UNIVERSIDADE FEDERAL DO PAMPA

Henrique Fan da Silva

Previsão do Tamanho de Enxames BitTorrent com Redes Neurais

Alegrete
2022

Henrique Fan da Silva

Previsão do Tamanho de Enxames BitTorrent com Redes Neurais

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Rodrigo Brandão Mansilha

Alegrete
2022

HENRIQUE FAN DA SILVA

PREVISÃO DO TAMANHO DE ENXAMES BITTORRENT COM REDES NEURAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação

Trabalho de Conclusão de Curso defendido e aprovada em: 02, agosto e 2022.

Banca examinadora:

_____

Prof. Dr. Rodrigo Brandão Mansilha

Orientador

Universidade Federal do Pampa

_____

Prof. Dr. Diego Kreutz

Universidade Federal do Pampa

_____

Prof. MSc. Celso Nobre da Fonseca

Universidade Federal do Pampa

Assinado eletronicamente por **RODRIGO BRANDAO MANSILHA**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 02/08/2022, às 19:26, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **CELSO NOBRE DA FONSECA**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 02/08/2022, às 20:33, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **DIEGO LUIS KREUTZ**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 02/08/2022, às 21:07, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0883828** e o código CRC **74B33B32**.

RESUMO

Em redes par-a-par (P2P) é importante compreender o comportamento de das entidades que fazem parte da rede ou sistema. Somente compreendendo como o sistema realmente funciona por completo é possível identificar limitações e oportunidades de melhoria. Monitorar enxames Bitorrent é desafiador pois exige um sistema distribuído com uma capacidade que deve ser proporcional ao tamanho do enxame. Propomos tentar entender o a dinâmica dos enxames ao longo do tempo, a partir dos dados do próprio monitoramento. Para isso, avaliamos dois métodos de predição a Rede Neural Convolucional de Grafo Temporal (T-GCN) e o Modelo de Média Móvel Integrado Autorregressivo (ARIMA), comparando qual é a melhor escolha para prever o tamanho do enxame ao longo do tempo. Constatamos que a rede neural pode ser uma alternativa mais rápida de executar do que o ARIMA, embora seu resultado seja um pouco pior.

Palavras-chave: Aprendizado de Máquina. ARIMA. BitTorrent. Enxames. GCN-LSTM. Monitoramento. Previsão. T-GCN.

ABSTRACT

In peer-to-peer (P2P) networks it is important to understand the behavior of the entities that are part of the network or system. Only by fully understanding how the system really works is it possible to identify limitations and opportunities for improvement. Monitoring Bitorrent swarms is challenging as it requires a distributed system with a capacity that must be proportional to the size of the swarm. We propose to try to understand the swarm dynamics over time, based on the monitoring data itself. For this, we evaluated two prediction methods, the Temporal Graph Convolutional Neural Network (T-GCN) and the Autoregressive Integrated Moving Average Model (ARIMA), comparing which is the best choice to predict the size of the swarm over time. We found that the neural network can be a faster alternative to execute than ARIMA, although its result is a little worse.

Key-words: Machine Learning. ARIMA. BitTorrent. Swarms. GCN-LSTM. Monitoring. Prediction. T-GCN

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AR Auto Regressive

ARIMA Autoregressive Integrated Moving Average Model

DiGCN Digraph Inception Convolutional Networks

GCN Graph Convolutional Network

GCN-LSTM combinação do **GCN** com o **LSTM**

GPU Graphics Processing Unit

GRU Gated Recurrent Unit

IP Internet Protocol

LSTM Long short-term memory

MA Moving Average

P2P Peer-to-peer

RNA Rede Neural Artificial

SVR Support Vector Regression model

T-GCN Temporal Graph Convolutional Network

SUMÁRIO

1 INTRODUÇÃO

Em redes Peer-to-peer (P2P) é importante compreender o comportamento das entidades que fazem parte da rede ou sistema (STEEN; TANENBAUM, 2017). Somente compreendendo como o sistema realmente funciona por completo é possível identificar limitações e oportunidades de melhoria em redes de geração atual e futuras (NAIK; KESHAVAMURTHY, 2020). Um relevante aspecto que pode ser melhor estudado e compreendido são sessões *online* de usuários (CORDEIRO et al., 2021), para, por exemplo, acompanhar a disseminação de conteúdo ilegal na Internet (SCHMIDT; BARCELLOS; GASPARY, 2011). Outro exemplo é a quantidade de pares presentes no sistema (TAN et al., 2019),

Uma das estratégia de monitoração de redes P2P geralmente empregada é a captura de amostras periódica das sessões dos usuários, utilizando monitores, em intervalos de tempo regulares, e usá-las para sintetizar traços (HOSSFELD et al., 2011). Essa abordagem de monitoramento permite uma certa autonomia, pois ela não requer acesso privilegiado a nenhuma entidade do sistema (MANSILHA et al., 2011). Em alguns casos é possível obter traços através das informações do próprio banco de dados do sistema (GUO et al., 2007), mas isso requer acesso privilegiado ao sistema. Outra estratégia que pode ser utilizada é a instalação de *plug-ins* nos dispositivos dos usuários, que fornecem informações de como eles utilizam *web* (ALEXA. . . , 2021).

Uma das principais aplicações de redes P2P é o compartilhamento de arquivos. BitTorrent é um exemplo de rede P2P de compartilhamento popular na Internet, sendo responsável por 3,35% da largura de banda mundial (APPLICATION. . . , 2021). Um componente importante em redes BitTorrent são os **rastreadores** (*trackers*) (LEHMANN et al., 2011). Eles atuam como ponto de encontro entre os pares de um **enxame** (*swarm*), mantendo uma **lista de Internet Protocol (IP) dos pares** (*peer list*), o que auxilia pares a encontrar outros pares no mesmo enxame. Os rastreadores também podem ser consultados para coletar informações sobre eles e os pares conectados à ele e assim monitorar a evolução da rede.

Neste contexto, identificou-se alguns trabalhos relacionados recentes (LAREIDA; HOSSFELD; STILLER, 2017; LAREIDA; STILLER, 2018; MUSA, 2020). O primeiro trabalho procurou mensurar a quantidade de pares compartilhando um determinado conteúdo. Os autores realizaram estimativa de tamanho de enxames empregando uma adaptação do Problema do Coletor de Cupom, formulado como Problema do Coletor de Pares BitTorrent (LAREIDA; HOSSFELD; STILLER, 2017). Os resultados mostram que esta estimativa funciona bem para rastreadores clássicos e que *churn* (*i.e.*, entrada ou saída em massa de usuários em um período relativamente curto de tempo) influencia frequentemente e significativamente nas medições, o que reforça a importância de previsões adequadas. Alguns outros trabalhos de monitoramento de redes BitTorrent focam no conteúdo compartilhado para monitorar a pirataria de vídeos, identificar principais atores e mu-

danças comportamentais ao longo do tempo, e assim compreender a eficácia das medidas antipirataria (LAREIDA; STILLER, 2018). Outro trabalho aborda as ameaças do compartilhamento de *malwares* e outros programas maliciosos em redes BitTorrent usando a ferramenta Wireshark para monitorar e analisar pacotes da camada de transporte de redes P2P (MUSA, 2020).

A realização de monitoramento do Universo BitTorrent é desafiador pois exige um sistema distribuído com uma capacidade que deve ser proporcional à demanda exigida pelos objetivos de monitoramento estabelecidos (MANSILHA et al., 2011). Por exemplo, quanto maior o número de pares a serem monitorados, mais recursos são necessários. Além disso, devem ser previstos recursos adicionais para provimento de tolerância a falhas, cuja probabilidade de ocorrência aumenta conforme a duração do monitoramento (*e.g.* escala de semanas ou meses). Como o universo BitTorrent é dinâmico e suscetível a fenômenos de *churn*, *prever a evolução da quantidade de pares torna-se um problema chave para provisionamento adequado de recursos de monitoramento.*

1.1 Objetivos

O objetivo geral deste trabalho é realizar a predição do tamanho dos enxames no futuro com base em resultados obtidos previamente para provisionamento de monitoramento de sistemas P2P. Investigamos técnicas do estado da arte de Aprendizado Profundo (*Deep Learning*), que é um ramo da Aprendizado de Máquina (*Machine Learning*). Este trabalho pode ser considerado um esforço para qualificar monitoramento de redes P2P em tempo de execução, que é complementar a esforços recentes voltados para qualificação de *datasets* de maneira posterior a realização do monitoramento (CORDEIRO et al., 2014; CORDEIRO et al., 2021; PAIM et al., 2021; PAIM et al., 2022).

Inicialmente empregamos o modelo de rede convolucional de grafo temporal Temporal Graph Convolutional Network (T-GCN) (ZHAO et al., 2019), anteriormente usado para previsão de tráfego de redes viárias urbanas. Para efeitos de comparação, também empregamos um método clássico de predição de séries temporais Autoregressive Integrated Moving Average Model (ARIMA), que é capaz de entender os dados e prever pontos futuros na série. Especificamente, comparamos os resultados da Rede Neural Artificial (RNA) com o modelo ARIMA em termos de qualidade do resultado (*i.e.*, eficácia) e tempo de processamento (*i.e.*, eficiência) para ajudar em futuras escolhas de método de predição do tamanho de enxames. Nesse processo avaliativo, usamos o conjunto de dados capturados do trabalho de (CORDEIRO et al., 2021), que oferece um grau de confiança estatística sobre o *ground truth*.

Na avaliação usamos o conjunto de dados capturados do trabalho de (CORDEIRO et al., 2021), que oferece um grau de confiança estatística sobre o *ground truth*. Com base nesse conjunto de dados, especificamos um conjunto de experimentos para comparar sistematicamente os dois métodos de predição. Disponibilizamos o protótipo de todos

os *scripts* implementados e os conjuntos de dados do monitoramento para permitir a reprodutibilidade dos experimentos da pesquisa.

1.2 Organização deste Trabalho

O restante desta monografia está organizado como segue. O Capítulo 2 revisa o referencial teórico da pesquisa. O Capítulo 3 apresenta a definição, modelagem do problema, e a solução proposta para predizer o tamanho de enxames em redes BitTorrent usando T-GCN. O Capítulo 4 demostra a metodologia e os resultados da solução proposta. O Capítulo 5 conclui este trabalho com considerações finais e direções de trabalhos futuros.

2 MATERIAIS E MÉTODOS

Neste capítulo revisamos o referencial teórico da pesquisa. A Seção 2.1 exemplifica os componentes e ralações do universo BitTorrent. A Seção 2.2 demostra o funcionamento do monitoramento. E a Seção 2.3 apresenta o modelo T-GCN. A Seção 2.4 finaliza o capítulo trazendo o funcionamento do método ARIMA.

2.1 Universo BitTorrent

Redes BitTorrent são compostas por enxames, pares, rastreadores e conteúdo (LEHMANN et al., 2011). Um par é um usuário que participa de um ou mais enxames de acordo com o conteúdo que deseja compartilhar ou receber. Rastreadores atuam como ponto de encontro entre pares: um par contata um rastreador e recebe uma lista de IPs de outros pares, *peer list*, que o permite participar do enxame para compartilhar um conteúdo.

Para compartilhar um novo conteúdo, um par deve gerar um *torrent* e torná-lo público. Esse conteúdo é organizado em peças. As informações dos metadados, contidos no arquivo, são usadas pelos usuário para participarem de um enxame. Esse arquivo contém informações como o nome e tamanho dos arquivos, *hashes* para os dados, e um ou mais endereços IP de rastreadores (KONRATH et al., 2007).

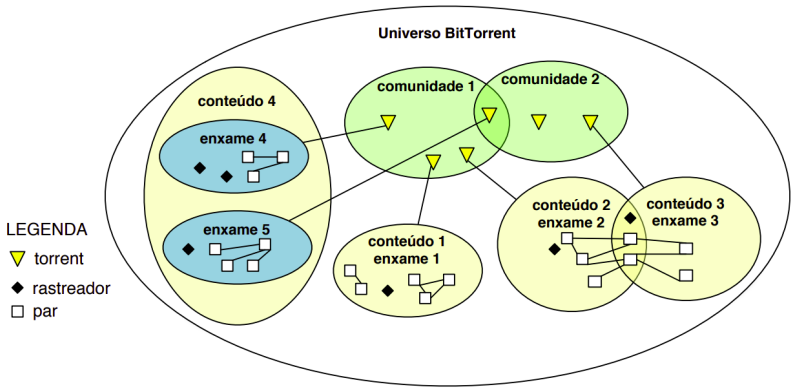
Geralmente as comunidades *torrent* promovem o compartilhamento dos conteúdos. Além disso, algumas comunidades disponibilizam informações de rastreadores e quantidade pares. Existem comunidade públicas, que são facilmente encontradas na Internet, e privadas de acesso restrito a membros cadastrados.

A Figura 1 exemplifica o universo BitTorrent, apresentando os possíveis cenários que pode ser encontrados na rede. O enxame 1, que compartilha o conteúdo 1, apresenta o caso em que um determinado conteúdo é compartilhado por apenas um enxame. Enxames distintos podem reunir pares em comum, por exemplo, enxames 2 e 3. Também existe o caso em que dois ou mais enxames independentes compartilham o mesmo conteúdo, enxames 4 e 5.

2.2 Sistema de Monitoramento do Universo BitTorrent

Existem diversas formas de monitoramento do universo BitTorrent (MANSILHA et al., 2011) com diferentes compromissos entre custos e objetivos de monitoramento. Sistemas de monitoramento são projetados para alcançar quatro objetivos principais: cobertura, detalhamento, frequência e duração. A cobertura refere-se a quantidade de elementos a serem observados e se representam exatamente a população a ser investigada. Detalhamento corresponde às variáveis a serem coletadas, por exemplo, a quantidade de pares. A frequência é o inverso do período em que as amostras são coletadas. Monitoramentos com frequência relativamente alta nos permite observar fenômenos de curta

Figura 1 – Exemplo de componentes e relações no universo de redes BitTorrent.

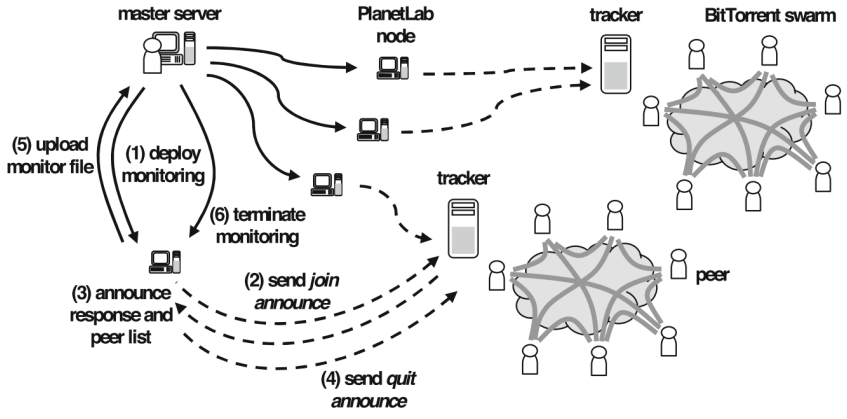


Fonte – (MANSILHA et al., 2011)

duração. Quanto maior a duração do monitoramento, maior são as chances de observarmos fenômenos recorrentes.

Especificamente, consideramos o sistema de monitoramento TorrentU (MANSILHA et al., 2011). Para coletar as informações dos rastreadores é instanciado um conjunto de nós agentes de monitoramento distribuídos, todos conectados a um nó gerente de monitoramento (*master server* - servidor mestre) que contactam um subconjunto dos rastreadores da rede. A Figura 2 fornece uma visão geral do processo de coleta.

Figura 2 – Visão geral do processo de coleta.



Fonte – (CORDEIRO et al., 2021)

O monitoramento ocorre em instantes de tempo, determinados pelo mestre. Começando com o *upload* dos arquivos contendo os destinos de monitoramento, lista de enxames (*torrent list*), do servidor mestre para os coletores (seta 1). Além disto o mestre também envia o endereço do depósito de *log* e a periodicidade dos instantes de tempo do

monitoramento.

Primeiro, o coletor envia um anúncio para a juntar o enxame (seta 2). Em reposta, o rastreador envia a *peer list* do enxame (seta 3). Segundo, o coletor deixa o enxame (seta 4), para que seu IP não seja enviado para outros pares, interferindo no enxame. E também evitará a sua inclusão na lista negra, uma vez que o colecionador não será considerado inscrito em vários rastreadores. O Processo representado pelas setas 2-4 acontece em todos os enxames. Depois de obter as *peer lists* dos rastreadores monitorados, os coletores as carregam para o depósito de *log* (seta 5) para evitar perda de dados. Por fim, o servidor mestre envia um sinal para cada coletor encerrar o processo de colata (seta 6).

No BitTorrent, o rastreador oferece apenas uma fração da sua *peer list*. Para que essa limitação técnica não impeça o monitoramento de coletar todos os usuário de um enxame, devemos solicitar uma certa quantidade de listas de pares, maximizando a probabilidade de que todos os pares apareçam pelo menos uma vez. Utilizando a metodologia proposta por (HOSSFELD et al., 2011), baseada numa variação do problema do coletor de cupom com amostras de tamanho $k > 1$ (KOBZA; JACOBSON; VAUGHAN, 2007). Com a Equação 2.1 podemos estimar um limite mínimo de respostas do rastreador necessárias para obter a população de um determinado enxame ($E[X]$).

$$E[X] \approx \frac{E[N] \cdot h_N}{L_{max} \cdot |B|} \tag{2.1}$$

, onde:

- $E[N]$ = tamanho esperado da população do enxame;
- h_N = n-ésimo número harmônico;
- L_{max} = tamanho máximo de uma lista de pares;
- B = conjunto de rastreadores consultados ($|B|$ sua cardinalidade).

Considerando uma estimativa inicial de 6.000 usuários para a população do enxame e 5 rastreadores a serem monitorados. Temos $L_{max} = 200$ e $|B| = 5$, ou seja, 1.000 usuários obtidos a cada solicitação, então teremos $E[X] \approx 55$ respostas para cobrir todo o enxame.

2.3 Rede Neural T-GCN

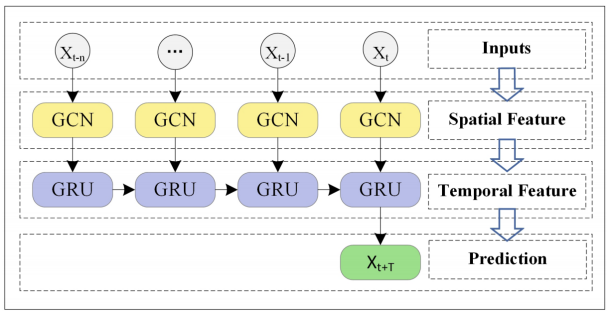
O modelo T-GCN proposto por (ZHAO et al., 2019) combina a rede convolucional de grafos Graph Convolutional Network (GCN) (LECUN et al., 1998) e a Unidade de recorrente fechada Gated Recurrent Unit (GRU) (CHO et al., 2014). O GCN é usado para aprender estruturas topológicas complexas para capturar a dependência espacial. Esse modelo já vem sendo usado na classificação de documentos (DEFFERRARD; BRESSON; VANDERGHEYNST, 2016), aprendizado não supervisionado (KIPF; WELLING, 2016) e classificação de imagens (BRUNA et al., 2013). Já a GRU é usada para aprender

mudanças dinâmicas de dados para modelar a dependência temporal, tanto de curto como de longo prazo (CHO et al., 2014).

No trabalho de (ZHAO et al., 2019) o modelo T-GCN foi usado para fazer a predição do volume de tráfico de redes viárias urbanas. Onde para descrever a estrutura topológica da rede viária foi usado um grafo $G = (V, E)$, V é o conjunto dos nós rodoviários $V = \{v1, v2, ..., vN\}$, N é o tamanho do conjunto de nós e E é o conjunto de arestas. Para representar as conexões entre as estradas foi usada uma matriz de adjacência A, $A \in R^{N \times N}$. Uma outra matriz $X \in R^{N \times P}$ foi usada para descrever as características (*features*), P representa o número de características de atributo do nó, ou seja, o comprimento da série histórica, e $X_t \in R^{N \times i}$ é usado para representar a velocidade em cada estrada no tempo i. Como recurso de atributo dos nós foi usado a velocidade do tráfego, mas também poderia ser o fluxo do tráfego ou a densidade de tráfego.

A Figura 3 exemplifica o funcionamento do modelo T-GCN, onde primeiro é usado os dados históricos de n série temporal como entrada para as unidades GCN, capturando a estrutura topológica da rede viária urbana. Em seguida as séries temporais obtidas com características espaciais, as saídas das unidade GCN, são inseridas nas unidades GRU e a mudança dinâmica é obtida pela transmissão de informações entre as unidades, para capturando as características temporais.

Figura 3 – Arquitetura T-GCN.



Fonte – (ZHAO et al., 2019)

Percebe-se que a entrada para o modelo é a matriz de *features* X. O problema de previsão espaço-temporal é o mapeamento na função f na topologia de G e na matriz de características X. Onde n é o comprimento da série histórica e T é o comprimento da série temporal que precisa ser prevista, como mostra a Equação 2.2:

$$[X_{t+1}, \cdots, X_{t+T}] = f(G; (X_{t-n}, \cdots, X_{t-1}, X_t)) \tag{2.2}$$

A Biblioteca StellarGraph para aprendizado de máquina em grafos e redes (DATA61, 2018), tem a sua própria versão do T-GCN, chamada combinação do **GCN** com o **LSTM** (GCN-LSTM), inspirada no modelo proposto por (ZHAO et al., 2019). Por ser man-

tida em constante atualização, ser fácil de usar, modular e extensível, decidimos usar o GCN-LSTM. O que difere os dois modelos é os módulos da camada oculta, onde o T-GCN usa a unidade GRU enquanto o GCN-LSTM usa a unidade Long short-term memory (LSTM) (HOCHREITER; SCHMIDHUBER, 1997). O modelo LSTM e o modelo GRU são variantes da rede neural recorrente, seus princípios básicos são aproximadamente os mesmos, sendo igualmente eficazes para várias tarefas (CHUNG et al., 2014). Na prática, não existem diferenças notáveis entre os dois tipos de modelo.

2.4 ARIMA

O modelo de média móvel integrado autorregressivo ARIMA (AHMED; COOK, 1979) é um dos modelos de série temporal mais populares e amplamente utilizados em diversas aplicações. Assim como o T-GCN, também já foi usado para prever o volume de tráfego em vias arteriais urbanas (HAMED; AL-MASAEID; SAID, 1995). O ARIMA é capaz de determinar séries temporais com base em seus próprios valores passados, ou seja, seus próprios atrasos e os erros de previsão defasados, para que a equação possa ser usada para prever valores futuros (BABAI et al., 2013).

Os modelos Auto Regressive (AR) e Moving Average (MA) são baseados na suposição de que a série temporal seja gerada através de um sistema linear, e que possuem um termo de erro aleatório não correlacionado, com média zero e variância constante, ou seja, um ruído branco (GUJARATI; PORTER, 2011).

No modelo AR, p refere-se ao número de defasamento a serem usados como preditores, ou seja, a ordem do modelo expressa o número de observações atrasadas (*lags*) das séries temporais analisadas que estão envolvidas na equação. Dada uma série de dados Y, descrita por seus valores históricos $Y_{t-1}, Y_{t-2}, \cdots$, e pelo ruído branco ϵ_t, deste modo Y_t é uma função dos *lags* de Y_t:

$$Y_t = C + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \cdots + \beta_p Y_{t-p} + \epsilon_t \tag{2.3}$$

onde, p é a ordem do modelo auto regressivo, o valor de Y no período t, depende do seu valor no p períodos anteriores. Sendo assim, AR (p) é um processo auto-regressivo de p-ésima ordem (GUJARATI; PORTER, 2011). Onde, o Y_1 é o lag_1 da série temporal, $\beta 1$ é o coeficiente do lag_1, que o modelo estima, e c é uma constante.

O modelo de MA, por sua vez, é aquele em que Y_t depende apenas dos erros de previsão defasados, ou seja, da média móvel dos termos de erro corrente e passado. Além de também considerar o termo de erro (ruído branco) e mais o valor de uma constante. Y no período t é igual a uma constante mais uma média móvel, sendo que a média móvel é simplesmente uma combinação linear dos termos de erro ruído branco (MORETTIN;

TOLOI, 2006).

$$Y_t = c + \epsilon_t + \phi_1\epsilon_{t-1} + \phi_2\epsilon_{t-2} + \cdots + \epsilon_q Y_{t-q} \tag{2.4}$$

onde, c é a uma constante, $\phi_1, \cdots, \phi_q$ são os parâmetros do modelo e os $\epsilon_t, \epsilon_{t-1}, \cdots, \epsilon_{t-q}$ são termos de erro do modelo autorregressivos das respectivas defasagens. Assim MA (q) é um processo de média móvel de ordem q.

O modelo ARIMA, em síntese, é a combinação dos termos AR e MA. Assim como nos seus componentes, possui como parâmetros as ordens de auto regressão e média móvel. Além disso, o ARIMA contém um terceiro parâmetro d, que representa o número mínimo de diferenciações necessárias para tornar a série temporal estacionária.

Portanto, o modelo ARIMA recebe como entrada três termos p, q, d:

- p é a ordem do termo Auto Regressivo (*Auto Regressive* - AR).
- q é a ordem do termo Média Móvel (*Moving Avarege* - MA).
- d é o número mínimo de diferenciações necessárias para tornar a série temporal estacionária.

$$Y_t = C + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \cdots + \beta_p Y_{t-p}\epsilon_t + \phi_1\epsilon_{t-1} + \phi_2\epsilon_{t-2} + \cdots + \epsilon_q Y_{t-q} \tag{2.5}$$

3 PROBLEMA E SOLUÇÃO PROPOSTA

Neste capítulo definimos o problema e apresentamos a nossa proposta de solução. A Seção 3.1 define o problema que abordamos. A Seção 3.2 demostra a forma como os dados foram estruturados e o funcionamento do GCN-LSTM. A Seção 3.3 explicamos a nossa implementação. A Seção 3.4 justifica a nossa escolha pelo modelo GCN-LSTM.

3.1 Definição do Problema

Focamos na abordagem de captura de amostras através da observação de rastreadores. Isso nos oferece a possibilidade de identificar a população de pares, o que aumenta o nível detalhamento do enxame a um custo razoável, significativamente menor que contactar cada par. Prevendo a quantidade de pares de enxames podemos qualificar o monitoramento em tempo de execução, não só na quantidade de monitores necessários, mas também no número de requisições (seta 2, Figura 2), visando poupar recursos. Seguindo a Equação 2.1, é possível utilizar os resultados da RNA para encontrar os próximos valores de $E[N]$, tamanho esperado da população. Tendo $E[N]$ podemos encontrar $E[X]$, que é o limite mínimo de respostas do rastreador necessárias para obter a população de um determinado enxame.

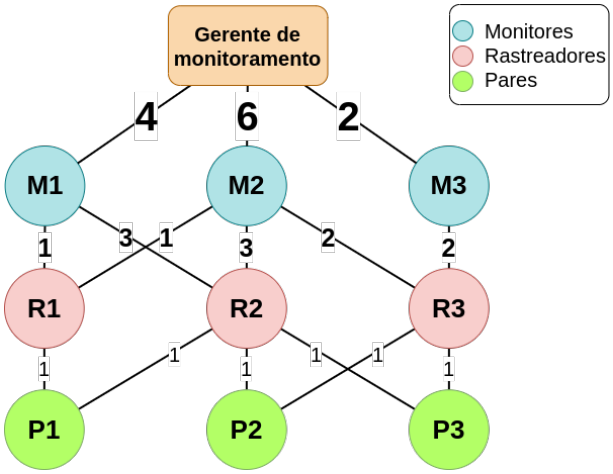
3.2 Modelo

A Figura 4 apresenta um grafo representando um exemplo básico do sistema modelado. No grafo, os monitores recebem os dados dos rastreadores e repassam para o gerente de monitoramento. Os pesos das arestas ligando os monitores e os rastreadores representam o somatório da quantidade de pares de cada rastreador monitorado, por um determinado monitor. Já nas arestas entre o gerente e os monitores é o somatório dos valores que cada monitor recebeu do conjunto de rastreadores monitorados por ele. Existe uma hierarquia nas entidade da rede (gerente, monitores, rastreadores, pares), criando um fluxo de informação dos pares até o gerente.

Percebe-se que os rastreadores R2 e R3 estão conectados a mais de um par, R2 a três pares (P1, P2 e P3) e R3 a dois pares (P2 e P3). Essa quantidade de pares, tamanho da *peer list*, é enviada pra os monitores. Que por sua vez acumulam os valores dos rastreadores, por exemplo M1 e M2, conectados a (R1 e R2) e (R1, R2 e R3) respectivamente, que recebem mais de um valor, e em seguida enviam o resultado do somatório para o gerente de monitoramento.

A Figura 5 representa todo o monitoramento, dividido em *snapshots* de tempo de 15 minutos, totalizando um número n de *snapshots*. Para cada *snapshots* temos um grafo de monitoramento, como na Figura 4, com o gerente, monitores, rastreadores e os pares. O que difere o grafo de cada *snapshots* é os pesos das aresta durante o monitoramento – ou

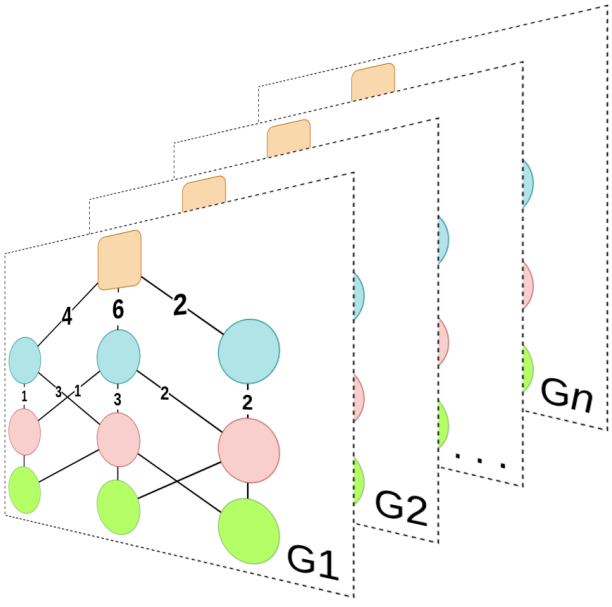
Figura 4 – Modelo do monitoramento.



Fonte – Autor

seja, o grafo permite armazenar a dinâmica nos dados ao longo do tempo, a dependência temporal.

Figura 5 – Divisão do monitoramento em grafos de tempo.

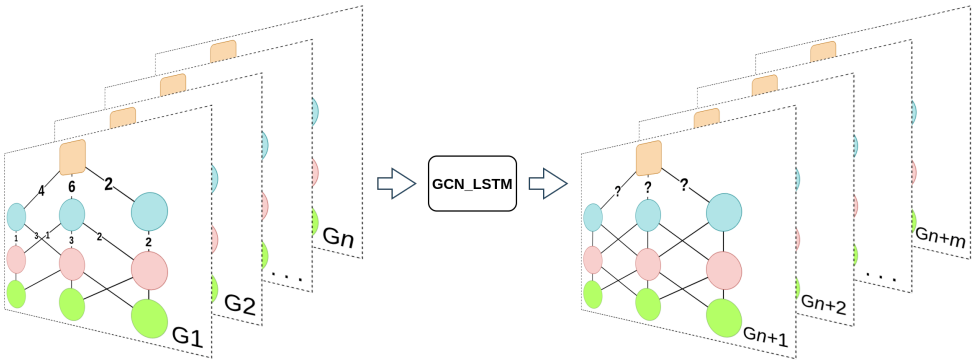


Fonte – Autor

A Figura 6 demostra o funcionamento do modelo GCN-LSTM. O modelo observa um número de n grafos e gera como a previsão os próximos m grafos. As interrogações no grafo $Gn + 1$, arestas entre o gerente e os monitores, representam os pesos que queremos de prever. Isso também acontece nos grafos seguintes, $Gn + 2, \cdots, Gn + m$. O

conjunto $\{G_1, G_2, \cdots, G_n\}$ é a janela deslizante usada como entrada para o modelo GCN-LSTM, que por sua vez irá gerar a janela deslizante de previsão, definida pelo conjunto $\{G_{n+1}, G_{n+2}, \cdots, G_{n+m}\}$.

Figura 6 – Visão geral do funcionamento do modelo GCN-LSTM.



Fonte – Autor

3.3 Prevendo a Expansão do Universo Torrent usando Redes Neurais

O Algoritmo 1 apresenta o procedimento proposto. Ele realiza as seguintes funções principais: leitura dos dados, a estruturação do modelo (como na Figura 4), e a divisão do monitoramento em *snapshots* de tempo (como na Figura 5), e, por fim, criação das matrizes de adjacências A e características X, que servem de entrada para o GCN-LSTM.

Algorithm 1 Predição

Entrada: $\alpha = 15, n = 7, m = 2$
início
  $D \leftarrow$ Leitura dos dados;
  $S \leftarrow$ Divisão de D em *snapshots* de α *minutos*;
  para $s \in S$ **faça**
    $G_s \leftarrow$ Grafo a partir s;
  fim
  $A, X \leftarrow$ Cria as matrizes a partir G_s; ▷ A, X são usadas como entrada para o GCN-LSTM
  para $i \leftarrow 0$ *até* $|S| - n - m - 1$ **faça**
    $\{G_{i+0+n}, G_{i+1+n}, \cdots, G_{i+n+m-1}\} \leftarrow \{G_{i+0}, G_{i+1}, \cdots, G_{i+n-1}\}$ ▷ Janela de entrada gera a janela de predição
  fim
fim

3.4 Justificativa de escolha da RNA Candidata: GCN-LSTM

Escolhemos o modelo GCN-LSTM por ela ter a capacidade de capturar as dependências espaciais e temporais, uma vez que, o nosso problema tem essas características. A dependência espacial é a hierarquia das entidades da rede e o fluxo da informação, que é encontrada no grafo (Figura 4). A dependência temporal é claramente encontrada na variação dos pesos nas arestas durante o monitoramento (Figura 5). É importante observar

que outras RNAs e algoritmos poderão ser considerados em trabalhos futuros, como GRU (CHO et al., 2014), Support Vector Regression model (SVR) (SMOLA; SCHÖLKOPF, 2004) e os próprios GCN e LSTM de forma independente.

4 AVALIAÇÃO

Este capítulo demostra a metodologia e os resultados da avaliação da solução proposta para predição da expansão de enxames BitTorrent. A Seção 4.1 apresenta o conjunto de dados, os parâmetros de entrada do GCN-LSTM, ARIMA e o ambiente de teste. A Seção 4.3 apresenta e discute os resultados.

4.1 Conjunto de Dados

Em nossa avaliação utilizamos um conjunto de dados de enxames capturados pelo monitoramento, apresentados em (CORDEIRO et al., 2021). A Tabela 1 apresenta um resumo das características de cada enxame. A coluna "Tamanho do arquivo" indica o tamanho do arquivo que está sendo compartilhado. A data de quando o enxame foi criado é listado na coluna "Criado em". As colunas "Nº rastreadores" e "Qt. de pares no início" indicam quantos rastreadores tinham no enxame, e quantos pares faziam parte do enxame quando o monitoramento iniciou, respectivamente.

Tabela 1 – Parâmetros de carga de Trabalho ($E[x]$ calculado usando $L = 200, |B| = 1$).

Id.	Tamanho arq.	Criado em	Qt. inicial de pares	Nº Rastreadores	$E[x]$
S1	168,52 GB	Jan-2018	69	28	1,66
S2	121,26 GB	Jan-2018	57	33	1,31
S3	4,96 GB	Nov-2018	1.737	22	69,80
S4	8,25 GB	Nov-2018	2.459	30	103,09
S5	3,53 GB	Nov-2018	5.392	29	247,22
S6	2,38 GB	Nov-2018	5.872	22	271,73

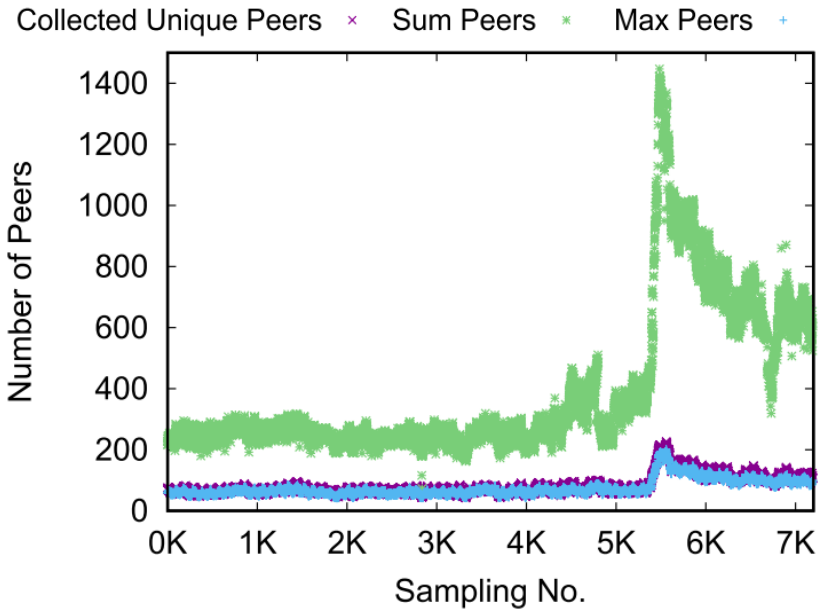
Fonte – Adaptada de (CORDEIRO et al., 2021)

A Figura 7 mostra a evolução da quantidade de pares em função das amostras de monitoramento em S1 considerado três métricas. A primeira, "Collected Unique Peers", mostra o número de pares coletados únicos em cada rodada de amostragem. A curva "Sum Peers" apresenta a soma dos pares conhecidos por cada um dos rastreadores. Por fim, o maior número de pares conhecidos por um rastreador naquela rodada de amostragem é mostrado na curva "Max Peers".

Na Figura 8, o eixo vertical apresenta o número de listas de pares e o eixo horizontal todas as amostras obtidas a partir de S1. A curva "*Collected Peer Lists*" (listas de pares coletadas) representa o número de listas de pares obtidas em cada rodada de amostragem. De acordo com a Equação 2.1 foi traçado o número estimado de listas de pares que deveriam ter sido obtidas.

A curva "Est. Req., N=(Sum), L=200" considera N como um dos pares conhecidos por cada um dos rastreadores nessa rodada de amostragem, e L como o número médio de pares contidos nas *peer lists* recuperadas. Portanto, N representa um limite superior

Figura 7 – Número de pares relatados em cada resposta obtida de cada rastreador do enxame S1 e número de pares únicos coletados do enxames S1 em cada rodada de amostragem.



Fonte – Adaptado de (CORDEIRO et al., 2021)

seguro para o número de pares no enxame, assumindo que cada par está registrado apenas em um único rastreador.

A curva "Est. Req., N=(Max), L=200" considera N como o maior número de pares conhecidos por um rastreador nessa rodada de amostragem. Nesse caso, N é um seguro inferior limite para o número de pares no enxame.

Em síntese, a Figura 7 e a Figura 8 mostram que o S1 é um traço que pode efetivamente servir como verdade fundamental, pois o número de listas de pares obtidas é significativamente maior do que o número de listas de pares necessárias estimadas (CORDEIRO et al., 2021). Isso é necessário para validar adequadamente a nossa proposta.

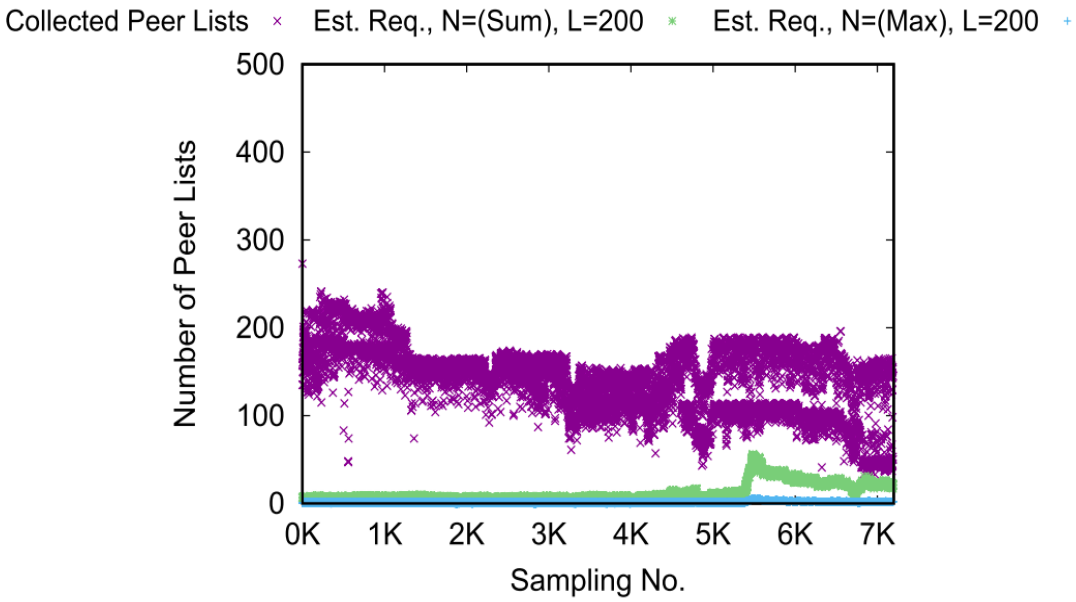
4.2 Procedimento

4.2.1 RNA

Para modelar a RNA usamos como base o exemplo de uso do GCN-LSTM disponibilizado diretamente da própria biblioteca StellarGraph (FORECASTING. . . , 2022). A partir dos dados do enxame S1, usamos o Algoritmo 1 para criar as matrizes de adjacências e *features*, que são entradas para a RNA. A Tabela 2 descreve todas as combinações de parâmetros e valores usados para avaliar a RNA.

Os dados são divididos em conjunto de treinamento e conjunto de validação, sendo

Figura 8 – Número estimado de listas de pares necessárias para monitorar o enxame S1 (com base no número de pares relatado em cada resposta obtida do rastreador) e o número de listas de pares realmente coletadas em cada rodada de amostragem.



Fonte – Adaptado de (CORDEIRO et al., 2021)

80% como treinamento e o restante como validação (Taxa de Treinamento na Tabela 2). Outro parâmetro de entrada é o tamanho da saída das camadas GCN e LSTM. Na saída das camadas GCN foi usada a função de ativação Rectified Linear Unit (ReLU) (Na Tabela 2 "relu"), geralmente usada em aprendizado profundo, com redes de multicamadas (HARA; SAITO; SHOUNO, 2015). Já na saída das camadas LSTM é aplicada a Tangente hiperbólica como funções de ativação (Na Tabela 2 "tanh"), sendo das funções mais utilizada em redes neurais (ZAMANLOOY; MIRHASSANI, 2013). Outros parâmetros relevantes do modelo são tamanho da janela deslizante de entrada, quantidade de *snapshots* observadas para a previsão, tamanho da janela deslizante de previsão, *snapshots* futuros gerados pela RNA. Ainda temos o tamanho do lote (*Batch Size*) que é um termo usado em aprendizado de máquina e refere-se ao número de exemplos de treinamento usados em uma iteração (ACADEMY, 2022). Em relação ao treinamento, também é necessário considerar o número de épocas e quantidade de iterações.

4.2.2 ARIMA

Para modelar o ARIMA, usamos como base o modelo (ARIMA..., 2022). No ARIMA, apenas a matriz de características foi usada como entrada, já que o modelo não necessita das adjacências. Tendo as matrizes prontas, fizemos uma avaliação da

Tabela 2 – Parâmetros de entrada para GCN-LSTM.

Descrição	Parâm.	Valor(es)
Épocas de treinamento		{500, 1000}
Tamanhos da saída da camadas GCN		{8, 16, 32}
Ativações aplicadas à saída de cada camada GCN		"relu"
Tamanhos da saídas da camadas LSTM		{100, 200, 300}
Ativações aplicadas à saída de cada camada LSTM		"tanh"
batch size		{16, 32, 64}
Taxa de treinamento		0,8
Séries históricas usadas no treinamento (janela de entrada)	n	{1, 2, 4, 8, 16}
Séries de previsão do treinamento (janela de predição)	m	{1, 2, 4, 8, 16}

Fonte – Autor

sensibilidade dos parâmetros de entrada de ambos os modelos. Na Tabela 3 listamos as cinco execuções do ARIMA usadas em nossos testes.

Na avaliação do ARIMA também dividimos os dados em 80% como treinamento e 20% como validação. Assim como mencionado na Seção 2.4, o ARIMA possui três parâmetros de entrada p, q e d. Onde p é a ordem do termo Auto Regressivo, q é a ordem do termo da Média Móvel e o d é o número mínimo de diferenciações necessárias para tornar a série estacionária. Para escolher os parâmetros do ARIMA realizamos um estudo exploratório de valores paramétricos considerando os seguintes conjuntos de parâmetros para cada termo do ARIMA: $p = \{0, 1, 2, 3, 4\}$, $q = \{0, 1, 2, 3, 4\}$ e $d = \{0, 1, 2\}$. Analisamos a combinação dos resultados dos três conjuntos e selecionamos os cinco melhores resultados para usar como base de comparação com o GCN-LSTM.

Visando entender o comportamento da RNA no problema, fizemos algumas simplificações no grafo do monitoramento. Em nossa avaliação, levamos em conta apenas as quantidades de pares (tamanho da *peer list*) vinda diretamente dos rastreadores, ou seja, pegamos apenas os valores das arestas entre os rastreadores e os monitores. Com isso, não estamos considerando a dependência espacial do problema, já que o modelo completo do monitoramento (Figura 4) não foi representado. Esperamos superar essa limitação em trabalhos futuros.

Como resultado, a RNA gera uma previsão independente de cada *tracker*. Assim, para obter uma visão global do enxame é feita uma média de cada *snapshot* gerado pela rede neural, tendo assim um observação da quantidade de pares do enxame. No ARIMA além dos parâmetros iniciais recebe a matriz de características, sendo para cada coluna, que representa um *trackers*, uma execução do ARIMA e, por fim é tirada uma média dos resultados de todos os ARIMAs, produzindo uma visão global do enxame, assim como na RNA. A partir da média dos resultados, obtemos o gráfico da previsão média da quantidade de pares por rastreador. É importante observar que para estimar o tamanho total do enxame, alguém poderia sugerir o somatório dos resultados. Porém, o somatório não

revelaria o real tamanho do enxame pois muitos pares conectam com múltiplos rastreador (*i.e.*, há sobreposição) e isso geraria contagem duplicada de (potencialmente muitos) pares.

Tabela 3 – Parâmetros de entrada para o ARIMA.

ARIMA	p	q	d
A1	2	3	2
A2	1	4	1
A3	2	4	1
A4	2	2	1
A5	2	3	1

Fonte – Autor

4.2.3 Métricas

Para comparar os algoritmos, usamos duas métricas quantitativas principais:

- Média do erro quadrático médio (MEQM). Obtido através da diferença entre o valor previsto e o valor considerado real. Usado como métrica de eficácia da solução.
- Tempo de execução. Obtido através da medição em tempo real da execução de um algoritmo para uma entrada em um ambiente de teste. Usado como métrica de eficiência da solução.

Além disso, considerandos a diferença visual entre curva prevista e a curva considerada verdadeira, e o erro quadrático médio (EQM) ao longo dos *snaphots*. Para cada experimento foram realizadas dez execuções. Apresentamos a média dos dez resultados ou o melhor resultado dependendo do contexto. Todas as execuções foram realizados em um mesmo ambiente de teste, como segue.

4.2.4 Ambiente de Testes

A Tabela 4 elenca os parâmetros de software e hardware do ambiente de testes usado em todos os experimentos.

4.3 Resultados

Nesta seção apresentamos os resultados das avaliações realizadas conforme metodologia apresentada anteriormente. Apresentamos os resultados obtidos com RNA e ARIMA nas próximas duas subseções, respectivamente. Em seguida, realizamos uma comparação dos resultados.

Tabela 4 – Parâmetros do ambiente de teste.

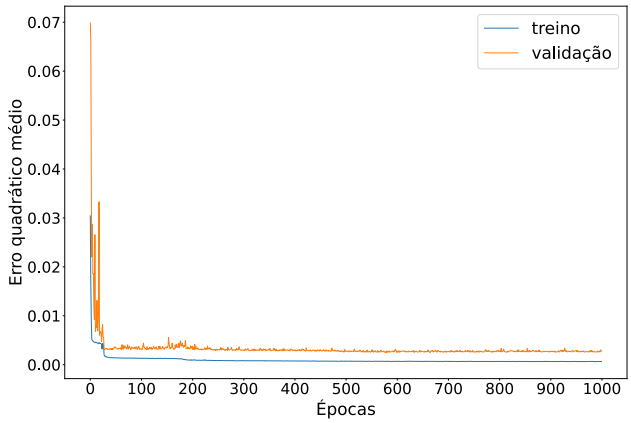
Componente	Valor(es)
Sistema Operacional	Ubuntu 20.04.3 LTS (Focal Fossa)
Kernel	5.13.0-51-generic
Python3	3.8.10
StellarGraph	stellargraph-1.2.1
ARIMA	statsmodels-0.13.0
Processador	AMD Ryzen 7 5800X - 8-CORE, 16-THREADS, 3.8GHZ
Memória	32GB

Fonte – Autor

4.3.1 RNA

Nesta seção apresentamos 7 avaliações de sensibilidade paramétrica. Na primeira, avaliamos o impacto da quantidade de épocas de treinamento na qualidade da resposta da RNA, evitando assim o subajuste (*underfitting*) e sobre-ajuste (*overfitting*) no treinamento da RNA. Para isso, monitoramos o impacto do número de épocas de treinamento entre 0 e 1.000. A Figura 9 mostra a evolução do erro médio quadrático, da RNA em função do número entre 0 e 1000 épocas de treinamento, comparando os resultados obtidos nas amostras de treinamento com os resultados obtidos nas amostras de validação, conjunto de teste. Observe-se que as curvas tendem a estabilidade indicando a inexistência tanto *underfitting* como de *overfitting* a partir de 300 épocas. Outro indício da inexistência de *underfitting* é que as curvas de treinamento e validação estão próximas. Nas próximas avaliações consideramos 500 e 1.000 épocas de treinamento.

Figura 9 – Validação até 1.000 épocas (SeqLen = 8, PredLen = 1, GCN = 16, LSTM = 200, *batch size* = 32).

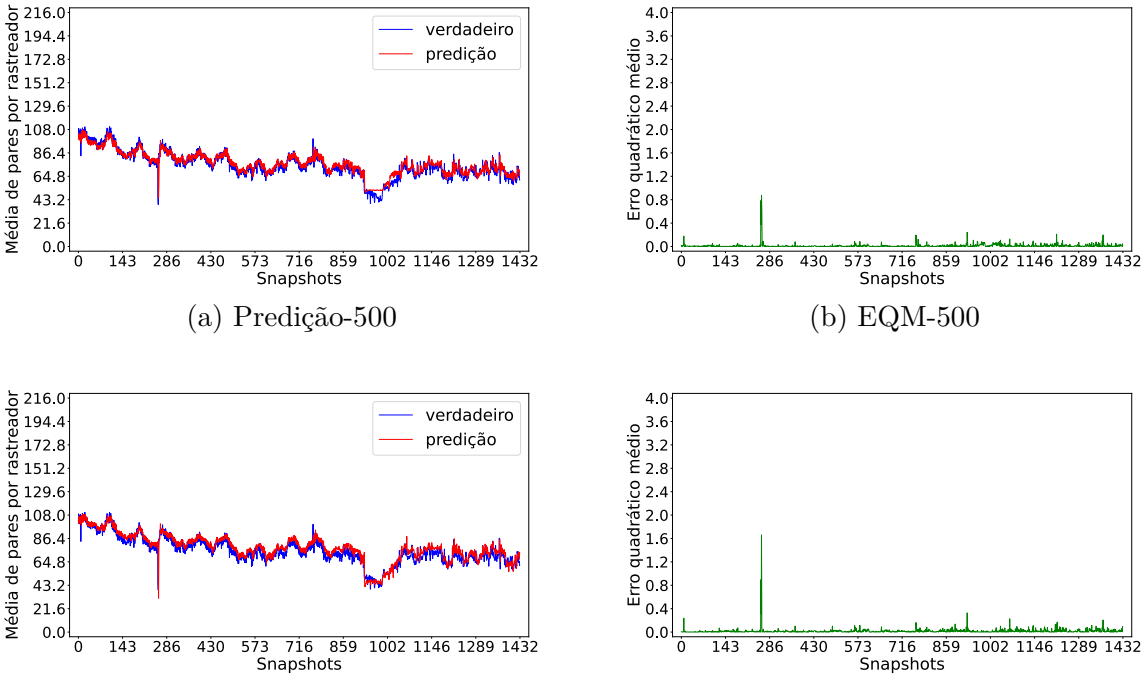


Fonte – Autor

A segunda avaliação de sensibilidade paramétrica é sobre número de épocas de

treinamento. A Figura 10 apresenta resultados de aplicação do algoritmo para dois casos de número de épocas de treinamento e duas métricas – diferença visual e erro média quadrático.

Figura 10 – Predição e erro quadrático médio de {500, 1000} épocas (SeqLen = 8, PredLen = 1, GCN = 16, LSTM = 200, *batch size* = 32, épocas = {500, 1000}).

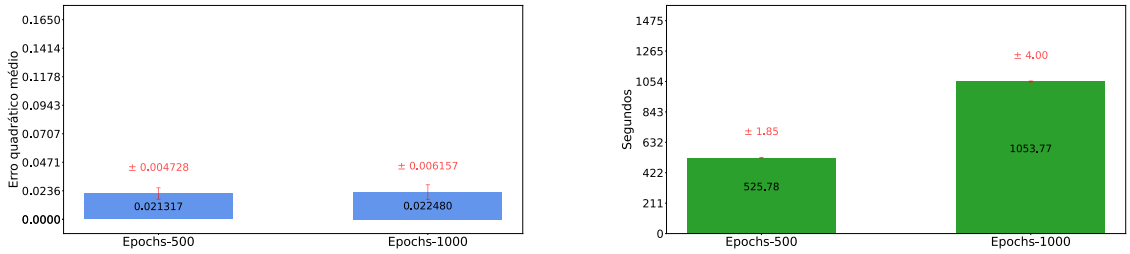


(a) Predição-500 (b) EQM-500

(c) Predição-1000 (d) EQM-1000

Fonte – Autor

Figura 11 – Médias do erro quadrático médio e tempo de execução de {500, 1000} épocas em dez execuções (SeqLen = 8, PredLen = 1, GCN = 16, LSTM = 200, *batch size* = 32, épocas = {500, 1000}).



(a) Média do erros quadrático médio. (b) Média do tempo de execução.

Fonte – Autor

As figuras 10a e 10c mostram visualmente que não há diferença nos resultados das predições. Isso também não é perceptível nas imagens que apresentam o EQM ao longo

dos *snaphots*, nas figuras 10b e 10d. A média de dez execuções foi tirada e apresentada na Figura 11a. Nota-se que quinhentas épocas teve um menor erro, além de que, o tempo médio das dez execuções é cerca de metade do valor, Figura 11b. Com base nesses resultados, todas os demais experimentos foram feitos com quinhentas épocas de treinamento.

A terceira avaliação de sensibilidade paramétrica é sobre a quantidade de saídas da camada GCN. Fixamos os de mais parâmetros de entrada, e para a camada CGN testamos os valores de {8, 16, 32}, estabilizando a camada LSTM em 200, *batch size* em 32 e janelas de entrada e previsão em 8 pra 1, respectivamente. A Figura 12 mostra os resultados dessa avaliação.

Percebe-se na Figura 12, que o resultado das três avaliações é praticamente indistinguível. Porém, na Figura 13a, onde temos a média das dez execuções, é possível notar que usar 8 saídas não é a melhor escolha. Além da média do erro ser um pouco maior que os outros casos, o desvio padrão é o mais alto entre os três. Olhando para o gráfico da Figura 13b e levando em conta que 16 e 32 saídas tem um erro próximo, buscando desempenho da rede neural a melhor escolha seria usar 16 saídas na camada, já que 32 saídas tem quase o dobro de tempo de execução.

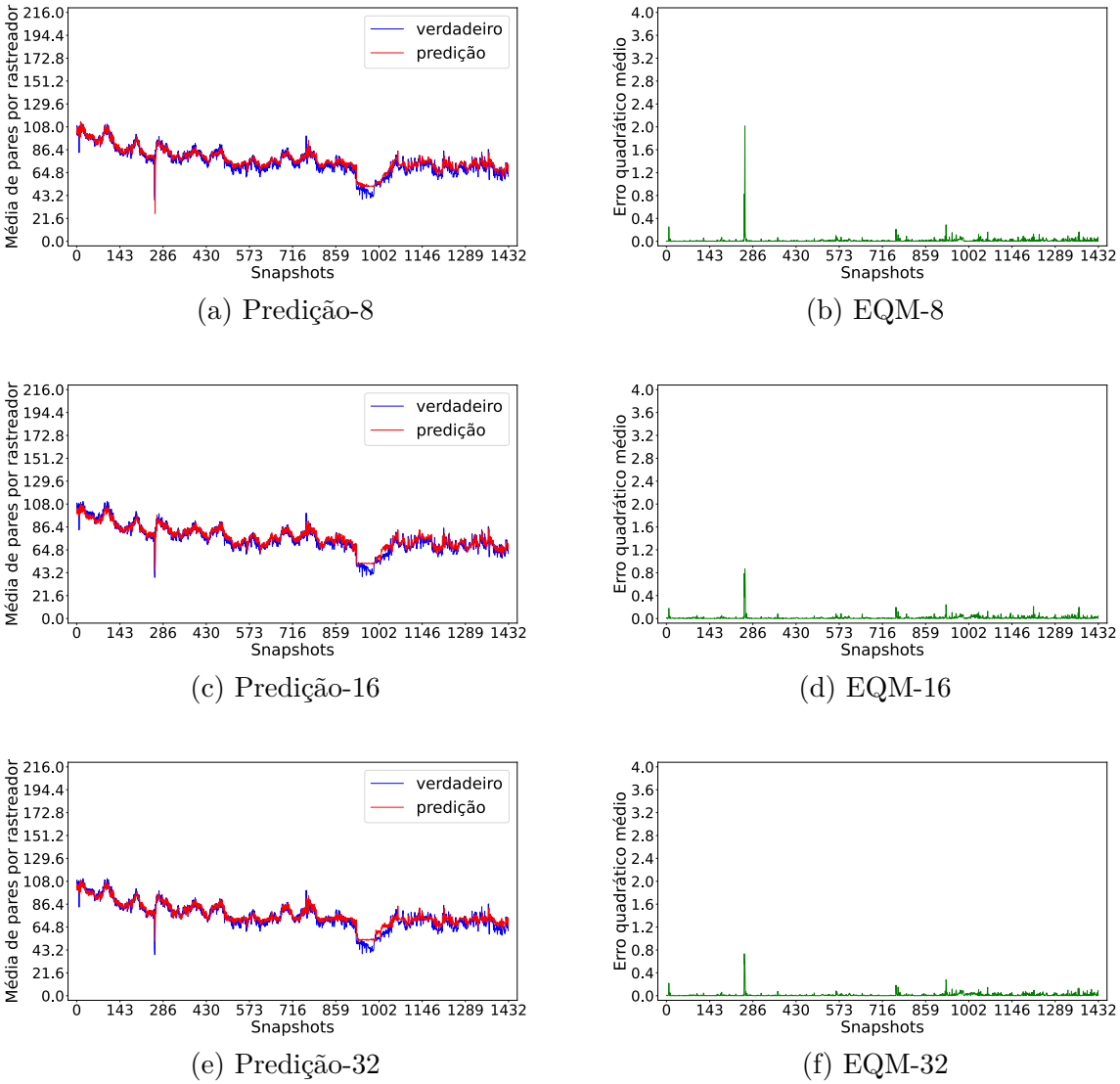
A quarta avaliação de sensibilidade paramétrica é sobre as saídas da camada LSTM com o conjunto de teste {100, 200, 300}. Para os demais parâmetros usamos 16 na GCN, 32 como *batch size*, 8 para janela de entrada e 1 para janela de previsão. A Figura 14 mostra os resultados dessa etapa.

Assim como no caso anterior, neste caso não é possível identificar visualmente diferenças significativas nos resultados. Concluímos que a melhor escolha é usar 100 saídas na camada, já que é a que tem o menor erro, conforme a Figura 15a. Além do menor erro também tem o menor tempo de execução, como pode ser visto na Figura 15b.

A quinta avaliação de sensibilidade paramétrica é sobre o *batch size*. A Figura 16 mostra o resultado do impacto do *batch size* = {16, 32, 64} nas métricas consideradas. Novamente, não notamos diferenças claras. Uma análise quantitativa é mostrada Figura 17. Nela, é possível observar resultados semelhantes do erro com tamanho de 16 e 32. Contudo, o erro e o tempo de execução de 64 para o tamanho do lote é o menor, então escolhemos usar 64 como *batch size*.

A sexta avaliação de sensibilidade paramétrica é sobre o tamanho da janela de predição (PredLen), onde testamos o conjunto de valores {1, 2, 4, 8, 16}. Novamente fixamos os outros parâmetros, GCN 16, LSTM 200, lote 32 e janela de entrada 8. Os resultados da predição são mostrados nas Figuras 18a, 18c, 18e, 18g e 18i. Nelas, vemos claramente uma diferenças na predição, com janela de tamanho 1 e 2 a linha vermelha, de predição, acompanha a azul, dados verdadeiros, o que já não acontece nos casos de tamanho 4, 8, e 16. Nas imagens do EQM, Figuras 18b, 18d, 18f, 18h e 18j, também é possível notar a distinção, no serrilhamento da linha verde. Nas médias, Figuras 19, a

Figura 12 – Predição e erro quadrático médio de {8, 16, 32} saídas GCN (SeqLen = 8, PredLen = 1, GCN = {8, 16, 32}, LSTM = 200, *batch size* = 32, épocas = 500).



(a) Predição-8 (b) EQM-8

(c) Predição-16 (d) EQM-16

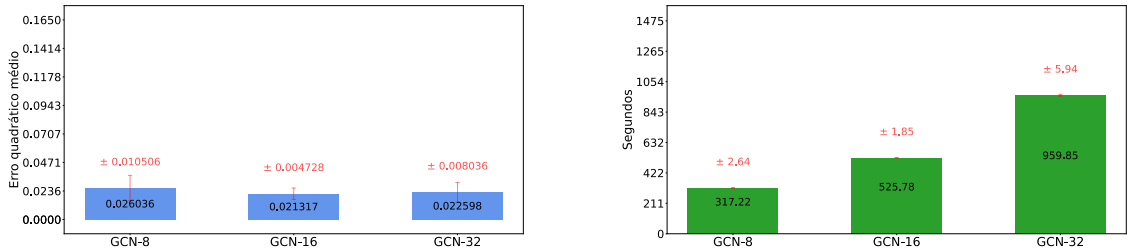
(e) Predição-32 (f) EQM-32

Fonte – Autor

janela de tamanho um é a que tem o menor erro. Como o tempo de execução praticamente não se alterou drasticamente no restante do conjunto de teste, evidentemente a preferência pelo tamanho de apenas uma janela de predição.

A sétima e última avaliação de sensibilidade paramétrica é sobre o tamanho o tamanho da janela de entrada (SeqLen), para a qual usamos com o conjunto de {1, 2, 4, 8, 16}. Aqui também não fomos capazes de enxergar a disparidade dos casos, tanto nas predições como nos erros. Mas observando a Figura 21a, nota-se que os tamanho de 1, 8 e 16, não são boas escolhas, 16 tendo o pior desvio padrão de todos os teste realizados para a RNA. Duas e quatro janelas tiveram resultados próximos, mas note-se que o desvio padrão de 2 é praticamente a metade do valor de 4. Tendo em conta que a variação no

Figura 13 – Médias do erro quadrático médio e tempo de execução de {8, 16, 32} saídas GCN em dez execuções. (SeqLen = 8, PredLen = 1, GCN = {8, 16, 32}, LSTM = 200, *batch size* = 32, épocas = 500).



(a) Médias dos erros quadrático médio. (b) Média do tempo de execução.

Fonte – Autor

tempo de execução é mínima a melhor alternativa é usar o tamanho da janela como 2.

4.3.2 ARIMA

Nesta seção apresentados os resultados das avaliações realizadas com cinco combinações de parâmetros do ARIMA listadas anteriormente. Nesta avaliação, nós usamos as mesmas entradas e ambiente das avaliações sobre a RNA.

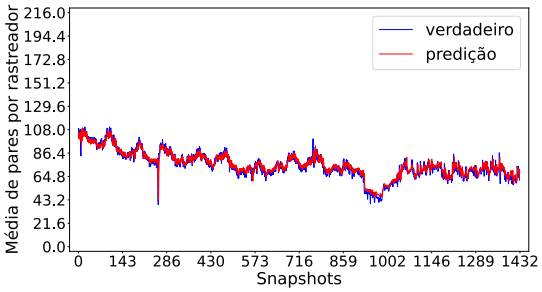
A Figura 22 apresenta os resultados de predição e erro quadrático médio, e a Figura 23b apresenta os dados sintetizados. Observe-se que todos os casos apresentam bons resultados. Por exemplo, observe-se na Figura 23a que todos os casos resultado em valores de desvio padrão próximos de zero. Em outras palavras todas as execuções deram o praticamente o mesmo erro sempre. A principal diferença se deu no tempo de execução, Figura 23b. Para comparar com a RNA selecionamos o A4.

4.3.3 Discussão

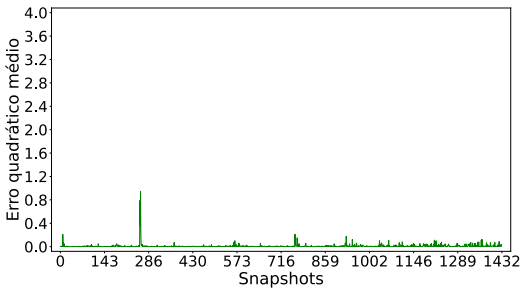
Nós comparamos RNA e ARIMA considerando os melhores resultados obtidos por cada. Observe-se na Figura 24a que o erro quadrático médio da rede neural tem pouco mais que o dobro do erro do ARIMA – o que é pouco significativo considerando que o erro do ARIMA é pequeno. Por outro lado, o tempo de execução é mais de vinte e cinco vezes maior no ARIMA, conforme a Figura 24b. Com isso, concluímos que a rede neural GCN-LSTM pode ser uma alternativa mais rápida de executar do que o ARIMA, embora seu resultado seja um pouco pior.

Lembrando que estamos usando a versão simplificada do problema, pois com o modelo GCN-LSTM não conseguimos avaliar o grafo de monitoramento (Figura 4) por completo. Já que a camada GCN só pode ser aplicável a grafos não direcionados, onde os relacionamentos entre os nós conectados são simétricos de duas vias, ou seja, as in-

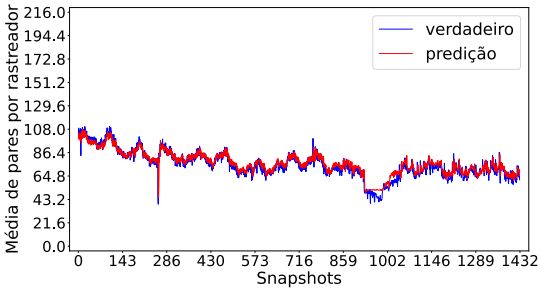
Figura 14 – Predição e erro quadrático médio de {100, 200, 300} saídas LSTM (SeqLen = 8, PredLen = 1, GCN = 16, LSTM = {100, 200, 300}, *batch size* = 32, épocas = 500).



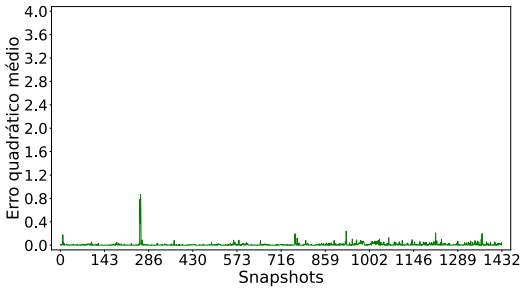
(a) Predição-100



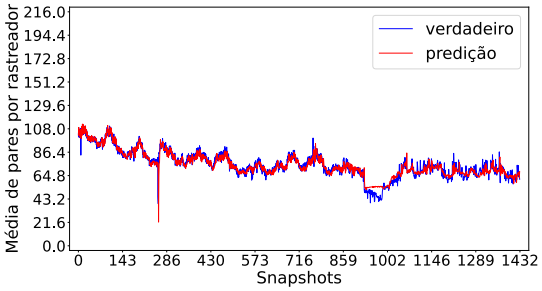
(b) EQM-100



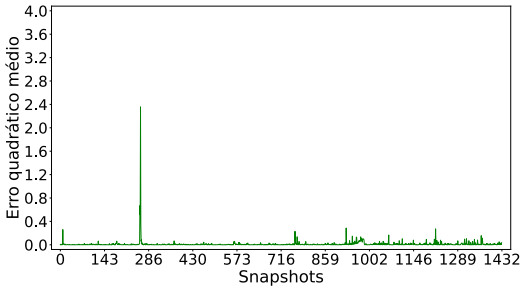
(c) Predição-200



(d) EQM-200



(e) Predição-300



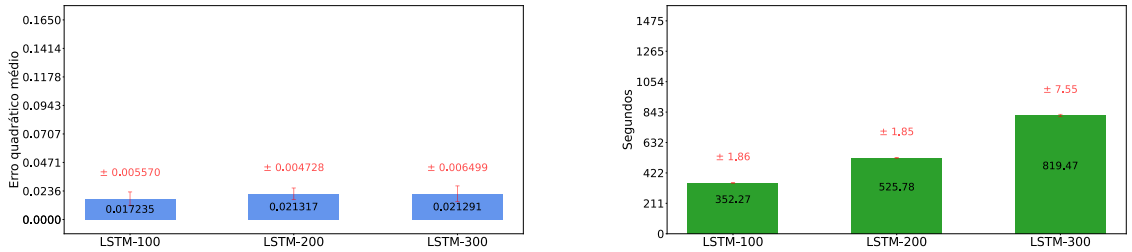
(f) EQM-300

Fonte – Autor

formações podem ser passadas para frente e para trás(TAN; LIU; YIN, 2021). No nosso problema temos um grafo direcionado (dígrafo), onde fluxo de informação é assimétrico, a informação é passada em uma única direção. Talvez usando a versão simplificada do problema os dados do grafo de monitoramento possam estar passando apenas pela camada LSTM, sofrendo apenas alguma ou nenhuma interferência da camada GCN, já que estamos considerando apenas o tamanho da *peer list*.

Para aplicações onde o fluxo de informação é assimétrico, a informação só pode ser passada em uma direção, foi criado o Digraph Inception Convolutional Networks (DiGCN) que utiliza convolução em dígrafos e proximidade de ordem k para alcançar campos receptivos maiores e aprender recursos multiescala em dígrafos. DiGCN podem

Figura 15 – Médias do erro quadrático médio e tempo de execução de {100, 200, 300} saídas LSTM em dez execuções (SeqLen = 8, PredLen = 1, GCN = 16, LSTM = {100, 200, 300}, *batch size* = 32, épocas = 500).



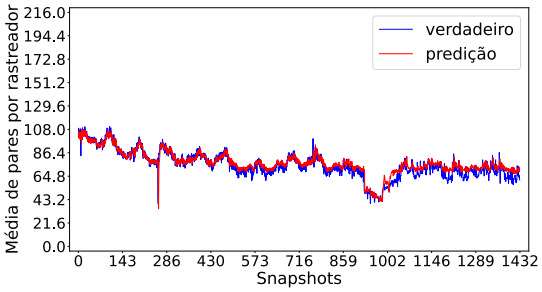
(a) Média do erros quadrático médio.

(b) Média do tempo de execução.

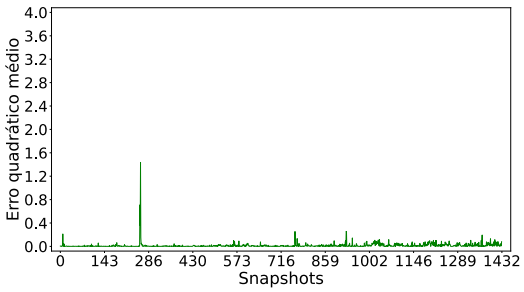
Fonte – Autor

codificar mais informações estruturais em dígrafos do que GCNs (TONG et al., 2020). Talvez o DiGCN possa ser combinada com o LSTM assim como GCN foi. Criando uma nova rede neural capaz de aprender a estrutura topológica em grafos direcionados, pelo DiGCN e a mudança na dinâmica dos dados ao longo do tempo, pelo LSTM. Caso essa nova rede neural venha a ser criada este trabalho pode ser usado como uma possível avaliação a mesma, talvez pequenos ajustes tenham que ser feitos.

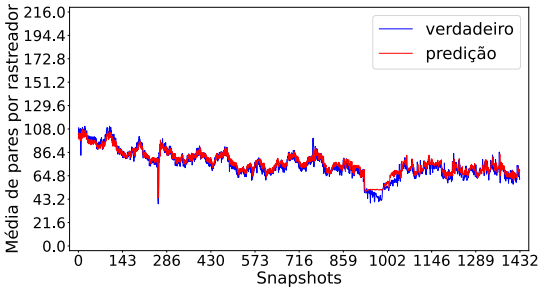
Figura 16 – Predição e erro quadrático médio de {16, 32, 64} *batch size*. (SeqLen = 8, PredLen = 1, GCN = 16, LSTM = 200, *batch size* = {16, 32, 64}, épocas = 500).



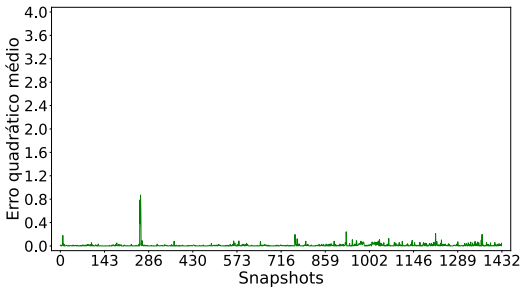
(a) Predição-16



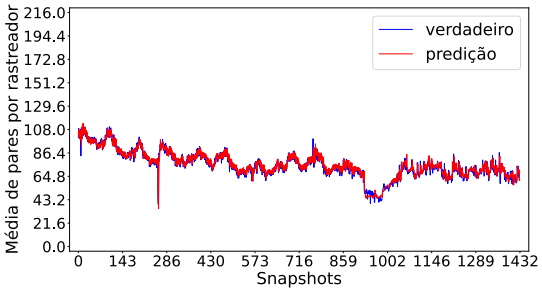
(b) EQM-16



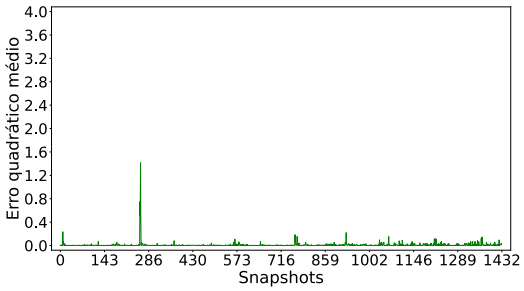
(c) Predição-32



(d) EQM-32



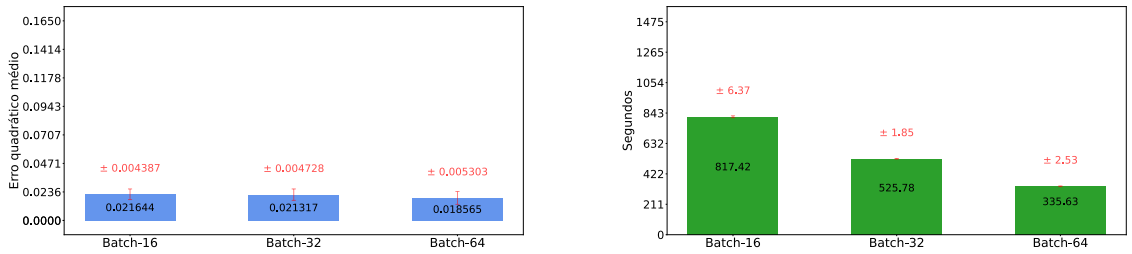
(e) Predição-64



(f) EQM-64

Fonte – Autor

Figura 17 – Médias do erro quadrático médio e tempo de execução de {16, 32, 64} *batch size* em dez execuções (SeqLen = 8, PredLen = 1, GCN = 16, LSTM = 200, *batch size* = {16, 32, 64}, épocas = 500).

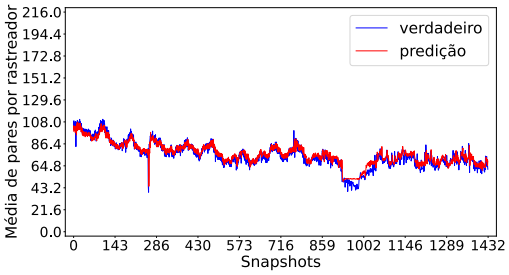


(a) Média do erros quadrático médio.

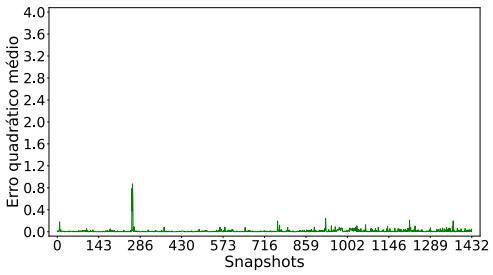
(b) Média do tempo de execução.

Fonte – Autor

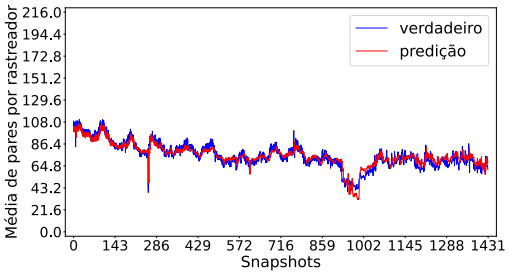
Figura 18 – Predição e erro quadrático médio de {1, 2, 4, 8, 16} janelas de predição. (SeqLen = 8, PredLen = {1, 2, 4, 8, 16}, GCN = 16, LSTM = 200, *batch size* = 32, épocas = 500).



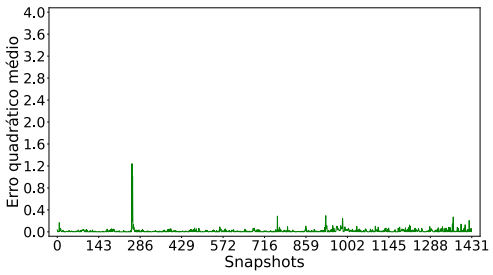
(a) Predição-1



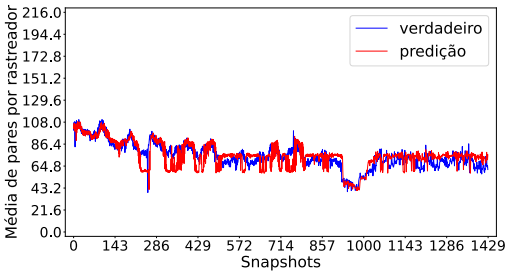
(b) EQM-1



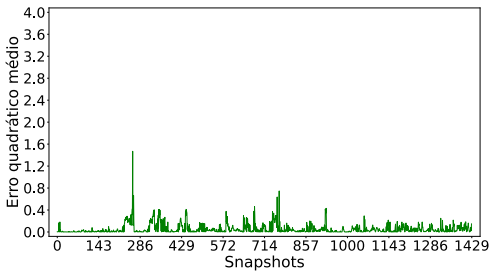
(c) Predição-2



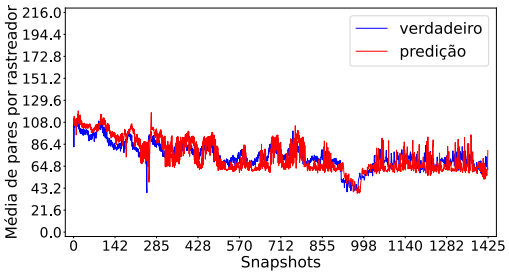
(d) EQM-2



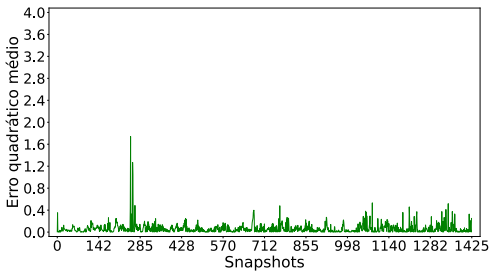
(e) Predição-4



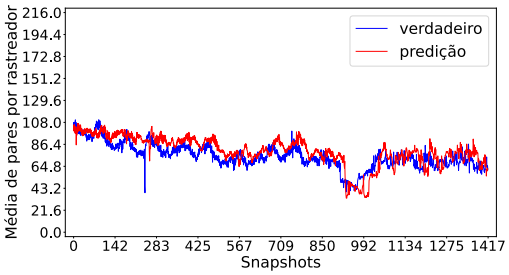
(f) EQM-4



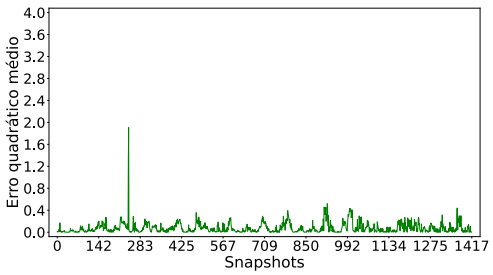
(g) Predição-8



(h) EQM-8



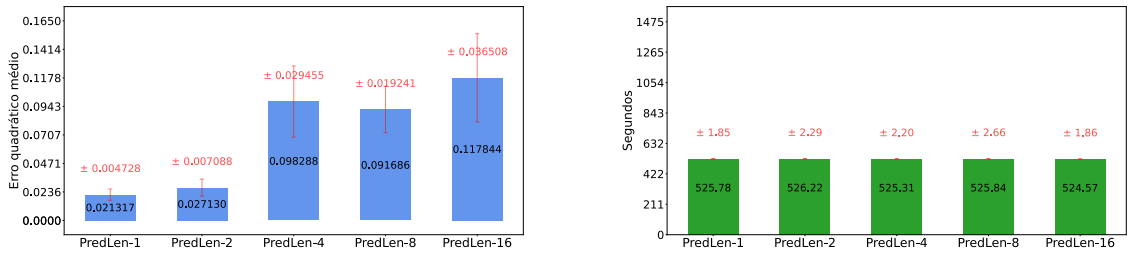
(i) Predição-16



(j) EQM-16

Fonte – Autor

Figura 19 – Médias do erro quadrático médio e tempo de execução de {1, 2, 4, 8, 16} janelas de predição em dez execuções. (SeqLen = 8, PredLen = {1, 2, 4, 8, 16}, GCN = 16, LSTM = 200, *batch size* = 32, épocas = 500).

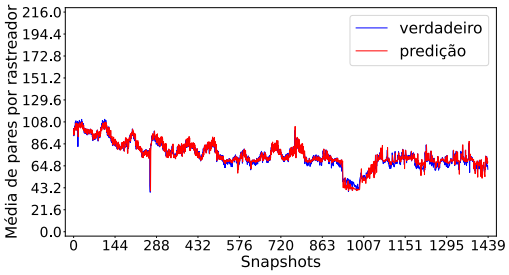


(a) Média do erros quadrático médio.

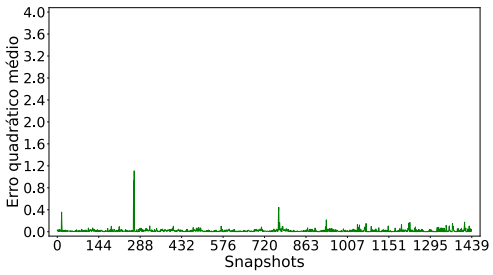
(b) Média do tempo de execução.

Fonte – Autor

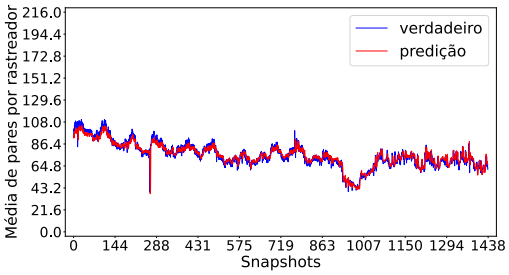
Figura 20 – Predição e erro quadrático médio de {1, 2, 4, 8, 16} janelas de entrada. (SeqLen = {1, 2, 4, 8, 16}, PredLen = 1, GCN = 16, LSTM = 200, *batch size* = 32, épocas = 500).



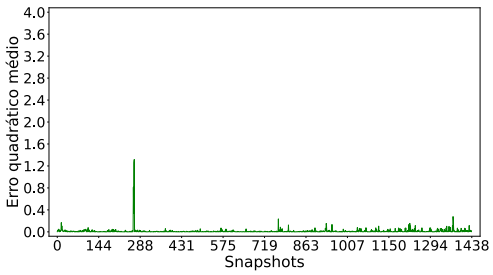
(a) Predição-1



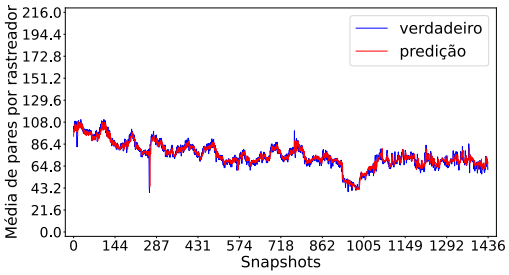
(b) EQM-1



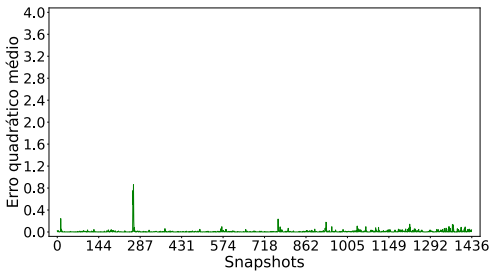
(c) Predição-2



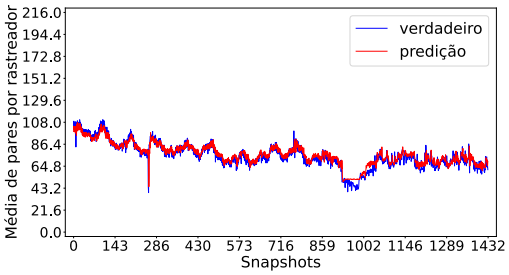
(d) EQM-2



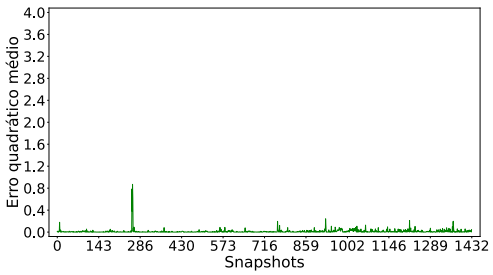
(e) Predição-4



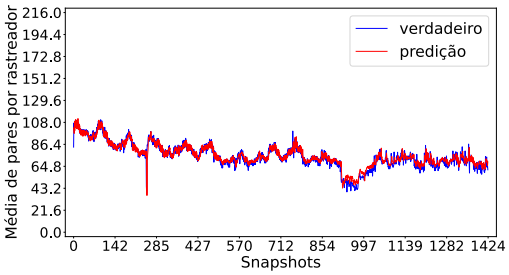
(f) EQM-4



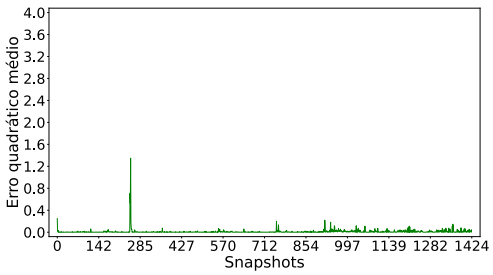
(g) Predição-8



(h) EQM-8



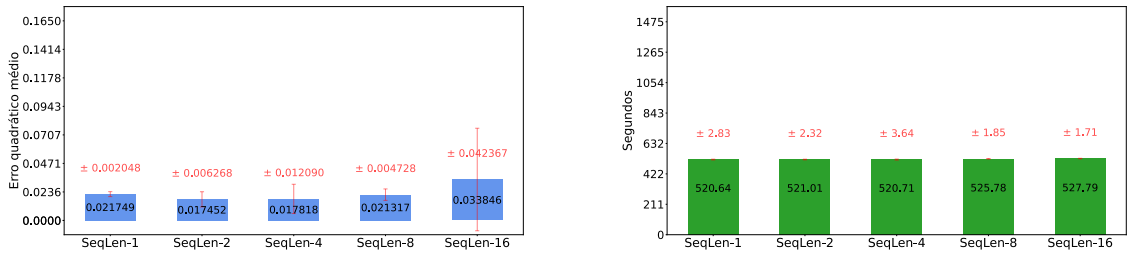
(i) Predição-16



(j) EQM-16

Fonte – Autor

Figura 21 – Médias do erro quadrático médio e tempo de execução de {1, 2, 4, 8, 16} janelas de entrada em dez execuções. (SeqLen = {1, 2, 4, 8, 16}, PredLen = 1, GCN = 16, LSTM = 200, *batch size* = 32, épocas = 500).

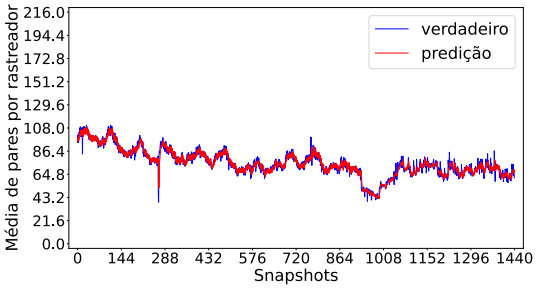


(a) Média do erros quadrático médio.

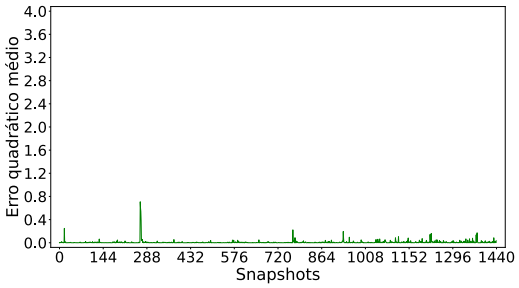
(b) Média do tempo de execução.

Fonte – Autor

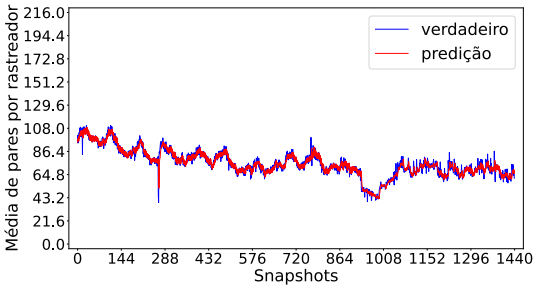
Figura 22 – Predição e erro quadrático médio de {A1, A2, A3, A4, A5}.



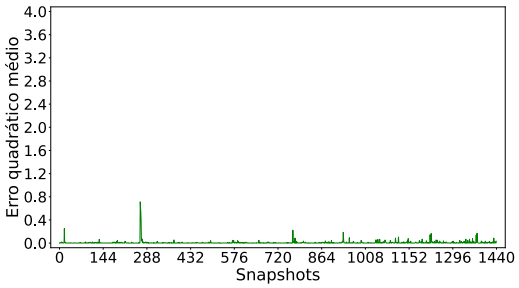
(a) Predição-A1



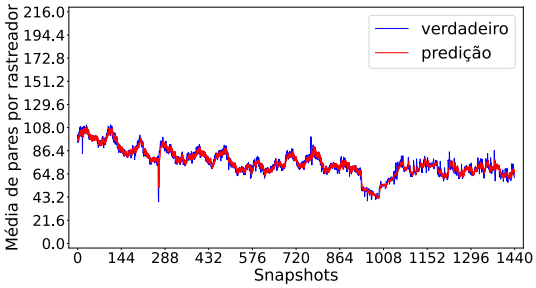
(b) EQM-A1



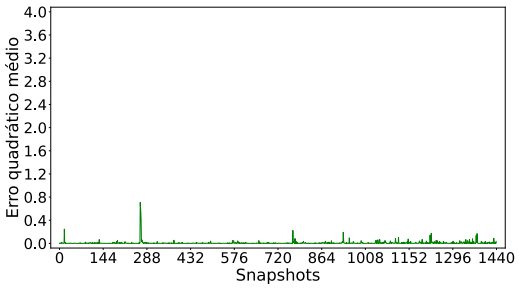
(c) Predição-A2



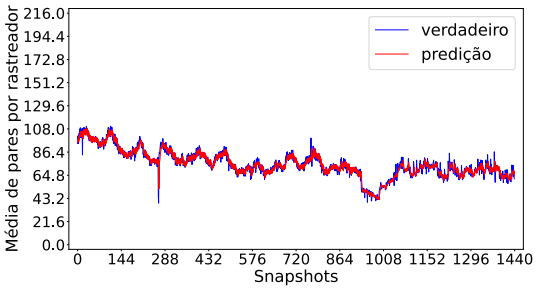
(d) EQM-A2



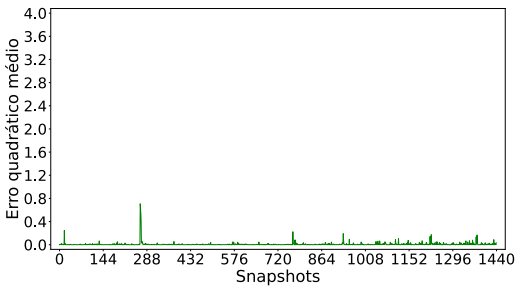
(e) Predição-A3



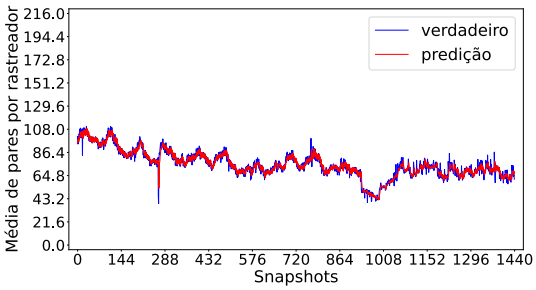
(f) EQM-A3



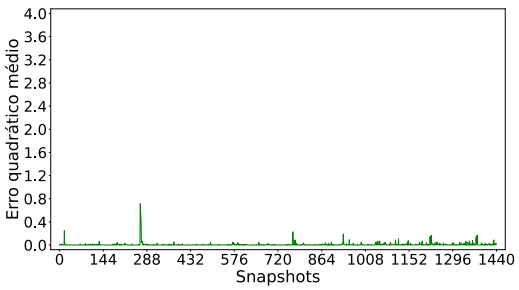
(g) Predição-A4



(h) EQM-A4



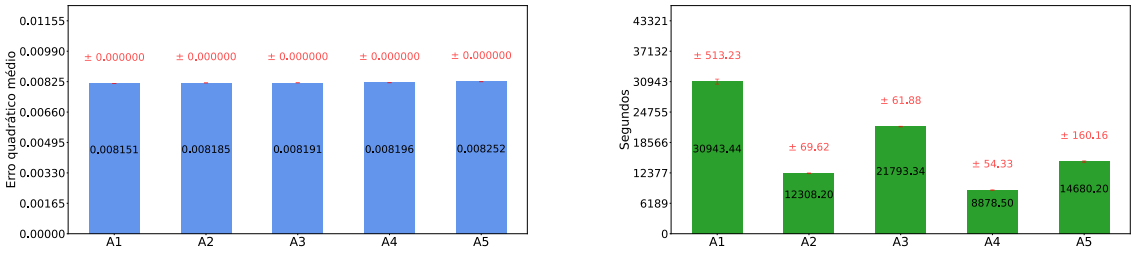
(i) Predição-A5



(j) EQM-A5

Fonte – Autor

Figura 23 – Médias do erro quadrático médio e tempo de execução de {A1, A2, A3, A4, A5} em dez execuções.

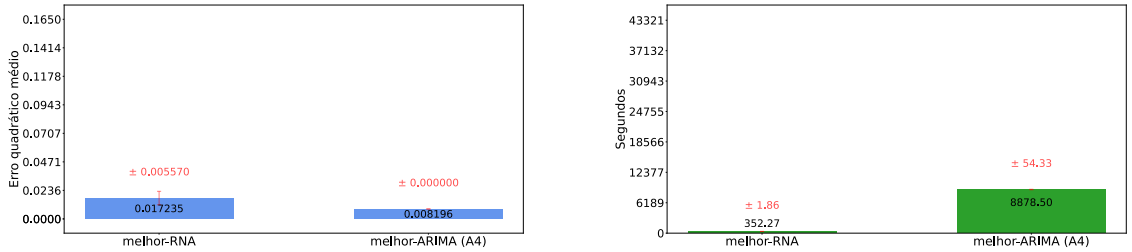


(a) Média do erros quadrático médio.

(b) Média do tempo de execução.

Fonte – Autor

Figura 24 – Médias do erro quadrático médio e tempo de execução do melhores RNA (SeqLen = 8, PredLen = 1, GCN = 16, LSTM = 100, *batch size* = 32, épocas = 500) e ARIMA (A4) em dez execuções.



(a) Média do erros quadrático médio.

(b) Média do tempo de execução.

Fonte – Autor

5 CONCLUSÃO

5.1 Considerações Finais

Prever a evolução da quantidade de usuários em enxames é um desafio, pois enxames são dinâmicos e perturbáveis por fenômenos de *churn*. Descobrir o tamanho da população, em um dado momento futuro, nos permitiria qualificar o monitoramento em tempo de execução, contribuindo para os trabalhos (CORDEIRO et al., 2014; CORDEIRO et al., 2021; PAIM et al., 2021), que são voltados para qualificação dos dados do monitoramento de maneira posterior ao monitoramento. Com isso, teremos um monitoramento mais eficiente e dinâmico, adaptando-se aos comportamentos de redes BitTorrent.

Utilizamos os dados do próprio monitoramento para avaliar os dois métodos empregamos neste trabalho. Para isso, modelamos a rede de monitoramento (Figura 4), dividimos o tempo total do monitoramento, em intervalos de tempo regulares (Figura 5). Mesmo que tenhamos usado a versão simplificada do problema, ainda assim conseguimos de certo modo comparar os resultados dos métodos. Atestando que o modelo GCN-LSTM pode ser uma alternativa ao ARIMA.

5.2 Trabalhos Futuros

Vislumbramos alguns trabalhos futuros. No campo da avaliação, poderíamos rodar os experimentos de RNA em alguma Graphics Processing Unit (GPU) para verificar o impacto na eficiência, já que sua estrutura paralela diminuiria consideravelmente o tempo de execução. Além disso, poderíamos estudar como os métodos se comportam nos diferente conjunto de dados, já que só realizamos testes no enxame S1. Testar nos outros enxames, para ver o comportamento dos métodos em fenômenos de *churn*. Por fim, no campo de desenvolvimento, poderíamos explorar novas topologias de rede neural. Por exemplo, poderíamos considerar a DiGCN, que é uma versão de GCN para grafos direcionados, que poderia ser unida com o LSTM, criando assim DiGCN-LSTM.

REFERÊNCIAS

ACADEMY, D. S. **Deep Learning Book**. 2022. <https://www.deeplearningbook.com.br>. (Acesso em 05/06/2022). Citado na página 33.

AHMED, M. S.; COOK, A. R. **Analysis of freeway traffic time-series data by using Box-Jenkins techniques**. [S.l.: s.n.], 1979. Citado na página 25.

ALEXA Internet, Inc. 2021. <https://www.alexa.com/toolbar>. (Acesso em 05/08/2021). Citado na página 17.

APPLICATION Usage & Threat Report. 2021. <https://www.paloaltonetworks.com/blog/app-usage-risk-report-visualization/#>. (Acesso em 08/09/2021). Citado na página 17.

ARIMA Model - Complete Guide to Time Series Forecasting in Python | ML+. 2022. <https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/>. (Acesso em 28/07/2022). Citado na página 33.

BABAI, M. Z. et al. Forecasting and inventory performance in a two-stage supply chain with arima (0, 1, 1) demand: Theory and empirical analysis. **International Journal of Production Economics**, Elsevier, v. 143, n. 2, p. 463–471, 2013. Citado na página 25.

BRUNA, J. et al. Spectral networks and locally connected networks on graphs. **arXiv preprint arXiv:1312.6203**, 2013. Citado na página 23.

CHO, K. et al. On the properties of neural machine translation: Encoder-decoder approaches. **arXiv preprint arXiv:1409.1259**, 2014. Citado 3 vezes nas páginas 23, 24 e 30.

CHUNG, J. et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. **arXiv preprint arXiv:1412.3555**, 2014. Citado na página 25.

CORDEIRO, W. et al. Revisiting the coupon collector's problem to unveil users' online sessions in networked systems. **Peer-to-Peer Networking and Applications**, Springer, v. 14, n. 2, p. 687–707, 2021. Citado 7 vezes nas páginas 17, 18, 22, 31, 32, 33 e 51.

CORDEIRO, W. L. da C. et al. Were you there? bridging the gap to unveil users' online sessions in networked, distributed systems. In: **2014 Brazilian Symposium on Computer Networks and Distributed Systems**. [S.l.: s.n.], 2014. p. 239–248. Citado 2 vezes nas páginas 18 e 51.

DATA61, C. **StellarGraph Machine Learning Library**. [S.l.]: GitHub, 2018. <https://github.com/stellargraph/stellargraph>. Citado na página 24.

DEFFERRARD, M.; BRESSON, X.; VANDERGHEYNST, P. Convolutional neural networks on graphs with fast localized spectral filtering. **Advances in neural information processing systems**, v. 29, p. 3844–3852, 2016. Citado na página 23.

FORECASTING using spatio-temporal data with combined Graph Convolution + LSTM model — StellarGraph 1.2.1 documentation. 2022. <https://stellargraph.readthedocs.io/en/stable/demos/time-series/gcn-lstm-time-series.html>. (Acesso em 12/07/2022). Citado na página 32.

GUJARATI, D. N.; PORTER, D. C. **Econometria básica-5**. [S.l.]: Amgh Editora, 2011. Citado na página 25.

GUO, L. et al. A performance study of bittorrent-like peer-to-peer systems. **IEEE Journal on Selected Areas in Communications**, v. 25, n. 1, p. 155–169, 2007. Citado na página 17.

HAMED, M. M.; AL-MASAEID, H. R.; SAID, Z. M. B. Short-term prediction of traffic volume in urban arterials. **Journal of Transportation Engineering**, American Society of Civil Engineers, v. 121, n. 3, p. 249–254, 1995. Citado na página 25.

HARA, K.; SAITO, D.; SHOUNO, H. Analysis of function of rectified linear unit used in deep learning. In: IEEE. **2015 international joint conference on neural networks (IJCNN)**. [S.l.], 2015. p. 1–8. Citado na página 33.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997. Citado na página 25.

HOSSFELD, T. et al. Characterization of bittorrent swarms and their distribution in the internet. **Computer Networks**, Elsevier, v. 55, n. 5, p. 1197–1215, 2011. Citado 2 vezes nas páginas 17 e 23.

KIPF, T. N.; WELLING, M. Semi-supervised classification with graph convolutional networks. **arXiv preprint arXiv:1609.02907**, 2016. Citado na página 23.

KOBZA, J. E.; JACOBSON, S. H.; VAUGHAN, D. E. A survey of the coupon collector's problem with random sample sizes. **Methodology and Computing in Applied Probability**, Springer, v. 9, n. 4, p. 573–584, 2007. Citado na página 23.

KONRATH, M. A. et al. Atacando um enxame com um bando de mentirosos: vulnerabilidades em bittorrent. **XXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2007)**, v. 2, p. 883–896, 2007. Citado na página 21.

LAREIDA, A.; HOSSFELD, T.; STILLER, B. The bittorrent peer collector problem. In: IEEE. **2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)**. [S.l.], 2017. p. 449–455. Citado na página 17.

LAREIDA, A.; STILLER, B. Big torrent measurement: a country-, network-, and content-centric analysis of video sharing in bittorrent. In: IEEE. **NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium**. [S.l.], 2018. p. 1–9. Citado 2 vezes nas páginas 17 e 18.

LECUN, Y. et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, Ieee, v. 86, n. 11, p. 2278–2324, 1998. Citado na página 23.

LEHMANN, M. et al. Swarming: como bittorrent revolucionou a internet. **Atualizações em Informática. Ed. por PUC-Rio**, v. 1, p. 209–258, 2011. Citado 2 vezes nas páginas 17 e 21.

MANSILHA, R. B. et al. Observing the bittorrent universe through telescopes. In: **12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops**. [S.l.: s.n.], 2011. p. 321–328. Citado 4 vezes nas páginas 17, 18, 21 e 22.

MORETTIN, P. A.; TOLOI, C. Análise de séries temporais. In: **Análise de séries temporais**. [S.l.: s.n.], 2006. p. 538–538. Citado na página 26.

MUSA, A. Packet tracing and analysis of tcp traffic on transport layer of peer to peer networks. **ATBU Journal of Science, Technology and Education**, v. 8, n. 2, p. 270–276, 2020. Citado 2 vezes nas páginas 17 e 18.

NAIK, A. R.; KESHAVAMURTHY, B. N. Next level peer-to-peer overlay networks under high churns: a survey. **Peer-to-Peer Networking and Applications**, v. 13, n. 3, p. 905–931, 2020. Citado na página 17.

PAIM, K. O. et al. Usando redes neurais para reconstruir traços de sessões de usuários de sistemas de larga escala. In: SBC. **Anais do XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos**. [S.l.], 2021. p. 826–839. Citado 2 vezes nas páginas 18 e 51.

PAIM, K. O. et al. Fix me if you can: Using neural networks to regenerate networked systemsx2019; monitoring traces. In: **NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium**. [S.l.: s.n.], 2022. p. 1–9. Citado na página 18.

SCHMIDT, A.; BARCELLOS, M.; GASPARY, L. P. Rumo à caracterização de disseminação ilegal de filmes em redes bittorrent. **Simpósio Brasileiro em Segurança da Informação e Sistemas Computacionais - SBSEG 2011**, Nov 2011. Citado na página 17.

SMOLA, A. J.; SCHÖLKOPF, B. A tutorial on support vector regression. **Statistics and computing**, Springer, v. 14, n. 3, p. 199–222, 2004. Citado na página 30.

STEEN, M. V.; TANENBAUM, A. S. **Distributed Systems**. 3. ed. [s.n.], 2017. Disponível em: <https://www.distributed-systems.net/>. Citado na página 17.

TAN, R. et al. Bitcoin network size estimation based on coupon collection model. In: SPRINGER. **International Conference on Artificial Intelligence and Security**. [S.l.], 2019. p. 298–307. Citado na página 17.

TAN, Z.; LIU, B.; YIN, G. Asymmetric graph representation learning. **arXiv preprint arXiv:2110.07436**, 2021. Citado na página 41.

TONG, Z. et al. Digraph inception convolutional networks. **Advances in Neural Information Processing Systems**, v. 33, 2020. Citado na página 42.

ZAMANLOOY, B.; MIRHASSANI, M. Efficient vlsi implementation of neural networks with hyperbolic tangent activation function. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, IEEE, v. 22, n. 1, p. 39–48, 2013. Citado na página 33.

ZHAO, L. et al. T-gcn: A temporal graph convolutional network for traffic prediction. **IEEE Transactions on Intelligent Transportation Systems**, IEEE, v. 21, n. 9, p. 3848–3858, 2019. Citado 3 vezes nas páginas 18, 23 e 24.