term Evolution Advanced*

MCS *Modulation and Coding Scheme*

MIMO *Multiple Input Multiple Output*

ML *Machine Learning*

mmWave *Milimeter-Wave Communications*

OCEAN *Online CSI prEdiction scheme for 5G wireless communicAtioNs*

OFDM *Orthogonal Frequency-Division Multiplexing*

PDCCH *Physical Downlink Control Channel*

QoS *Quality of Service*

RB *Resource Block*

RMC *Reference Measurement Channel*

RNN *Recurrent Neural Network*

SINR *Signal-to-interference-plus-noise ratio*

TBS *Transport Block Size*

TS *Technical Specification*

TTI *Transmission Time Interval*

UE *User Equipment*

Lista de tabelas

Tabela 1 – Parâmetros de simulação usados. 37
Tabela 2 – Critérios usados na rede neural 40
Tabela 3 – Análise estatística dos resultados para diferentes *seeds*. 46
Tabela 4 – Resumo dos resultados obtidos. 49
Tabela 5 – Valor dos *Average Difference Ratio* (ACR) para os *seeds* simulados. . . 50

Sumário

1 INTRODUÇÃO 21
1.1 Motivação 21
1.2 Objetivos e Desafios da Pesquisa 23
1.3 Hipótese 23
1.4 Organização do Trabalho 24

2 FUNDAMENTAÇÃO TEÓRICA 25
2.1 Channel State Information (CSI) 25
2.2 Redes neurais 27
2.3 LSTM 29
2.4 TensorFlow 31

3 PROPOSTA 33

4 EXPERIMENTOS E ANÁLISE DOS RESULTADOS 35
4.1 Configuração para simulação da rede 35
4.1.1 Modelagem do UE 35
4.1.2 Configuração do modelo de propagação do canal 36
4.1.3 Estimador do canal e delay do CQI 37
4.1.4 Resumo dos parâmetros de simulação 37
4.2 Simulação da rede 37
4.3 Construção da rede neural 40
4.4 Avaliação de desempenho da predição 42
4.5 Variando o *seed* da simulação 44
4.5.1 CQI e *Signal-to-interference-plus-noise ratio* (SINR) obtidos 44
4.5.2 Treinamento e avaliação das redes 46
4.6 Avaliação preliminar dos resultados 49

5 CONCLUSÃO 53
5.1 Avaliação final dos resultados 53
5.2 Trabalhos Futuros 54

REFERÊNCIAS 55

ANEXOS 57

ANEXO A – CÓDIGO PYTHON PARA CRIAÇÃO E TESTE DA REDE NEURAL 59

Capítulo 1

Introdução

1.1 Motivação

Conforme visto na *TS*38.104 a nova geração de redes 5G vai operar em frequências que vão de 450 MHz a 52,6 GHz, apresentando, em canais com frequências de ordem superior, uma grande dependência da qualidade do canal e da magnitude dos sinais interferentes (3RD GENERATION PARTNESHIP PROJECT, 2018a, p. 29). Para canais na faixa de 38 GHz mesmo janelas de vidro e muros são obstáculos que podem dificultar ou impossibilitar a comunicação (RAPPAPORT et al., 2017).

Estes desafios são só parte dos problemas que serão enfrentados na implantação de redes móveis de última geração. Conforme registrado no *Ericsson Mobility Report* (ERICSSON, 2019) no quarto trimestre de 2018 haviam 5,9 bilhões de dispositivos conectados, com 43 milhões de novos dispositivos conectados em rede no período. Neste contexto, espera-se um crescimento ainda maior do número dispositivos, que irão demandar uma banda de frequência cada vez maior. Assim, a rede 5G deve ser pensada desde o início como uma infraestrutura robusta e que consiga atender dezenas de bilhões de dispositivos conectados, conforme visto na figura 1.

Para administrar os recursos de rede no 5G, um subprocesso chamado escalonador está em execução na BS. Este agente capta uma série de parâmetros da rede como qualidade do canal, banda requerida (por usuário e por aplicação), justiça, entre outros, e busca alocar a banda disponível da melhor forma possível.

A medida da qualidade do canal é um parâmetro importante que pode ser usado no escalonamento de recursos pela BS. O conceito da BS ter a sensibilidade e a percepção da qualidade do enlace entre ela e todos os UEs é chamada de *channel sensitivity* (sensitividade de canal), onde a ideia básica é saber o quão boa são as condições enfrentadas por cada nó da rede. Esta medição, similar a feita no 4G (HELIX, 2017) é chamada de *Link Monitoring* (monitoramento de enlace) e considera outros parâmetros do canal, chamados de *Channel State Information (CSI).*

O CQI em específico é calculado pelo UE a partir de um sinal de referência chamado

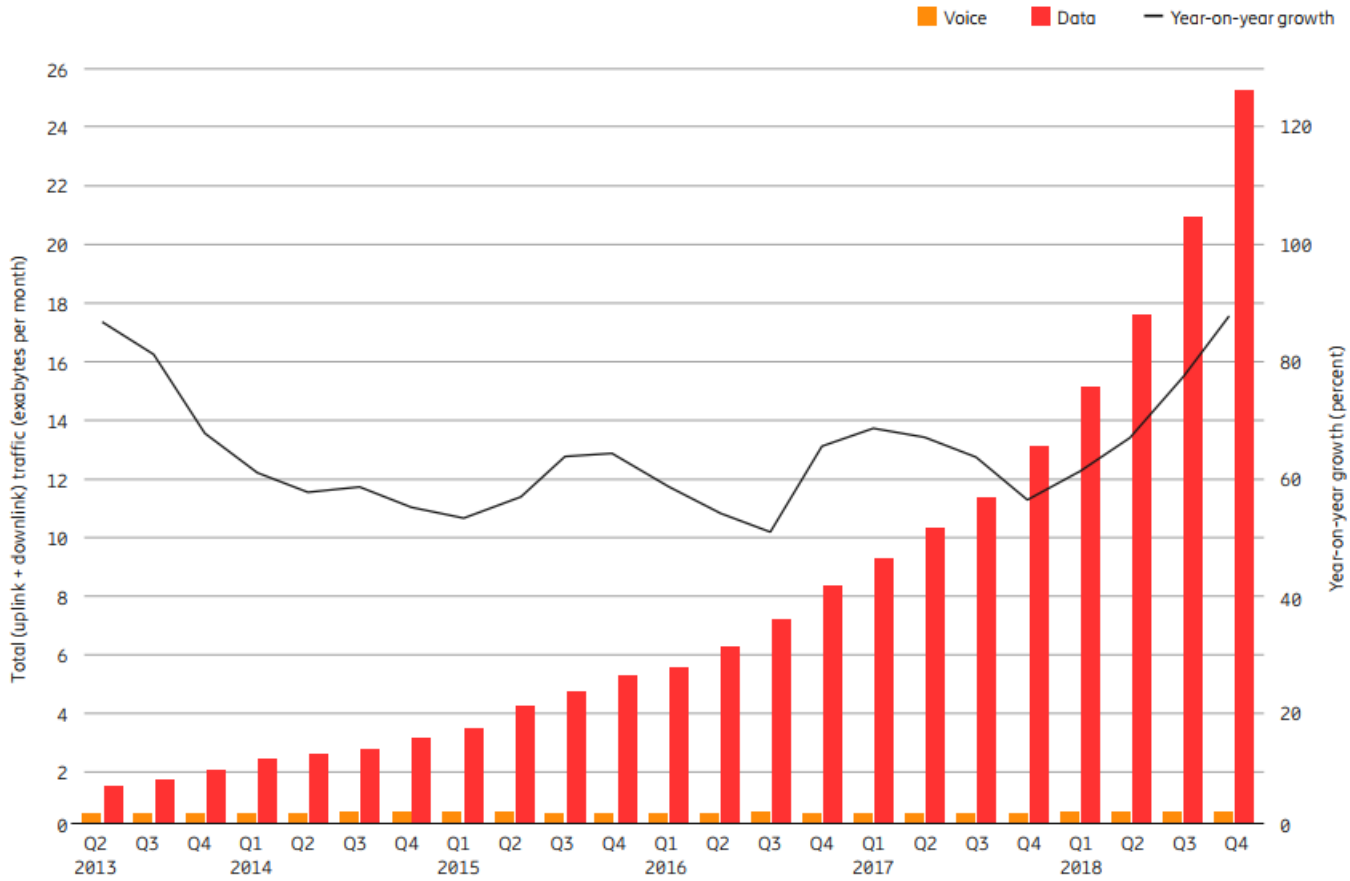

Figura 1 – Consumo de banda ano a ano em dispositivos móveis.

Fonte: Ericsson (2019, p. 4)

Channel State Information Reference Signal (CSI-RS) enviado pela BS.

Após a aquisição dos CQI de todos os usuários conectados, para um escalonamento eficiente, a BS utiliza este e outros parâmetros, para que, no intervalo de tempo posterior, os UEs possam realizar as transmissões com os recursos recebidos.

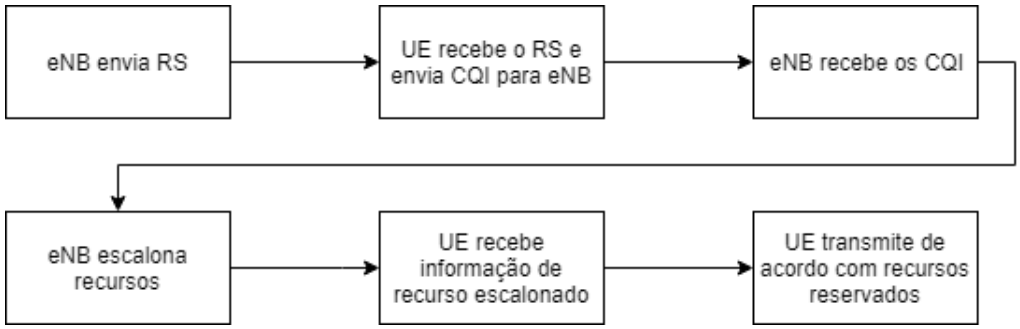

Figura 2 – Diagrama do escalonamento de recursos.

Conforme observado na figura 2, há uma natural diferença de tempo entre a estimativa

dos CQIs pelos UEs e o escalonamento dos recursos pela BS. Neste contexto, dado que grande parte das interferências de canal são de ordem pseudo-aleatória e com um curto intervalo de tempo, os CSI usados no escalonamento podem estar datados, podendo haver perda de desempenho do escalonador.

Para superar esta barreira, a *SINR* (que interfere diretamente nos CSI) deve ser analisada ao longo de uma comunicação a partir do CQI reportado. Com uma base de dados suficientemente grande (com o registro do CQI ao longo do tempo), pode-se avaliar a implementação de um algoritmo preditivo para estimar a qualidade do canal em momentos futuros ao envio do CQI, podendo ser utilizado para um escalonamento mais eficiente.

Além disso, conforme visto em (BI et al., 2015) as redes móveis geram uma grande quantidade de dados, sendo uma fonte de *big data* importante comparável aos ramos de telemedicina, redes sociais e computação distribuída.

1.2 Objetivos e Desafios da Pesquisa

Com o objetivo de avaliar a viabilidade de uma rede preditiva de CQI no escalonador em uma rede 5G, há o desafio da geração de uma base de dados e o desenvolvimento e avaliação de um algoritmo preditivo. Esta base foi criada baseada em simulações de redes 4G com algumas alterações para entrar em conformidade com as mais recentes TS disponíveis para o 5G, como as TS 38.104 (*Base Station (BS) radio transmission and reception*) e a TS 38.213 (*Physical layer procedures for control*). As especificações técnicas do *3GPP* foram criadas para padronizar todos os parâmetros técnicos de redes móveis na Europa, testando e padronizando todos os âmbitos de redes móveis desde o 3G e sendo referência para os órgãos competentes de diversos países ao redor do globo.

A validação da base de dados gerada será feita buscando a conformidade com as especificações técnicas do 5G. Além disso, técnicas empregadas, como por exemplo, *Milimeter-Wave Communications (mmWave)* e *Orthogonal Frequency-Division Multiplexing (OFDM)*, juntamente com o número crescente de canais, trazem desafios adicionais na predição do estado do canal quando comparado a redes anteriores.

Um desafio é encontrar um esquema para predição que não seja computacionalmente muito complexo, e portanto, apropriado para redes 5G. Neste sentido, após a aquisição dos dados, a construção do algoritmo preditivo será feita usando o TensorFlow, uma biblioteca de *Machine Learning (ML)* em Python que permite a criação de redes neurais em um alto nível de programação.

1.3 Hipótese

A hipótese aqui defendida é a respeito da sazonalidade dos eventos de perturbação em um enlace de rede móvel. Alguns efeitos como reflexões, interferência entre antenas

de um sistema *Multiple Input Multiple Output (MIMO)*, interferência co-canal e muitos outros têm impacto na alocação de recursos, uso da banda e *Quality of Service (QoS)* da aplicação em uso no momento.

Para usar esta periodicidade dos eventos como vantagem, redes de ML são grandes aliadas e já são usadas em diversas áreas de análise de dados com cases de sucesso em áreas como previsão de demanda de tráfego em redes móveis (WANG et al., 2017), (SCIANCALEPORE et al., 2017).

Em (KATO; et al., 2017) está documentado o uso de ML para otimizar o processo de roteamento nas redes. Outro caso de sucesso documentado similar ao explorado aqui está descrito em (LUO et al., 2018), onde a qualidade do canal é prevista por uma rede neural e testada de forma prática.

Com estas hipóteses comprovadas, propõe-se o treinamento de uma *Recurrent Neural Network* (RNN) com células *Long Short-Term Memory* (LSTM) para avaliar o CQI temporalmente, na tentativa de desenvolver um modelo preditivo para a variável.

Para realizar a implementação do modelo, foi escolhida a linguagem Python e a biblioteca TensorFlow, visto que esse ambiente é uma combinação bastante usada no meio científico, já bem documentada e testada.

1.4 Organização do Trabalho

Este trabalho está estruturado de acordo com a descrição a seguir: no capítulo 2 é apresentada uma fundamentação teórica com os conceitos básicos de qualidade de canal em redes móveis, redes neurais e uma introdução à biblioteca TensorFlow usada na construção da rede neural.

Nos capítulos 3 e 4 a proposta para implementação prática é apresentada e documentada através da descrição da simulação e avaliação dos resultados.

No capítulo 5 uma conclusão final é apresentada, avaliando a performance do sistema, pontos de melhoria e potencial do modelo desenvolvido para estudos futuros.

Capítulo 2

Fundamentação Teórica

2.1 Channel State Information (CSI)

O parâmetro CSI possui grande impacto na alocação de recursos de rádio, sendo importante sua estimava adequada no intuito de distribuir de forma otimizada os recursos disponíveis para o enlace.

Conforme tratado anteriormente, os CSI são parâmetros que podem representar o "efeito combinado de perda do caminho, espalhamento, difração, desvanecimento, sombreamento, etc" (LUO et al., 2018), determinando o estado da camada física do enlace da rede. Estes parâmetros são usados no escalonador para determinar características do enlace (como os *Modulation and Coding Scheme (MCS)*) e "ranquear"os usuários no escalonador.

O escalonamento dos recursos tem como característica objetivo obter um desempenho adequado e é feito de forma que a rede tenha o melhor desempenho e consiga atender a maior quantidade de usuários possível. De fato, em (KWAN, 2008) é provado que, quanto maior o número de usuários em uma rede, maior a probabilidade de encontrar um usuário com boas condições de canal, demandando uma pequena quantidade de recursos em troca de uma QoS alta. Este efeito é chamado de *multi-user diversity* (diversidade de múltiplos usuários) e é muito importante para entender a otimização e o máximo aproveitamento dos recursos de uma rede.

No caso específico do CQI o procedimento de cálculo pode ser visto na figura 3, sendo feito da seguinte forma:

1. A BS envia um sinal de referência CSI-RS para os UE;
2. Cada UE calcula o CQI baseado no CSI-RS recebido e envia o valor para a BS;
3. O escalonador recebe os CQI de todos os usuários da rede e escalona os recursos de acordo com os parâmetros do momento. O aglomerado de recursos alocados é chamado de *Resource Block (RB)*;

4. Os RB são enviados aos UE a partir do *Physical Downlink Control Channel (PDCCH)*;
5. Cada UE lê as informações do canal compartilhado PDCCH e busca a informação do RB alocado para si.

É importante destacar que o RB é um alocação na frequência e no tempo, delimitando qual ou quais canais poderão ser ocupados pelo UE e por quanto tempo ele poderá transmitir nestes canais.

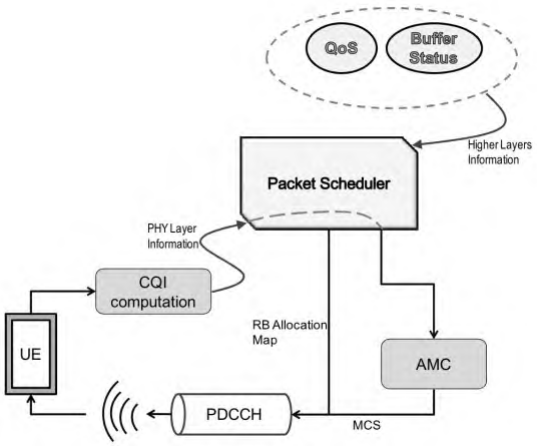

Figura 3 – Reporte dos CSI em uma rede móvel.

Fonte: Kwan (2008, p. 7)

Em um sistema MIMO com t antenas transmissoras e r antenas receptoras considera-se matematicamente um sinal de referência CSI-RS vetorizado do tipo $p_i = p_1, ..., p_n$. No UE o sinal recebido será do tipo:

$$y_i = Hp_i + n_i \tag{1}$$

Em que:
H - Matriz de dimensão $r \times t$ que representa a depreciação do canal (chamada de *complex channel matrix*);
p_i - Sinal de referência padronizado CSI-RS enviado da BS para o UE;
n_i - Vetor complexo $r \times 1$ que representa o ruído *Additive White Gaussian Noise (AWGN)* do canal.

Este processo indica que um sinal p enviado da BS para o UE vai ser depreciado pelo ruído do canal n e pelos efeitos de propagação (reflexão, atenuação, desvanecimento, etc) H. Como as transmissões são feitas em esquemas MIMO, todas estas variáveis são matrizes.

Por vezes o processo de cálculo do CQI é extremamente penoso para a BS, visto que entre os *Transmission Time Interval (TTI)* este procedimento completo deve ser

executado em todos os nós da rede. Nas redes LTE este processo é feito em intervalos de milissegundos (KWAN, 2008).

Alguns algoritmos de previsão de qualidade de canal já foram propostos, podendo ser citados (DU; et al., 2011) e (KARAMI, 2007). Em (MA; et al., 2018) há uma implementação de um algoritmo que usa o erro quadrático médio para fazer predições similares, porém todos os exemplos são limitados por apresentarem cálculos computacionais pesados e pouco eficientes para a aplicação aqui descrita.

Em (LUO et al., 2018) está documentado uma implementação prática de uma rede neural para a predição do indicador de qualidade do canal. Aqui é usada uma rede que leva em conta o gradiente das variáveis utilizadas e que apresentou um bom desempenho geral.

Em todos os exemplos citados o uso do ML foi peça fundamental do sistema, dada a sazonalidade dos efeitos no canal dos enlaces. O próximo tópico irá trazer uma abordagem geral e aprofundar nas redes neurais para análise de dados temporal.

2.2 Redes neurais

Uma rede neural é um modelo computacional baseado em elementos não-lineares (chamados de neurônios ou também *perceptrons*). Fundamentalmente, um neurônio é uma estrutura com vários valores de entrada x_n, um vetor de pesos w_n e uma função de ativação, além de um *bias* que é artificialmente inserido para que a resultante balanceada $d(x)$ seja diferente de 0 quando as entradas forem nulas, conforme mostrado na figura 4.

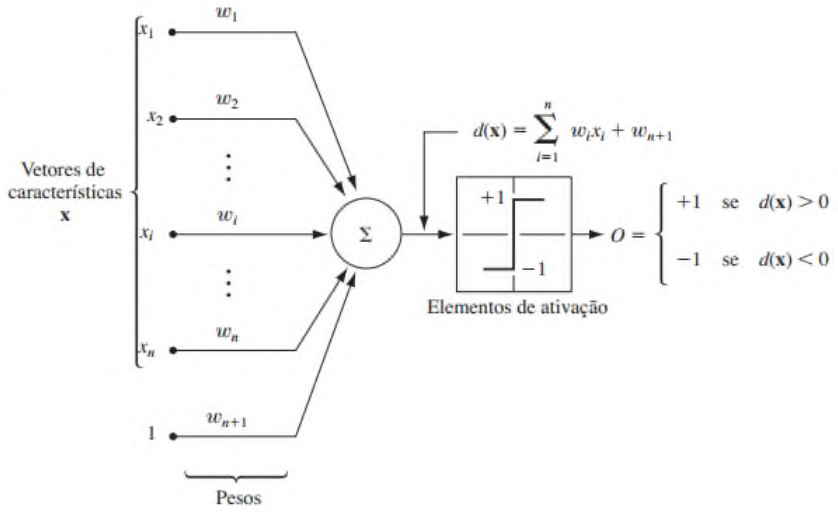

Figura 4 – Representação gráfica de um neurônio.

Fonte: Gonzalez e Woods (2015, p. 582)

Estes neurônios são agrupados em redes de neurônios, chamadas de redes neurais, que podem ser configuradas para balancear uma série de variáveis de entrada x para relacionar as mesmas a uma saída y. Estas redes podem se prolongar por diversas camadas e podem ser construídas para os mais diversos fins, conforme a figura 5 mostra.

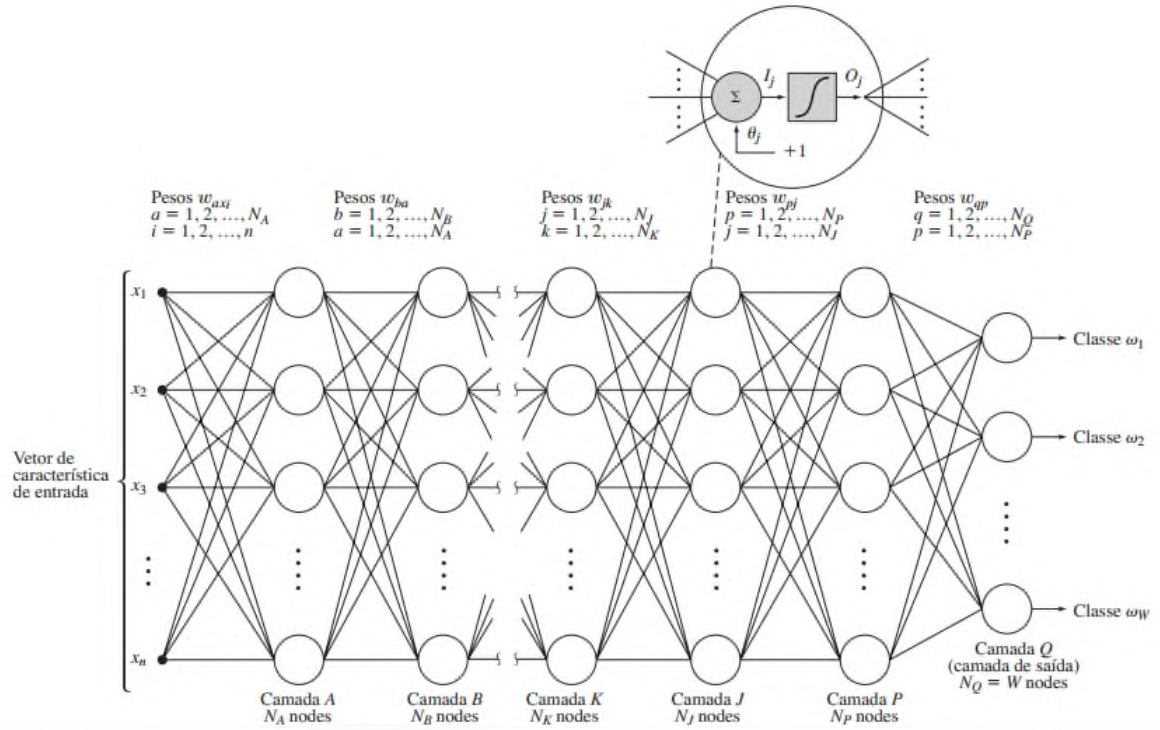

Figura 5 – Representação de uma rede neural genérica.

Fonte: Gonzalez e Woods (2015, p. 585)

Dentre as várias configurações e modelos de redes neurais, aquela a ser explorada neste estudo é chamada de RNN, uma família de modelos especializados em análise de dados sequenciais (GOODFELLOW, 2016, p. 367), conforme observado na figura 6.

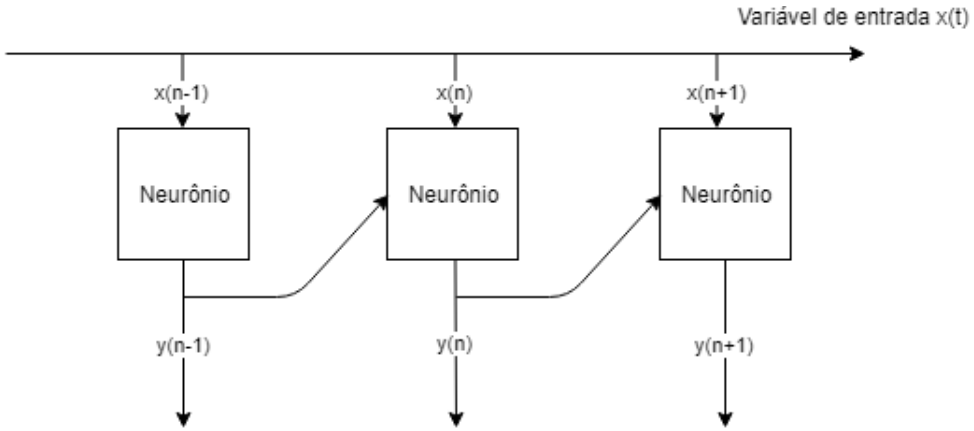

Figura 6 – Representação de uma RNN.

2.3 LSTM

Um dos problemas enfrentados pelas redes RNN é que a mesma é escalada de acordo com a quantidade de entradas anteriores que deseja-se guardar. Conforme apresentado na figura 6 este modelo simplificado costuma funcionar bem, porém para a análise com 6 ou mais métricas passadas, o problema fica complexo e computacionalmente pesado.

Adicionalmente, há os problemas de desvanecimento (pesos vão diminuindo até se tornarem insignificantes), gradientes ascendentes (onde os pesos vão aumentando até a saída da rede ficar destonante da realidade) e o ruído que cresce nas camadas mais profundas da rede.

Para resolver este problema um novo modelo foi proposto (HOCHREITER, 1997). A rede montada é chamada de LSTM, que usa uma série de elementos de memória para considerar o gradiente de uma variável ao longo do tempo, não somente o seu valor atual.

A célula é composta por 4 parâmetros: estado celular c_{t-1} e c_t, portão de esquecimento f, *input gate* i_t e *output gate* o_t, conforme visto na figura 7.

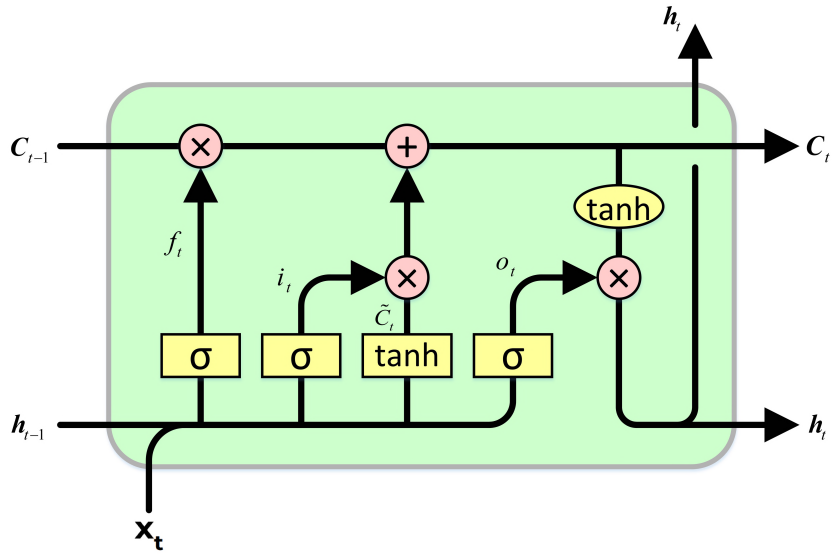

Figura 7 – Célula LSTM convencional.

O portão do estado celular c representa uma ponderação em função das entradas em tempos anteriores da rede.

$$c_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2}$$

O portão do esquecimento f decide qual informação do estado anterior vai ser descartada. A variável pode ter 2 resultados, sendo 1 representando a retenção integral do parâmetro e 0 o descarte.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3}$$

O portão de entrada *input gate* i_t decidirá qual informação da entrada c_{t-1} será inserida no candidato a saída c_t. Um $i_t = 0$ simboliza um valor de entrada desconsiderado e $i_t = 1$ simboliza um valor de entrada considerado integralmente.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{4}$$

$$\widetilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_i \tag{5}$$

O *output gate* determina qual informação nova será enviada para a saída da rede.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{6}$$

$$h_t = o_t * tanh(C_t) \tag{7}$$

Para maiores detalhes do conceito da célula LSTM, recomenda-se a leitura de (HOCHREITER, 1997) e (COLAH'S BLOG, 2015).

A célula LSTM possui aplicações que vão desde reconhecimento de imagem e padrões até tradução de textos (WU; ET, 2016), sendo aplicáveis ao desafio proposto também, devido a sua capacidade de lidar com séries temporais de dados.

2.4 TensorFlow

Com o avanço na área de redes neurais e ML houve também um aprimoramento de *frameworks* e APIs que implementam redes cada vez mais complexas de aprendizado em um nível superior de programação, exigindo menos código e menos conhecimento para a implementação de algoritmos preditivos, por exemplo.

A ferramenta explorada neste trabalho foi o TensorFlow, uma API *open-source* em Python da Google. (TENSORFLOW, 2019)

A mesma já apresenta as principais ferramentas de redes neurais (LSTM, manipulação de dados, etc) implementadas, sendo chamadas por meio de funções prontas, além de criar códigos portáveis entre diferentes sistemas.

Capítulo 3

Proposta

Conforme feito em (WANG et al., 2017) e (SCIANCALEPORE et al., 2017) uma aproximação do comportamento das futuras redes 5G podem ser feitas a partir de simulações e análise de redes 4G. Mais ainda, conforme visto em (WANG et al., 2017) foi feito um experimento prático usando equipamentos de transmissão WiFi para estimar o comportamento de redes 5G.

De fato, a similaridade dos protocolos de medição de qualidade do canal em sistemas de comunicação móveis podem ser usados para que possa se estimar o comportamento de futuras redes 5G, usando as já bem estabelecidas redes de quarta geração e até mesmo WiFi.

Com isso, e visto a pequena disponibilidade de implementações práticas e simulações de redes de quinta geração, será proposto, num primeiro momento, a análise de dados em uma simulação para redes de quarta geração com adaptações e parâmetros vistos na *Release 15* do 3GPP, de forma a gerar uma base de dados temporal com o CQI do canal em intervalos periódicos e com uma duração grande o suficiente a ponto de poder ser usada como insumo para uma rede neural.

Após a geração da base, uma rede neural temporal será montada para tentar prever a qualidade do canal em momentos futuros aos da análise. Para fins de comparação e para enriquecer a análise, serão testadas duas redes: uma com célula LSTM e outra com célula *Gated Recurrent Unit (GRU).*

Fundamentalmente, ambas as células são usadas em redes RNN de análise temporal, porém a célula GRU é mais simples computacionalmente, porém não costuma apresentar resultados muito assertivos. Em contrapartida, a célula LSTM é mais complexa, gerando redes neurais com um maior esforço computacional, porém apresenta melhores resultados. Na prática, é indicado testar ambas as células, sendo este um dos objetivos deste trabalho.

Podemos ver o esquema a ser montado (para a célula LSTM) na figura 8.

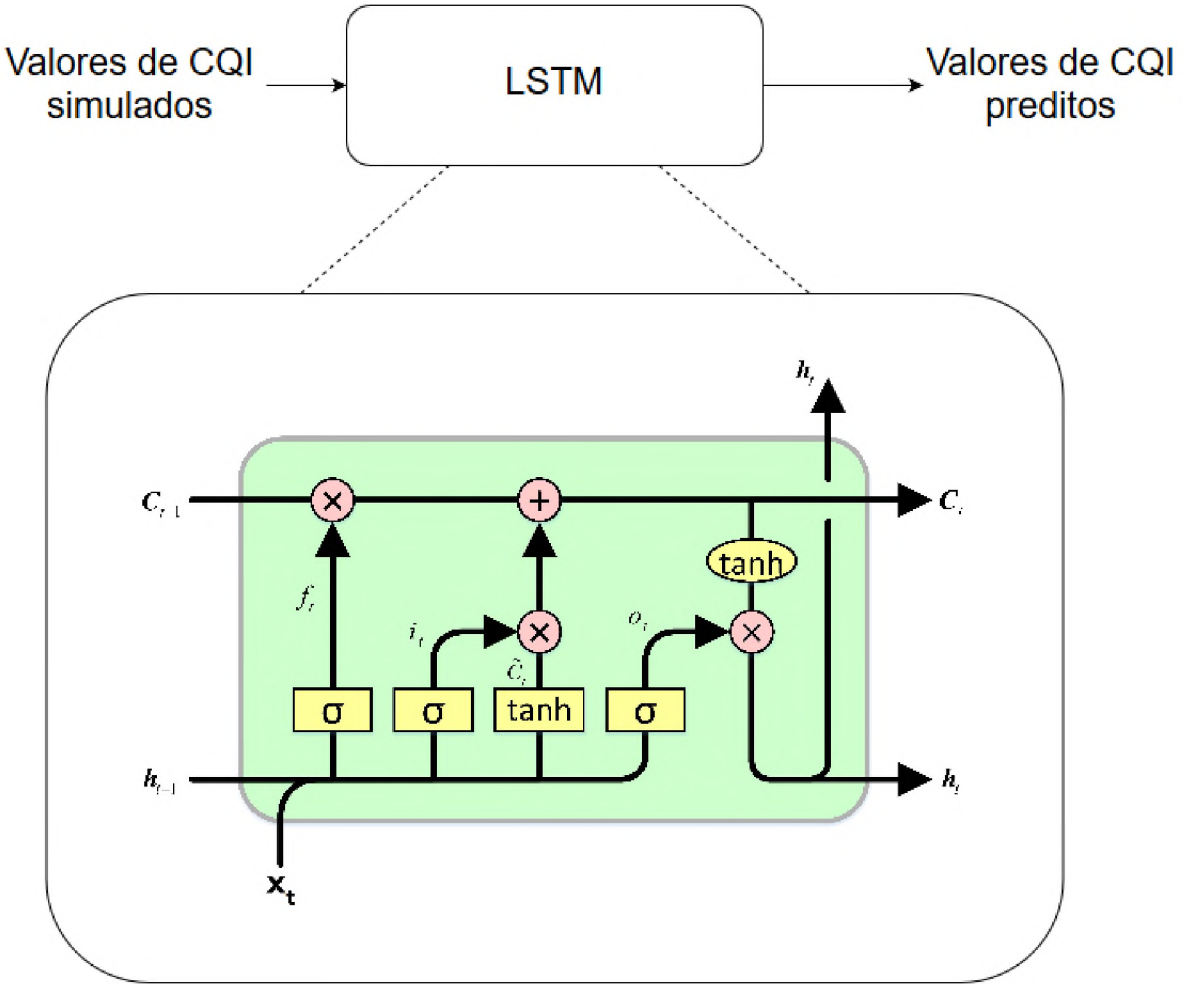

Figura 8 – Estrutura da rede.

Capítulo 4

Experimentos e Análise dos Resultados

Neste capítulo apresentam-se os experimentos realizados e os resultados obtidos com o emprego da proposta de predição, seguindo a seguinte estrutura:

Na seção 4.1 os parâmetros usados para gerar a base de dados são apresentados. Já na seção 4.2 os resultados das simulações são apresentados e analisados.

Na seção 4.3 o ambiente de programação da rede neural e os principais parâmetros são apresentados e explicados. Além disso, variações da rede com células LSTM e GRU são apresentadas e, posteriormente, o seu resultado é avaliado.

Por fim, na seção 4.4 as redes desenvolvidas são testadas e o seu desempenho é avaliado.

4.1 Configuração para simulação da rede

Os primeiros parâmetros importantes são a configuração do número de *frames* a serem simulados e a SNR em dB. O número de *frames* usado foi de 2500 (para gerar uma base de dados sólida) e $SNR = 6dB$.

4.1.1 Modelagem do UE

Na configuração do UE a primeira estrutura configurada é o *Reference Measurement Channel (RMC)*. O RMC é um padrão que representa um exemplo de configuração do canal físico para diferentes taxas de dados e é usado para fins computacionais.

No exemplo, usou-se o RMC R.3, uma das várias configurações de exemplo para o canal de configuração, visto na figura 9.

A seguir o *Control Format Indicator (CFI)* é configurado. O CFI indica quantos símbolos OFDM serão usados para realizar o controle de canal em cada *subframe* e pode ser igual a 1, 2 ou 3 dependendo da banda do sistema (3RD GENERATION PARTNESHIP PROJECT, 2018b, p. 125).

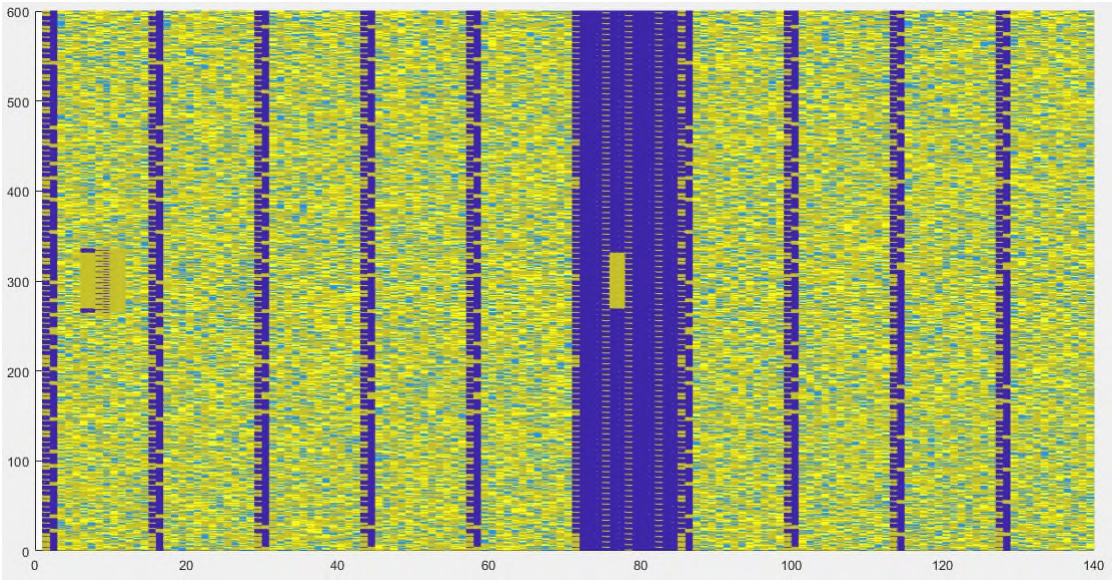

Figura 9 – Configuração de canal RMC R.3.

- CFI = 3 para sistemas com bandas de 1.4 MHz, 3 MHz e 5MHz;
- CFI = 2 para bandas de 10 MHz, 15 MHz e 20 MHz.

Como deseja-se aproximar a simulação para um sistema 5G com altas taxas de transmissão e recepção, que possuem uma elevada banda, foi usado um CFI = 2. Além disso, na linha 54 o código foi configurado para cada *frame* conter um único *subframe*.

O *Hybrid Automatic Repeat Request (HARQ)* foi desabilitado, visto que este sistema é uma mistura de códigos de correção de erro (como o FEC) e o *Automatic Repeat Request* (ARQ). Com a combinação destas 2 técnicas o HARQ possibilita ao receptor corrigir um dado número de erros e, excedido esse valor, requisitar uma retransmissão ao transmissor.

Como o objetivo é avaliar os efeitos do canal de forma pura, optou-se por desabilitar algoritmos corretivos no receptor.

4.1.2 Configuração do modelo de propagação do canal

O primeiro parâmetro configurado é o *seed* da simulação. Este parâmetro nada mais é do que uma forma de padronizar os componentes aleatórios da simulação, de forma que, se feita sucessivas vezes, a simulação apresentará os mesmos resultados. O *seed* escolhido foi de 10.

Além disso o número de antenas receptoras foi configurado como 4. Este número foi escolhido por já ser observado em implementações *LTE-A* em sistemas MIMO 4x4 (QUALCOMM, 2016).

O modelo de *delay* escolhido foi o modelo de delay EPA por já ser implementado na simulação original. Em (MATLAB, 2019) maiores detalhes podem ser vistos.

A seguir a Frequência de Doppler é configurada. Quanto maior for essa variável, maior é a oscilação do CQI reportado, indicando que este parâmetro tem grande influência no CQI. Para aumentar a variabilidade do canal, a variável foi usada como 5 Hz.

A correlação das antenas foi setado como "*high*". Os modos de correlação podem ser: *Low correlation*, *Medium Correlation*, *Medium Correlation A* e *High Correlation* (3RD GENERATION PARTNESHIP PROJECT, 2018c, p. 1623).

O modelo de desvanecimento *Generalized Method of Exact Doppler Spread (GMEDS)* é usado para simular uma série de formas de onda de desvanecimento de Rayleigh não-relacionadas e é uma simplificação computacional usada para tornar a simulação mais leve. Mais detalhes sobre o modelo podem ser vistos em (PäTZOLD, 2009).

4.1.3 Estimador do canal e delay do CQI

A métrica de modelo de canal configuração usada "*perfectChanEstimator = false*"configura uma estimativa de canal imperfeita, sendo mais realista. Além disso, o *delay* do CQI é setado em 8 *subframes*, um valor dado em (3RD GENERATION PARTNESHIP PROJECT, 2018c, 9.3.2.1.1-1).

4.1.4 Resumo dos parâmetros de simulação

Os principais parâmetros configurados podem ser vistos na tabela 1:

Parâmetro	**Valor atribuído**
SNR	6 dB
RMC	R.3
CFI	2
Número de antenas receptoras	4
Delay	Modelo EPA
Frequência de Doppler	5 Hz
Correlação de antenas	*High*
Modelo de desvanecimento	GMEDS
Estimador de canal	Modelo imperfeito
Delay do CQI	*8 subframes*

Tabela 1 – Parâmetros de simulação usados.

Todos os valores foram configurados de forma a aproximar a simulação da rede LTE-A ao máximo de uma transmissão de alta taxa (tal qual o 5G).

4.2 Simulação da rede

Após toda a configuração inicial, a simulação é iniciada. Em um primeiro momento o MCS da transmissão é escolhido baseado no CQI reportado.

1. O CQI é reportado a cada 2 *subframes*;

2. O CQI reportado está com um atraso de 8ms (ou 8 *subframes*, já que o *frame* é de 1ms).

Num segundo momento é calculado o valor médio do CQI e o processamento do mesmo é feito da seguinte forma:

1. O CQI lido é enviado ao *buffer* (quando recebido) ou uma média é calculada (no *subframe* em que não é recebido);

2. MCS é calculado por "*lteMCS*"de acordo com o CQI usado para o cálculo;

3. O MCS é reportado para "*lteTBS*"e o *Transport Block Size (TBS)* e os RBs são calculados;

4. Com os TBS a forma de onda do *Transport Block* (TB) é calculada;

5. A forma de onda do TB recebe um ruído AWGN;

6. O sinal é enviado e o processo se reinicia.

A simulação apresenta como resultado a SINR e o CQI ao longo dos *frames* configurados (no caso, 2500). Os resultados foram salvos em arquivos CSV para futura análise e uso na rede neural construída. Parte dos resultados podem ser vistos na figura 10.

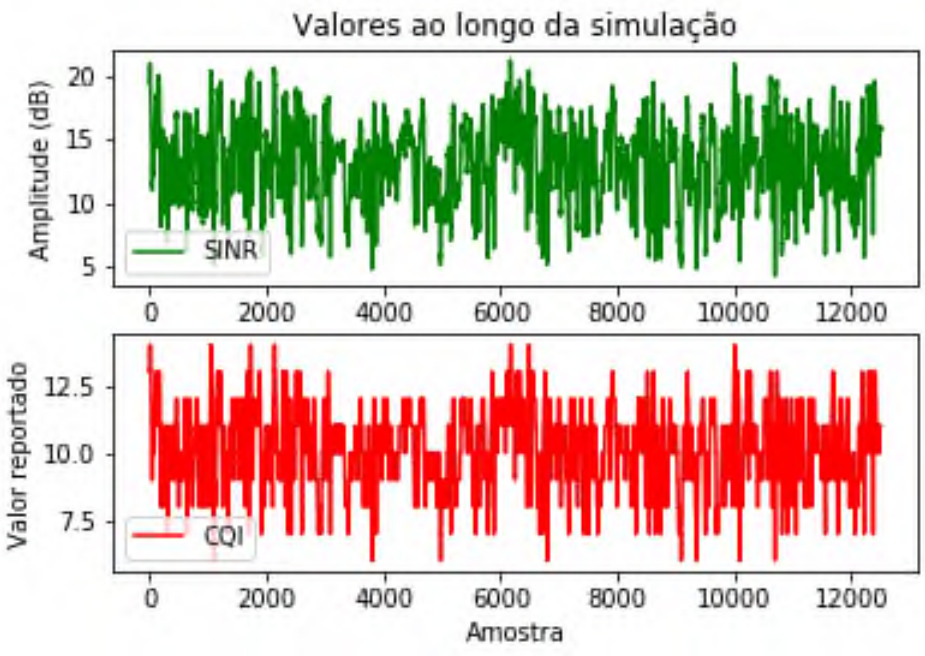

Figura 10 – SINR e CQI obtidos ao longo da simulação.

Como pode ser visto, o CQI acompanha o comportamento da SINR do canal, já que é um reflexo direto desta ao longo do tempo. Enquanto a SINR varia entre 5dB e 20dB, o CQI varia entre aproximadamente 6 e 15. Para uma melhor visualização dos resultados, foca-se em uma pequena parcela dos resultados na figura 11.

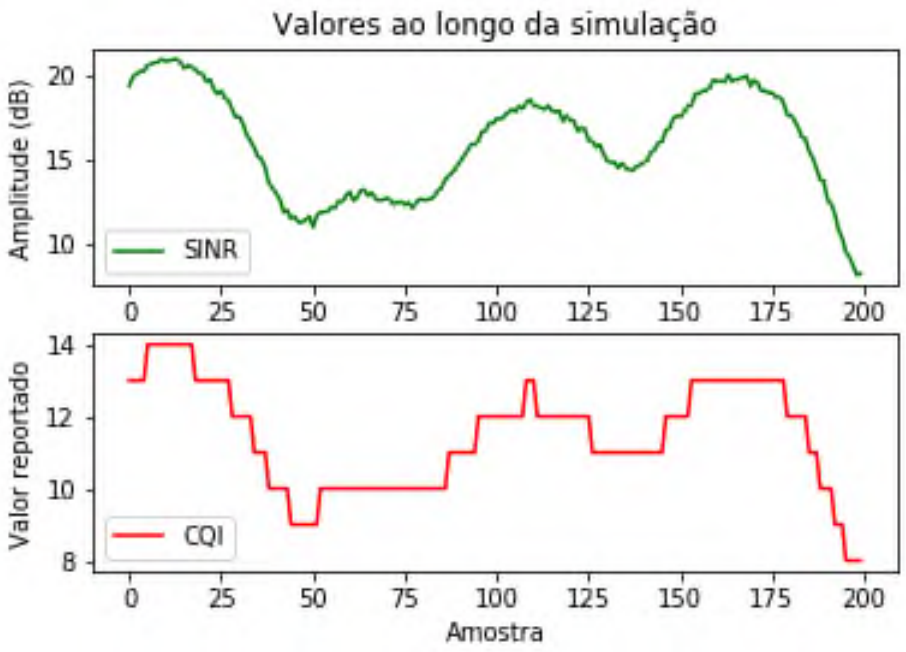

Figura 11 – SINR e CQI obtidos ao longo de um intervalo de tempo selecionado.

4.3 Construção da rede neural

Após a obtenção de uma base de dados confiável, é possível iniciar o treinamento da rede para a construção e teste da rede neural usada. Todo o código pode ser consultado no anexo A.

Todas as execuções foram feitas no ambiente Google Colaboratory, uma ferramenta disponibilizada pela Google para a execução de redes neurais usando o Python, com ênfase na biblioteca TensorFlow. As vantagens do ambiente são a grande velocidade das máquinas alocadas e a gratuidade do serviço.

Para uma melhor avaliação dos resultados da rede neural, realiza-se uma comparação entre as células LSTM (já mencionada) e a célula GRU, sendo similar a aquela, porém computacionalmente mais simples. Para implementações práticas, é válido o teste de ambas as células para ver aquela que se adapta melhor ao problema.

Os seguintes passos são tomados na construção da rede:

1. As bibliotecas que serão usadas são importadas;
2. Os dados em CSV são lidos e adaptados para serem usados na execução;
3. Os dados de CQI são divididos em uma base de treinamento da rede e outra de teste;
4. Os parâmetros da rede são configurados;
5. A base de treinamento é passada pela rede e a mesma é treinada;
6. Após treinada, a base de teste é processada pela rede e os valores preditos são gerados.

Os principais parâmetros da rede podem ser vistos na tabela 2:

Parâmetro	**Valor atribuído**
Modelo de rede	Rede RNN com um único parâmetro temporal
Neuron	LTSM ou GRU
Número de neurons	100
Ritmo de aprendizado	0,001
Número de iterações totais	8000
Tamanho do batch	100 valores
Condição de parada	A partir de 4000 interações e erro quadrático médio abaixo de 0,0006

Tabela 2 – Critérios usados na rede neural

.

A RNN foi usada por ser a rede indicada para a predição de uma variável temporal. Os neurons LSTM e GRU foram testados para a obtenção dos melhores resultados, sendo usados em uma rede com 100 neurons.

O ritmo de aprendizado ideal foi de 0,001, baseado em uma série de testes variando o ritmo de aprendizado e o número de iterações. Esta segunda variável, por sua vez, foi setada em 8000 para uma boa predição e um *batch* (número de amostras processadas por vez) de 100 foi usado. Já a condição de parada da rede foi pensada para evitar o *overfitting* e o treinamento insuficiente da rede.

O *overfitting* é um evento no qual uma rede é treinada em cima de uma única base de dados com um grande número de iterações, ficando "viciada", enviesada nestes dados ao qual está sendo treinada. Do outro lado, treinar a rede com um pequeno número de iterações significa que a mesma ainda não está adequada para o desafio para o qual está sendo preparada.

Em uma das rotinas de processamento o erro quadrático médio (MA; et al., 2018) foi salvo a cada 100 interações em todas as 8000 (a condição de parada em função do erro quadrático médio foi retirada para fins didáticos) conforme mostrado na figura 12.

Figura 12 – Erro quadrático médio a cada 100 iterações durante o treinamento da rede.

Pode-se observar que próximo de 2000 iterações o erro quadrático médio convergiu para um valor muito pequeno, indicado que se o treinamento fosse interrompido naquele ponto poderiam haver bons resultados no seu uso. Porém, devido ao baixo número de iterações, é difícil afirmar que a rede não poderia convergir para resultados melhores com mais iterações.

Ao mesmo tempo, em torno de 7000 iterações passam a não haver muitos pontos de mínimo abaixo de 0,001. Por fim podemos ver que em 6000 iterações há um ponto de mínimo interessante.

Baseado neste comportamento, a condição de parada dinâmica foi feita: a partir de 4000 iterações, para um erro quadrático médio menor que 0,0006 o treinamento é interrompido. Após repetidos testes de treinamento, a rede foi treinada com sucesso.

Após treinada, a rede foi usada para prever o CQI com um conjunto de 100 valores reportados, onde têm-se os valores reais e os preditos lado a lado para as células GRU e LSTM conforme os testes descritos na seção 4.4.

4.4 Avaliação de desempenho da predição

Após uma série de treinamentos com as condições de parada propostas, os melhores resultados para uma célula GRU apresentaram um índice de acerto de aproximadamente 40%, não sendo um bom resultado para viabilizar uma aplicação prática. A plotagem dos valores reais e dos valores preditos podem ser vistos na figura 13.

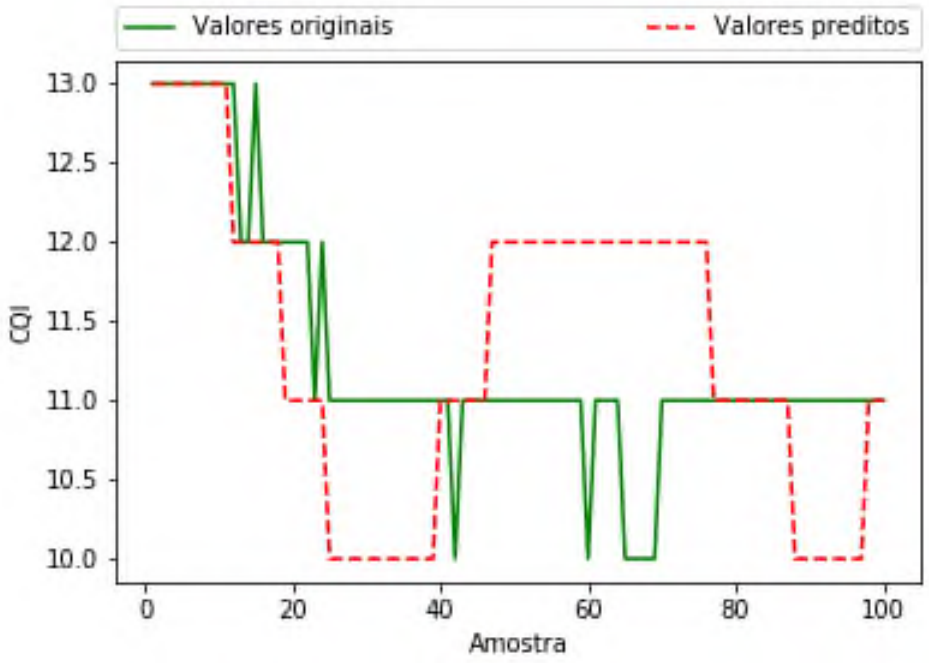

Figura 13 – Valores reais e valores preditos para uma rede composta de células GRU.

Para a composição da rede por células LSTM, houveram resultados melhores girando em torno de 76% de acerto, conforme mostrado na figura 14.

O processo de treinamento da rede pode ser feito diversas vezes, variando de forma sutil os parâmetros de velocidade de aprendizado e número de iterações para obter me-

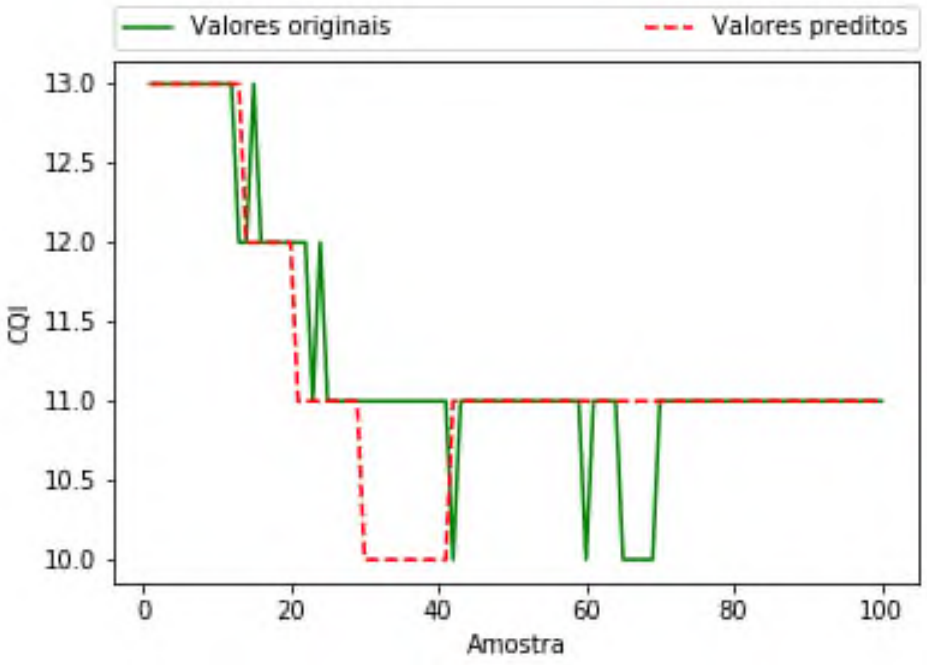

Figura 14 – Valores reais e valores preditos para uma rede composta de células LSTM.

lhores resultados. Em último caso, recomenda-se, como foi feito, uma condição de parada dinâmica nas iterações.

Um índice de acerto em torno de 76% já é um valor bastante satisfatório para provar o conceito de previsão da qualidade do canal, sendo, para o desafio, as células LSTM as mais indicadas. A partir deste ponto todos os experimentos foram feitos com as células LSTM.

4.5 Variando o *seed* da simulação

Para uma maior precisão nos resultados obtidos, o *seed* da simulação que gerou a base de dados no Matlab foi variado para a obtenção de valores de SINR e CQI mais variados. O *seed* é um parâmetro de simulação que fixa a aleatoriedade da simulação onde, com um mesmo *seed*, a mesma sequência de valores aleatórios sempre será apresentado, independente da simulação.

Até agora os testes foram feitos com um *seed* de 10 e, para ampliar a análise do sistema proposto, foram feitas simulações variando este valor com 7, 8 e 9. Novas simulações foram feitas para gerar novas sequências de SINR e CQI e, a partir das sequências geradas, o algoritmo de aprendizado desenvolvido foi treinado novamente e seu desempenho avaliado.

4.5.1 CQI e SINR obtidos

Após realizar as simulações da geração da base de dados no Matlab variando os *seeds*, os resultados para o SINR podem ser vistos na figura 15.

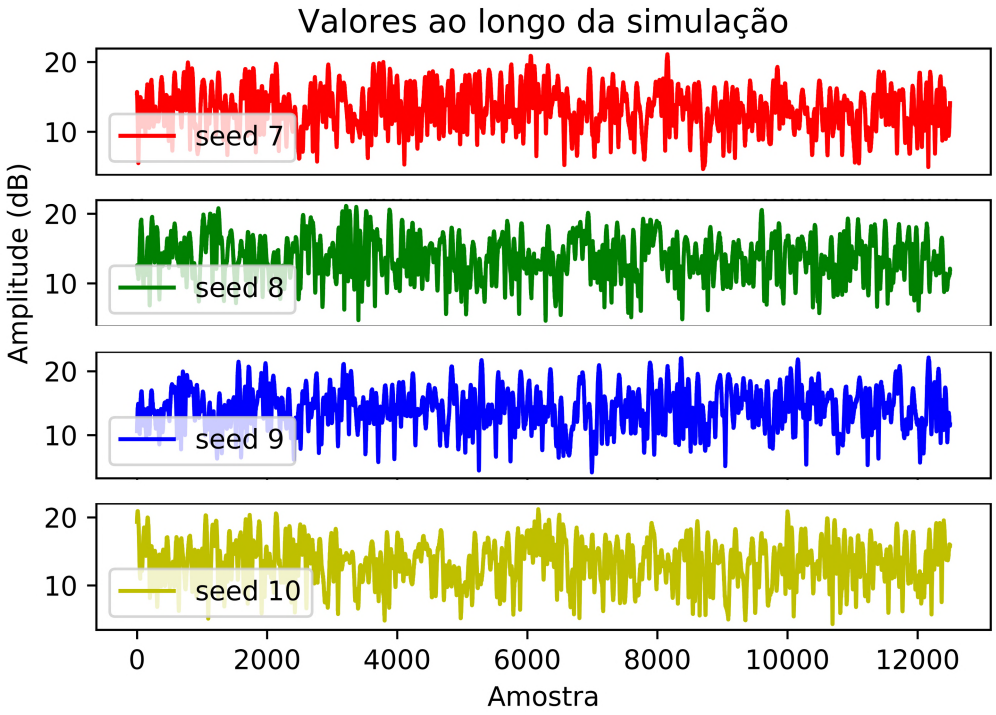

Figura 15 – SINR para diferentes *seeds*.

Para ter uma melhor visualização dos resultados, os 200 primeiros resultados podem ser vistos na figura 16.

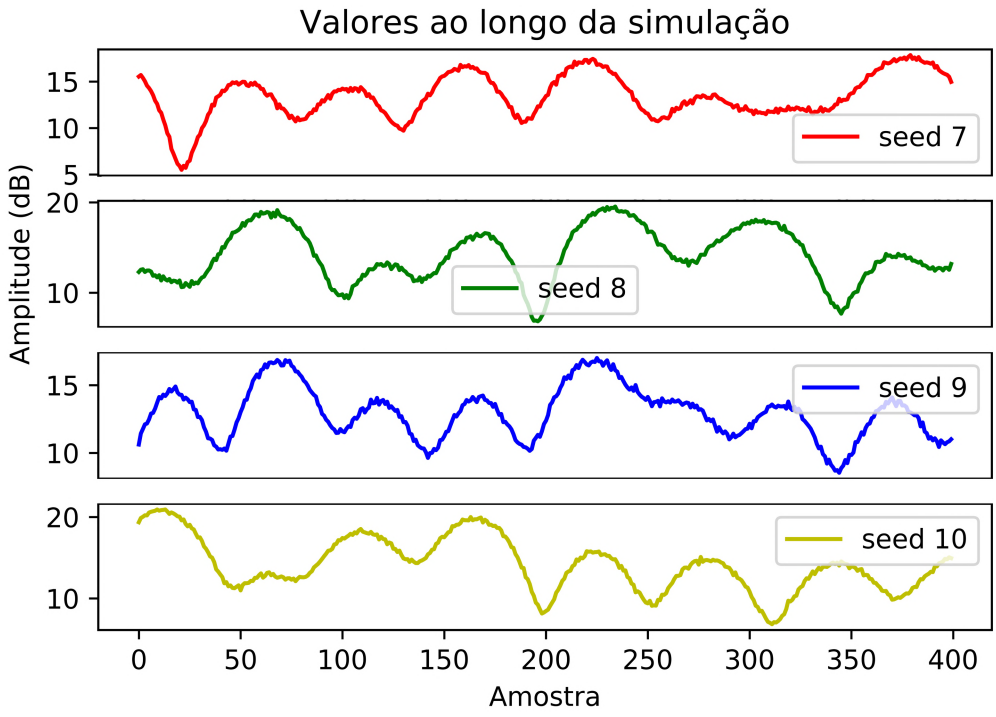

Figura 16 – 200 Primeiros registros da SINR para diferentes *seeds*.

De uma forma geral, o comportamento dos resultados variando os *seeds* de simulação apresentam pequenas variações nos valores. As curvas ainda apresentam uma tendência senoidal, com certa periodicidade nas variações.

Para uma melhor avaliação do montante total dos resultados, uma análise estatística pode ser vista na tabela 3.

Seed	Métrica	Média	Desvio	Mínimo	25%	50%	75%	Máximo
7	CQI	10,40144	1,427154	6,0	9,0	10,0	11,0	14,0
	SINR	13,353637	2,928241	4,6396	11,35175	13,381	15,52725	21,1
8	CQI	10,40144	1,427154	6,0	9,0	10,0	11,0	14,0
	SINR	13,75108	2,942685	4,6377	11,744	13,78	15,94	21,114
9	CQI	10,61792	1,496808	6,0	10,0	11,0	12,0	14,0
	SINR	14,222477	3,082902	4,1707	12,13075	14,2405	16,451	22,13
10	CQI	10,26784	1,483495	6,0	9,0	10,0	11,0	14,0
	SINR	13,474901	3,079816	4,3084	11,309	13,719	15,725	21,214

Tabela 3 – Análise estatística dos resultados para diferentes *seeds*.

Os parâmetros avaliados acima foram a média, o desvio, valores mínimo e máximo, primeiro quartil, segundo quartil e terceiro quartil.

Conforme visto, os *seeds* 7 e 8 apresentam parâmetros similares, enquanto os demais apresentam alguma variação entre si. Outro fato a se destacar é que os primeiro, segundo e terceiro quartis foram os mesmos para o CQI com os *seeds* 7, 8 e 10.

Os valores de média ficaram em torno de 10 para o CQI em todos os *seeds*, enquanto a SINR teve um valor médio em torno de 13, exceto para o *seed*=9.

4.5.2 Treinamento e avaliação das redes

O modelo de avaliação para múltiplos *seeds* adotado foi o de treinamento de uma rede para cada *seed*. Com isso, cada variação das simulações foi avaliada isoladamente e o resultado final analisado.

Para o *seed*=7 uma rede com 61% de acerto foi obtida. A rede convergiu para um erro quadrático médio inferior a 0,0006 em torno de 5000 iterações, conforme visto na figura 17. Os resultados finais de previsão podem ser vistos na figura 18.

Figura 17 – Convergência do erro quadrático médio no treinamento da rede para o *seed*=7.

Para o *seed*=8 uma rede com 82% de acerto foi obtida. A rede não convergiu para um erro quadrático médio inferior a 0,0006 e 8000 iterações foram processadas, conforme visto na figura 19. Os resultados finais de previsão podem ser vistos na figura 20.

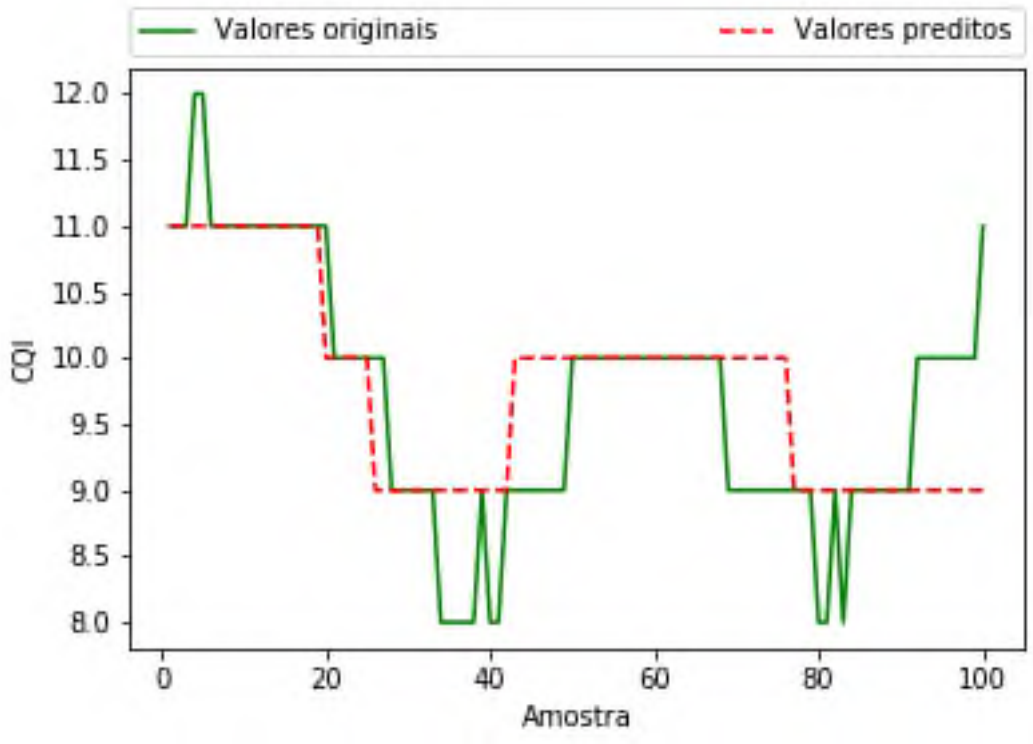

Figura 18 – Teste de previsão para o *seed*=7.

Para o *seed*=9 uma rede com 44% de acerto foi obtida. A rede não convergiu para um erro quadrático médio inferior a 0,0006 e 8000 iterações foram processadas, conforme visto na figura 21. Os resultados finais de previsão podem ser vistos na figura 22.

Figura 19 – Convergência do erro quadrático médio no treinamento da rede para o *seed*=8.

4.6 Avaliação preliminar dos resultados

O desempenho do algoritmo proposto é sintetizado a partir da tabela 4 a seguir, em que a porcentagem de acerto é tabelada em função do *seed*.

Seed	Porcentagem de acerto do algoritmo
7	61
8	82
9	44
10	76
Média	**65,75**

Tabela 4 – Resumo dos resultados obtidos.

O valor final médio de 65,75% de precisão da previsão do algoritmo proposto é um valor um pouco inferior ao obtido com o *seed*=10 adotado inicialmente. Porém, para uma comparação com (LUO et al., 2018) será avaliado o *ACR*, dado pela equação 8.

$$ACR = \frac{1}{T}\sum_{1}^{T}\frac{ValorPredito - ValorReal}{ValorReal} \tag{8}$$

Este parâmetro é particularmente importante na avaliação da predição pois, mesmo que o valor predito do CQI não seja numericamente igual ao valor real, eles podem estar próximos. Pode-se verificar a avaliação dos resultados em função do ACR na tabela 5

O algoritmo *Online CSI prEdiction scheme for 5G wireless communicAtioNs (OCEAN)* proposto em (LUO et al., 2018) apresentou um ACR médio entre 2,730% e 3,457%, po-

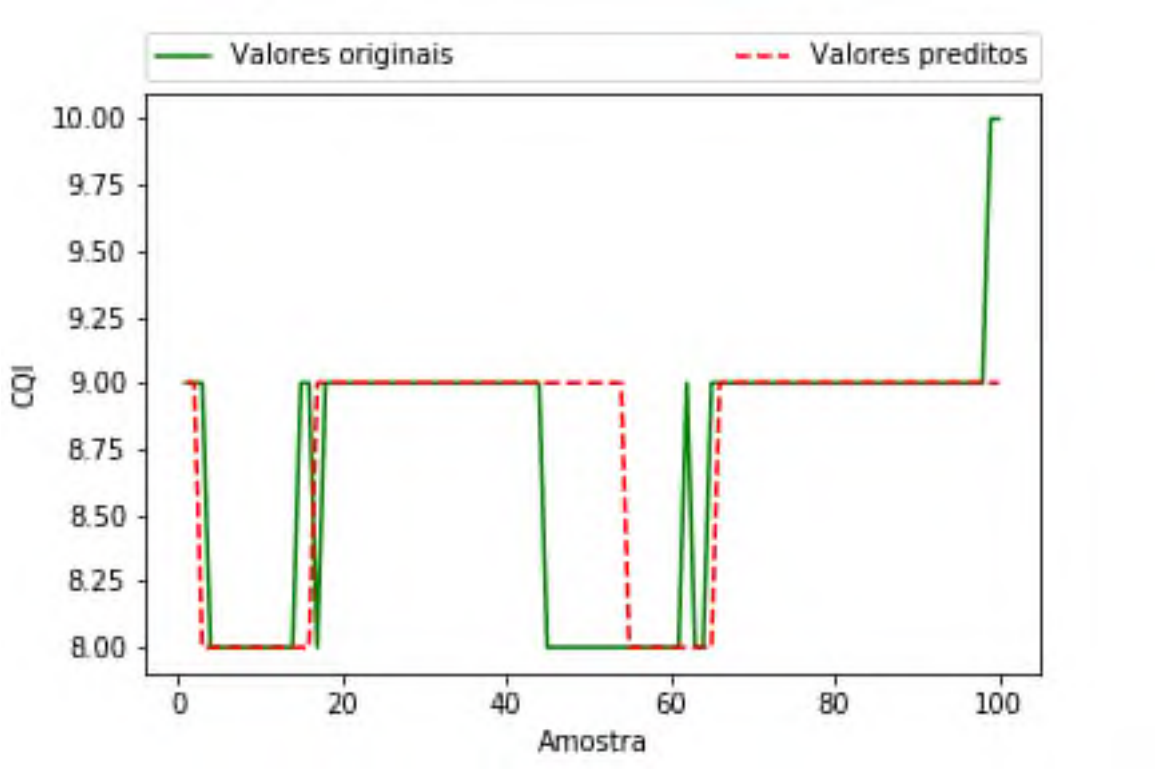

Figura 20 – Teste de previsão para o *seed*=8.

Seed	ACR
7	4,3560
8	2,1306
9	6,2957
10	4,4931

Tabela 5 – Valor dos ACR para os *seeds* simulados.

rém a rede desenvolvida é muito mais complexa, levando em consideração uma série de parâmetros extras e com dois sistemas adicionais de *Convolutional Neural Networks*.

Figura 21 – Convergência do erro quadrático médio no treinamento da rede para o *seed*=9.

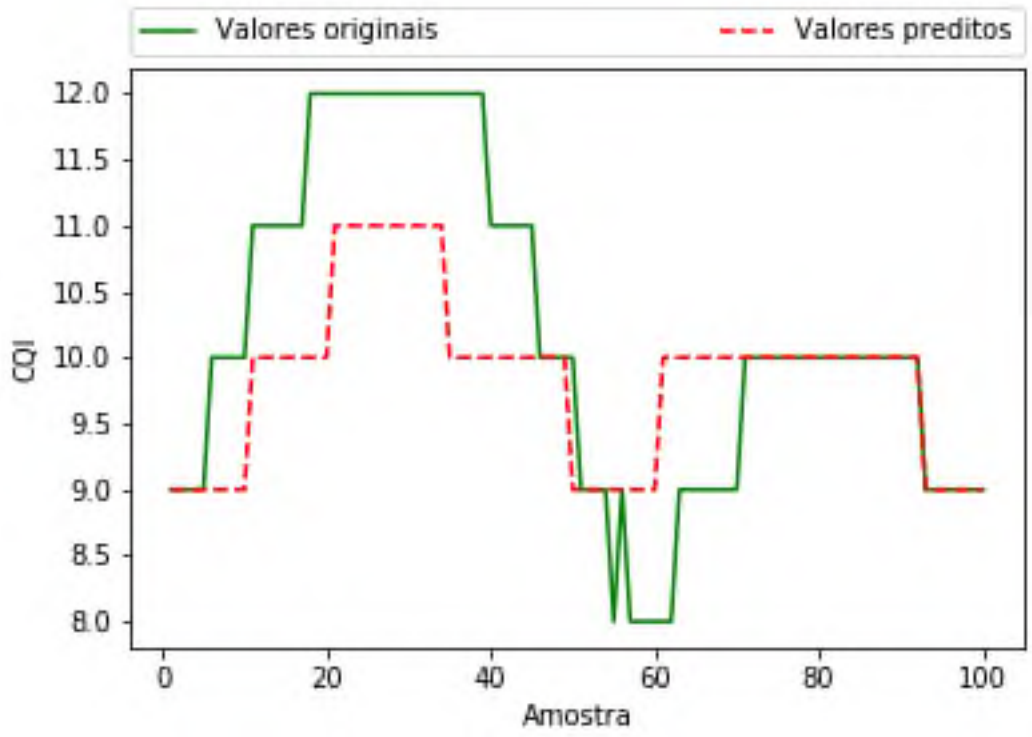

Figura 22 – Teste de previsão para o *seed*=9.

Capítulo 5

Conclusão

A grande ideia explorada neste trabalho é a da viabilidade do uso de redes neurais em sistemas de redes móveis, dado que:

1. Os eventos em uma rede móvel são sazonais (movimentação, variação do meio, etc);
2. A cada dia mais e mais aparelhos estarão na rede enviando uma quantidade cada vez maior de informações para a estação base, havendo uma base de dados gigante em potencial para ser explorada;
3. As redes de quinta geração serão totalmente virtualizadas, havendo a abertura para o desenvolvimento de uma série de subsistemas não previstos nas *releases* atualmente vistas;
4. O aumento do poder computacional dos nós na rede permite a implementação de algoritmos cada vez mais complexos.

Como foi abordado, ainda há uma série de desafios a serem vencidos nas redes de quinta geração, principalmente por causa da grande quantidade de usuários na rede e do aumento da demanda de tráfego. Muitos destes desafios poderão ser superados com a exploração de canais de frequência cada vez maiores e com o uso de técnicas como as micro-células. Aliado a futura virtualização de infraestrutura, a idealização do uso de redes neurais nas redes 5G já é um objeto de estudo em várias frentes para solucionar os desafios da nova geração de redes celulares.

5.1 Avaliação final dos resultados

Conforme visto a rede com os neurons LSTM apresentou um desempenho satisfatório para a previsão da qualidade do canal de comunicação. Apesar de um resultado médio de 65,75% de acerto obtido com a rede construída ser mediano, a análise a partir do ACR mostrou que a rede está com ótimos resultados, mesmo quando comparada a redes mais complexas como o algoritmo OCEAN (LUO et al., 2018).

5.2 Trabalhos Futuros

Há a proposta de três novas linhas para o aprimoramento da ideia:

A primeira ideia é a respeito do aprimoramento da rede, tanto no sentido de aprimorar a sua estrutura quanto a inclusão de novas variáveis para a análise. Conforme visto em (LUO et al., 2018), outras variáveis podem ser usadas para prever a qualidade do canal, como a frequência, temperatura, umidade do ar, horário do dia, etc.

A segunda ideia é a respeito da integração da técnica proposta com mecanismos de escalonamento de recursos que utilizam o CQI para tomada de decisões.

Extrapolando esta linha, pode-se pensar inclusive na implementação de redes de *Reinforcment Learning*, onde a rede é moldada em tempo real. Assim, com o passar do tempo, o sistema estaria continuamente sendo aprimorado e pronto para as mais diversas situações de mobilidade e qualidade do canal.

Comprovada a eficiência da caracterização e predição da qualidade do canal com redes neurais, um outro sistema poderia ser modelado na estação-base, no qual o CQI de diversos dispositivos estaria sendo recebido em tempo real e, em função da localização relativa dos aparelhos em rede, a qualidade do canal poderia ser prevista para cada dispositivo.

Referências

3RD GENERATION PARTNESHIP PROJECT. *Base Station (BS) radio transmission and reception*. [S.l.], 2018. Release 15. Disponível em: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3202>.

3RD GENERATION PARTNESHIP PROJECT. *Common test environments for User Equipment (UE) conformance testing*. [S.l.], 2018. Release 15. Disponível em: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2467>.

3RD GENERATION PARTNESHIP PROJECT. *User Equipment (UE) radio transmission and reception*. [S.l.], 2018. Release 15. Disponível em: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2411>.

BI, S. et al. Wireless communications in the era of big data. **IEEE Communications Magazine**, p. 190–199, 2015.

COLAH'S BLOG. **Understanding LSTM Networks**. 2015. Disponível em: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Acesso em: 23 mai. 2019.

DU, Z.; et al. Maximum likelihood based channel estimation for macrocellular ofdm uplinks in dispersive time-varying channels. **IEEE Tansactions on Wireless Communications**, p. 176–187, 2011.

ERICSSON. **The Ericsson Mobility Report**. [S.l.], 2019. Disponível em: <https://www.ericsson.com/assets/local/mobility-report/documents/2019/emr-q4-update-2018.pdf>.

GONZALEZ, R. C.; WOODS, R. E. **Processamento digital de imagens**. 3. ed. [S.l.]: Pearson, 2015.

GOODFELLOW, Y. B. e. A. C. I. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

HELIX, E. **5G physical layer specifications**. 2017. Acessado pela última vez em 3 de abril de 2019. Disponível em: <https://medium.com/5g-nr/5g-physical-layer-specifications-e025f8654981>.

HOCHREITER, J. S. S. Long short-term memory. **Neural Computation**, p. 1735–1780, 1997.

KARAMI, E. Tracking performance of least squares mimo channel estimation algorithm. **IEEE Tansactions on Communications**, p. 2201–2209, 2007.

KATO, N.; et al. The deep learning vision for heterogeneous network traffic control: Proposal, challenges, and future perspective. **IEEE Wireless Communications**, 2017.

KWAN, C. L. e. J. Z. R. Multiuser scheduling on the downlink of an lte cellular system. **Research Lett. Commun.**, 2008.

LUO, C. et al. Channel state information prediction for 5g wireless communications: A deep learning approach. **IEEE Tansactions on Network Science and Engineering**, 2018.

MA, J.; et al. Iterative lmmse individual channel estimation over relay networks with multiple antennas. **IEEE Tansactions on Vehicular Technology**, p. 423–435, 2018.

MATLAB. **Propagation Channel Models**. 2019. Disponível em: <https://www.mathworks.com/help/lte/ug/propagation-channel-models.html>. Acesso em: 17 mai. 2019.

PäTZOLD, C.-X. W. e. B. O. H. M. Two new sum-of-sinusoids-based methods for the efficient generation of multiple uncorrelated rayleigh fading waveforms. **IEEE Tansactions on Wireless Communications**, p. 3122–3131, 2009.

QUALCOMM. **Delivering on the LTE Advanced promise**. 2016. Disponível em: <https://www.qualcomm.com/documents/delivering-on-the-lte-advanced-promise>. Acesso em: 17 mai. 2019.

RAPPAPORT, T. S. et al. Overview of millimeter wave communications for fifth-generation (5g) wireless networks-with a focus on propagation models. **IEEE Transactions on Antennas and Propagation**, IEEE, p. 97–111, 2017.

SCIANCALEPORE, V. et al. Mobile traffic forecasting for maximizing 5g network slicing resource utilization. **IEEE Conference on Computer Communications**, 2017.

TENSORFLOW. **TensorFlow**. 2019. Disponível em: <https://www.tensorflow.org/>. Acesso em: 05 abr. 2019.

WANG, J. et al. Channel state information prediction for 5g wireless communications: A deep learning approach. **IEEE Conference on Computer Communications**, 2017.

WU, Y.; ET al. Google's neural machine translation system: Bridging the gap between human and machine translation. 2016.

Anexos

ANEXO A

Código Python para criação e teste da rede neural

```
# -*- coding: utf-8 -*-
"""TCC.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1KASWl8KG-z1vGccTZykfNGthS9CB1EIe

# Construcao da rede neural - Analise do CQI
A rede neural proposta vai analisar o report de CQI de um UE a uma BSS de LTE-A em 2500 frames. As configuracoes de simulacao sao:

- NFrames = 2500;

- SNRdB = 6.0.


Configuracao UE:

- RMC R.3;

- CFI = 1;

- HARQ desligado.
```

```
24
25 Modelo de propagacao:
26
27 - Seed = 10;
28 - 4 Antenas receptoras;
29 - Modelo de delay EPA;
30 - Doppler Freq = 5Hz;
31 - Alta correlacao (MIMO);
32 - Modelo de Rayleigh GMEDS;
33 - 16 osciladores no modelo.
34
35
36 Delay do CQI:
37
38 - CQI Delay = 8 subframes.
39
40 ## Importando bibliotecas
41 """
42
43 import numpy as np
44 import pandas as pd
45
46 import matplotlib.pyplot as plt
47 # %matplotlib inline
48
49 import tensorflow as tf
50
51 """## An lise dos dados gerados"""
52
53 cqi = pd.read_csv("CQIReport.csv", index_col = "Indice")
54 sinr = pd.read_csv("SINRReport.csv", index_col = "Indice")
55
56 cqi.head()
57
58 cqi.plot()
59
60 sinr.head()
61
62 """Conforme podemos ver, o Matlab teve problema para importar o
```

```
    SINR. Como eram valores com ate 3 casas decimais apos a
    virgula, o Matlab salvou estes valores sem discernir as casas
     decimais, por isso e necessario um codigo de adaptacao dos
    dados antes de qualquer coisa."""
63
64 for i in range (1, len(sinr)+1):
65   if sinr['SINR'][i] > 1000:
66     sinr['SINR'][i] = sinr['SINR'][i] * 0.001
67
68 sinr.head()
69
70 sinr.plot()
71
72 """Plotando as duas imagens lado a lado, podemos ter uma nocao
    da dependencia do SINR que o CQI tem."""
73
74 x = range(0, 12500)
75 plt.subplot(2, 1, 1)
76 plt.title("Valores ao longo da simula  o")
77 plt.plot(x, sinr['SINR'], 'g', label='SINR')
78 plt.ylabel('Amplitude (dB)')
79 plt.legend()
80
81 plt.subplot(2,1,2)
82 plt.plot(x, cqi['CQI'], 'r', label='CQI')
83 plt.xlabel('Report')
84 plt.ylabel('Valor reportado')
85 plt.legend()
86 plt.savefig('Resultadosfull.png')
87
88 """Analisando somente uma parcela dos resultados, podemos ter a
    mesma confirmacao."""
89
90 x = range(0, 200)
91 plt.subplot(2, 1, 1)
92 plt.title("Valores ao longo da simula  o")
93 plt.plot(x, sinr['SINR'][0:200], 'g')
94 plt.ylabel('Amplitude (dB)')
95 plt.legend()
```

```
96
97
98  plt.subplot(2,1,2)
99  plt.plot(x, cqi['CQI'][0:200], 'r')
100 plt.xlabel('Report')
101 plt.ylabel('Valor reportado')
102 plt.legend()
103 plt.savefig('Resultados.png')
104
105 """## Transformacao do indice em uma serie temporal
106 Aqui, uma "artimanha" foi usada. Para transformar a serie de
       reports do CQI em uma serie temporal, a coluna de Indice foi
       transformada em "datetime". Inicialmente, isto implicara em
       um erro, ja que a coluna ficara inteiramente confusa, mas
       mais adiante isto sera crucial para o funcionamento da RNN.
107 """
108
109 cqi.index = pd.to_datetime(cqi.index)
110
111 cqi.head()
112
113 """## Train Test Split
114 Divisao dos CQI reportados entre base de treinamento e base de
       teste.
115 """
116
117 cqi.info()
118
119 train_set = cqi.head(12400)
120 test_set = cqi.tail(100)
121
122 """## Adapta  o dos dados
123
124 Os dados serao adaptados pelo algoritmo MinMaxScaler para que
       possam ser processados.
125 """
126
127 from sklearn.preprocessing import MinMaxScaler
128
```

```
129 scaler = MinMaxScaler()
130
131 train_scaled = scaler.fit_transform(train_set)
132 test_scaled = scaler.transform(test_set)
133
134 """## Fun  o de batch
135
136 Para diminuir a complexidade do script, uma funcao para
      alimentar a rede em batches (lotes) sera usada.
137 """
138
139 def next_batch(training_data,batch_size,steps):
140
141     rand_start = np.random.randint(0,len(training_data)-steps)
142
143     y_batch = np.array(training_data[rand_start:rand_start+steps
        +1]).reshape(1,steps+1)
144
145     return y_batch[:, :-1].reshape(-1, steps, 1), y_batch[:,
        1:].reshape(-1, steps, 1)
146
147 """## Constru  o da rede
148
149 Primeiramente vamos setar as variaveis da rede, que sao:
150
151 - Numero de entradas: quantas variaveis serao usadas como input
      na rede (no caso, apenas uma);
152 - Numero de variaveis em cada batch;
153 - Numero de neurons: quantidade de celulas que a rede neural
      tera;
154 - Numero de saidas: quantas saidas a rede tera (no caso, apenas
      uma);
155 - Ritmo de aprendizado: variavel que dita o tamanho do "salto"
      entre cada iteracao, ditando o ritmo de aprendizado. Este
      valor deve ser equilibrado para uma solucao convergente da
      rede;
156 - Numero de interacoes: quantas vezes os dados serao processados
       para a solucao;
157 - Tamanho do batch.
```

```
158 """
159
160 # Numero de entradas
161 num_inputs = 1
162 # Numero de variaveis em cada batch
163 num_time_steps = 100
164 # Numero de neurons
165 num_neurons = 100
166 # Numero de saidas
167 num_outputs = 1
168 # Ritmo de aprendizado
169 learning_rate = 0.001
170 # Numero de interacoes
171 num_train_iterations = 8000
172 # Tamanho do batch
173 batch_size = 1
174
175 """### Criando "espacos" (placeholders) para as variaveis de
      entrada e saida na rede."""
176
177 X = tf.placeholder(tf.float32, [None, num_time_steps, num_inputs
      ])
178 y = tf.placeholder(tf.float32, [None, num_time_steps,
      num_outputs])
179
180 """### Configuracao da celula da rede
181 Aqui a celula usada na rede e configurada. Conforme podemos ver
      (pela linha comentada e a sua equivalente usada no codigo) a
      configuracao das celulas LSTM e GRU e bastante simples de ser
       feita.
182 """
183
184 cell = tf.contrib.rnn.OutputProjectionWrapper(
185     tf.contrib.rnn.BasicLSTMCell(num_units=num_neurons,
          activation=tf.nn.relu),
186     #tf.contrib.rnn.GRUCell(num_units=num_neurons, activation=tf
          .nn.relu),
187     output_size=num_outputs)
188
```

```
189 outputs, states = tf.nn.dynamic_rnn(cell, X, dtype=tf.float32)
190
191 """### Criacao da funcao de erro/perda"""
192
193 loss = tf.reduce_mean(tf.square(outputs - y)) # MSE
194 optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate)
195 train = optimizer.minimize(loss)
196
197 """### Inicializando as variaveis globais"""
198
199 init = tf.global_variables_initializer()
200
201 """### Criando a instancia de salvar
202 Para que a rede possa ser usada na predicao, ela precisa ser
      salva dentro da sessao de treinamento.
203 """
204
205 saver = tf.train.Saver()
206
207 """## Inicio da sessao da rede
208 A sessao sera iniciada e a cada 100 interacoes o erro quadratico
      medio (MSE) sera plotado.
209 """
210
211 i=1
212 mse_plot = []
213 with tf.Session() as sess:
214     sess.run(init)
215
216     for iteration in range(num_train_iterations):
217
218         X_batch, y_batch = next_batch(train_scaled,batch_size,
              num_time_steps)
219         sess.run(train, feed_dict={X: X_batch, y: y_batch})
220
221         if iteration % 100 == 0:
222
223             mse = loss.eval(feed_dict={X: X_batch, y: y_batch})
224             print(iteration, "\tMSE:", mse)
```

```
225             if iteration > 4000 and mse < 0.0006:
226               break;
227             mse_plot.append(mse)
228 
229     # Salvando o modelo
230     saver.save(sess, "rede")
231 
232 plt.plot(mse_plot[2:80])
233 plt.xlabel('Centenas de itera  es')
234 plt.ylabel('Erro quadr tico m dio')
235 plt.title('Erro quadr tico m dio em fun  o da itera  o')
236 plt.savefig('mse.png')
237 
238 """## Teste de previsao
239 A previsao da rede funciona da seguinte forma:
240 
241 1.  A rede ja treinada e restaurada na sessao de avaliacao;
242 2.  O dado em t e inserido na rede ja treinada;
243 3.  O valor predito de t+1  salvo.
244 """
245 
246 with tf.Session() as sess:
247     saver.restore(sess, "./rede")
248 
249     train_seed = list(train_scaled[-100:])
250 
251     for iteration in range(100):
252         X_batch = np.array(train_seed[-num_time_steps:]).reshape
              (1, num_time_steps, 1)
253         y_pred = sess.run(outputs, feed_dict={X: X_batch})
254         train_seed.append(y_pred[0, -1, 0])
255 
256 """Os resultados sao passados para o conjunto de dados de teste
    em uma nova coluna "Generated"""
257 
258 results = scaler.inverse_transform(np.array(train_seed[100:]).
    reshape(100,1))
259 
260 test_set['Generated'] = results
```

```
261
262 """Conforme podemos ver a seguir, a base de teste tem 3 colunas:
263
264
265 *   Indice: valor que representa a passagem temporal dos frames.
     Conforme citado anteriormente, ele foi manipulado e
    apresenta-se "errado";
266 *   CQI: valor original, oriundo das simula  es;
267 *   Generated: valores preditos pela rede.
268 """
269
270 test_set.head()
271
272 """## Arredondamento dos resultados e plotagem
273 Conforme visto anteriormente, a rede neural tenta prever o CQI
    do canal, porem na sua saida ha numeros com virgula. Como o
    valor do CQI e um numero inteiro, e necessario realizar o
    arredondamento dos valores da coluna "Generated"
274 """
275
276 test_set['Generated'] = round(test_set['Generated'])
277
278 test_set
279
280 """Para realizar a correta plotagem, a correcao da base temporal
     sera feita. Como os reports do CQI nao tem base temporal,
    mas sim ordem, usaremos a variavel "x" gerada a seguir para
    realizar a plotagem dos resultados"""
281
282 x = range(1, len(test_set)+1)
283
284 plt.plot(x, test_set['CQI'], 'g', label = "Valores originais")
285 plt.plot(x, test_set['Generated'], 'r', label = "Valores
    preditos")
286 plt.legend(bbox_to_anchor=(0., 1.02, 1., .102), loc=3,
287            ncol=2, mode="expand", borderaxespad=0.)
288 plt.ylabel('CQI')
289 plt.xlabel('Registro')
290 plt.savefig('Resultadossimulacao.png')
```

```
291
292 """# Avaliacao dos resultados"""
293
294 total = 0
295 acertos = 0
296 i = 0
297
298 adr = 0
299
300 for i in range (0, num_time_steps):
301   if test_set['CQI'][i] == test_set['Generated'][i]:
302     acertos=acertos+1
303   total=total+1
304
305 for i in range (0, num_time_steps):
306     adr = adr + abs((test_set['Generated'][i] - test_set['CQI'][
        i])) / test_set['CQI'][i]
307
308 adr = 100*adr/num_time_steps
309
310 porcentagem_acertos = (acertos/total) * 100
311 print ("Em ", total, "subframes totais,houve um total de ",
    acertos, "acertos, sendo uma assertividade de ",
    porcentagem_acertos, "%")
312
313 print ("ADR = ", adr)
```