Redes de Aprendizado Profundo para Classificação e Controle de Congestionamento em Redes TCP/IP

Cesar Augusto C. Marcondes[1], **Marcelo R. da Siva**[1]

[1]Divisão de Ciência da Computação – ITA
São José dos Campos – SP – Brasil

cesar.marcondes@gp.ita.br, marcelo.silva@ga.ita.br

Abstract. *Nowadays, digital networks are present in practically all human activity areas. TCP model is the great driver of this rapid escalation, particularly of the Internet. Such a scenario, however, poses serious challenges to Congestion Control (CC). Seeking an efficient CC mechanism, this work proposes the construction of deep learning neural networks (MLP, LSTM and CNN) to classify network congestion, based on the buffer occupancy level of an edge router. The results attest models that can distinguish, with more than 90% accuracy, between high and low congestion levels, capable to increase the throughput by five times.*

Resumo. *Atualmente as redes digitais permeiam praticamente todas as áreas de atividade humana. O modelo TCP é o grande propulsor desta rápida escalada, em particular da Internet. Tal cenário, porém, impõe sérios desafios ao Controle de Congestionamento (CC). Buscando um mecanismo de CC eficiente, este trabalho propõe a construção de redes neurais de aprendizado profundo (MLP, LSTM e CNN) para classificação o nível congestionamento de uma rede, por meio da ocupação do buffer de um roteador de borda. Os resultados atestam modelos capazes distinguir, com mais de 90% acerto, entre momentos de alto e baixo grau de congestionamento, podendo elevar o throghput em até cinco vezes.*

1. Introdução

As redes digitais, regidas pela camada de transporte TCP, são praticamente imprescindíveis no dia a dia da sociedade moderna. Tal constatação se deve principalmente à maciça difusão de serviços disponibilizados na internet, que podem ser acessados dos mais diversos tipos de terminais, dotados das mais variadas arquiteturas. Por isso, o acesso à internet passou rapidamente de uma exceção para uma regra, uma vez que está ao alcance de 95% da população mundial [ITU 2023].

Por trás da internet, existe um conjunto de convenções pré-definidas pelo protocolo TCP, que viabiliza a comunicação entre os terminais de uma rede IP. Embarcados nos diversos sistemas operacionais, tais convenções, ao serem seguias, permitem às aplicações trocarem dados entre si, acionando um conjunto de serviços organizadas em camadas que dão origem à pilha de protocolos conhecida como TCP/IP.

Um dos componentes da Pilha TCP é a Camada de Transporte, responsável pelo Controle de Congestionamento (CC). O CC regula a quantidade de dados inserida na infraestrutura de comunicações a cada rodada de transmissão. Dessa forma, constitui

parte extremamente relevante nas transmissões TCP, pois é o módulo que efetivamente regula as solicitações dos canais de transmissão por parte dos seus usuários.

Todo CC precisa de meios para estimar o grau de saturação do canal utilizado. Como as Redes de Aprendizado profundo são estruturas computacionais capazes de gerar classificadores extremamente eficientes, as Multilayer Perceptron (MLP), Long Short-Term memory (LSTM) e Convolutional neural network (CNN) despontam como arquiteturas de aprendizado promissoras para otimização de CC, como poderá ser observado no decorrer deste trabalho.

Embalado pelos desafios impostos aos mecanismos de CC e o atual estágio de desenvolvimento das redes neurais, este trabalho se pauta pela investigação das seguintes *Research Questions* (RQ):

- **RQ1: (Aprendizado)** - É possível que uma rede neural aprenda a classificar o grau de congestionamento de uma conexão, tomando por base a ocupação do roteador de saída?
  - **RQ1.1: (Tolerância à variação da quantidade de fluxos)** - O aprendizado é tolerante à variação na quantidade de fluxo estabelecidos na rede?
  - **RQ1.2: (Performance para outros CC)** - Este aprendizado pode ser generalizado para fluxos regidos por mecanismos de CC distintos?
- **RQ2: (Dimensão ótima)** - Qual combinação de medidas (componentes/dimensão) é suficiente para construção de classificadores de congestionamento eficientes?
- **RQ3: (Ganhos com o aprendizado)** - O modelo extraído de uma rede neural, treinada com vetores de baixa dimensão, pode, efetivamente, otimizar a utilização da banda disponível?

Para facilitar a investigação das questões acima, foi definida uma metodologia ilustrada pela figura a seguir:

Figura 1. Metologia empregada.

Na ilustração, é possível identificar o resultado que cada etapa recebe da anterior, encadeadas da seguinte forma: Coleta dos Dados, Preparação dos Dados, Treinamento das Redes Neurais - Obtenção dos Modelos, Avaliação e seleção dos Modelos, Otimização de CC, que compõem, nesta ordem, a organização deste artigo, após a apresentação dos Trabalhos Relacionados.

2. Trabalhos Relacionados

Muitas pesquisas sugerem a construção de CC baseados em princípios de Aprendizado de Máquina. Um exemplo típico dessa tendência é o Remmy [Winstein and Balakrishnan 2013], que se utiliza de uma tabela de ações, modelada para maximizar o desempenho numa rede aderente a pressupostos assumidos durante a fase de treinamento. No Remmy, as medidas ack_ewma, snd_ewma e rtt_ratio, mais bem

detalhadas na sessão seguinte, são apresentadas como indicadoras do grau de congestionamento de uma rede. Infelizmente, passada a fase de treinamento, as reações às flutuações da rede permanecem fixas, o que faz com que o Remy sofra das mesmas restrições de abordagens que se utilizam de funções fixas para atualização da *cwnd*.

Outros trabalhos permanecem explorando os benefícios do Reinforcement Learning (RL) [Sutton and Barto 2015] na construção de tabelas que mapeiam o estado apresentado por uma rede em uma ação (atualização da *cwnd*), mais promissora. Alguns desses trabalhos se baseiam em treinamento prévio a fim de extrapolarem CC mais flexíveis, tais como [Jay et al. 2019], [Li et al. 2016], [Abbasloo et al. 2020]; outros, na mudança *online* de políticas para a escolha de ações, tais como [Li et al. 2019], [Badarla and Murthy 2011] e [Ramana et al. 2005]. Entretanto, métodos baseados em RL sofrem com a natureza contínua do espaço de estados ao longo de uma transmissão, podendo até colapsar em alguns casos, como ressaltado em [Emara et al. 2020].

As técnicas de *Deep Learning* também surgem como opção promissora para predição de tráfego de rede. Há esforços diretamente relacionados ao tema, com utilização de LSTM [Hochreiter and Schmidhuber 1997]. Exemplos neste sentido podem ser encontrados em [Kazama et al. 2022] e [Bai et al. 2020]. Entretanto, as abordagens propostas por estes trabalhos não incluem as redes MLP simples[Lippmann 1987], nem as CNN [Fukushima 1980]. Além disso, se utilizam de uma quantidade expressiva de parâmetros, obtendo precisão abaixo de 80%, mesmo em cenários de baixa complexidade.

Buscando contribuir com o tema, esta pesquisa faz uma comparação entre as principais técnicas de Deep Learnin utilizando as Arquiteturas de Redes Neurais (ARN). Para isto, as arquiteturas MLP, LSTM e CNN, são comparadas na classificação do nível de congestionamento de uma rede TCP/IP. O trabalho também investiga uma dimensão ótima para os vetores de entradas destas redes neurais. A presente pesquisa confronta a eficiência dos modelos construídos em cenários extremamente complexos (chegando a 40 fluxos), regidos por diferentes variantes TCP.

Além das contribuições já mencionadas, a pesquisa prossegue, buscando demonstrar os ganhos efetivos de um classificador de CC eficiente. Para isso, é apresentada uma maneira de se integrar os modelos construídos às implementações de CC, de modo a otimizá-lo. Em seguida, a performance dos CC assistidos pelos modelos extraídos das Redes Neurais são comparados com algumas das variantes TCP mais comumente disponibilizados nos Sistemas Operacionais.

3. Coleta dos Dados

Para aquisição dos dados de treinamento e testes, foi desenvolvido em ns-3 [ns3 2023] um Analisador de Tráfego (AT), que cria dois arquivos .csv para cada fluxo TCP estabelecido, numa topologia *dumbell*. No primeiro arquivo, a cada ACK recebido pelo transmissor, são registrados os seguintes parâmetros (figura 2):

- #seq: Número de sequencia presente no pacote ACK.
- ack_ewma: Média móvel exponencial ponderada do tempo de chegada entre pacotes ACK.
- snd_ewma: Média móvel exponencial ponderada do tempo decorrido entre o envio de um seguimento e a chegada do respectivo ACK.
- rtt_ratio: Calculado a partir da divisão dos RTT pelo RTT_{min}.

O segundo arquivo é atualizado da seguinte forma: Ao ser enviado um segmento no tempo t, o simulador agenda um evento em $t+RTT_{router}/2$, em que RTT_{router} é o RTT esperado entre o transmissor e o roteador de borda. Neste evento, são registrados os seguintes parâmetros:

- #seq: Número de sequencia presente no segmento para o qual foi agendado o evento.
- lro_ewma: Média móvel exponencial ponderada do percentual de ocupação do *buffer* do roteador de borda, ao qual está conectado o transmissor (Roteador 01 na figura 2).
- cng_nvl: Nível congestionamento da rede. Se $lro\_ewma <= 40\%$, $cng\_nvl = 1$ ou $cng\_nvl = 2$ caso $lro\_ewma >= 70\%$.

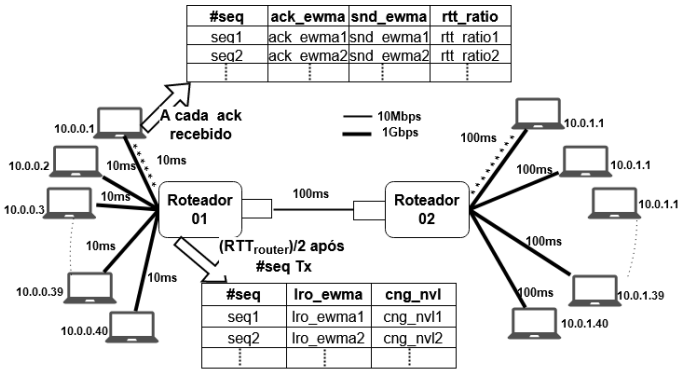


Figura 2. Coleta dos Dados

Todas as médias móveis foram calculadas em milissegundos(ms), com peso 0.8 para novas medidas, a fim de se privilegiar a condição momentâneas da rede. Além disso, visando um treinamento mais preciso da rede, o analisador foi configurado para registrar dados apenas em situações em que $lro\_ewma <= 40\%$ ou $lro\_ewma >= 70\%$. Esse design visa deixar o modelo mais robusto na classificação de situações extremas, de subutilização ou de sobrecarga.

Para o treinamento, os arquivos, atualizados conforme descrito acima, vieram de uma simulação ns3 de 2h, em que, na topologia dumbell apresentada, foram estabelecidas 20 conexões p2p TCP NewReno. As aplicações TCP estão nos terminais conectados ao roteador 01 e os servidores TCP naqueles ligados ao roteador 2. O terminal 10.0.0.2 do roteador 01 se liga ao terminal 10.1.0.2 do roteador 2. Da mesma forma, o terminal 10.0.2.2 com o 10.1.2.2; o 10.0.3.2 com 10.1.3.2 e assim sucessivamente até a última conexão, entre as estações 10.0.19.2 e 10.1.19.2. Os canais das estações ligadas ao Roteador 1 possuem delay de 10ms. Em todos os demais, este tempo é de 100ms. A taxa de transmissão é de 1Gbs em todos os canais, com exceção daquele entre os roteadores, que é de 10Mbps.

O início das conexões são espaçados de 10s para evitar sincronização. As aplicações dos terminais foram configuradas para agendarem envio de dados sempre que houver espaço na fila de transmissão do socket. Uma vez identificado tal espaço, cada aplicação aguarda um intervalo de tempo aleatório entre 0 e 17,25ms para, só então, esgotar a referida fila novamente. A amplitude do intervalo foi escolhida de forma a não

prejudicar a intensidade dos fluxos e, ao mesmo tempo, tornar as simulações mais realistas quanto à aleatoriedade no envio de dados pelas aplicações.

4. Preparação dos Dados

Os dados levantados pelo (AT) passam pelos seguintes estágios de preparação, antes de formarem os vetores de entrada das redes neurais:

1. *Inner join* (IJ): A primeira etapa consiste gerar uma tabela que associe as medidas ack_ewma, snd_ewma e rtt_ratio, com lro_ewma e cng_nvl, por meio das colunas #seq, presentes nos arquivos levantados pelo AT.
2. Eliminação de redundâncias (ER): Consiste no descarte das linhas com ack_ewma, snd_ewma e rtt_ratio idênticos, mantendo-se o último registro ao longo da simulação.
3. Balanceamento de Classes (BC): Consiste em balancear os registros do AT, fazendo com que a quantidade de linhas com $cng\_nvl = 1$ seja igual a daquelas com $cng\_nvl = 2$.
4. Eliminação de atenuadores (EA): Registros que apresentem ack_ewma, snd_ewma e rtt_ratio acima do nonagésimo percentil (P_{90}) são descartados.
5. Normalização das Entradas (NE): Cada uma das colunas passa por um processo de normalização, em que suas medidas são divididas pelo valor máximo nela presente. Posteriormente, o valor do rtt_ratio é efetivamente calculado, quando o menor valor dos RTT normalizados divide os demais. Os normalizadores levantados dos dados de treinamento serão, posteriormente, empregados em todos os demais dados que vierem a adentrar os modelos construídos.

A partir de então, a tabela resultante é utilizada para construir o conjunto X, em que cada elemento x_i é um vetor formado pelas componentes (ack_ewma,snd_ewma,rtt_ratio) devidamente tratadas. Cada x_i recebe uma classe $y_i \in Y = \{0, 1\}$, igual a 0 ou 1, de acordo com o valor de cng_nvl. Após as etapas acima mencionadas, chegou-se a um conjunto X de 9.461 entradas.

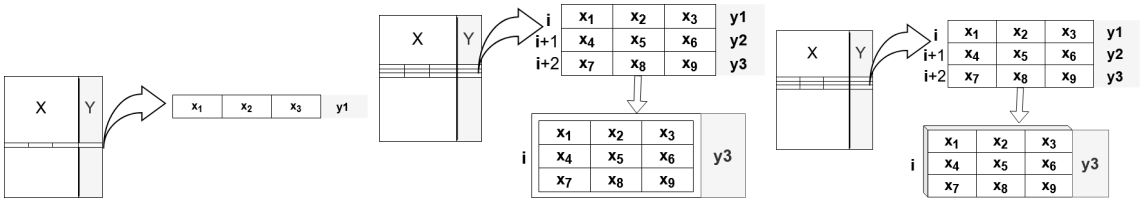
5. Treinamento das redes Neurais - Obtenção dos Modelos

5.1. Especificação dos Modelos

Para alcançar os objetivos da pesquisa, foram gerados 12 modelos, variando-se as ARN (MLP, LSTM, CNN), cujas especificações estão na tabela 1, associada a combinações das componentes (ack_ewma,snd_ewma,rtt_ratio). Assim sendo, os modelos passam a ser designados pelo tipo de rede neural e as respectivas componentes dos vetores de entrada. MLP_{123}, por exemplo, remete ao modelo obtido pela rede MLP, recebendo como entrada as três componentes. LSTM_{13} significa modelo obtido pela rede neural LSTM, recebendo como entrada (ack_ewma,rtt_ratio); CNN_{23} à CNN, que recebe (snd_ewma,rtt_ratio).

5.2. Transformações dos Vetores de Treinamento

Para cada uma das ARN houve necessidade de transformação dos vetores do conjunto X. A descrição das transformações será baseada na utilização de vetores de três dimensões, formado pelas componentes ack_ewma, snd_ewma e rtt_ratio por ser análoga às demais combinações destas componentes. Para as MLP, por exemplo, a transformação é bem direta, sem qualquer rearranjo especial (figura 3a).



(a) Transformação MLP. **(b) Transformação LSTM.** **(c) Transformação CNN.**

Figura 3. Transformações de dados para as ARN.

As ARN baseadas em LSTM e CNN precisaram de transformações mais elaboradas. Nas LSTN, considerando que X possua m elementos, cada elemento x_i $(i+2 \leq m)$, acompanhada de dois vetores consecutivos, dará origem a uma entrada da rede LSTM, associada à classe do terceiro vetor da sequência (figura 3b). Para a arquitetura CNN, os vetores X também foram grupados consecutivamente de 3 em 3 e posteriormente reformatados, formando uma espécie de "imagem" de 3x3 pixels com um canal. (figura 3c).

Tabela 1. Redes neurais utilizadas

MLP	LSTM	CNN
-Entrada: Array de três dimensões (ver seção 5.1). -Camadas internas: 3 (20 RELU cada). -Saída: Sigmoid.	-Entrada: Matriz 3x3 (ver seção 5.1). -Camadas Internas: 2 (3 LSTM cada; *dropout* de 30%). -Saída: Sigmoid.	-Entrada: Vetores como imagem 3X3@1(ver seção 5.1). -Convolução:16 mapas de características 1x3 -Stride: [1,1] -Saída Convolução: RELU -Pooling: *maxpooling*, [1,2]. -Flattening: Alimenta 64 unidades RELU. Saída: Sigmoid.

Todos os treinamentos foram realizados com 3000 épocas, batch size igual a 64 e taxa de aprendizado de 0.0001, utilizando a biblioteca keras da configuração default do ambiente Google Colab. Das 9461 entradas, 20% foram usadas para testes e o restante para treinamento/validação.

6. Avaliação e Seleção dos Modelos (RQ1)

6.1. Análise dos Resultados - Dados de Teste

No que diz respeito aos dados de teste, tanto a distribuição dos pontos no espaço ROC (figura 4) quanto os valores da tabela 2 apontam para o bom desempenho da arquitetura CNN_{123}. No espaço ROC, o referido modelo é o que mais se aproxima do cando superior esquerdo da área do gráfico. Na tabela, a arquitetura CNN treinada com as três componentes apresenta melhor desempenhos em 4 das 5 métricas apresentadas, perdendo por muito pouco $(0,069)$ para a MLP_{13} no quesito precisão. Os modelos MLP_{23}, juntamente com CNN_{13} e CNN_{23} também despontam como os de menores FPR.

Tabela 2. Desempenho dos modelos sobre os dados de Teste

Modelo	Acurácia	Erro	Precisão	Recall	F1
MLP_{123}	$0,900$	$0,100$	$0,882$	$0,913$	$0,897$
MLP_{13}	$0,895$	$0,105$	$0,869$	$0,912$	$0,890$
MLP_{23}	$0,876$	$0,124$	$0,983$	$0,809$	$0,887$
MLP_{12}	$0,672$	$0,328$	$0,678$	$0,667$	$0,672$
$LSTM_{123}$	$0,891$	$0,109$	$0,854$	$0,921$	$0,886$
$LSTM_{13}$	$0,878$	$0,122$	$0,872$	$0,869$	$0,871$
$LSTM_{23}$	$0,862$	$0,138$	$0,907$	$0,828$	$0,866$
$LSTM_{12}$	$0,695$	$0,305$	$0,717$	$0,676$	$0,696$
CNN_{123}	$0,926$	$0,074$	$0,914$	$0,931$	$0,923$
CNN_{13}	$0,869$	$0,131$	$0,981$	$0,794$	$0,878$
CNN_{23}	$0,863$	$0,137$	$0,982$	$0,785$	$0,873$
CNN_{12}	$0,668$	$0,332$	$0,638$	$0,678$	$0,657$

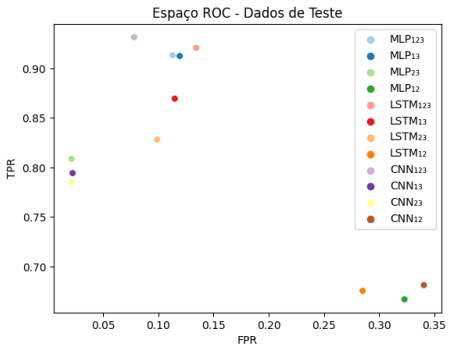


Figura 4. Espaço ROC - dados de teste

Destacam-se ainda, positiva ou negativamente, outras estratégias. Com acurácia próxima a 90%, os modelos MLP_{123}, $LSTM_{123}$, $LSTM_{13}$ despontam como bem promissores. Já os MLP_{23}, $LSTM_{23}$ e CNN_{23} produziram bons resultados, mas com acurácia entre 86% e 88%. Os modelos treinados com ack_ewma e $send\_ewma$ apresentaram os piores resultados, com acertos abaixo de 70%, ocupando o canto inferior direito no espaço ROC. Sendo assim, os modelos MLP_{12}, $LSTM_{12}$ e CNN_{12} não farão parte das próximas análises apresentadas.

> **Constatação 1:** É possível que uma rede neural aprenda e gere modelos capazes de classificar o nível de congestionamento de uma rede, com mais de 90% de precisão, tomando por base o nível de ocupação do roteador de saída.

6.2. Capacidade de generalização dos modelos (RQ1.1, RQ1.2, RQ 2)

Os modelos acima obtidos tiveram sua capacidade de generalização medida sobre dados de fluxos que não participaram do treinamento. Para isso, foram executadas novas simulações na topologia da figura 2, nos mesmos moldes daquela realizada para levantamento dos dados de treinamento, variando-se, entretanto, o número de estações em cada roteador (5, 10, 20, 40) e o CC (NewReno, CUBIC, Bbr e Vegas). No final, foram consideradas 15 simulações, uma vez que o TCP-Vegas, para 05 fluxos, não produziu ocupação acima de 70% no buffer do roteador de borda.

Cada uma das simulações forneceu, após o processo de preparação dos dados, um conjunto X de 4000 entradas. Dessas 4000 entradas, 70% (2.800) foram selecionadas aleatoriamente, transformadas e classificadas por cada uns dos 09 modelos com desempenho plausível (seção 6.1). Os valores médios das métricas é apresentada na tabela 3:

De uma forma geral, os modelos demonstraram excelente capacidade de generalização. Na tabela 3, estão destacados, em cada coluna, os dois valores correspondentes ao melhor desempenho, incluindo os empates. A partir disso, percebe-se que, novamente, os modelos MLP_{13}, CNN_{123} e CNN_{13} sobressaem.

Tabela 3. Desempenho médio dos modelos sobre os dados que não participaram dos treinos - Generalização

Modelo	$\overline{\text{Acurácia}}$	$\overline{\text{Erro}}$	$\overline{\text{Precisão}}$	$\overline{\text{Recall}}$	$\overline{\text{F1}}$
MLP_{123}	$0,913$	$0,024$	$0,904$	$0,922$	$0,913$
MLP_{13}	$0,906$	$0,031$	$0,889$	$0,923$	$0,905$
MLP_{23}	$0,917$	$0,020$	$0,904$	$0,934$	$0,918$
LSTM_{123}	$0,910$	$0,027$	$0,899$	$0,920$	$0,909$
LSTM_{13}	$0,902$	$0,035$	$0,881$	$0,922$	$0,901$
LSTM_{23}	$0,895$	$0,043$	$0,866$	$0,926$	$0,894$
CNN_{123}	$0,916$	$0,021$	$0,906$	$0,926$	$0,916$
CNN_{13}	$0,916$	$0,022$	$0,902$	$0,933$	$0,917$
CNN_{23}	$0,910$	$0,024$	$0,895$	$0,928$	$0,911$

Constatação 1.1 e 1.2: É possível construir classificadores de congestionamento tolerantes à variações no CC e na quantidade de fluxos estabelecidos.

O bom desempenho dos modelos acima citados também pode ser constatado no espaço ROC correspondente (figura 5), levantado a partir da matriz de confusão obtida a cada rodada de classificação. A densidade de pontos em torno da posição (0,1) e da reta TPR=1 reafirmam os modelos MLP_{13}, CNN_{123} e CNN_{13} como os de melhor desempenho. Merece destaque o modelo CNN_{23}, com boa distribuição no espaço ROC, ficando, entretanto, aquém dos três mencionados anteriormente.

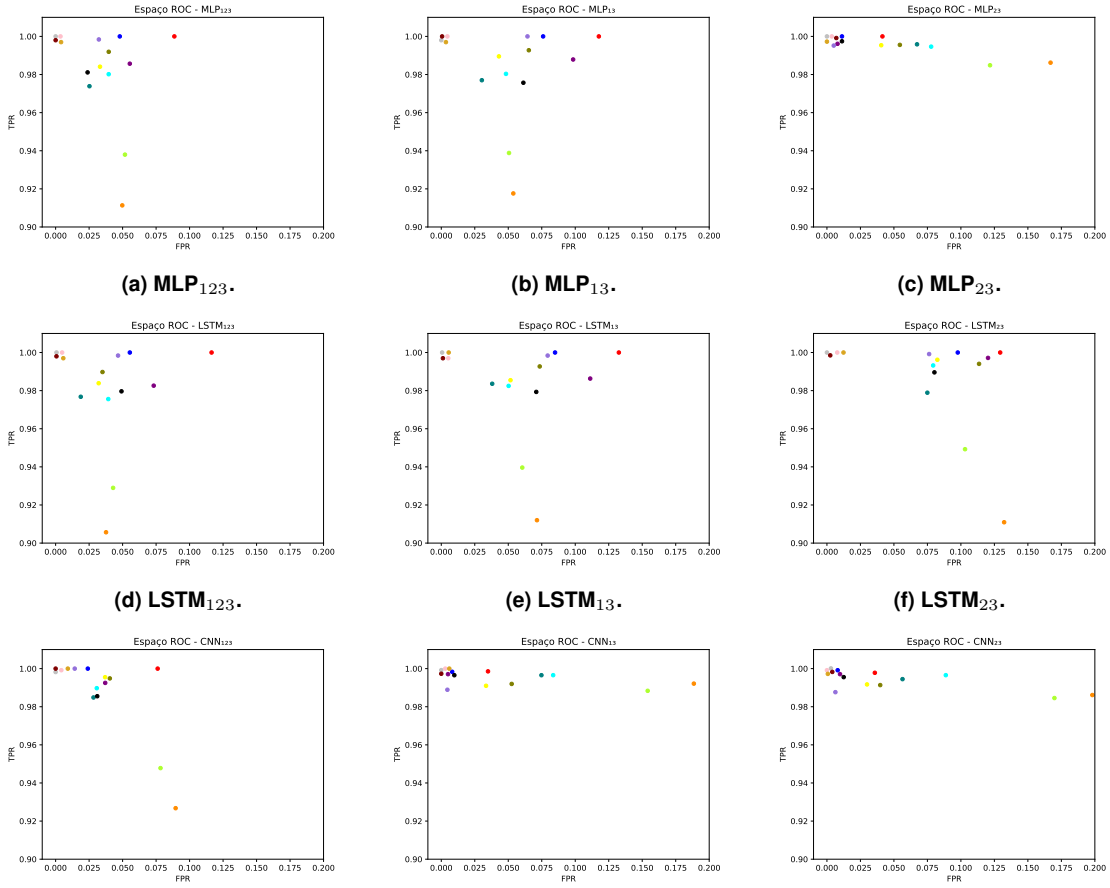
Constatação 2: Vetores de 3 (ack_ewma,snd_ewma,rtt_ratio), ou duas (ack_ewma,rtt_ratio), (snd_ewma,rtt_ratio) dimensões são suficientes para construção de classificadores de congestionamento eficientes.

7. Otimização de CC (RQ3)

7.1. Implantação dos Modelos no ns3

A implantação dos modelos no ns3 se desdobrou em duas fases: exportação das redes neurais do keras para C++ e a integração dos modelos ao AT. A exportação dos modelos para C++ foi realizado com auxílio da biblioteca keras2c [Conlin et al. 2021], cujas listagens produzidas para os modelos MLP_{23}, CNN_{123} e CNN_{13} foram aproveitadas para construção das API correspondentes. Cada uma das implementações passou por uma bateria de 1889 testes em que se procurava observar a diferença das suas respostas em relação àquelas fornecidas pelo keras. Em todos os testes, as respostas apresentaram diferença inferior a 10^{-4}.

Uma vez construídas as API, o AT foi configurado para receber como configuração qual dos modelos em estudo será ativado (figura 6). Então, para cada fluxo, O AT, a cada ACK, extrai da rede simulada os valores de ack_ewma, snd_ewma e rtt_ratio ((1)) e os repassa por meio da API para que o modelo em utilização classifique o nível de congestionamento da rede ((2)). A resposta obtida é então disponibilizada para a variante TCP, assistida pelo modelo em funcionamento((3)), que a utiliza para atualizar a $cwnd$ da próxima rodada de transmissão ((4)), conforme detalhado na seção a seguir.



(a) MLP$_{123}$. **(b) MLP$_{13}$.** **(c) MLP$_{23}$.**

(d) LSTM$_{123}$. **(e) LSTM$_{13}$.** **(f) LSTM$_{23}$.**

(g) CNN$_{123}$. **(h) CNN$_{13}$.** **(i) CNN$_{23}$.**

Figura 5. Espaço ROC produzidos pelos modelos sobre dados estranhos ao treinamento.

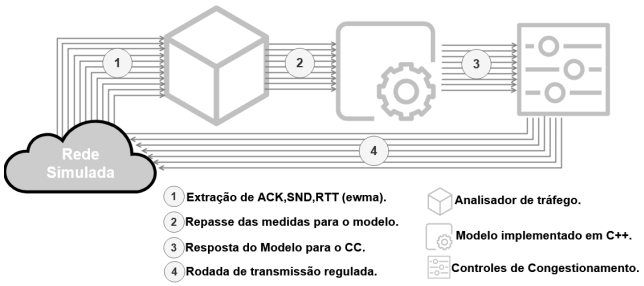


Figura 6. Integração dos Modelos ao AT

7.2. Construção das variantes TCP assistidas pelos modelos

Para utilização da resposta fornecida pelos modelos, a Classe TCPVegas, presente na versão 3.40 do ns3, sofreu uma refatoração. A classe passou a armazenar as saídas dos modelos para atualizar a $cwnd$, da seguinte forma:

$$cwnd = \begin{cases} cwnd + 3, & \text{se modelo classifica nível de congestionamento 1;} \\ \sqrt[4]{cwnd^3}, & \text{se modelo classifica nível de congestionamento 2 e } \sqrt[4]{cwnd^3} > 2; \\ 3, & \text{caso contrário.} \end{cases}$$

A equação acima foi elaborada com intuito de deixar o CC mais agressivo do que o

SlowStart, em situações de subutilização. Em contra partida, em situações de sobrecarga, a função produz maiores reduções do que a obtida no início do Fast Recovery, uma vez que $\sqrt[4]{cwnd^3} < \frac{x}{2}$ para $x > 16$.

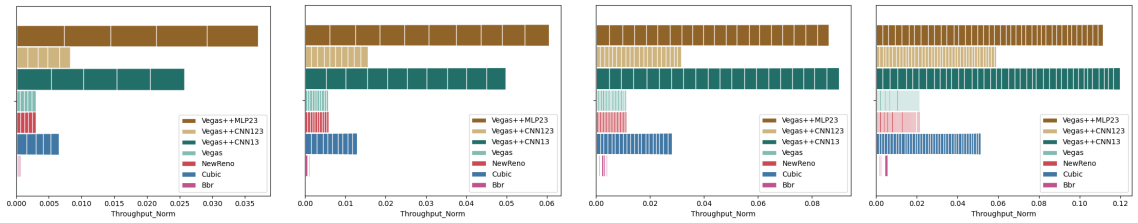
A mencionada refatoração atou pontualmente na atualização da janela de congestionamento, quando o Vegas busca eliminar a diferença entre a taxa de transmissão esperada e a observada [Brakmo et al. 1994], o que inclui desconsiderar a ssthresh, uma vez que o novo valor de $cwnd$ é baseado única e exclusivamente na situação da rede fornecida pelo modelo ativado. Todos os demais componentes da implementação Vegas permaneceram inalterados. Dessa forma, foram gerados três CC assistidos: Vegas++MLP$_{23}$, Vegas++CNN$_{123}$ e Vegas++CNN$_{13}$.

7.3. Experimentos com as Variantes TCP Assistidas

Para avaliação dos ganhos com os modelos construídos, foram realizadas simulações de 30 min, nos mesmos moldes das anteriores, acrescentando-se os CC assistidos. O objetivo é avaliar o quanto as variantes assistidas podem aumentar o Throughput Normalizado (Tput_Norm) Médio [1], em um ambientes hostis às variantes TCP clássicas. Para tanto, foi acrescentado uma taxa de erro de 10^{-5} no link entre os dois roteadores da topologia dumbell (figura 2). Os resultados desta atividade são apresentados na tabela[2] 4 e na figura 7:

Tabela 4. Throughput Normalizado Médios(TputNM) das Variantes TCP.

TCP	$\overline{\text{Tput\_Norm}}$ (5 Fluxos)	$\overline{\text{Tput\_Norm}}$ (10 Fluxos)	$\overline{\text{Tput\_Norm}}$ (20 Fluxos)	$\overline{\text{Tput\_Norm}}$ (40 Fluxos)
Vegas++MLP$_{23}$	0.037	0.061	0.086	0.111
Vegas++CNN$_{123}$	0.008	0.016	0.031	0.059
Vegas++CNN$_{13}$	0.026	0.050	0.090	0.120
Vegas	0.003	0.006	0.011	0.021
NewReno	0.003	0.006	0.011	0.021
Cubic	0.007	0.013	0.028	0.051
Bbr	0.001	0.001	0.004	0.007



(a) 05 Fluxos. **(b) 10 Fluxos.** **(c) 20 Fluxos.** **(d) 40 Fluxos.**

Figura 7. Desempenho das variantes TCP por total de fluxos na simulação.

> **Constatação 3:** Modelos classificadores do nível de congestionamento, extraídos de redes neurais, treinadas com vetores de baixa dimensão (2 ou 3), podem, efetivamente, otimizar a utilização da banda disponível.

[1] O Tput Máximo foi considerado $(1 - p)C$, em que C é a taxa entre os roteadores e p a taxa de erro.

[2] Todos os valores apresentados foram extraídos por meio do Flow Monitor do ns3 (`https://www.nsnam.org/docs/models/html/flow-monitor.html`)

Os resultados destacam as variantes Vegas++MLP$_{23}$ Vegas++CNN$_{13}$ como as que obtiveram o melhor desempenho. Os experimentos mostram que, quando considerada a diferença entre o throughput normalizado médio obtidos com estas variante e o do Cubic, a quantidade de dados transmitidos pode praticamente quintuplicar, como mostram os dados levantados para 05 e 10 fluxos; triplicar para 20 fluxos; e duplicar para 40 fluxos. As seções ao longo das barras, que destacam a contribuição de cada fluxo no Tput_Norm global, também revelam elevado grau de equidade das variantes assistidas.

Os valores mostram que os TCP assistidos não se limitam à fórmula $\frac{\sqrt{3}.MSS}{\sqrt{2p}RTT}$, para o throughput máximo de uma conexão TCP, obedecida de forma aproximada pelo TCP NewReno [Ng and Chan 2005]. Como o MSS da simulação é de 1460 B e o RTT total é 420 ms, essa fórmula fornece um valor de aproximadamente $0,13$ para o Tput_Norm máximo do NewReno, obedecida, também, por todas as demais variantes não assistidas durante os experimentos.

8. Conclusão

Ficou comprovado que é possível construir bons classificadores do nível de congestionamento de uma rede por meio de redes neurais profundas. Os experimentos, acusando acurácia acima de 90%, mostraram que a técnica de levantamento e transformação de dados apresentada, associando as medidas ack_ewma, snd_ewma e rtt_ratio com o nível de ocupação do buffer do roteador de borda, é bem eficiente no fornecimento de vetores de treinamento de baixa dimensão para as ARN apresentadas.

Outro aspecto extremamente revelante foi estabelecido por meio do estudo da capacidade de generalização dos modelos. Os modelos apresentados mostraram elevada tolerância à variação da quantidade e do CC dos fluxos. A precisão dos modelos sobre dados obtidos com a variação destes parâmetros permaneceu extremamente elevada, chegando a picos de mais de 91,5% de acurácia para os de melhor desempenho (MLP$_{23}$, CNN$_{123}$ e CNN$_{13}$).

Os ganhos com a utilização dos modelos construídos na pesquisa também foram altamente expressivos. Os esquemas de CC assistidos proporcionaram, no pior caso, um throughput duas vezes maior, chegando a ampliar em cinco vezes esta quantidade em cenários de poucas conexões simultâneas. Cumpre salientar ainda que os experimentos mostraram que esta elevação na troca de dados foi atingida sem sacrifício da equidade entre os fluxos Vegas++MLP$_{23}$, Vegas++CNN$_{123}$ e Vegas++CNN$_{13}$.

Entretanto, algumas questões práticas ainda precisam ser melhor avaliadas. O tempo de resposta dos modelos exportados para o ns3 ficou na ordem de 10^{-7}s, o que deve ser levado em conta em canais de baixíssimo *inter arrival time*, como aqueles da ordem de 1Gbps. Outro fator a considerar, é a condução dos experimentos da pesquisa em ambientes minimamente controlados, o que nem sempre é possível em redes reais. Tais considerações abrem caminho para aproveitamento dos resultados iniciais desta pesquisa em testes envolvendo redes reais.

Referências

Abbasloo, S., Yen, C.-Y., and Chao, H. J. (2020). Classic Meets Modern: a Pragmatic Learning-Based Congestion Control for the Internet. In *Proceedings of the Annual*

conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication, pages 632–647, Virtual Event USA. ACM.

Badarla, V. and Murthy, C. S. R. (2011). Learning-TCP: A stochastic approach for efficient update in TCP congestion window in ad hoc wireless networks. *Journal of Parallel and Distributed Computing*, 71(6):863–878.

Bai, L., Abe, H., and Lee, C. (2020). RNN-based Approach to TCP throughput prediction. In *2020 Eighth International Symposium on Computing and Networking Workshops (CANDARW)*, pages 391–395, Naha, Japan. IEEE.

Brakmo, L. S., O'Malley, S. W., and Peterson, L. L. (1994). TCP Vegas: New Techniques for Congestion Detection and Avoidance.

Conlin, R., Erickson, K., Abbate, J., and Kolemen, E. (2021). Keras2c: A library for converting Keras neural networks to real-time compatible C. *Engineering Applications of Artificial Intelligence*, 100:104182. 0000063.

Emara, S., Li, B., and Chen, Y. (2020). Eagle: Refining Congestion Control by Learning from the Experts. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pages 676–685, Toronto, ON, Canada. IEEE.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202. 0000061.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780. 0000060 Conference Name: Neural Computation.

ITU, G. C. R. (2023). Global Connectivity Report 2022 - `https://www.itu.int/itu-d/reports/statistics/2022/05/29/gcr-chapter-1` acesso em 13/12/23.

Jay, N., Rotman, N. H., Godfrey, P. B., Schapira, M., and Tamar, A. (2019). A Deep Reinforcement Learning Perspective on Internet Congestion Control.

Kazama, R., Abe, H., and Lee, C. (2022). Evaluating TCP throughput predictability from packet traces using recurrent neural network. In *2022 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. 0000057 ISSN: 2642-7389.

Li, W., Zhou, F., Chowdhury, K. R., and Meleis, W. (2019). QTCP: Adaptive Congestion Control with Reinforcement Learning. *IEEE Transactions on Network Science and Engineering*, 6(3):445–458.

Li, W., Zhou, F., Meleis, W., and Chowdhury, K. (2016). Learning-Based and Data-Driven TCP Design for Memory-Constrained IoT. In *2016 International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 199–205, Washington, DC, USA. IEEE.

Lippmann, R. (1987). An introduction to computing with neural nets. *IEEE ASSP Magazine*, 4(2):4–22. 0000062 Conference Name: IEEE ASSP Magazine.

Ng, S. W. and Chan, E. (2005). Equation-based TCP-friendly congestion control under lossy environment. *Journal of Systems Architecture*, 51(9):542–569.

ns3 (2023). A discrete-event network simulator for internet systems - https://www.nsnam.org - vistado em 13/12/05.

Ramana, B., Manoj, B., and Murthy, C. (2005). Learning-TCP: a novel learning automata based reliable transport protocol for ad hoc wireless networks. In *2nd International Conference on Broadband Networks, 2005.*, pages 521–530, Boston, MA. IEEE.

Sutton, R. S. and Barto, A. G. (2015). Reinforcement Learning: An Introduction.

Winstein, K. and Balakrishnan, H. (2013). TCP ex Machina: Computer-Generated Congestion Control.