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
Instituto Tecnológico de Aeronáutica (ITA)
São José dos Campos – SP – Brasil

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

Agradecimento ao Projeto FAPESP GRANT 2021/06872-6, pelo apoio oferecido ao longo da pesquisa.

Abstract. *The advancement and ubiquity of digital networks have fundamentally transformed numerous spheres of human activity. At the heart of this phenomenon, lies the Transmission Control Protocol (TCP) model, whose influence is particularly notable in the exponential growth of the Internet due to its ability to transmit flexibly to any device, through its advanced Congestion Control (CC). Seeking an even more efficient CC mechanism, this work proposes the construction of deep learning neural networks (MLP, LSTM, and CNN) for classifying the level of network congestion. The results attest to models capable of distinguishing, with over 90% accuracy, between moments of high and low degrees of congestion. With this, it becomes possible to differentiate between congestion and random losses, potentially increasing throughput by up to five times in environments with random losses when combined with CC algorithms.*

Resumo. *O avanço e a ubiquidade das redes digitais têm transformado fundamentalmente inúmeras esferas da atividade humana. No cerne desse fenômeno, encontra-se o modelo de protocolo de controle de transmissão (TCP), cuja influência é particularmente notável no crescimento exponencial da Internet, devido à sua capacidade de transmitir de maneira flexível para qualquer dispositivo, por meio do seu avançado Controle de Congestionamento (CC). Buscando um mecanismo de CC ainda mais eficiente, este trabalho propõe a construção de redes neurais de aprendizado profundo (MLP, LSTM e CNN) para classificação do nível de congestionamento. Os resultados atestam modelos capazes de distinguir, com mais de 90% de acerto, entre momentos de alto e baixo grau de congestionamento, e, com isso, realizar a diferenciação de perdas por congestionamento versus aleatórias, podendo elevar a vazão em até cinco vezes em ambientes de perdas aleatórias, quando combinado com algoritmos de CC.*

1. Introdução

As redes digitais, fundamentais no cotidiano da sociedade moderna, são amplamente governadas pela camada de transporte TCP. Essa predominância deriva da vasta gama de serviços disponíveis na internet, acessíveis através de uma diversidade de terminais com diferentes arquiteturas. A internet, que rapidamente se tornou acessível para cerca de 95% da população global [ITU 2023], opera sobre um conjunto de regras estabelecidas pelo protocolo TCP. Tais regras, embutidas nos sistemas operacionais, facilitam a troca

de dados entre aplicações, através de uma série de serviços estruturados em camadas, conhecidos coletivamente como a pilha de protocolos TCP/IP.

Um elemento crítico dessa pilha é a Camada de Transporte, encarregada do Controle de Congestionamento (CC). O CC é essencial às transmissões TCP, ajustando a quantidade de dados enviados a cada ciclo de transmissão. Uma parte crucial do CC é a estimativa do nível de congestionamento da rede, que, na maioria das implementações disponíveis, é realizada por meio de perdas já ocorridas ou variações no atraso durante a troca de dados.

Nesse contexto, as Redes de Aprendizado Profundo, como Multilayer Perceptron (MLP) [Lippmann 1987], Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber 1997] e Convolutional Neural Network (CNN) [Fukushima 1980], emergem como arquiteturas promissoras para estimar precisamente o nível de congestionamento, antes da saturação da infraestrutura.

1.1. Questões de pesquisa

Diante do exposto, este trabalho investiga o desempenho dessas arquiteturas de IA, diretamente empregadas no CC, buscando responder às seguintes questões de pesquisa (QP):

- **QP1: (Aprendizado de Congestionamento por Redes Neurais)** Investigar a capacidade de uma rede neural aprender a classificar o grau de congestionamento em uma conexão de rede, baseada na ocupação de um *buffer* de saída numa topologia *dumbell*, presente em grande parte das ligações da Internet.
  - **QP1.1: (Tolerância à Variação da Quantidade de Fluxos)** Avaliar se o modelo de aprendizado é capaz de manter sua precisão e eficácia mesmo quando há variações significativas no volume de tráfego (fluxo) na rede.
  - **QP1.2: (Aplicabilidade a Diversos Mecanismos de CC)** Determinar se o modelo aprendido pode ser efetivamente aplicado a fluxos que são controlados por diferentes tipos de mecanismos de CC.
- **QP2: (Identificação da Dimensão Ótima para Classificadores de Congestionamento)** Definir qual é a combinação ideal de medidas (tais como componentes e dimensões de dados) para desenvolver classificadores de congestionamento que sejam ao mesmo tempo eficientes e precisos.
- **QP3: (Ganhos com o Aprendizado em Cenários de Perdas Aleatórias)** Examinar se um modelo desenvolvido a partir de uma rede neural, treinada com vetores de baixa dimensão, pode levar a um melhor uso da banda disponível, especialmente em cenários caracterizados por perdas aleatórias de pacotes.

Com base nas questões de pesquisa delineadas, realizamos um estudo amplo por meio de simulações em nível de pacotes utilizando o ns-3 [ns3 2023]. A flexibilidade do ns-3 permite explorar variadas versões do TCP, modelos de aprendizado de máquina e quantidade de fluxos (chegando até 40). Os resultados das simulações revelaram desempenho eficaz das redes neurais na detecção de níveis de congestionamento.

Essa eficácia serviu ainda para a aplicação desses modelos em caso prático: a distinção entre perdas aleatórias de pacotes, um fenômeno comum em redes sem fio, incluindo conexões Wi-Fi e redes celulares 4G e 5G, mostrando a utilidade do mecanismo com melhoria significativa no desempenho das transmissões nessas condições desafiadoras. Para validar nossos resultados, recorremos ao uso de equação analítica específica para

CC em cenários de perdas, comprovando que o desempenho dos protocolos concorrentes é negativamente afetado na simulação, conforme a literatura.

2. Trabalhos Relacionados

A evolução da tecnologia de redes e a crescente complexidade dos sistemas de comunicação demandam avanços contínuos em protocolos como o TCP, especialmente no que tange ao CC. Assim, a utilização de técnicas de Otimização, Aprendizado de Máquina (AM) e Deep Learning (DL) surge como uma promissora fronteira de pesquisa no contexto do transporte TCP. Este artigo faz um levantamento de uma série de estudos relevantes nesse domínio, cada um contribuindo de maneira única para a evolução do CC e a aplicação de redes neurais em TCP.

Um dos primeiros esforços nesse sentido é representado pelo TCP Rational ou Remmy [Winstein and Balakrishnan 2013], cujo foco foi na construção automática de algoritmos de CC, por meio de um aprendizado de máquina feito por simulação e otimização de um jogo teórico entre agentes de uma rede, competindo por um gargalo e armazenando as melhores ações e ganhos. Como resultado do aprendizado final, uma tabela de ações é modelada para otimizar o desempenho em redes específicas, utilizada no treinamento.

Apesar de se destacar por sua abordagem inovadora, o Remmy apresenta limitações, principalmente pela sua incapacidade de adaptar-se a mudanças na rede após a intensa fase de treinamento. Tal fato se deve à manutenção de reações fixas diante das métricas ack_ewma, snd_ewma e rtt_ratio (detalhadas no decorrer do trabalho), o que limita sua aplicabilidade em cenários dinâmicos, algo comum a todos os mecanismos que se utilizam de funções fixas de atualização de *cwnd*.

Além do Remmy, outros estudos exploram os benefícios do *Reinforcement Learning* (RL) no desenvolvimento de CC mais adaptativos. Vários trabalhos empregam RL para criarem tabelas que correlacionem o estado da rede a ações específicas, como a atualização da *cwnd* [Sutton and Barto 2018]. Enquanto alguns desses estudos se baseiam em treinamento prévio para desenvolver CC mais flexíveis [Abbasloo et al. 2020], [Jay et al. 2019], [Li et al. 2016], outros adotam uma abordagem de mudança *online* das políticas de escolha de ações, tais como [Li et al. 2019], [Badarla and Murthy 2011] e [Ramana et al. 2005]. Essas pesquisas, no entanto, enfrentam desafios devido à natureza contínua e complexa do espaço de estados em redes de transmissão, podendo até colapsar em alguns casos [Emara et al. 2020].

As técnicas de *Deep Learning* também ganham destaque em publicações recentes, principalmente por sua capacidade de predição. Estudos modernos, como [Kazama et al. 2022] e [Bai et al. 2020] demonstram o potencial das LSTM. Na literatura, entretanto, não foram identificados trabalhos que incluam as redes MLP, nem as CNN. Além disso, muitas das abordagens operam com uma precisão abaixo de (80%), especialmente em cenários de baixa complexidade, i.e. com um fluxo somente e taxa de gargalo fixa.

Confrontando estas limitações, a presente pesquisa inova, comparando diversas Arquiteturas de Redes Neurais (ARN), como MLP, LSTM e CNN, na tarefa de classificar o nível de congestionamento da rede, ao longo de uma transmissão TCP/IP. Além de avaliar a eficácia, o estudo também investiga qual seria a dimensão ótima para os vetores de

entrada dos modelos oferecidos por essas arquiteturas. Tal dimensão impacta diretamente o tempo de resposta da implementação TCP dos sistemas operacionais nos quais esses modelos forem eventualmente embarcados.

O presente trabalho não se limita à comparação entre as arquiteturas, mas avança na aplicação desses modelos em cenários complexos, envolvendo até 40 fluxos e variadas versões do TCP. A abordagem permite avaliar a robustez e a eficiência dos modelos propostos em ambientes extremamente desafiadores, um passo crucial para a validação prática de qualquer nova tecnologia de CC.

Adicionalmente, em comparação com outros trabalhos, que apenas atestam o desempenho do modelo de aprendizado, o nosso busca também demonstrar os ganhos efetivos de um classificador, quando integrado a implementações de CC, de modo a otimizá-los. De forma pioneira, a performance dos CC assistidos pelos modelos extraídos das Redes Neurais são comparados com algumas das variantes TCP, mais tradicionais e comumente disponibilizadas nos Sistemas Operacionais.

3. Visão Geral da Metodologia de Redes Neurais para CC

Após a apresentação dos trabalhos relacionados, que fornecem um panorama das abordagens existentes no campo de CC com o uso de redes neurais, esta seção detalha a metodologia adotada em nossa pesquisa. Tal detalhamento é essencial para o entendimento de como serão abordadas as questões de pesquisa levantadas anteriormente e como os modelos propostos foram desenvolvidos e avaliados.

A metodologia adotada é ilustrada na Figura 1, que esquematiza o fluxo das etapas de nossa investigação. Começando pela Coleta e subsequente Preparação dos Dados, a cadeia prossegue com o Treinamento das Redes Neurais - Obtenção dos Modelos, a Avaliação e Seleção dos Modelos, e a Otimização de CC. Cada uma das etapas alimenta a seguinte, numa progressão lógica, até culminar na construção de um CC otimizado por IA.

Figura 1. Metologia empregada.

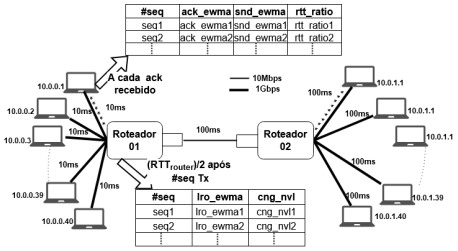


Figura 2. Coleta dos Dados

Para possibilitar a aquisição dos dados necessários ao treinamento, foi desenvolvido o Analisador de Tráfego (aqui denominado de AT) no simulador ns-3, ferramenta

amplamente validada para estudos de CC [ns3 2023]. O AT realiza diversas simulações de uma topologia do tipo *dumbbell* (Figura 2), uma configuração comum em pesquisas de rede, que possui gargalo central e vários enlaces de acesso independentes. Essas simulações geram dois arquivos .csv para cada fluxo TCP estabelecido nos enlaces independentes, transportando dados capazes de saturar o gargalo.

No primeiro arquivo .csv, uma série de parâmetros, inspirados no Remmy, são registrados a cada ACK recebido pelo transmissor. São eles:

- #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 intervalo entre as marcas de tempo presentes nos respectivos ACK.
- rtt_ratio: Calculado a partir da divisão dos RTT pelo RTT_{min}.

O segundo arquivo .csv é atualizado da seguinte maneira: sempre que um segmento é enviado em um 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. Durante o evento são registrados parâmetros adicionais, que constituem o alvo do aprendizado:

- #seq: Número de seq 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 de congestionamento da rede. Se $lro\_ewma <= 40\%$, $cng\_nvl = 1$ ou $cng\_nvl = 2$, caso $lro\_ewma >= 70\%$.

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. Visando um treinamento mais preciso da rede neural, o analisador foi configurado para registrar dados apenas em situações em que $lro\_ewma <= 40\%$ ou $lro\_ewma >= 70\%$. Esse design busca deixar o modelo mais robusto na classificação de situações extremas, de subutilização ou de sobrecarga.

Para a fase de treinamento, os arquivos, atualizados conforme descrito acima, vieram de uma simulação ns-3 de 15 minutos, em que, na topologia descrita, foram estabelecidas 20 conexões distribuídas P2P de TCP NewReno. As aplicações TCP estão nos terminais conectados ao Roteador 01 e os servidores TCP naqueles ligados ao Roteador 02. O terminal 10.0.0.2 do Roteador 01 se liga ao terminal 10.1.0.2 do Roteador 02. 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 01 possuem *delay* de 10ms. Em todos os demais, esse tempo é de 100ms. A taxa de transmissão é de 1Gbs em todos os canais, com exceção daquele entre os roteadores, o gargalo, que é de 10Mbps.

O início das conexões são espaçados de 10s em 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.

Após a fase de coleta, os dados gerados 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 em 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): 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 desprezados.
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. 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.

4. Treinamento das redes Neurais - Obtenção dos Modelos

4.1. Conceitos Gerais

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).

A comparação entre os modelos extraídos destas redes de aprendizado será realizada por meio de métricas amplamente adotadas neste tipo de atividade (Acurácia, Erro, Precisão, Recall e F1) e da análise ROC [1] *(Receiver Operating Characteristics)*. O espaço ROC é construído a partir de um plano cartesiano, com a abscissa e a ordena associadas, respectivamente, às taxas de falsos positivos, ou *False Positive Rate* (FPR), e de verdadeiros positivos, ou *True Positive Rate* (TPR), para um conjunto de dados oferecidos ao modelo.

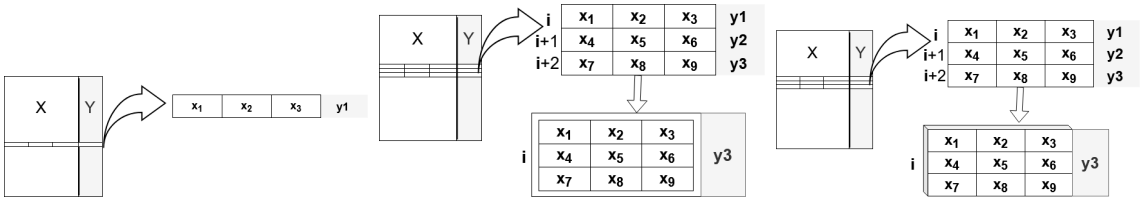
[1] `http://mlwiki.org/index.php/ROC_Analysis#ROC_Space`

Tabela 1. Redes neurais utilizadas

MLP	LSTM	CNN
-Entrada: Array de três dimensões (ver seção 4.2). -Camadas internas: 3 (20 RELU cada). -Saída: Sigmoid.	-Entrada: Matriz 3x3 (ver seção 4.2). -Camadas Internas: 2 (3 LSTM cada; *dropout* de 30%). -Saída: Sigmoid.	-Entrada: Vetores como imagem 3X3@1(ver seção 4.2). -Convolução:16 mapas de características 1x3 -Stride: [1,1] -Saída Convolução: RELU -Pooling: *maxpooling*, [1,2]. -Flattening: Alimenta 64 unidades RELU. Saída: Sigmoid.

4.2. 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 dessas 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). 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 9.461 entradas, 20% foram usadas para testes e o restante para treinamento e validação.

Apesar da baixa dimensão dos vetores, a utilização das LSTM e CNN visa a extração de eventuais padrões temporais ou espaciais na distribuição dos dados de entrada. Tais arquiteturas, apesar de mais complexas, são preparadas para identificar eventuais dependências temporais (LSTM) ou porções mais relevantes no espaço de células das "imagens" acima descritas (CNN). Com os experimentos realizados, é possível atestar se estas arquiteturas, com características próprias, se adéquam à construção de classificadores de congestionamento eficientes.

5. Avaliação dos Modelos e Posterior Seleção (QP1)

5.1. Análise dos Resultados - Dados de Teste

No que diz respeito aos resultados para os 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 canto superior esquerdo da área do gráfico. Na tabela, a arquitetura CNN_{123} tem melhor desempenhos em 4 das 5 métricas apresentadas, perdendo por muito pouco $(0,069)$ para a MLP_{23}, no quesito precisão. Os modelos MLP_{23}, juntamente com CNN_{13} e CNN_{23}, também despontam com as menores taxas de falsos positivos.

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}	$0,670$	$0,330$	$0,638$	$0,681$	$0,659$

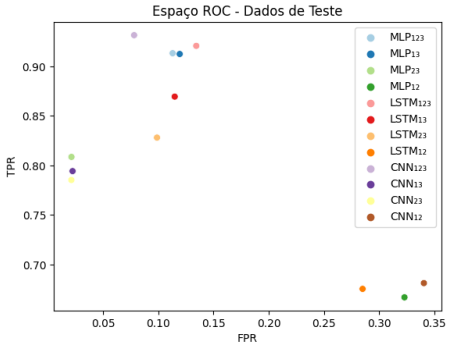


Figura 4. Espaço ROC - dados de teste

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 o par ack_ewma e $send\_ewma$ apresentaram os piores resultados, com acertos abaixo de 70%, ocupando o canto inferior direito no espaço ROC. Isso pode ser justificado pelo aspecto de que rtt_ratio é uma medida mais direta de ocupação de buffer e, em conjunto com as outras, melhora o desempenho. 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, com mais de 90% de precisão, tomando por base no treinamento o nível de ocupação de um roteador de saída da rede.

5.2. Capacidade de generalização dos modelos (QP1.1, QP1.2, QP 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 (e seus fluxos) ligadas ao gargalo (5, 10, 20, 40) e o algoritmo de CC (NewReno, CUBIC, BBR e Vegas). No final, foram consideradas 15 combinações, 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 descrita anteriormente, um conjunto X de 4000 entradas. Dessas 4000 entradas, 70% (2.800) foram selecionadas aleatoriamente, transformadas e classificadas por cada uns dos 09 modelos sobreviventes (Seção 5.1). Os resultados, usando valores médios das métricas, são apresentados na Tabela 3:

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

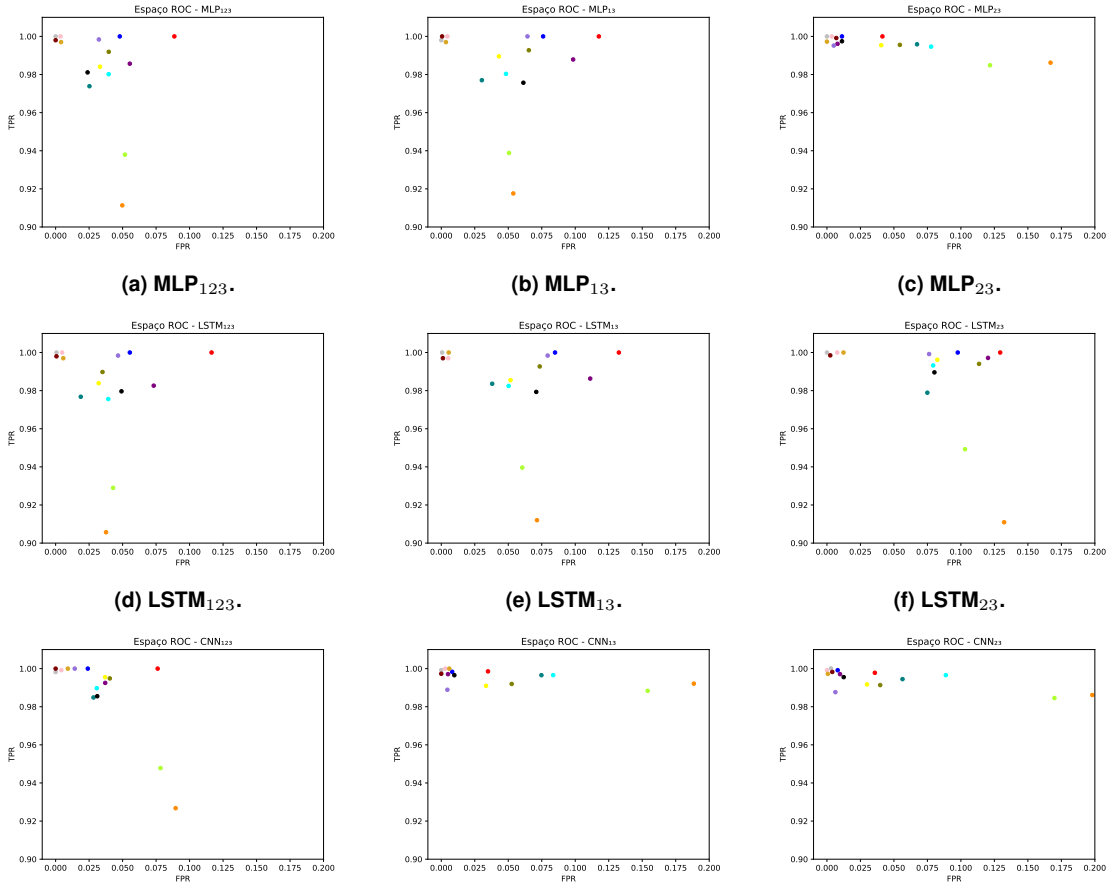
Modelo	Acurácia	Erro	Precisão	Recall	F1
MLP_{123}	$0,913$	$0,024$	$0,904$	$0,922$	$0,913$
MLP_{13}	$0,906$	$0,031$	$0,889$	$0,923$	$0,905$
MLP_{23}	$0,917$	$0,020$	$0,904$	$0,934$	$0,918$
$LSTM_{123}$	$0,910$	$0,027$	$0,899$	$0,920$	$0,909$
$LSTM_{13}$	$0,902$	$0,035$	$0,881$	$0,922$	$0,901$
$LSTM_{23}$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

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. Percebe-se que, novamente, os modelos MLP_{23}, CNN_{123} e CNN_{13} sobressaem. Note que a componente 3, rtt_ratio está sempre presente nas melhores respostas.

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 (cada cor é uma variação de CC e número de fluxos). A densidade de pontos em torno da posição (0,1) e da reta TPR=1 reafirmam os modelos MLP_{23}(Fig.5c), CNN_{123}(Fig. 5g) e CNN_{13}(Fig. 5h) como os de melhor desempenho. Merece destaque o modelo CNN_{23} (Fig. 5i), com boa distribuição no espaço ROC, ficando, entretanto, aquém dos três mencionados anteriormente.

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



(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 que não participaram do treinamento.

6. Otimização de CC (RQ3)

6.1. Implantação dos Modelos diretamente nos CC usando o ns-3

A implantação dos modelos no ns-3 se desdobrou em duas fases: exportação das redes neurais do keras para C++ de alto desempenho e a integração dos modelos ao simulador 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 procurando 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 seria 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, por meio da API, os repassa 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.

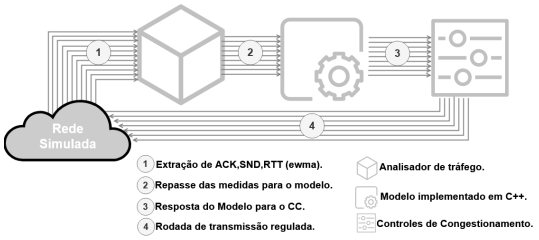


Figura 6. Integração dos Modelos ao AT

6.2. Construção das variantes TCP assistidas pelos modelos

Tendo em vista a grande influência do RTT na eficiência dos modelos, a Classe TCP-Vegas, presente na versão 3.40 do ns-3, foi refatorada, passando a utilizar 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 *Congestion Avoidance*, em situações de subutilização. Em contrapartida, caso haja sobrecarga, a função produz maiores reduções do que a obtida no início do Fast Recovery, uma vez que $\sqrt[4]{x^3} < \frac{x}{2}$ para $x > 16$.

A adaptação descrita atou pontualmente na atualização da janela de congestionamento, quando o Vegas busca eliminar a diferença entre a taxa de transmissão esperada e a observada, baseado no RTT mínimo [Brakmo et al. 1994]. 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$_{13}$, intercambiáveis no AT.

6.3. 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 Médio (TputMN) [2], em um ambiente hostil (com perdas aleatórias) às variantes TCP clássicas. Para tanto, foi acrescentado uma taxa de erro de bits de 10^{-5} no gargalo da topologia *dumbbell* (Figura 2). Os resultados obtidos são apresentados na Tabela 4 e na Figura 7. [3]:

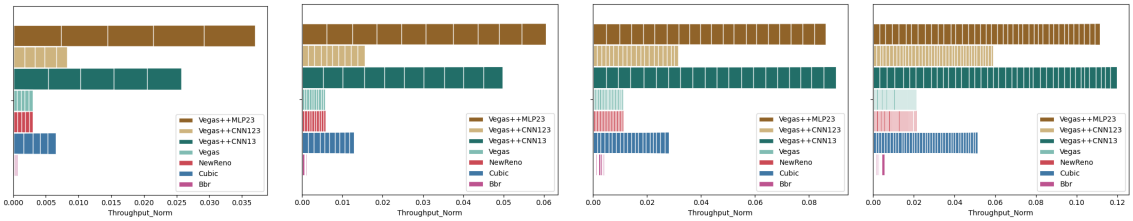
> **Constatação 3:** Modelos classificadores do nível de congestionamento, extraídos de redes neurais, treinadas com vetores de baixa dimensão (2 ou 3), podem, efetivamente, otimizar a utilização da banda disponível.

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

[3] Todos os valores apresentados foram extraídos por meio do *Flow Monitor* do ns-3 - `https://www.nsnam.org/docs/models/html/flow-monitor.html`

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

TCP Variant	Throughput Normalizado Médio (Tput_NM)			
	5 Fluxos	10 Fluxos	20 Fluxos	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.

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 Tput_NM 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 ainda que os TCP assistidos se limitaram à fórmula $\frac{\sqrt{3}.MSS}{\sqrt{2}pRTT}$ [Ng and Chan 2005], que, no caso, fornece um valor de $0,13$ para o Tput_Norm máximo para uma conexão TCPNewReno.

7. Conclusão

A acurácia acima de 90%, obtida nos dados de teste, superando em mais de 10% os resultados apresentados por pesquisas mais recentes, mostraram que a técnica de levantamento e transformação de dados apresentada é bem eficiente no fornecimento de vetores de treinamento de baixa dimensão para as ARN apresentadas. Além disso, a recorrência do rtt_ratio nestas entradas de baixa dimensão estimulam novas pesquisas com foco em técnicas mais simples, tais como Árvores de Decisão.

Quanto à capacidade de generalização, os modelos mostraram elevada tolerância à variação da quantidade de fluxos e dos algoritmos CC por debaixo. A precisão dos modelos sobre dados obtidos com a variação desses 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$_{13}$).

Com CC assistidos, os modelos desenvolvidos demonstraram ganhos significativos em relação àqueles que não se utilizam de Redes Neurais. Pior caso de vazão de

duas vezes maior, e melhor de cinco vezes, em cenários de poucas conexões simultâneas. Tais patamares foram atingidos sem sacrifício da equidade entre os fluxos das variantes Vegas++MLP$_{23}$, Vegas++CNN$_{123}$ e Vegas++CNN$_{13}$. Por terem apresentado bom desempenho, a função de atualização da $cwnd$ dessas variantes pode ser considerada, também, uma importante contribuição desta pesquisa.

No entanto, ainda se faz necessária uma avaliação mais aprofundada de algumas questões práticas para emprego das presentes propostas na vida real. Os tempos de inferência dos modelos exportados para o ns-3 ficou na ordem de 10^{-7}s ou fração de microssegundos, o que deve ser considerado em canais de baixíssimo *inter arrival time*, como os de 1Gbps. Cabe ainda a investigação de questões relacionadas à quantidade de processamento e memória necessários ao funcionamento dos modelos em clientes reais.

Apesar da diversidade de cenários e testes trazidos, na prática, o CC é afetado por fatores tais como a diversidade de tecnologia da camada física, rotas, topologia, latência, *jitter*, etc, que podem levar os modelos apresentados a um quadro de *Concept Drift* [Lu et al. 2018], abrindo espaço para trabalhos que explorem melhor a influência de outras variáveis no desempenho dos modelos construídos de acordo com as técnicas ora apresentadas.

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. In *Proceedings of the Conference on Communications Architectures, Protocols and Applications*, SIGCOMM '94, page 24–35, New York, NY, USA. Association for Computing Machinery.

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., Godfrey, B., Schapira, M., and Tamar, A. (2019). A deep reinforcement learning perspective on internet congestion control. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3050–3059. PMLR.

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.

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., and Zhang, G. (2018). Learning under Concept Drift: A Review. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1. 0000065 arXiv:2004.05785 [cs, stat].

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. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

Winstein, K. and Balakrishnan, H. (2013). Tcp ex machina: Computer-generated congestion control. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, page 123–134, New York, NY, USA. Association for Computing Machinery.