

UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Uma Abordagem de Aprendizagem Federada para Classificação de Enlaces de Rede a partir de Matrizes de Tráfego Usando Computação de Borda

Nicolas André Alves da Silva

Orientador
Dr. Carlos Alberto Vieira Campos

Co-orientador
Dr. Sidney Cunha de Lucena

RIO DE JANEIRO, RJ - BRASIL
MARÇO 2026

Uma Abordagem de Aprendizagem Federada para Classificação de Enlaces de Rede a partir de Matrizes de Tráfego Usando Computação de Borda

Nicolas André Alves da Silva

TESE APRESENTADA COMO REQUISITO PARCIAL PARA OBTENÇÃO DO TÍTULO DE DOUTOR PELO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA DA UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO (UNIRIO). APROVADA PELA COMISSÃO EXAMINADORA ABAIXO ASSINADA.

Aprovada por:

Documento assinado digitalmente
CARLOS ALBERTO VIEIRA CAMPOS
Data: 29/05/2026 14:17:15-0300
Verifique em https://validar.iti.gov.br

Dr. Carlos Alberto Vieira Campos - UNIRIO

Documento assinado digitalmente
SIDNEY CUNHA DE LUCENA
Data: 29/05/2026 11:19:00-0300
Verifique em https://validar.iti.gov.br

Dr. Sidney Cunha de Lucena - UNIRIO

Documento assinado digitalmente
JEFFERSON ELBERT SIMOES
Data: 28/05/2026 16:30:11-0300
Verifique em https://validar.iti.gov.br

Dr. Jefferson Elbert Simões - UNIRIO

Documento assinado digitalmente
PEDRO NUNO DE SOUZA MOURA
Data: 29/05/2026 07:58:16-0300
Verifique em https://validar.iti.gov.br

Dr. Pedro Nuno de Souza Moura - UNIRIO

Documento assinado digitalmente
DIEGO GIMENEZ PASSOS
Data: 29/05/2026 09:05:50-0300
Verifique em https://validar.iti.gov.br

Dr. Diego Gimenez Passos - Instituto Superior de Engenharia de Lisboa

Documento assinado digitalmente
DIOGO MENEZES FERRAZANI MATTOS
Data: 29/05/2026 09:18:29-0300
Verifique em https://validar.iti.gov.br

Dr. Diogo Menezes Ferrazani Mattos - UFF

RIO DE JANEIRO, RJ - BRASIL
MARÇO 2026

Catalogação informatizada pelo(a) autor(a)

S586 Silva, Nicolas André Alves da
Uma Abordagem de Aprendizagem Federada para Classificação de Enlaces de Rede a partir de Matrizes de Tráfego Usando Computação de Borda / Nicolas André Alves da Silva. -- Rio de Janeiro : UNIRIO, 2026.
92 f

Orientador: Carlos Alberto Vieira Campos.
Coorientador: Sidney Cunha de Lucena.
Tese (Doutorado) - Universidade Federal do Estado do Rio de Janeiro, Programa de Pós-Graduação em Informática, 2026.

1. Aprendizado Federado. 2. Matrizes de Tráfego. 3. Computação de Borda. I. Campos, Carlos Alberto Vieira, orient. II. Lucena, Sidney Cunha de, coorient. III. Título.

Dedico esta tese à minha família e a todos que contribuíram e contribuem na minha vida pessoal, acadêmica e profissional.

Agradecimentos

Agradeço aos meus pais, Andréa Alves da Silva e Nelson Antônio de Mello Silva, por sempre me darem suporte desde o início dos meus estudos e por viabilizarem as minhas oportunidades acadêmicas e profissionais. Agradeço, também, aos meus irmãos Nelson Antônio de Mello Silva Junior e Davi Alves da Silva pelo companheirismo desde a nossa infância e ao longo dos nossos anos de vida.

Agradeço à minha esposa Maria Camila Do Coutto Prado Valladares por compartilhar comigo esta caminhada, cada um em seu processo de doutoramento, porém trocando experiências e auxílios mútuos. Agradeço, também, a ela pelo seu afeto, por todo o seu incentivo em momentos difíceis e por sempre acreditar no meu potencial.

Agradeço, em conjunto, a todos os membros da minha família, em especial aos meus avós Pedrina da Silva Alves e Manoel Marinho Alves, por contribuírem com minha história de vida e apoio emocional em momentos difíceis.

Agradeço à Prof. Aline Gesualdi Manhães por sua orientação no meu trabalho de conclusão de curso de graduação no Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ) e por ter me incentivado a seguir meus estudos na pós-graduação. Também, agradeço a todos os demais professores e colegas do CEFET/RJ que me acompanharam neste início dos meus estudos.

Agradeço à Prof. Rosa Maria Meri Leão pelos ensinamentos e por sua orientação no meu curso de mestrado. Neste mesmo contexto, agradeço também ao Prof. Edmundo Albuquerque de Souza e Silva e aos demais professores e alunos da Universidade Federal do Rio de Janeiro (UFRJ) que contribuíram para este início da minha carreira na área de pesquisa acadêmica e científica.

Agradeço ao Prof. Carlos Alberto Vieira Campos, primeiramente, por toda sua orientação no meu curso de doutorado. Também sou grato pelas suas aulas e reuniões usadas para me passar sua experiência na pesquisa acadêmica e científica através de ideias, diretrizes e conselhos importantes que viabilizaram submissões de artigos e que resultaram neste presente trabalho de tese.

Agradeço ao Prof. Sidney Cunha de Lucena por sua coorientação no meu curso de doutorado e por toda a sua participação e contribuição para o meu trabalho de pesquisa. Sua grande experiência me proporcionou muitos resultados.

Agradeço aos demais professores e colegas do Programa de Pós-Graduação em Informática (PPGI) da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) por todo o conhecimento adquirido ao longo do meu curso de doutorado e pela oportunidade de desenvolver meu trabalho de pesquisa na área de Ciência da Computação.

Agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo suporte ao longo dos meus estudos. Agradeço ao Instituto de Pesquisas da Marinha (IPqM) pela sua contribuição na minha experiência profissional e pelo apoio dos meus colegas deste instituto.

Por fim, a todos aqueles que contribuíram, direta ou indiretamente, para a realização desta tese de doutorado, o meu sincero agradecimento.

Silva, Nicolas André Alves da. **Uma Abordagem de Aprendizagem Federada para Classificação de Enlaces de Rede a partir de Matrizes de Tráfego Usando Computação de Borda**. UNIRIO, 2026. 77 páginas. Tese de Doutorado. Programa de Pós-Graduação em Informática, UNIRIO.

RESUMO

Monitorar enlaces de rede é uma tarefa importante para fornecer serviços de rede, tais como o roteamento de pacotes em Redes Definidas por *Software* (SDNs). O nível de utilização de um enlace de rede está diretamente relacionado com a probabilidade de ocorrer congestionamento de pacotes, logo, monitorar frequentemente os níveis de utilização dos enlaces de rede garante o resultado de tarefas que dependem de requisitos mínimos de Qualidade de Serviço (QoS). Ao invés de usar sensores nos enlaces de rede durante o monitoramento, o controlador da SDN pode calcular os níveis de utilização de seus enlaces usando informações de demanda de ponta a ponta e o conhecimento das rotas de pacotes. Para otimizar esse cálculo, este trabalho propõe uma solução geral de um modelo de Aprendizado de Máquina (ML) que classifica um enlace de rede quanto a sua à sua taxa de utilização. Essa classificação usa como dado de entrada uma matriz de tráfego que contém todas as demandas de tráfego de ponta a ponta da rede em um determinado intervalo de tempo. De acordo com esta solução geral, são apresentados também os modelos *LinkClassifier* e *NetLinkClassifier*. Para treinar o *LinkClassifier*, é necessário o valor numérico de taxa de utilização do enlace de rede, além das informações de entrada e saída do modelo. Enquanto no cenário do *NetLinkClassifier*, as matrizes de tráfego são montadas numa camada de Computação de Borda de Acesso Múltiplo (MEC) da SDN, utilizando informações dos nós da SDN. Antes do treinamento do *NetLinkClassifier*, um conjunto de dados supervisionados é armazenado de forma distribuída nessa camada MEC. Com isso, o *NetLinkClassifier* é treinado utilizando o paradigma de Aprendizado Federado (FL). Para garantir a segurança no armazenamento e nas transmissões de dados, recomenda-se um sistema de criptografia. Um estudo fornecido neste trabalho avalia essa influência da criptografia no desempenho da classificação. Outras análises de desempenho comparam o uso do modelo *NetLinkClassifier*, variando os parâmetros do modelo e as configurações de SDN. Após isto, é mostrado como o modelo *NetLinkClassifier* pode ser aplicado para dar suporte a uma Função de Rede Virtualizada (VNF) que realiza o roteamento de pacotes da SDN. Por fim, dado o potencial de generalização da solução proposta e baseando-se neste modelo geral de classificação de enlaces de rede, um estudo apresenta o modelo *CityZoneClassifier* que classifica uma zona de uma cidade quanto à sua utilização em trajetos de veículos.

Palavras-chave: Qualidade de Serviço, Matrizes de Tráfego, Aprendizado Federado, Computação de Borda de Acesso Múltiplo, Sistemas Inteligentes de Transporte.

ABSTRACT

Monitoring network links is an important task for providing network services, such as packet routing in Software-Defined Networks (SDNs). The utilization level of a network link is directly related to the probability of packet congestion; therefore, frequently monitoring network link utilization levels ensures the success of tasks that depend on minimum Quality of Service (QoS) requirements. Instead of using sensors on network links during monitoring, the SDN controller can calculate the utilization levels of its links using end-to-end demand information and knowledge of packet routes. To optimize this calculation, this work proposes a general solution for a Machine Learning (ML) model that classifies a network link according to its utilization rate. This classification uses as input data a traffic matrix containing all end-to-end network traffic demands over a given time interval. According to this general solution, the *LinkClassifier* and *NetLinkClassifier* models are also presented. To train the *LinkClassifier*, the numerical value of the network link utilization rate is required, in addition to the model's input and output information. In the *NetLinkClassifier* scenario, traffic matrices are assembled in a Multi-Access Edge Computing (MEC) layer of the SDN, using information from the SDN nodes. Before training the *NetLinkClassifier*, a set of supervised data is stored in a distributed manner in this MEC layer. The *NetLinkClassifier* is then trained using the Federated Learning (FL) paradigm. To ensure security in data storage and transmission, a cryptographic system is recommended. A study provided in this work evaluates the influence of cryptography on classification performance. Other performance analyses compare the use of the *NetLinkClassifier* model, varying the model parameters and SDN configurations. Following this, it is shown how the *NetLinkClassifier* model can be applied to support a Virtualized Network Function (VNF) that performs SDN packet routing. Finally, given the potential for generalization of the proposed solution and based on this general network link classification model, a study presents the *CityZoneClassifier* model, which classifies a city zone according to its use in vehicle routes.

Keywords: Quality of Service, Traffic Matrices, Federated Learning, Multi-Access Edge Computing, Intelligent Transportation Systems.

Sumário

1 Introdução **1**

1.1 Justificativa e Motivação 2

1.2 Problema 2

1.3 Objetivo 3

1.4 Questões de Pesquisa 4

1.5 Contribuições 4

1.6 Organização 5

2 Fundamentação Teórica **6**

2.1 Gradiente Descendente 6

2.2 Aprendizado Federado 7

2.3 Criptografia AES-GCM 8

3 Monitoramento de redes através da classificação de enlaces **10**

3.1 Revisão da Literatura 10

3.2 Formulação do problema 14

3.3 Proposta de solução do problema 15

4 Inferência da utilização de enlaces para caminhos de rede a partir de matrizes de tráfego **17**

4.1 Formulação do problema 17

4.2 *LinkClassifier* 18

4.2.1 Pré-processamento e montagem de um conjunto de dados 20

4.2.2 Treinamento 20

4.2.3 Utilização do modelo treinado 22

4.3 Resultados e desempenho de classificação 23

4.3.1 Conjuntos de dados 23

4.3.2 Avaliação de desempenho 24
4.3.3 Resultados comparados com modelos de classificação tradicionais 25
4.3.4 O *LinkClassifier* em uma tarefa de roteamento 28
4.4 Comentários finais 30

5 Uso de Aprendizado Federado para classificação de enlaces de rede baseada em matrizes de tráfego 31
5.1 Formulação do problema 32
5.2 NetLinkClassifier 34
5.2.1 Pré-processamento e montagem do *dataset* 35
5.2.2 Treinamento do modelo 36
5.2.3 Utilização do modelo treinado 37
5.3 Resultados e desempenho da classificação 38
5.3.1 Datasets 39
5.3.2 Avaliação de desempenho 41
5.3.3 Resultados comparados com modelos de classificação tradicionais 42
5.3.4 Resultados variando parâmetros do modelo e tipos de rede do modelo . . . 44
5.3.5 Resultados variando as características do conjunto de dados e as configurações SDN 48
5.4 Estudo de caso numa tarefa de roteamento 51
5.5 Comentários finais 53

6 Aplicação da Proposta em Sistemas Inteligentes de Transporte 54
6.1 Formulação do problema de ITS 55
6.2 CityZoneClassifier 57
6.2.1 Pré-processamento e montagem de um conjunto de dados 57
6.2.2 Treinamento 58
6.2.3 Utilizando o modelo treinado 59
6.3 Resultados e desempenho de classificação 60
6.3.1 Dataset 60
6.3.2 Avaliação de desempenho 61

6.3.3 Resultados variando os parâmetros do modelo e da rede 62
6.3.4 Resultados comparados com modelos de classificação tradicionais 63
6.4 Usabilidade do *CityZoneClassifier* numa tarefa de ITS 65
6.5 Comentários finais 65

7 Conclusão **67**
7.1 Respostas às Questões de Pesquisa 68
7.2 Trabalhos Futuros 69

References **70**

Lista de Figuras

Figure 1: Rede R com o monitoramento realizado por uma VNF. 15

Figure 2: Modelo de classificação da rede. 15

Figure 3: Exemplo de uma rede com 5 nós: (a) topologia e (b) matriz de tráfego. 16

Figure 4: Instâncias do modelos de ML para a classificação de cada enlace k da rede. 16

Figure 5: Modelo de classificação de enlace de rede usando uma matriz de tráfego como entrada. 18

Figure 6: Distribuição dos valores de cada métrica do *LinkClassifier* (3 classes). 26

Figure 7: Distribuição dos valores de cada métrica do *LinkClassifier* (4 classes). 27

Figure 8: Distribuição dos valores de cada métrica do *LinkClassifier* (5 classes). 28

Figure 9: Resultado das classificações dos enlaces de rede, baseadas nas demandas dos nós. Legenda de rótulos: 1 - azul-claro, 2 - azul-médio; 3 - azul-escuro; 4 - amarelo; 5 - vermelho. 29

Figure 10: Classificação dos enlaces após as ações de desvio de rota de pacotes. Legenda de rótulos: 1 - azul-claro, 2 - azul-médio; 3 - azul-escuro; 4 - amarelo; 5 - vermelho. 30

Figure 11: Arquitetura de uma rede SDN com uma camada MEC. 33

Figure 12: Modelo de classificação de um enlace de rede k. 34

Figure 13: Representação gráfica do modelo *NetLinkClassifier*. 34

Figure 14: Dados dos nós para rotular VNF e, em seguida, para a camada MEC. 36

Figure 15: Iterações do FL. 37

Figure 16: Utilização do modelo *NetLinkClassifier*. 38

Figure 17: Exemplo da rotulagem do conjunto de dados Abilene com 3 classes. 41

Figure 18: Distribuição dos valores de cada métrica do *NetLinkClassifier*. . . 43

Figure 19: Valores de precisão e pontuação F1 variando a taxa de aprendizado. 44

Figure 20: Valores de precisão por iteração de treinamento variando a taxa de aprendizado. 45

Figure 21: Valores de acurácia variando ep e bs. 46

Figure 22: Atraso médio de processamento local por iteração variando ep e bs. 46

Figure 23: Acurácia variando o tipo de rede do modelo do *NetLinkClassifier*. 47

Figure 24: Valores de precisão variando o número de classes. 48

Figure 25: Valores de precisão variando o número de nós MEC. 49

Figure 26: Valores de *F1-score* do *NetLinkClassifier* com ou sem criptografia AES-GCM. 49

Figure 27: Valores de acurácia variando o sistema de roteamento da SDN. . 50

Figure 28: Valores de acurácia variando a arquitetura da rede SDN e o número de links para classificação. 50

Figure 29: Identificações de nós e links na topologia da rede Abilene. 52

Figure 30: Classificação de links antes das ações de desvio de rota de pacotes. 52

Figure 31: Classificação de links após ações de desvio de rota de pacotes. . . 53

Figure 32: Arquitetura de uma rede VEC com camadas MEC e *Cloud*. . . . 56

Figure 33: Representação gráfica do modelo *CityZoneClassifier*. 57

Figure 34: Aquisição de amostra do conjunto de dados na camada MEC. . . 58

Figure 35: Iterações do treinamento do modelo usando o algoritmo *FedAvg*. . 59

Figure 36: Utilizando a saída do modelo *CityZoneClassifier* em uma função ITS da rede VEC da Cidade Inteligente. 60

Figure 37: Valores de acurácia variando a taxa de aprendizado. 62

Figure 38: Valores de acurácia variando ep e bs. 62

Figure 39: Distribuição dos valores de cada métrica do *CityZoneClassifier*. . 64

Lista de Tabelas

Table 1: Caracterização dos trabalhos relacionados 13

Table 2: Desempenho do *LinkClassifier* e outros modelos (3 classes). . . . 25

Table 3: Desempenho do *LinkClassifier* e outros modelos (4 classes). . . . 26

Table 4: Desempenho do *LinkClassifier* e outros modelos (5 classes). . . . 27

Table 5: Desempenho do modelo *NetLinkClassifier* em comparação com modelos de classificação tradicionais. 42

Table 6: A quantidade de dados transferidos para um controlador SDN antes e depois de 100 iterações. 44

Table 7: Número de pesos e quantidade de dados transferidos por iteração em cada tipo de rede modelo. 48

Table 8: Taxas máximas de utilização do enlace antes e depois da ação de desvio de pacotes. 53

Table 9: Desempenho do *CityZoneClassifier* em diferentes configurações MEC. 63

Table 10: Desempenho do *CityZoneClassifier* com diferentes números de classes. 63

Table 11: Desempenho do modelo *CityZoneClassifier* comparado com modelos de classificação tradicionais. 64

Lista de Nomenclaturas

AES	Padrão de Criptografia Avancada (em inglês, *Advanced Encryption Standard*)
AES-GCM	Padrão de Criptografia Avancada com o mode Galois/Counter (em inglês, *Advanced Encryption Standard with Galois/Counter Mode*)
Anatel	Agência Nacional de Telecomunicações
DT	Árvore de Decisão (em inglês, *Decision Tree*)
DTN	Rede Gêmea Digital (em inglês, *Digital Twin Network*)
FL	Aprendizado Federado (em inglês, *Federated Learning*)
IMT-2020	*International Mobile Telecommunications*-2020
INT	Telemetria *In-Band* de Rede (em inglês, *In-band Network Telemetry*)
ISP	Provedores de Serviços de Internet (em inglês, *Internet Service Provider*)
ITS	Sistemas Inteligentes de Transporte (em inglês, *Intelligent Transportation Systems*)
LR	Regressão Linear (em inglês, *Linear Regression*)
LGR	Regressão Logística (em inglês, *Logistic Regression*)
MEC	Computação de Borda de Acesso Múltiplo, (em inglês, *Multi-Access Edge Computing*)
ML	Aprendizado de Máquina (em inglês, *Machine Learning*)
MLP	*Perceptron* Multi-camada (em inglês, *Multi-Layer Perceptron*)
NIST	*National Institute of Standards and Technology*
OSPF	*Open Shortest Path First*
QoS	Qualidade de serviço (em inglês, *Quality of Service*)
ReLU	Unidade Linear Retificada (em inglês, *Rectified Linear Unit*)
SDN	Rede Definida por *Software* (em inglês, *Software-Defined Network*)

SVM	Máquina de Vetores de Suporte (em inglês, *Support Vector Machine*)
VEC	Computação de Borda Veicular (em inglês, *Vehicular Edge Computing*)
VM	Máquina Virtual (em inglês, *Virtual Machine*)
VNF	Funções Virtualizadas de Rede (em inglês, *Virtualized Network Function*)

1. Introdução

Num mundo cada vez mais conectado, existem muitos serviços de comunicação entre computadores, servidores locais, pontos de acesso de dispositivos móveis e etc. Além disso, ao longo dos anos, esses serviços de comunicação necessitam de mais e mais capacidade e velocidade de tráfego.

No contexto nacional, a procura por mais conexões de rede de alta velocidade ainda é crescente nos últimos anos. Segundo a Agência Nacional de Telecomunicações (Anatel), no Brasil, a quantidade dos acessos de Banda Larga saltou de 36,7 milhões no início de 2021 para 54,4 milhões de acessos no início de 2026[1].

No contexto internacional, também existem alguns registros desta tendência. De acordo com (Clausen *et al.*, 2022), em 2022, o consumo de dados em redes domésticas no Reino Unido continuava em crescimento e grande parte desta tendência vinha do crescente consumo de conteúdos relacionados a *streaming* de vídeo, um conteúdo que demandam alta capacidade de rede. Além disso, esse aumento de demanda também pode ser notado pela própria evolução de capacidade ao longo das gerações de redes móveis. Os autores de (Navarro-Ortiz *et al.*, 2020) mostram que o padrão da *International Mobile Telecommunications*-2020 (IMT-2020) para redes 5G pode alcançar picos de velocidade de 10 Gbps e garante ao usuário uma taxa de transferência de dados de 100 Mbps para *download*.

Por isso, empresas de serviços de rede, tais como Provedores de Serviços de Internet (*Internet Service Providers* – ISPs) ou servidores de conteúdo *on-line*, necessitam do monitoramento da utilização dos enlaces de suas redes (Labonne *et al.*, 2020). Essas empresas realizam o monitoramento de seus enlaces de rede para prevenir o congestionamento do tráfego que passa por eles, o que pode levar a problemas como atrasos ou perdas de pacotes.

Segundo (Hu *et al.*, 2014), a tecnologia de Redes Definidas por *Software* (*Software-Defined Networks* – SDNs) permite que seus administradores de rede tenham mais controle sobre o gerenciamento do tráfego de rede. Isso ocorre pois, em SDNs, há uma separação entre o Plano de Dados e o Plano de Controle da rede. O Plano de Dados contém os nós e enlaces que trafegam os dados dos usuários da rede. E o Plano de Controle contém o controlador ou controladores da rede que recebem as informações do tráfego da rede e podem enviar comandos para a infraestrutura do Plano de Dados da SDN.

As tarefas de gerenciamento de uma SDN podem se encontrar muito distantes dos nós clientes da rede, se esta SDN possui seus controladores operados através de uma camada de computação *Cloud* (Basu *et al.*, 2018).

Usando computação *Cloud*, o conceito de Funções Virtualizadas de Rede (*Virtualized Network Functions* – VNFs) pode ser aplicado para comandar as tarefas de gerenciamento da SDN. Segundo (Kaur *et al.*, 2022), a implantação e a configuração de

[1] <http://www.anatel.gov.br/paineis/acessos/banda-larga-fixa>

VNFs são fáceis em comparação com instalações de novos dispositivos físicos para executar uma determinada função de rede.

Em redes tradicionais, alcançar a escalabilidade exige recursos, tempo e muito planejamento. Por outro lado, a utilização de VNFs para realizar serviços da SDN permite reduzir ou expandir recursos, dinamicamente, de maneira mais fácil, com base na demanda.

1.1 Justificativa e Motivação

De acordo com (Uhlig *et al.*, 2006), as taxas de utilização dos enlaces de uma rede num determinado instante de tempo podem ser calculadas, ao invés de obtidas numa medição ativa ou numa medição passiva diretamente nos enlaces.

Uma medição ativa nos enlaces da rede de (Uhlig *et al.*, 2006) necessitaria de inserção de pacotes na rede e um sistema de controle desses meta dados dentro da rede principal. Uma medição passiva diretamente nos enlaces de rede necessitaria de uma infraestrutura com, pelo menos, uma unidade de medida (i.e., sensor) em cada enlace, escalando esta quantidade de sensores em redes com muitos enlaces.

Logo, o cálculo de (Uhlig *et al.*, 2006) para obtenção das taxas de utilização dos enlaces de rede usa a matriz de tráfego do instante de tempo em análise e o conhecimento da topologia da rede e do algoritmo de roteamento de pacotes. Uma matriz de tráfego contém medidas do tráfego de origem e destino entre todos os nós da rede, num dado instante de tempo.

Trabalhos tais como (Mesquita; Assis, 2019; Amoroso *et al.*, 2020; Amoroso *et al.*, 2023) encontraram formas menos custosas de processar dados com alta periodicidade no formato de matrizes de tráfego, através do treinamento de um modelo que utiliza Aprendizado de Máquina (*Machine Learning* – ML).

Por outro lado, revisando a literatura mais recente sobre o gerenciamento de redes, encontram-se muitas abordagens cujo objetivo principal se refere à ação corretiva a ser realizada para mitigar problemas encontrados na rede. Poucos trabalhos são voltados para o diagnóstico deste problema.

Há uma diferença nestes dois tipos de abordagens; pois, se um objeto de estudo é o monitoramento da rede, podem ser consideradas análises mais detalhadas sobre a identificação de problemas em diferentes configurações de redes. Essa lacuna existente na literatura justifica a necessidade de estudos nesta área.

1.2 Problema

Visto que o monitoramento dos enlaces de rede ajuda a prevenir e se antecipar frente à ocorrência de possíveis problemas relacionados ao congestionamento de pacotes, é importante que o sistema de gerenciamento de uma rede meça regularmente a taxa de utilização de seus enlaces.

Algumas tarefas deste monitoramento podem ser realizadas através de VNFs. Segundo (Gao *et al.*, 2020), muitos ISPs utilizam Máquinas Virtuais (Virtual Machines – VMs) de alto desempenho localizadas em grandes *datacenters* pertencentes a provedores de infraestrutura de Nuvem (*Cloud*) pública para implantar suas VNFs. Uma Nuvem pública opera através de recursos de computação pela internet, disponibilizados por provedores terceirizados tais como *Amazon Web Services*, *Google Cloud* e *Microsoft Azure*).

Conforme apresentado na Seção 1.1, as taxas de utilização de enlaces podem ser calculadas a partir de matrizes de tráfego, conhecendo o sistema de roteamento de pacotes e a topologia da rede (Uhlig *et al.*, 2006).

Se a tarefa de cálculo da taxa de utilização de enlaces de uma SDN for realizada através de uma VNF numa rede *Cloud* pública, pode não ser interessante, do ponto de vista de segurança e de custo de recursos, que esta VNF receba e armazene tais informações de sistema de roteamento de pacotes e topologia da rede. Isto porque VNFs em computação *Cloud* pública se encontram fisicamente muito distantes da rede em análise.

Além disso, ao invés calcular e analisar os valores numéricos de taxa de utilização de enlace, algumas aplicações que monitoram enlaces de rede apenas consideram alguns níveis discretos e aproximados de utilização de enlace (Varyani *et al.*, 2020). Assim, o sistema de monitoramento lida com rótulos definidos que representam uma faixa desses valores de utilização de enlace.

1.3 Objetivo

O objetivo principal deste trabalho é investigar soluções para o problema de realizar o monitoramento de enlaces de rede, quando esta tarefa não possui as informações do sistema de roteamento de pacotes nem o conhecimento da topologia da rede. Essas soluções de monitoramento devem realizar a classificação dos enlaces da rede através de rótulos de classificação que representam as respectivas taxas de utilização desses enlaces. O dado de entrada dessa classificação é uma matriz de tráfego referente a um determinado instante de tempo, contendo medidas do tráfego entre os nós da rede.

Os objetivos parciais deste trabalho são:

- Investigar a solução de classificação de enlaces de rede em diferentes cenários e configurações de SDN;
- Analisar a viabilidade de implementação de sistemas de segurança para a tarefa de classificação de enlaces;
- Analisar uma forma de solução distribuída para o problema de classificação de enlaces de rede;
- Demonstrar como a classificação de enlaces pode auxiliar na execução de outros serviços da rede; e

- Investigar como a solução de classificação de enlaces de rede pode ser aplicada em outros problemas de outras áreas de pesquisa.

1.4 Questões de Pesquisa

A partir da definição do problema e do objetivo desta pesquisa, as seguintes questões de pesquisa foram elaboradas:

Q1 – É viável treinar um modelo de ML para classificar os enlaces de rede quanto à sua utilização, sem o conhecimento da sua topologia ou a informação do sistema de roteamento de pacotes?

Q2 – Caso o *dataset* para treinar este modelo se encontre distribuídos próximos dos nós de origem dos dados, é efetivo usar a abordagem FL?

Q3 – Qual é o impacto de adicionar uma criptografia nestes dados durante o aprendizado do modelo?

Q4 – Como a classificação de enlaces da rede pode auxiliar um serviço de distribuição de cargas nesta rede?

Q5 – É possível aplicar esta ideia de classificação de enlaces de rede em outras áreas? Em qual área, por exemplo?

1.5 Contribuições

As contribuições deste trabalho se encontram listadas, abaixo:

- Proposta geral de um modelo de classificação de enlaces de rede baseado em ML para inferir um rótulo de classificação usando uma matriz de tráfego;
- Metodologia para coleta de dados e montagem do *dataset* usado para treinar o modelo em diferentes cenários;
- Aplicação desta metodologia usando dados reais;
- Demonstração de como a classificação de enlaces pode ser usada para auxiliar uma tarefa de roteamento de pacotes;
- Modelo *LinkClassifier* que classifica um enlace de rede usando duas etapas;
- Modelo *NetLinkClassifier* que classifica um enlace de rede usando numa camada MEC da SDN e é treinado usando FL; e

- Modelo *CityZoneClassifier* que classifica uma zona de uma cidade quanto à sua utilização nas rotas de um determinado transporte modal.

1.6 Organização

Esta seção fornece, de forma resumida, a estrutura e o conteúdo desta tese de doutorado. A lista, a seguir, apresenta uma descrição dos tópicos abordados em cada um dos capítulos presentes neste trabalho:

- Capítulo 1: introduz a área de conhecimento na qual a tese está inserida, explica a motivação e justificativa da pesquisa, apresenta um breve resumo do problema abordado e lista os objetivos e contribuições deste trabalho;
- Capítulo 2: apresenta a fundamentação teórica e a formalização de conceitos que estão presentes nos capítulos subsequentes, auxiliando na compreensão de todo o conteúdo abordado neste trabalho;
- Capítulo 3: lista trabalhos relacionados ao assunto do problema apresentado e discutido nesta tese, formaliza e detalha este problema, e apresenta uma solução para ele;
- Capítulo 4: apresenta um trabalho derivado do problema e solução apresentados no Capítulo 3, contendo mais detalhes de metodologia e implementação em dados reais de redes conhecidas na literatura;
- Capítulo 5: apresenta um segundo trabalho derivado do problema e solução apresentados no Capítulo 3, porém no cenário de uma SDN que armazena e processa os dados de medidas de forma distribuída, aplicando sua metodologia e implementando sua proposta usando dados reais de redes conhecidas na literatura;
- Capítulo 6: apresenta um terceiro trabalho derivado do problema e solução apresentados no Capítulo 3, porém no contexto de Sistemas Inteligentes de Transporte;
- Capítulo 7: conclui o trabalho, discutindo sobre os resultados das análises presentes na tese e listando alguns tópicos de pesquisas para os trabalhos futuros.

2. Fundamentação Teórica

Este capítulo introduz os conceitos básicos necessários para a elaboração da proposta de solução do problema apresentado. Portanto, este capítulo apresenta um conhecimento básico sobre gradiente descendente, o conceito geral de Aprendizado Federado e uma breve introdução sobre criptografia AES-GCM.

2.1 Gradiente Descendente

O Aprendizado de Máquina do tipo supervisionado é constituído, basicamente, por um modelo inicial f e um conjunto de dados, cujas amostras possuem valores de entrada X para este modelo e seus respectivos valores esperados de saída Y. O modelo f possui parâmetros $w = \{w_0, w_1, w_2, \ldots, w_j\}$ e uma função de ativação q. Uma exemplo de escolha desta função de ativação q é a função logística. Logo, o modelo f é definido pela Equação 2.1:

$$f(\mathbf{X}) = q\left(w_0\mathbf{x}_0 + w_1\mathbf{x}_1 + w_2\mathbf{x}_2 + \cdots + w_j\mathbf{x}_j\right) \tag{2.1}$$

Segundo (Hartmann, 2018), o treinamento deste modelo se inicia com valores aleatórios para os seus parâmetros w e, a cada instante de iteração t, novos valores de w são escolhidos para minimizar uma função de perda p. Esta função de perda $p(\mathbf{X}^{(t)})$ deve ser definida para realizar uma comparação entre a previsão $f(\mathbf{X}^{(t)})$ do modelo e a resposta correta $\mathbf{Y}^{(t)}$ encontrada no *dataset*. De acordo com (Hartmann, 2018), uma escolha popular para essa função de perda é o erro quadrático, apresentado abaixo pela Equação 2.2:

$$p(\mathbf{X}^{(t)}) = (\ f(\mathbf{X}^{(t)}) - \mathbf{Y}^{(t)}\ )^2 \tag{2.2}$$

Na literatura, há diferentes algoritmos capazes de minimizar $p(\mathbf{X}^{(t)})$. De acordo com (Goodfellow *et al.*, 2016) e (Bishop; Nasrabadi, 2006), um desses algoritmos é o Gradiente Descendente. Trata-se de um algoritmo iterativo em que os pesos do modelo são movidos repetidamente para uma direção que otimiza a função do modelo $f(\mathbf{X}^{(t)})$, através da minimização da função de perda $p(\mathbf{X}^{(t)})$.

Portanto, o algoritmo Gradiente Descendente deve ser inicializado com um conjunto arbitrário de pesos w e as funções $f(\mathbf{X}^{(t)})$ e $p(\mathbf{X}^{(t)})$, que devem ser diferenciáveis. Em seguida, começa o processo de otimização iterativa.

Em cada rodada de iteração t, são calculadas as derivadas parciais da função de perda $p(\mathbf{X}^{(t)})$ em relação aos pesos w do modelo. Um fator de escala η, chamado de taxa de aprendizado, é aplicado às derivadas parciais. Cada parâmetro $w_i \in w$ é, então, atualizado repetidamente usando a Equação 2.3:

$$w_i^{(t)} = w_i^{(t-1)} - \eta \times \left(\frac{\partial p}{\partial w_i} \right) \tag{2.3}$$

O algoritmo Gradiente Descendente termina quando as atualizações de $w_i^{(t)}$, mediante a função acima, convergem. Isto significa que as atualizações de pesos em iterações adjacentes, e.g. $w_i^{(t)}$ e $w_i^{(t+1)}$, são praticamente iguais de acordo com uma tolerância definida durante a aplicação do algoritmo. Com os valores finais dos pesos do modelo, o aprendizado pode ser avaliado através do conjunto de dados separado para teste. A função f do modelo passa a ser composta com os pesos $w = \{w_0, w_1, w_2, \ldots, w_j\}$ e, através da comparação de $f(\mathbf{X}^{(t)})$ e $\mathbf{Y}^{(t)}$, o aprendizado do modelo é avaliado.

2.2 Aprendizado Federado

De acordo com (Thonglek *et al.*, 2020; McMahan *et al.*, 2017), a abordagem federada de aprendizado de máquina treina um modelo global em um servidor centralizado usando processamento de modelos locais em diferentes dispositivos. A aplicação do aprendizado federado tem, como vantagens, a utilização dos recursos de processamento dos dispositivos locais e a falta da necessidade de transferência de seus dados brutos para um servidor.

Dentre outros algoritmos de aprendizado federado, existem dois que são muito difundidos na literatura: *Federated Stochastic Gradient Descent* (FedSGD) e *Federated Averaging* (FedAVG). Ambos os algoritmos federados são baseados no aprendizado de modelos que utilizam Gradiente Descendente.

No FedSGD, o modelo global e os modelos locais são treinados da seguinte maneira: em cada rodada de iteração, o servidor transmite o modelo global atual com os parâmetros $w^{(i)}$ para todos os clientes (i é a iteração atual). Cada cliente k, então, calcula o seu vetor gradiente g_k usando seus dados de treinamento locais, de acordo com a Equação 2.4:

$$g_k = \left(\frac{\partial p}{\partial w_0}, \frac{\partial p}{\partial w_1}, \ldots, \frac{\partial p}{\partial w_j} \right) \tag{2.4}$$

Após o cálculo de g_k, este valor é enviado para o servidor. O servidor calcula a média dos gradientes recebidos de todos os clientes e gera o novo modelo global com parâmetros $w^{(i+1)}$, de acordo com a Equação 2.5:

$$w^{(i+1)} = w^{(i)} - \eta \times \sum_{k=1}^{K} g_k^{(i)} \tag{2.5}$$

No algoritmo FedAVG, de acordo com (McMahan *et al.*, 2017), as equações de atualização dos modelos, em cada iteração, são diferentes em relação ao FedSGD, tanto para o modelo global no servidor e quanto para os modelos locais nos dispositivos cliente. Em FedAVG, os dispositivos calculam os seus pesos locais através dos seus

dados, de acordo com a equação:

$$w^{(i+1)} = w^{(i)} - \eta \times g_k^{(i)} \tag{2.6}$$

O servidor agrega os pesos de todos os dispositivos para atualizar os parâmetros do modelo global:

$$w^{(i+1)} = \frac{1}{K} \sum_{k=1}^{K} w_k^{(i+1)} \tag{2.7}$$

Ao fim de cada iteração, o servidor transmite o novo modelo $w^{(i+1)}$ para os clientes. Cada um deles recebe o modelo global mais recente e, novamente, são atualizados os modelos locais $w_k^{(i+2)}$, que serão enviados ao servidor para finalizar mais uma iteração.

2.3 Criptografia AES-GCM

De acordo com (Kim; Seo, 2025; Selent, 2010), o AES é uma cifra de bloco de chave simétrica que foi padronizada pelo *National Institute of Standards and Technology* (NIST) dos Estados Unidos da América. Este tipo de criptografia usa blocos de 128 bits usando chaves de 128, 192 ou 256 bits, realizando 10, 12 ou 14 rodadas, respectivamente. Cada rodada aplica quatro transformações:

- *SubBytes*: Substituição não linear de bytes usando uma S-box fixa.
- *ShiftRows*: Deslocamento cíclico de linhas em uma matriz de bytes 4×4.
- *MixColumns*: Mistura de colunas baseada em polinômios sobre um Corpo de Galois $GF(2^8)$.
- *AddRoundKey*: Aplicação de porta *Exclusive OR* (XOR) com a chave da rodada, derivada da chave principal.

Segundo (Danger *et al.*, 2018), um Corpo de Galois (*Galois Fields* em inglês), mencionado acima, é uma estrutura matemática fundamental na criptografia moderna, representando um conjunto finito de elementos sobre o qual operações aritméticas podem ser realizadas com segurança e eficiência. Eles são amplamente utilizados para "embaralhar" dados (criptografia), e detectar ou corrigir erros (codificação). Eles são denotados como $GF(a^b)$, onde a é um número primo e b é um inteiro positivo. Por exemplo, $GF(2)$ é o campo mais simples, com elementos 0 e 1, que é usado para operações lógicas booleanas; enquanto, $GF(2^8)$ é comumente usado em computação porque seus 256 elementos correspondem exatamente a 1 byte (8 bits) de dados.

Existem também abordagens que tornam o AES mais leve ao modificar o próprio algoritmo principal. Os autores de (Dhanda *et al.*, 2025) propõem o AES-8, que reduz a largura do estado interno para 8 bits e simplifica a Caixa de Substituição (*S-box*) do AES.

A simplificação de S-Boxes do padrão AES foca em reduzir o custo computacional (área de silício, consumo de energia e latência) sem comprometer a segurança, que depende de alta não-linearidade e resistência a ataques diferenciais. Diferente de tais variantes específicas para hardware, este trabalho (Dhanda *et al.*, 2025) foca em implementações de software otimizadas do AES padrão, particularmente no ARM Cortex-M4.

O modo *Counter* (CTR) do AES transforma uma cifra de bloco em uma cifra de fluxo ao criptografar um contador combinado com um vetor de inicialização (IV) único. Para cada bloco, um valor de contador incrementado $J_i = IV||i$ é criptografado usando a chave κ, e o resultado é mesclado via XOR com o bloco de texto simples $B_{p,i}$ para produzir o texto cifrado:

$$C_i = B_i \oplus E_\kappa(J_i)$$

Onde E_κ denota a criptografia AES sob a chave κ, e J_i é o i-ésimo bloco do contador. Como o contador muda para cada bloco e o IV nunca é reutilizado, o CTR fornece confidencialidade, mas requer um mecanismo adicional para integridade.

O AES-GCM combina o AES-CTR com a função de autenticação GHASH sobre $GF(2^{128})$ (McGrew; Viega, 2004). GHASH é uma função de hash (hash universal) utilizada no modo de operação AES-GCM para fornecer autenticação de dados e garantir a integridade da mensagem. No AES-GCM, uma subchave de hash H é derivada criptografando um bloco de zeros com AES. Os dados adicionais e os blocos de texto cifrado são então processados pelo GHASH:

$$X_i \leftarrow (X_{i-1} \oplus B_i) \times H$$

Onde X_i é o estado acumulador do GHASH após o processamento do bloco i, e X_{i-1} é o estado anterior. B_i denota o i-ésimo bloco de 128 bits formado a partir do dado adicional ou do texto cifrado (dependendo da fase do GHASH), e H é a subchave de hash. Um bloco final de comprimento é então anexado antes de realizar o XOR da saída do GHASH com uma criptografia AES para formar a tag de autenticação T. Normalmente, um IV de 96 bits é recomendado para o GCM, e sua estrutura favorável ao paralelismo o torna bem adequado para protocolos de alta velocidade, como o TLS.

3. Monitoramento de redes através da classificação de enlaces

Este capítulo aborda a área de monitoramento de redes. Dentro desta área, existem monitoramentos ativos e monitoramentos passivos de rede. Monitoramentos ativos enviam pacotes (ex: Pings, solicitações) para testar respostas, gerando pequeno tráfego extra da rede. Por outro lado, monitoramentos passivos analisam o tráfego da rede sem interferir nele.

Apesar de não interferir no tráfego da rede, monitoramentos passivos tais como (Tan *et al.*, 2021; Alkenani; Nassar, 2022) usam muito processamento para replicar o tráfego completo dos nós da rede ou acrescentam metadados nos pacotes dos clientes da rede.

O sistema de monitoramento apresentado neste capítulo classifica enlaces de rede, como nos trabalhos (Silva *et al.*, 2023) e (Silva *et al.*, 2026).

A seguir, a Seção 3.1 apresenta uma revisão da literatura sobre o assunto. Logo após, a Seção 3.2 formaliza algumas definições deste problema de monitoramento de redes e a Seção 3.3 apresenta as características de uma proposta de monitoramento de rede baseado na classificação dos enlaces de rede às suas taxas de utilização.

3.1 Revisão da Literatura

Na literatura, existem muitos trabalhos sobre monitoramento de redes. Uma forma de monitoramento de redes é através da coleta ativa de dados de medidas usando inserção de meta dados no tráfego da rede. Um exemplo deste tipo de medição utiliza a técnica de Telemetria *In-Band* de Rede (INT). Segundo o *survey* (Tan *et al.*, 2021), a técnica de INT realiza a medição de desempenho da rede a partir da inserção de alguns *bits* de metadados durante as transmissões de dados entre os nós da rede, encapsulando os pacotes em seus saltos ao longo dos caminhos percorridos. Em alguns desses saltos, há uma retirada de metadados para realizar a análise do monitoramento da rede. O sistema deve conhecer a topologia da rede para entender as informações contidas nas inserções e coletas dos metadados.

O trabalho de (Wang *et al.*, 2024) apresenta um exemplo ainda mais recente que utiliza uma técnica de INT, de forma inteligente, para realizar o monitoramento de rede através da uma Rede Gêmea Digital (Digital Twin Network – DTN). Segundo (Wu *et al.*, 2021), uma DTN simula o comportamento de uma rede física, de acordo com informações coletadas desta rede física original. Logo, a utilização de DTN é menos invasiva à rede física por estar simulando o comportamento das transmissões entre os nós e inferindo medidas de desempenho. Porém, para a criação do modelo de simulação, a técnica de (Wang *et al.*, 2024) não dispensa o conhecimento de características da rede física, como a sua topologia.

Em outros trabalhos, o monitoramento da rede é realizada através de uma coleta passiva do tráfego da rede. Para que o monitoramento ocorra com sucesso, deve-se

assumir que há o conhecimento específico de características da rede, tais como a topologia e o algoritmo de roteamento.

Este é o caso do trabalho (Uhlig *et al.*, 2006; Fortz; Thorup, 2000). Em (Uhlig *et al.*, 2006), os autores medem as taxas de utilização dos enlaces da rede de forma passiva, calculando-as a partir da informação das quantidades de tráfego geradas nas comunicações entre todos os os nós da rede. Para realizar este cálculo, o sistema de monitoramento dos enlaces deve conhecer o caminho que os pacotes percorrem durante as transmissões de cada nó de origem para cada nó de destino na rede. O conhecimento desses caminhos de pacotes é obtido através da combinação de informações de topologia da rede e o seu algoritmo de roteamento. Em (Fortz; Thorup, 2000), os autores propõem um modelo de otimização para a escolha de novos pesos para o protocolo de roteamento *Open Shortest Path First* (OSPF).

Com o surgimento e a disseminação das SDNs, as aplicações de engenharia de tráfego passaram a ser direcionadas ao Plano de Controle de uma SDN (Akyildiz *et al.*, 2014). O levantamento (Akyildiz *et al.*, 2014) apresenta alguns trabalhos sobre monitoramento e análise de tráfego de rede utilizando uma arquitetura SDN (Tootoonchian *et al.*, 2010; Chowdhury *et al.*, 2014; Al-Fares *et al.*, 2010). Existem trabalhos recentes que propõem a coleta de medições de tráfego de rede utilizando telemetria em SDNs (Chen *et al.*, 2023; Song *et al.*, 2021). Assim como o trabalho desta, os autores de (Tajiki *et al.*, 2021; Li; Oh, 2024) utilizam as medições de tráfego coletadas dos nós da rede para obter informações sobre os enlaces da rede.

Outros trabalhos da literatura sobre monitoramento de redes propõem modelos de ML para inferir ou prever medições de rede, como nos trabalhos (Zilli *et al.*, 2025; AlQiam *et al.*, 2024; Bothra *et al.*, 2023; Shahrbabaki *et al.*, 2022; Ye *et al.*, 2021; Chen *et al.*, 2022; Pappone *et al.*, 2024; Liu *et al.*, 2023).

Os trabalhos (Zilli *et al.*, 2025; AlQiam *et al.*, 2024; Bothra *et al.*, 2023; Shahrbabaki *et al.*, 2022) utilizam medições de tráfego dos nós da rede como entrada para o modelo proposto, mas suas saídas são ações tomadas para otimizar um serviço de rede. Enquanto isso, a proposta geral apresentada neste capítulo, o *LinkClassifier* apresentado no Capítulo 4 e o *NetLinkClassifier* apresentado no Capítulo 5 são modelos que possuem uma saída que representam classificação de enlaces de rede e uma entrada que recebe uma matriz de tráfego, como nos artigos publicados (Silva *et al.*, 2023; Silva *et al.*, 2026).

O trabalho em (Ye *et al.*, 2021) propõe um modelo baseado em Aprendizado por Reforço (RL) para selecionar os fluxos críticos de cada matriz de tráfego e redirecioná-los para rotas mais livres. Tal modelo de RL recebe como entrada um estado baseado em matrizes de tráfego de rede e a função de recompensa possui um componente que representa os valores de utilização dos enlaces da rede. Portanto, o modelo de RL em (Ye *et al.*, 2021) utiliza matrizes de tráfego para avaliar o nível de utilização dos enlaces de rede dado um conjunto de rotas de fluxo. Essa tarefa é semelhante à tarefa do modelo deste capítulo, que classifica o nível dos enlaces de rede dada uma matriz de tráfego.

Os trabalhos (Chen *et al.*, 2022; Pappone *et al.*, 2024; Liu *et al.*, 2023) propõem técnicas de ML para prever a carga nos enlaces da rede e, assim, rebalancear a carga na rede. A semelhança com o presente trabalho é a tarefa inicial de estimar a carga nos enlaces da rede.

A arquitetura SDN, proposta no trabalho do Capítulo 5, inclui uma camada MEC para computar as medições de rede e processar esses dados medidos antes de enviá-los ao controlador SDN, como nos trabalhos (Chen *et al.*, 2023; Shahrbabaki *et al.*, 2022; Al-Fares *et al.*, 2010; Silva *et al.*, 2026). Além disso, a utilização do paradigma FL para o treinamento com dados de matrizes de tráfego, proposta no trabalho do Capítulo 5, também se encontra nos trabalhos (Amoroso *et al.*, 2020; Amoroso *et al.*, 2023; Silva *et al.*, 2026).

Por fim, o modelo *CityZoneClassifier* apresentado no Capítulo 6 se assemelha com trabalhos do estado da arte que propõem soluções para o monitoramento de tráfego utilizando conceitos de Cidades Inteligentes (Smart Cities) ou redes veiculares. Como nos trabalhos (Cai *et al.*, 2022; Munawar; Piantanakulchai, 2024), o trabalho do Capítulo 6 possui o conceito de divisão da cidade em zonas cujos tráfegos são monitorados. O trabalho do Capítulo 6 e os trabalhos (Li *et al.*, 2025; Zhang *et al.*, 2025; Ibrahim; Mokhtar, 2024; Moon; Lim, 2024) compartilham a ideia de treinamento de modelo utilizando o paradigma FL em redes veiculares.

A Tabela 1 apresenta um resumo das características dos trabalhos listados nesta seção. A primeira coluna desta tabela representa a presença de medição de tráfego dos nós da rede nos trabalhos listados; a segunda coluna representa a presença de monitoramento de enlaces ou caminhos da rede; a terceira coluna representa a ausência do uso da informação da topologia da rede (por exemplo, DTN e somatório de tráfegos de mesmo enlace) e a ausência de medição ativa (por exemplo, telemetria); a quarta coluna representa a presença de um modelo de ML; a quinta coluna representa a utilização do paradigma FL no treinamento do modelo apresentado pelo trabalho e a sexta coluna representa a presença de uma camada MEC ou uma camada *Cloud* na arquitetura do trabalho.

Table 1: Caracterização dos trabalhos relacionados

Trabalhos	Medir tráfego dos nós	Monitorar enlaces/ caminhos da rede	Sem a topologia da rede ou medição ativa	ML	FL	MEC/ *Cloud*
(Tan *et al.*, 2021)	✓					
(Wang *et al.*, 2024)	✓	✓				
(Uhlig *et al.*, 2006)	✓	✓				
(Fortz; Thorup, 2000)	✓	✓				
(Tootoonchian *et al.*, 2010)	✓					
(Chowdhury *et al.*, 2014)	✓					
(Al-Fares *et al.*, 2010)	✓					✓
(Chen *et al.*, 2023)	✓					✓
(Song *et al.*, 2021)	✓					
(Tajiki *et al.*, 2021)	✓	✓				
(Li; Oh, 2024)	✓	✓				
(Zilli *et al.*, 2025)	✓	✓		✓		
(AlQiam *et al.*, 2024)	✓	✓		✓		
(Bothra *et al.*, 2023)	✓	✓		✓		
(Shahrbabaki *et al.*, 2022)	✓	✓		✓		✓
(Ye *et al.*, 2021)	✓	✓		✓		
(Chen *et al.*, 2022)	✓	✓		✓		
(Pappone *et al.*, 2024)	✓	✓		✓		
(Liu *et al.*, 2023)	✓	✓		✓		
(Amoroso *et al.*, 2020)	✓			✓	✓	
(Amoroso *et al.*, 2023)	✓			✓	✓	
(Cai *et al.*, 2022)	✓	✓				
(Munawar; Piantanakulchai, 2024)	✓	✓				
(Li *et al.*, 2025)	✓	✓		✓		
(Zhang *et al.*, 2025)	✓	✓		✓		
(Ibrahim; Mokhtar, 2024)	✓	✓		✓		
(Moon; Lim, 2024)	✓	✓		✓		
(Silva *et al.*, 2023)	✓	✓	✓	✓		
(Silva *et al.*, 2026)	✓	✓	✓	✓	✓	✓
Proposta geral (neste capítulo)	✓	✓	✓	✓		
LinkClassifier (Capítulo 4)	✓	✓	✓	✓		
NetLinkClassifier (Capítulo 5)	✓	✓	✓	✓	✓	✓
CityZoneClassifier (Capítulo 6)	✓	✓	✓	✓	✓	✓

A seguir, a Seção 3.2 apresenta mais detalhes de alguns contextos nos quais é importante realizar o monitoramento da rede sem a informação da sua topologia e do algoritmo de roteamento da rede. Enquanto, a Seção 3.3 explica como esse monitoramento é realizado.

3.2 Formulação do problema

Numa rede R, são encontrados N nós e K enlaces. Nesta rede R, os pacotes gerados por uma demanda de tráfego $x_{i,j}$ partem de um nó de origem i e viajam até um nó de destino j, através de um caminho que passa por um enlace k. Se os caminhos de pacotes de múltiplas demandas de tráfego passarem pelo mesmo enlace k e a soma dessas demandas resultar em um valor alto, o enlace k pode ficar congestionado. O congestionamento de pacotes em um enlace pode levar a problemas na rede, tais como aumento do atraso (*delay*), perda de pacotes, variação do atraso (*jitter*), etc.

Para evitar problemas de rede devido ao congestionamento de pacotes, é necessário medir frequentemente o desempenho de cada enlace na rede. Um exemplo de medida de desempenho para um enlace de rede k é a taxa de utilização y_k, que pode ser definida como sendo a porcentagem da capacidade total do enlace que está sendo efetivamente consumida pelo tráfego de dados passante neste enlace. Além de avaliar o desempenho da rede, esta medida é um indicador crítico da qualidade de conexão entre seus nós e da necessidade de futuros *upgrades* na rede.

Através do conhecimento da topologia de rede e do algoritmo de roteamento, é possível determinar por quais enlaces de rede os pacotes de uma demanda $x_{i,j}^{(t)}$ passam, dado o par de nós $\{i, j\}$ em qualquer instante t. Consequentemente, todos os valores $y_k^{(t)}$ podem ser calculados em cada instante de tempo t para realizar o monitoramento completo da rede (Uhlig *et al.*, 2006).

Outras formas de medir o tráfego que passa pelos enlaces de rede são através da técnica de INT(Tan *et al.*, 2021) e através da criação de um modelo DTN (Wu *et al.*, 2021). Apesar destas técnicas serem amplamente aplicadas em trabalhos do estado da arte, elas assumem que a função de monitoramento da rede deve conhecer a topologia desta rede ou o seu algoritmo de roteamento.

Se o monitoramento de uma rede R for realizado, por exemplo, através de uma VNF ou através de uma aplicação de *Software* baseada na arquitetura de microsserviços, é provável que esta função de monitoramento esteja isolada do sistema e não possua o conhecimento da topologia ou do algoritmo de roteamento de R.

Isto porque NFVs e microsserviços podem contar com uma *Cloud* pública que se trata de uma infraestrutura de *datacenters* e servidores localizados muito distantes da rede analisada. Essa infraestrutura *Cloud* pública é acessada via Internet e disponibilizada por empresas privadas. Além disso, no caso de aplicações baseadas na arquitetura de microsserviços, a ideia é diminuir ao máximo as tarefas a serem realizadas e o processamento delas (Blinowski *et al.*, 2022; Alshuqayran *et al.*, 2016; Kaur *et al.*, 2022).

Por isso, deve ser definida uma outra forma de monitoramento passivo dos enlaces da rede R para este caso da aplicação responsável não possuir a informação da topologia ou do algoritmo de roteamento dos pacotes de R. A Figura 1 apresenta a rede R com o seu monitoramento sendo realizado por uma VNF.

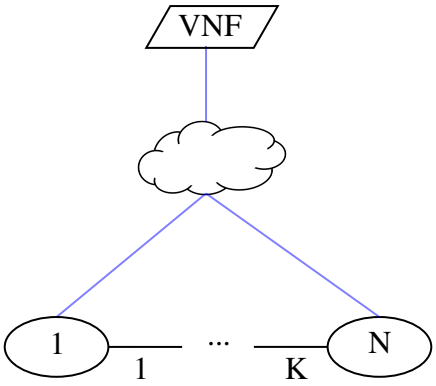


Figure 1: Rede R com o monitoramento realizado por uma VNF.

3.3 Proposta de solução do problema

Para solucionar o problema apresentado na seção anterior, a ideia inicial é criar um modelo de monitoramento da rede R que aprende a classificar os K enlaces, a cada instante de tempo t, usando como entrada as quantidades de demanda de tráfego $\mathbf{x}_{i,j}^{(t)}$ entre todos os nós de R, ou seja, $1 \leq i \leq N$ e $1 \leq j \leq N$.

Como em (Varyani *et al.*, 2020), a classificação de enlaces realizada por este modelo significa obter a inferência de rótulos de $\hat{\mathbf{y}}_k^{(t)}$ que representam o nível aproximado da taxa de utilização $y_k^{(t)}$ de cada enlace k tal que $1 \leq k \leq K$. A Figura 2 mostra o esquema das entradas e saídas do modelo de classificação.

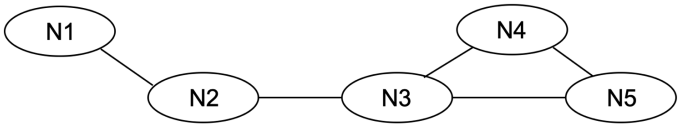


Figure 2: Modelo de classificação da rede.

No modelo de classificação da rede demonstrado pela Figura 2, os valores de demanda de tráfego $\mathbf{x}_{i,j}^{(t)}$ dos nós da rede podem ser organizados numa matriz de tráfego $\mathbf{X}^{(t)}$, como em (Zilli *et al.*, 2025; Pappone *et al.*, 2024; Liu *et al.*, 2023).

Uma matriz de tráfego é uma estrutura de dados que possui os valores de medidas de tráfego dos clientes da rede, organizados nas posições de seus elementos internos. Dessa forma, as quantidades de linhas e colunas de uma matriz de tráfego são iguais à quantidade de nós da rede. Então, as coordenadas de linha e coluna de uma medida de tráfego que se encontra nesta matriz correspondem às informações de origem e destino deste tráfego medido. A Figura 3 mostra um exemplo da montagem de uma matriz de tráfego numa rede com 5 nós.

A estratégia de utilizar rótulos de classificação na saída do modelo, em vez de



(a)

Matriz de tráfego **X**

De \ Para	N1	N2	N3	N4	N5
N1	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{1,4}$	$x_{1,5}$
N2	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	$x_{2,4}$	$x_{2,5}$
N3	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$	$x_{3,5}$
N4	$x_{4,1}$	$x_{4,2}$	$x_{4,3}$	$x_{4,4}$	$x_{4,5}$
N5	$x_{5,1}$	$x_{5,2}$	$x_{5,3}$	$x_{5,4}$	$x_{5,5}$

(b)

Figure 3: Exemplo de uma rede com 5 nós: (a) topologia e (b) matriz de tráfego.

valores exatos de utilização de enlace, economiza recursos de armazenamento e processamento dos dados, além de ser uma solução mais segura por permitir uma codificação na interpretação destes rótulos. De acordo com (Kakkavas *et al.*, 2024) e (Xu *et al.*, 2021), as demandas de tráfego dos nós podem ser inferidas utilizando os valores de utilização dos enlaces. Essa inferência seria mais difícil de ser realizada se o sistema de monitoramento armazenasse apenas rótulos de classificação.

A fim de particionar a solução inicial, a proposta deste modelo de classificação da rede é dividida pela elaboração de K instâncias de um modelo de ML, cada um para obter o rótulo de classificação $\mathbf{y}_k^{(t)}$ de um enlace k da rede. A Figura 4 apresenta o esquema das entradas e saídas do modelo de classificação da rede e de seus componentes para cada enlace da rede.

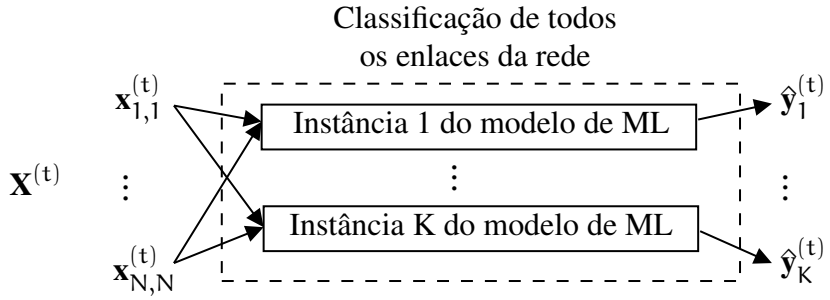


Figure 4: Instâncias do modelos de ML para a classificação de cada enlace k da rede.

As instâncias do modelo de ML para classificação de enlace devem ser treinados separadamente. O *dataset* de treino de cada modelo de ML deve ser supervisionado e suas amostras devem conter a matriz de tráfego $\mathbf{X}^{(t)}$ e o rótulo de classificação $\mathbf{y}_k^{(t)}$ do enlace k correspondente ao modelo.

O Capítulo 4 e o Capítulo 5 também apresentam problemas e soluções de monitoramento de enlaces de rede no contexto de SDN. Enquanto, o Capítulo 6 apresenta problemas e soluções semelhantes ao monitoramento de enlaces de rede para o contexto de ITS.

4. Inferência da utilização de enlaces para caminhos de rede a partir de matrizes de tráfego

Este capítulo apresenta um ambiente de rede baseado em SDN no qual a solução de monitoramento proposta no Capítulo 3 pode ser aplicada.

Além disso, o trabalho deste capítulo propõe um modelo chamado *LinkClassifier* que obtém, em sua saída, um rótulo de classificação para um enlace de rede com base em seu valor aproximado de utilização. O modelo *LinkClassifier* utiliza uma matriz de tráfego como entrada.

O modelo *LinkClassifier* é validado por meio de experimentos utilizando dois conjuntos de dados distintos. Nesses experimentos, as métricas de desempenho de classificação obtidas para o modelo proposto são apresentadas e comparadas com as de outros experimentos, que variam alguns parâmetros de treinamento. Após avaliar o modelo, outro experimento é conduzido para demonstrar como o modelo *LinkClassifier* pode ser utilizado para apoiar a aplicação de um serviço de roteamento de rede.

Dessa forma, as contribuições do trabalho deste capítulo são:

- A proposta de um modelo de classificação que recebe uma matriz de tráfego como entrada e calcula um rótulo de classificação para um enlace de rede, representando o nível de utilização do respectivo enlace.
- A avaliação deste modelo de classificação por meio de experimentos utilizando dois conjuntos de dados contendo demandas de tráfego.
- Uma comparação de desempenho entre as classificações utilizando o modelo proposto e outros modelos de classificação tradicionais na literatura.
- Um experimento de simulação para explicar como o modelo de classificação pode ser usado para auxiliar o serviço de roteamento de pacotes de uma rede.

A organização das seções deste capítulo é a seguinte: a Seção 4.1 apresenta a formulação deste problema; a Seção 4.2 descreve o modelo e o método de solução do problema; a Seção 4.3 demonstra o desempenho desta solução através de resultados de experimentos; e a Seção 4.4 apresenta uma conclusão do conteúdo deste capítulo.

4.1 Formulação do problema

Em uma rede de N nós e K enlaces, os pacotes gerados por uma demanda de tráfego x viajam do nó de origem ao nó de destino através de alguns dos K enlaces. O nível de utilização y de um enlace é o resultado da quantidade de dados transferidos pelos pacotes que passam por este enlace e sua capacidade máxima.

Seja $\mathbf{X}^{(t)}$ uma matriz de tráfego contendo todas as demandas de tráfego $\mathbf{x}$ para um mesmo instante de tempo t. Em cada posição de linha e coluna (i,j) da matriz $\mathbf{X}^{(t)}$, existe um valor $\mathbf{x}_{i,j}$ que representa a quantidade de demanda de tráfego na rede originada no nó i e destinada ao nó j. Portanto, as matrizes de tráfego de uma rede com N nós são de dimensão $N \times N$.

Seja $Y^{(t)}$ um vetor contendo todos os valores de utilização y dos enlaces. Cada elemento de $Y^{(t)}$ possui um valor $y_k^{(t)}$que representa o nível de utilização de um enlace k em um determinado instante de tempo t. A rede determina o caminho dos pacotes para cada demanda de tráfego $\mathbf{x}$. Portanto, o valor $y_k^{(t)}$ é igual à soma dos valores das demandas de tráfego cujos pacotes passam pelo mesmo enlace k em um determinado instante de tempo t.

Em alguns serviços de roteamento de rede, como o exemplificado em (Uhlig *et al.*, 2006), é possível determinar por quais enlaces da rede os pacotes de cada demanda $\mathbf{x}_{i,j}$ passam, em qualquer instante de tempo. Conhecendo os caminhos e a intensidade de todas as demandas $\mathbf{x}_{i,j}$ em cada tempo t, todos os valores y_k podem ser calculados em cada instante de tempo.

Por outro lado, se um sistema de monitoramento de enlaces de rede não conhece o algoritmo de roteamento de pacotes e necessita apenas de um nível aproximado de utilização para esses enlaces, como em (Varyani *et al.*, 2020) e em (Boucetta *et al.*, 2021), então é necessário treinar um modelo para classificar os K enlaces usando rótulos $\mathbf{y}_k$ baseados nos níveis y_k de utilização dos enlaces.

Portanto, o modelo de classificação de enlaces proposto, conforme ilustrado na Figura 5, consiste em duas etapas: a primeira gera um valor estimado $\hat{y}_k$ utilizando uma matriz de tráfego $\mathbf{X}^{(t)}$ como entrada, e a segunda estima um rótulo $\hat{\mathbf{y}}_k$ baseado em $\hat{y}_k$ para cada enlace de rede k em um determinado tempo t.

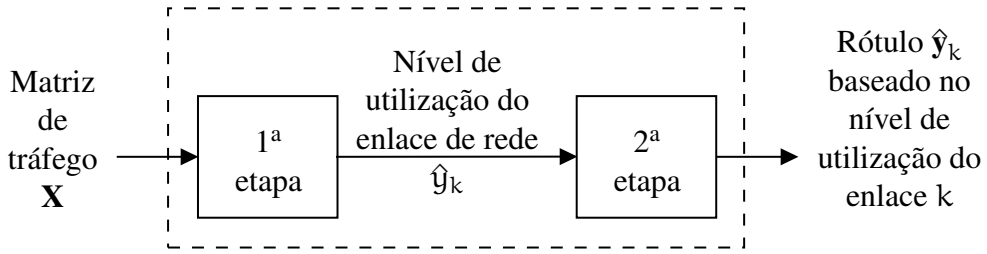


Figure 5: Modelo de classificação de enlace de rede usando uma matriz de tráfego como entrada.

4.2 *LinkClassifier*

O modelo *LinkClassifier* propõe uma classificação de k enlaces em uma rede com base em suas medidas de demanda de tráfego $\mathbf{x}_{i,j}$. As medições $\mathbf{x}_{i,j}$, utilizadas como entrada para o modelo *LinkClassifier*, são organizadas na forma de uma matriz de tráfego $\mathbf{X}$. A classificação resultante, encontrada na saída do modelo *LinkClassifier*, é um rótulo

$\hat{\mathbf{y}}_k$ baseado no nível de utilização estimado $\hat{y}_k$ de um enlace k na rede. O conjunto Z de rótulos possíveis para classificar o nível de utilização do enlace k é um parâmetro do modelo *LinkClassifier*. Em qualquer instante t, o resultado do modelo *LinkClassifier* é $\hat{y}_k^{(t)} = z$, de modo que $z \in Z$.

Como mencionado acima, o classificador *LinkClassifier* possui duas etapas: a primeira etapa infere o valor de utilização $\hat{y}_k$ de um enlace de rede a partir de uma matriz de tráfego $\mathbf{X}$ e a segunda etapa classifica este valor de utilização do enlace usando um rótulo $\hat{\mathbf{y}}_k$.

A primeira etapa do *LinkClassifier* é representada pela Equação 4.1, na qual uma regressão linear é utilizada para estimar a taxa de utilização do enlace k, $\hat{y}_k$. Os coeficientes $w_{1,i,j,k}$ são usados para indicar se alguma porção do tráfego da demanda $\mathbf{x}_{i,j}$ passa pelo enlace k. Em cenários com caminhos únicos de ponta a ponta e assumindo zero perda de pacote, $w_{1,i,j,k}$ receberá valores 1 ou 0, indicando se o caminho de i para j utiliza o enlace k, e tal informação pode ser derivada do protocolo de roteamento utilizado na rede. No entanto, em cenários mais sofisticados que utilizam balanceamento de carga por pacote ou protocolos de roteamento de múltiplos caminhos, apenas uma parte do tráfego de i para j pode usar o enlace k, portanto $w_{1,i,j,k}$ pode ser ajustado para um valor diferente. Além disso, nossa solução é projetada para ser independente de informações derivadas de protocolos de roteamento. O coeficiente ϵ_k representa todos os fatores residuais de conta e possíveis erros de medição. Nos experimentos deste trabalho, seu valor é sempre ajustado para zero ou um valor muito próximo a ele.

A forma como os coeficientes $w_{1,i,j,k}$ da Equação 4.1 são ajustados é através da inspeção de um conjunto de dados que deve conter uma quantidade adequada de instâncias para $\mathbf{x}_{i,j}$ e y_k, obtidos para a mesma rede na qual este método será aplicado — ou seja, mesma topologia, mesmo algoritmo de roteamento e mesmo perfil para demandas de ponta a ponta.

$$\hat{y}_k = \epsilon_k + \sum_{i,j}^{N} w_{1,i,j,k} \times \mathbf{x}_{i,j} \tag{4.1}$$

A segunda etapa do *LinkClassifier* recebe o valor estimado da utilização do enlace k, $\hat{y}_k$, como entrada e estima uma classificação do enlace k usando um rótulo $\hat{\mathbf{y}}_k$. Para realizar isso, os valores $w_{2,z,k}$ são usados como limiares de utilização do enlace a fim de definir faixas Z para associações de rótulos. Portanto, a escolha de um rótulo $\hat{\mathbf{y}}_k$ pode ser alcançada comparando $\hat{y}_k$ e os valores $w_{2,z,k}$. A Equação 4.2 mostra a representação de $\hat{y}_k$.

Para ajustar os limiares $w_{2,z+1,k}$ para cada enlace k, o tipo de modelo utilizado nesta etapa é a Máquina de Vetores de Suporte (*Support Vector Machine* – SVM). O SVM é um método de ML supervisionado que pode inferir um hiperplano para separar duas classes diferentes de instâncias n-dimensionais. No nosso caso, o modelo SVM contém um *kernel* parametrizado como um polinômio de grau zero para inferir as diferentes faixas de valores para $w_{2,z,k}$, a fim de associar um rótulo diferente $\hat{\mathbf{y}}_k$ para cada faixa z.

$$\hat{\mathbf{y}}_k = \begin{cases} Z & \text{, if } \hat{y}_k \geq w_{2,Z,k} \\ \vdots & \vdots \\ z & \text{, if } w_{2,z+1,k} > \hat{y}_k \geq w_{2,z,k} \\ \vdots & \vdots \\ 1 & \text{, if } w_{2,1,k} > \hat{y}_k \geq 0 \end{cases} \tag{4.2}$$

O valor do rótulo $\hat{\mathbf{y}}_k$ encontrado na saída desta segunda etapa é também a saída final do modelo *LinkClassifier* para um determinado enlace k. Cada instância do modelo proposto é, assim, aplicada K vezes, onde K representa o número de enlaces unidirecionais na rede.

4.2.1 Pré-processamento e montagem de um conjunto de dados

O caso ideal é ter um conjunto de dados contendo amostras com as seguintes três informações para todos os instantes de tempo t: uma matriz de tráfego $\mathbf{X}^{(t)}$, o valor da medida de utilização do enlace $y_k^{(t)}$ e um rótulo $\mathbf{y}_k^{(t)}$ para classificar este mesmo enlace com base em algum critério de qualidade, conforme descrito na seção anterior. Se o conjunto de dados contiver apenas amostras com as duas primeiras informações, ou seja, $\mathbf{X}^{(t)}$ e $y_k^{(t)}$, e o usuário do modelo *LinkClassifier* possuir todas as informações necessárias para implementar uma política de rotulagem associada a $\mathbf{X}^{(t)}$, $y_k^{(t)}$, então tal procedimento de rotulagem deve ser realizado para montar um conjunto de dados ideal cujas amostras contenham todas as três informações: $\mathbf{X}^{(t)}$, $y_k^{(t)}$ e $\mathbf{y}_k^{(t)}$.

Em outro cenário, se um conjunto de dados contiver amostras apenas com matrizes de tráfego $\mathbf{X}^{(t)}$, o usuário do modelo *LinkClassifier* precisará acessar as informações do protocolo de roteamento da rede para inferir os valores de $\mathbf{y}_k^{(t)}$ baseados em $\mathbf{X}^{(t)}$. Os autores de (Uhlig *et al.*, 2006) exemplificam o processo de cálculo dos níveis de utilização $y_k^{(t)}$ para cada $\mathbf{X}^{(t)}$ quando o algoritmo de roteamento que escolhe o caminho mais curto da rede de i para j utiliza o algoritmo de *Dijkstra*. Após calcular $y_k^{(t)}$, um procedimento de rotulagem deve ser realizado como no caso anterior.

Considerando que um conjunto de dados finalmente contenha os valores $\mathbf{X}^{(t)}$, $y_k^{(t)}$ e $\mathbf{y}_k^{(t)}$em todas as instâncias, a última tarefa envolve a divisão do conjunto de dados em duas partes: uma contendo amostras com as informações $\mathbf{X}^{(t)}$ e $y_k^{(t)}$ e a outra contendo amostras com as informações $y_k^{(t)}$ e $\mathbf{y}_k^{(t)}$. Tal divisão é necessária para treinar cada etapa do modelo *LinkClassifier* separadamente.

4.2.2 Treinamento

O modelo *LinkClassifier* treina cada uma de suas etapas separadamente. Portanto, ele requer dois conjuntos de dados supervisionados, conforme indicado na Seção 4.2.1, e dois processos de treinamento diferentes para cada etapa.

A primeira etapa é treinada utilizando um conjunto de dados supervisionado no qual cada amostra, referente a um instante de tempo t, contém uma matriz de tráfego $\mathbf{X}^{(t)}$ e

um valor de utilização $y_k^{(t)}$ de um enlace k na rede. Durante o treinamento do modelo de regressão linear da primeira etapa, os valores dos coeficientes $w_{1,i,j,k}$ e da constante ϵ_k são encontrados utilizando o Método dos Mínimos Quadrados ((Bro *et al.*, 2002)). Esses valores devem minimizar a função de erro quadrático entre o valor $\hat{y}_k^{(t)}$, computado por meio da Equação 4.1, e o valor $y_k^{(t)}$ encontrado no *dataset* supervisionado. A Equação 4.3 descreve este procedimento de otimização.

$$\begin{aligned}\vec{w}_1^{\,opt} &= \arg\min_{\vec{w}_1} \sum^{t} \left(\hat{y}_k^{(t)} - y_k^{(t)}\right)^2 \\ &= \arg\min_{\vec{w}_1} \sum^{t} \left(\left(\epsilon_k + \sum_{i,j}^{N} w_{1,i,j,k} \times \mathbf{x}_{i,j}^{(t)}\right) - y_k^{(t)}\right)^2\end{aligned} \tag{4.3}$$

tal que:

$$\vec{w}_1 = \left(\epsilon_k, w_{1,1,1,k}, w_{1,1,2,k}, \ldots, w_{1,i,j,k}, \ldots, w_{1,N,N,k}\right)$$

A Equação 4.4 mostra como computar o valor ótimo dos coeficientes $w_{1,i,j,k}$ e da constante ϵ_k.

$$\vec{w}_1 = \left(\mathbf{X}_{mod}^{T}\,\mathbf{X}_{mod}\right)^{-1}\mathbf{X}_{mod}\,y_{mod}$$

tal que:

$$\mathbf{X}_{mod} = \begin{pmatrix} 1, \mathbf{x}_{1,1}^{(0)}, \mathbf{x}_{1,2}^{(0)}, \ldots, \mathbf{x}_{i,j}^{(0)}, \ldots, \mathbf{x}_{N,N}^{(0)} \\ 1, \mathbf{x}_{1,1}^{(1)}, \mathbf{x}_{1,2}^{(1)}, \ldots, \mathbf{x}_{i,j}^{(1)}, \ldots, \mathbf{x}_{N,N}^{(1)} \\ \vdots \\ 1, \mathbf{x}_{1,1}^{(t)}, \mathbf{x}_{1,2}^{(t)}, \ldots, \mathbf{x}_{i,j}^{(t)}, \ldots, \mathbf{x}_{N,N}^{(t)} \\ \vdots \end{pmatrix} \tag{4.4}$$

$$y_{mod} = \begin{pmatrix} y^{(0)} \\ y^{(1)} \\ \vdots \\ y^{(t)} \\ \vdots \end{pmatrix}$$

A segunda etapa do modelo *LinkClassifier* é treinada utilizando um conjunto de dados supervisionados no qual cada amostra contém um valor de utilização $y_k^{(t)}$ de um enlace k e um rótulo de classificação $\mathbf{y}_k^{(t)}$. Os rótulos $\mathbf{y}_k^{(t)}$ representam faixas de valores de utilização do enlace $y_k^{(t)}$. Durante o treinamento do modelo SVM na segunda etapa, os valores $w_{2,z,k}$ são determinados. Eles são os valores de fronteira entre as faixas de $y_k^{(t)}$ que minimizam o erro entre o valor $\hat{\mathbf{y}}_k$, computado pela Equação 4.2, e o valor $\mathbf{y}_k^{(t)}$ encontrado no conjunto de dados supervisionado.

Como a dimensão dos valores de entrada y_k para o SVM da segunda etapa é unitária, os valores $y_k^{(t)}$ e $\mathbf{y}_k^{(t)}$ das amostras do *dataset* supervisionado podem ser organizados em um hiperplano H com apenas um eixo, ordenados por $y_k^{(t)}$. Em uma classificação binária, as distâncias entre esses pontos e o ponto de origem do hiperplano H são usadas para definir um valor $w_{2,z,k}$ tal que amostras com diferentes valores $\mathbf{y}_k^{(t)}$ sejam separadas neste hiperplano H.

Em classificações com múltiplas classes, uma abordagem de SVM chamado One Against All (Um Contra Todos) (Liu; Zheng, 2005) divide este problema em múltiplas classificações binárias. Por exemplo, em uma classificação com três valores de rótulo possíveis, três valores de limiar são escolhidos: $w_{2,0,k}$, $w_{2,1,k}$ e $w_{2,2,k}$. Se $y_k^{(t)} \geq w_{2,2,k}$, então $\mathbf{y}_k^{(t)}$ recebe o valor "3"; se $w_{2,2,k} > y_k^{(t)} \geq w_{2,1,k}$, $\mathbf{y}_k^{(t)}$ recebe o valor "2" e se $w_{2,1,k} > y_k^{(t)} \geq 0$, $\mathbf{y}_k^{(t)}$ recebe o valor "1".

4.2.3 Utilização do modelo treinado

Após o treinamento de todas as K instâncias do modelo *LinkClassifier*, todos os rótulos de classificação $\mathbf{y}_k^{(t)}$ para os enlaces k podem ser encontrados a partir de uma matriz de tráfego $\mathbf{X}^{(t)}$. O Algoritmo 1 demonstra como essa classificação é realizada para cada enlace k.

Algorithm 1 *LinkClassifier*

Entradas: $\mathbf{X}^{(t)}$, $\vec{w}_1 = \left(\epsilon_k, w_{1,1,1,k}, \ldots, w_{1,i,j,k}, \ldots, w_{1,N,N,k}\right)$, $\vec{w}_2 = \left(w_{2,1,k}, \ldots, w_{2,z,k}, \ldots, w_{2,Z,k}\right)$.

Saída: $\hat{\mathbf{y}}_k^{(t)}$

1. $\hat{y}_k = \epsilon_k + \sum_{i,j}^{N} w_{1,i,j,k} \times \mathbf{x}_{i,j}$
2. **se** $\hat{y}_k \geq w_{2,Z,k}$ **então**
3. | $\hat{\mathbf{y}}_k^{(t)} = Z$
4. **fim**
5. **para** $w_{2,z,k}$ **em** $(w_{2,1,k}, \ldots, w_{2,z,k}, \ldots, w_{2,Z-1,k})$ **faça**
6. | **se** $w_{2,z+1,k} > \hat{y}_k \geq w_{2,z,k}$
7. | | $\hat{\mathbf{y}}_k^{(t)} = z$
8. | **fim**
9. **fim**
10. **se** $w_{2,1,k} > \hat{y}_k \geq 0$ **então**
11. | $\hat{\mathbf{y}}_k^{(t)} = 0$
12. **fim**

Portanto, a fim de classificar todos os k enlaces na rede utilizando os rótulos $\mathbf{y}_k^{(t)}$, o modelo *LinkClassifier* deve ser aplicado K vezes, uma para cada enlace k.

4.3 Resultados e desempenho de classificação

Para avaliar o modelo *LinkClassifier*, são conduzidos experimentos utilizando diferentes conjuntos de dados, os quais são apresentados na Seção 4.3.1. O desempenho deste modelo de classificação é avaliado por métricas definidas na Seção 4.3.2. A Seção 4.3.3 compara o desempenho do modelo proposto por meio de experimentos. A Seção 4.3.4 exemplifica como a classificação de enlaces, obtida pelo modelo proposto, pode auxiliar no roteamento de pacotes de rede.

4.3.1 Conjuntos de dados

Os experimentos utilizam 2 conjuntos de dados[1]: *Abilene* e *Nobel-Germany*. O conjunto de dados *Abilene* contém 48.384 matrizes de tráfego de uma rede com 12 nós e 30 enlaces (Zhang *et al.*, 2003). O conjunto de dados *Nobel-Germany* contém 288

[1]Disponíveis nos links da web <https://sndlib.put.poznan.pl/home.action> e <https://www.cs.utexas.edu/~yzhang/research/AbileneTM/>

matrizes de tráfego de uma rede com 17 nós e 52 enlaces (Orlowski *et al.*, 2010).

Esses conjuntos de dados são originalmente não supervisionados. Eles contêm as matrizes de tráfego $\mathbf{X}^{(t)}$ para a entrada do modelo *LinkClassifier*, mas não contêm os valores de utilização de enlace $y^{(t)}$ e os rótulos de classificação $\mathbf{y}^{(t)}$ que são usados para treinar o modelo. Assim, esses dois conjuntos de dados devem passar por alguns processos de cálculo e rotulagem para que sejam capazes de ser utilizados no treinamento e teste do modelo..

O processo de cálculo dos conjuntos de dados obtém os valores de utilização $y^{(t)}$ dos enlaces da rede, de acordo com cada matriz de tráfego $\mathbf{X}^{(t)}$. Este cálculo envolve o conhecimento da topologia da rede e seu algoritmo de roteamento. As demandas de origem e destino, representadas por cada posição de uma matriz de tráfego $\mathbf{X}^{(t)}$, devem ser somadas se o caminho de seus pacotes passar pelo mesmo enlace k. O resultado desta soma deve ser dividido pela capacidade máxima do enlace k. A partir disso, obtém-se o valor $y_k^{(t)}$, que representa a taxa de utilização do enlace k. O processo de rotulagem escolhe uma regra para classificar cada valor $y_k^{(t)}$, transformando-o em $\mathbf{y}_k^{(t)}$. Ao realizar um processamento análogo para todos os enlaces da rede e utilizando todas as matrizes de tráfego, obtém-se um conjunto de dados supervisionado cujas amostras são os valores $\mathbf{X}^{(t)}$, $y_k^{(t)}$ e $\mathbf{y}_k^{(t)}$.

4.3.2 Avaliação de desempenho

A fim de avaliar o modelo proposto, utilizam-se os dados de teste como entrada para obter os rótulos de classificação inferidos $\hat{\mathbf{y}}^{(t)}$, os quais são comparados aos rótulos esperados correspondentes $\mathbf{y}^{(t)}$.

De acordo com (Fawcett, 2006), métricas de avaliação de modelo, denominadas acurácia, precisão, revocação (*recall*) e *F1-score*, podem ser calculadas a partir da comparação entre a resposta da aplicação do modelo $\hat{\mathbf{y}}^{(t)}$ e seu valor de referência $\mathbf{y}^{(t)}$ no conjunto de dados supervisionado.

Portanto, experimentos foram conduzidos para treinar e testar o *LinkClassifier*. Nesses experimentos, o conjunto de dados, o número de classes e os parâmetros de treinamento são variados.

Para a comparação de desempenho de classificação nos diferentes cenários apresentados na Seção 4.3.3, foram realizados experimentos utilizando os dois conjuntos de dados (*datasets*). Para maximizar o uso dos dados disponíveis na medição do desempenho, foi utilizada uma Validação Cruzada de *Fold* igual a 10 (*10-Fold Cross-Validation*) no *dataset* Abilene. No *dataset* Nobel-Germany, utilizou-se uma validação cruzada de 5 *Folds* devido ao seu pequeno tamanho de amostra.

Além disso, os experimentos foram executados com as mesmas configurações 20 vezes. Dessa forma, foi possível obter uma breve análise estatística através dos valores de média, mediana, primeiro percentil e terceiro percentil da distribuição dos valores de cada métrica.

4.3.3 Resultados comparados com modelos de classificação tradicionais

Nesta Seção, os resultados de desempenho obtidos pelo *LinkClassifier* são comparados com os resultados de outros modelos de classificação tradicionais da literatura. Esses modelos, para comparação com o *LinkClassifier*, são Árvore de Decisão (DT), Perceptron Multicamadas (MLP), Regressão Linear (LR), Regressão Logística (LGR) e SVM.

O modelo DT utiliza o índice de *Gini* como critério de decisão, e as divisões ao longo dos ramos desta árvore são sempre feitas com o melhor valor encontrado. O modelo MLP contém 4 camadas densas: a primeira possui um número de neurônios igual ao dobro do tamanho do vetor de entrada (ou seja, $2 \times N \times N$); a segunda camada possui $2 \times N \times N$ neurônios; a terceira camada possui $2 \times N \times N$ neurônios; e a última camada possui um número de neurônios igual ao número de classes. O modelo LR utiliza as equações de otimização do Método dos Mínimos Quadrados para ser treinado. O modelo LGR é treinado utilizando o algoritmo Broyden-Fletcher-Goldfarb-Shanno (LBFGS). O modelo SVM utiliza um kernel polinomial de grau igual a 1 para treinar seus pesos, e a dimensão do hiperplano de entrada do SVM é igual a N^2.

Após o treinamento e teste de todos os modelos, as métricas de desempenho resultantes são comparadas entre si. Em seguida, a comparação dos modelos é repetida, porém variando o número de classes.

A primeira comparação de modelos ocorre quando três classes são definidas: Rótulo 1 representa o nível baixo de utilização de enlace; o Rótulo 2 representa o nível intermediário e o Rótulo 3, o nível alto.

A Tabela 2 ilustra esta comparação de modelos com três classes. Ela mostra que o maior valor de *F1-score* é obtido pelo *LinkClassifier* quando 3 classes são definidas. O valor mais alto de *F1-score* é 0,969, sendo cerca de 4% superior ao segundo maior valor de *F1-score*, que é obtido pelo modelo LGR.

Table 2: Desempenho do *LinkClassifier* e outros modelos (3 classes).

Métrica	DT	MLP	LR	LGR	SVM	**Link-Classifier**
Acurácia	0,994	0,943	0,968	0,997	0,995	**0,999**
Precisão	0,886	0,709	0,698	0,933	0,973	**0,986**
Recall	0,902	0,923	0,804	0,925	0,870	**0,955**
F1-score	0,893	0,734	0,714	0,929	0,899	**0,969**

Na Tabela 2, os valores que representam as métricas do *LinkClassifier* são as médias dos resultados das 20 execuções do experimento com 3 classes. A Figura 6 apresenta um diagrama *Boxplot* (ou diagrama de caixa) sobre as distribuições dos valores das métricas de *LinkClassifier* com 3 classes.

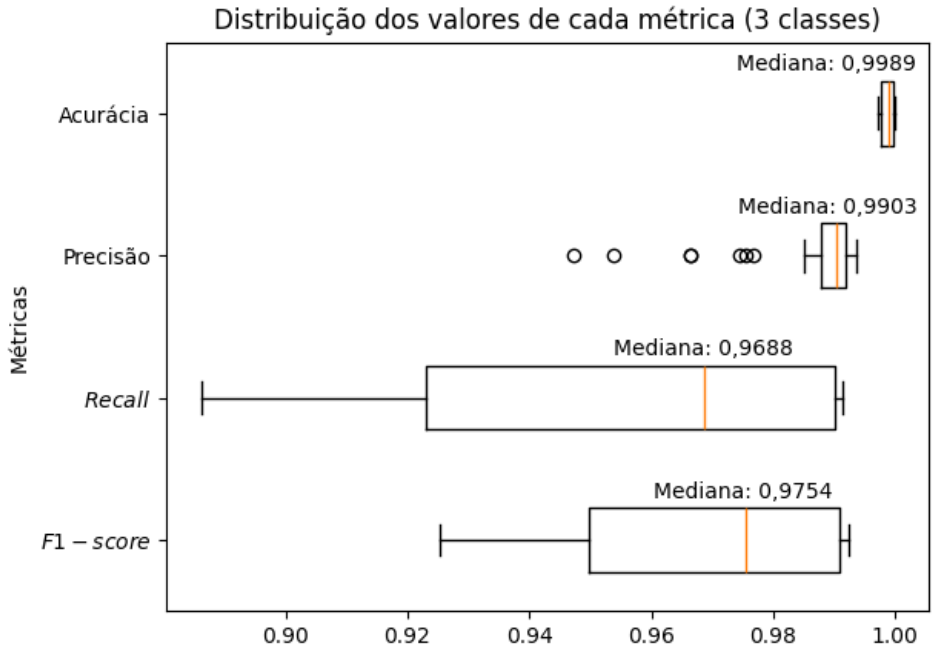


Figure 6: Distribuição dos valores de cada métrica do *LinkClassifier* (3 classes).

Em seguida, a segunda comparação de modelos ocorre quando quatro classes são definidas: Rótulo 1 representa o nível baixo de utilização de enlace; o Rótulo 2 representa o nível médio-baixo; o Rótulo 3, o nível médio-alto e o Rótulo 4, o nível alto.

A Tabela 3 apresenta esta comparação dos modelos com quatro classes. Ela mostra que o maior valor de *F1-score* é obtido pelo *LinkClassifier* quando quatro classes são definidas. O valor mais alto de *F1-score* é 0,958, aproximadamente 6% superior ao segundo maior valor de *F1-score*, obtido pelo modelo DT.

Table 3: Desempenho do *LinkClassifier* e outros modelos (4 classes).

Métrica	DT	MLP	LR	LGR	SVM	**Link-Classifier**
Acurácia	0,995	0,870	0,964	0,996	0,994	**0,998**
Precisão	0,902	0,649	0,656	0,877	0,825	**0,970**
Recall	0,895	0,859	0,794	0,873	0,805	**0,947**
F1-score	0,898	0,653	0,665	0,875	0,814	**0,958**

Na Tabela 3, os valores que representam as métricas do *LinkClassifier* são as médias dos resultados das 20 execuções do experimento com 4 classes. A Figura 7 apresenta um diagrama *Boxplot* sobre as distribuições dos valores das métricas de *LinkClassifier* com 4 classes.

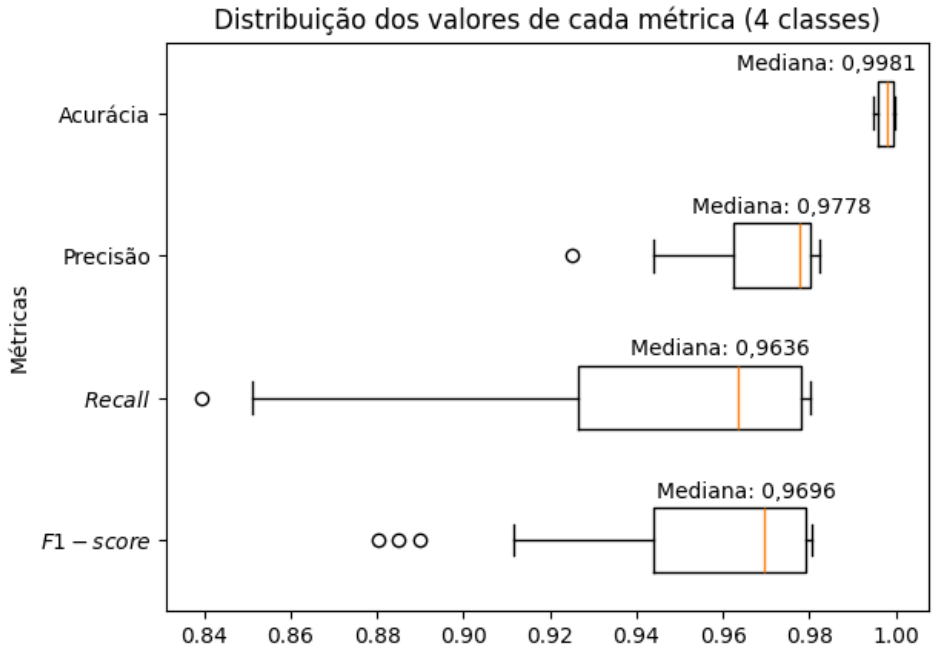


Figure 7: Distribuição dos valores de cada métrica do *LinkClassifier* (4 classes).

Por fim, a terceira comparação de modelos ocorre quando cinco classes são definidas: Rótulo 1 representa o nível baixo de utilização de enlace; o Rótulo 2 representa o nível médio-baixo; o Rótulo 3, o nível médio; o Rótulo 4, o nível médio-alto e o Rótulo 5, o nível alto.

A Tabela 4 apresenta esta comparação dos modelos com cinco classes. Ela mostra que o maior valor de *F1-score* é obtido pelo *LinkClassifier* quando cinco classes são definidas. O valor mais alto de *F1-score* é 0,939, sendo cerca de 11% superior ao segundo maior valor de *F1-score*, que é obtido pelo modelo LGR.

Table 4: Desempenho do *LinkClassifier* e outros modelos (5 classes).

Métrica	DT	MLP	LR	LGR	SVM	**Link-Classifier**
Acurácia	0,958	0,823	0,941	0,991	0,986	**0,996**
Precisão	0,868	0,604	0,625	0,867	0,794	**0,949**
Recall	0,842	0,833	0,725	0,859	0,782	**0,931**
F1-score	0,852	0,614	0,635	0,863	0,787	**0,939**

Na Tabela 4, os valores que representam as métricas do *LinkClassifier* são as médias dos resultados das 20 execuções do experimento com 5 classes. A Figura 8 apresenta um diagrama *Boxplot* sobre as distribuições dos valores das métricas de *LinkClassifier* com 5 classes.

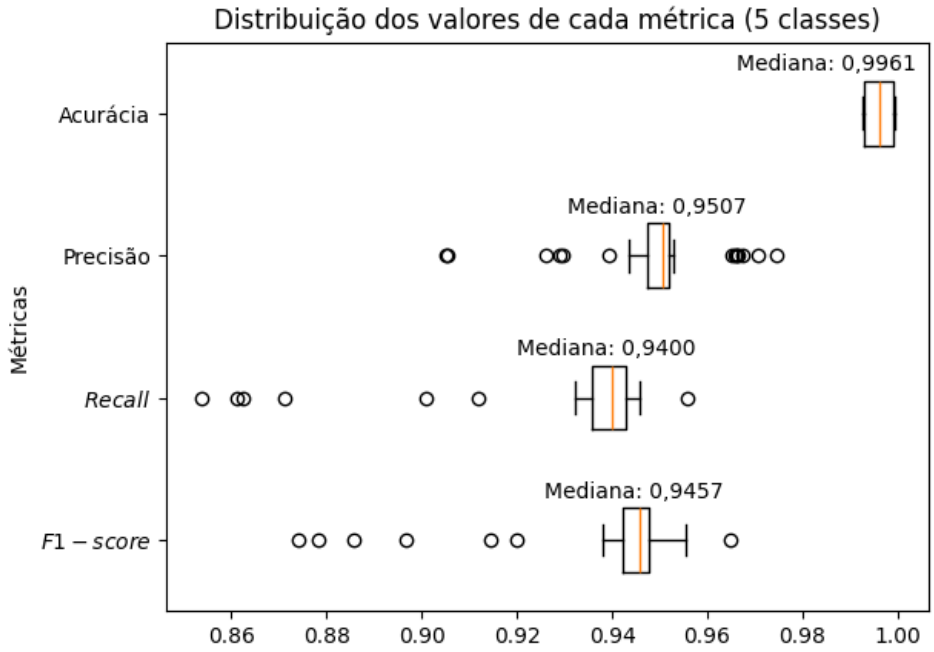


Figure 8: Distribuição dos valores de cada métrica do *LinkClassifier* (5 classes).

4.3.4 O *LinkClassifier* em uma tarefa de roteamento

Esta seção examina o problema do roteamento de pacotes como uma função de rede que visa evitar a sobrecarga dos enlaces, distribuindo a carga uniformemente entre eles.

A distribuição de carga na rede ocorre dividindo o tráfego entre dois ou mais enlaces, formando caminhos diferentes para o mesmo par de pacotes de origem e destino. De acordo com (Kodialam; Lakshman, 2022), o problema de determinar as proporções ótimas para alocar o tráfego entre os enlaces da rede, a fim de minimizar o congestionamento de pacotes, é NP-difícil (*NP-hard*). Essa complexidade é reduzida ao criar e treinar um modelo de ML para escolher quando é necessário dividir o tráfego entre um número finito de rotas predefinidas.

Portanto, uma função de roteamento pode otimizar sua tarefa utilizando modelos *LinkClassifier* para decidir quando dividir o tráfego. Para isso, assume-se que os próprios nós de origem contribuem para a função de roteamento enviando informações sobre suas demandas futuras planejadas. Em seguida, os modelos *LinkClassifier* são executados para classificar o uso dos enlaces com base nessas demandas futuras. Com essas classificações de enlaces, ações predefinidas de divisão de tráfego podem ser empregadas.

Neste experimento, 90% das amostras do conjunto de dados *Abilene* são utilizadas para treinar os modelos *LinkClassifier*, um para cada enlace da rede. Os outros 10% do conjunto de dados são as amostras de teste para os modelos *LinkClassifier*. Os valores de **Y** são gerados a partir de um processo de rotulagem de **X** utilizando 5 classes, ou seja, rótulos 1, 2, 3, 4 ou 5. O rótulo 1 representa um nível baixo de utilização do enlace de rede; o rótulo 2, um nível médio-baixo; o rótulo 3, um nível médio; o rótulo 4, um nível médio-alto; e o rótulo 5, um nível alto.

Após a fase de treinamento, os modelos *LinkClassifier* utilizam as amostras de teste $\mathbf{X}^{(t)}$ para gerar os rótulos $\hat{\mathbf{y}}_k^{(t)}$ para cada enlace k na rede. A Figura 9 mostra os números de identificação dos nós e enlaces da topologia da rede *Abilene* e as classificações desses enlaces, utilizando os modelos *LinkClassifier*, considerando algumas amostras de teste.

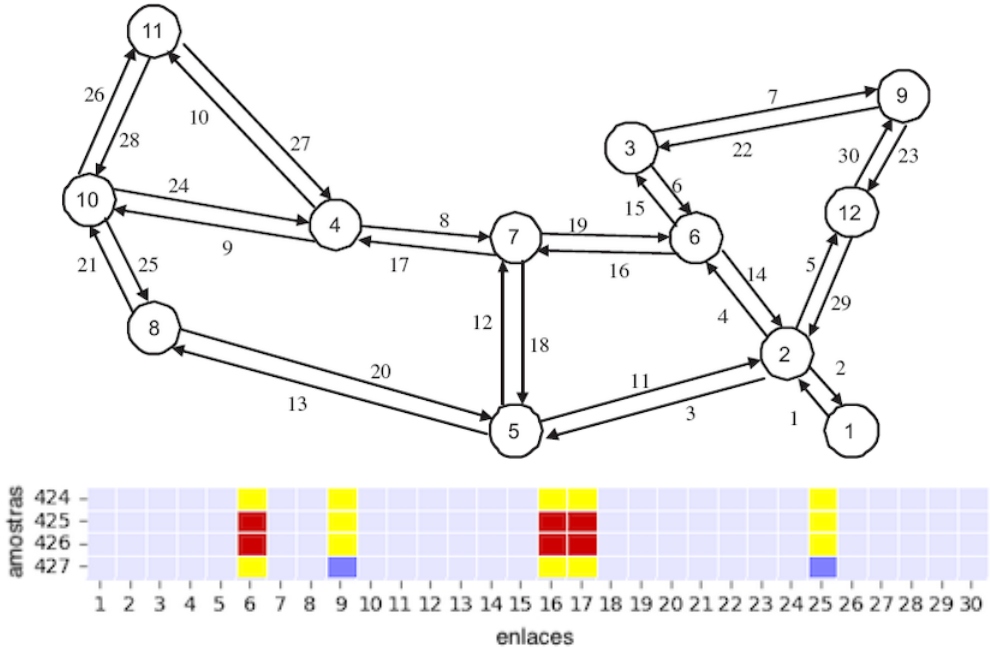


Figure 9: Resultado das classificações dos enlaces de rede, baseadas nas demandas dos nós. Legenda de rótulos: 1 - azul-claro, 2 - azul-médio; 3 - azul-escuro; 4 - amarelo; 5 - vermelho.

De acordo com a Figura 9, os enlaces 6, 16 e 17 recebem o rótulo 5 de seus respectivos modelos *LinkClassifier*, nas amostras 425 e 426. Utilizando essa informação, o sistema de roteamento realiza um desvio de rota dos pacotes de acordo com o conjunto de caminhos alternativos na rede. O comportamento padrão da rede é utilizar apenas o caminho mais curto (shortest path) entre a origem e o destino dos fluxos de dados. Se um modelo *LinkClassifier* classifica um enlace com o rótulo 5, todos os fluxos de dados que utilizam este enlace devem ser identificados. Esses fluxos identificados devem desviar parte de seus pacotes para caminhos alternativos que não contenham o enlace classificado com o rótulo 5.

Após o desvio parcial dos pacotes utilizando esses caminhos alternativos, os modelos *LinkClassifier* são utilizados novamente com as demandas de tráfego $\mathbf{X}^{(t)}$ para obter novos rótulos de classificação $\mathbf{y}^{(t)'}$. Neste caso, deve-se utilizar um *LinkClassifier* treinado com as novas rotas de pacotes. A Figura 10 mostra essa nova classificação dos enlaces, após o desvio parcial das rotas de pacotes em algumas amostras de teste.

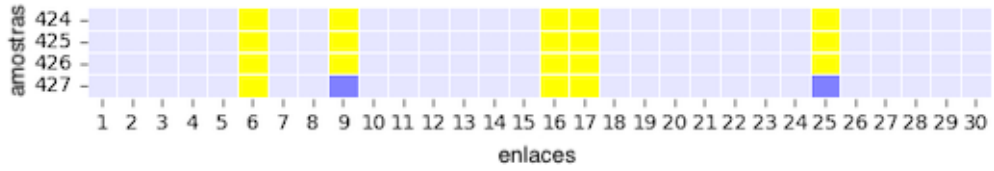


Figure 10: Classificação dos enlaces após as ações de desvio de rota de pacotes. Legenda de rótulos: 1 - azul-claro, 2 - azul-médio; 3 - azul-escuro; 4 - amarelo; 5 - vermelho.

Ao analisar as Figuras 9 e 10, pode-se observar que as ações tomadas para desviar as rotas dos pacotes evitaram a ocorrência de problemas de rede relacionados à alta utilização de enlaces em 23 amostras de teste. As classificações de enlaces realizadas pelo modelo *LinkClassifier* foram fundamentais para a identificação e adoção de medidas preventivas contra esses problemas potenciais.

4.4 Comentários finais

O trabalho deste capítulo propôs o modelo *LinkClassifier* que realiza a classificação de um enlace de rede de acordo com seu nível de utilização, a partir de uma matriz de tráfego.

O desempenho do modelo proposto é obtido por meio de experimentos utilizando dois conjuntos de dados. Nos primeiros experimentos, são realizadas comparações dos resultados do método proposto variando alguns de seus parâmetros.

Em outros experimentos, encontra-se uma comparação entre os resultados do *LinkClassifier* e os de outros modelos de classificação clássicos da literatura. Os resultados de desempenho do modelo *LinkClassifier* superaram os resultados dos outros modelos de comparação.

Por fim, o modelo *LinkClassifier* também foi testado utilizando um serviço de roteamento de pacotes de rede. Este teste demonstra como o modelo proposto pode auxiliar este serviço de roteamento. Após o modelo *LinkClassifier* classificar os enlaces da rede, o plano de gerência da rede usa os rótulos de classificação para modificar o caminho dos pacotes quando necessário.

5. Uso de Aprendizado Federado para classificação de enlaces de rede baseada em matrizes de tráfego

Este capítulo apresenta um ambiente de rede baseado em SDN no qual a solução de monitoramento proposta no Capítulo 3 pode ser aplicada. Essa SDN possui uma camada MEC que é usada para armazenar os dados de demanda de tráfego e treinar o modelo de classificação através do paradigma FL.

De forma semelhante ao Capítulo 4, este Capítulo 5 investiga o problema de realizar a classificação de enlaces de rede baseada em ML, ao invés de usar sensores dedicados ou calcular a utilização dos enlaces usando informações de topologia e caminho dos pacotes. Porém, diferentemente do Capítulo 4, este Capítulo 5 inclui características específicas de como esta classificação de enlaces pode ser feita num sistema descentralizado, definindo detalhes sobre o modelo de classificação de ML apresentado, suas etapas de procedimento de treinamento e o processamento dos dados brutos para montar o conjunto de dados supervisionado, como em (Silva *et al.*, 2026).

Logo, este capítulo propõe o *NetLinkClassifier*, um modelo de ML projetado para classificar enlaces de rede com base em seus níveis aproximados de utilização usando matrizes de tráfego como entrada. Desenvolvido para ambientes SDN, o modelo é treinado em conjuntos de dados distribuídos em uma camada MEC. Seguindo a abordagem descrita por (Amoroso *et al.*, 2023), essas matrizes de tráfego distribuídas são usadas para treinar o modelo via FL. Isso garante o processamento descentralizado e a privacidade dos dados, pois as matrizes de tráfego não são transmitidas para VNFs na camada *Cloud* da SDN. Para aumentar ainda mais a privacidade durante a montagem do conjunto de dados e as iterações de FL, um sistema criptográfico é empregado.

O modelo *NetLinkClassifier* é validado por meio de experimentos usando três conjuntos de dados diferentes. Nesses experimentos, as métricas de desempenho de classificação obtidas para o Modelo Proposto são apresentadas e comparadas com as métricas de desempenho de outros experimentos, variando alguns parâmetros de treinamento. Após avaliar o modelo, outro experimento é conduzido para exemplificar como o modelo *NetLinkClassifier* pode ser usado para auxiliar a aplicação de um serviço de roteamento de rede.

Em resumo, as contribuições do trabalho deste capítulo são:

- A proposta de um modelo de classificação que recebe uma matriz de tráfego como entrada e infere um rótulo de classificação para um enlace de rede, representando seu nível de utilização;
- O Modelo Proposto representa uma alternativa a outros sistemas de monitoramento de enlace mais dispendiosos;
- Para garantir o processamento descentralizado e a privacidade dos dados, uma abordagem de FL e um sistema de criptografia são empregados;

- Este modelo de classificação foi avaliado por meio de experimentos usando três conjuntos de dados contendo demandas de tráfego;
- O desempenho de classificação do modelo proposto é avaliado comparando seus resultados de métricas com os de outros modelos de classificação na literatura e variando as características do modelo e do cenário; e
- Um experimento de simulação mostra como o modelo de classificação pode ser usado para auxiliar um serviço de roteamento de pacotes.

A organização das seções deste capítulo é a seguinte: a Seção 5.1 apresenta a formulação deste problema; a Seção 5.2 descreve o modelo e o método de solução do problema; a Seção 5.3 demonstra o desempenho desta solução através de resultados de experimentos; a Seção 5.4 apresenta um exemplo de como o modelo proposto pode ser usado numa tarefa de roteamento de pacotes; e a Seção 5.5 apresenta uma conclusão do conteúdo deste capítulo.

5.1 Formulação do problema

Em uma SDN, existem N nós e K enlaces. De acordo com (Alam *et al.*, 2020), a arquitetura de uma SDN é basicamente dividida em Plano de Dados, Plano de Controle e Plano de Gerenciamento.

O Plano de Dados contém os nós da rede que transmitem pacotes entre si, gerando as demandas de tráfego da rede. O Plano de Controle contém os controladores SDN que recebem metainformações dos nós sobre suas demandas, atuando em decisões de roteamento quando solicitado. O Plano de Gerenciamento contém aplicações de rede, tais como roteamento de pacotes de rede, treinamento dos modelos *NetLinkClassifier* e classificação de enlaces de rede.

Semelhante aos trabalhos (Qian *et al.*, 2022) e (Shahrbabaki *et al.*, 2022), esta SDN também contém uma camada MEC para lidar com os dados de medição obtidos pelo monitoramento dos nós da rede. Esta camada MEC está localizada entre os nós da rede no Plano de Dados e os controladores SDN no Plano de Controle. Os controladores SDN e as aplicações SDN são executados por VNFs localizadas em uma camada de Nuvem (*Cloud*), distante dos nós da rede. A Figura 11 mostra esta arquitetura SDN com suas camadas MEC e Nuvem.

Nesta rede, os pacotes gerados por uma demanda de tráfego $\mathbf{x}_{i,j}$ partem de um nó de origem i e viajam para um nó de destino j, através de um caminho que passa por um enlace k. Se os caminhos de pacotes de múltiplas demandas de tráfego passarem pelo mesmo enlace k e a soma dessas demandas resultar em um valor alto, o enlace k pode ficar congestionado. O congestionamento de pacotes em um enlace pode levar a problemas de rede, tais como aumento de atraso (*delay*), perda de pacotes, *jitter*, etc.

Para evitar problemas de rede devido ao congestionamento de pacotes, é necessário medir frequentemente o desempenho de cada enlace na rede. Um exemplo de medida de desempenho para um enlace de rede k é a taxa de utilização y_k.

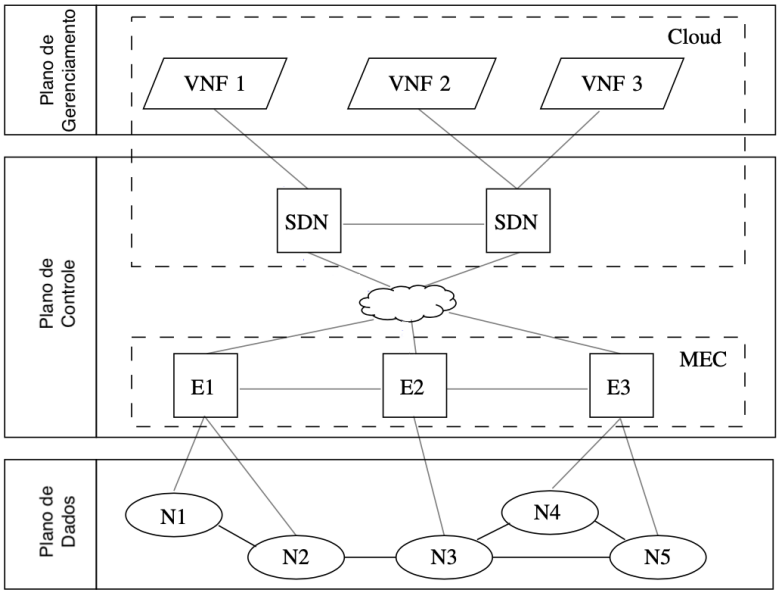


Figure 11: Arquitetura de uma rede SDN com uma camada MEC.

Em alguns serviços de roteamento, como o apresentado em (Uhlig *et al.*, 2006), é possível determinar por quais enlaces de rede os pacotes de uma demanda $\mathbf{x}_{i,j}^{(t)}$ passam, dado cada par de nós $\{i, j\}$ e em qualquer instante t. Conhecendo o caminho fixo e a intensidade de todas as demandas $\mathbf{x}_{i,j}^{(t)}$, todos os valores $y_k^{(t)}$ podem ser calculados em cada instante de tempo t.

Em sistemas de monitoramento de rede, como (Varyani *et al.*, 2020), são utilizadas classificações qualitativas dos enlaces de rede. Essas classificações são obtidas a partir dos valores de utilização y dos enlaces de rede.

O monitoramento desta rede SDN é realizado pelos nós da camada MEC, pois eles estão mais próximos dos nós da rede no Plano de Dados. No entanto, os nós da camada MEC não recebem diretamente as medições de desempenho nos enlaces da rede. Em vez disso, eles recebem de cada nó i na rede a quantidade de sua demanda de tráfego $\mathbf{x}_{i,j}$ para cada nó j na rede.

Se os nós da camada MEC não conhecem a topologia SDN nem o sistema de roteamento SDN, então eles não são capazes de calcular a taxa de utilização dos enlaces da rede.

Portanto, um modelo de classificação de ML deve ser treinado na camada MEC para inferir um rótulo de classificação $\mathbf{y}_k$ que representa o valor de utilização y_k do enlace de rede k, dadas as medições de demanda de tráfego ponto a ponto entre todos os nós da rede, ou seja, uma matriz de tráfego.

Finalmente, os valores $\mathbf{x}_{i,j}$ são organizados na matriz $\mathbf{X}$. A Figura 12 mostra um esquema das entradas e saídas deste modelo de classificação.

A entrada do modelo de classificação de ML são os valores da matriz de tráfego $\mathbf{X}$ e a saída é um rótulo de classificação $\mathbf{y}_k$, que representa o nível de utilização do enlace.

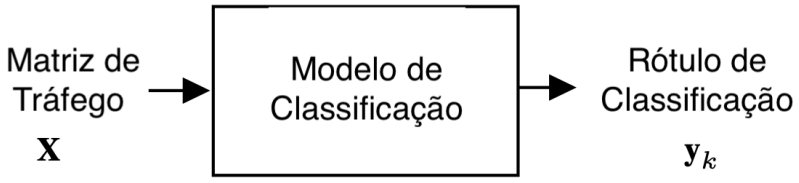


Figure 12: Modelo de classificação de um enlace de rede k.

Se for necessário, um sistema de criptografia pode ser aplicado para os valores de demanda de tráfego, medidos pelos nós SDN. Como resultado, os dados de medição tornam-se mais seguros contra ataques durante seu transporte e futuro armazenamento.

5.2 NetLinkClassifier

O modelo *NetLinkClassifier* propõe uma solução para o problema de classificação de enlaces, que foi apresentado na Seção 5.1. Este modelo infere o rótulo de classificação $\hat{\mathbf{y}}_k^{(t)}$ relacionado a um enlace k utilizando como entrada as medidas de demanda de tráfego ponto a ponto $\mathbf{X}^{(t)}$ em um determinado instante de tempo t.

Para inferir os rótulos $\hat{\mathbf{y}}_k^{(t)}$, o modelo *NetLinkClassifier* utiliza um Perceptron Multicamadas (MLP) contendo 4 camadas densas. A Figura 13 abaixo ilustra graficamente a arquitetura deste modelo.

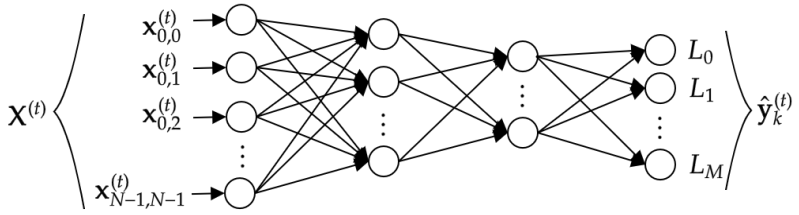


Figure 13: Representação gráfica do modelo *NetLinkClassifier*.

No exemplo da Figura 13, o Plano de Dados da SDN contém N nós e o tamanho do conjunto de classes é M. Este conjunto de classes é $\{L_1, L_2, \ldots, L_M\}$.

A primeira camada deste modelo é a de entrada e seu número de neurônios é igual ao número de nós ao quadrado (ou seja, $N \times N$); a segunda camada é a primeira camada oculta e seu número de neurônios é igual à metade do número de neurônios da camada anterior (ou seja, $(N \times N)/2$); a terceira camada é a segunda camada oculta e seu número de neurônios também é igual à metade do número de neurônios da camada anterior (ou seja, $(N \times N)/4$); e, finalmente, a última camada é a camada de saída e seu número de neurônios é igual ao número de rótulos possíveis. A função de ativação das primeiras 3 camadas é a função Unidade Linear Retificada (Rectified Linear Unit – ReLU) (He *et al.*, 2018) e a última camada utiliza uma função logística para sua ativação (Cai *et al.*, 2015; Asadi; Jiang, 2020). A saída do *NetLinkClassifier* é um vetor com as probabilidades para cada rótulo.

O modelo *NetLinkClassifier* é treinado de forma distribuída e, posteriormente, utilizado na camada MEC da SDN. Os dados de entrada do modelo são coletados do

Plano de Dados. A saída do modelo é enviada para os controladores SDN e VNFs na camada de Nuvem (*Cloud*) da SDN.

De acordo com os requisitos de segurança, um sistema de criptografia deve ser utilizado para as transmissões e armazenamento de dados. Uma sugestão de algoritmo de criptografia é o *Advanced Encryption Standard in Galois/Counter Mode* (AES-GCM), pois é tradicional e ainda amplamente utilizado em trabalhos recentes sobre redes (Gupta *et al.*, 2025; Kottur *et al.*, 2022).

5.2.1 Pré-processamento e montagem do *dataset*

Para treinar o modelo *NetLinkClassifier*, é necessário um conjunto de dados supervisionado do qual cada amostra t contém: uma matriz de tráfego cujos pontos representam valores de demanda de tráfego, ou seja, $\mathbf{X}^{(t)}$; e um rótulo de classificação para um enlace k da rede baseado na taxa de utilização deste enlace k, ou seja, $\mathbf{y}_k^{(t)}$.

A montagem deste conjunto de dados começa através das medições de cada nó no Plano de Dados da SDN. A cada instante de tempo t, um nó i na rede monta o seu vetor $\mathbf{x}_i^{(t)} = (\mathbf{x}_{i,1}^{(t)}, \ldots, \mathbf{x}_{i,N}^{(t)})$ com os valores de suas demandas de tráfego para todos os N nós desta rede. Em seguida, cada nó i da rede envia seus dados $\mathbf{x}_i^{(t)}$, por meio da camada MEC, para uma VNF definida no Plano de Gerenciamento da SDN.

Esta VNF conhece a topologia da rede e os caminhos do fluxo de dados de acordo com a função de roteamento da SDN. Assim, ela usa os vetores recebidos $\mathbf{x}_i^{(t)}$ para obter os valores das taxas de utilização $y_k^{(t)}$ do enlace k da rede.

De acordo com os parâmetros do modelo *NetLinkClassifier* relativos ao número de classes e às regras para defini-las, a VNF escolhe um rótulo de classificação $\mathbf{y}_k^{(t)}$ com base no valor $y_k^{(t)}$.

Então, a VNF então agrega os vetores $\mathbf{x}_i^{(t)}$, empilhando-os linha por linha, para montar a matriz de tráfego $\mathbf{X}^{(t)}$. Em seguida, ela envia os dados processados $\mathbf{X}^{(t)}$ e $\mathbf{y}_k^{(t)}$ de volta para a camada MEC da SDN.

A Figura 14 mostra, em azul, o caminho dos dados de medição $\mathbf{x}_i^{(t)}$ dos nós da rede para a VNF. E ela mostra, em verde, o caminho da matriz de tráfego $\mathbf{X}^{(t)}$ e do correspondente rótulo processado $\mathbf{y}_k^{(t)}$ que são enviados da VNF de volta para a camada MEC.

Esses dados $\mathbf{X}^{(t)}$ e $\mathbf{y}_k^{(t)}$ representam uma amostra do *dataset*. Um nó MEC é escolhido para armazenar cada amostra. Ao repetir esse procedimento em outros instantes de tempo, o conjunto de dados supervisionado e distribuído é formado. Esse conjunto de dados é usado para treinar o modelo *NetLinkClassifier*.

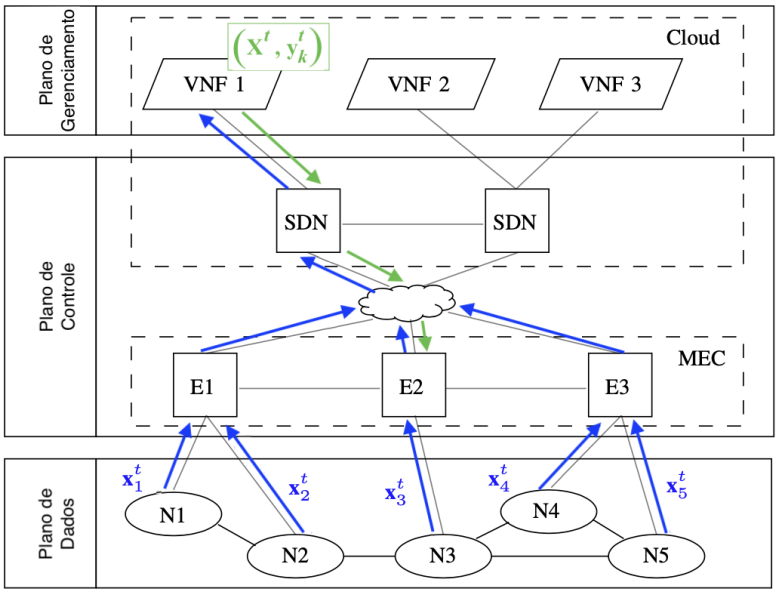


Figure 14: Dados dos nós para rotular VNF e, em seguida, para a camada MEC.

5.2.2 Treinamento do modelo

O modelo *NetLinkClassifier* é treinado na camada MEC da SDN usando o paradigma FL. Este tipo de aprendizado requer um conjunto de dados supervisionado e distribuído. Neste caso, o *dataset* é distribuído entre os nós da camada MEC da SDN.

Devido à sua simplicidade, o algoritmo definido para ser usado nesse treinamento FL é o Federated Averaging (FedAvg) (McMahan *et al.*, 2017). Este treinamento visa obter o conjunto de pesos w do modelo *NetLinkClassifier* que otimiza a inferência do rótulo de classificação $\hat{\mathbf{y}}_k^{(t)}$ dada uma matriz $\mathbf{X}^{(t)}$. A otimização de w é realizada através de iterações ι, nas quais os nós locais do FL calculam suas contribuição locais w_e^{ι} e um nó global do FL agrega todas as contribuições locais num vetor global da iteração w^{ι}.

Usando a arquitetura de SDN mostrada na Figura 11, os nós locais do FL são os nós MEC, uma vez que eles mantêm as amostras do conjunto de dados distribuído. A função do nó FL global é desempenhada por um controlador no Plano de Controle SDN.

Para o algoritmo FedAvg, a taxa de aprendizagem η e uma função de erro $l(\hat{\mathbf{y}}^{(t)}, \mathbf{y}^{(t)}, w^{(\iota)})$ são parâmetros importantes a serem definidos pelo projetista do modelo. A função de erro $l(\hat{\mathbf{y}}^{(t)}, \mathbf{y}^{(t)}, w^{(\iota)})$ representa a razão da diferença entre o rótulo de classificação $\mathbf{y}^{(t)}$ encontrado na amostra t do conjunto de dados supervisionado e o rótulo de classificação $\hat{\mathbf{y}}^{(t)}$ inferido pelo modelo *NetLinkClassifier* usando a matriz de tráfego $\mathbf{X}^{(t)}$ da amostra t e o conjunto de pesos $w^{(\iota)}$ da iteração atual ι do FL.

Um exemplo de função $l(\hat{\mathbf{y}}^{(t)}, \mathbf{y}^{(t)}, w^{(\iota)})$ a ser escolhida é a função de erro quadrático p, demonstrada pela Equação 2.2; logo, $l(\hat{\mathbf{y}}^{(t)}, \mathbf{y}^{(t)}, w^{(\iota)}) = (\hat{\mathbf{y}}^{(t)} - \mathbf{y}^{(t)})^2$ tal que $\hat{\mathbf{y}}^{(t)}$ é obtido através de $\mathbf{X}^{(t)}$ e $w^{(\iota)}$.

O algoritmo FedAvg começa com um conjunto aleatório de pesos globais $w^{(0)}$ para executar a primeira iteração. Durante o treinamento FL, cada nó local e recebe o conjunto

de pesos globais $w^{(\iota-1)}$ da iteração anterior e atualiza seu conjunto de pesos locais w_e^{ι} para a iteração atual ι, usando pelo menos uma de suas amostras. Esta atualização do conjunto de pesos locais w_e^{ι} é realizada utilizando a Equação 5.1, abaixo:

$$w_e^{\iota} = w^{\iota-1} - \nabla l(\hat{\mathbf{y}}^{(t)}, \mathbf{y}^{(t)}, w^{(\iota)}) \tag{5.1}$$

O nó global do FL recebe as contribuições locais dos E nós locais que participam do aprendizado. Esse nó global do FL, então, atualiza o modelo global usando a Equação 5.2, abaixo:

$$w^{\iota} = \sum_{e=1}^{E} \left(\frac{1}{E}\right) w_e^{\iota} \tag{5.2}$$

A Figura 15, abaixo, demonstra graficamente como a comunicação ocorre ao longo das iterações do treinamento FL. Nesta Figura, as setas azuis representam o envio do resultado da agregação do modelo global da iteração anterior. E as setas verdes representam o envio dos modelos locais dos nós da camada MEC a serem agregados pelo nó global do FL, na iteração atual.

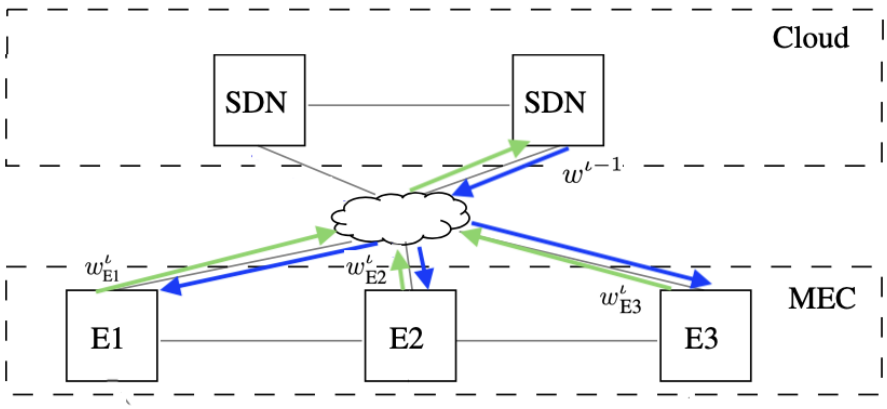


Figure 15: Iterações do FL.

O treinamento FL termina quando os valores dos vetores de peso globais de duas ou I iterações adjacentes $w^{\iota},\ldots,w^{\iota-I-1}$ convergem. Isso significa que as diferenças entre os valores dos elementos correspondentes de $w^{\iota},\ldots,w^{\iota-I-1}$ atingem um valor de limite inferior definido para a convergência.

5.2.3 Utilização do modelo treinado

Após o treinamento do modelo *NetLinkClassifier*, ele pode ser utilizado para inferir um rótulo $\hat{\mathbf{y}}_k^{(t)}$ que representa a taxa de utilização de um enlace k de rede a partir do Plano de Dados SDN. Este modelo está localizado em todos os nós da camada MEC da SDN. A entrada do modelo *NetLinkClassifier* é a matriz de tráfego $\mathbf{X}^{(t)}$.

Pelo menos um nó da camada MEC deve ter a matriz $\mathbf{X}^{(t)}$ do instante atual t, antes de usá-la no modelo *NetLinkClassifier*. Para montar a matriz $\mathbf{X}^{(t)}$, um nó selecionado

da camada MEC recebe de cada nó de origem i o vetor $\mathbf{x}_i$, contendo os valores das suas demandas de tráfego para cada nó de destino.

A Figura 16 mostra como é realizada a comunicação de dados para o uso de um modelo *NetLinkClassifier* treinado.

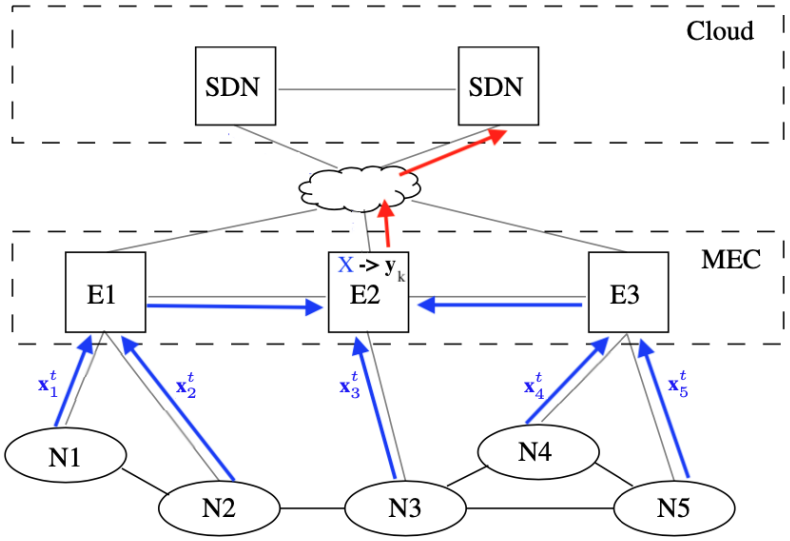


Figure 16: Utilização do modelo *NetLinkClassifier*.

Na Figura 16 acima, as setas azuis representam a comunicação necessária para montar a matriz $\mathbf{X}^{(t)}$. Posteriormente, a classificação $\hat{\mathbf{y}}_k$ do enlace k da rede é inferida na camada MEC.

Por fim, o sistema de monitoramento da SDN define que uma regra para informar ao controlador da SDN o resultado das classificações dos enlaces da rede. Por exemplo, ele pode definir que informará ao controlador SDN sobre a classificação dos seus enlaces, somente se o rótulo de classificação inferido representar uma alta taxa de utilização. A comunicação de dados da classificação de *NetLinkClassifier* entre um nó da camada MEC e um controlador SDN é representada pelas setas vermelhas na Figura 16.

5.3 Resultados e desempenho da classificação

Para avaliar o modelo *NetLinkClassifier* em diferentes cenários, são realizadas experiências com diferentes conjuntos de dados supervisionados, cujos processos de montagem são apresentados na Seção 5.3.1.

As inferências de rótulos de classificação do modelo *NetLinkClassifier* são avaliadas usando algumas métricas de desempenho definidas na seção 5.3.2.

A Seção 5.3.3 apresenta uma comparação dos valores de desempenho do modelo proposto. Enquanto isso, as seções 5.3.4 e 5.3.5 comparam os valores de desempenho do Modelo Proposto variando os parâmetros do modelo, os tipos do modelo, as características do conjunto de dados e as configurações de SDN.

5.3.1 Datasets

Esta seção apresenta como os conjuntos de dados supervisionados foram montados para os experimentos, usando três conjuntos de dados não supervisionados tradicionais com matrizes de tráfego. Um deles é o conjunto de dados Abilene, disponível em (Zhang, 2008; Idzikowski; Raack, 2023). Ele se trata de um conjunto de dados que contém 48.384 matrizes de tráfego de uma rede com 12 nós e 15 enlaces bidirecionais, ou seja, 30 links unidirecionais (Zhang *et al.*, 2003). Dois desses enlaces unidirecionais (ou seja, um enlace bidirecional) têm capacidade de 2.480.000 Kbps (kilobits por segundo) e os outros enlaces têm capacidade de 9.920.000 Kbps.

Os nós da rede Abilene estão localizados em diferentes cidades dos Estados Unidos, com exceção de dois nós que estão localizados na mesma cidade. Para montar este conjunto de dados Abilene, foram feitas medições da quantidade de tráfego na rede, com um tempo de amostragem de 5 minutos. As medições foram feitas de 3 de março de 2004 a 10 de setembro de 2004.

Outro conjunto de dados não supervisionado, descrito nesta seção, é o *dataset* Géant, disponível em (Idzikowski; Raack, 2023). Este conjunto de dados é composto por 11.460 matrizes de tráfego. A rede Géant contém 22 nós e 36 enlaces bidirecionais, ou seja, 72 enlaces unidirecionais (Orlowski *et al.*, 2010). Todas os enlaces têm uma capacidade de 40000000 Kbps (Idzikowski; Raack, 2023).

Nesta rede, a maioria dos nós está localizada numa cidade europeia e um nó está localizado em Nova Iorque, EUA. Para montar este conjunto de dados Géant, foram realizadas medições repetidas do volume de tráfego na rede, com um intervalo de amostragem de 15 minutos, ao longo de quatro meses.

Por fim, o último conjunto de dados não supervisionado desta lista é o *dataset* da rede Brain, disponível em (Idzikowski; Raack, 2023). Ele é composto por 9723 matrizes de tráfego de uma rede com 161 nós e 332 enlaces bidirecionais, ou seja, 664 enlaces unidirecionais (Orlowski *et al.*, 2010). Nesta rede Brain, todos os enlaces têm a capacidade de 1000000 Kbps (Idzikowski; Raack, 2023).

Todos os nós desta rede estão localizados em cidades da Alemanha. Para montar este conjunto de dados, foram feitas medições repetidas da quantidade de tráfego na rede, com tempo de amostragem de 1 minuto, ao longo de 7 dias.

Os três conjuntos de dados descritos acima são originalmente não supervisionados, pois eles contêm apenas a matriz de tráfego $\mathbf{X}^{(t)}$ para a entrada do modelo *NetLinkClassifier*, mas não contêm os valores dos rótulos de classificação $\mathbf{y}^{(t)}$ que representam a saída do modelo. Portanto, esses três conjuntos de dados devem passar por alguns processos de rotulação antes de serem usados para poder treinar e testar o modelo *NetLinkClassifier*.

O processo de rotulagem para obter conjuntos de dados supervisionados começa com o cálculo dos valores de utilização $y_k^{(t)}$ de um enlace k da rede, de acordo com cada matriz de tráfego $\mathbf{X}^{(t)}$. Esse cálculo envolve o conhecimento da topologia da rede e do sistema de roteamento de pacotes.

O sistema de roteamento de pacotes padrão dos experimentos nesta seção usa o algoritmo *Shortest-Path* (SP). Neste algoritmo, os pacotes de cada demanda de tráfego passam apenas pelo caminho mais curto entre seus nós de origem e destino. Esses caminhos são calculados usando o algoritmo *Dijkstra* e os valores de custo encontrados em (Idzikowski; Raack, 2023) para os enlaces da rede. Então, sabendo os caminhos entre todos os nós da rede, uma taxa de utilização $y_k^{(t)}$ do enlace k é calculada para cada matriz de tráfego $\mathbf{X}^{(t)}$ do conjunto de dados não supervisionado.

O cálculo da taxa de utilização $y_k^{(t)}$ ocorre da seguinte forma: os valores de demanda de uma origem e um destino, representados por cada posição de uma matriz de tráfego $\mathbf{X}^{(t)}$, devem ser somados se o caminho de seus pacotes passar pelo mesmo enlace k; o resultado dessa soma deve ser dividido pela capacidade máxima do enlace k para obter o valor $y_k^{(t)}$.

O processo de rotulagem continua para a obtenção dos rótulos de classificação $\mathbf{y}_k^{(t)}$ necessários para treinar e testar o modelo *NetLinkClassifier*. Cada rótulo $\mathbf{y}^{(t)}$ corresponde a um valor $y_k^{(t)}$ da taxa de utilização do enlace k da rede. A regra de rotulagem escolhida para transformar as taxas de utilização $y_k^{(t)}$ nos rótulos $\mathbf{y}_k^{(t)}$ se baseia numa divisão uniforme do intervalo entre os valores $y_k^{(t)} = 0$ e $y_k^{(t)} = 1$. Essa divisão ocorre em partes iguais de acordo com a quantidade de rótulos definida para o modelo. E se $y_k^{(t)} > 1$, o valor $\mathbf{y}_k^{(t)}$ é igual ao valor de rótulo mais significativo.

Por isso, para esta etapa de rotulagem, é necessário definir quantas classes são consideradas no modelo de classificação *NetLinkClassifier*. Este número de classes é usado para definir os valores limite de taxa de utilização de enlace que realizam as transições de rótulo de classificação.

Por exemplo, dado um conjunto de classes $\{0,1,2\}$, os valores da taxa de utilização do enlace $y = 0,333...$ e $y = 0,666...$ são usados como limites entre os rótulos de classificação $\mathbf{y}$. Portanto, em uma amostra t do conjunto de dados e em um enlace k da rede:

- $\mathbf{y}_k^{(t)} = 0$, if $y_k^{(t)} < 0,333...$;
- $\mathbf{y}_k^{(t)} = 1$, if $0,333... \leq y_k^{(t)} < 0,666...$; e
- $\mathbf{y}_k^{(t)} = 2$, if $y_k^{(t)} \geq 0,666....$

Ao realizar um processamento análogo para todas as matrizes de tráfego do *dataset* original, obtém-se um conjunto de dados supervisionado cujas amostras são os valores $\mathbf{X}^{(t)}$ e $\mathbf{y}_k^{(t)}$.

A Figura 17 demonstra o exemplo do processo de rotulagem apresentado nesta seção, considerando um enlace k da rede. Os valores $y_k^{(t)}$ foram calculados usando as amostras $X^{(t)}$ do conjunto de dados Abilene. Nesta Figura 17, o eixo horizontal do gráfico identifica

as amostras t. O eixo vertical à esquerda mostra a graduação dos valores da taxa de utilização $y_k^{(t)}$ e, à direita, os valores dos rótulos $\mathbf{y}_k^{(t)}$. Os níveis y que delimitam os rótulos $\mathbf{y}$ são representados pelas linhas vermelhas contínuas, na horizontal.

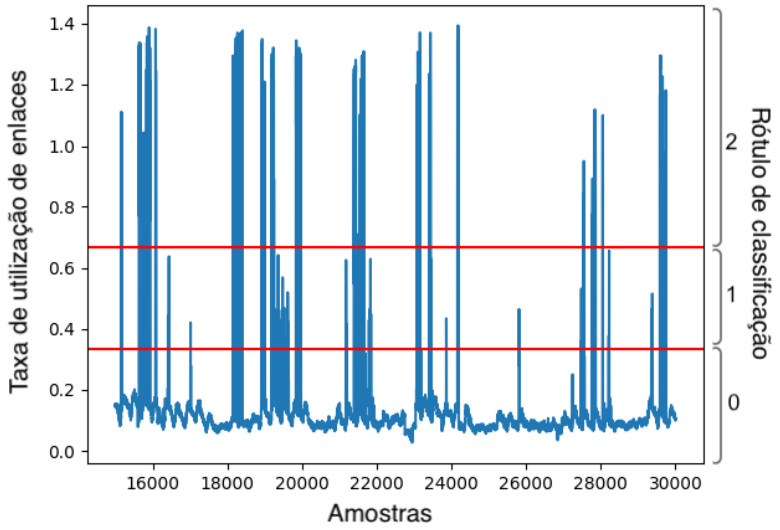


Figure 17: Exemplo da rotulagem do conjunto de dados Abilene com 3 classes.

5.3.2 Avaliação de desempenho

Para avaliar o modelo proposto, este trabalho utiliza os dados de teste como entrada do modelo para obter os rótulos de classificação inferidos $\hat{\mathbf{y}}^{(t)}$, que são comparados com os rótulos esperados correspondentes $\mathbf{y}^{(t)}$.

De acordo com (Fawcett, 2006), as métricas de avaliação, chamadas de acurácia, precisão, *recall* e *F1-score*, podem ser calculadas a partir da comparação entre a saída do modelo $\hat{\mathbf{y}}^{(t)}$ e seu valor correspondente de referência $\mathbf{y}^{(t)}$ no conjunto de dados supervisionado, montado de acordo com a Seção 5.3.1.

Para calcular a sobrecarga e os custos de comunicação FL, as métricas de avaliação são o atraso médio de processamento necessário para os nós locais do FL e a quantidade total de dados transferidos durante cada iteração de treinamento. O atraso médio de processamento é medido em segundos (s) e a quantidade total de dados transferidos é medida em bytes (B) ou múltiplos de bytes.

Portanto, foram realizados experimentos para treinar e testar o *NetLinkClassifier*. Para maximizar o uso dos dados disponíveis na medição do desempenho, o conceito de validação cruzada *K-Fold* foi usado para dividir os dados de treinamento e teste de cada um dos conjuntos de dados. A validação cruzada *5-Fold* foi usada em todos os conjuntos de dados.

Quando não for mencionado, os experimentos consideram um número fixo de classes, que neste caso é 2; e também um conjunto de dados fixo e um enlace de rede para a saída do modelo, que são, respectivamente, o conjunto de dados Abilene e o enlace exemplificado na Figura 17.

Nestes experimentos, muitos cenários são simulados para comparar seus resultados de desempenho. Primeiramente, o modelo *NetLinkClassifier* é comparado a outros modelos tradicionais de classificação. Em seguida, o desempenho do modelo *NetLinkClassifier* é avaliado variando o conjunto de dados, os parâmetros de treinamento e do modelo, a configuração SDN, o conjunto de dados e o tipo de rede do modelo.

5.3.3 Resultados comparados com modelos de classificação tradicionais

Nesta seção, os resultados de desempenho obtidos pelo *NetLinkClassifier* são comparados com os resultados de outros modelos de classificação tradicionais presentes na literatura. Esses modelos, para comparação com o *NetLinkClassifier*, são Árvore de Decisão (DT), MLP, Regressão Linear (LR), Regressão Logística (LGR) e Máquina de Vetor de Suporte (SVM). Exceto o modelo *NetLinkClassifier*, que usa o paradigma FL, todos os outros modelos de classificação são treinados usando um método centralizado.

Na Tabela 5, o modelo *NetLinkClassifier* possui os parâmetros de treinamento FL: taxa de aprendizado $\eta = 0,01$, tamanho de lote do conjunto de dados local (em inglês, *batch size*) bs igual a 32 e número de épocas ep igual a 1.

O modelo DT usa o índice de Gini como critério de decisão e as divisões ao longo dos ramos dessa árvore são sempre feitas com o melhor valor encontrado. O modelo MLP é igual ao definido para o modelo proposto, porém, usando um treinamento tradicional e centralizado. O modelo LR usa as equações de otimização do método chamado Método dos Mínimos Quadrados para ser treinado. O modelo de Regressão Logística é treinado usando o algoritmo *Broyden-Fletcher-Goldfarb-Shanno* (LBFGS). O modelo SVM usa um kernel polinomial de grau igual a 1 para treinar seus pesos e a dimensão do hiperplano de entrada do SVM é igual a N^2.

Após o treinamento de todos os modelos, as métricas de desempenho obtidas pelos resultados dos testes são comparadas entre eles. A Tabela 5 mostra os valores dessas comparações de métricas de desempenho.

Table 5: Desempenho do modelo *NetLinkClassifier* em comparação com modelos de classificação tradicionais.

Métrica	DT	MLP	LR	LGR	SVM	*NetLink-Classifier*
Acurácia	0,999	0,999	0,998	0,995	0,999	0,999
Precisão	0,995	1,000	0,959	0,931	0,995	0,999
Recall	0,997	0,971	0,999	0,964	0,993	0,987
F1-score	0,996	0,985	0,978	0,947	0,994	0,993

Na Tabela 5, os valores que representam as métricas do *NetLinkClassifier* são médias dos resultados de 20 execuções do experimento. A Figura 18 apresenta um diagrama *Boxplot* sobre as distribuições dos valores das métricas de *NetLinkClassifier*.

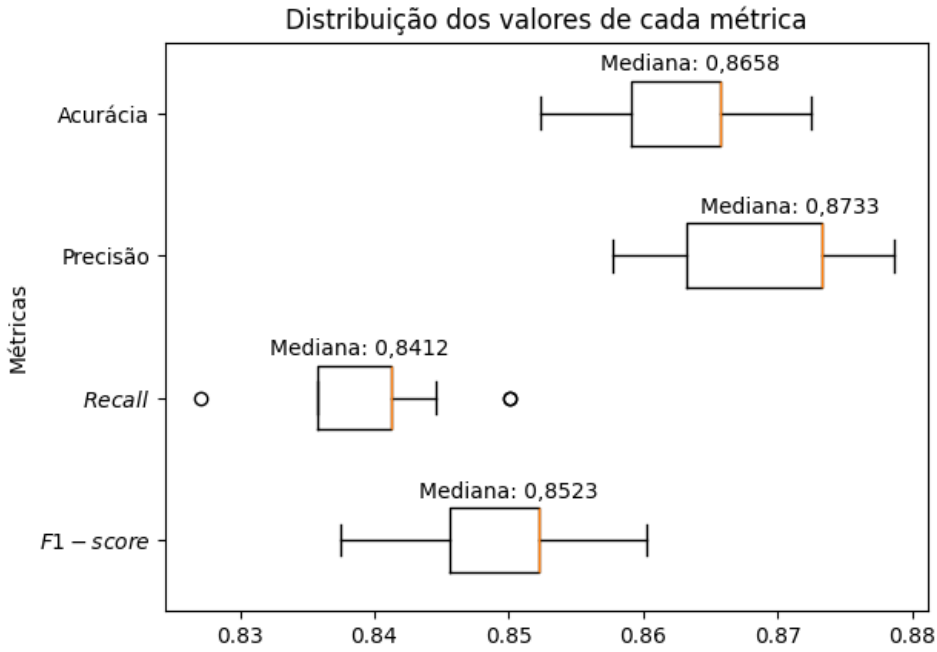


Figure 18: Distribuição dos valores de cada métrica do *NetLinkClassifier*.

De acordo com a Tabela 5, pode-se observar que, considerando esses experimentos, o modelo *NetLinkClassifier* apresenta uma ligeira diferença de valores de desempenho em comparação com outros modelos de classificação tradicionais. A vantagem do modelo proposto *NetLinkClassifier* é o seu método que utiliza o paradigma FL para descentralizar o processamento de treinamento e para manter os dados de tráfego na camada MEC da SDN.

A Tabela 6 mostra a quantidade de dados transferidos para o controlador SDN antes do treinamento e após 100 iterações, considerando o mesmo modelo MLP usando um aprendizado centralizado e usando FL (no caso do modelo proposto *NetLinkClassifier*). Para um treinamento de modelo centralizado, todo o conjunto de dados deve ser transferido para um controlador da SDN. Considerando 8 bytes por número nas amostras, é necessária uma quantidade de 58,12544 MB para transferir o conjunto de dados Abilene com 48384 amostras, cada uma com matrizes de tráfego 12x12 e um rótulo. Por outro lado, para um treinamento FL, é necessária uma quantidade de 29,4368 MB para transferir os dados de cada iteração. Isto porque o modelo MLP proposto na Figura 13 usando o conjunto de dados Abilene contém 18398 pesos e uma camada MEC com um MEC de 2 nós deve transferir 3679 pesos por iteração.

Table 6: A quantidade de dados transferidos para um controlador SDN antes e depois de 100 iterações.

	Quantidade de dados transferidos para um controlador SDN (em MB)	
Modelo (tipo de aprendizado)	Antes do treinamento	Depois de 100 iteraçõess
MPL (centralizado)	56.12544	0
Modelo proposto (FL)	0	29.4368

5.3.4 Resultados variando parâmetros do modelo e tipos de rede do modelo

Esta seção apresenta algumas comparações entre os valores de desempenho de experimentos usando o *NetLinkClassifier* com diferentes parâmetros e diferentes tipos de rede do modelo. Esta primeira comparação de desempenho é realizada entre experimentos com o *NetLinkClassifier* variando o parâmetro de taxa de aprendizado η.

A Figura 19 mostra que os valores mais altos de acurácia e *F1-score* são obtidos pelo modelo *NetLinkClassifier* com as taxas de aprendizado $\eta = 0,005$, $\eta = 0,01$ e $\eta = 0,05$. Esses valores mais altos de acurácia e *F1-score* são, respectivamente, 0,999 e 0,9851.

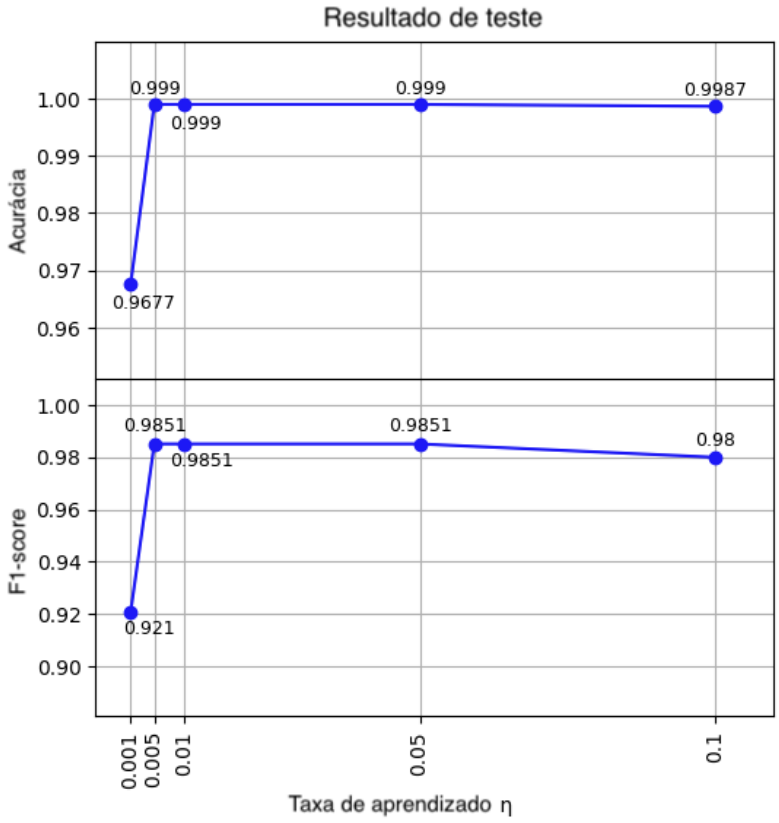


Figure 19: Valores de precisão e pontuação F1 variando a taxa de aprendizado.

No entanto, a Figura 20 demonstra que o valor de acurácia obtido pelo modelo *NetLinkClassifier* usando a taxa de aprendizado $\eta = 0,01$ é mais estável após 40 iterações.

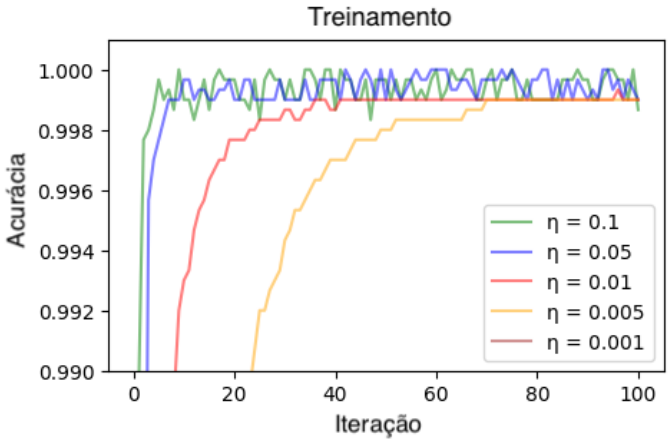


Figure 20: Valores de precisão por iteração de treinamento variando a taxa de aprendizado.

Outra comparação, apresentada nesta seção, é sobre a resposta do modelo *NetLinkClassifier* usando diferentes combinações dos parâmetros de treinamento ep e bs.

Definindo o valor da taxa de aprendizado $\eta = 0,01$, as combinações de ep e bs são feitas variando ep entre os valores 1, 50, 100 e 200 e variando bs entre 32 e 128.

A Figura 21 mostra que, quando $\eta = 0,01$, o valor de acurácia obtido pelo modelo *NetLinkClassifier* é muito alto em todas essas combinações de parâmetros ep e bs; ou seja, essa acurácia é de cerca de $0,999$.

No entanto, a Figura 22 mostra que o tempo médio aproximado de processamento (atraso de processamento) para as iterações locais atinge o maior valor (ou seja, 740,585 s) na combinação de parâmetros $ep = 200$ e $bs = 32$. Enquanto isso, o atraso médio aproximado de processamento para as iterações locais obtém os valores mais baixos, dado $ep = 1$. Ao definir $ep = 1$ e $bs = 32$, esse atraso médio de processamento é de 3,87 s e, ao definir $ep = 1$ e $bs = 128$, é de 2,66 s.

O hardware usado para executar iterações de treinamento locais e obter a média aproximada de seus atrasos de processamento é composto por um processador M4, uma Unidade Central de Processamento (CPU) de 10 núcleos e 24 Gb de Memória de Acesso Aleatório (RAM).

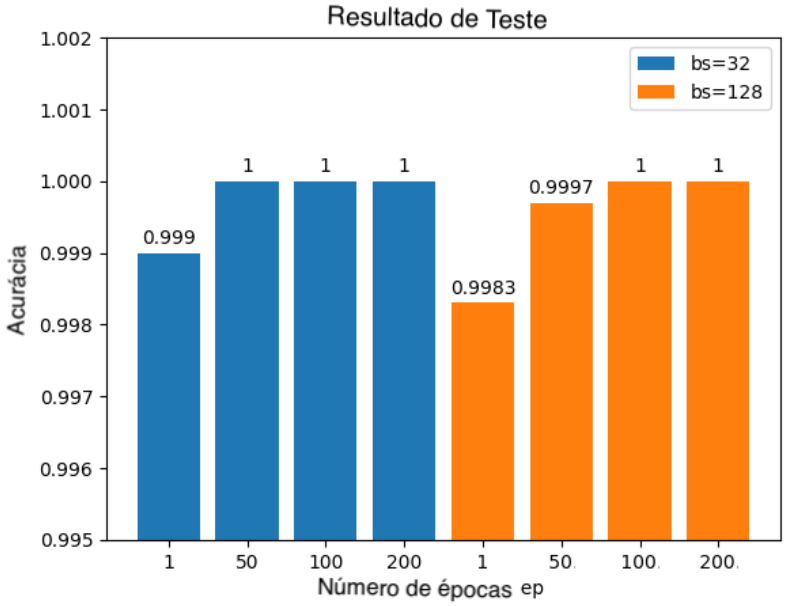


Figure 21: Valores de acurácia variando ep e bs.

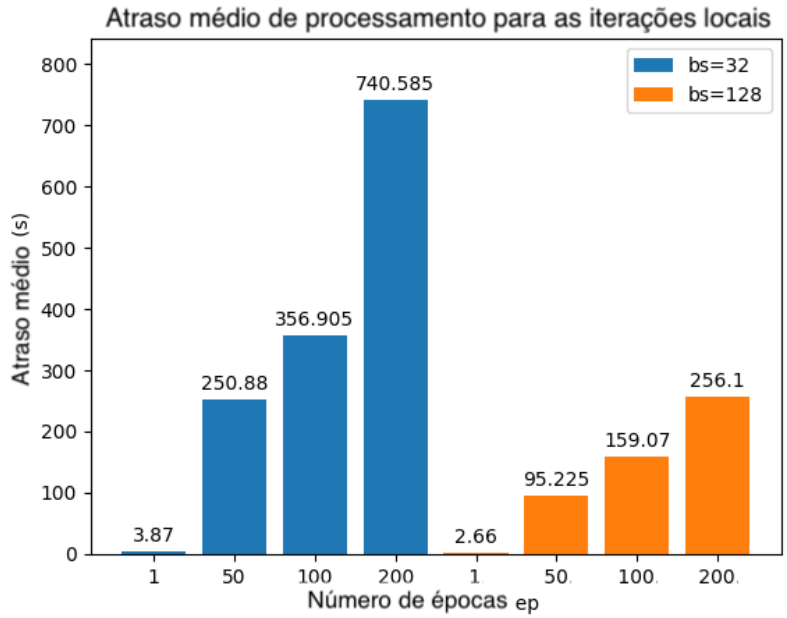


Figure 22: Atraso médio de processamento local por iteração variando ep e bs.

A última comparação nesta seção é feita variando o tipo de rede de modelo padrão do *NetLinkClassifier*. A rede de modelo padrão é uma MLP apresentada na Figura 13. Outra rede de modelo para esta comparação é uma Rede Neural Convolucional (CNN) com 2 camadas convolucionais cujo tamanho espacial de seus filtros é 2x2; 2 camadas de pooling e 3 camadas densas totalmente conectadas para a tarefa de classificação final. Outra rede modelo para esta comparação é uma rede de Memória de Curto Prazo Longo (LSTM) com 64 neurônios. Finalmente, a última rede modelo a ser comparada com o modelo padrão do *NetLinkClassifier* é uma rede de Unidade Recorrente Comportada (GRU) com 50 unidades.

A Figura 23 mostra que os valores de precisão obtidos pelo *NetLinkClassifier* são mais elevados quando modelados utilizando um MLP ou um CNN (ou seja 0,999) e mais baixos quando utilizando um LSTM ou um GRU (ou seja 0,966).

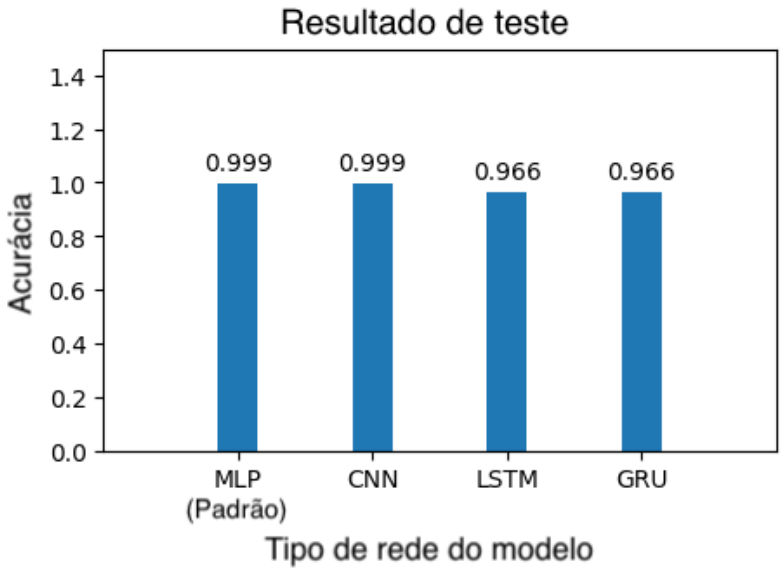


Figure 23: Acurácia variando o tipo de rede do modelo do *NetLinkClassifier*.

Embora os valores de acurácia representem uma avaliação de classificação muito satisfatória, as redes CNN podem não ser indicadas para essa classificação de link com base em uma matriz de tráfego da rede analisada (SDN). As CNNs usam camadas convolucionais para identificar a correlação espacial entre os pontos da matriz na entrada do modelo. Portanto, as CNNs são amplamente utilizadas em classificações de imagens. Neste caso, um ponto de uma matriz de tráfego representa um fluxo de tráfego da rede e dois pontos adjacentes desta matriz podem não ter correlação alguma.

No caso do LSTM e do GRU, eles identificam a correlação temporária entre duas ou mais amostras de entrada adjacentes. Essa correlação temporária das amostras da matriz de tráfego não é uma regra, ela depende do padrão de utilização dos clientes da rede e da taxa de amostragem que está presente em cada conjunto de dados.

A Tabela 7 mostra o número de pesos e a quantidade de dados transferidos por iteração, no caso de cada tipo de modelo de rede.

Table 7: Número de pesos e quantidade de dados transferidos por iteração em cada tipo de rede modelo.

Tipo de rede do modelo	Número de pesos em cada modelo	Número de pesos transferidos por iteração	Qauntidade de dados transferidos após 100 iterações (em MB)
MPL (padrão)	18398	36796	29.4368
CNN	4556	9112	7.2896
LSTM	29378	58756	47.0048
GRU	23352	46704	37.3632

5.3.5 Resultados variando as características do conjunto de dados e as configurações SDN

Esta seção apresenta comparações de desempenho entre experimentos usando o modelo *NetLinkClassifier* em cenários de diferentes características de conjunto de dados e diferentes configurações SDN.

A primeira comparação, presente nesta seção, é sobre o modelo *NetLinkClassifier* sendo treinado com diferentes números de classes. Portanto, os experimentos para esta comparação treinam e testam o modelo com os seguintes números de classes: 2, 3, 4 e 5.

A Figura 24 mostra que o valor de acurácia mais alto, que é $0,999$, é obtido pelo modelo *NetLinkClassifier* com 2 classes e o valor de acurácia mais baixo é $0,9932$, com 5 classes.

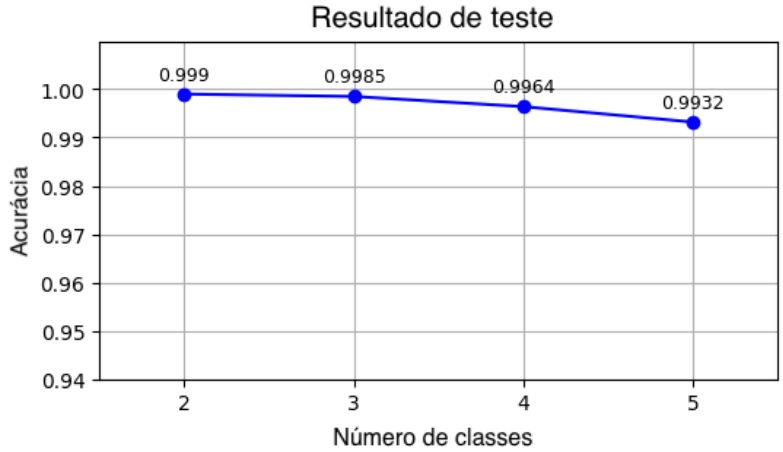


Figure 24: Valores de precisão variando o número de classes.

Outra comparação de desempenho desta seção é entre os experimentos com diferentes números de nós da camada MEC na arquitetura SDN. Esses nós da camada MEC são os nós locais que participam do FL que treina o modelo *NetLinkClassifier*.

A Figura 25 mostra que o valor de acurácia mais alto, que é $0,999$, é obtido pelo modelo *NetLinkClassifier* em uma SDN com uma camada MEC que contém 2 nós, e o

valor de acurácia mais baixo é $0,9973$, em uma SDN com uma MEC de 10 nós.

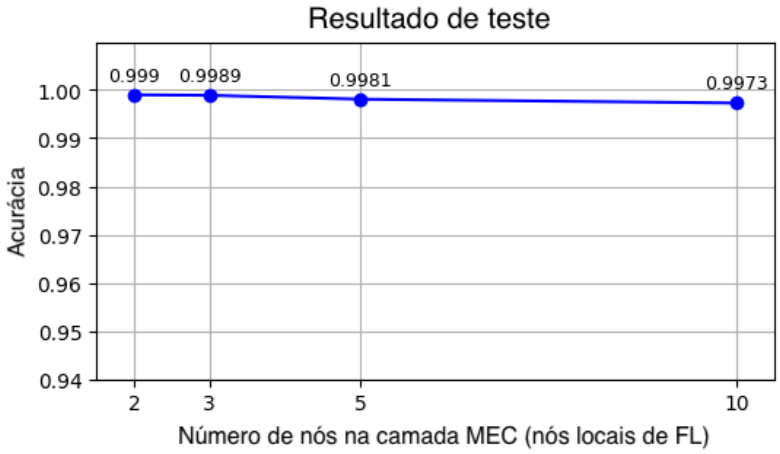


Figure 25: Valores de precisão variando o número de nós MEC.

Outra análise de desempenho nesta seção ocorre com a inclusão de um método de criptografia na configuração SDN. Neste caso, as transmissões das contribuições parciais do modelo durante as iterações de treinamento FL são criptografadas. O algoritmo de criptografia definido para esta análise é o AES-GCM.

O valor de acurácia obtido após o treinamento do modelo *NetLinkClassifier* com a criptografia AES-GCM é o mesmo que no caso do treinamento sem criptografia. Esse valor de acurácia é $0,999$. A Figura 26 mostra que os valores de *F1-score* em ambos os casos (com e sem a criptografia AES-GCM) também são muito próximos.

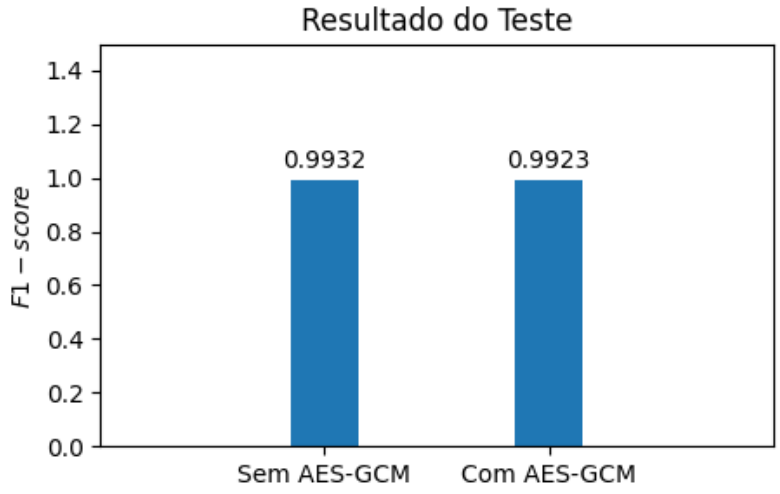


Figure 26: Valores de *F1-score* do *NetLinkClassifier* com ou sem criptografia AES-GCM.

A próxima análise de desempenho desta seção é feita alterando o algoritmo de roteamento da SDN. Ao invés de avaliar a classificação do modelo apenas no cenário de roteamento de Caminho Mais Curto (SP), esta análise verifica o desempenho do modelo usando conjuntos de dados cuja parte supervisionada de suas amostras (y_k) é obtida quando o SDN usa outros dois algoritmos de roteamento: o roteamento de Custo Igual

Multicaminho (ECMP) e um sistema de roteamento que usa os 3 Caminhos Mais Curtos (3-SP).

A Figura 27 mostra que os valores de acurácia obtidos pelo *NetLinkClassifier* são mais elevados nos sistemas de roteamento SP e 3-SP (ou seja, $0,999$) e apenas ligeiramente inferiores no sistema de roteamento ECMP (ou seja, $0,994$). Uma vantagem de usar o modelo *NetLinkClassifier* para classificar os links SDN é que o sistema de roteamento do SDN não é uma entrada direta do modelo. O resultado do sistema de roteamento do SDN é aprendido pelo modelo por meio das amostras do conjunto de dados.

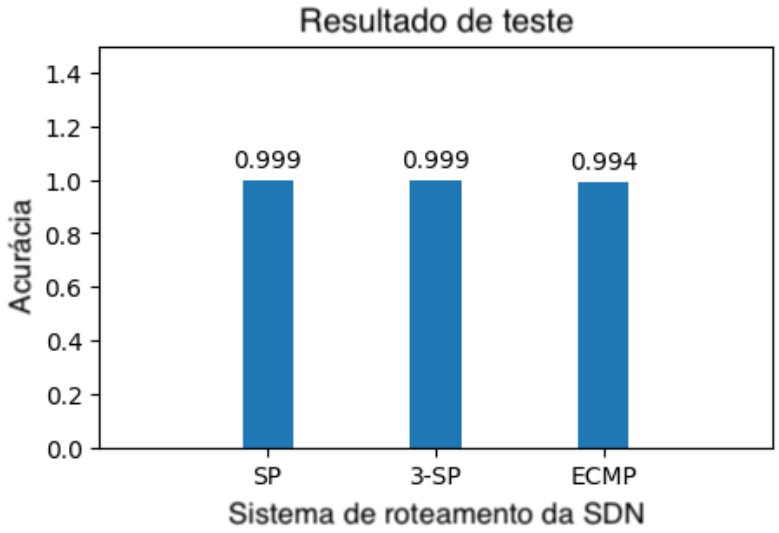


Figure 27: Valores de acurácia variando o sistema de roteamento da SDN.

A Figura 28 mostra que os valores de acurácia mais altos são obtidos pelo modelo *NetLinkClassifier* com os conjuntos de dados da rede Abilene (ou seja, $0,999$). E os valores de acurácia mais baixos são obtidos com os conjuntos de dados da rede Brain (ou seja, $0,8567$ e $0,8813$).

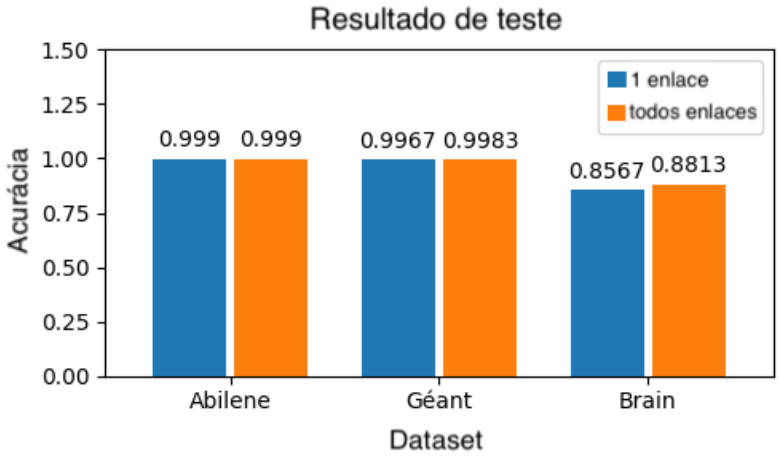


Figure 28: Valores de acurácia variando a arquitetura da rede SDN e o número de links para classificação.

As experiências que utilizam dados da rede Brain apresentam os valores de acurácia mais baixos porque, neste caso, o tamanho da entrada do modelo é grande e há menos

amostras do conjunto de dados para treinar o modelo. Os valores de acurácia são geralmente mais altos considerando todos os enlaces da rede, pois há muitos enlaces com tráfego baixo ou nenhum tráfego e, com isso, o modelo aprende facilmente como classificá-los.

5.4 Estudo de caso numa tarefa de roteamento

Esta seção investiga como o modelo *NetLinkClassifier* pode ser usado para auxiliar uma função de rede que realiza o roteamento de pacotes para evitar a sobrecarga dos enlaces de rede, distribuindo a carga entre eles.

A distribuição de carga na rede ocorre através da divisão do tráfego entre dois ou mais enlaces da rede, formando diferentes caminhos para os pacotes de mesma origem e mesmo destino. De acordo com (Kodialam; Lakshman, 2022), a escolha das proporções de divisão do tráfego entre os enlaces da rede a fim de minimizar o congestionamento de pacotes é um problema *NP-difícil*. Essa complexidade é reduzida pela criação e treinamento de um modelo de ML para escolher quando é necessário dividir o tráfego entre um número finito de rotas predefinidas.

Assim, a função de roteamento é otimizada usando instâncias do modelos *NetLinkClassifier* para decidir quando dividir o tráfego. Para fazer isso, presume-se que os próprios nós de origem contribuem para a função de roteamento, enviando informações sobre suas demandas de tráfego planejadas para o instante de tempo posterior. Em seguida, uma instância do modelo *NetLinkClassifier* é treinado para classificar a utilização de cada enlace da rede, com base nessas demandas de tráfego futuras dos nós. Com essas classificações de enlaces, ações predefinidas de divisão de tráfego podem ser empregadas.

Nesta experiência, são usados conjuntos de dados supervisionados formados a partir do *dataset* Abilene. Os *datasets* supervisionados usados em todas as instâncias do modelo *NetLinkClassifier* possuem amostras com as matrizes de tráfego $\mathbf{X}$, porém cada um destes *datasets* possui em suas amostras os rótulos $\mathbf{y}_k$ relativos a um enlace da rede.

Os valores $\mathbf{y}_k$ de cada *dataset* são gerados a partir de um processo de rotulagem usando três classes, ou seja, rótulos 0, 1 ou 2. O rótulo 0 representa um baixo nível de utilização do enlace de rede; o rótulo 1, um nível médio de utilização; e o rótulo 2, um nível alto.

Os 90% das amostras de cada conjunto de dados são utilizadas para treinar o respectivo modelo *NetLinkClassifier*. Os restantes 10% de cada conjunto de dados são as amostras de teste de cada o modelo *NetLinkClassifier*.

Após a fase de treinamento, os modelos *NetLinkClassifier* usam as amostras de teste $\mathbf{X}^{(t)}$ para gerar os rótulos $\hat{\mathbf{y}}_k^{(t)}$ para cada link k na rede.

A Figura 29 mostra os números de identificação dos nós e enlaces da topologia da rede Abilene e a Figura 30 mostra as classificações desses enlaces, usando os modelos *NetLinkClassifier*, considerando algumas amostras de teste.

De acordo com a Figura 30, os enlaces 6, 9, 16, 17 e 25 recebem o rótulo 2 de seus

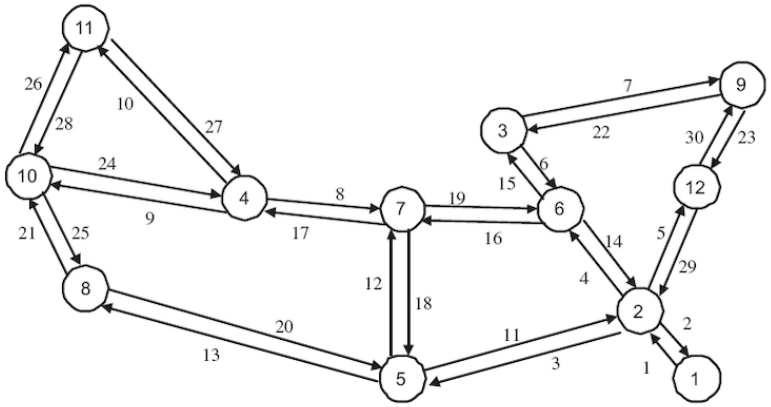


Figure 29: Identificações de nós e links na topologia da rede Abilene.

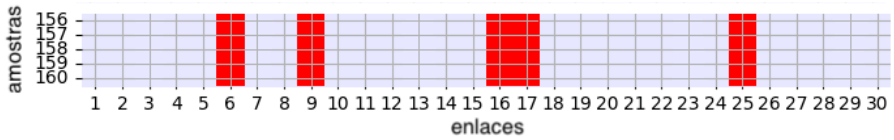


Figure 30: Classificação de links antes das ações de desvio de rota de pacotes.

respectivos modelos *NetLinkClassifier*, nas amostras de 156 a 160.

O desvio de rota dos pacotes, realizado pelo Plano de Gerenciamento da SDN, é definido de acordo com o conjunto de caminhos alternativos na rede. O comportamento padrão da rede é usar apenas o caminho mais curto entre a origem e o destino dos fluxos de dados.

No entanto, se um modelo *NetLinkClassifier* classifica um enlace com o rótulo 2 de acordo com as demandas de tráfego planejadas pelos nós de origem, todos os fluxos de dados que usam esse enlace devem ser identificados. Esses fluxos identificados devem desviar parte de seus pacotes para caminhos alternativos que não contenham o enlace classificado com o rótulo 2.

O Plano de Gerenciamento da SDN conhece os caminhos alternativos para todos os fluxos de dados. Quando necessário, o Plano de Gerenciamento escolhe caminhos alternativos que não contenham enlaces classificados com o rótulo 2, usando os respectivos modelos *NetLinkClassifier*.

Felizmente, existem caminhos alternativos para as rotas de pacotes padrão dos enlaces a classificação de rótulo 2 na Figura 30. Após o desvio parcial dos pacotes utilizando esses caminhos alternativos, os modelos *NetLinkClassifier* são usados novamente com as demandas de tráfego $\mathbf{X}^{(t)}$ para obter as novas classificações $\mathbf{y}^{(t)'}$. Nesse caso, deve-se usar modelos *NetLinkClassifier* treinados com as novas rotas de pacotes.

A Figura 31 apresenta essa nova classificação de enlaces de rede, após o desvio parcial das rotas de pacotes nas mesmas amostras de teste.

Para demonstrar a eficácia do desvio de rotas usando classificação de enlaces, a tabela 8 apresenta as taxas máximas de utilização do enlace a partir das amostras do conjunto de dados analisado, antes e depois da ação do controlador da SDN de desvio de pacotes.



Figure 31: Classificação de links após ações de desvio de rota de pacotes.

Table 8: Taxas máximas de utilização do enlace antes e depois da ação de desvio de pacotes.

		Antes do desvio de pacotes	Após o desvio de pacotes
Taxa máxima de utilização de enlace	Amostra 156	0,848	0,437
	Amostra 157	0,861	0,582
	Amostra 158	1.035	0,534
	Amostra 159	0,935	0,454
	Amostra 160	0,823	0,337

Ao analisar as Figuras 30 e 31 e a Tabela 8, observa-se que as ações tomadas para desviar rotas de pacotes impediram a ocorrência de problemas de rede relacionados à alta utilização de enlaces em 5 amostras de teste. As classificações de enlaces realizadas pelos modelos *NetLinkClassifier* foram fundamentais para identificar e tomar medidas preventivas contra esses problemas potenciais da rede.

5.5 Comentários finais

Este capítulo propõe o modelo *NetLinkClassifier* para classificar um enlace SDN de acordo com seu nível de utilização, com base em uma matriz de tráfego. O Modelo Proposto é baseado em ML e é treinado utilizando o paradigma de FL. Uma camada MEC entre o Plano de Dados e o Plano de Gerenciamento da SDN contém os nós participantes do FL. Após o treinamento de um modelo para cada enlace, esta camada MEC aplica esses modelos para informar ao Plano de Gerenciamento da SDN as classificações dos enlaces. Os rótulos de classificação representam o nível de utilização do enlace. Assim, este modelo proposto traz uma alternativa a outros sistemas de monitoramento de enlace mais dispendiosos.

O desempenho do modelo de ML proposto é obtido por meio de experimentos utilizando dados das redes Abilene, Géant e Brain. Esses experimentos são realizados variando os parâmetros do modelo, tipos de rede do modelo, características do conjunto de dados e configurações da SDN.

O modelo *NetLinkClassifier* também foi testado utilizando um serviço de roteamento de pacotes. Neste experimento, o Modelo Proposto classifica os enlaces de rede e o plano de gerenciamento da SDN toma providências para modificar o caminho dos pacotes quando necessário.

6. Aplicação da Proposta em Sistemas Inteligentes de Transporte

Este capítulo investiga como a ideia de solução proposta para o problema de redes apresentado no Capítulo 3 pode ser generalizada em um outro contexto, neste caso relacionado a ITS, voltado para a aplicação em Cidades Inteligentes.

De acordo com (Cai *et al.*, 2022)(Bibri; Krogstie, 2017), Cidade Inteligente é um conceito proposto há algumas décadas para otimizar a gestão de sistemas urbanos e melhorar a eficiência do transporte urbano, do uso do solo e do desenvolvimento econômico. Atualmente, o conceito de Cidade Inteligente já está implementado em diversas cidades ao redor do mundo e se torna cada vez mais popular (Marco; Mangano, 2021). O principal objetivo das Cidades Inteligentes é a otimização da utilização dos recursos disponíveis para o benefício de moradores, prestadores de serviços, governos e empresas (Munawar; Piantanakulchai, 2024; Humayun *et al.*, 2022; Athanasaki; Tsikaloudaki, 2022; Kolhe *et al.*, 2023).

Um dos desafios no planejamento do uso do solo urbano e do tráfego urbano reside na obtenção de informações ocultas a partir de dados de mobilidade e atividade humana. Isso ajuda a compreender como os espaços urbanos são utilizados diariamente. Esses dados de trajetória humana são geralmente obtidos por meio de registros de GPS de veículos, dados de posicionamento de telefones celulares e dados de *check-in* em mídias sociais (Yang; Gidofalvi, 2018; Pei *et al.*, 2014; Martí *et al.*, 2019; Hu *et al.*, 2021).

Os táxis são muito importantes para os sistemas de transporte público urbano (Munawar; Piantanakulchai, 2024). Os registros de GPS que contêm dados de trajetória de táxis geralmente cobrem grandes regiões urbanas, proporcionando uma precisão muito maior do que outras fontes de dados. Assim, os dados de GPS de táxis são amplamente utilizados na literatura para auxiliar no monitoramento de situações de tráfego em tempo real e na detecção de padrões de interação espacial (Hu *et al.*, 2021).

A complexidade de analisar e resolver problemas relacionados a um sistema de transporte urbano pode ser reduzida dividindo-se todo o sistema urbano em zonas de tráfego (Cai *et al.*, 2022; Li; Yan, 2019). Utilizando dados de GPS de táxis e esse conceito de zonas de tráfego, um sistema básico de monitoramento consistiria em uma verificação periódica da quantidade de táxis localizados em cada zona de tráfego da cidade.

Em analogia ao trabalho (Uhlig *et al.*, 2006), é possível determinar a quantidade de táxis em uma zona de tráfego, utilizando informações sobre a demanda de passageiros (as zonas de embarque e desembarque de cada corrida) e conhecendo todos os trajetos preferidos pelos taxistas, considerando essa demanda. Assim como em (Varyani *et al.*, 2020), um sistema de monitoramento pode usar rótulos para representar as quantidades em análise. Nesse caso, rótulos de classificação são definidos para representar a quantidade de táxis em uma zona do mapa da cidade durante um intervalo de tempo.

Se um sistema de monitoramento de táxis não dispõe de dados de GPS em tempo real

nem de informações sobre os trajetos preferidos pelos taxistas, então esse sistema deve utilizar um modelo de ML para obter rótulos de classificação para uma zona de tráfego da cidade, considerando a demanda de passageiros.

Como alternativa à abordagem tradicional e centralizada de treinamento de ML, a abordagem de FL permite que nós descentralizados (clientes) processem dados localmente, contribuindo para o treinamento de um modelo global. O FL é a abordagem de treinamento de ML indicada para conjuntos de dados distribuídos. Além disso, o uso de Aprendizado de Máquina (AM) impede a transmissão de dados brutos para servidores em nuvem.

Assim, no trabalho deste capítulo, propõe-se um modelo de ML para classificar uma zona de tráfego urbano com base na quantidade de táxis, considerando as informações de demanda de passageiros de táxi em um determinado instante. Este modelo de AM, o *CityZoneClassifier*, é treinado utilizando a abordagem de AM e um conjunto de dados distribuído.

O *CityZoneClassifier* é validado por meio de experimentos executados em diferentes cenários de configuração do modelo, ML e rede VEC (Computação de Borda Veicular ou *Vehicular Edge Computing*). Após a avaliação do modelo, os valores das métricas de desempenho do *CityZoneClassifier* são comparados com os de outros modelos de classificação tradicionais.

A organização das seções deste capítulo é a seguinte: a Seção 6.1 apresenta a formulação deste problema; a Seção 6.2 descreve o modelo e o método de solução do problema; a Seção 6.3 demonstra o desempenho desta solução através de resultados de experimentos; a Seção 6.4 apresenta um exemplo de como o modelo proposto pode ser usado numa tarefa para evitar o congestionamento de vias exclusivas de táxi; e a Seção 6.5 apresenta uma conclusão do conteúdo deste capítulo.

6.1 Formulação do problema de ITS

O sistema de monitoramento de tráfego de uma Cidade Inteligente divide seu mapa em múltiplas zonas para obter informações periódicas sobre a quantidade aproximada de táxis que circulam em cada zona. Ao invés de usar dados de localização GPS dos táxis em tempo real, este sistema de monitoramento de tráfego pretende usar os dados de demanda de viagens dos passageiros de táxi, ou seja, a zona de embarque e desembarque de cada passageiro em cada instante.

A infraestrutura desta Cidade Inteligente contém uma rede VEC e servidores em nuvem. De acordo com (Li *et al.*, 2025), uma rede VEC é desenvolvida integrando redes veiculares e rede MEC. Similarmente aos trabalhos (Li *et al.*, 2025; Hou *et al.*, 2022; Zhang *et al.*, 2025; Ibrahim; Mokhtar, 2024; Moon; Lim, 2024), a arquitetura da rede VEC deste estudo é composta por uma camada de usuário, uma camada MEC e uma camada *Cloud*.

Na camada de usuário desta rede VEC, encontram-se os terminais que enviam informações de táxi para os nós da camada MEC. Os nós da camada MEC são Unidades

de Borda de Estrada (RSUs) com servidores de borda para processamento e armazenamento de dados. Já na camada *Cloud*, existem servidores que recebem os dados processados pela camada MEC, encaminhando-os para as funções FL ou ITS e enviando os resultados de volta para os nós MEC. A Figura 32 mostra a arquitetura desta rede VEC com suas camadas MEC e *Cloud*.

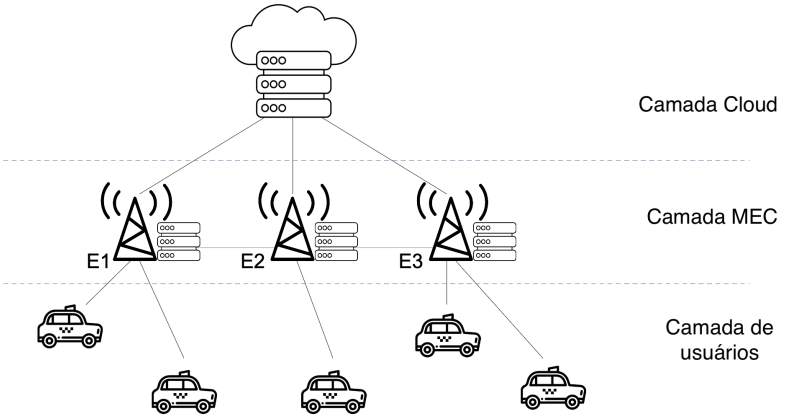


Figure 32: Arquitetura de uma rede VEC com camadas MEC e *Cloud*.

Neste sistema de monitoramento de tráfego, um valor de demanda de táxi $\mathbf{x}_{i,j}$ representa a quantidade de viagens de táxi que partem de qualquer lugar dentro da zona urbana i e viajam para um local de destino dentro da zona urbana j. Além disso, um percurso $P_{i,j}$ das viagens de uma demanda de táxi é representado por um conjunto de zonas urbanas cujos elementos são i, j e as zonas urbanas que são mais prováveis de serem usadas pelos táxis para viajar de i para j. Este percurso padrão, escolhido pelos taxistas dadas as mesmas zonas de origem e destino, é desconhecido pelo sistema de monitoramento.

Além disso, diferentes percursos com diferentes origens e destinos também podem passar por uma mesma zona urbana k. Assim, o sistema de monitoramento de tráfego pode calcular a quantidade de táxis que passam pela zona urbana k em um determinado intervalo de tempo t, ou seja, o valor $y_k^{(t)}$, somando todos os valores de demanda de táxi $\mathbf{x}^{(t)}$ cujos percursos contêm a zona urbana k.

Em sistemas de monitoramento como o descrito em (Varyani *et al.*, 2020), classificações qualitativas são usadas em vez dos valores da métrica de monitoramento. Neste trabalho, um rótulo de classificação $\mathbf{y}_k^{(t)}$ é obtido a partir do valor correspondente $y_k^{(t)}$, que representa a quantidade de táxis que passam pela zona urbana k durante o período t.

Como os nós da camada MEC não conhecem os trajetos P utilizados pelos taxistas em suas viagens, eles não são capazes de calcular os valores $y_k^{(t)}$ nem de obter os rótulos $\mathbf{y}_k^{(t)}$. Assim, um modelo de ML será treinado para inferir um rótulo $\hat{\mathbf{y}}_k^{(t)}$, dados os valores de demanda de táxi de todas as combinações possíveis de zona de origem i e zona de destino j na cidade.

Para usar o modelo de ML, todos os nós MEC compartilham seus valores $\mathbf{x}_{i,j}^{(t)}$ para organizá-los no vetor $\mathbf{X}^{(t)}$, que representa a entrada do modelo. A saída do modelo é

um rótulo $\hat{\mathbf{y}}_{k}^{(t)}$, que representa uma classificação para a zona da cidade k com base na quantidade de táxis que passam por ela considerando um determinado intervalo de tempo t.

6.2 CityZoneClassifier

O modelo *CityZoneClassifier* propõe uma solução para o problema de classificação de zonas em mapas urbanos, apresentado na Seção 6.1. Este modelo infere o rótulo de classificação $\hat{\mathbf{y}}_{k}^{(t)}$ para uma zona urbana k usando como entrada o vetor de demandas de táxi $\mathbf{X}^{(t)}$ em um determinado intervalo de tempo t.

Para inferir o rótulo $\hat{\mathbf{y}}_{k}^{(t)}$, o modelo *CityZoneClassifier* utiliza um MLP com 3 camadas densas. A Figura 33 ilustra graficamente a arquitetura deste modelo.

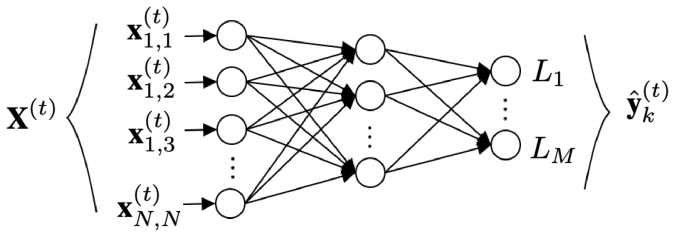


Figure 33: Representação gráfica do modelo *CityZoneClassifier*.

A Cidade Inteligente é dividida em N zonas e o índice k indicado na Figura 33 representa a zona da cidade na qual o modelo foi treinado para classificar. O tamanho do conjunto de classes é M, ou seja, este conjunto de classes é $\{L_1, \ldots, L_M\}$.

A primeira camada deste modelo é a camada de entrada e seu número de neurônios é igual ao quadrado do número de nós (ou seja, $N \times N$); a segunda camada é a camada oculta e seu número de neurônios é aproximadamente igual à metade do número de neurônios da camada anterior (ou seja, $(N \times N)/2$); e a terceira camada é a camada de saída e seu número de neurônios é igual ao número de rótulos possíveis.

A função de ativação das duas primeiras camadas é a função ReLU (He *et al.*, 2018) e a última camada usa uma função logística para sua ativação (Cai *et al.*, 2015; Asadi; Jiang, 2020). A saída de *CityZoneClassifier* é um vetor com as probabilidades para cada rótulo. A inferência do rótulo de classificação $\hat{\mathbf{y}}_{k}^{(t)}$ é realizada através da escolha da classe de maior probabilidade neste vetor da camada de saída do modelo.

6.2.1 Pré-processamento e montagem de um conjunto de dados

O conjunto de dados, utilizado no treinamento e teste do modelo, deve conter amostras com: um vetor $\mathbf{X}^{(t)}$ e um rótulo $\mathbf{y}_{k}^{(t)}$. Antes de montar este conjunto de dados, o parâmetro δ_t deve ser definido. Este parâmetro δ_t representa o tamanho do intervalo de tempo considerado para os valores de informação na entrada e saída do modelo.

Este conjunto de dados é montado na camada MEC utilizando informações dos táxis

sobre as viagens de seus passageiros na cidade. Primeiramente, os táxis compartilham com o nó MEC mais próximo o registro de data e hora e a localização de embarque e desembarque de cada viagem do passageiro, bem como a informação se a viagem passa pela zona urbana k.

Após receber esses dados dos táxis, a camada MEC os compartilha entre seus nós, escolhendo um deles para armazenar todos os valores dos dados das viagens desde o instante t até o instante $t+\delta_t$. Esse nó MEC escolhido contabiliza quantas dessas viagens têm a mesma zona de embarque i e a mesma zona de desembarque j, resultando no valor $\mathbf{x}_{i,j}^{(t)}$. Usando os valores da quantidade de viagens com as outras combinações de zonas de embarque e desembarque, obtém-se o vetor $\mathbf{X}^{(t)}$. Em seguida, o nó MEC também contabiliza quantas dessas viagens passam pela zona urbana k, resultando no valor $y_k^{(t)}$.

Para finalizar as etapas de processamento de uma amostra do conjunto de dados, o nó MEC deve escolher um rótulo de classificação $\mathbf{y}^{(t)}$ para representar o valor $y_k^{(t)}$. A Figura 34 representa as etapas mencionadas para a aquisição de uma amostra do conjunto de dados na camada MEC.

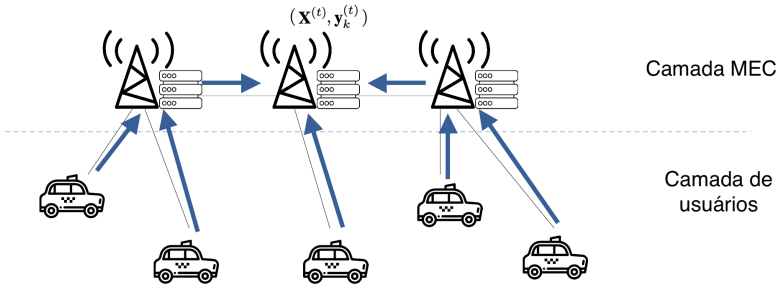


Figure 34: Aquisição de amostra do conjunto de dados na camada MEC.

Usando as etapas de processamento mencionadas anteriormente com os valores de dados de táxi de t' até $t'+\delta_t$, outro nó MEC pode ser usado para salvar a amostra $(\mathbf{X}^{(t')}, \mathbf{y}^{(t')})$.

Após repetir esses passos para todos os valores de dados de táxi disponíveis, a camada MEC obtém um conjunto de dados supervisionado distribuído, pronto para o treinamento e teste do modelo.

6.2.2 Treinamento

O modelo *CityZoneClassifier* é treinado na camada MEC usando o paradigma FL. Isso requer um conjunto de dados supervisionado que é distribuído entre os nós dessa camada MEC.

Nesse treinamento FL, o algoritmo utilizado é o *FedAvg* (McMahan *et al.*, 2017). O objetivo desse treinamento é obter o conjunto de pesos w do modelo *CityZoneClassifier* que otimiza a inferência do rótulo de classificação $\hat{\mathbf{y}}_k^{(t)}$ dado um vetor $\mathbf{X}^{(t)}$. Para o algoritmo *FedAvg*, define-se uma taxa de aprendizagem η e uma função de erro $l(\hat{\mathbf{y}}^{(t)}, \mathbf{y}^{(t)}, w^{(\iota)})$. $l(\hat{\mathbf{y}}^{(t)}, \mathbf{y}^{(t)}, w^{(\iota)}) = \left(\hat{\mathbf{y}}^{(t)} - \mathbf{y}^{(t)}\right)^2$.

Usando a arquitetura da rede VEC mostrada na Figura 32, os nós locais do treinamento FL do modelo são os nós da rede MEC, já que eles armazenam as amostras do conjunto de dados distribuído. O papel do nó global do FL é desempenhado por um servidor na camada *Cloud* da rede VEC.

A Figura 35 representa a comunicação que ocorre durante as iterações para o treinamento do modelo *CityZoneClassifier* usando o algoritmo *FedAvg*. Em azul, é mostrado o caminho dos pesos locais dos nós locais do FL (nós MEC) para o nó global do FL na camada *Cloud*. Em verde, é mostrado o caminho dos pesos globais do nó global do FL para os nós locais do FL na camada MEC.

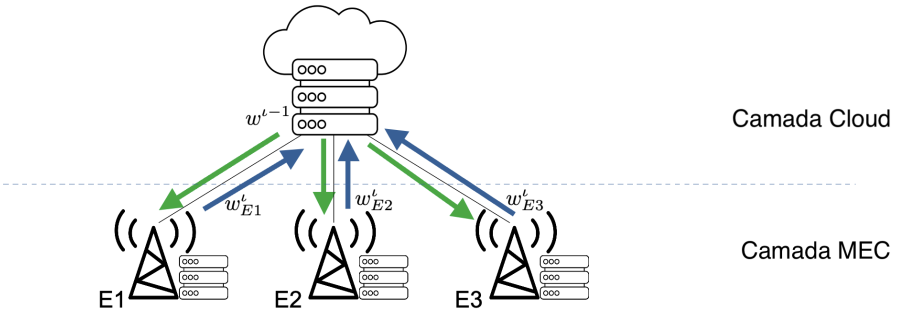


Figure 35: Iterações do treinamento do modelo usando o algoritmo *FedAvg*.

O treinamento FL termina quando os valores dos vetores de peso global de duas ou I iterações adjacentes $w^{\iota},\ldots,w^{\iota-I-1}$ convergem. Isso significa que as diferenças entre os valores dos elementos correspondentes dos vetores $w^{\iota},\ldots,w^{\iota-I-1}$ atingem um limite inferior que define a convergência.

6.2.3 Utilizando o modelo treinado

Após o treinamento do modelo, ele pode ser testado com novas amostras de dados. Se o desempenho do modelo for aceitável, ele pode ser usado para inferir um rótulo de classificação $\hat{\mathbf{y}}_k^{(t)}$ que representa o nível de utilização da zona urbana k nas viagens de táxi.

O modelo treinado e testado está localizado na camada MEC da rede VEC. A entrada do modelo é um vetor $\mathbf{X}^{(t)}$. Portanto, a camada MEC deve processar o vetor $\mathbf{X}^{(t)}$ antes de utilizá-lo no modelo *CityZoneClassifier*. Para construir esse vetor $\mathbf{X}^{(t)}$, a camada MEC recebe inicialmente dos táxis da cidade os valores de dados referentes ao embarque e desembarque de seus passageiros, no intervalo de tempo atual entre os instantes t e $t+\delta_t$.

Em seguida, um nó da camada MEC é selecionado para concentrar os valores de dados recebidos e agregá-los nos valores $\mathbf{x}_{i,j}^{(t)}$, correspondentes a todas as combinações de zona de embarque i e zona de desembarque j. Utilizando todos os valores $\mathbf{x}_{i,j}^{(t)}$ e organizando-os em um vetor, o nó MEC obtém o vetor $\mathbf{X}^{(t)}$.

Utilizando $\mathbf{X}^{(t)}$ como entrada, o modelo *CityZoneClassifier* obtém o rótulo de classificação inferido $\hat{\mathbf{y}}_k^{(t)}$. Finalmente, o rótulo de classificação $\hat{\mathbf{y}}_k^{(t)}$ pode ser utilizado em outras aplicações de monitoramento ou gerenciamento executadas como uma VNF

na camada *Cloud* da rede VEC da Cidade Inteligente.

A Figura 36 representa a comunicação que ocorre na rede VEC ao utilizar o modelo *CityZoneClassifier* para uma função ITS que monitora e gerencia a demanda de táxis na Cidade Inteligente.

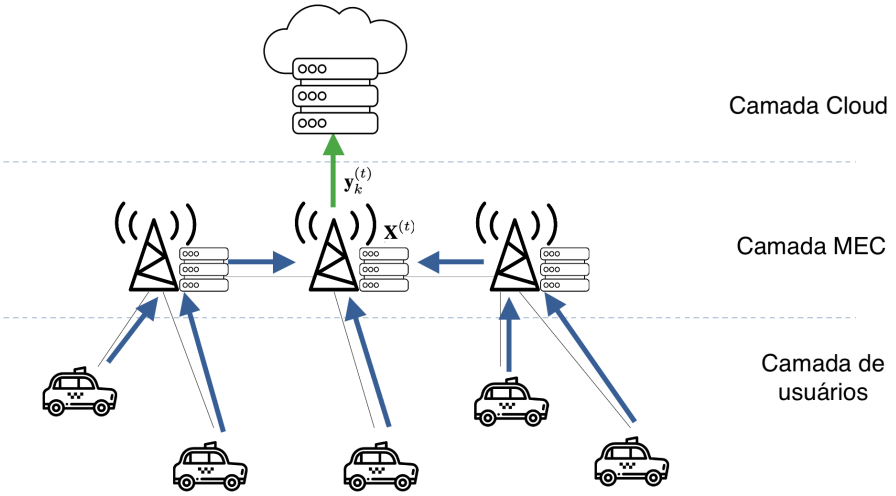


Figure 36: Utilizando a saída do modelo *CityZoneClassifier* em uma função ITS da rede VEC da Cidade Inteligente.

6.3 Resultados e desempenho de classificação

Para avaliar o modelo *CityZoneClassifier*, alguns experimentos foram realizados com um conjunto de dados apresentado na Seção 6.3.1. Esta avaliação utiliza algumas métricas de desempenho definidas na Seção 6.3.2. A Seção 6.3.3 compara o desempenho do Modelo Proposto utilizando diferentes parâmetros e cenários. Já a Seção 6.3.4 compara o Modelo Proposto com outros modelos de classificação tradicionais.

6.3.1 Dataset

Os experimentos descritos nesta Seção 6.3 utilizam um conjunto de dados supervisionado, composto através de dados históricos de veículos e sensores móveis da Tailândia (**??**). Os arquivos baixados contêm o rastreamento GPS de táxis em Bangkok, Tailândia, entre 1º e 31 de dezembro de 2024.

Esses dados de rastreamento GPS baixados contêm registros periódicos com carimbo de data e hora, valores de latitude e longitude da localização do táxi e a informação se a luz de sinalização do táxi está acesa ou apagada. Se a luz de sinalização estiver acesa, o táxi provavelmente está sem passageiros; caso contrário, provavelmente está com passageiros.

Para montar um conjunto de dados supervisionado com esses dados de táxi baixados, os carimbos de data e hora das viagens dos passageiros de táxi serão marcados usando as informações das luzes de sinalização do táxi. Após a marcação desses carimbos de data e hora, as zonas de embarque e desembarque das viagens dos passageiros de táxi

são calculadas usando os valores de latitude e longitude correspondentes, bem como a informação se a viagem passa pela zona urbana escolhida k.

Assim como no trabalho (Munawar; Piantanakulchai, 2024), este conjunto de dados concentrou-se em uma região específica de Bangkok, entre as latitudes 13,5009º e 14,10965º e as longitudes 100,24778º e 100,87591º. De acordo com (Munawar; Piantanakulchai, 2024), essa região é representativa da situação geral dos táxis na cidade, pois contém diversos locais, como lojas, estações de trem, bairros e atrações turísticas. Essa região é dividida em um *grid* 7×7, cada *grid* cobrindo 10 km × 10 km, assim como em (Munawar; Piantanakulchai, 2024).

As zonas urbanas deste conjunto de dados são definidas como as 49 *grids*, obtidas pela divisão da região representativa de Bangkok. A zona urbana k escolhida para a classificação do modelo é aquela localizada no centro do mapa *grid*, ou seja, o k no conjunto de dados corresponde ao índice 25. O parâmetro δ_t do conjunto de dados é definido como 1 hora.

O resultado da montagem do conjunto de dados é um *dataset* supervisionado com 743 amostras, cada uma contendo: um vetor $\mathbf{X}^{(t)}$ com o formato 49×49; e um rótulo $\mathbf{y}_k^{(t)}$ tal que k seja igual a 25.

Além de ser supervisionado, o conjunto de dados também será distribuído para ser utilizado no treinamento do modelo. Durante os experimentos, a quantidade de nós MEC pode variar. Portanto, ambos os conjuntos de dados são divididos utilizando regras específicas em cada experimento.

6.3.2 Avaliação de desempenho

Para avaliar o Modelo Proposto, experimentos foram executados para coletar valores de métricas de desempenho. O conjunto de dados supervisionado foi dividido em dados de treinamento e teste. Os dados de teste foram usados como entrada do modelo para obter os rótulos de classificação inferidos $\hat{\mathbf{y}}^{(t)}$, que foram comparados com os rótulos esperados correspondentes $\mathbf{y}^{(t)}$ no conjunto de dados supervisionado.

Para maximizar o uso dos dados disponíveis na medição do desempenho, o conceito de Validação Cruzada *K-Fold* foi usado para dividir os dados de treinamento e teste de cada um dos conjuntos de dados. A Validação Cruzada 5-*Fold* foi usada no conjunto de dados.

De acordo com (Fawcett, 2006), as métricas de avaliação de modelos, denominadas acurácia, precisão, *recall* e *F1-score*, podem ser calculadas a partir da comparação entre a resposta da aplicação do modelo $\hat{\mathbf{y}}^{(t)}$ e seu valor de referência $\mathbf{y}^{(t)}$ no conjunto de dados supervisionado.

Nos experimentos, existe uma configuração padrão do modelo que varia de acordo com o parâmetro que está sendo analisado. Os valores padrão dos parâmetros são: taxa de aprendizado $\eta = 0,01$, tamanho do lote $bs = 32$, número de épocas $ep = 10$, número de classes igual a 2 e número de nós MEC igual a 3.

6.3.3 Resultados variando os parâmetros do modelo e da rede

Esta seção apresenta comparações de desempenho entre os experimentos com diferentes parâmetros do modelo, parâmetros FL e configurações da rede VEC. Esta primeira comparação de desempenho é entre experimentos com o modelo *CityZoneClassifier* variando o parâmetro η.

Na Figura 37, os maiores valores de acurácia são obtidos pelo modelo com $\eta = 0,05$, $\eta = 0,01$ e $\eta = 0,005$; respectivamente 0,846, 0,846 e 0,859. Dentre essas taxas de aprendizado, o modelo com $\eta = 0,01$ é mais estável do que com $\eta = 0,05$ e aprende mais rapidamente do que com $\eta = 0,005$.

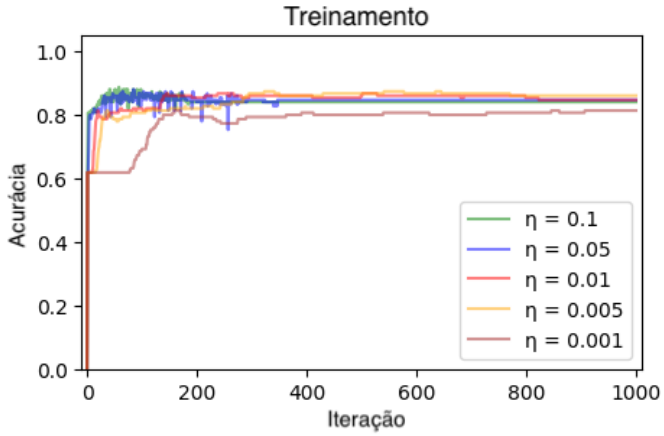


Figure 37: Valores de acurácia variando a taxa de aprendizado.

Outra comparação, presente nesta seção, é entre experimentos com o modelo variando o número de épocas ep e o tamanho do lote bs. A Figura 38 mostra que a combinação de $ep = 10$ e $bs = 32$ alcançou uma acurácia satisfatória com os menores valores de parâmetros.

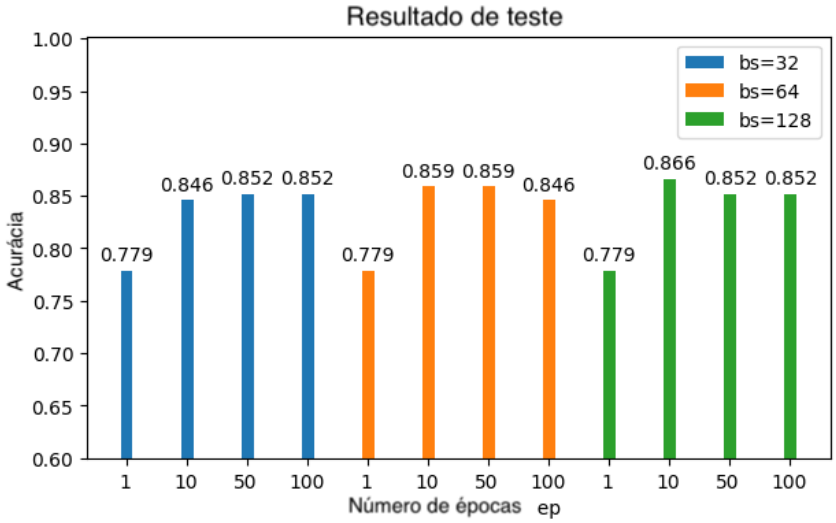


Figure 38: Valores de acurácia variando ep e bs.

Outra comparação de desempenho é feita entre experimentos com diferentes números

de nós da camada MEC na arquitetura da rede VEC. Esses nós da camada MEC são nós locais que participam do FL que treina o modelo *CityZoneClassifier*.

A tabela 9 mostra que o maior valor de *F1-score* é obtido quando o modelo está sendo executado numa configuração MEC de 3 nós. O maior valor de *F1-score* obtido foi de 0,8469, cerca de 1% superior ao segundo maior valor de *F1-score*, encontrado na configuração MEC de 10 nós. O número de nós MEC na arquitetura de rede VEC não interferiu significativamente no desempenho do modelo com este conjunto de dados.

Table 9: Desempenho do *CityZoneClassifier* em diferentes configurações MEC.

Métrica	2-node MEC	3-node MEC	5-node MEC	10-node MEC
Precisão	0,8524	**0,8591**	0,8456	**0,8591**
Precisão	0,8501	0,8593	0,8442	**0,8632**
Recall	0,8337	**0,8392**	0,8249	0,8358
F1-score	0,8403	**0,8469**	0,8323	0,8456

A última comparação desta seção é feita treinando o modelo variando o número de classes entre 2, 3, 4 e 5 classes. A tabela 10 mostra que o maior valor de *F1-score* é obtido quando o modelo é treinado com 2 classes. Esse maior valor de *F1-score* é 0,960 e é cerca de 4% maior que o segundo maior valor de *F1-score*, que é obtido no modelo treinado com 3 classes. Isso significa que o número de nós interferiu significativamente no desempenho do modelo usando este conjunto de dados.

Table 10: Desempenho do *CityZoneClassifier* com diferentes números de classes.

Métrica	2 classes	3 classes	4 classes	5 classes
Acurácia	**0,8591**	0,8317	0,8084	0,799
Precisão	**0,8593**	0,8151	0,7490	0,705
Recall	**0,8392**	0,7855	0,6923	0,622
F1-score	**0,8469**	0,8046	0,7118	0,646

6.3.4 Resultados comparados com modelos de classificação tradicionais

Nesta seção, os resultados de desempenho obtidos pelo *CityZoneClassifier* são comparados com os resultados de outros modelos de classificação tradicionais da literatura. Esses modelos, para comparação com o *CityZoneClassifier*, são Árvore de Decisão (DT), MLP, Regressão Linear (LR), Regressão Logística (LGR) e Máquina de Vetores de Suporte (SVM). Com exceção do modelo *CityZoneClassifier*, que utiliza o paradigma de FL, todos os outros modelos de classificação são treinados utilizando um método centralizado.

Na Tabela 11, o modelo *CityZoneClassifier* é treinado com os seguintes parâmetros: taxa de aprendizado $\eta = 0,01$, tamanho de lote $\mathtt{bs} = 32$ e número de épocas $\mathtt{ep} = 1$.

O modelo DT utiliza o índice de Gini como critério de decisão e as divisões ao longo dos ramos dessa árvore são sempre feitas com o melhor valor encontrado. O modelo MLP é o mesmo definido para o modelo proposto, porém, este modelo MLP utiliza um

treinamento tradicional e centralizado. O modelo LR utiliza as equações de otimização do método dos mínimos quadrados para ser treinado. O modelo de Regressão Logística é treinado utilizando o algoritmo *Broyden-Fletcher-Goldfarb-Shanno* (LBFGS). O modelo SVM utiliza um kernel polinomial de grau igual a 1 para treinar seus pesos, e a dimensão do hiperplano de entrada do SVM é igual a N^2.

Após o treinamento de todos os modelos, seus valores de métricas de desempenho foram obtidos. A Tabela 11 mostra que o maior valor de *F1-score* foi obtido pelo Modelo Proposto. Esse maior valor de *F1-score* é 0,850 e é cerca de 2% maior que o segundo maior valor de *F1-score*, obtido pelo modelo MLP com treinamento centralizado.

Table 11: Desempenho do modelo *CityZoneClassifier* comparado com modelos de classificação tradicionais.

Métrica	DT	MLP	LR	LGR	SVM	CityZone-Classifier
Acurácia	0,757	0,846	0,770	0,834	0,823	**0,864**
Precisão	0,715	0,839	0,801	0,817	0,799	**0,870**
Recall	0,709	0,832	0,552	0,779	0,771	**0,840**
F1-score	0,712	0,835	0,653	0,793	0,783	**0,850**

Na Tabela 11, os valores que representam as métricas do *CityZoneClassifier* são médias dos resultados de 20 execuções do experimento. A Figura 39 apresenta um diagrama *Boxplot* sobre as distribuições dos valores das métricas de *CityZoneClassifier*.

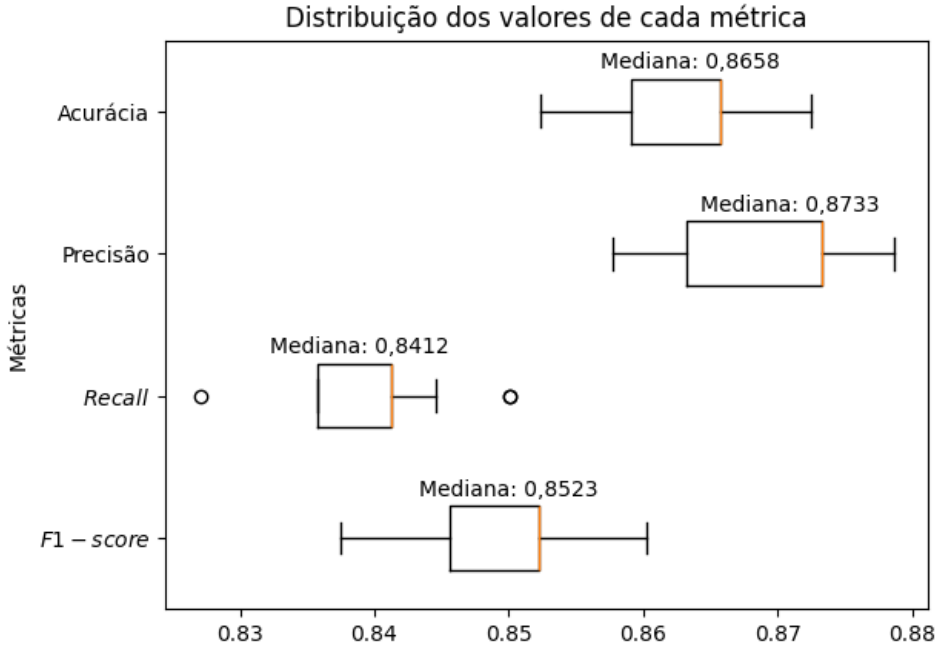


Figure 39: Distribuição dos valores de cada métrica do *CityZoneClassifier*.

De acordo com a Tabela 11, a acurácia e o *F1-score* do modelo *CityZoneClassifier* são ligeiramente superiores aos de outros modelos de classificação tradicionais. A vantagem do modelo *CityZoneClassifier* reside no método que utiliza o paradigma FL

para distribuir os custos de processamento e armazenamento e, com isso, os dados sensíveis não precisam ser enviados para um servidor *Cloud*.

6.4 Usabilidade do *CityZoneClassifier* numa tarefa de ITS

Esta seção apresenta um exemplo de como o modelo *CityZoneClassifier* pode ser usado para auxiliar numa tarefa de ITS que evita o congestionamento para táxis numa Cidade Inteligente.

Neste exemplo, o mapa *grid* da Cidade Inteligente possui o formato 7×7, como em (Munawar; Piantanakulchai, 2024). Logo, ao colher as informações de origem e destino de todas as corridas de táxis nesta cidade numa determinada faixa de tempo, um vetor $\mathbf{X}^{(t)}$ pode ser montado com o formato 49×49. Cada posição deste vetor representa a demanda de corrida de táxi de uma mesma origem e destino.

Desta forma, o vetor $\mathbf{X}^{(t)}$ pode ser usado como entrada de cada instância do modelo *CityZoneClassifier* que classifica cada posição k do mapa *grid* da cidade através de um rótulo $\mathbf{y}_k^{(t)}$ representando a utilização de k por táxis.

Supondo que as vias desta cidade possuem uma faixa exclusiva para táxis, é possível inferir o congestionamento nestas faixas de táxis das posições k do mapa *grid*, através dos rótulos de classificação $\mathbf{y}_k^{(t)}$ obtidos pelas instâncias do modelo *CityZoneClassifier*.

Usando essas inferências de congestionamento das faixas de táxis das posições k do mapa *grid*, sugestões alternativas de rota podem ser enviadas para os motoristas dos táxis para evitarem as posições k congestionadas.

6.5 Comentários finais

O trabalho deste capítulo propõe o modelo *CityZoneClassifier* para classificar uma zona urbana de acordo com seu nível de utilização em viagens de passageiros de táxi, baseando-se em informações sobre as zonas de embarque e desembarque dessas viagens.

O *CityZoneClassifier* é baseado em ML e é treinado utilizando o paradigma FL. Os nós que participam do FL estão em uma camada MEC entre os táxis e a camada de Nuvem da rede VEC da Cidade Inteligente. Após o treinamento de um modelo para cada zona da cidade, esta camada MEC aplica esses modelos para informar os servidores em Nuvem da rede VEC sobre as classificações das zonas urbanas. Os rótulos de classificação representam o nível de utilização das zonas da cidade em viagens de táxi com passageiros.

O desempenho do *CityZoneClassifier* é obtido por meio de experimentos utilizando um conjunto de dados composto por informações reais de GPS. Esses experimentos são realizados variando o número de nós na camada MEC e o número de classes no modelo *CityZoneClassifier*. Posteriormente, o desempenho do modelo é comparado com o desempenho de outros modelos tradicionais de classificação. Por fim, é apresentado um exemplo de como o modelo pode ser usado para tomar uma ação de gerenciamento de

tráfego usando ITS numa Cidade Inteligente.

7. Conclusão

Esta tese de doutorado teve o objetivo de investigar soluções para realizar o monitoramento de enlaces de rede, quando esta tarefa não possui o auxílio de medidores dedicados e nem informações de topologia da rede e sistema de roteamento de pacotes. Por exemplo, em tarefas virtualizadas, é comum que algumas informações não sejam disponibilizadas para garantir privacidade. Além disso, em sistemas de orquestração de tarefas, também é comum que tarefas sejam reduzidas para gastarem o mínimo de recursos de armazenamento e processamento de dados.

Então, este monitoramento de rede acontece através da classificação dos enlaces usando rótulos que representam as suas respectivas taxas de utilização. A inferência dos rótulos ocorre a partir de matrizes de tráfego, contendo a demanda de comunicação entre todos os nós da rede.

Portanto, a solução geral é treinar um modelo de ML para realizar esta tarefa de classificação de enlaces de rede. Neste treinamento, é usado um conjunto de dados supervisionado, no qual cada amostra contém uma matriz de tráfego e um rótulo de classificação para um enlace de rede num determinado instante de tempo. Com isto, o modelo aprende a classificar os enlaces de rede usando novas matrizes de tráfego, sem a necessidade do conhecimento dos caminhos dos pacotes que trafegam nesta rede.

Em paralelo, foram investigadas algumas versões deste problema em três diferentes cenários, cada um com suas características particulares. O primeiro cenário apresenta o monitoramento de enlaces de uma SDN usando um modelo de ML com duas etapas. A primeira delas infere os valores de taxa de utilização do enlace e a segunda infere o seu rótulo de classificação.

O segundo cenário apresenta o monitoramento de enlaces de uma SDN, realizado por uma camada MEC entre os nós clientes dessa rede e o controlador da SDN. A utilização desta camada MEC traz mais segurança e economia de recursos do sistema. Isto porque os dados de tráfego dos nós da rede chegam nesta camada MEC, são usados pelo modelo para classificar os enlaces, e somente os rótulos de classificação dos enlaces são enviados para as tarefas virtualizadas do controlador da SDN, que se encontram mais distantes dos nós clientes da rede.

Além disso, ainda neste segundo cenário, a camada MEC armazena de forma distribuída o conjunto de dados de treinamento do modelo de ML para classificação dos enlaces de rede. O treinamento ocorre, através do paradigma FL, usando os nós desta camada MEC como locais e um controlador da SDN como agregador do modelo global. Também são investigados os impactos da aplicação de uma técnica de criptografia durante as comunicações deste treinamento.

O terceiro cenário, que foi apresentado neste estudo, aborda um problema e uma solução semelhantes, porém no contexto de ITS e Cidades Inteligentes. Ao invés da tarefa de classificar enlaces de rede quanto à sua taxa de utilização, o problema deste cenário se trata da tarefa de classificação de zonas de uma cidade quanto à sua utilização

em caminhos percorridos por um determinado transporte modal, e.g. táxis. Neste caso, as zonas desta cidade realizam o mesmo papel tanto dos nós da rede (disponibilizando as suas demandas de trajetos para outras zonas da cidade) quanto dos enlaces de rede (as zonas da cidade também fazem parte dos caminhos percorridos pelos táxis).

Como no segundo cenário, este terceiro cenário também conta com uma camada MEC para organizar os dados das zonas da cidade que estão mais próximas. E os nós desta camada MEC se comunicam de forma dedicada entre si para pre-processar dados, treinar o modelo usando o paradigma FL e utilizar o modelo treinado para classificar as zonas da cidade. O resultado desta classificação é enviado para funções virtualizadas do controlador de ITS da Cidade Inteligente.

Em todos estes cenários, são apresentadas diversas análises dos modelos propostos, tais como a sua comparação de desempenho frente a outros modelos tradicionais de classificação encontrados na literatura e a comparação do desempenho destes modelos propostos variando alguns parâmetros e características do ambiente inserido.

Estas análises apresentaram resultados satisfatórios de desempenho dos modelos propostos, justificando a utilização das metodologias associadas a eles, como por exemplo as vantagens obtidas pelo fato de se treinar e executar o modelo de classificação numa camada MEC da rede. Além disso, foram demonstrados os impactos de realizar modificações no sistema de monitoramento e, nos cenários 1 e 2, foram demonstrados como os modelos de classificação de enlaces de rede podem auxiliar na aplicação de ações corretivas para mitigar problemas relacionados ao congestionamento de pacotes.

7.1 Respostas às Questões de Pesquisa

As questões de pesquisa propostas no primeiro capítulo desta tese podem ser respondidas conforme abaixo:

Q1 – É viável treinar um modelo de ML para classificar os enlaces de rede quanto à sua utilização usando apenas as demandas de tráfego dos nós, sem o conhecimento da topologia da rede ou a informação do sistema de roteamento de pacotes?

R1 –> Sim. Os experimentos dos Capítulos 4 e 5 demonstraram que após os modelos *LinkClassifier* e *NetLinkClassifier* obtiveram um nível alto de desempenho em suas classificações de enlaces.

Q2 – Caso o *dataset* para treinar este modelo se encontre distribuídos próximos dos nós de origem dos dados, é efetivo usar a abordagem FL?

R2 –> Sim. Os Capítulos 5 e 6 demonstraram a convergência e um alto desempenho dos modelos *NetLinkClassifier* e *CityZoneClassifier*, respectivamente.

Q3 – Qual é o impacto de adicionar uma criptografia nestes dados durante o aprendizado do modelo?

R3 –> O Capítulo 5 demonstrou que o modelo obteve o mesmo nível de acurácia, com ou sem o sistema de criptografia em treinamento. Porém, com a criptografia, a convergência do modelo ocorreu com mais rodadas.

Q4 – Como a classificação de enlaces da rede pode auxiliar um serviço de distribuição de cargas nesta rede?

R4 –> Os Capítulos 4 e 5 demonstraram que o modelo de classificação de enlaces de rede pode ser usado para auxiliar na decisão de divisão de tráfego entre 2 ou mais rotas, ao detectar um congestionamento no enlace.

Q5 – É possível aplicar esta ideia de classificação de enlaces de rede em outras áreas? Em qual área, por exemplo?

R5 –> Sim. O Capítulos 6 demonstrou que, de forma semelhante à classificação de enlaces de rede, um modelo pode ser treinado para classificar zonas de uma cidade quanto ao seu uso em rotas de um transporte modal.

7.2 Trabalhos Futuros

Ao longo deste trabalho de doutorado, foram identificadas algumas novos tópicos para pesquisas futuras.

Primeiramente, foram identificadas novas ideias de pesquisas futuras para a incorporação, ao problema principal, algumas características particulares já existentes nos cenários apresentados nesta tese. Com isso, tais características receberiam novas análises e com mais detalhes do que nas que foram apresentadas neste trabalho. Um exemplo de possível incorporação ao problema principal é a utilização da camada MEC e a aplicação de algoritmo de criptografia durante o transporte de dados de medida para a montagem do *dataset* distribuído e durante o treinamento do modelo usando o paradigma FL.

Outra ideia de pesquisa futura é realizar um estudo mais específico sobre a segurança de dados de medidas usadas no modelo de classificação. Dessa forma, além de aplicações de novos algoritmos de criptografia, outras metodologias e configurações de rede podem ser aplicadas para trazer mais segurança no transporte e armazenamento dos dados medidos pelos clientes da rede. Um exemplo de técnica a ser investigada para otimizar a segurança de dados durante a realização de operações matemáticas é a aplicação de Criptografia Homomórfica. E um exemplo de configuração de rede paralela à rede principal que pode ser considerada para trafegar os metadados que operam o monitoramento é uma *Blockchain*.

Mais uma ideia de pesquisa futura é investigar a implementação de Aceleração de Hardware de MLP. Este conceito se refere ao uso de componentes especializados (tais como Unidades de Processamento Gráfico, Unidades de Processamento de Tensor ou Unidades de Processamento Neural) para acelerar o treinamento e a inferência de redes

neurais, descarregando tarefas computacionalmente intensivas da CPU principal, para alcançar menor interferência de escalabilidade no desempenho do modelo de classificação (Chen *et al.*, 2025; Khan; Alhartomi, 2024).

Por fim, mais uma ideia de pesquisa futura é investigar em quais outros contextos existem problemas semelhantes ao problema de monitoramento de enlaces de rede. Como no terceiro cenário desta tese que explorou um problema de ITS e Cidades Inteligentes, é provável que exista problemas semelhantes em outras áreas ou subáreas de pesquisa. Alguns exemplos dessas área e subáreas de pesquisa são redes de espaços aéreos, redes hidráulicas e redes de linhas de transmissão de energia elétrica.

References

AKYILDIZ, I. F.; LEE, A.; WANG, P.; LUO, M.; CHOU, W. A roadmap for traffic engineering in sdn-openflow networks. **Computer Networks**, Elsevier, v. 71, p. 1–30, 2014.

AL-FARES, M. *et al.* Hedera: dynamic flow scheduling for data center networks. In: SAN JOSE, USA. **Nsdi**. [S.l.], 2010. v. 10, n. 8, p. 89–92.

ALAM, I. *et al.* A survey of network virtualization techniques for internet of things using sdn and nfv. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 53, n. 2, p. 1–40, 2020.

ALKENANI, J.; NASSAR, K. A. Network performance analysis using packets probe for passive monitoring. **Informatica**, v. 46, n. 7, 2022.

ALQIAM, A. A. *et al.* Transferable neural wan te for changing topologies. In: **Proceedings of the ACM SIGCOMM 2024 Conference**. [S.l.: s.n.], 2024. p. 86–102.

ALSHUQAYRAN, N.; ALI, N.; EVANS, R. A systematic mapping study in microservice architecture. In: IEEE. **2016 IEEE 9th international conference on service-oriented computing and applications (SOCA)**. [S.l.], 2016. p. 44–51.

AMOROSO, R.; ESPOSITO, F.; MERANI, M. L. Estimation of traffic matrices via super-resolution and federated learning. In: **16th ACM CoNEXT**. [S.l.: s.n.], 2020. p. 560–561.

AMOROSO, R.; PAPPONE, L.; ESPOSITO, F. A federated learning approach to traffic matrix estimation using super-resolution techniques. In: IEEE. **2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)**. [S.l.], 2023. p. 473–476.

ASADI, B.; JIANG, H. On approximation capabilities of relu activation and softmax output layer in neural networks. **arXiv preprint arXiv:2002.04060**, 2020.

ATHANASAKI, S.; TSIKALOUDAKI, K. Smart buildings for smart cities: Analysis of the smart readiness indicator. **Green Energy Sustain**, v. 2, n. 0005, 2022.

BASU, K.; YOUNAS, M.; TOW, A. W. W.; BALL, F. Performance comparison of a sdn network between cloud-based and locally hosted sdn controllers. In: IEEE. **2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService)**. [S.l.], 2018. p. 49–55.

BIBRI, S. E.; KROGSTIE, J. Smart sustainable cities of the future: An extensive interdisciplinary literature review. **Sustainable cities and society**, Elsevier, v. 31, p. 183–212, 2017.

BISHOP, C. M.; NASRABADI, N. M. **Pattern recognition and machine learning**. [S.l.]: Springer, 2006. v. 4.

BLINOWSKI, G.; OJDOWSKA, A.; PRZYBYŁEK, A. Monolithic vs. microservice architecture: A performance and scalability evaluation. **IEEE access**, IEEE, v. 10, p. 20357–20374, 2022.

BOTHRA, C.; GAO, J.; RAO, S.; RIBEIRO, B. Veritas: Answering causal queries from video streaming traces. In: **Proceedings of the ACM SIGCOMM 2023 Conference**. [S.l.: s.n.], 2023. p. 738–753.

BOUCETTA, C.; NOUR, B.; CUSIN, A.; MOUNGLA, H. Qos in iot networks based on link quality prediction. In: IEEE. **ICC 2021-IEEE International Conference on Communications**. [S.l.], 2021. p. 1–6.

BRO, R.; SIDIROPOULOS, N. D.; SMILDE, A. K. Maximum likelihood fitting using ordinary least squares algorithms. **Journal of Chemometrics: A Journal of the Chemometrics Society**, Wiley Online Library, v. 16, n. 8-10, p. 387–400, 2002.

CAI, C.; XU, Y.; KE, D.; SU, K. Deep neural networks with multistate activation functions. **Computational Intelligence and Neuroscience**, Wiley Online Library, v. 2015, n. 1, p. 721367, 2015.

CAI, M.; HONG, L.; XIONG, C. Data-driven traffic zone division in smart city: Framework and technology. **Sustainable Energy Technologies and Assessments**, Elsevier, v. 52, p. 102251, 2022.

CHEN, B. *et al.* Comet: Co-optimization of a cnn model using efficient-hardware obc techniques. **arXiv preprint arXiv:2510.03516**, 2025.

CHEN, J. *et al.* Alblp: Adaptive load-balancing architecture based on link-state prediction in software-defined networking. **Wireless Communications and Mobile Computing**, Wiley Online Library, v. 2022, n. 1, p. 8354150, 2022.

CHEN, S.; HE, D.; MING, Z.; CUI, L. Fogeye: Towards performance efficient fog networking telemetry. In: IEEE. **2023 IEEE Globecom Workshops (GC Wkshps)**. [S.l.], 2023. p. 1662–1667.

CHOWDHURY, S. R.; BARI, M. F.; AHMED, R.; BOUTABA, R. Payless: A low cost network monitoring framework for software defined networks. In: IEEE. **2014 IEEE Network Operations and Management Symposium (NOMS)**. [S.l.], 2014. p. 1–9.

CLAUSEN, H.; MANOCHA, A.; GIBSON, M. Detecting proxies relaying streaming internet traffic. In: **IEEE INFOCOM WKSHPS**. [S.l.: s.n.], 2022. p. 1–6.

DANGER, J.-L. *et al.* On the performance and security of multiplication in gf (2 n). **Cryptography**, MDPI, v. 2, n. 3, p. 25, 2018.

DHANDA, S. S.; SINGH, B.; JINDAL, P.; KUMAR, V.; GUPTA, S. K. Aes-8: A lightweight aes for resource-constrained iot devices. **Transactions on Emerging Telecommunications Technologies**, Wiley Online Library, v. 36, n. 3, p. e70094, 2025.

FAWCETT, T. An introduction to roc analysis. **Pattern recognition letters**, Elsevier, v. 27, n. 8, p. 861–874, 2006.

FORTZ, B.; THORUP, M. Internet traffic engineering by optimizing ospf weights. In: IEEE. **Proceedings IEEE INFOCOM 2000. conference on computer communications. Nineteenth annual joint conference of the IEEE computer and communications societies (Cat. No. 00CH37064)**. [S.l.], 2000. v. 2, p. 519–528.

GAO, T. *et al.* Cost-efficient vnf placement and scheduling in public cloud networks. **IEEE Transactions on Communications**, IEEE, v. 68, n. 8, p. 4946–4959, 2020.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. [S.l.]: MIT press, 2016.

GUPTA, R.; DANILOV, C.; ECKHARDT, J.; BERNARD, K.; NAHRSTEDT, K. Characterizing container performance in edge computing. In: **Proceedings of the ACM SIGCOMM 2025 Posters and Demos**. [S.l.: s.n.], 2025. p. 94–96.

HARTMANN, F. **Federated learning**. Dissertação (Mestrado) — Institut fur Informatik der Freien Universität Berlin, Available on: https://florian.github.io/federated-learning/, 2018.

HE, J.; LI, L.; XU, J.; ZHENG, C. Relu deep neural networks and linear finite elements. **arXiv preprint arXiv:1807.03973**, 2018.

HOU, L.; GREGORY, M. A.; LI, S. A survey of multi-access edge computing and vehicular networking. **IEEE Access**, IEEE, v. 10, p. 123436–123451, 2022.

HU, F.; HAO, Q.; BAO, K. A survey on software-defined network and openflow: From concept to implementation. **IEEE Communications Surveys & Tutorials**, IEEE, v. 16, n. 4, p. 2181–2206, 2014.

HU, S. *et al.* Urban function classification at road segment level using taxi trajectory data: A graph convolutional neural network approach. **Computers, Environment and Urban Systems**, Elsevier, v. 87, p. 101619, 2021.

HUMAYUN, M.; ALSAQER, M. S.; JHANJHI, N. Energy optimization for smart cities using iot. **Applied artificial intelligence**, Taylor & Francis, v. 36, n. 1, p. 2037255, 2022.

IBRAHIM, A.; MOKHTAR, B. What if vec is moving: Probabilistic model of task execution through offloading in vehicular computing environments. **IEEE Access**, IEEE, 2024.

IDZIKOWSKI, F.; RAACK, C. **Thrid versior of SNDlib – Survivable fixed telecommunication Network Design**. 2023. Disponível em: <https://sndlib.put.poznan.pl/home.action>.

KAKKAVAS, G.; FRYGANIOTIS, N.; KARYOTIS, V.; PAPAVASSILIOU, S. Generative deep learning techniques for traffic matrix estimation from link load measurements. **IEEE Open Journal of the Communications Society**, IEEE, 2024.

KAUR, K.; MANGAT, V.; KUMAR, K. A review on virtualized infrastructure managers with management and orchestration features in nfv architecture. **Computer Networks**, Elsevier, v. 217, p. 109281, 2022.

KHAN, M. T.; ALHARTOMI, M. A. Digit-serial da-based fixed-point rnns: A unified approach for enhancing architectural efficiency. **IEEE Transactions on Neural Networks and Learning Systems**, IEEE, 2024.

KIM, H.; SEO, H. Optimizing aes-gcm on 32-bit arm cortex-m4 microcontrollers: fixslicing and face-based approach. **ACM Transactions on Embedded Computing Systems**, ACM New York, NY, v. 24, n. 6, p. 1–24, 2025.

KODIALAM, M.; LAKSHMAN, T. Network link weight setting: A machine learning based approach. In: IEEE. **IEEE INFOCOM 2022-IEEE Conference on Computer Communications**. [S.l.], 2022. p. 2048–2057.

KOLHE, R. V.; WILLIAM, P.; YAWALKAR, P. M.; PAITHANKAR, D. N.; PABALE, A. R. Smart city implementation based on internet of things integrated with optimization technology. **Measurement: Sensors**, Elsevier, v. 27, p. 100789, 2023.

KOTTUR, S. Z.; KADIYALA, K.; TAMMANA, P.; SHAH, R. Implementing chacha based crypto primitives on programmable smartnics. In: **Proceedings of the ACM SIGCOMM Workshop on Formal Foundations and Security of Programmable Network Infrastructures**. [S.l.: s.n.], 2022. p. 15–23.

LABONNE, M.; CHATZINAKIS, C.; OLIVEREAU, A. Predicting bandwidth utilization on network links using machine learning. In: IEEE. **2020 European Conference on Networks and Communications (EuCNC)**. [S.l.], 2020. p. 242–247.

LI, Y.; FENG, L.; MEI, M.; ALI, A.; SHAH, Z. Joint optimization for volatile federated learning in vehicular edge computing: A deep reinforcement learning approach. **IEEE Transactions on Vehicular Technology**, IEEE, 2025.

LI, Y.; OH, B.-H. Mptcp-based link congestion detection and traffic control scheme in software-defined networks. **ICT Express**, Elsevier, v. 10, n. 4, p. 735–746, 2024.

LI, Y.; YAN, X. Research on traffic zone partition method based on two-level partition theory. In: IOP PUBLISHING. **IOP Conference Series: Materials Science and Engineering**. [S.l.], 2019. v. 688, n. 2, p. 022015.

LIU, L. *et al.* Multilevel federated learning-based intelligent traffic flow forecasting for transportation network management. **IEEE Transactions on Network and Service Management**, IEEE, v. 20, n. 2, p. 1446–1458, 2023.

LIU, Y.; ZHENG, Y. F. One-against-all multi-class svm classification using reliability measures. In: IEEE. **Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.** [S.l.], 2005. v. 2, p. 849–854.

MARCO, A. D.; MANGANO, G. Evolutionary trends in smart city initiatives. **Sustainable futures**, Elsevier, v. 3, p. 100052, 2021.

MARTÍ, P.; SERRANO-ESTRADA, L.; NOLASCO-CIRUGEDA, A. Social media data: Challenges, opportunities and limitations in urban studies. **Computers, Environment and Urban Systems**, Elsevier, v. 74, p. 161–174, 2019.

MCGREW, D.; VIEGA, J. The galois/counter mode of operation (gcm). **submission to NIST Modes of Operation Process**, NIST US, v. 20, p. 0278–0070, 2004.

MCMAHAN, B.; MOORE, E.; RAMAGE, D.; HAMPSON, S.; ARCAS, B. A. y. Communication-efficient learning of deep networks from decentralized data. In: **20th AISTATS**. [S.l.: s.n.], 2017. p. 1273–1282.

MESQUITA, L. A. J.; ASSIS, K. D. R. Traffic matrix prediction for optical networks. In: **IEEE IMOC**. [S.l.: s.n.], 2019. p. 1–3.

MOON, S.; LIM, Y. Client selection for federated learning in vehicular edge computing: A deep reinforcement learning approach. **IEEE Access**, IEEE, 2024.

MUNAWAR, A.; PIANTANAKULCHAI, M. A collaborative privacy-preserving approach for passenger demand forecasting of autonomous taxis empowered by federated learning in smart cities. **Scientific Reports**, Nature Publishing Group UK London, v. 14, n. 1, p. 2046, 2024.

NAVARRO-ORTIZ, J. *et al.* A survey on 5g usage scenarios and traffic models. **IEEE Communications Surveys & Tutorials**, IEEE, v. 22, n. 2, p. 905–929, 2020.

ORLOWSKI, S.; PIÓRO, M.; TOMASZEWSKI, A.; WESSÄLY, R. SNDlib 1.0–Survivable Network Design Library. **Networks**, v. 55, n. 3, p. 276–286, 2010. Disponível em: <http://www3.interscience.wiley.com/journal/122653325/abstract>.

PAPPONE, L.; SACCO, A.; ESPOSITO, F. On traffic matrix estimation via super-resolution and federated learning. **IEEE Transactions on Network and Service Management**, IEEE, 2024.

PEI, T. *et al.* A new insight into land use classification based on aggregated mobile phone data. **International Journal of Geographical Information Science**, Taylor & Francis, v. 28, n. 9, p. 1988–2007, 2014.

QIAN, P.; WANG, N.; TAFAZOLLI, R. User intent driven path switching in video delivery-an edge computing based approach. In: IEEE. **IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)**. [S.l.], 2022. p. 1–6.

SELENT, D. Advanced encryption standard. **Rivier Academic Journal**, v. 6, n. 2, p. 1–14, 2010.

SHAHRBABAKI, P. P.; COUTINHO, R. W.; SHAYAN, Y. R. A novel sdn-enabled edge computing load balancing scheme for iot video analytics. In: IEEE. **GLOBECOM 2022-2022 IEEE Global Communications Conference**. [S.l.], 2022. p. 5025–5030.

SILVA, N. A. A. D.; CAMPOS, C. A. V.; LUCENA, S. C. D. Federated learning approach for edge application of network link classification based on traffic matrices. **IEEE Access**, IEEE, 2026.

SILVA, N. A. A. da; CAMPOS, C. A. V.; LUCENA, S. C. de. Inferência da qualidade do serviço em enlaces de rede através de um método baseado em aprendizado federado. In: SBC. **Workshop de Gerência e Operação de Redes e Serviços (WGRS)**. [S.l.], 2023. p. 71–84.

SONG, E. *et al.* Int-label: Lightweight in-band network-wide telemetry via interval-based distributed labelling. In: IEEE. **IEEE INFOCOM 2021-IEEE Conference on Computer Communications**. [S.l.], 2021. p. 1–10.

TAJIKI, M. M.; PETROUDI, S. G.; SALSANO, S.; UHLIG, S.; CASTRO, I. Optimal estimation of link delays based on end-to-end active measurements. **IEEE Transactions on Network and Service Management**, IEEE, v. 18, n. 4, p. 4730–4743, 2021.

TAN, L. *et al.* In-band network telemetry: A survey. **Computer Networks**, Elsevier, v. 186, p. 107763, 2021.

THONGLEK, K.; TAKAHASHI, K.; ICHIKAWA, K.; IIDA, H.; NAKASAN, C. Federated learning of neural network models with heterogeneous structures. In: **19th IEEE ICMLA**. [S.l.: s.n.], 2020. p. 735–740.

TOOTOONCHIAN, A.; GHOBADI, M.; GANJALI, Y. Opentm: traffic matrix estimator for openflow networks. In: SPRINGER. **International Conference on Passive and Active Network Measurement**. [S.l.], 2010. p. 201–210.

UHLIG, S.; QUOITIN, B.; LEPROPRE, J.; BALON, S. Providing public intradomain traffic matrices to the research community. **ACM SIGCOMM Computer Communication Review**, ACM New York, NY, USA, v. 36, n. 1, p. 83–86, 2006.

VARYANI, N.; ZHANG, Z.-L.; DAI, D. Qroute: An efficient quality of service (qos) routing scheme for software-defined overlay networks. **IEEE Access**, IEEE, v. 8, p. 104109–104126, 2020.

WANG, Z.; JIANG, D.; MUMTAZ, S. Network-wide data collection based on in-band network telemetry for digital twin networks. **IEEE Transactions on Mobile Computing**, IEEE, v. 24, n. 1, p. 86–101, 2024.

WU, Y.; ZHANG, K.; ZHANG, Y. Digital twin networks: A survey. **IEEE Internet of Things Journal**, IEEE, v. 8, n. 18, p. 13789–13804, 2021.

XU, S.; KODIALAM, M.; LAKSHMAN, T.; PANWAR, S. S. Learning based methods for traffic matrix estimation from link measurements. **IEEE Open Journal of the Communications Society**, IEEE, v. 2, p. 488–499, 2021.

YANG, C.; GIDOFALVI, G. Fast map matching, an algorithm integrating hidden markov model with precomputation. **International Journal of Geographical Information Science**, Taylor & Francis, v. 32, n. 3, p. 547–570, 2018.

YE, M.; ZHANG, J.; GUO, Z.; CHAO, H. J. Date: Disturbance-aware traffic engineering with reinforcement learning in software-defined networks. In: IEEE. **2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)**. [S.l.], 2021. p. 1–10.

ZHANG, P. *et al.* Privacy-preserving task offloading in vehicular edge computing using federated multi-agent reinforcement learning. **IEEE Transactions on Vehicular Technology**, IEEE, 2025.

ZHANG, Y. **Index of / yzhang/research/AbileneTM**. 2008. Disponível em: <https://www.cs.utexas.edu/~yzhang/research/AbileneTM/>.

ZHANG, Y.; ROUGHAN, M.; DUFFIELD, N.; GREENBERG, A. Fast accurate computation of large-scale ip traffic matrices from link loads. **ACM SIGMETRICS Performance Evaluation Review**, ACM New York, NY, USA, v. 31, n. 1, p. 206–217, 2003.

ZILLI, C.; SACCO, A.; ESPOSITO, F.; MARCHETTO, G. Clearnet: Enhancing transparency in opaque network models using explainable ai (xai) for efficient traffic engineering. **IEEE Transactions on Network and Service Management**, IEEE, 2025.