UNIVERSIDADE FEDERAL DO PAMPA

Kayuã Oleques Paim

Usando Redes Neurais para Reconstruir Traços de Sessões de Usuários de Sistemas de Larga Escala

Alegrete
março, 2022

Kayuã Oleques Paim

Usando Redes Neurais para Reconstruir Traços de Sessões de Usuários de Sistemas de Larga Escala

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

Coorientador: Prof. Dr. Weverton Luis da Costa Cordeiro

Alegrete
março, 2022

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais) .

P143u Paim, Kayuã Oleques
Usando redes neurais para reconstruir traços de sessões de usuários de sistemas de larga escala / Kayuã Oleques Paim.
64 p.

Trabalho de Conclusão de Curso(Graduação)-- Universidade Federal do Pampa, CIÊNCIA DA COMPUTAÇÃO, 2022.
"Orientação: Rodrigo Brandão Mansilha".

1. Aprendizado de Máquina. 2. Sistemas Distribuídos. 3. Monitoramento.. I. Título.

KAYUÃ OLEQUES PAIM

USANDO REDES NEURAIS PARA RECONSTRUIR TRAÇOS DE SESSÕES DE USUÁRIOS DE SISTEMAS DE LARGA ESCALA

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 aprovado em 10 de março de 2022.

Banca examinadora:

Prof. Dr. Rodrigo Brandão Mansilha
Orientador
Presidente da banca examinadora

Prof. Dr. Weverton Luis Cordeiro
Coorientador

Prof.ª Dr.ª Mariana Recamonde Mendoza
Examinadora

Prof. Dr. Marcelo Caggiani Luizelli
Examinador

Assinado eletronicamente por **RODRIGO BRANDAO MANSILHA**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 10/03/2022, às 11:58, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **MARCELO CAGGIANI LUIZELLI**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 10/03/2022, às 15:03, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **Weverton Luis da Costa Cordeiro**, **Usuário Externo**, em 10/03/2022, às 18:17,

conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **Mariana Recamonde Mendoza Guerreiro**, **Usuário Externo**, em 11/03/2022, às 15:54, 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 **0749003** e o código CRC **566D66E0**.

RESUMO

O processo de monitoramento de entidades online (por exemplo, usuários, dispositivos IoT, hosts) em sistemas distribuídos de larga escala é fundamental para a compreensão de seus comportamentos, análise de sua dinâmica, propriedades, limitações e identificação de oportunidades de melhoria. Esse processo pode ser relevante em diversas situações, como monitoramento de disponibilidade de conexões de internet na última milha e avaliação do impacto de eventos naturais extremos, como clima, terremotos, e incêndios, na infraestrutura da região. Em muitos sistemas, a presença online de entidades pode ser monitorada via amostragem – em intervalos regulares – das entidades atualmente online. Exemplos incluem usuários online em aplicações distribuídas, ou estações ativas na internet. A monitoração pode ser falha, e algumas entidades podem não aparecer como online em uma ou mais listas, comprometendo assim a acurácia dos dados coletados. Investigações anteriores aplicaram métodos estatísticos para identificar a ocorrência de tais falhas, e usaram limiares para corrigi-las. No presente trabalho, propõe-se investigar a potencial de métodos de aprendizado de máquina para regenerar dados de monitoração coletados via amostragem. Em particular, é avaliado o potencial de se corrigir dados usando técnicas de aprendizado profundo, que podem ser substancialmente melhorada em comparação com os métodos estatísticos existentes.

Palavras-chave: Aprendizado de Máquina. Sistemas Distribuídos. Monitoramento.

ABSTRACT

The process of monitoring online entities (e.g., users, IoT devices, hosts) in large-scale distributed systems is fundamental for understanding their behaviors, analyzing their dynamics, properties, limitations and identifying opportunities for improvement. This process can be relevant in a variety of situations, such as monitoring the availability of internet connections in the last mile and assessing the impact of extreme natural events such as weather, earthquakes, and fires, in the region's infrastructure. In many systems, the online presence of entities can be monitored by sampling – at regular intervals – the entities currently online. Examples include online users in distributed applications, or active stations on the internet. Monitoring can be flawed, and some entities may not appear as online in one or more lists, thus compromising the accuracy of the data collected. Previous investigations have applied statistical methods to identify the occurrence of such failures, and used thresholds to correct them. In the present work, we investigate the potential of machine learning methods to regenerate monitoring data collected via sampling. In particular, we study the potential of correcting data using deep learning techniques, which can be substantially improve results in comparison with existing statistical methods.

Key-words: Machine Learning. Distributed Systems. Monitoring.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

AUC *Area Under the Curve*

CDF *Cumulative distribution function*

EMQ Erro Médio Quadrático

IoT *Internet of Things*

ICMP *Internet Control Message Protocol*

LSTM *Long short-term memory*

MICE *Multivariate Imputation by Chained Equations*

RNA Rede neural artificial

UDP *User Datagram Protocol*

PCA *Principal Component Analysis*

SUMÁRIO

1 INTRODUÇÃO

O processo de monitoramento de entidades *online* (por exemplo, usuários, dispositivos IoT, *hosts*) em sistemas distribuídos de larga escala é fundamental para a compreensão de seus comportamentos, análise de sua dinâmica, propriedades, limitações e identificação de oportunidades de melhoria. Esse processo pode ser relevante em diversas situações, como monitoramento de disponibilidade de conexões de internet na última milha e avaliação do impacto de eventos naturais extremos, como clima (PADMANABHAN et al., 2019), terremotos (MAYER et al., 2021), e incêndios (ANDERSON; BARFORD; BARFORD, 2020), na infraestrutura da região. No contexto de sistemas em rede de larga escala, trabalhos anteriores se concentraram, por exemplo, no monitoramento de usuários *online* em sistemas peer-to-peer (NAIK; KESHAVAMURTHY, 2020; MASINDE; GRAFFI, 2020; SHIRAISHI et al., 2021).

Uma estratégia tipicamente empregada para o monitoramento desses sistemas é o processo de amostragem periódica do conjunto de entidades *online*. Para verificar a disponibilidade de conexão de internet na última milha para clientes residenciais, a amostragem pode ser feita enviando solicitações periódicas de *eco ICMP* via `ping` para um conjunto de endereços IP de um determinado domínio (PADMANABHAN et al., 2019). No caso de sistemas distribuídos baseados no paradigma peer-to-peer, onde pode não existir uma entidade central coordenando todas as entidades *online*, a amostragem pode ser feita através do envio de requisições periódicas para pontos de encontro (entidades de inicialização) para obter listas parciais das entidades *online* (HOSSFELD et al., 2011). No sistemas peer-to-peer BitTorrent, por exemplo, o processo de monitoramento por amostragem significa enviar solicitações periódicas de anúncio aos rastreadores para obtenção das listas de pares (MANSILHA et al., 2011; LAREIDA; HOSSFELD; STILLER, 2017). Para uma série de amostras (ou snapshots)[1] obtida a dinâmica de entrada e saída de usuários, o sistema pode ser recriado; (i) colocando de forma sequencial as capturas e (ii) assumindo que os usuários estiveram *online* durante toda a sequência de captura em que eles aparecem - por esse motivo, é importante equilibrar o intervalo entre os snapshots dependendo da natureza do sistema alvo. Por exemplo, considere tirar dez *snapshots* de um sistema (numerado 1..10), uma amostra por minuto. Considere também que um usuário u foi visto online nos instantâneos 2, 3, 4, 7, 8, 9. Portanto, é razoável supor que u entrou no sistema (ou seja, tornou-se *online*) no instante $t = 2$ (quando o *snapshot* 2 foi tirado) e deixado em $t = 4$, voltou em $t = 7$ e saiu novamente em $t = 9$. O termo *traços de monitoramento* é utilizado para se referir à dinâmica de entrada e saída dos usuários recriada a partir da sequência de *snapshots* tirados do sistema destino, e *sessão online* o período em que um usuário foi visto *online*.

Um desafio do processo de monitoramento associado a estratégias baseadas em

[1] Os termos *amostras* e *snapshot* são usados alternadamente no restante deste trabalho para se referir a um subconjunto de entidades vistas em um sistema de destino em um determinado instante e obtidos usando uma estratégia de monitoramento suportada no sistema.

amostragem é que as entidades podem deixar de aparecer aleatoriamente em um ou mais *snapshot*, embora tenham permanecido *online* durante esse período. Por exemplo, os rastreadores de BitTorrent determinam um tamanho máximo (por exemplo, 200 pares) para cada lista de pares, o que limita o número de usuários que podem ser observados em cada amostra[2]. Pode-se superar essa limitação tirando vários *snapshots* simultaneamente e mesclando-os em um único *snapshot* maior. No entanto, essa alternativa eleva o custo de monitoramento e sobrecarga do sistema. A obtenção de um conjunto de dados de monitoramento de longo prazo também é um desafio, pois exige um sistema de monitoramento que deve ser distribuído em uma escala proporcional à do sistema destino. Por exemplo, os rastreadores de BitTorrent estabelecem um intervalo mínimo entre solicitações consecutivas. Atualmente, a maioria dos rastreadores BitTorrent utilizam protocolos UDP que, aliados à política de intervalo mínimo, tornam o monitoramento suscetível a falhas. Dessa forma, a coleta precisa e abrangente de traços de monitoramento pode ser tornar inviável, especialmente para uma rede de grande porte e por um período prolongado, resultando em rastreamentos com ruídos.

Existem vários motivos para a *regeneração* de traços ruidosos, ou seja, corrigir sessões falhas. Por exemplo, um traço de monitoramento pode ser associado a um *evento único*, por exemplo, comportamento de entrada e saída de usuários ao longo do ano ou durante um desastre natural, para o qual não se pode refazer o monitoramento. Em outros casos, refazer o monitoramento pode ser caro em relação aos recursos computacionais necessários (ou seja, número de monitores necessários e largura de banda disponível). As análises baseadas nos traços defeituosos também podem gerar conclusões imprecisas sobre o comportamento do sistema de destino, incluindo um número inflado de sessões online dos usuários e desvios na duração média da sessão.

Para lidar com traços ruidosos, pode-se utilizar *imputação de dados*. Imputação de dados é um método estatístico para preencher dados faltantes com valores estimados (BUUREN, 2018). Para ilustrar o método, é descrito na Figura 1 um recorte de um traço de monitoramento de verdade (esquerda), ruidoso (meio) e restaurado (direita) de um sistema BitTorrent. Soluções populares de imputação de dados (por exemplo, *Imputação Múltipla por Equações Encadeadas*, MICE (BUUREN; GROOTHUIS-OUDSHOORN, 2011)), não se aplicam aos traços de monitoramento. O motivo deve-se principalmente à natureza distribuída e indireta do sistema de monitoramento, o que impossibilita a distinção entre ausência de dados (uma entidade estava online, mas não foi vista no instantâneo) de vazio (a entidade não foi vista no rastreamento porque estava *offline*). Caso todos os dados vazios sejam considerados ausentes, a relação dados ausentes/dados presentes inviabilizaria o método. Uma solução ingênua é ignorar os dados ausentes, que têm como custo análises imprecisas e potencialmente enganosas derivadas de rastreamento possivelmente falhos.

[2] Tamanho da lista de pares BitTorrent: <https://www.bittorrent.org/beps/bep_0003.html>

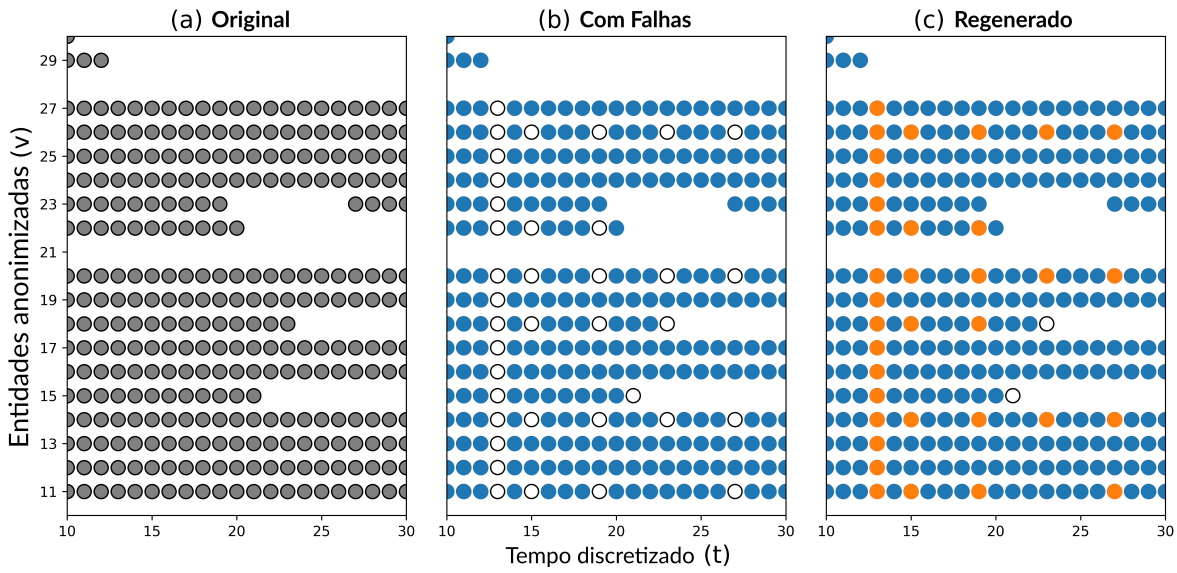


Figura 1 – Recorte de traços de monitoramento coletados de um sistema BitTorrent. Mostra um subconjunto de *snapshots* tirados do sistema (x axis) e usuários vistos em cada instantâneo (y axis). A vista à esquerda mostra o segmento completo, obtido com 27 entidades de monitoramento. O segmento no centro mostra o mesmo segmento porém com apenas 7 monitores (com falha). Por fim, o segmento à direita mostra o traço do centro após a correção dos dados.

Fonte: o autor, 2022

Outra solução é preencher N valores nulos subsequentes entre *snapshots* que um determinado usuário aparece, o que leva à questão de qual valor escolher N; se não for feito metodicamente, o rastreamento resultante pode acabar ainda mais ruidoso. Em um trabalho anterior (CORDEIRO et al., 2021), foi proposto um método baseado no processos de Bernoulli para estimar N e regenerar rastreamentos de sessão do usuário em sistemas distribuídos de grande escala. Em resumo, o método proposto (i) estima a probabilidade de o sistema de monitoramento não conseguir capturar todos os usuários online em cada fatia de tempo de monitoramento e (ii) corrige o rastreamento escolhendo um valor apropriado para N. Apesar dos visíveis avanços alcançados, observa-se uma clara limitação na definição de um N fixo, pois em um cenário real, o comportamento de uma entidade pode variar. Por exemplo, um usuário pode ser mais estável enquanto outros podem entrar e sair com mais frequência.

1.1 Objetivos

O presente trabalho propõe uma nova metodologia de correção de traços de monitoramento usando técnicas de aprendizado profundo. Mas especificamente as seguintes contribuições são apresentadas:

- Nova metodologia baseada em *Deep Learning* para regenerar traços de monitora-

mento de sessões online de usuários, incluindo um método processual e duas topologias de rede neural;

- Uma avaliação experimental da eficácia da nova metodologia, com base em traços de reais obtidos através de um sistema BitTorrent;
- Dois modelos de injeção de falhas artificiais;
- Uma avaliação do impacto da metodologia nas análises de traços regenerados;

Contribuições desta monografia foram reportadas em artigos apresentados para a comunidade científica (PAIM et al., 2021a; PAIM et al., 2022), que podem ser encontrados em anexo. O protótipo implementado no escopo desse trabalho, bem como *scripts* auxiliares, traços de monitoração usados nos experimentos e os resultados obtidos, estão disponíveis sob domínio público para permitir a reprodutibilidade da pesquisa (PAIM et al., 2021b).

1.2 Organização deste trabalho

O restante deste trabalho está organizado como segue. Nos Capítulos 2 e 3 são apresentados, respectivamente, a base teórica e trabalhos relacionados à proposta. No Capítulo 4 são apresentadas a proposta de algoritmo e as topologias de rede neural profunda para corrigir traços de monitoramento ruidosos. No Capítulo 5 é discutido a metodologia e os resultados de uma avaliação experimental baseada em traços reais. Por fim, o Capítulo 6 encerra o trabalho com considerações finais e orientações para trabalhos futuros.

2 MATERIAIS E MÉTODOS

2.1 Monitoramento

O processo de monitoramento consiste na amostragem consecutiva de um sistema expresso na forma de uma matriz $S = [s_{v,t}]$. Na matriz, as linhas representam entidades observadas pelo menos uma vez durante o monitoramento. As colunas representam a amostragem do sistema capturados em intervalos de tempo regulares durante o processo de monitoramento. Cada célula $s_{v,t}$ da matriz S contém um valor binário, onde 1 ("verdadeiro") indica que a entidade v foi vista online no sistema quando a amostra t foi coletado; O valor 0 ("falso") indica que a entidade não foi vista online quando a amostra foi coletada. A Figura 2 representa o seccionamento da matriz de monitoramento no intervalo $[1, 14]$ (rótulos $1, 2 \ldots 14$) para 3 pares monitorados (rótulos 1, 2, 3).

Conforme discutido anteriormente, uma entidade pode estar ausente em um determinado *snapshot* mesmo estando online no sistema naquele instante. Na Figura 2, para representar esse fenômeno e seus impactos, as posições circuladas indicam que a entidade v estava online no instante t. Por exemplo, o círculo em $s_{3,5}$ indica que a entidade 3 estava online no momento 5, embora a amostra coletada não relate que o valor era $s_{3,5}$ =0.

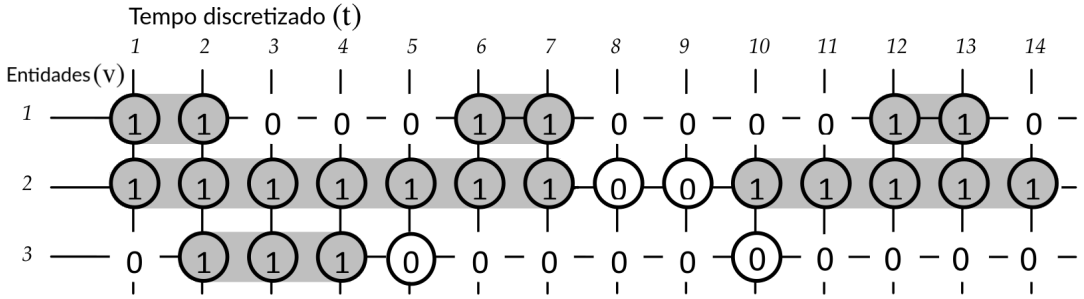


Figura 2 – Matriz de amostras $S = [s_{v,t}]$ indicando a presença (1) ou ausência (0) de usuários v (linhas). Observe que um *snapshot* pode ou não ter uma falha para um ou mais usuários. Por exemplo, os *snapshots* 5 e 10 têm uma falha para o usuário 3, enquanto os *snapshots* 8 e 9 têm uma falha para o usuário 2.

Fonte: adaptado de Cordeiro et al. 2021

A partir das sequências de *snapshots*, pode-se recriar a dinâmica de entrada e saída de entidades no sistema. Assim, uma sessão online é representada como uma sequência de valores positivos. Na Figura 2, cada sessão é destacada em cinza. Observe que a entidade 2 teve duas sessões *online*, de acordo com os *snapshots* coletados: na primeira sessão, ela se juntou em $t = 1$ e deixou em $t = 7$; na segunda, juntou-se $t = 10$ e deixou em $t = 14$. Quando o valor contido na amostra corresponde ao status real da entidade no sistema (ou seja, $s_{v,t}$ é circulado e $s_{v,t} = 1$, significando que a entidade estava *online*). Em contrapartida, quando a amostra não condiz com a realidade há uma falha no *snapshot*. Conforme discutido anteriormente, tal situação pode impactar no número e na duração das sessões *online* das entidades. No exemplo, entidade v_2 teve apenas uma sessão durante

o acompanhamento, com duração de 14 períodos. No entanto, os dados de monitoramento indicam que teve duas sessões, a primeira com duração de sete e a segunda com duração de cinco.

2.2 Redes neurais artificiais

Redes neurais artificiais são estruturas bioinspiradas formadas por conjuntos de neurônios artificias interligados, geralmente dispostos em camadas (MALSBURG, 1986; RUMELHART; HINTON; WILLIAMS,). Essas estruturas são capazes de assimilarem conhecimento através do ajuste de pesos de sinapses distribuídas ao longo de sua estrutura. A Figura 3 representa o modelo de um neurônio artificial, componente base das redes neurais artificiais. Assim como estruturas neurais biológicas, a partir de um dado estímulo

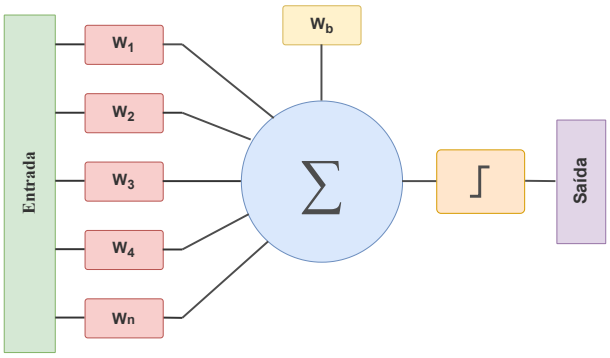


Figura 3 – Diagrama simplificado de um neurônio artificial.

Fonte: adaptado de Gomide et al. 2012

de entrada, as redes neurais artificiais respondem com um sinal de saída. Esse sinal varia seu valor conforme os pesos internos da rede e do conjunto de entrada (redes *feedfoward*). Ajustando-se esses pesos de forma a reduzir a distância entre a saída da rede e a saída desejada (fase de treinamento), é possível utilizar essas estruturas para resolver tarefas envolvendo reconhecimento de padrões, classificação, regressão, etc. Diversas áreas tem aproveitado essa capacidade para automatizar processos envolvendo reconhecimento de padrões e minerar informações (por exemplo, medicina (SHI et al., 2009; SHELL; GREGORY, 2017; ROHANI; ESLAHCHI, 2019), engenharia (OLUDELE; JEGEDE, 2009; LAPEDES; FARBER, 1987), computação (BOUTABA et al., 2018a; PATEL; PATEL, 2020), etc.).

Na literatura, redes neurais *feedfoward* têm sido amplamente divulgadas como aproximadores universais de funções, (por exemplo, (CYBENKO, 1989; HORNIK; STINCHCOMBE; WHITE, 1989; HORNIK, 1991)). Diversos trabalhos envolvendo funções de mapeamento de múltiplas variáveis têm usado redes neurais artificiais como aproximadores (LI, 1996; COSTARELLI; SPIGLER, 2013; KAINEN; KůRKOVá; SANGUINETI, 2013), incluindo aplicações em processos de inferência estatística (WARNER; MISRA,

1996). Embora eficazes em tarefas de aproximação de funções, rede de arquitetura *feed-foward* limitam-se a apenas interpretar os dados de entrada, não levam em consideração o impacto dos resultados anteriormente preditos. Dessa forma, uma convergência ótima pode torna-se difícil de ser atingida caso haja uma dependência temporal entre o conjunto de entradas. Essa limitação atingem em especial problemas envolvendo o uso de dados de entrada obtidos através do deslizamento de janelas (por exemplo, (HOTA; HANDA; SHRIVAS, 2017)).

2.2.1 Redes recorrentes

Uma das limitações das redes neurais *feedfoward* é sua incapacidade de considerar o contexto anterior em suas previsões. Essa limitação impede, por exemplo, que dados de entrada idênticos sejam diferenciados quando sua correta rotulação depende de resultados anteriores. Assim, problemas na continuidade de funções aproximadas, previsão de próximas valores de serie temporais e dificuldade de reconhecer padrões recorrentes, podem ocorrer. Para transpor essa limitação, adaptações nas estruturas tradicionais de neurônios artificiais foram propostas, por exemplo, com o uso de ligações recorrentes entre neurônios na mesma camada ou a eles próprios (RUMELHART; HINTON; WILLIAMS, 1986).

Embora as modificações citadas anteriormente representem um avanço em relação a modelos não recorrentes, problemas relacionados a atualização dos pesos durante a fase de treinamento podem inviabilizar a ótima convergência. No caso de modelos treinados por retro-propagação do erro, (por exemplo, *multilayer-perceptron* (ROSENBLATT, 1958)), pode ocorrer dissipação parcial ou total do gradiente, em especial se o modelo em questão for recorrente. Dessa forma, a eficácia da adoção de modelos recorrentes fica limitada a eventos periódicos de curto médio prazo, não levando em consideração possíveis comportamentos de longo prazo.

2.2.2 Long short-term memory

Para superar as limitações impostas pela dissipação do gradiente em redes recorrentes, uma nova arquitetura de rede neural foi proposta baseada na seletividade do aprendizado. Em síntese, o novo modelo denominado *Long Short-Term Memory* (LSTM) (HOCHREITER; SCHMIDHUBER, 1997) é capaz de, a partir de uma estrutura de portas reguladores internas, aprender series temporárias com dependência de curto e longo prazo, sem porém esbarrar em problemas relacionados às arquiteturas tradicionais. Diversos trabalhos envolvendo o uso de rede (LSTM), em substituição a redes recorrentes simples, foram relatados. Por exemplo, previsão de transações financeiras e valor de ativos (YAN; WEIHAN; CHANG, 2021; WANG; WU; CHEN, 2021). A Figura 4 exibe o diagrama de uma célula recorrente (LSTM).

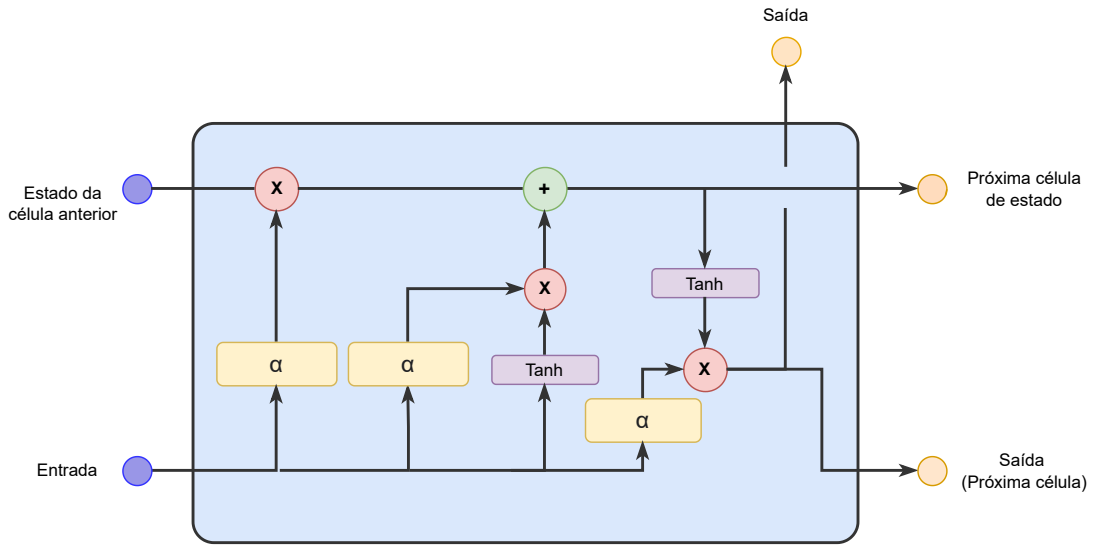


Figura 4 – Diagrama de célula Long Short-Term Memory (LSTM)

Fonte: adaptado de Varsamopoulos et al. 2018

2.2.3 Funções de ativação

Como visto anteriormente, redes neurais artificiais são compostos por camadas de neurônios conectados. O processamento, em cada neurônio, ocorrem por meio do produto euclidiano dos pesos neurais pelo vetor de entrada e são repassados as camadas subjacente no sentido da camada de entrada para a camada de saída. No entanto, a adoção dessas estrutura simples limita a rede neurais a funcionar como uma serie de transformações linear (CHOLLET, 2017). Para superar essa limitação, a saída de um neurônio padrão é representada como composição de uma função linear e uma função escalar arbitrária, onde a função linear é o produto euclidiano dos pesos neurais e vetor de entrada, e a função arbitrária é um hiper-parâmetro chamado função de ativação (NADER; AZAR, 2021).

2.2.3.1 Função sigmoide

A sigmoide, as vezes referida como logística na literatura, é uma função diferenciável no domínio real . Onde $F(x) \in [0, 1]$, sendo $x \in R$ com derivada positiva em todo seu domínio (NWANKPA et al., 2018a). É amplamente citado na literatura como função de ativação para redes neurais, por exemplo em aplicações envolvendo classificação binária (GUO et al., 2017; YIN et al., 2017). Principalmente devido sua propriedade de manter seu valor de saída no intervalo $I \in [0, 1]$. A Equação 2.1 define a função sigmoide.

$$F(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

2.2.3.2 Função tangente hiperbólica (tanh)

A tangente hiperbólica é uma função diferenciável no domínio real. Onde $F(x) \in [-1, 1]$, sendo $x \in R$ com derivada contínua em todo seu domínio (NWANKPA et al., 2018a). É utilizada em redes neurais como em células LSTM (HOCHREITER; SCHMIDHUBER, 1997). A Equação 2.2 define a função tangente hiperbólica.

$$F(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.2}$$

2.3 Indicadores de qualidade de modelos

A eficácia de classificadores binários podem ser mensurada através de métricas de desempenho específicas. Essas avaliações incluem tipicamente o uso de matriz de confusão como indicador de qualidade do modelo (HOSSIN; M.N, 2015). Em análise preditiva, esse instrumento possibilita a identificação de problemas de classificação e facilita a comparação entre modelos distintos. A estrutura base de uma matriz de confusão (classificação binária) é composta por quatro grupos de classificação, sendo dois relacionado a acertos do modelo (verdadeiro positivo (VP), verdadeiro negativo (VN)) e dois relacionados a erros (falso positivo (FP), falso Negativo (FN)) (HOSSIN; M.N, 2015). A Figura 5 exibe a estrutura de uma matriz de confusão. Em destaque estão cada uma das quatro classes possíveis para uma classificação binária.

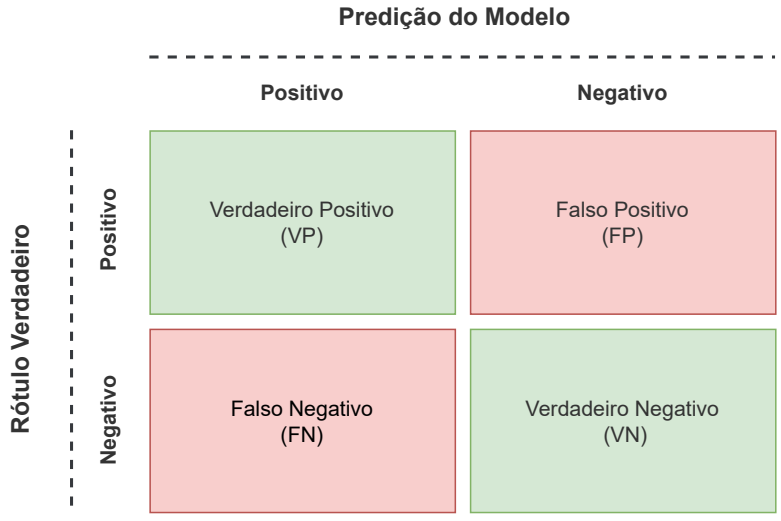


Figura 5 – Estrutura de matriz de confusão para classificação binária. As cores vermelho e verde representam respectivamente erros e acertos do modelo.

Fonte: adaptado de Hossin et al. 2015

2.3.1 Acurácia

A qualidade de um modelo preditivo depende diretamente de sua capacidade de rotular de forma correta os diferentes conjunto de entradas. A acurácia é uma métrica relevante nesse sentido, pois fornece uma maneira simples de quantificar a relação entre acertos e total de predições. Em síntese, a acurácia é o indicador que permite quantificar o quão próximo os rótulos preditos pelo modelo estão dos rótulos reais. O processo de calculo da acurácia levam em consideração o tamanho do conjunto de entradas e o número de entradas corretamente rotuladas (verdadeiro positivo, verdadeiro negativo) (HOSSIN; M.N, 2015). A Equação 2.3 define acurácia em termos das classes definidas na matriz de confusão, sendo o numerador e denominador respectivamente o número total de acertos e número total de previsões feitas.

$$A = \frac{(VP + VN)}{VP + VN + FN + FP} \tag{2.3}$$

2.3.2 Precisão

A precisão é um indicador de qualidade importante na avaliação de modelos preditivos com rotulação binária. É especialmente útil quando o crescimento da taxa de falsos positivos é considerada mais prejudiciais que a de falsos negativos. Por exemplo, casos que incluem a completação de matrizes de tráfego em redes (CHENG et al., 2017), onde o objetivo é obter um dados próximo do realístico, porém não perder padrões de falha que possam indicar possíveis problemas. A Equação 2.4 define precisão em termos das classes definidas na matriz de confusão, onde o denominador é o número total de previsões positivas.

$$P = \frac{VP}{VP + FP} \tag{2.4}$$

2.3.3 Recall

A efetividade de modelos preditivos em situações onde a qualidade depende, dentre outros coisas, da correta rotulação de amostras positivas, pode ser mensurada através do *recall* . Em síntese, essa métrica avalia, dentro das amostras verdadeiramente positivas, quantas foram de fato preditas como positivas pelo modelo. A Equação 2.5 (KUBAT, 2015) define *recall* em termos das classes definidas na matriz de confusão, sendo o denominador, o valor total de amostras verdadeiramente positivas no conjunto avaliado.

$$R = \frac{VP}{VP + FN} \tag{2.5}$$

2.3.4 F1-score

Como visto anteriormente, o processo efetivo de avaliação de modelos preditos binário deve levar em consideração mais de um indicador de desempenho, em especial, se o impacto das taxas de falso positivo ou falso negativo não forem iguais. O indicador F1-Score é uma tentativa de estabelecer uma métrica de avaliação que leve em consideração tanto a sensibilidade quanto a precisão do modelo. A Equação 2.6 define F-Score como a média harmônica entre precisão e recall, onde $\beta \in [0, \infty)]$ é um parâmetro, geralmente $\beta = 1$ para F1-score. Esse parâmetro determina a importância de cada métrica componente (KUBAT, 2015).

$$F = \frac{(\beta^2 + 1) \times Precisão \times Recall}{(\beta^2 \times Precisão) + Recall} \tag{2.6}$$

3 TRABALHOS RELACIONADOS

A ocorrência de falhas no processo de monitoramento de sistemas distribuídos é comum e deve ser levada em consideração no projeto (MANSILHA et al., 2011). Em geral, os sistemas de monitoramento se concentraram em atingir objetivos fundamentais como cobertura, detalhamento, frequência e duração (MANSILHA et al., 2011). A cobertura é relevante do ponto de vista estatístico, para que a amostra coletada represente de forma adequada a população investigada. O detalhamento afeta o nível de estratificação permitido pelos dados. A frequência altera a granularidade de detecção de fenômenos. A duração impacta na chance da observação de fenômenos de longo prazo. Como esperado, o custo e a probabilidade de falha para a obtenção de um monitoramento com maior cobertura, detalhamento, frequência e duração, aumentam. Uma abordagem típica para fornecer tolerância a falhas é utilizar redundância de recursos, geralmente redundância de agentes de monitoramento (JUNIOR; CORDEIRO; GASPARY, 2018). Por exemplo, o número de monitores usados para coletar *snapshots* pode ser supercompensado (ZHANG et al., 2011).

A adoção de técnicas de regeneração de traços podem complementar as técnicas de prevenção e tolerância a falhas. A aplicação de técnicas a priori só tem efeito durante o processo de monitoramento. A adoção de medidas de regeneração de traços podem melhorar significativamente a qualidade dos dados obtidos, sem aumentar os custos do processo de monitoramento. Vale destacar que a regeneração é a única opção de aprimoramento que pode ser utilizado após o processo de monitoramento.

A imputação de dados é uma área da matemática e da estatística com uma longa trajetória de pesquisa (RUBIN, 1996; BUUREN, 2018). *Multiple Imputation by Chained Equations* (MICE) (BUUREN; GROOTHUIS-OUDSHOORN, 2011) e *MissForest* (STEKHOVEN; BüHLMANN, 2011) são soluções populares aplicadas a áreas como medicina (WALJEE et al., 2013), por exemplo. A imputação de dados também tem sido estudada no contexto de redes. A recuperação de falhas de dados com base na completação de tensor, mineração de dados e análise de componentes principais (PCA) foi estudada em várias investigações anteriores (CHENG et al., 2017; Xie et al., 2018; Xie et al., 2019). Em geral, essas soluções assumem que os dados ausentes podem ser facilmente reconhecidos. Este não é o caso do monitoramento de traços de entidades online em sistemas de rede em grande escala. Em sistemas de larga escala não é possível garantir a existência de uma entidade central, dessa forma faz-se necessário o aproveitamento de padrões indireto para obtenção de dados mais acurados.

Recentemente, Cordeiro et al. 2021 propõem uma solução probabilística para o problema de perda de dados de sessão do usuário em traços de monitoramento de sistemas em rede. Em síntese, o processo consiste em modelar a probabilidade de falha durante uma sessão de usuário como um processo de Bernoulli para distinguir um valor nulo entre ausência do usuário (o usuário estava *offline*) e falha de monitoramento (o usuário estava

online, mas não apareceu no *snapshot*). A partir desse modelo, é possível estimar o número de valores nulos consecutivos máximos (ou seja, *gap* máximo) que devem ser considerados uma falha de monitoramento e, portanto, ser regenerados. Embora represente um grande avanço em relação ao uso de dados brutos, o desempenho alcançado pela técnica é limitado devido a seleção de um único valor de intervalo máximo para decidir sobre todos os casos presentes dos dados. O principal desafio relacionado a técnica anterior reside na dificuldade de estabelecer o impacto de cada vizinho na probabilidade de falha de uma entidade em um determinado *snapshot*, de forma a obter uma curva de correção. Dessa forma, métodos que permitem modelar a curva de relevância de cada vizinhança do ponto de inferência com base em dados do mundo real podem ser mais promissores do que métodos puramente estatísticos. Passos em direção a regeneração de traços baseados na modelagem dessas curvas através do uso de redes neurais foram dados pelo autor desse trabalho em Paim et al. (2021, 2022).

4 PROBLEMA E SOLUÇÃO PROPOSTA

Como visto nos capítulos anteriores, a regeneração de traços de sistemas de larga escala é relevante em diversas áreas como gerenciamento de redes, tanto do ponto de vista da análise e diagnóstico, como também otimização. Nesse capitulo são apresentadas duas proposta de solução para regeneração de traços de monitoramento baseadas em aprendizado profundo (*Deep learning*). Na Seção 4.1 é apresentada a descrição formal do modelo, bem como o embasamento matemático necessário ao entendimento do problema. Em seguida, na Seção 4.2 é apresentado e discutido um protótipo de algoritmo de correção. Por fim, a Seção 4.3 propõe dois modelos de rede neural.

4.1 Modelo

Seja $G = \langle V, B, E \rangle$ um grafo onde:

- V é o conjunto de todas entidades principais $v \in V = \{1, 2, \ldots, |V|\}$, por exemplo, usuários do sistemas peer-to-peer;
- B é o conjunto de entidades *bootstrap* (ou seja, entidades que servem como pontos de encontro para entidades principais no sistema) $b \in B = \{1, 2, \ldots, |B|\}$, por exemplo, rastreadores BitTorrent;
- E é o conjunto de arestas $(i, j) \in (V \times B)$,que indicam o conhecimento da presença online de uma entidade principal i ; por um *bootstrap* j, por exemplo, lista de pares de um rastreador BitTorrent;

O processo de registro da dinâmica de entrada e saída de todas as entidades principais V consiste na obtenção, em intervalos regulares, do estados do sistema observado. Assim, para obter esses estados, define-se G^t como o grafo contento todas entidades V presentes em t, onde t é um instante de tempo discretizado $t \in T = \{1, 2, \ldots, |T|\}$. Nesse processo, as amostras são coletadas em intervalos regulares Δ (por exemplo, 15 minutos), o que resulta em $t \in [\Delta t, \Delta t + 1[$ recorte de tempo (por exemplo, $[45, 60[$ minutos para t =3 e Δ =15 minutos).

Os traços de monitoramento de um sistema representado pelo modelo acima pode ser dado por uma matriz $S = [s_{v,t}]$ de tamanho $|V| \times |T|$ (como exemplificado na Figura 2). Nessa matriz, $s_{v,t} = 1$ se o nodo $v \in V$ foi visto pelo menos uma vez no instante $t \in T$ (ou seja, para um par v, $\sum_{j=1}^{|B|} E_{v,j}^t \geq 1$), e $s_{v,t} = 0$ por outro lado. Para valores positivos, assume-se que representa a presença do nó no sistema. No entanto, essa prática pode não representar de forma precisa a realidade, pois uma entidade pode permanecer conhecida por *bootstrap* como online após uma desconexão planejada. No entanto, devido ao baixo impacto dessa limitação em monitoramentos de longo prazo, o escopo deste trabalho limita-se apenas a ignorar essa possível falha e deixa-se para trabalhos futuros

investigações mais aprofundadas sobre essa limitação. Observe-se também que um valor 0 pode significar uma ausência de entidade v no instante t, ou uma falha no sistema de monitoramento em um determinado instante t. Nas próximas seções são propostas técnicas de regeneração desses traços baseados em técnicas de aprendizagem de máquinas.

4.2 Algoritmo de correção

O processo de regeneração de traços de monitoramento de sistemas *online* é mostrado no algoritmo 1. A entrada é uma matriz $S = [s_{v,t}]$ representando o traço a ser corrigido, um limite de correção α ($\alpha \in [0, 1]$), e uma janela de deslizante W. A saída é uma matriz $S' = [s_{v,t}]$ que representa o traço regenerado.

Algoritmo 1: Correção de rastreamento usando redes neurais

```
1  Entrada S = [s_{v,t}], α, W
2  Saída S' = [s'_{v,t}]
3  begin
4      S' ← S ;
5      V ← conjunto de entidades de S;
6      T ← conjunto de recortes de tempo de S;
7      for v ∈ V do
8          for t ∈ T do
9              if s_{v,t} == 0 then
10                 p_{v,t} ← g(X^{v,t});
11                 if p_{v,t} >= α then
12                     s'_{v,t} ← 1;
13                 end
14             end
15         end
16     end
17     return S'
18 end
19 Fonte: o autor, 2022
```

De acordo com Cordeiro et al. 2021, a probabilidade de uma determinada amostra nula $s_{v,t} = 0$ sendo um falso negativo, para um usuário v, diminui com a quantidade de amostras nulas em sua vizinhança. Com base nessa observação, nosso algoritmo avalia os arredores de cada $s_{v,t} \in S$. Usando uma rede neural, ele obtém a probabilidade de que uma determinada amostra nula seja um falso negativo – ou seja, o usuário está presente no sistema, mas não apareceu na amostra. Se a probabilidade for maior que o limite α, a amostras $s_{v,t}$ é corrigido – ou seja, o valor de $s_{v,t}$ é invertido para 1.

Para simplificar a notação e o treinamento do modelo, para cada $s_{v,t} \in S$ definimos um vetor $X^{v,t} = [x_k^{v,t}]$, $k \in \{t - c, t - c + 1, \ldots, t, \ldots, t + c - 1, t + c\}$, onde seu centro $c = \lfloor W/2 \rfloor$, e cada elemento $x_k^{v,t} \in X^{v,t}$ corresponde a um elemento $x_{v,z} \in S$ usando a

relação $k = z$. A função de previsão de probabilidade de falha de uma janela $X^{v,t}$ assim pode ser formulada como:

$$p_{v,t} = g(s_{v,t-c}, s_{v,t-c+1}, \ldots, s_{v,t}, \ldots, s_{v,t+c-1}, s_{v,t+c}) \tag{4.1}$$

onde $g(\cdot)$ é uma função da janela centrada na amostra $s_{v,t}$ de interesse.

O algoritmo inicialmente cria uma cópia de S para S' e inicializa os conjuntos V e T (linhas 4-6). Em seguida, o algoritmo itera pelo conjunto de amostra $s_{v,t} \in S$ para cada usuário $v \in V$ (linha 7) e cada fatia de tempo $t \in T$ (linha 8). Se o elemento indicar *online* (ou seja, amostra "positiva"), nenhuma alteração é realizada em S'. É importante lembrar que uma situação oposta (ou seja, amostra nula) pode significar que a entidade está *offline* ou houve uma falha do sistema de monitoramento em capturar sua presença *online*. Nesses casos, o valor é avaliado considerando a janela ao seu redor por um rede neural previamente treinado (linha 10). Essa RNA retorna a probabilidade $p_{v,t} \in R$ tal que $0 \leq p_{v,t} \leq 1$ do valor nulo encontrado na posição central da janela corresponde a um falha de monitoramento. Dado um limite de configuração mais baixo α, o valor da posição central da janela $X^{v,t}$ é alterado para 1 se $p_{v,t} \geq \alpha$; ou mantido como 0 se $p_{v,t} < \alpha$ (linhas 11 e 12). Em resumo, a correção realizada em nossa proposta é mais agressiva quanto menor for o valor de α.

4.3 Topologias RNA propostas

Para o processo de regeneração de seções de monitoramento, este trabalho propõe o uso de duas de redes neurais genéricas aqui denominadas Densa e LSTM. A Figura 6 ilustra a estrutura das duas redes neurais, note que ambos compartilham as mesmas estruturas de entrada, intermediária e saída. Nesse contexto, a principal diferença entre as topologias, chamadas aqui genericamente de Densa e a LSTM está em suas camadas intermediárias (também conhecidas como camadas ocultas). A topologia Densa é composta por 20 neurônios completamente conectados, enquanto a LSTM consiste em 20 células recorrentes LSTM padrão (HOCHREITER; SCHMIDHUBER, 1997).

A camada de entrada é uma janela deslizante com tamanho W e centrado em $s_{v,t}$. A camada de saída é um valor referente à probabilidade p usando uma função de ativação sigmoide. As camadas intermediárias são organizadas em uma ou mais *abrangentes*, cada uma contendo uma camada Densa (ou uma camada LSTM na primeira disposição da topologia LSTM) e uma camada *Dropout*. Enquanto as camadas Densa/LSTM assimilam padrões, as camadas *Dropout* evitam o *overfitting* (SRIVASTAVA et al., 2014).

Ambas topologias foram propostas com o objetivo de mapear um domínio multivariado em uma saída de intervalo $[0, 1]$, cuja função representa a probabilidade de uma determinada entrada ser um falso negativo, levando em consideração o status (*online*, *offline*) de uma vizinhança próxima. Devido à dificuldade em estimar o número de camadas

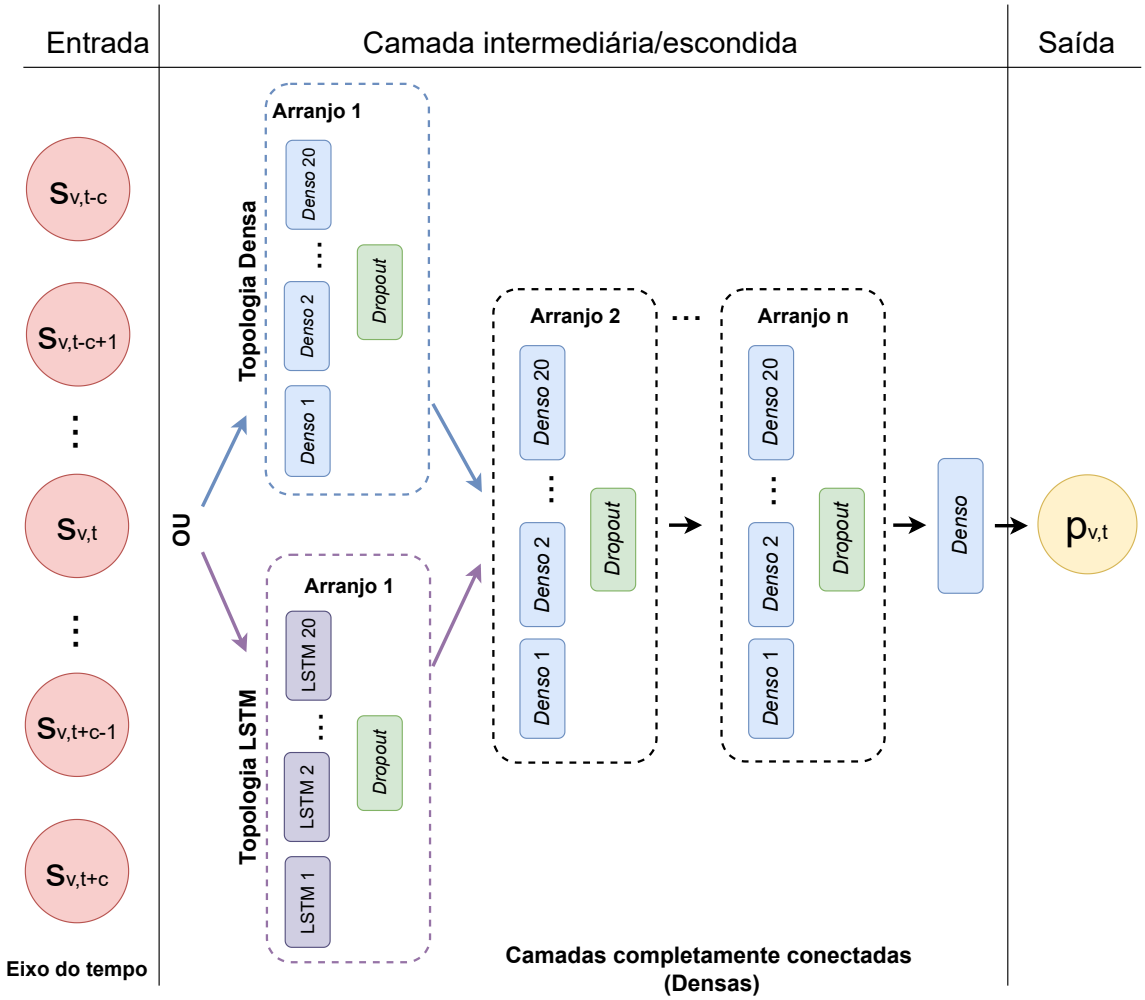


Figura 6 – Topologias da Rede Neural Artificial (RNA): primeiro arranjo pode ser usado tanto um Dense quanto um LSTM como primeira camada.

Fonte: o autor, 2022

necessárias para aproximar a função, optou-se por manter uma estrutura intermediária composta por *arrays* em quantidades variadas. Além da estrutura completamente densa, os impactos da recorrência de falhas nos resultados também foram explorados. A primeira camada densa da topologia densa foi substituído por células recorrentes e entradas em lote sequenciais. Para evitar problemas comuns relacionados a redes recorrentes, como dissipação do gradiente (KOLEN; KREMER, 2001), células recorrentes LSTM, cujas propriedades minimizam a probabilidade de tais problemas, foram escolhidas (HOCHREITER; SCHMIDHUBER, 1997).

5 AVALIAÇÃO

5.1 Metodologia

Para propósitos de avaliação, as topologias RNAs foram implementadas usando a API Keras do TensorFlow (ABADI et al., 2015) (v2.4.1). As classes Dense, LSTM e *Dropout* foram usadas para implementar as redes neurais. A classe Dense de*arrays* é configurada com 20 unidades de neurônios por camada, função de ativação linear entre camadas ocultas. A última instância da classe *Dense* é configurada com uma unidade de neurônio, uma função de ativação logística para estimar probabilidades (NWANKPA et al., 2018b) e mantenha a saída no intervalo $[0, 1]$. A classe *Dropout* é configurada com uma taxa de decaimento de 20% na primeira instância do *array* e 50% nos seguintes casos, conforme recomendado no trabalho original sobre a camada *Dropout* (SRIVASTAVA et al., 2014). Como método para otimização estocástica de funções objetivo, usamos a Adam (KINGMA; BA, 2015) implementado na biblioteca Keras. O algoritmo é configurado seguindo as sugestões desse estudo: α_{adam} =0.0001, β_1 =0.9 e β_2 =0.999. Para facilitar a convergência do treinamento, nos casos em que há dois resultados diferentes possíveis uma determinada janela de entrada, escolhemos uma resposta de amostra nula, que é a escolha mais conservadora. Prevemos como investigação futura e outras configurações para o processo de correção. O código fonte das RNAs, os resultados dos experimento e as topologia implementadas estão disponíveis no GitHub (PAIM et al., 2021c).

O processo de avaliação das topologias RNAs foi dividido em quatro etapas: análise de treinamento (Seção 5.2), análise de sensibilidade paramétrica (Seção 5.3), comparação com o estado da arte (Seção 5.4), e estudo de caso (Seção 5.5).

Tabela 1 – Propriedades dos traços usados no estudo (Δ =15 minutos).

Arquivo	Duração total $\|T\|$	#Pares $\|V\|$	Dataset	Dataset (%) utilizado	Snapshots verdadeiros
S1.bz2	≈2,5 meses 7,196	12,504	$S1$	100%	526,303
S2.bz2	≈2,5 meses 7,196	5,956	$S2$	25%	46,181
S3.bz2	≈6 meses 17,323	258,582	$S3$	100%	15,181,100

Fonte: o autor, 2022

A avaliação é baseada em alguns dos *snapshots* de monitoramento do BitTorrent publicados por Cordeiro et al. (CORDEIRO et al., 2021) e resumido na Tabela 1. Os traços S1 e S2 foram escolhidos porque incluem pequenos enxames de BitTorrent ($|V| <$ *lista de pares* enviado por rastreadores). Nesses casos, é possível demonstrar matematicamente que o traço está completo com probabilidade muito alta (seguindo a metodologia de captura de *snapshots* de Hoßfeld et al. (HOSSFELD et al., 2011)) e, portanto, pode ser usado como *totalmente completos*. Para agilizar o processo de avaliação foram usados

apenas 25% do arquivo S2, pois os 75% restantes não devem impactar significativamente os resultados. Na última etapa, o processo foi repetido para *snapshots* do monitoramento de um enxame (traço S3) com população média superior à capacidade de monitoramento.

Tabela 2 – Parâmetros e valores utilizados em cada etapa desta avaliação.

	Parâmetros		Valores
Treinamento	Conjunto de validação		$S1$
	Conjunto de treinamento		$S2$
	Topologias		{Densa (DE), LSTM (LS)}
	Arranjos	A	3
	Tamanho de janela	W	11
	Épocas de treinamento		{1, 2, . . ., 20}
Avaliação paramétrica	Traços de avaliação		$S1$
	Traços de treinamento		$S2$
	Topologias		{Densa (DE), LSTM (LS)}
	Arranjos	A	{1, 3, 5}
	Tamanho de janela	W	{7, 11, 21}
	Limiar	α	{0,50, 0,75, 0,95}
	Falha probabilística injetada	F_{prob}	{1, 10, 50}%
	Falha real de monitorando	F_{mon}	{11(6,77%), 17(20,04%), 20(39,24%)}
	Épocas de treinamento		15
	Ensaios		10
Comparação	Traços de avaliação		$S1$
	Traços de treinamento		$S2$
	Topologias		{LSTM (LS), Probabilístico (PR)}
	Arranjo	A	3
	Tamanho de janela	W	11
	Limiar	α	0,75
	Falha probabilística injetada	F_{prob}	{1, 2, 5, 10, 15, 20, 25, 40, 50}%
	Falha real de monitorando	F_{mon}	{7 0,16%), 10(2,35%), 11(6,77%), 15(6,95%), 16(7,32%), 17(20,04%), 18(21,07%), 19(24,02%), 20(39%)}
	Épocas de treinamento		15
	Ensaios		10
Estudo de caso	Traços de avaliação		{$S1$, $S3$}
	Traços de treinamento		$S2$
	Topologia		LSTM (LS)
	Modelo de falha injetado		F_{mon} =20
	Arranjo	A	3
	Tamanho de janela	W	11
	Limiar	α	0,75
	Épocas de treinamento		15

Fonte: o autor et al. 2022

Vários níveis de ruído foram considerados durante o processo de avaliação. Os Traços utilizados foram tomados como verdadeiro e neles foram injetadas falhas seguindo dois modelos de injeção de falhas F: probabilístico (F_{prob}) e monitoramento (F_{mon}). O primeiro oferece um controle mais refinado, enquanto o segundo é mais realista e segue a abordagem clássica de Hoßfeld et al. (HOSSFELD et al., 2011). O modelo de falha probabilística usa uma distribuição uniforme como a probabilidade de injeção de falha.

Por exemplo, para F_{prob} =1%, cada amostra tem 1% de chance de ser "defeituosa" (1 virada para 0 para alguns usuários). No caso de uma falha injetada, uma amostra positiva é alterada para nula para um usuário específico.

Para emular falhas no rastreamento (F_{mon}), os monitores foram classificados de acordo com o número de *snapshots* coletados e selecionado M em ordem crescente. Para maior clareza, assumimos F_{mon} =27-M como complemento de um máximo de 27 monitores, o que resulta em 0% de falhas, e também mostra a porcentagem correspondente de dados ausentes. Por exemplo, para F_{mon} =20 (40%) em S1, temos 40% de dados ausentes, pois consideramos apenas os sete piores monitores que foram capazes de coletar 319.763 *snapshots* de um total de 526.303 no rastreamento de verdade do S1.

Depois da inserção de falhas, os traços são submetidos para regeneração. Por fim, os traço resultante são comparados com os originais (Completos) e com os traço com falha para construir a matriz de confusão e extrair métricas de desempenho. Para propósito de avaliação, métricas tradicionais na literatura de aprendizado de máquina foram utilizadas (BOUTABA et al., 2018b; CHENG et al., 2017): acurácia, precisão, *recall* e F1.

Na Tabela 2 estão resumidas as parametrizações utilizadas durante a avaliação, composta por 582 ensaios. Na análise de sensibilidade dos parâmetros e na etapa de comparação foram realizadas dez repetições para cada combinação de valores.

5.2 Análise de ajuste

Para evitar o subajuste ou sobreajuste das RNAs, o impacto do número de épocas de treinamento na qualidade da resposta da rede neural foi avaliado. Esta etapa utilizou as duas topologias, número de arranjos A =3, e tamanho da janela W =11.

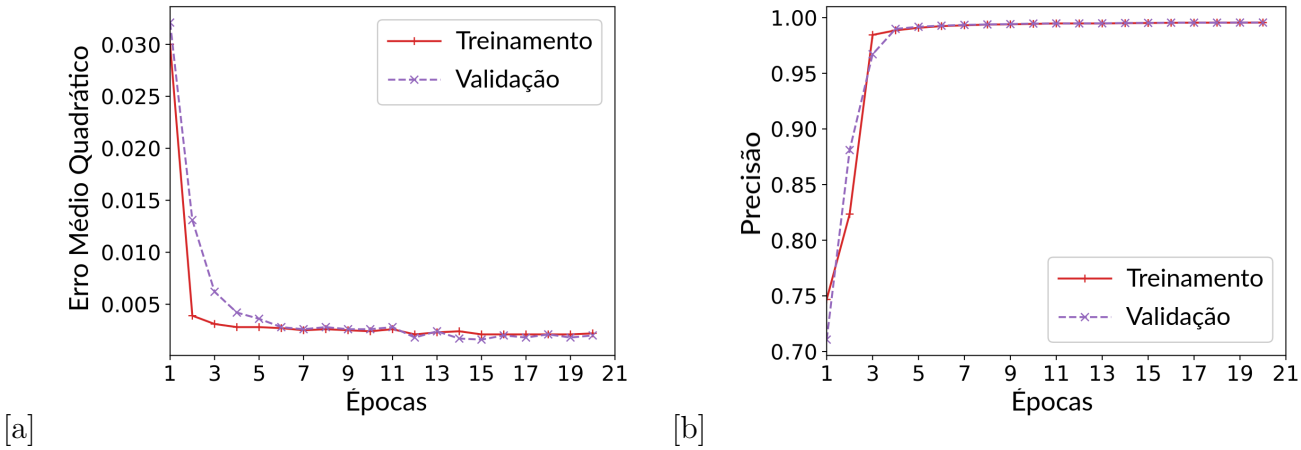
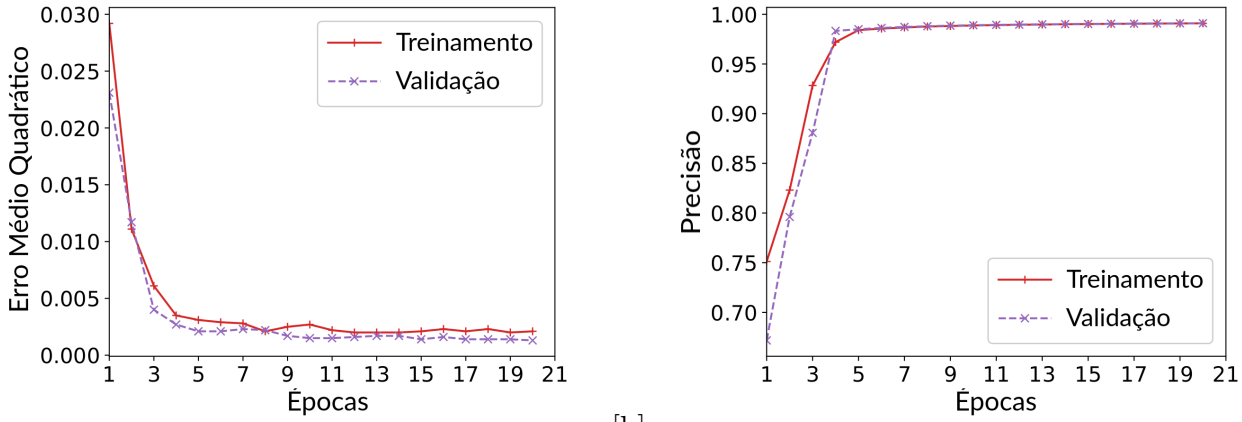


Figura 7 – Impacto do número de épocas em (a) erro médio e em (b) precisão (topologia=Dense, arranjos A =3, tamanho da janela W =11).

Fonte: O autor, 2022

A Figura 7 mostra, respectivamente, a evolução do Erro Quadrado Médio (EMQ) e a precisão das RNAs em função do número de épocas de treinamento para a topologia



[a] [b]

Figura 8 – Impacto do número de épocas em (a) erro médio e em (b) precisão (topologia=LSTM, arranjos A =3, tamanho da janela W =11).

Fonte: O autor, 2022

Densa. De forma similar, as Figura 8, mostram, respectivamente, a evolução do EMQ e a precisão da RNA em função do número de épocas de treinamento para LSTM. Pode-se notar que as curvas tendem à estabilidade - o que indica a ausência de sub-ajuste e sobre-ajuste. Além disso, vale destacar que as curvas de treinamento e validação estão próximas - uma indicação adicional da ausência de subajuste. Como resultado desta etapa, adotou-se 15 épocas de treinamento para todas as avaliações restantes.

5.3 Análise de sensibilidade do parâmetro

As Figuras 9 e 10 mostram os resultados da análise de sensibilidade dos parâmetros, em relação ao número de arranjos dentro das camadas ocultas, limite de correção inferior α e tamanho da janela. Em cada figura, para cada valor do parâmetro analisado, apresentamos um conjunto de pares de barras – uma para cada métrica, com uma barra para LSTM (LS) e outra para Dense (DE). Cada barra exibe a média e o desvio padrão de dez tentativas.

Duas observações principais em relação ao Figura 10. Primeiro, a sensibilidade aos parâmetros e valores escolhidos é relativamente pequena. Uma hipótese é que a possível correção já esteja próxima de um limite superior. Segundo, como esperado, a precisão aumenta à medida que usamos um limite mais conservador, ainda que minimamente; observa-se um comportamento inverso para recall.

Na Figura 11 os resultados das topologias LSTM e Dense, levando em consideração múltiplas falhas probabilísticas injetadas (esquerda) e de monitores (direita). Por brevidade, apenas nove casos são mostrados (três para cada modelo de falha). O mesmo comportamento pode ser observado em todos os dezoito casos (9 para cada modelo de falha), que pode ser encontrado no repositório GitHub (PAIM et al., 2021c). Observe que a topologia LSTM apresenta uma vantagem sobre a topologia Dense, embora mínima,

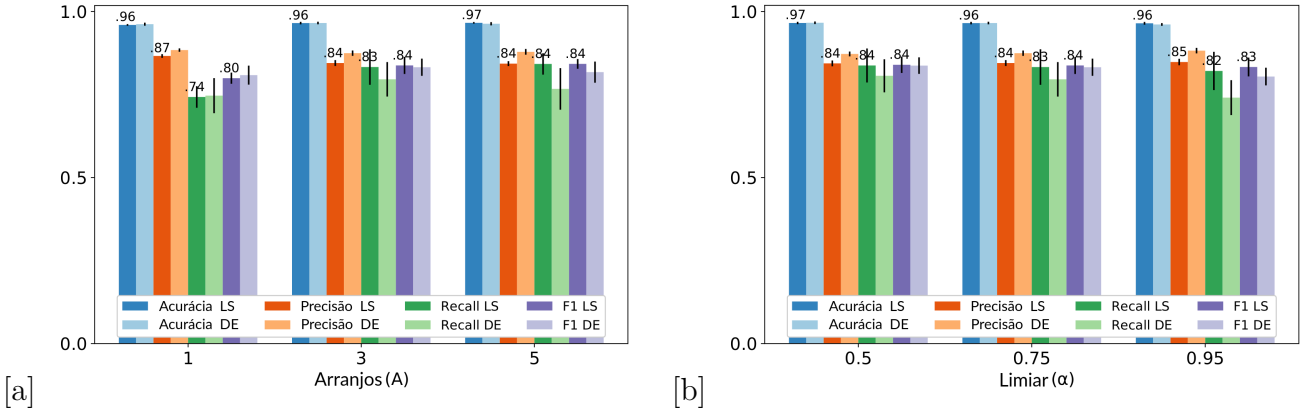


Figura 9 – Sensibilidade dos parâmetros das topologias LSTM (LS) e Dense (DE) em relação a (a) número de arranjos $A \in \{1, 3, 5\}$ e (b) limite $\alpha \in \{0{,}50, 0{,}75, 0{,}95\}$ com falha injetada probabilística uniforme F_{prob} =10%.

Fonte: O autor, 2022

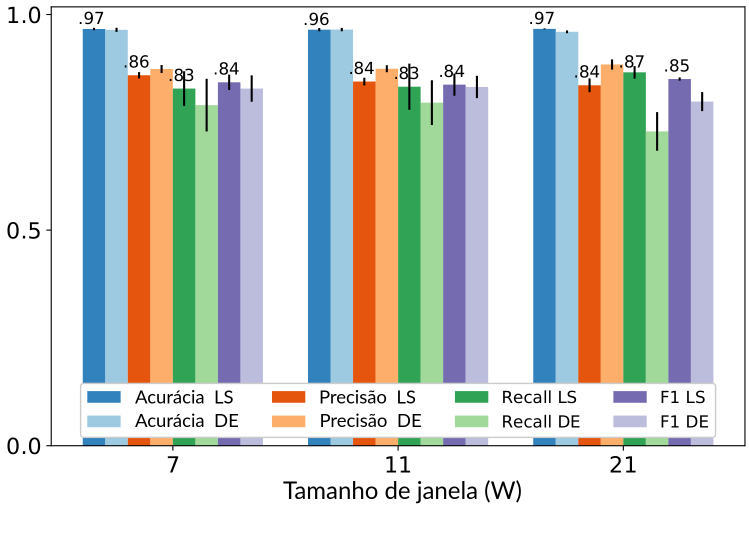


Figura 10 – Sensibilidade dos parâmetros das topologias LSTM (LS) e Dense (DE) em relação ao tamanho da janela $W \in \{7, 11, 21\}$, com falha injetada probabilística uniforme F_{prob} =10%.

Fonte: O autor, 2022

e também mais estável em geral. Portanto, as avaliações a seguir são concentradas na topologia LSTM.

5.4 Comparação com o estado da arte

Para validar a melhoria obtida com a técnica proposta neste trabalho, comparou-se os resultados obtidos com trabalhos existentes na literatura, mais especificamente com a metodologia estatística (CORDEIRO et al., 2021). Para esse propósito, usou-se um limite de correção superior a 0,75. Esse valor foi escolhido porque aparenta ser o cenário mais favorável entre os valores avaliados nesse estudo.

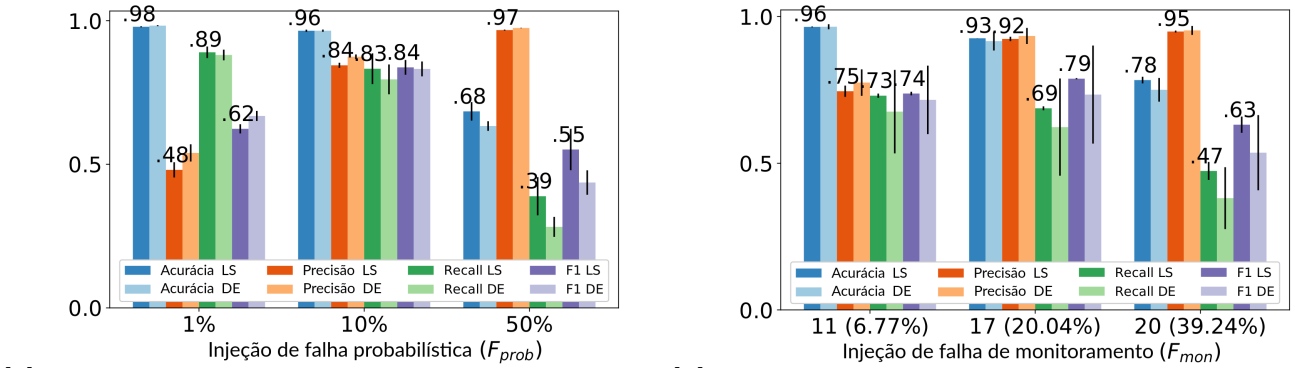


Figura 11 – Topologias LSTM (LS) e Dense (DE) para diferentes falhas injetadas.

Fonte: O autor, 2022

A Figura 12 mostra os resultados para LSTM e o método probabilístico considerando múltiplas falhas injetadas probabilísticas. Resultados semelhantes foram obtidos usando a topologia Densa, e estão disponíveis no repositório GitHub (PAIM et al., 2021c). Oito barras são exibidas para cada probabilidade de falha injetada (F_{prob}). Há um par de barras para cada métrica: uma para LSTM (sufixo LS) e outra para o método probabilístico (sufixo PR). Cada barra mostra o valor médio e o desvio padrão de 10 tentativas. Observe que os resultados do algoritmo probabilístico variam conforme usamos diferentes sementes aleatórias para gerar cada um dos traços de falha injetados usados em cada execução. Para maior clareza, os gráfico exibem apenas os valores das barras referentes à técnica proposta baseada em LSTM (LS). Na Figura 13 os resultados de LSTM (LS) e método probabilístico (PR) são apresentados, mas para várias falhas injetadas de monitoramento (F_{mon}). Observe que os resultados do algoritmo probabilístico não variam neste caso. Há quatro aspectos dignos de nota a partir dos resultados mostrados na Figura 12 e

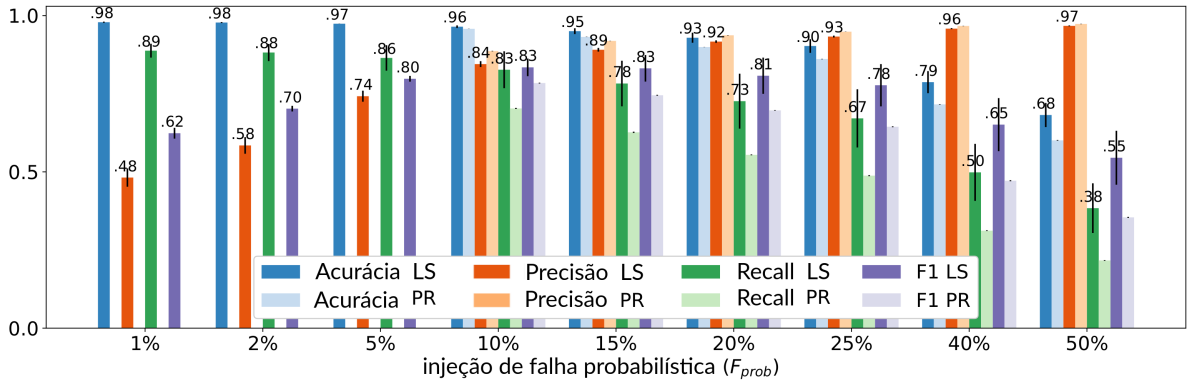


Figura 12 – Comparação da solução LSTM proposta (rótulos com sufixo LS) configurada com limite α =0,75, arranjos A =3, e janela W =11) com a técnica probabilística (CORDEIRO et al., 2021) (PR) configurado com α =0,75 para diferentes falhas injetadas probabilísticas (F_{prob}).

Fonte: o autor, 2022

Figura 13. Primeiro, LS empata ou ganha PR em ambos os modelos de falha injetada em geral. Isso é mais evidente considerando F1. Embora possa haver uma pequena desvantagem em termos de precisão, essa diferença é superada no recall. Segundo, considerando os níveis de falha injetada, como esperado dado o complemento do processo de Bernoulli, há um pico no meio e um vale nos cenários de fronteira. Este cruzamento valida a técnica RNAs. Terceiro, o LSTM tende a aproximar o método probabilístico no meio, enquanto nas bordas ele é superado. Quarto, considerando o modelo de falha injetada, retorna ainda mais ganhos no modelo realista.

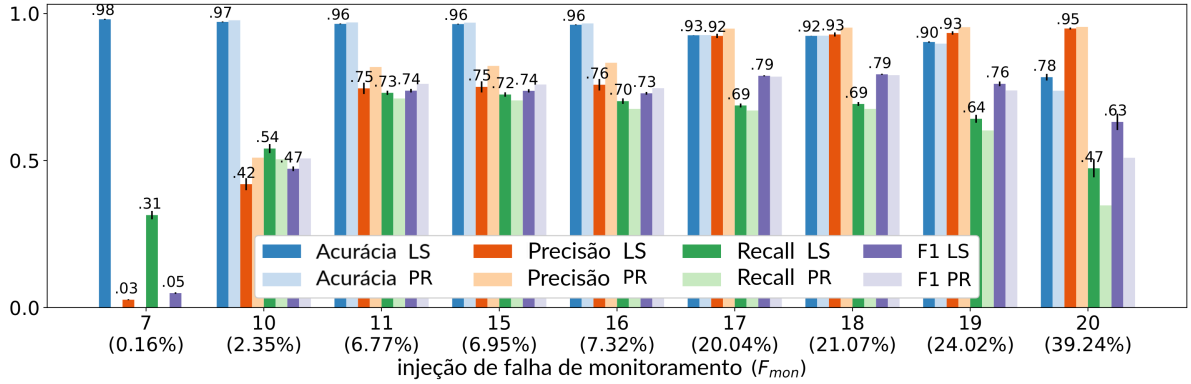


Figura 13 – Comparação do LSTM proposto (rótulos com sufixo LS) configurado com limite α =0,75, arranjos A =3, e tamanho da janela W =11) com técnica probabilística previamente proposta (CORDEIRO et al., 2021) (PR) configurado com α =0,75 para diferentes falhas injetadas de monitoramento (F_{mon}).

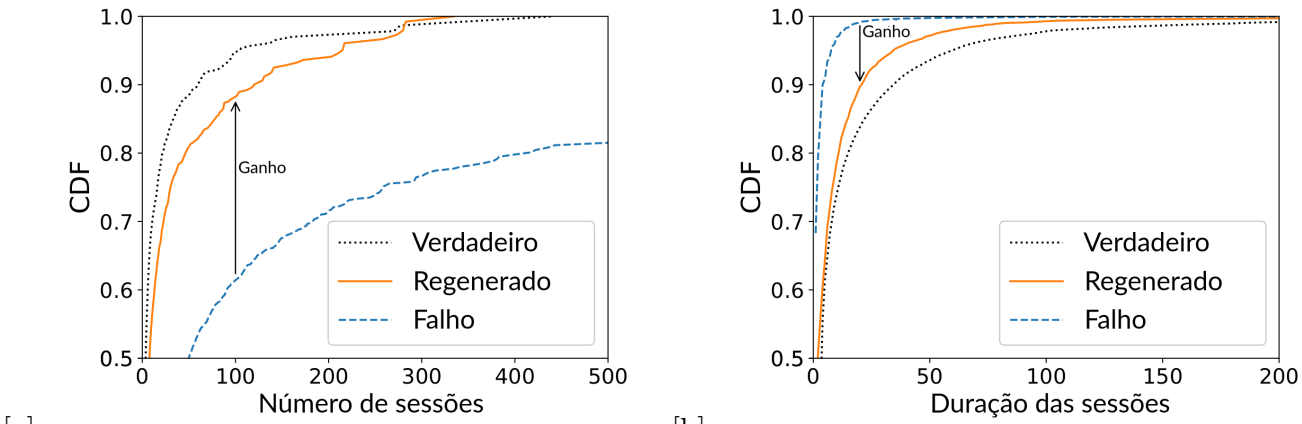
Fonte: o autor, 2022

5.5 Estudo de caso

Dando prosseguimento as avaliações, esta seção avalia o impacto das correções em duas métricas de interesse nos traços de sessões de usuários de sistemas distribuídos (CORDEIRO et al., 2021):

- **Número de Sessões**. Indica o número de sequências de amostras positivas consecutivas observadas para um determinado usuário;
- **Duração das Sessões**. Indica o comprimento de uma sequência de amostras positivas consecutivas para um determinado usuário;

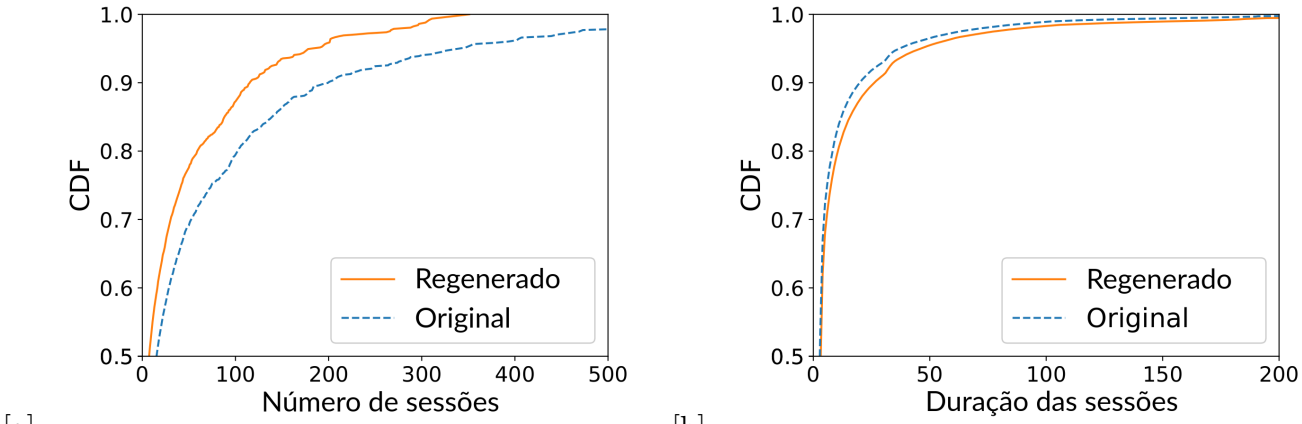
A Figura 14 mostra o impacto de RNAs para regenerar traços. Cada par de subfiguras apresenta a distribuição cumulativa (CDF) para as duas métricas, para um traço de entrada (S) e regenerado (S') usando nossa proposta com LSTM. A Figura 14 mostra três curvas: rastreamento do dado verdadeiro obtido de S1 (F_{mon} =0), rastreamento com falha de S1 (F_{mon} =20) e rastreamento regenerado. Como não temos uma verdade básica



[a] [b]

Figura 14 – Impacto, em termos de número (esquerda) e duração (direita) de rastreamentos regenerados usando nossa solução (topologia = LSTM, limite α =0,75, arranjos A =3, tamanho da janela W =11).

Fonte: o autor, 2022



[a] [b]

Figura 15 – Impacto, em termos de número (esquerda) e duração (direita) de rastreamentos regenerados usando nossa solução (topologia = LSTM, limite α =0,75, arranjos A =3, tamanho da janela W =11).

Fonte: O autor, 2022

para $S3$, a Figura 15 tem apenas duas curvas: $S3$ original e regenerada. Para maior clareza, na Figura 14, os ganhos com base no traço original são destacados.

Na Figura 15 é possível observar que foi obtido um aumento de 12% na área sob a curva para o CDF do número de sessões, e uma diminuição de 0,24% na área sob a curva para o CDF da duração das sessões.

Isso significa uma redução na diferença entre a área sob a curva (AUC) de 223,36 para 13,61, aproximadamente 16x, para o número de sessões, e uma redução de 13,08 para 6, 12, aproximadamente 2x, para a duração das sessões. Esses números quantificam a melhoria geral da qualidade do rastreamento de monitoramento após a correção das falhas cuja probabilidade ultrapassou o limite definido. Em suma, esses resultados sugerem que

as correções de traço tiveram um impacto positivo na aproximação do traço defeituoso dos traços completos.

6 CONCLUSÃO

6.1 Considerações finais

O custo de monitorar de forma eficaz e eficiente as principais entidades em sistemas distribuídos dinâmicos de grande escala - garantindo ampla cobertura e minimizando a probabilidade de falhas - requer estratégias para melhorar a qualidade dos dados coletados. Remover falhas e ruídos que ocorreram durante monitoramento é fundamental para obter dados precisos sobre o comportamento do sistema observado. Embora métodos probabilísticos tenha apresentado resultados importantes, é possível constatar que ainda há um longo caminho a ser percorrido em direção a melhoria da correção.

Neste trabalho, um passo importante em direção a criação de uma nova abordagem foi dado. Em especial, foi apresentada uma nova técnica baseada em rede neural para regenerar traços de monitoramento de sessões online de entidades em sistemas distribuídos. Especificamente, este trabalho fornece fortes evidências de que RNAs podem alcançar resultados superiores, na maioria dos casos, em contraponto a técnicas probabilísticas de última geração. Foi observado ganhos adicionais em casos de bordas tanto em falhas probabilísticas quanto no monitoramento de falhas injetadas - ou seja, para traços menos defeituosos e mais defeituosos.

Este trabalho pode representar um passo importante em direção a técnicas de aprendizado profundo mais sofisticadas para corrigir traços de sistemas de monitoramento distribuído em larga escala.

6.2 Trabalhos em andamento

Os trabalhos em andamento vislumbram as seguintes direções de pesquisa: (i) ampliar a avaliação das técnica proposta até agora, incluindo outros parâmetros e dados; (ii) compreender mais a fundo os impactos causados por diferentes parâmetros selecionados e modelos utilizados; (iii) explorar novas técnicas de redes neurais artificiais profundas (por exemplo, redes convolutivas), para aumentar a qualidade dos resultados; e (iv) explorar traços de outros ambientes, como monitoramento de redes de sensores e datacenters.

REFERÊNCIAS

ABADI, M. et al. **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**. 2015. Software available from tensorflow.org. Disponível em: <https://www.tensorflow.org/>. Citado na página 35.

ANDERSON, S.; BARFORD, C.; BARFORD, P. Five alarms: Assessing the vulnerability of us cellular communication infrastructure to wildfires. In: **ACM Internet Measurement Conference**. New York, NY, USA: ACM, 2020. (IMC '20), p. 162–175. ISBN 9781450381383. Disponível em: <https://doi.org/10.1145/3419394.3423663>. Citado na página 17.

BOUTABA, R. et al. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. **Journal of Internet Services and Applications**, v. 9, p. 1–99, 2018. Citado na página 22.

BOUTABA, R. et al. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. **Journal of Internet Services and Applications**, v. 9, n. 1, p. 16, Jun 2018. ISSN 1869-0238. Disponível em: <https://doi.org/10.1186/s13174-018-0087-2>. Citado na página 37.

BUUREN, S. van. **Flexible Imputation of Missing Data. Second Edition**. Boca Raton, FL: Chapman & Hall/CRC Press, 2018. Disponível em: <https://stefvanbuuren.name/fimd/>. Citado 2 vezes nas páginas 18 e 29.

BUUREN, S. van; GROOTHUIS-OUDSHOORN, K. mice: Multivariate imputation by chained equations in r. **Journal of Statistical Software**, v. 45, n. 3, 2011. Disponível em: <https://www.jstatsoft.org/index.php/jss/article/view/v045i03>. Citado 2 vezes nas páginas 18 e 29.

CHENG, L. et al. Compressive sensing based data quality improvement for crowd-sensing applications. **Journal of Network and Computer Applications**, v. 77, p. 123 – 134, 2017. ISSN 1084-8045. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1084804516302338>. Citado 3 vezes nas páginas 26, 29 e 37.

CHOLLET, F. **Deep Learning with Python**. 1st. ed. USA: Manning Publications Co., 2017. ISBN 1617294438. Citado na página 24.

CORDEIRO, W. et al. Revisiting the coupon collector's problem to unveil users' online sessions in networked systems. **Peer-to-Peer Networking and Applications**, 2021. Citado 6 vezes nas páginas 9, 19, 35, 39, 40 e 41.

COSTARELLI, D.; SPIGLER, R. Multivariate neural network operators with sigmoidal activation functions. **Neural Networks**, v. 48, p. 72–77, 2013. ISSN 0893-6080. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0893608013001950>. Citado na página 22.

CYBENKO, G. V. Approximation by superpositions of a sigmoidal function. **Mathematics of Control, Signals and Systems**, v. 2, p. 303–314, 1989. Citado na página 22.

GUO, T. et al. Simple convolutional neural network on image classification. In: **2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)**. [S.l.: s.n.], 2017. p. 721–724. Citado na página 24.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural Comput.**, MIT Press, Cambridge, MA, USA, v. 9, n. 8, p. 1735–1780, nov. 1997. ISSN 0899-7667. Disponível em: <https://doi.org/10.1162/neco.1997.9.8.1735>. Citado 4 vezes nas páginas 23, 25, 33 e 34.

HORNIK, K. Approximation capabilities of multilayer feedforward networks. **Neural Networks**, v. 4, n. 2, p. 251–257, 1991. ISSN 0893-6080. Disponível em: <https://www.sciencedirect.com/science/article/pii/089360809190009T>. Citado na página 22.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. **Neural Networks**, v. 2, n. 5, p. 359–366, 1989. ISSN 0893-6080. Disponível em: <https://www.sciencedirect.com/science/article/pii/0893608089900208>. Citado na página 22.

HOSSFELD, T. et al. Characterization of BitTorrent swarms and their distribution in the Internet. **Computer Networks**, v. 55, n. 5, p. 1197–1215, 2011. Citado 3 vezes nas páginas 17, 35 e 36.

HOSSIN, M.; M.N, S. A review on evaluation metrics for data classification evaluations. **International Journal of Data Mining Knowledge Management Process**, v. 5, p. 01–11, 03 2015. Citado 2 vezes nas páginas 25 e 26.

HOTA, H. S.; HANDA, R.; SHRIVAS, A. K. Time series data prediction using sliding window based rbf neural network. In: . [S.l.: s.n.], 2017. Citado na página 23.

JUNIOR, N. A. A.; CORDEIRO, W. L. d. C.; GASPARY, L. P. Permitindo Maior Reprodutibilidade de Experimentos em Ambientes Distribuídos com Nodos de Baixa Confiabilidades. In: **36º Simpósio Brasileiro de Redes de Computadores e de Sistemas Distribuídos (SBRC 2018)**. [S.l.: s.n.], 2018. p. 1–14. Citado na página 29.

KAINEN, P.; KůRKOVá, V.; SANGUINETI, M. Approximating multivariable functions by feedforward neural nets. **Intelligent Systems Reference Library**, v. 49, p. 143–181, 01 2013. Citado na página 22.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. In: BENGIO, Y.; LECUN, Y. (Ed.). **3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings**. [s.n.], 2015. Disponível em: <http://arxiv.org/abs/1412.6980>. Citado na página 35.

KOLEN, J. F.; KREMER, S. C. Gradient flow in recurrent nets: The difficulty of learning longterm dependencies. In: _____. **A Field Guide to Dynamical Recurrent Networks**. [S.l.: s.n.], 2001. p. 237–243. Citado na página 34.

KUBAT, M. **An Introduction to Machine Learning**. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2015. ISBN 3319200097. Citado 2 vezes nas páginas 26 e 27.

LAPEDES, A.; FARBER, R. **Nonlinear signal processing using neural networks: Prediction and system modelling**. [S.l.], 1987. Citado na página 22.

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.

LI, X. Simultaneous approximations of multivariate functions and their derivatives by neural networks with one hidden layer. **Neurocomputing**, v. 12, n. 4, p. 327–343, 1996. ISSN 0925-2312. Disponível em: <https://www.sciencedirect.com/science/article/pii/0925231295000704>. Citado na página 22.

MALSBURG, C. v. d. Frank rosenblatt: Principles of neurodynamics: Perceptrons and the theory of brain mechanisms. In: . [S.l.: s.n.], 1986. Citado na página 22.

MANSILHA, R. B. et al. Observing the bittorrent universe through telescopes. In: **2011 IFIP/IEEE International Symposium on Integrated Network Management**. [S.l.: s.n.], 2011. Citado 2 vezes nas páginas 17 e 29.

MASINDE, N.; GRAFFI, K. Peer-to-Peer-Based Social Networks: A Comprehensive Survey. **SN Computer Science**, Springer, v. 1, n. 5, p. 1–51, 2020. Citado na página 17.

MAYER, J. et al. On the resilience of internet infrastructures in pacific northwest to earthquakes. In: **Passive and Active Measurement**. Cham: Springer International Publishing, 2021. p. 247–265. ISBN 978-3-030-72582-2. Citado na página 17.

NADER, A.; AZAR, D. Evolution of activation functions: An empirical investigation. **ACM Trans. Evol. Learn. Optim.**, Association for Computing Machinery, New York, NY, USA, v. 1, n. 2, jul 2021. ISSN 2688-299X. Disponível em: <https://doi.org/10.1145/3464384>. Citado na página 24.

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.

NWANKPA, C. et al. Activation functions: Comparison of trends in practice and research for deep learning. **ArXiv**, abs/1811.03378, 2018. Citado 2 vezes nas páginas 24 e 25.

NWANKPA, C. et al. Activation functions: Comparison of trends in practice and research for deep learning. **ArXiv**, abs/1811.03378, 2018. Citado na página 35.

OLUDELE, A.; JEGEDE, O. Neural networks and its application in engineering. **Proceedings of Informing Science IT Education Conference (InSITE)**, 01 2009. Citado na página 22.

PADMANABHAN, R. et al. Residential links under the weather. In: **ACM Special Interest Group on Data Communication**. New York, NY, USA: ACM, 2019. (SIGCOMM '19), p. 145–158. ISBN 9781450359566. Disponível em: <https://doi.org/10.1145/3341302.3342084>. Citado na página 17.

PAIM, K. et al. Usando redes neurais para reconstruir traços de sessões de usuários de sistemas de larga escala. In: **Anais do XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (in Portuguese)**. Porto Alegre, RS, Brasil: SBC, 2021. p. 826–839. ISSN 2177-9384. Disponível em: <https://sol.sbc.org.br/index.php/sbrc/article/view/16766>. Citado na página 20.

PAIM, K. O. et al. **GitHub - Correcting_Datasets_With_DL_SBRC21 repo**. 2021. Available: <https://github.com/kayua/Correcting_Datasets_With_DL_SBRC21>. Citado na página 20.

PAIM, K. O. et al. **GitHub - Regenerating_Datasets_With_NN_NOMS22 repo**. 2021. Available: <https://github.com/kayua/Regenerating_Datasets_With_NN_NOMS22>. Citado 3 vezes nas páginas 35, 38 e 40.

PAIM, K. O. et al. Fix me if you can: Using neural networks toregenerate networked systems' monitoring traces. In: **IEEE/IFIP Network Operations and Management Symposium (NOMS 2022)**. [S.l.]: IEEE, 2022. Citado na página 20.

PATEL, R.; PATEL, S. A comprehensive study of applying convolutional neural network for computer vision. **International Journal of Advanced Science and Technology**, v. 6, p. 2161–2174, 01 2020. Citado na página 22.

ROHANI, N.; ESLAHCHI, C. Drug-drug interaction predicting by neural network using integrated similarity. **Scientific Reports**, v. 9, 2019. Citado na página 22.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, p. 65–386, 1958. Citado na página 23.

RUBIN, D. B. Multiple imputation after 18+ years. **Journal of the American Statistical Association**, Taylor Francis, v. 91, n. 434, p. 473–489, 1996. Disponível em: <https://www.tandfonline.com/doi/abs/10.1080/01621459.1996.10476908>. Citado na página 29.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Citado na página 22.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning internal representations by error propagation. In: _____. **Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations**. Cambridge, MA, USA: MIT Press, 1986. p. 318–362. ISBN 026268053X. Citado na página 23.

SHELL, J.; GREGORY, W. D. Efficient cancer detection using multiple neural networks. **IEEE Journal of Translational Engineering in Health and Medicine**, v. 5, p. 1–7, 2017. Citado na página 22.

SHI, Z. et al. Survey on neural networks used for medical image processing. **International journal of computational science**, v. 3, 02 2009. Citado na página 22.

SHIRAISHI, F. et al. Torrente, a micropayment based bittorrent extension to mitigate free riding. In: **Companion Proceedings of the 21st Brazilian Symposium on Information and Computational Systems Security**. Porto Alegre, RS, Brasil: SBC, 2021. p. 82–89. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/sbseg_estendido/article/view/17343>. Citado na página 17.

SRIVASTAVA, N. et al. Dropout: A simple way to prevent neural networks from overfitting. JMLR.org, v. 15, n. 1, p. 1929–1958, jan. 2014. ISSN 1532-4435. Citado 2 vezes nas páginas 33 e 35.

STEKHOVEN, D. J.; BüHLMANN, P. MissForest—non-parametric missing value imputation for mixed-type data. **Bioinformatics**, v. 28, n. 1, p. 112–118, 10 2011. ISSN 1367-4803. Disponível em: <https://doi.org/10.1093/bioinformatics/btr597>. Citado na página 29.

WALJEE, A. K. et al. Comparison of imputation methods for missing laboratory data in medicine. **BMJ Open**, British Medical Journal Publishing Group, v. 3, n. 8, 2013. ISSN 2044-6055. Disponível em: <https://bmjopen.bmj.com/content/3/8/e002847>. Citado na página 29.

WANG, C.-H.; WU, X.; CHEN, Y.-T. An empirical analysis for forecasting stock index based on lstm neural network. In: **Proceedings of the 2021 5th International Conference on Electronic Information Technology and Computer Engineering**. [S.l.: s.n.], 2021. p. 636–641. Citado na página 23.

WARNER, B.; MISRA, M. Understanding neural networks as statistical tools. **The American Statistician**, [American Statistical Association, Taylor Francis, Ltd.], v. 50, n. 4, p. 284–293, 1996. ISSN 00031305. Disponível em: <http://www.jstor.org/stable/2684922>. Citado na página 23.

Xie, K. et al. Graph based tensor recovery for accurate internet anomaly detection. In: **IEEE INFOCOM 2018 - The 37th Annual IEEE Conference on Computer Communications**. [S.l.: s.n.], 2018. p. 1502–1510. Citado na página 29.

Xie, K. et al. Accurate recovery of missing network measurement data with localized tensor completion. **IEEE/ACM Transactions on Networking**, v. 27, n. 6, p. 2222–2235, 2019. Citado na página 29.

YAN, X.; WEIHAN, W.; CHANG, M. Research on financial assets transaction prediction model based on lstm neural network. **Neural Computing and Applications**, v. 33, 01 2021. Citado na página 23.

YIN, C. et al. A deep learning approach for intrusion detection using recurrent neural networks. **IEEE Access**, v. 5, p. 21954–21961, 2017. Citado na página 24.

ZHANG, C. et al. Unraveling the bittorrent ecosystem. **IEEE Transactions on Parallel and Distributed Systems**, IEEE Press, Piscataway, NJ, USA, v. 22, n. 7, p. 1164–1177, jul. 2011. ISSN 1045-9219. Citado na página 29.

Apêndices

APÊNDICE A – ARTIGO PUBLICADO NO SBC SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES 2021

- Título: Usando Redes Neurais para Reconstruir Traços de Sessões de Usuários de Sistemas de Larga Escala
- Autores: Kayuã Oleques Paim, Rafael Duarte Beltran, Rodrigo Brandão Mansilha, Weverton Cordeiro
- DOI: <https://doi.org/10.5753/sbrc.2021.16766>
- Disponível: <https://sol.sbc.org.br/index.php/sbrc/article/view/16766/16608>

APÊNDICE B – ARTIGO PUBLICADO NO IEEE/IFIP NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM 2022

- Título: Fix Me If You Can: Using Neural Networks toRegenerate Networked Systems' Monitoring Traces
- Autores: Kayuã Oleques Paim, Vagner E. Quincozes, Diego Kreutz, Rodrigo Brandão Mansilha, Weverton Cordeiro
- Disponível: <https://noms2022.ieee-noms.org/program/technical-sessions>

APÊNDICE C – REPOSITÓRIO PARA REPRODUZIR OS RESULTADOS PUBLICADOS NO SBRC'21

- Primeiro clonar o repositório do Correcting Datasets With DL SBRC21 Public, disponível em <https://github.com/kayua/Correcting_Datasets_With_DL_SBRC21>.
- O próximo passo é rodar o comando para instalar as dependências:

  ```
  pip install -r requirements.txt
  ```

- Em seguida rodar os *script* usando o comando:

  ```
  python3 run_sbrc21.py -c sbrc
  ```

- Os resultados serão salvos no arquivo noms21.txt

APÊNDICE D – REPOSITÓRIO PARA REPRODUZIR OS RESULTADOS PUBLICADOS NO NOMS'22

- Primeiro clonar o repositório do Regenerating Datasets With NN NOMS22, disponível em <https://github.com/kayua/Regenerating_Datasets_With_NN_NOMS22>.
- O próximo passo é rodar o comando para instalar as dependências:

  ```
  pip install -r requirements.txt
  ```

- Em seguida rodar os *script* usando o comando:

  ```
  python3 run_noms22.py -c noms
  ```

- Os resultados serão salvos no arquivo noms22.txt