Universidade Federal da Bahia
Instituto de Matemática e Estatística

Programa de Pós-Graduação em Ciência da Computação

APRENDIZADO DE MÁQUINA PARA REDUÇÃO DO TRÁFEGO DE DADOS E DA LATÊNCIA NA NÉVOA DAS COISAS

Brenno de Mello Alencar

DISSERTAÇÃO DE MESTRADO

Salvador
26 de Setembro de 2020

BRENNO DE MELLO ALENCAR

APRENDIZADO DE MÁQUINA PARA REDUÇÃO DO TRÁFEGO DE DADOS E DA LATÊNCIA NA NÉVOA DAS COISAS

Esta Dissertação de Mestrado foi apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Cássio Vinicius Serafim Prazeres
Coorientador: Prof. Dr. Ricardo Araújo Rios

Salvador
26 de Setembro de 2020

Sistema de Bibliotecas - UFBA

Alencar, Brenno Mello.

Aprendizado de Máquina para Redução do Tráfego de Dados e da Latência na Névoa das Coisas / Brenno de Mello Alencar – Salvador, 2020.

91p.: il.

Orientador: Prof. Dr. Cássio Vinicius Serafim Prazeres.

Coorientador: Prof. Dr. Ricardo Araújo Rios.

Dissertação (Mestrado) – Universidade Federal da Bahia, Instituto de Matemática e Estatística, 2020.

1. Internet das Coisas. 2. Fluxo de dados. 3. Mineração de fluxo de dados. 4. Concept Drift; Wavelet. 5. Computação em Névoa. 6. Névoa das Coisas. 7. Redes Neurais.. I. Prazeres, Cássio Vinicius Serafim. II. Rios, Ricardo Araújo. III. Universidade Federal da Bahia. Instituto de Matemática e Estatística. IV. Título.

TERMO DE APROVAÇÃO

BRENNO DE MELLO ALENCAR

APRENDIZADO DE MÁQUINA PARA REDUÇÃO DO TRÁFEGO DE DADOS E DA LATÊNCIA NA NÉVOA DAS COISAS

Esta Dissertação de Mestrado foi julgada adequada à obtenção do título de Mestre em Ciência da Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia.

Salvador, 26 de Setembro de 2020

Prof. Dr. Cássio Vinicius Serafim Prazeres
Universidade Federal da Bahia

Prof. Dr. Manoel Gomes de Mendonça Neto
Universidade Federal da Bahia

Profa. Dra. Flavia Coimbra Delicato
Universidade Federal Fluminense

Aos meus pais e minha família, pelo amor e apoio incondicional em todas as fases da minha vida.

AGRADECIMENTOS

Agradeço inicialmente a Deus, que me concedeu saúde, força e coragem para conquistar mais esse objetivo em minha vida.

Agradeço aos meus pais João e Marlucia por todo apoio e amor dado. Agradeço a toda minha família, que é um alicerce indispensável em todos os momentos de minha vida.

Aos Professores Cássio Prazeres e Ricardo Rios, pela orientação e suporte ao longo deste trabalho, e por sempre mostrar o caminho certo a seguir durante as reuniões.

Agradeço aos membros da banca examinadora, Prof. Manoel Mendonça e Prof. Flávia Coimbra, pela disponibilidade de participar e contribuir com este trabalho.

Agradeço aos colegas do Grupo WISER e do PGCOMP, em especial, Jeferson, Ernando, Ramon, Leandro, Pedro, Cléber, Andressa, George, Nilson, Eudes, Leandro e Jurandir, pelo apoio e momentos de descontração. Agradeço, em especial, aos colegas Rui e João Paulo pelas contribuições técnicas no desenvolvimento deste trabalho.

A Capes pelo apoio financeiro.

Por fim, agradeço a todos que participaram direta ou indiretamente deste trabalho.

. . . visto que o mundo, perante nossos olhos, é como um livro formoso, em que todas as criaturas, grandes e pequenas, servem de letras que nos fazem contemplar 'os atributos invisíveis de Deus', isto é, 'o seu eterno poder e a sua divindade', como diz o apóstolo Paulo em Romanos 1.20: Todos estes atributos são suficientes para convencer os homens e torná-los indesculpáveis.

—IGREJA REFORMADA DA BÉLGICA (Confissão de Fé Belga, 1561)

RESUMO

A Internet of Things tem produzido infraestruturas e aplicações que geram grande volume de dados. Esses dados são geralmente fluxos de dados que têm a característica de serem contínuos e infinitos, e também apresentam a particularidade de modificar o seu comportamento ao longo do tempo. Devido a grande capacidade de armazenamento, processamento de dados e provisionamento de recursos, esses dados, em geral, são processados e analisados em ambientes de Cloud Computing. Embora a Cloud Computing forneça à infraestrutura IoT o tratamento adequado sob aspectos relacionados à escalabilidade e centralização dos recursos, a distância entre os dispositivos e a nuvem pode impor limitações para atingir baixa latência no tráfego de dados. Visando manter a escalabilidade, obter baixa latência e diminuir o tráfego de dados entre os dispositivos IoT e a Nuvem, a Fog Computing foi proposta. Apesar da Fog Computing estabelecer a disponibilização dos recursos na borda da rede, as tecnologias e técnicas utilizadas atualmente para processamento e análise de dados IoT podem não ser suficientes para suportar os fluxos contínuos e ilimitados de dados que as plataformas e aplicações IoT produzem. Além disso, as aplicações de fluxo de dados na Fog devem ser suportadas pelos dispositivos computacionalmente limitados empregados na Fog. Dessa forma, este trabalho apresenta uma abordagem para processamento e análise de fluxos de dados da Internet das Coisas em tempo real na Fog. Essa abordagem tem como objetivo principal reduzir a quantidade de dados transmitidos na infraestrutura de rede, o que permite, como consequência, realizar uma modelagem de dados online, detectando mudanças no comportamento do fluxo de dados e redução do uso da Internet. Além disso, a plataforma proposta não requer conexão constante com a Internet. Por fim, avaliamos a proposta a partir da perspectiva de desempenho num cenário de objetos inteligentes na borda da rede.

Palavras-chave: Internet das Coisas; Mineração de fluxo de dados; Concept Drift; Wavelet; Computação em Névoa; Névoa das Coisas; Redes Neurais Artificiais.

ABSTRACT

The Internet of Things (IoT) has produced infrastructures and applications that generate large amounts of data. These data are usually data streams, that have the characteristic of being continuous and infinite and also have the peculiarity of modifying their behavior over time. Due to the large capacity of storage, data processing and provisioning of resources, this data is generally processed and analyzed in cloud computing environments. Although Cloud Computing provides the IoT infrastructure with adequate scalability and resource centric features, the distance between devices and the cloud can impose limitations to achieve low latency in data traffic. In order to maintain scalability, achieve low latency and reduce data traffic between the IoT devices and the Cloud, the Fog Computing was proposed. Although the Fog Computing paradigm establishes resource availability at the edge of the network, the technologies and techniques currently used for IoT data processing and analysis may not be sufficient to support the continuous and unlimited data stream that IoT platforms produce. In this way, this work presents an approach for processing and analyzing data stream from the Internet of Things in real time in Fog. The main advantage of using our approach is the possibility of reducing the amount of data transmitted on the network infrastructure, which allows, as a consequence, to perform an online data modeling, by detecting changes in data behavior, and a reduction of the Internet usage. In addition, the proposed platform does not require a constant Internet connection. Finally, we evaluate the proposal from the perspective of performance in a scenario of intelligent objects at the edge of the network.

Keywords: Internet of Things; Data Stream Mining; Concept Drift; Wavelet; Fog Computing; Fog of Things; Artificial Neural Network.

SUMÁRIO

Capítulo 1—Introdução 1

1.1 Motivação e Problema . . . 3
1.2 Hipóteses de Pesquisa . . . 5
1.3 Objetivos . . . 5
1.4 Contribuições . . . 6
1.5 Metodologia . . . 6
1.6 Estrutura da Dissertação . . . 7

Capítulo 2—Revisão Sistemática da Literatura 9

2.1 Questões de Pesquisa . . . 10
2.2 Estratégia de Pesquisa . . . 11
2.3 Processo de Seleção dos Trabalhos . . . 12
2.4 Avaliação da Qualidade dos Trabalhos . . . 13
2.5 Análise dos Resultados . . . 13
2.6 Trabalhos Relacionados . . . 15
2.7 Considerações Finais . . . 18

Capítulo 3—Fundamentação Teórica 19

3.1 Internet das Coisas . . . 19
3.2 Fog Computing . . . 23
3.2.1 Fog of Things e SOFT-IoT . . . 25
3.3 Big Data Stream e Mineração de fluxo de dados . . . 29
3.3.1 Transformada Wavelet Discreta . . . 31
3.3.2 Concept Drift para Internet das Coisas . . . 32
3.3.2.1 Cumulative Sum . . . 33
3.3.2.2 Page-Hinkley . . . 33
3.3.2.3 Exponentially Weighted Moving Average . . . 34
3.3.3 Redes Neurais Artificiais . . . 35
3.3.4 Rede Neural Recorrente . . . 36
3.3.5 Redes Neurais Long Short-Term Memory (LSTM) . . . 37
3.4 Considerações Finais . . . 39

Capítulo 4—Processamento e Análise de Fluxo de Dados na Computação em Névoa 41

4.1 Arquitetura de Fog para processamento e análise de fluxo de dados IoT . 41
4.2 Implementação da Arquitetura na Plataforma SOFT-IoT 44
4.3 Abordagem de Análise de Fluxo de dados na Fog Computing 47
4.4 Considerações Finais 52

Capítulo 5—Planejamento dos Experimentos e Avaliação dos Resultados 53

5.1 Experimentos Cenário Real da IoT 53
5.2 Experimentos Cenário Emulado da IoT 62
5.2.1 Emulador de Redes Mininet 64
5.2.2 Modelagem da Emulação no Mininet 65
5.2.3 Configurações do Ambiente e Experimentos 71
5.2.4 Avaliação dos Resultados 71
5.3 Considerações Finais 76

Capítulo 6—Conclusão 79

6.1 Contribuições do Trabalho Realizado 79
6.2 Limitações 80
6.3 Trabalhos Futuros 80

Apêndice A—Resultados da Revisão Sistemática 87

LISTA DE FIGURAS

1.1 Abordagem proposta. 7
1.2 Metodologia da pesquisa desenvolvida. 8

3.1 Visões da Internet das Coisas 20
3.2 Tecnologias Habilitadoras da IoT 21
3.3 Funcionamento do MQTT 22
3.4 Domínios de Aplicação IoT 23
3.5 Internet das Coisas e *Fog Computing* 24
3.6 Níveis de análise de dados na computação em névoa 26
3.7 Visão Geral SOFT-IoT 27
3.8 Gateway plataforma SOFT-IoT 28
3.9 $x(t)$ é um sinal TR e $V1(t) - V9(t)$ representa diferentes resoluções da wavelet. 32
3.10 Exemplo de *Concept Drift* 35
3.11 38
3.12 39

4.1 Fluxo de dados arquitetura centralizada. 42
4.2 Arquitetura da proposta. 43
4.3 Visão geral da arquitetura da proposta. 44
4.4 Análise de fluxo de dados IoT. 45
4.5 Arquitetura FoT-StreamGateway para Fluxo de Dados. 46
4.6 Arquitetura FoT-StreamServer para Fluxo de Dados. 46

5.1 Modelagem do Experimento. 54
5.2 Fluxo de dados de temperatura na Fog e Cloud analisados pelo CUSUM. 55
5.3 Fluxo de dados Temperatura na Fog e Cloud. Experimento realizado utilizando Haar and CUSUM. 56
5.4 Fluxo de dados de umidade na Fog e Cloud analisados pelo CUSUM. . . 56
5.5 Fluxo de dados Umidade na Fog e Cloud. Experimento realizado utilizando Haar and CUSUM. 57
5.6 Fluxo de dados de poeira na Fog e Cloud analisados pelo CUSUM. . . . 57
5.7 Fluxo de dados Poeira na Fog e Cloud. Experimento realizado utilizando Haar and CUSUM. 57
5.8 Fluxo de dados de luminosidade na Fog e Cloud analisados pelo CUSUM. 58
5.9 Fluxo de dados Luminosidade na Fog e Cloud. Experimento realizado utilizando Haar and CUSUM. 58

5.10 Fluxo de dados de som na Fog e Cloud analisados pelo CUSUM. 58
5.11 Fluxo de dados Som na Fog e Cloud. Experimento realizado utilizando Haar and CUSUM. 59
5.12 Dados trafegados entre a Fog e a Cloud. Experimento realizado utilizando somente CUSUM. 59
5.13 Dados trafegados entre a Fog e a Cloud. Experimento realizado utilizando Haar e CUSUM. 60
5.14 Throughput do gateway com 5 sensores. 60
5.15 Throughput do gateway com 10 sensores. 61
5.16 Tempo médio de processamento calculado para Haar e CUSUM no gateway. 61
5.17 Tempo médio de processamento para janela de dados processados com Haar e CUSUM no gateway. 62
5.18 Configuração 1 da Emulação. 63
5.19 Configuração 2 da Emulação. 63
5.20 Rede Mininet. 64
5.21 Visão geral do experimento no Mininet. 66
5.22 Comunicação entre Sensor e Gateway emulados no Mininet. 66
5.23 Comunicação entre Servidor e Gateway emulados no Mininet. 67
5.24 Comparação temperatura real e predita Execução 1. 74
5.25 Comparação temperatura real e predita Execução 2. 74
5.26 Quantidade de dados processados nas Execuções 5, 6. 75
5.27 Quantidade de dados processados nas Execuções 1, 2, 3, 4. 76
5.28 Tráfego de dados nas Execuções 5, 6. 76
5.29 Tráfego de dados nas Execuções 1, 2, 3, 4. 77

LISTA DE TABELAS

2.1 Número de artigos retornados nas buscas. 14
2.2 Número de artigos publicados por ano. 14
2.3 Tipo de publicação dos artigos. 14
2.4 Comparação dos trabalhos relacionados. 17

5.1 Configurações utilizadas nos experimentos. 71
5.2 Resultados médios das métricas dos Execução 1, 3, 5. 72
5.3 Resultados médios das métricas dos Execução 2, 4, 6. 73
5.4 Resultados médios das métricas dos Experimentos 2, 4, 6. 73

A.1 Artigos selecionados na revisão sistemática. 88
A.1 Artigos selecionados na revisão sistemática. 89
A.1 Artigos selecionados na revisão sistemática. 90
A.1 Artigos selecionados na revisão sistemática. 91

LISTA DE SIGLAS

IoT	*Internet of Things*
IP	*Internet Protocol*
RFID	*Radio-Frequency Identification*
FoT	*Fog of Things*
MQTT	*Message Queue Telemetry Transport*
COAP	*Constrained Application Protocol*
HTTP	*Hypertext Transfer Protocol*
QoS	*Quality of Service*
OSGi	*Open Services Gateway Initiative*
LSTM	*Long short-term memory*
W3C	*World Wide Web Consortium*
NFC	*Near-field communication*
IERC	*IoT European Research Cluster*
IPSO	*IP for SmartObjects*
M2M	*Machine to Machine*
LTE	*Long-Term Evolution*
RESTful	*Representational state transfer*

Capítulo 1

INTRODUÇÃO

O principal conceito proposto pela Internet das Coisas (IoT - *Internet of Things*) é a conexão de dispositivos com a Internet com capacidade de sensoriamento, atuação em diferentes ambientes e coordenação na tomada de decisões (AL-FUQAHA et al., 2015). A IoT tem ganhado destaque no cenário tecnológico atual devido o seu alto potencial de impactar os diversos segmentos da sociedade, permitindo gestão inteligente de ambientes de energia e água (BECKEL et al., 2012) (PRANATA; LEE; KIM, 2017), suporte para vida de idosos (DOHR et al., 2010), monitoramento de segurança pública (GIMENEZ et al., 2012) e gerenciamento de cadeias de suprimentos (LOU et al., 2011).

Uma das questões mais desafiadoras da IoT é a coordenação do grande número de dispositivos que, de acordo com o relatório sobre mobilidade da Ericsson (HEUVELDOP et al., 2017), deve chegar a cerca de 29 bilhões de dispositivos IoT conectados até 2022. Como consequência, um grande volume de dados deve ser produzido pelos dispositivos como, por exemplo, sensores, atuadores, dispositivos inteligentes e outros objetos conectados à Internet (SOLDATOS, 2016). Para exemplificar a quantidade de dados produzidos, Marjani et al. (2017) apresentam resultados de coleta de dados com sensores médicos que geram 1000 eventos por segundo, produzindo cerca de 55 milhões de novas instancias de dados por dia. Similarmente, Biswas, Dupont e Pham (2016) expõem que a General Electric reúne diariamente 50 milhões de dados de 10 milhões de sensores. Esses dois exemplos demonstram o desafio de viabilizar o processamento em tempo real dos dados produzidos pela IoT.

Os dados de várias fontes e domínios produzidos pela IoT são, em geral, fluxos de dados, como, por exemplo, dados numéricos de diferentes sensores. De acordo com Gama (2010) os fluxos de dados têm as seguintes características: i) são disponibilizados de maneira ilimitada, continuamente e ao longo do tempo e às vezes em alta velocidade; ii) os dados podem mudar o seu comportamento com o passar do tempo, em vez de ficarem estacionários, não podendo ser considerados independentes e identicamente distribuídos. Baseado nessas características, os dados produzidos na IoT também são conceituados como *Big Data* em tempo real.

Entretanto, a utilização dos métodos tradicionais de *Big Data* pode ser proibitiva nesse contexto devido à natureza de novos dados em volume crescente, dificultando a análise em tempo real (PECORI, 2018). Por outro lado, os dados produzidos pela IoT apresentam diferentes aspectos para os V's (Volume, Velocidade, Variedade, Veracidade) explorados pelo *Big Data* tradicional. O volume de dados produzidos pela IoT, na maioria dos casos, excede os recursos de armazenamento e processamento dos bancos de dados tradicionais. A velocidade que os dados da IoT são produzidos é muito alta, reduzindo consideravelmente o intervalo de tempo de coleta geralmente adotado em modelos discretos. A variedade está ligada à diversidade de dispositivos e fontes de dados encontrados na IoT. Por fim, a veracidade pode ser compreendida pelo problema de os dados produzidos pelos sistemas da IoT conterem ruídos (SOLDATOS, 2016).

Para preencher essa lacuna de trabalhar com *Big Data* para fluxo de dados, um novo paradigma, denominado como *Big Data Streams* ou *Big Stream* (BIFET, 2013) (BELLI et al., 2016) (MOHAMMADI et al., 2018), está surgindo na comunidade de pesquisa mais envolvida no tráfego de dados da IoT. Na verdade, esse paradigma se concentra na combinação dos métodos tradicionais de *Big Data* com os requisitos em tempo real, como, por exemplo, a taxa de geração de dados, a gestão de fluxos de dados, os requisitos de tempo real e a baixa latência que alguns ambientes IoT necessitam. As principais diferenças entre o *Big Data* tradicional e o *Big Data Streams* estão relacionadas ao fato de que, no *Big Data Streams*, os dados são produzidas em um ritmo mais rápido, podem ser gerados por várias fontes heterogêneas e precisam ser analisadas em tempo real (PECORI, 2018).

O *Big Data Streams* da IoT pode se beneficiar dos métodos desenvolvidos para mineração de fluxo de dados. Na mineração de fluxo de dados é possível analisar dados em tempo real com a utilização de uma quantidade limitada de processamento e memória. Considerando essas características da mineração de fluxo de dados, o trabalho de Bifet (2013) apresenta quatro características importantes que devem ser consideradas antes da mineração de fluxos de dados no *Big Data Streams*: i) primeira característica é processar um exemplo de cada vez e inspecionar também apenas uma vez os dados na ordem de chegada, sendo que pode ser executada usando a memória de curto prazo, como uma fila, para armazenar um subconjunto dos dados; ii) a segunda é utilizar uma quantidade limitada de memória. iii) a terceira é a complexidade temporal e espacial, que deve ser linear ou sublinear para operar dentro de um tempo de execução limitado; e finalmente, iv) a partir de novos dados os algoritmos devem responder a qualquer momento. Por essas razões, o processo de mineração em fluxo de dados aproveita áreas de pesquisa complementares, como, por exemplo, redução de dados e detecção de mudança de conceito (NGUYEN; WOON; NG, 2015).

Cabe destacar que a maior parte do poder computacional da IoT pode ser empregado na borda da rede, conceito denominado como Computação em Névoa (do inglês *Fog Computing - Fog*), permitindo descentralizar tarefas de processamento e transferir apenas dados filtrados de dispositivos localizados na borda da rede para a nuvem. De acordo com Bonomi et al. (2014), a *Fog Computing* estende a Computação em Núvem (do inglês *Cloud Computing - Cloud*) para próximo da borda da rede, tendo como objetivo principal atender as aplicações que requerem baixa latência, aplicações geo-distribuídas, sistemas

que precisam de mobilidade e produzem grandes fluxo de dados e aplicações em tempo real. Por causa dessas características, a *Fog Computing* é regularmente utilizada no desenvolvimento de aplicações para IoT.

Apesar da *Fog Computing* permitir a utilização da borda da rede para tarefas de processamento de dados, a enorme quantidade de dados sendo produzidos em tempo real é um problema atual desses ambientes (ZOU et al., 2019). O volume de dispositivos de IoT na *Fog* produzindo *Big Data* permite, por exemplo, gerar decisões inteligentes, análises e outros valores agregados como retorno. Por outro lado, a grande quantidade de dados pode sobrecarregar os sistemas de armazenamento e a largura de banda da rede sem fio (ZOU et al., 2019). O relatório global da Cisco (Cisco Global Cloud Index, 2016–2021) apresenta uma perspectiva de 850 ZB que serão gerados até 2021 pelos ambientes de *Fog Computing*. Esses dados serão de natureza efêmera, não sendo necessários para serem armazenados, mas é estimado que aproximadamente 10% sejam úteis. De acordo com relatório, haverá 10 vezes mais dados úteis sendo criados (85 ZB, 10% do total de 850) do que serão utilizados ou armazenados (7,2 ZB) em 2021. Os dados úteis também devem exceder o tráfego suportado pelos *data centers* (21 ZB por ano). Esse problema apresentado é uma limitação para implantação de sistemas IoT que produzem fluxo de dados na *Fog Computing*. Além disso, essa falta de processamento para identificar dados úteis limita a precisão de técnicas de análise de dados aplicadas nas camadas da *Fog*.

1.1 MOTIVAÇÃO E PROBLEMA

Diante disso, combinando as análises propostas para o *Big Data Stream* com a *Fog Computing* é possível processar fluxo de dados na borda da rede, podendo melhorar as aplicações que necessitam de resposta em tempo real e devem tomar decisões em curto espaço de tempo. Outro fator para análises em *Big Data Stream* na *Fog* é que as informações que podem ser extraídas dos dados da IoT são temporais e espaciais (YANG, 2017), ou seja, estão diretamente ligadas a um determinado contexto, possivelmente próximo onde os sensores estão implantados. Por isso, é mais indicado que as informações sejam processadas em curto espaço de tempo e dentro do seu contexto. Contudo, como mencionado anteriormente, as análises de dados tradicionais não são as mais adequadas para os requisitos que a Internet das Coisas impõem (SOLDATOS, 2016). Sendo assim, a análise de fluxo de dados em tempo real na borda da rede é uma tarefa desafiadora, levando a necessidade de aproveitar todo poder dos recursos computacionais disponíveis distribuídos localmente.

Recentemente, pesquisas em IoT têm trabalhado no desafio de implementar o *Big Data* para IoT. Porém, na maioria dos casos, para processar e analisar esses dados, são utilizadas plataformas de análise de dados centralizadas na nuvem, como, por exemplo, Microsoft Azure IoT Suite e Amazon Web Service (PATEL; ALI; SHETH, 2017). Nas arquiteturas centralizadas, os dados dos sensores são enviados pela rede para as plataformas na nuvem, onde todo o processamento ocorre e as tomadas de decisões são realizadas.

Na literatura, alguns trabalhos abordam o processamento e análise de fluxos de dados da IoT na *Fog Computing* ou na borda da rede. O trabalho citado anteriormente de Yang (2017) aborda essa área de pesquisa, apresentando direcionamentos sobre tecnologias que

dão suporte para o processamento de fluxo de dados na borda da rede. Outro trabalho é o de Harth, Delakouridis e Anagnostopoulos (2018), que trata de análise preditiva dos dados na rede de sensores e agregação de dados na borda da rede. Um outro trabalho é de o Patel, Ali e Sheth (2017), que define uma arquitetura para manter a infraestrutura de análise de dados IoT na *Fog Computing*.

Apesar da existência desses trabalhos desenvolvidos para fluxo de dados na IoT, o relatório de Manyika et al. (2015) apresenta que, de 30 mil sensores numa plataforma de petróleo, menos de 1% desses dados são examinados e que poderiam ser utilizados para otimização e predição na plataforma. Isso pode ser considerado um entrave para maximizar o valor comercial da IoT. Atualmente, como apresentado por Yang (2017), a maioria das aplicações IoT adotam a arquitetura cliente-servidor, com dispositivos inteligentes e a nuvem como *back-end*. Porém, Yang (2017) também afirma que esse estilo de aplicações tem dois problemas. Primeiro, a latência entre os dispositivos não deve atender aos requisitos das aplicações de fluxo de dados. O segundo problema é que a grande quantidade de dados do fluxos de dados IoT pode não ser suportada pela infraestrutura de rede atual.

Harth, Delakouridis e Anagnostopoulos (2018) explicam que o envio dos dados de bilhões de dispositivos para a nuvem pode sobrecarregar a infraestrutura existente. Sendo assim, os pesquisadores afirmam que a utilização de análise de dados na borda da rede permite: tempos de resposta mais rápidos e tráfego reduzido. Os autores concluem que transferir grandes volumes de dados para nuvem é, em alguns casos, inviável. Isso se deve as limitações de energia do equipamento de rede (provavelmente tecnologia de rádio) e da largura de banda. A partir do que foi apresentado, a seguinte questão de pesquisa é formulada: como processar e analisar fluxos de dados em tempo real na *Fog Computing* com baixo atraso de processamento e diminuindo a quantidade de dados trafegados na rede sem necessidade de conectividade constante com Internet nos ambientes de *Fog Computing* para Internet das Coisas?

Diante desse problema, neste trabalho de pesquisa é apresentado uma abordagem e uma arquitetura para processamento e análise de fluxo de dados na *Fog Computing*. A abordagem proposta utiliza técnicas de mineração de fluxo de dados, como, por exemplo, técnicas de detecção de mudanças de comportamento (*Concept Drift*) combinada com técnica de processamento de sinais (transformada de Wavelets), permitindo analisar os dados em tempo real na *Fog Computing*. A transformada de Wavelet permite decompor os fluxos de dados em um conjunto de componentes capazes de manter o comportamento geral dos dados sem usar todos os valores de observação. Complementarmente, a detecção mudança de conceito foi adotada para reduzir a transmissão de dados na rede, o que só acontece quando o comportamento geral do fluxo de dados muda com o tempo. Além disso, com a abordagem proposta é possível melhorar predições na aplicação de algoritmo de aprendizagem de máquina nos fluxo de dados na *Fog Computing*. Para suportar a abordagem descrita anteriormente foi proposta um arquitetura de fluxo de dados na Névoa das Coisas (do inglês *Fog of Things* - FoT), sendo essa arquitetura uma extensão da SOFT-IoT proposta por Prazeres e Serrano (2016). A diferença principal desta pesquisa de mestrado em relação aos trabalhos da literatura é a abordagem utilizada para tratar o fluxo de dados da IoT na *Fog*.

1.2 HIPÓTESES DE PESQUISA

Com base nas observações citadas anteriormente, a seguinte hipótese foi formulada: o desenvolvimento de uma abordagem utilizando técnicas de mineração de fluxos de dados para *Fog Computing* pode permitir que dados sejam processados, filtrados e analisados à medida que são produzidos. Com essas técnicas, identificando apenas mudanças na distribuição e realizando resumo do fluxo de dados, é possível diminuir a quantidade de dados enviados para a nuvem mantendo a representatividade dos dados dos sensores. Além disso, é possível obter predições nas camadas da *Fog Computing* aplicando algoritmo de Redes Neurais Artificiais (RNA) utilizando uma menor quantidade de dados no treinamento. Diante disso, através desse proposta, os seguintes aspectos podem ser otimizamos: a) reduzir a quantidade de dados que devem ser analisados; b) reduzir a utilização da rede, pois uma menor quantidade de dados é trafegado; c) obter respostas mais rápidas para alterações no fluxo de dados; d) necessidade de conectividade com a Internet em todos os momentos; e) aprimoramento da acurácia de predições na aplicação de algoritmo de redes neurais na *Fog Computing*.

1.3 OBJETIVOS

Este trabalho tem o objetivo principal avaliar como a definição e o desenvolvimento de uma abordagem para processamento e mineração de fluxos de dados IoT na *Fog Computing* pode diminuir a quantidade de dados trafegados na rede. Além disso, mostra-se que a abordagem proposta reduz a necessidade de conectividade constante com Internet por parte dos ambientes IoT, permite obter baixo atraso de processamento e análise no fluxo de dados na *Fog* e realiza predição com algoritmos de redes neurais na *Fog Computing*. O objetivo principal é subdividido nos seguintes objetivos específicos:

1. Estender uma plataforma de Internet das Coisas para suportar processamento e análise de fluxo de dados, servindo de infraestrutura para utilização das técnicas de mineração de fluxo de dados.

2. Identificar e adaptar técnicas de mineração de fluxos de dados que podem ser utilizadas na *Fog Computing* para aplicações da Internet das Coisas.

3. Definir e implementar uma abordagem de análise de fluxo de dados IoT na *Fog Computing*, isso utilizando as técnicas definidas no objetivo específico 2.

4. Utilizar um algoritmo de predição na *Fog Computing* para verificar a viabilidade de realizar predições no fluxo de dados através da abordagem proposta.

5. Modelar e implementar ambiente de simulação com os principais componentes da proposta.

6. Realizar experimentos como prova de conceito em uma rede de sensores IoT para monitoramento de ambientes.

1.4 CONTRIBUIÇÕES

As contribuições resultantes deste trabalho são:

- Abordagem com técnicas de mineração de fluxo de dados para reduzir tráfego de dados e latência na rede em ambientes de *Fog* e IoT;
- Modelagem e desenvolvimento de uma solução de *Fog Computing* para fluxo de dados aplicados no paradigma *Fog of Things* que poderá ser utilizada na realização de novos estudos e experimentos em abordagens do mesmo contexto (FoT);
- Integração entre os paradigmas de *Fog of Things* e mineração em *Big Data Stream*;
- Manutenção das métricas de acurácia para predição feita por aplicação de algoritmo de redes neurais em arquitetura de *Fog* através de técnicas de mineração de fluxo de dados com menos dados utilizados.

1.5 METODOLOGIA

Com a finalidade de atender aos objetivos propostos neste trabalho, as seguintes atividades foram executadas em sequência lógica: (1) execução de uma revisão sistemática da literatura para encontrar os estudos que abordam a análise de fluxo de dados na *Fog Computing*; (2) definição, modelagem e implementação da solução inicial da proposta por meio dos estudos realizados sobre os trabalhos relacionados; (3) implementação e experimentação da solução em ambiente de sensores físicos da IoT; (4) realização de planejamento e execução de experimentos iniciais; (5) com base nos resultados da etapa anterior, escrita e submissão de artigo; (6) baseado nos resultados obtidos nas etapas anteriores, realização de ajustes na modelagem e realizar implementação final; (7) implementação de uma simulação da modelagem final da proposta; (8) realização de experimentos e avaliação dos resultados da simulação da solução final; (9) escrita da dissertação de mestrado; (10) escrita e submissão de artigo com resultados finais.

A Figura 1.2 mostra o ciclo das atividades do projeto. Como apresentado na imagem, a maioria das atividades segue-se em encadeamento sequencial, assim, a conclusão de uma atividade torna-se ponto de partida para outra. A atividade que não tem encadeamento sequencial é a de revisão da literatura, ela é executada durante todo desenvolvimento das atividades para identificar novos trabalho na área, isso seguindo o protocolo definido na revisão sistemática. Nas atividades (3) e (7) o ciclo formado com a seta pontilhada denota eventuais modificações no modelo a partir de observações durante a análise prévia dos resultados.

Na fase 2 e 6 a modelagem da solução foi desenvolvida de acordo com Figura 1.1. Na abordadem proposta, o processamento e análise de fluxos de dados da IoT é iniciado a partir da *Fog Computing*. Em nossa pesquisa, adaptamos em ambientes *Fog Computing* duas técnicas bastante conhecidas e amplamente utilizadas na área de Processamento de Sinais: transformada Wavelet e detecção de mudança de conceito. A transformada Wavelet foi responsável por decompor os fluxos de dados em um conjunto de componentes capazes de manter o comportamento geral dos dados sem usar todos os valores de

observação. Complementarmente, foi adotada a detecção de mudança de conceito para reduzir a transmissão de dados na rede, que só ocorre quando o comportamento geral muda com o tempo. A partir dessas análises é treinado um modelo de predição de aprendizado de máquina em fluxo de dados. A abordagem pode ser adaptada para acontecer somente na *Fog* ou na *Cloud*, assim como os componentes podem ser divididos entre a *Fog* e a *Cloud*.

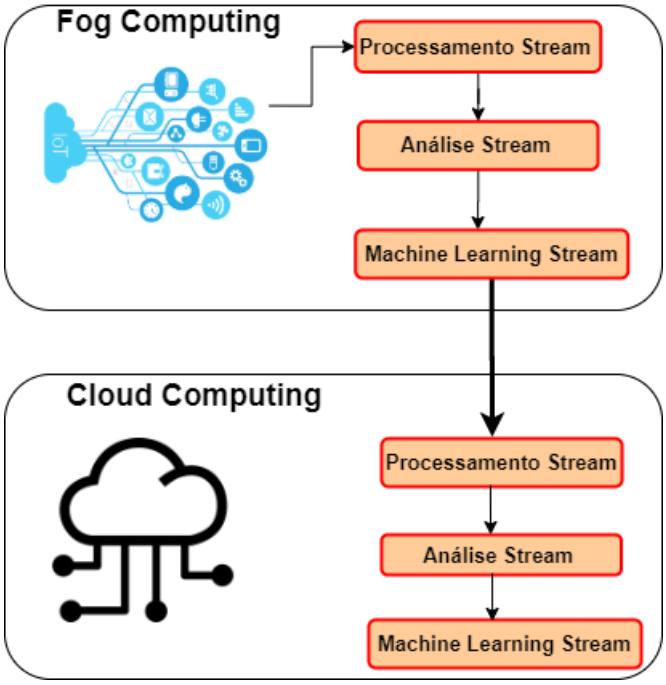

Figura 1.1 Abordagem proposta.

1.6 ESTRUTURA DA DISSERTAÇÃO

O Capítulo 2 descreve a revisão sistemática da literatura e os trabalhos relacionados encontrados. O capítulo 3 apresenta a fundamentação teórica sobre Internet das Coisas, *Fog Computing*, *Fog of Things*, *Data Stream Mining* e Redes Neurais Artificiais. No Capítulo 4, a abordagem proposta é apresentada a partir da sua arquitetura e algoritmos implementados. O planejamento de experimentos, apresentação e análise dos resultados são abordados no Capítulo 5. Por fim, no Capítulo 6 realizam-se as considerações finais e direcionamentos para trabalhos futuros.

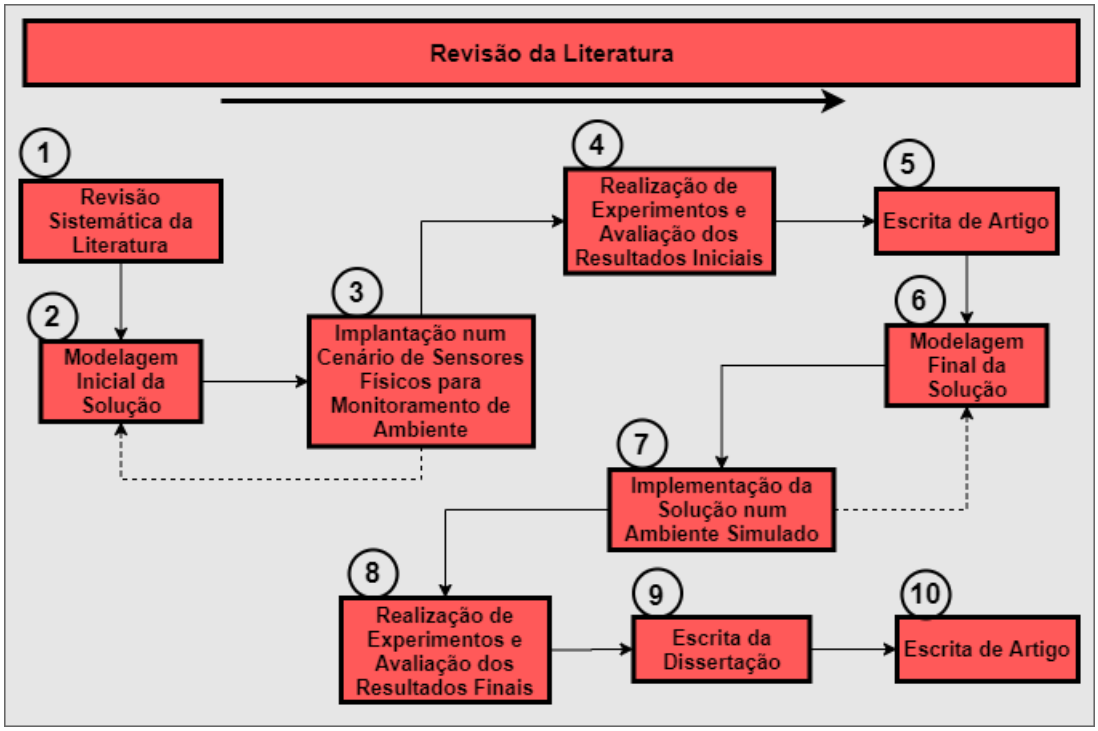

Figura 1.2 Metodologia da pesquisa desenvolvida.

Capítulo 2

REVISÃO SISTEMÁTICA DA LITERATURA

Este capítulo apresenta uma revisão sistemática da literatura relacionada ao projeto desenvolvido neste trabalho. A revisão identifica os principais trabalhos da área, assim como as técnicas de análise de dados, os cenários que foram utilizados e como foram avaliados os trabalhos.

Uma revisão sistemática da literatura é uma forma de identificar pesquisas relevantes para uma determinada questão de pesquisa. Além disso, permite avaliar e interpretar as pesquisas relevantes para uma determinada questão de pesquisa, tópico ou fenômeno de interesse. As principais razões para desenvolvimento de uma revisão sistemática são: sumarizar as evidências existentes sobre uma tecnologia ou tratamento para doenças; identificar lacunas na pesquisa atual para propor novas áreas de pesquisas; fornecer o alicerce para o posicionamento adequado de novas atividades de pesquisa (KITCHENHAM; CHARTERS, 2007). Sendo assim, esta revisão sistemática busca encontrar na literatura o estado da arte da IoT *Analytics* na *Fog Computing*, conceito que pode ser descrito como *Fog Analytics*, com maior atenção em análises de fluxo de dados distribuídas nos níveis da *Fog Computing* para Internet de Coisas. Além disso, a revisão busca encontrar problemas ainda não resolvidos.

As pesquisas na área de Internet das Coisas têm obtido progresso significativo no desenvolvimento de arquiteturas, aplicações e plataformas. Essas pesquisas propuseram soluções para os desafios clássicos que a IoT impõe, por exemplo, *middlewares* para tratar a heterogeneidade de *hardware* e *software* das tecnologias utilizadas na Internet das Coisas, plataformas de computação em nuvem para processamento e armazenamento dos dados produzidos pelos dispositivos da IoT. Entretanto, tratar a grande quantidade de dados da Internet das Coisas dos sistemas IoT na *Fog Computing* é um dos problemas da área ainda não resolvido com pesquisas em andamento.

Uma tendência encontrada nos sistemas para IoT atuais é a utilização da *Fog Computing*, que ajuda resolver problemas e aprimorar alguns aspectos das aplicações centralizadas na nuvem, como, por exemplo: conseguir baixa latência; melhorar qualidade de serviço; possibilitar que os dispositivos possam operar na *Fog Computing* mesmo sem

conectividade com a Internet; processamento de dados e aplicativos ocorrem dentro dos limites da rede local. Devido a essas características da *Fog*, uma nova área de pesquisa vem surgindo, seu maior desafio é processar e analisar a grande quantidade de dados gerados pelos ambientes da IoT em tempo real na *Fog Computing*, com objetivo de tirar proveito dos benefícios de uma arquitetura em *Fog*.

A revisão sistemática apresentada neste capítulo fornece evidências da relevância dos estudos de análise de fluxo de dados na *Fog Compunting*, também apresenta os métodos mais aplicados nesta área. Além disso, exibe a relação entre a quantidade de artigos publicados e o tipo de publicação e a quantidade de artigos da área por ano. A revisão sistemática executada nesse trabalho segue as diretrizes apresentas por Kitchenham e Charters (2007).

Na primeira fase da revisão sistemática foi definido o escopo da pesquisa. Nessa fase, foram selecionados os critérios para classificação se as pesquisas estão ou não relacionadas com o tema pesquisado. Para isso, foram definidos também nessa etapa o objetivo da pesquisa, a questão principal, as questões secundárias, os repositórios de pesquisas, a *string* de busca com as palavras-chave, os critérios de inclusão e exclusão, por fim, a linguagem padrão para os artigos.

Na segunda fase da revisão sistemática, o protocolo de pesquisa foi executado nos repositórios definidos e o resultado das buscas é apresentado na Seção 2.5. Por fim, na Seção 2.6 são discutidos os trabalhos relacionados a esta proposta.

2.1 QUESTÕES DE PESQUISA

A definição das questões de pesquisa é uma parte fundamental de uma revisão sistemática pelo fato de conduzir toda a metodologia da revisão (KITCHENHAM; CHARTERS, 2007). O objetivo principal da revisão sistemática realizada é encontrar trabalhos que abordaram técnicas de análise de fluxo de dados aplicadas na *Fog Computing*, tendo como objetivo otimizar os aspectos relacionados a aplicação da *Fog Computing* na Internet das Coisas. Diante disso, a questão principal de pesquisa é a seguinte:

> *Quais são as técnicas de análise de fluxo de dados empregadas nos níveis da Fog Computing aplicadas na Internet das Coisas?*

Além da questão principal definida, foram definidas questões secundárias que servem para ajudar a validar a pesquisa desenvolvida. As questões secundárias têm o objetivo de identificar aplicações práticas do tema estudado, também entender como são avaliados os trabalhos na área, por fim, encontrar tendências nos trabalhos. As questões secundárias são:

- QS.1 Quais são os cenários reais de aplicações da Internet das Coisas que podem tirar proveito das técnicas de análise de fluxo de dados para *Fog Computing*?

- QS.2 Como as técnicas de análise de fluxo de dados na *Fog Computing* para Internet das Coisas são avaliadas?

- QS.3 Por que são aplicadas técnicas de análise de fluxo de dados na *Fog Computing* para Internet das Coisas?

- QS.4 Quais são a limitações da aplicação de técnicas de análise de fluxo de dados na *Fog Computing* para Internet das Coisas?

2.2 ESTRATÉGIA DE PESQUISA

A revisão sistemática da literatura tem como ponto de partida a busca por todos os estudos primários relacionados às questões de pesquisa em foco. Sendo assim, a primeira etapa para encontrar estudos relevantes é identificar e selecionar os termos de pesquisas mais relacionados com o problema tratado. Para isso, a abordagem definida por Kitchenham e Charters (2007) foi seguida. Para cada perspectiva, os termos de pesquisa relevantes para esta revisão sistemática foram identificados assim:

- **População:** Internet of Things, Fog Computing, Edge Computing

- **Intervenção:** Data Stream, Big Data Stream, Data Stream Mining

- **Resultados:** Data Stream Mining Techniques

Depois de definidos os termos de pesquisa, também chamados de palavras-chave, a *string* de busca foi desenvolvida. O termo da perspectiva resultados da abordagem de Kitchenham e Charters (2007) não foi incluído nas buscas, pois em pesquisas preliminares a sua inclusão reduzia notavelmente o volume de trabalhos retornados na busca, aumentando assim o risco de faltar algum estudo relevante. Apesar dos paradigmas *Edge Computing* e *Fog Computing* apresentarem algumas diferenças, o termo *Edge Computing* foi utilizado entendendo que a *Edge* pode permitir, em alguns casos, a análise de dados na borda da rede como a *Fog Computing*, esta decisão foi tomada com base em pesquisas e leituras anteriores a realização desta revisão. A *string* de busca foi definida da seguinte forma:

(“Internet of Things”) AND (“Fog Computing”OR “Edge Computing”) AND (“Data Stream”OR “Big Data Stream”OR “Data Stream Mining”)

Após a definição da *string* de busca, a próxima etapa foi a escolha dos repositórios de pesquisa. Os repositórios utilizados foram os mais empregados pelas pesquisas científicas e sugeridos por Kitchenham e Charters (2007). A *string* de busca teve de ser ajustada de acordo com os requisitos de cada biblioteca digital. A pesquisa foi realizada a partir de janeiro de 2018 e teve a limitação para selecionar estudos publicados apenas com a linguagem em inglês. Os repositórios digitais selecionados foram:

- Scopus [1]

[1] https://www.scopus.com/home.uri

- IEEE Xplore [2]
- ACM Digital library [3]
- Google Academic [4]

2.3 PROCESSO DE SELEÇÃO DOS TRABALHOS

Inicialmente foram realizadas as buscas através dos motores de busca das bibliotecas digitais. As pesquisas foram realizadas nos títulos, *abstract* e palavras-chave dos artigos, foram retornados 738 trabalhos. Depois disso, os artigos duplicados foram identificados e removidos. Então, para realizar uma seleção mais elaborada dos possíveis trabalhos relevantes, foram definidos critérios de inclusão e exclusão dos trabalhos retornados. Os critérios de exclusão foram os seguintes:

- Trabalho não abordou análise de fluxo de dados na *Fog Computing* para IoT;
- A avaliação da proposta do trabalho não é adequada;
- Artigo não está em inglês.

Por outro lado, um artigo deve ser selecionado caso atenda algum dos seguintes critérios de inclusão:

- Trabalho estudou análise de fluxo de dados na *Fog Computing* para IoT;
- Trabalho estudou análise de fluxo de dados na *Edge Computing* para IoT;
- Trabalho apresenta modelo de análise de fluxo de dados no nível tecnológico na *Fog* para IoT;
- É um estudo primário.

Em síntese, o processo de seleção dos trabalhos ocorreu com o primeiro passo de execução da *string* de busca nos repositórios selecionados. O segundo passo foi a remoção dos trabalhos duplicados. Após isso, o título e o *abstract* de cada artigo foram analisados de acordo com os critérios de inclusão e exclusão. Por fim, os trabalhos relevantes foram completamente lidos.

[2] http://ieeexplore.ieee.org
[3] https://dl.acm.org/
[4] https://scholar.google.com.br/

2.4 AVALIAÇÃO DA QUALIDADE DOS TRABALHOS

Os pesquisadores Kitchenham e Charters (2007) recomendam a avaliação da qualidade dos estudos primários, isso para minimizar o viés e maximizar a validade das avaliações. Eles também listam cinco diferentes argumentos que torna a avaliação da qualidade fundamental. Neste trabalho, a avaliação da qualidade é utilizada como meio de verificar a importância dos estudos individuais para apresentação dos resultados da revisão sistemática. Para isso, foi desenvolvido um questionário de qualidade de estudo composto por 6 perguntas baseadas nas questões apresentadas nos trabalhos de Kitchenham e Charters (2007) e Dyba, Dingsoyr e Hanssen (2007). As seguintes perguntas foram utilizadas para avaliar a qualidade dos estudos primários:

- QA.1 Os objetivos e a justificativa do estudo primário estão claramente descritos?
- QA.2 O contexto em que a pesquisa foi realizada foi adequadamente apresentado?
- QA.3 A técnica de análise de dados utilizada no trabalho está claramente relatada?
- QA.4 Existe alguma validação aceitável da técnica/abordagem?
- QA.5 Os resultados estão claramente definidos e relacionados aos objetivos do estudo?
- QA.6 As conclusões têm relação ao objetivo definido e ao propósito do trabalho?

2.5 ANÁLISE DOS RESULTADOS

Para gerenciamento dos trabalhos retornados pelas bibliotecas digitais foi utilizado o *software* StArt. Esta ferramenta tem objetivo de ajudar o pesquisador na execução de uma revisão sistemática. As pesquisas nas bibliotecas digitas retornaram 739 artigos, os resultados das consultas em cada biblioteca digital estão apresentados na Tabela 2.1. Esses resultados foram avalidos pela funcionalidade da ferramenta StArt para encontrar artigos duplicados, foram encontrados 51 trabalhos duplicados. Sendo assim, depois da remoção dos artigos inválidos, o total de artigos restantes foi 688.

Depois deste processo de buscas e remoção dos trabalhos duplicados, todos os artigos foram analisados através da leitura dos títulos e *abstract*. As análises dos artigos relevantes foram feitas utilizando os critérios de inclusão e exclusão. Então, depois de realizada a análise, foram selecionados 43 artigos para serem totalmente lidos e avaliados. Após as leituras, alguns ainda foram descartados aplicando os critérios de exclusão, restando 13 trabalhos fortemente relacionados ao tema pesquisado.

A Tabela 2.2 apresenta a distribuição das publicações dos artigos aceitos por ano. Como apresentado na tabela, o ano 2017 é o que contém mais publicações relacionadas ao tema tratado.

A Tabela 2.3 apresenta a distribuição dos trabalhos de acordo com seu tipo de publicação. Diante disso, é possível perceber que os trabalhos foram mais publicados em conferências e revistas.

Tabela 2.1 Número de artigos retornados nas buscas.

Repositório	Número de Artigos
ACM	160
IEEE	166
Scopus	520
Google Academic	40
Artigos Selecionados	43
Artigos Aceitos	13

Tabela 2.2 Número de artigos publicados por ano.

Ano	Quantidade
2016	2
2017	10
2018	1

Tabela 2.3 Tipo de publicação dos artigos.

Tipo Publicação	Quantidade
Revistas	4
Simpósio	3
Conferência	5
Capítulo de livro	1

Essas análises apresentadas anteriormente trazem uma interpretação geral dos trabalhos selecionados relacionados ao tema tratado. Para um entendimento mais aprofundado dos artigos, algumas perguntas foram definidas, elas serviram de base para análise das leituras completas dos artigos selecionados. As peguntas definidas são as seguintes:

- Q1 - Qual nível da *Fog/Edge* Computing a análise de dados foi aplicada?
- Q2 - Quais foram as técnicas utilizadas na análise de dados na *Fog/Edge Computing*?
- Q3 - Em qual contexto a análise de dados foi utilizada?
- Q4 - Como a aplicação das técnicas de análise de dados na *Fog/Edge Computing* foram avaliadas?

A Tabela A.1 resume a análise dos trabalho selecionados. Diante dos resultados, é possível observar uma ampla variedade de cenários de aplicação da análise de dados

na *Fog* e *Edge Computing*, respondendo assim a QS.1. Os exemplos de cenários estão relacionados com diversos domínios. Por exemplo, o trabalho dos pesquisadores Lee et al. (2016) apresenta a aplicação das técnicas em um fogão inteligente e sistema de recirculação de água, onde o sistema pode identificar se uma panela está fervendo e também pode detectar possíveis problemas e atividades no fogão. O artigo dos autores Raafat et al. (2017) estudou análise de dados na *Fog* aplicada em dados de sensores de luminosidade, temperatura, CO2 e umidade, eles buscaram identificar a ocupação de uma sala de escritório. No trabalho de Bhargava et al. (2016) as análises foram realizadas numa rede de sensores sem fio, essa rede foi implantada numa fazenda de vaca leiteira para coleta de dados dos animais, dados de aceleração, longitude, latitude, entre outros. O artigo do autor Oyekanlu (2017) estuda a *Industrial Internet of Things (IIoT)*, ele tem o objetivo de interpretar os dados da vibração de máquinas industriais e analisar máquinas rotativas, isso com objetivo de identificar defeitos e possíveis falhas.

Outros trabalhos identificados estudaram cenários mais populares na área de Internet das Coisas. O estudo realizado por He et al. (2017) teve como aplicação o desenvolvimento de cidades inteligentes. O artigo desenvolvido por Qaisar e Usman (2017) explorou o contexto de classificação de imagens e prognóstico de saúde. Outra aplicação explorada nos trabalhos que é muito discutida na literatura é a detecção da poluição do ar, os trabalhos de Harth, Delakouridis e Anagnostopoulos (2018) e Tsai et al. (2017) trabalharam nesse tema.

Além de encontrar os ambientes de aplicação das técnicas dos trabalhos selecionados, os resultados da Tabela A.1 também apresentam como são avaliadas as técnicas de análise de dados dos trabalhos. Após os experimentos realizados nos artigos, alguns utilizaram técnicas específicas de avaliação dos resultados. Por exemplo, o trabalho desenvolvido por Raafat et al. (2017) utilizou 6 métricas de avaliação, são elas: sensibilidade; especificidade; precisão; acurácia; F1-score; G-mean. No artigo dos pesquisadores Bhargava et al. (2016) foi aplicado um método de avaliação muito utilizado na literatura, ele é conhecido como a Raiz do Erro Médio Quadrático (RMSE), nesse trabalho também foi avaliado ganho de memória utilizando a análise em R. Uma outra forma de avaliação, é a comparação dos resultados obtidos nos experimentos com simulações *offline* e separadas das aplicações. O artigo desenvolvido por Lee et al. (2016) faz a comparação dos resultados com uma implementação no MATLAB, já o trabalho de Lujic, Maio e Brandic (2017) faz comparação dos resultados com simulações do pacote de previsão do R.

2.6 TRABALHOS RELACIONADOS

Os principais trabalhos relacionados à área de análise de fluxo de dados na *Fog Compunting* para Internet das Coisas foram identificados nesta revisão e são apresentados na Tabela 2.4. Desses trabalhos, os que estão diretamente relacionados com esta proposta são os desenvolvidos por: Yang (2017); Patel, Ali e Sheth (2017); Harth, Delakouridis e Anagnostopoulos (2018); Oyekanlu (2017). Diante disso, uma tabela de comparação dos trabalhos relacionados foi desenvolvida comparando os seguintes aspectos dos trabalhos: objetivo da proposta; domínio de aplicação; dispositivos utilizados; implementação da solução; ambiente de execução.

Embora alguns desses trabalhos propõem utilização da *Fog/Edge Computing* para análise de fluxos de dados, nenhum dos trabalhos encontrados até então engloba soluções envolvendo técnicas de detecção de mudanças e resumo em fluxo contínuos e ilimitados de dados para otimizar os aspectos de latência, tráfego de dados e conectividade das aplicações IoT. Os trabalhos desenvolvidos encontrados aplicam técnicas clássicas para trabalhar com conjuntos de dados finitos, são eles: He et al. (2017); Oyekanlu (2017); Lujic, Maio e Brandic (2017). O trabalho desenvolvido por He et al. (2017) estudou regressão logística (LR) e máquina de vetor de suporte (SVM). Oyekanlu (2017) aplicou a comparação de série temporal de referência com a produzida pelos dispositivos para descobrir falhas em equipamentos industriais. Lujic, Maio e Brandic (2017) utilizou redes neurais para detecção de eventos na borda da rede.

O trabalho de Yang (2017) trata de fluxos de dados, porém é conceitual e apresenta direcionamentos sobre tecnologias. O trabalho de Patel, Ali e Sheth (2017) apresenta uma arquitetura para permitir que uma tarefa analítica seja dimensionada dinamicamente com base no contexto da névoa. O artigo desenvolvido por Harth, Delakouridis e Anagnostopoulos (2018) utiliza predição para decidir se as informações devem ser transmitidas da borda para nuvem e também realiza agregação nos dados. Além disso, uma outra diferença desta proposta de mestrado para as demais da literatura é que nenhum dos trabalhos apresentam um ambiente de simulação ou emulação da proposta desenvolvida.

A partir dos resultados é possível perceber também as técnicas mais utilizadas para análise de dados na borda da rede. As redes neurais são aplicados em dois trabalhos, nos trabalhos de Tsai et al. (2017) e Qaisar e Usman (2017). As redes neurais se baseiam na estrutura do funcionamento do sistema nervoso, tendo como objetivo simular a capacidade para aprendizado do cérebro humano (CARVALHO et al., 2011), nos trabalhos citados ela tem a funcionalidade de classificação do sinal e detecção de eventos, por exemplo, utilizando a técnica de detecção de objetos YOLO (*Real-Time Object Detection*). O método mais utilizado nos trabalhos selecionados foi a análise de séries temporais. O artigo de Lujic, Maio e Brandic (2017) estuda análise de dados de séries temporais resultantes de sensores IoT, isso com o objetivo de melhorar a eficiência no gerenciamento de dados na borda da rede. Utiliza um algoritmo adaptativo com métodos de previsão (ETS ou ARIMA). Outro trabalho que estudou séries temporais foi desenvolvido por Bandyopadhyay et al. (2016), utilizaram a técnica *Dynamic Time Warping* (DTW) para as análises. Por fim, o pesquisador Oyekanlu (2017) utilizou séries temporais para fazer predições de dados em sistemas industriais.

Uma outra área explorada nos artigos selecionados é o aprendizado de máquina. O Aprendizado de máquina é definido por Carvalho et al. (2011) como a capacidade de melhorar a execução de alguma atividade com base na experiência. Para isso, a indução é utilizada com objetivo de conseguir conclusões genéricas a partir de um conjunto de dados de exemplo. A pesquisa desenvolvida por Lee et al. (2016) utilizou um modelo de mistura gaussiana (GMM) que utiliza o algoritmo de maximização da expectativa (EM) com o critério de comprimento mínimo de descrição (MDL). Os pesquisadores He et al. (2017) utilizaram regressão logística e máquina de vetor de suporte para realização dos experimentos, eles tinham o objetivo de demostrar a viabilidade do trabalho desenvolvido com a aplicação das técnicas.

Tabela 2.4 Comparação dos trabalhos relacionados.

Autores	Proposta	Domínio Aplicação	Heterogeneidade dispositivos	Avaliação	Implementação Solução	Ambiente de execução
Yang (YANG, 2017)	Arquitetura tecnológica para processamento de fluxo de dados	Energia solar	Sim	Não informado	Não	Fog Computing
Harth, Delakouridis e Anagnostopoulos (HARTH; DELAKOURIDIS; ANAGNOSTOPOULOS, 2018)	Análise preditiva e agregação de dados	Dados poluição do ar	Sim	Não informado	Sim	Fog Computing
Oyekanlu (OYEKANLU, 2017)	Análise de série temporal: magnitude, assimetria, RMS e fatores de crista	Dados de vibração e rotação de máquinas industriais	Não	Não informado	Sim	Edge Computing
He et al. (HE et al., 2017)	Regressão logística (LR) e Máquina de vetor de suporte (SVM)	Cidades inteligentes	Sim	Benchmarking em A-Fogs	Sim	Fog Computing
Lujic, Maio e Brandic (LUJIC; MAIO; BRANDIC, 2017)	Análise de séries temporais: métodos de predição	Dados de habitação sustentável	Sim	Comparação de dados de previsão com previsão do R	Sim	Edge Computing
Ma, Fong e Millham (MA; FONG; MILLHAM, 2018)	Algoritmo Ensemble of Swarm Search	Dados do sistema de detecção automática de incêndio urbano	Sim	Acurácia, Kappa e tempo de execução dos algoritmos	Sim	Fog Computing
Dautov et al. (DAUTOV et al., 2017)	Arquitetura para processamento de fluxo de dados	Detecção de rostos	Sim	Tempo de atraso e Throughput	Sim	Edge/Fog Computing
Raafat et al. (RAAFAT et al., 2017)	Redes neurais para classificação de sinais e detecção de eventos	Dados de sensores: luminosidade; temperatura; CO2 e umidade	Não	Sensibilidade, especificidade, precisão, exatidão, F1-score, G-mean	Sim	Fog Computing
Lavassani, Forsström e Zhang (LAVASSANI et al., 2018)	Métodos preditivos baseados em simulação de stream	Dados de ambiente industrial	Não	Número de transmissão de pacotes, atraso, tempos de consulta	Sim	Fog Computing
Abordagem Proposta	**Abordagem de Fog para processamento e análise de fluxo de dados da IoT utilizando wavelet e concept drift**	**Dados dos sensores: temperatura; luminosidade; umidade; som e poeira**	**Sim**	**Dados trafegados na rede, delay de processamento, e acurácia dos dados analisados**	**Sim**	**Fog of Things**

Nesta revisão também foram selecionados trabalhos que apresentam direcionamentos tecnológicos. Os trabalhos com arquiteturas tecnológicas não apresentam técnicas específicas para serem aplicadas na *Fog* ou *Edge Computing*, porém, apresentam tecnologias que permitem a utilização de diversos métodos de análise de dados. A pesquisa desenvolvida por Patel, Ali e Sheth (2017) apresenta uma arquitetura para análises de dados IoT na *Fog Computing*. Essa arquitetura define perfis de aplicações que podem auxiliar no desenvolvimento de sistemas para análise de dados na borda da rede. O trabalho de Yang (2017) expõe direcionamentos sobre tecnologias que dão suporte para o processamento de fluxo de dados na borda da rede, essas tecnologias são dividas por camadas no artigo. O autor aponta que detecção de anomalias, agrupamento, classificação, sumarização e regressão fazem parte do ciclo de vida do processamento de fluxo de dados, sendo suportados pelas tecnologias citadas no trabalho.

Em resumo, esta revisão sistemática da literatura encontrou os principais trabalhos relacionados com análise de fluxo de dados na borda da rede. Apresentou os principais cenários que foram aplicadas, quais foram as formas de avaliação e com qual objetivo foram utilizadas as análises de dados. Com isso, esta revisão deixa evidente a importância dessa área para Internet das Coisas, assim como fornece a base para o posicionamento desta proposta de mestrado. Diante disso, pode-se entender a proposta deste trabalho como uma abordagem de *Fog* para mineração de fluxos de dados. Essa abordagem de mineração possibilita a criação de modelos de predição online reduzindo a quantidade necessária de dados para o treinamento.

2.7 CONSIDERAÇÕES FINAIS

Este capítulo descreveu a revisão sistemática da literatura empregada nesta pesquisa. A revisão permitiu sumarizar as técnicas de análise de dados na borda da rede encontradas nos trabalhos selecionados. A partir desses resultados primários foi possível definir os trabalhos relacionados a análises de fluxo de dados. A partir disso, foi discutido neste capítulo os trabalhos relacionados a pesquisa desenvolvida, posicionando este trabalho dentro do que a literatura da área tem trabalhado. Além disso, os resultados apresentados nesta seção são importantes para responder a questão principal da revisão sistemática. Os dados levantados de todos os trabalhos estão sumarizados na Tabela A.1 e os trabalhos relacionados estão apresentados na Seção 2.4. No próximo capítulo a abordagem de processamento e análise de fluxo de dados na *Fog Computing* será discutida com mais detalhes.

Capítulo 3

FUNDAMENTAÇÃO TEÓRICA

Neste Capítulo, são apresentados os principais aspectos conceituais que introduzem as áreas de Internet das Coisas e Aprendizado de Máquina. Esses aspectos conceituais descrevem as principais questões que envolvem os temas: Internet das Coisas, *Fog Computing*, *Fog of Things*, Mineração de Fluxo de Dados e Redes Neurais Artificiais. O objetivo é mostrar os conceitos (incluindo tecnologias, funcionalidades e características) que estão relacionados a esses temas e que serviram como suporte para as modelagens e estratégias apresentadas no decorrer deste trabalho.

3.1 INTERNET DAS COISAS

A Internet das Coisas é definida por Whitmore, Agarwal e Xu (2015) como a conexão de dispositivos, equipados com recursos de identificação, detecção, rede e processamento, com a internet, a qual permite uma comunicação entre eles com a finalidade de atingir um objetivo útil. Como explica também os autores, não existe uma definição universal para IoT, as pesquisas de cada área da Internet das Coisas têm sua própria visão sobre o tema.

O W3C (World Wide Web Consortium) aborda a Internet das Coisas como Web das Coisas, trazendo a perspectiva da IoT para as tecnologias Web. O W3C descreve a Web das Coisas como o uso das tecnologias Web para facilitar o desenvolvimento de aplicações para IoT, sendo composta por objetos físicos, por exemplo, código de barra NFC (Near-field communication), tecnologias Web como HTTP, serviços RESTful e APIs JavaScript (MINERVA; BIRU; ROTONDI, 2015).

Outros trabalhos buscam trazer modelos de referência para IoT. Um projeto que pode-se destacar nesse sentido é o IoT-A (BAUER et al., 2013). Ele é um projeto europeu com objetivo de desenvolver um modelo de referência para arquitetura de projetos da Internet das Coisas. O IoT-A define Internet das Coisas como um termo que engloba tecnologias, dispositivos, objetos e serviços.

A União Europeia financia um projeto chamado IERC (IoT European Research Cluster) que visa potencializar a utilização da Internet das Coisas na Europa, assim como

possibilitar a convergência das atividades em andamento. A IERC descreve IoT como um infraestrutura de rede global, com padrões definidos, protocolos que permitam a interoperabilidade, composta por coisas físicas e virtuais que possuem identidade, personalidades virtuais e atributos físicos (MINERVA; BIRU; ROTONDI, 2015).

Na literatura, diversos trabalhos têm discutido a Internet das Coisas com diferentes visões sobre o tema (MINERVA; BIRU; ROTONDI, 2015), assim como o seu impacto nas várias áreas sociais. Os pesquisadores Atzori, Iera e Morabito (2010) apresentam as visões, exibidas na Figura 3.1, para IoT. A visão *Things-oriented* (Orientada a Coisas) está ligada aos objetos simples da IoT como, por exemplo, identificação de rádio frequência (RFID). Essa visão surge em parte por causa do termo Internet das Coisas ter sido criado em uma rede de laboratórios de pesquisa no campo de RFID, chamado The Auto-ID Labs[1]. As pesquisas nessa área trabalham na criação de padrões para melhorar a rastreabilidade dos objetos RFID, de acordo com o seu status e sua localização. Essa tecnologia é um dos pilares da IoT, porém não é a única. Outras tecnologias como Near Field Communications (NFC) e Wireles Sensor e Actuator Networks (WSAN) são consideradas também bases do desenvolvimento da IoT, tendo o objetivo de conectar os dispositivos físicos ao mundo digital.

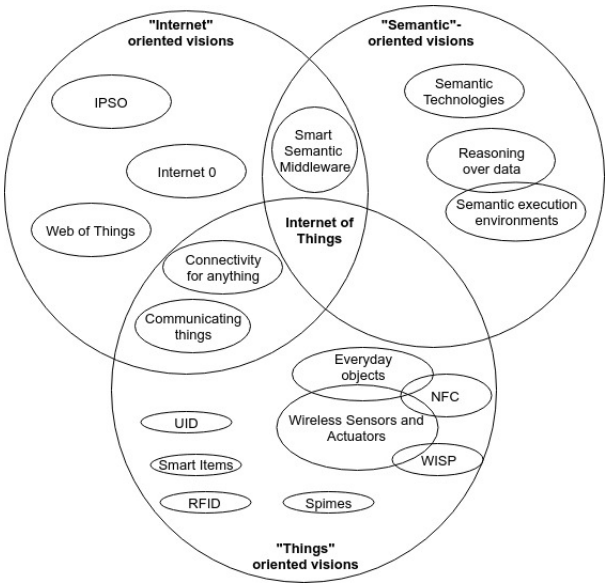

Figura 3.1 Visões da Internet das Coisas, adaptado de: (ATZORI; IERA; MORABITO, 2010).

A visão *Internet-oriented* (Orientada a Internet) foi proposta pelo IPSO[2] (IP for Smart Objects). A proposta do IPSO é que a pilha de protocolos IP já é utilizada para conectar uma grande quantidade de dispositivos e pode funcionar em dispositivos pequenos e com bateria limitada. Sendo assim, o IP parece ter todas característica para tornar a Internet das Coisas uma realidade. As tecnologias que dão suporte para essa visão é o IP e sua arquitetura incorporados ao padrão IEEE 802.15.4 com a visão 6LoWPAN (DUNKELS;

[1] www.autoidlabs.org
[2] http://www.ipso-alliance.org

VASSEUR, 2008).

Na visão *Semantic-Oriented* (Orientada a Semântica) a preocupação está em como representar, armazenar e interconectar a grande quantidade de dados gerados pela IoT. Nesse cenário, as tecnologias semânticas são importantes para solucionar os problemas relacionados a estruturação dos dados da IoT. Através das ferramentas semânticas, é possível realizar raciocínio sobre os dados, desenvolver propostas de arquiteturas para armazenamento escalável e ambientes de execução semântica. Por fim, como mencionado anteriormente, a última visão sobre IoT encontrada na literatura é a Web das Coisa, ela tem o objetivo de reutilizar as tecnologias da Web para integrar os objetos da IoT (ATZORI; IERA; MORABITO, 2010).

Nessas visões que a literatura apresenta sobre Internet das Coisas, existem várias tecnologias que dão sustentação para seu funcionamento. Na área de identificação de objetos a principal tecnologia foi, citada anteriormente, o RFID. Para sensoriamento, na maioria das vezes, é criada uma rede de nós de sensores que se comunicam através de tecnologias sem fio. As informações lidas são enviadas para os *gateways* IoT que funcionam como um mediador entre os dispositivos físicos e a conexão externa à da rede sensores. Outra tecnologia que dá suporte para o funcionamento da IoT é o *middleware*, ele tem a finalidade de abstrair as diferentes tecnologias envolvidas numa aplicação para Internet das Coisas, sendo composto por diferentes camadas com funcionalidades distintas.

A Figura 3.2 apresenta um resumo das tecnologias que compõem a Internet das Coisas. Nessa pilha de tecnologias, as duas partes importantes que vale a pena destacar são as camadas de computação e comunicação. A camada de computação apresenta os *hardwares* e *softwares* mais utilizados. Para os dispositivos físicos, o Raspberry Pi, na maioria das vezes, é utilizado como *gateway* IoT. Para coleta de informações dos sensores no ambiente o Arduino e o Intel Galileo são os mais utilizados. Na camada de comunicação, o WiFi é uma das principais tecnologias para comunicação dos dispositivos de longo alcance. Por outro lado, para uma rede mais limitada o Bluetooth é o mais utilizado.

Elementos da IoT		Exemplos
Identificação	Nome	EPC, uCode
	Endereço	IPv4, IPv6
Sensores		Sensores inteligentes RFID Tag, Atuadores, Wearable, Sensores embarcados
Comunicação		RFID, NFC, UWB, Bluetooth, BLE, IEEE, Wi-Fi, LTE-A, WiFiDirect, 802.15.4
Computação	Hardware	SmartThings, Arduino, Raspberry Pi, Intel Galileo, BeagleBone, Cubieboard, Phidgets, Cubieboard
	Software	OS(Contiki, TinyOS, LiteOS, Riot OS, Android); Clud(Nimbits, Hadoop, etc.)
Serviços		Agregação de informação (smart grid), identify-related (shipping), collaborative-aware (smart home), ubíquo (smart city)
Semântica		RDF, OWL, EXI

Figura 3.2 Tecnologias Habilitadoras da IoT, adaptado de: (AL-FUQAHA et al., 2015).

Os protocolos da camada de aplicação da IoT mais utilizados na literatura são o COAP (*Constrained Application Protocol*) e o MQTT(*Message Queue Telemetry Transport*). O COAP é um protocolo para transferência de informações na Web baseado em REST (*REpresentational State Transfer*) que utiliza os métodos *GET*, *POST*, *PUT* e *DELETE* do HTTP. O COAP utiliza UDP que é mais adequado para aplicações IoT. Além disso, o COAP altera as funcionalidades do HTTP para atender requisitos da IoT de baixo consumo de energia.

O MQTT é um protocolo de mensagens que tem o objetivo de conectar dispositivos e redes com aplicações e *middlewares*. As operações que o MQTT permite são: *one-to-one* (um-para-um); *one-to-many* (um-para-muitos); *many-to-many* (muitos-para-muitos). Outra característica do MQTT é a utilização do padrão *publish/subscribe*, ele fornece flexibilidade e simplicidade na implementação de aplicações IoT. Por causa dessas características, o MQTT também é adequado para dispositivos com baixo recurso computacional e com comunição de rede não confiável ou que possuem baixa largura de banda. O MQTT é composto por três partes: *subscriber*, *publisher* e *broker*. O funcionamento básico do MQTT, apresentado na Figura 3.3, ocorre quando algum dispositivo subscreve um tópico específico para que seja informado quando publicações forem feitas no tópico. O componente *publisher* fica responsável por gerenciar a publicação dos dados. Por fim, os dados são transmitidos aos *subscribers* pelo *broker*. Devido a essas características apresentadas, o MQTT é considerado um importante protocolo para o desenvolvimento de aplicações IoT e utilização em sistemas que precisam de comunicação M2M (Máquina a Máquina) (AL-FUQAHA et al., 2015).

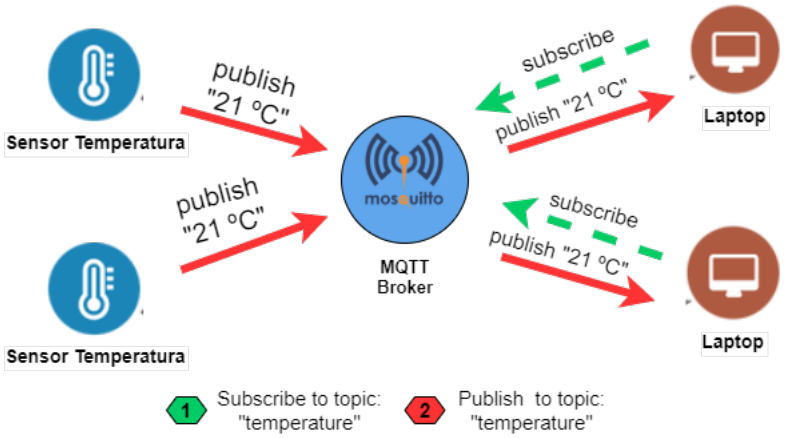

Figura 3.3 Funcionamento do MQTT, adaptado de: (AL-FUQAHA et al., 2015).

Diante das potencialidades da Internet das Coisas, alguns domínios da sociedade podem tirar proveito das possibilidades da sua utilização, alguns desses domínios são apresentados na Figura 3.4. Nesses cenários, um de grande impacto é o *smart environments* (Ambientes Inteligentes). Um exemplo desse domínio é a utilização de sensores e atuadores em escritórios e casas para tornar o ambiente inteligente, permitindo, por exemplo, a configuração de temperatura de um quarto de acordo com a preferência do morador e ao clima, a iluminação pode ser alterada de acordo com a hora do dia, acidentes domésticos

podem ser evitados com sistemas de monitoramento e alarme, e energia pode ser economizada através do desligamento dos equipamentos elétricos que não estiverem sendo utilizados (ATZORI; IERA; MORABITO, 2010).

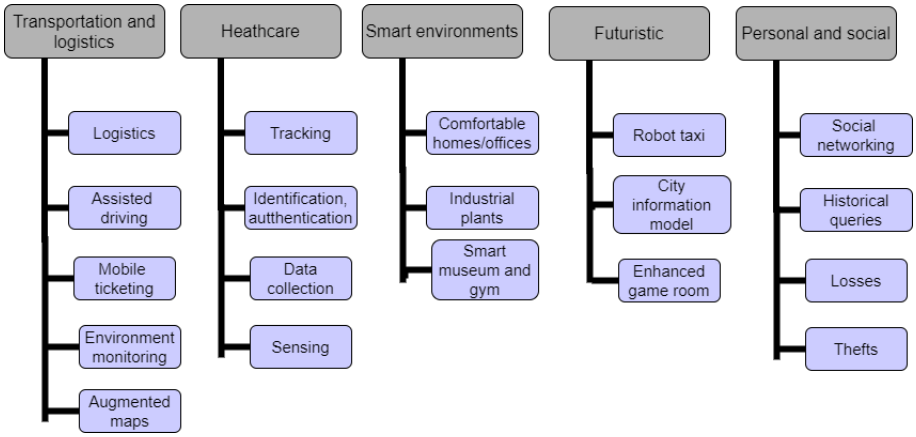

Figura 3.4 Domínios de Aplicação IoT, adaptado de: (ATZORI; IERA; MORABITO, 2010).

Os ambientes inteligentes também podem ser utilizados para melhorar instalações industriais. Para isso, *tags* RFID podem ser instaladas nos aparelhos industriais. Os dados das *tags* dos equipamentos podem ser lidos e disponibilizados na rede. Com a utilização de outros sensores, tomadas de decisão inteligente podem ser realizadas. Por exemplo, um sensor de pressão pode identificar algum desequilíbrio na linha de produção, e com isso imediatamente tomar a ação de parar a produção para evitar estragos e perdas de materiais e equipamentos.

Todos esses aspectos consistem o início do desenvolvimento da Internet das Coisas. O Gartner[3] prevê que, no futuro da IoT, 26 bilhões de dispositivos sejam conectados à Internet das Coisas em 2020. A IoT tem o potencial de transformar os processos de negócios das empresas e indústrias com disponibilização de informações da cadeia de produção em tempo real, melhora no fluxo de trabalho da fábrica e otimização dos custos de distribuição.

As pesquisas em IoT têm adotado a *Fog Computing*, em geral, para diminuir a latência das aplicações e reduzir a sobrecarga de dados na Cloud Computing. Na próxima Seção 3.2, serão apresentados os aspectos relacionados da Fog Computing para IoT.

3.2 FOG COMPUTING

A *Cloud Computing* é um paradigma que abstrai, para usuário final e empresas, especificações e detalhes da infraestrutura computacional necessárias para o funcionamento das aplicações. Porém, apesar dessa facilidade, para aplicações sensíveis à latência, que necessitam que os *nodes* na vizinhança funcionem de acordo com os requisitos de atrasos, a solução de *Cloud Computing* não é a mais indicada. As aplicações da Internet das Coisas estão enquadradas nessa perspectiva de sensibilidade a latência. Além disso,

[3]https://www.gartner.com/newsroom/id/2636073

requerem suporte de mobilidade, distribuição geográfica e conscientização de localização (BONOMI et al., 2012).

Diante desse cenário, surge uma nova plataforma conhecida como *Fog Computing* (Computação em Névoa) para atender os requisitos impostos pelas aplicações descritas anteriormente. Os pesquisadores Bonomi et al. (2012) explicam que *Fog Computing* não é um novo paradigma que exclui a *Cloud Computing*, a computação em névoa permite a criação de uma nova geração de aplicativos e serviços com interação com a computação em nuvem para gerenciamento e análise de dados. O OpenFog define *Fog Computing* como uma arquitetura horizontal de nível de sistema que distribui funções de computação, armazenamento, controle e rede para mais perto dos usuários ao longo de uma nuvem contínua para o objetos (CONSORTIUM et al., 2017).

A *Fog Computing* é uma plataforma virtualizada que fornece serviços de infraestrutura computacional, armazenamento e rede para comunicação entre os dispositivos finais e *data centers* da *Cloud Computing*, como apresentado na Figura 3.5 (BONOMI et al., 2012). Essa arquitetura é dividida em quatro níveis, como apresentado na Figura 3.5. O primeiro nível, Embedded Systems and Sensors, está relacionado à rede de sensores com baixo poder computacional e largura de banda, que são empregados para tarefas específicas no ambiente. O segundo nível, Multi-Service Edge, é onde ocorre a distribuição dos serviços na borda da rede, permitindo alcançar inteligência no ambiente de *Fog Computing*. As principais tecnologias utilizadas são 3G, 4G, LTE e WiFi. Na camada Core, são aplicadas tecnologias de comunicação de longo alcance (e.g IP/MPLS) e também são implementados requisitos de segurança e QoS. Por fim, o nível de Data Center e Cloud é onde as aplicações que necessitam de mais recursos computacionais são implantadas, i.e processamento, análise e armazenamento dos dados gerados pelos ambientes.

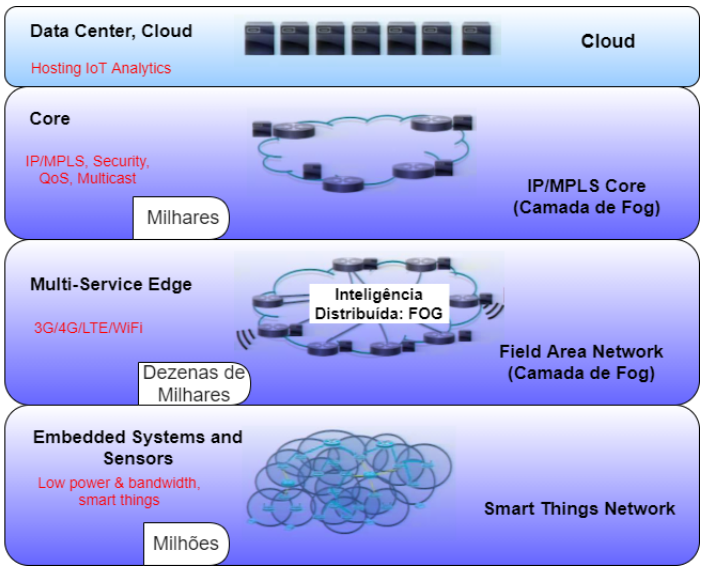

Figura 3.5 Internet das Coisas e *Fog Computing*, adaptado de: (BONOMI et al., 2012).

A computação em névoa apresenta características que não são triviais de serem implementadas como, por exemplo, distribuição geográfica dos nós, rede de sensores de larga

escala para monitorar o ambiente, exigindo recursos computacionais e de armazenamento distribuídos e comunicações em tempo real em vez de processamento em lote. Essas funcionalidades são difíceis de serem implementadas, pois demandam desenvolvimento de algoritmos e infraestruturas de alto desempenho para suportar as especificações que as aplicações da Internet das Coisas exigem.

Um ambiente que pode exemplificar os desafios e soluções que a computação em névoa propicia para Internet das Coisas é a rede de carros conectados. Essa rede apresenta um cenário de interações entre carros e pontos de acesso (e.g Wi-Fi, 3G, LTE, unidades rodoviárias, semáforos inteligentes), contexto que a *Fog Computing* pode oferecer soluções como, por exemplo, conexão dos objetos distribuídos ao longo das cidades e estradas, mobilidade, localização, baixa latência nas comunicações e suporte para interações em tempo real (BONOMI et al., 2012).

Os ambientes de computação em névoa têm a característica de gerar uma grande quantidade de dados, popularmente conhecido como *Big Data* o qual é um termo que descreve o imenso volume de dados e como conhecimentos podem ser gerados a partir da análise desses dados, hoje é caracterizado em três dimensões: volume, velocidade e variedade. No trabalho de Bonomi et al. (2014), é sugerida um nova dimensão para o conceito de *Big Data* chamada de geo-distribuição, que visa trabalhar com os sensores e atuadores distribuídos ao longo dos ambientes de Internet das Coisas, tendo a necessidade de serem gerenciados como uma única rede.

A Figura 3.6 apresenta os níveis de análise de dados que a geo-distribuição propõe na *Fog Computing*. No primeiro nível, a análise dos dados que os sensores geram na névoa é feita em milissegundos e segundos com baixa latência e em tempo real, produzindo dados operacionais. Subindo um nível, as análises são feitas em segundos e minutos, a velocidade de processamento e a latência são médias, ao contrário do anterior, as interações não são entre máquinas (M2M), existe a interação dos usuários com a aplicação, os dados gerados não são operacionais, mas podem conter informações históricas. No nível mais acima, a análise dos dados é feita em minutos e dias, gerando dados transacionais. O último nível trabalha com dados para *Business Intelligence*, os quais são analisados em dias e meses, com alto processamento e latência e interação dos usuários para acessar relatórios e *dashboards*.

3.2.1 Fog of Things e SOFT-IoT

Várias soluções para ambientes inteligentes da IoT baseadas em *Cloud Computing* foram propostas pela indústria e pela academia com foco na prestação de serviços e na interoperabilidade. No entanto, as soluções baseadas em nuvem têm algumas limitações, como escalabilidade, alta demanda por largura de banda e alta latência. Como apresentado na Seção 3.2, a *Fog Computing* foi projetada para resolver alguns desses problemas e também pode satisfazer os seguintes requisitos: a conectividade com a Internet não é fundamental; o processamento de dados e aplicativos ocorre dentro dos limites da rede local; o uso de largura de banda é otimizado; escalabilidade, através do uso de pequenos servidores locais (perto de dispositivos, aplicativos e/ou usuários) (PRAZERES; SERRANO, 2016).

Com o objetivo de tirar proveito das vantagens que a *Fog Computing* pode propiciar

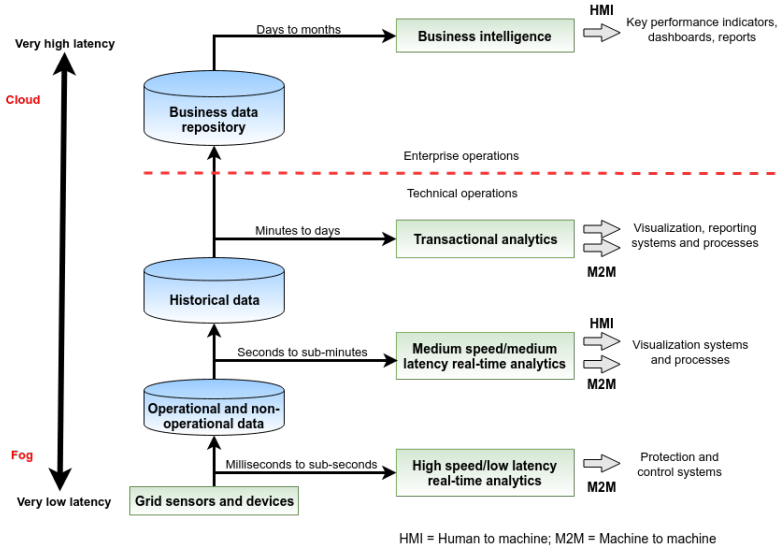

Figura 3.6 Níveis de análise de dados na computação em névoa, adaptado de: (BONOMI et al., 2014).

para Internet das Coisas, os pesquisadores Prazeres e Serrano (2016) propuseram um novo paradigma chamado de *Fog of Things* (FoT). Nesse paradigma, assim como a *Fog Computing*, parte da capacidade de processamento de dados e operações de entrega de serviços são processados localmente em pequenos servidores. A *Fog of Things* vai além *Fog Computing* nos seguintes aspectos:

- Utiliza toda a capacidade de processamento da borda da rede através do processamento de dados e entrega de serviços em dispositivos, *gateways* e pequenos servidores locais;
- Definindo os serviços da Internet das Coisas na borda da rede;
- Distribuindo os serviços IoT na borda da rede através de um *middleware* orientado para mensagens e serviços.

Como pode ser visto na Figura 3.7, uma plataforma baseada no paradigma *Fog of Things* deve ser composta pelos seguintes componentes: aplicações, dispositivos, *gateways*, servidores, *middleware* orientado a serviços de mensagens e provedores de segurança. Segundo Prazeres e Serrano (2016) a *Fog of Things* possui as seguintes características de organização:

- *FoT-Device* - dispositivos que realizam a integração de objetos físicos do mundo real com o mundo digital da Internet. Também realiza a transformação de dados brutos em conteúdo estruturado seguindo as diretrizes do *Linked Data*. Além disso, os dispositivos da FoT não requerem conectividade com a Internet porque eles realizam comunicação diretamente com os *gateways* que os gerenciam. Desse modo, os dispositivos podem continuar monitorando (sensores) e/ou atuando (atuadores)

no ambiente, mesmo sem conexão à Internet, e todos os dados são enviados para serem processados nos *gateways*;

- *FoT-Gateway* - *node* de rede que tem como objetivo principal mapear os protocolos da camada de comunicação (*Ethernet*, Wi-Fi, *ZigBee*, *Bluetooth*) para HTTP (camada de aplicativo da Web). Sendo assim, um *FoT-Gateway* oferece serviços Web RESTful para acesso as funcionalidades dos *FoT-Devices* e a outros serviços da IoT. Assim, as aplicações podem abstrair o protocolo de comunicação com os dispositivos e acessá-los através de uma API REST, padronizada para cada tipo de dispositivo, como é o funcionamento atual da maioria dos aplicativos da *Web*;

- *FoT-Server* - servidores mais robustos que o *FoT-Gateway* e podem ser de dois tipos, um tipo especial de *gateway* aprimorado com recursos de gerenciamento, fornecendo informações específicas que não são suportadas por dispositivos e *gateways*. Por exemplo, dados históricos de longo prazo de dispositivos;

- *Application* - aplicações que utilizam a API REST baseada em HTTP (RESTful Web Services) fornecida pelos *FoT-Gateways* para acessar os serviços IoT fornecidos pela *FoT* e oferecem para os usuários interação com serviços. Os aplicativos podem ser *Web*, dispositivos móveis (Android, iPhone ou Windows Phone) ou mesmo aplicativos *desktop* tradicionais;

- *FoT-Profile* - definição dos tipos de serviços que os *nodes* da *FoT* (*FoT-Gateway* e *FoT-Servers*) podem ser configurados. Os perfis que os *nodes* da FoT podem assumir são composição, descoberta, segurança, entre outros.

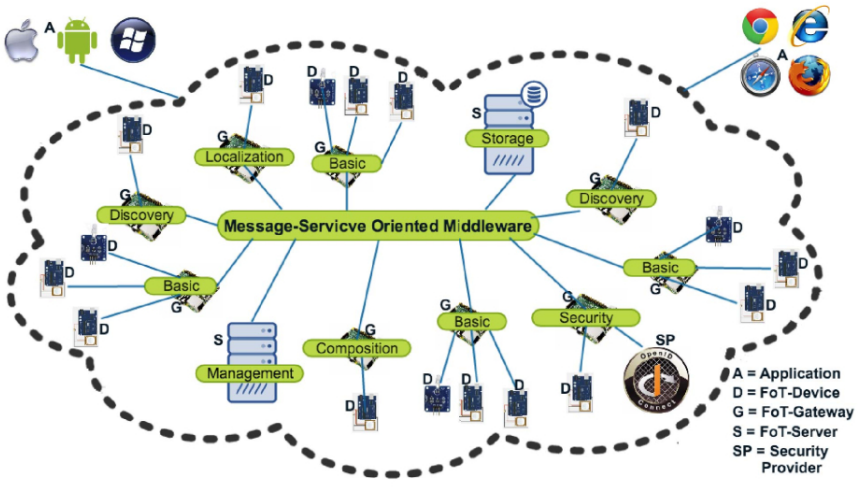

Figura 3.7 Visão Geral SOFT-IoT (PRAZERES; SERRANO, 2016).

Os pesquisadores Prazeres e Serrano (2016) apresentam também a plataforma SOFT-IoT (*Self-Organizing Fog of Things for The Internet of Things*) que é uma proposta de

implementação concreta do paradigma *Fog of Things*. Os *FoT-Devices* na plataforma SOFT-IoT são *nodes* de sensores ou atuadores embutidos com um *driver* (*TATUDevice*) o qual é implementado a partir de um protocolo leve denominado TATU (*The Accessible Thing Universe Protocol*). O protocolo TATU estende o protocolo MQTT através da padronização das mensagens para comunicação entre *FoT-Devices* e *FoT-Gateways*, além disso, *TATUDevice* também implementa o protocolo *Zeroconf* e uma descrição semântica baseada em uma taxonomia (ontologia) para dispositivos.

O *gateway* da SOFT-IoT implementa dois *middlewares*, um orientado para serviços e outro orientado a mensagens. O *middleware* orientado para serviços fornece comunicação entre os aplicativos e os serviços implantados nos *gateways*, isso utilizando um *middleware* ESB (*Enterprise Service Bus*) com base na especificação OSGi[4]. Por outro lado, a parte orientada a mensagens fornece comunicação entre os *FoT-Gateways* e os *FoT-Devices* da SOFT-IoT. Portanto, um *FoT-Gateway* na plataforma SOFT-IoT é, em geral, um dispositivo de baixo custo com processamento e recursos de memória limitados, que utiliza versões reduzidas do sistema operacional Linux, uma máquina virtual Java, uma implementação da especificação OSGi (parte orientada para o serviço) e um servidor MQTT (parte orientada para mensagens), as tecnologias utilizadas no *FoT-Gateway* são mostradas na Figura 3.8.

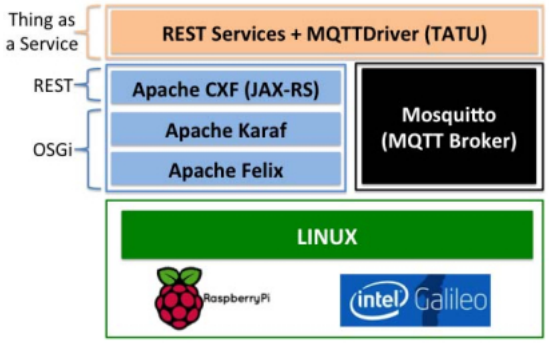

Figura 3.8 Gateway plataforma SOFT-IoT (PRAZERES; SERRANO, 2016).

Na arquitetura mostrada na Figura 3.8, o *middleware* orientado a serviços serve para fornecer interfaces para aplicativos acessarem os dispositivos via *Web Services* RESTful, utilizando a tecnologia Apache CXF. Por sua vez, o *broker* MQTT Mosquitto serve para fornecer comunicação através de mensagens entre o *gateway* e os dispositivos, utilizando um *driver* MQTT integrado nos *Web Services* RESTful (PRAZERES; SERRANO, 2016).

O MQTTDriver é um *driver* para permitir a comunicação entre *FoT-Gateways* e *FoT-Devices* utilizando o protocolo TATU. O MQTTDriver ajuda o desenvolvedor na tarefa de se comunicar com os dispositivos, implementando uma comunicação virtual entre o serviço e o dispositivo. Alguns dos recursos implementados no MQTTDriver são: alterar o valor de um atuador; ler o valor de um sensor; obter propriedades do dispositivo como o IP; e editar as propriedades do dispositivo.

[4]https://www.osgi.org/developer/architecture/

A plataforma SOFT-IoT propõe três tipos de servidores FoT: servidor de gerenciamento, servidor de armazenamento e um servidor provedor de segurança. O primeiro é um servidor FoT que funciona como um tipo especial de *gateway* e implementa todos os serviços relacionados com a auto-organização dinâmica da plataforma SOFT-IoT. Os dois últimos são *FoT-Servers* que funcionam como um tipo especial de recursos e implementa serviços relacionados aos aspectos de armazenamento e segurança, respectivamente. A plataforma SOFT-IoT, por se basear em serviços, torna-se uma plataforma suficientemente flexível para implantação dinâmica de novos tipos de servidores FoT oferecendo seus serviços para fins específicos. Por exemplo, os serviços IoT para análise de grandes volumes de dados (*Big Data Analytics*) podem ser desenvolvidos e implementados em *FoT-Servers* para análise de dados (PRAZERES; SERRANO, 2016).

A plataforma SOFT-IoT também propõe recursos de auto-organização que devem ser implantados nos servidores da FoT. Por exemplo, servidor de gerenciamento devem implementar todos os serviços relacionados à auto-organização da plataforma SOFT-IoT. Os principais serviços que devem ser implementados nos servidores são: serviço de monitoramento auto-organizado, serviço de implantação de *gateway*, serviço de recuperação de falhas e serviço de gerenciamento e balanceamento dos perfis.

Essas propostas de plataforma IoT produzem dados em fluxo contínuo e em tempo real, conforme descrito na literatura da área de *Big Data Streams*. Além dessa área, existe também a mineração de *data stream* (ou mineração de fluxo de dados) que utiliza algoritmos para tratar o volume de dados à medida que são produzidos. As próximas seções descrevem com mais detalhes conceitos dessas áreas.

3.3 BIG DATA STREAM E MINERAÇÃO DE FLUXO DE DADOS

Big Data Streams tem se tornando rapidamente um importante paradigma no campo da ciência de dados (BIFET, 2013). O *Big Data Streams* vem surgindo com a evolução do poder computacional, pois tornou-se possível a captura contínua de dados dos mais variados campos da ciência. Exemplos desses dados são as informações de satélites, fluxo de clicks em páginas Web, dados produzidos pelas redes de sensores, os quais são produtores contínuos de dados. *Data stream* é definido como $S = \{S_1, S_2, ..., S_n\}$, onde Si é gerado por uma distribuição Di. Devido à grande produção de *data stream*, Pecori e Riccardo (PECORI, 2018) afirmam que tecnologias e metodologias específicas são necessárias para lidar com esses fluxos praticamente infinitos e não estacionários.

O trabalho de Bifet e Kirkby (2009) apresenta quatro características importantes que os algoritmos desenvolvidos para tratar fluxos de dados devem considerar. A primeira característica é processar um exemplo de cada vez e inspecionar também apenas uma vez os dados na ordem de chegada. Entretanto, os algoritmos podem utilizar uma memória de curta duração para armazenar um subconjunto dos dados. A segunda característica é utilizar uma quantidade limitada de memória. A terceira característica para ser considerada é a complexidade de tempo que deve ser linear ou sublinear, tendo que operar em tempo limitado. Por fim, com novos dados para serem processados, os algoritmos devem responder a qualquer momento.

Em geral, os algoritmos tradicionais de *Big Data* não são adequados para *data stream*,

pois são executados em *batch*. Devido às limitações apresentadas pelos algoritmos em *batch*, discutidas em (GAMA, 2010), a área de pesquisa denominada mineração de fluxos de dados foi desenvolvida. Os algoritmos que são empregados para mineração de fluxos de dados, na maioria das vezes, não precisam processar todos os dados anteriores, mas apenas os dados mais recentes. Em alguns casos, são utilizadas janelas deslizantes de tamanho fixo. Esses tipos de janelas são parecidas com a estrutura de dados FIFO (first in, first out). Nas janelas, sempre que um elemento j é observado e inserido, outro elemento j - w, onde w representa o tamanho da janela, é esquecido. Existem dois tipos básicos de janelas deslizantes: baseada em sequência e baseada em tempo. Na janela baseada em sequência, o tamanho da janela é definido em termos do número de observações. Nas janelas baseadas em tempo, o tamanho da janela é definido em termos de duração. Uma janela baseada em tempo de tamanho t consiste em todos os elementos cujo registro de data e hora estão dentro de um intervalo de tempo t do período de tempo atual.

As janelas não são os únicos métodos aplicáveis em *data stream*, pois o grande volume de fluxos de dados apresenta restrições únicas de espaço e tempo para o processamento computacional. Os algoritmos de mineração de fluxo de dados exigem um processamento eficiente dos dados, o que pode ser difícil obter com a velocidade que os dados são gerados nos fluxos. Diante disso, métodos de redução dos dados podem ser utilizados. Na Seção 3.3.1, apresenta-se a técnica de redução de dados aplicada neste trabalho, denominada wavelet discreta Haar.

A mineração de fluxo de dados também leva em consideração o fato do fluxo de dados mudar seu comportamento ao longo do tempo. A detecção de mudanças no comportamento no *data stream* é realizada utilizando os algoritmos aplicados na identificação de *concept drift* (mudança de conceito). Os pesquisadores em (GAMA; RODRIGUES, 2009) afirmam que um dos principais benefícios do modelo de detecção de mudanças é que ele pode fornecer uma descrição significativa e quantificação das alterações nos dados, indicando pontos de mudança ou pequenos períodos de tempo onde elas ocorrem. Um *concept drift* não é um *outlier* ou ruído. No trabalho de Gama et al. (GAMA et al., 2014) é explicado que um dos desafios para os algoritmos de identificação de *concept drift* é não misturar a verdadeira alteração no comportamento com um *outlier* ou ruído.

Na literatura, técnicas são propostas para identificação da mudança de conceito (*concept drift*) no fluxo de dados. Cada técnica possui sua complexidade computacional. Sendo assim, na Seção 3.3.2, são apresentadas técnicas de menor complexidade computacional aplicável na *Fog Computing* da IoT. As técnicas de análise sequencial CUSUM (Cumulative Sum) e Page-Hinkley são descritas nas Seções 3.3.2.1 e 3.3.2.2. A técnica de análise baseada no controle estatístico EWMA (Exponentially Weighted Moving Average) é descrita na Seção 3.3.2.3.

Recentemente foram desenvolvidas tecnologias para mineração de dados em ambientes *Big Data Streams*. O Apache Flink[5] fornece distribuição de dados, comunicação e tolerância a falhas para cálculos distribuídos sobre fluxos de dados, incluindo também uma biblioteca de Aprendizado de Máquina. O Apache Spark[6] tem seu próprio módulo para

[5]http://flink.apache.org/
[6]http://spark.apache.org/

manipulação de fluxo de dados e biblioteca de algoritmos de Aprendizado de Máquina, chamada MLib. O H2O[7] é um framework de código aberto que fornece um mecanismo de processamento paralelo, bibliotecas analíticas e de Aprendizado de Máquina, juntamente com ferramentas de pré-processamento e avaliação. Apesar dessas tecnologias fornecerem suporte para fluxo de dados, no cenário de IoT e Fog Computing existe a limitação de recursos computacionais para funcionamento dessas tecnologias tais tecnologias são mais adequadas para hardwares mais robustos, pois necessitam de um maior poder de processamento.

Nas próximas seções, são apresentadas técnicas que podem ser aplicadas nos dispositivos IoT na Fog Computing para tratar o processamento de dados na borda da rede otimizando o uso do poder computacional.

3.3.1 Transformada Wavelet Discreta

Neste trabalho, foi utilizada a transformada wavelet para criar uma decomposição das características dos dados e realizar uma amostragem das observações. A técnica wavelet captura tendências em funções numéricas decompondo um sinal em um conjunto de coeficientes. Diferentes coeficientes contêm informações sobre as características do sinal em diferentes escalas. Coeficientes em escala mais alta capturam tendências globais do sinal, enquanto coeficientes em escalas mais baixas (finas) capturam detalhes e características locais. O conjunto reduzido de coeficientes é de grande interesse para algoritmos de *streaming* (GAMA, 2010). Dentre todas as funções da família Wavelet, pode-se destacar a de Haar, a qual pode ser escrita da seguinte maneira:

$$C_{j-1,i} = (C_{j,2i} + C_{j,2i+1})/2, i = 0, ..., 2^{j-1} - 1 \quad (3.1)$$

Nessa formulação, partindo de um vetor inicial C_J com dimensões $C_J = 2^J$, são aplicadas sequências de transformações aos vetores C_j, com j variando de J até 1, na qual apenas as médias de valores dois a dois disjuntos são calculadas. O índice i indica a posição dos elementos do novo vetor de informações médias ou coeficientes de escala $(Cj - 1)$, em cada nível de resolução j. A transformada Haar permite recuperar os valores de C_j a partir dos dados reduzidos. Para realizar uma reconstrução exata dos dados originais, conhecida como transformação inversa, é preciso que entre cada dois níveis consecutivos quaisquer, j e $j - 1$, seja também considerado um vetor D_{j-1}. Esse vetor de informações complementares é denominado de vetor de detalhes ou coeficientes wavelets entre C_j e C_{j-1}.

Com objetivo de compreender o efeito da transformada wavelet em um sinal, a Figura 3.9 apresenta um evento sísmico TR (x(t)) (CANARIO et al., 2020). As outras figuras (V1(t) – V9(t)) ilustram o coeficiente da wavelet variando a escala de resolução de 1 até 9. Conforme o nível de resolução aumenta, mais detalhes são removidos do sinal. Consequentemente, o número de observações também reduz em um fator por 2 (veja a escala do eixo x em todos os gráficos). Por exemplo, comparando V1(t) com o sinal original x(t), notamos que o comportamento geral persiste, mas o número de observações

[7] http://www.h2o.ai

reduz de 4.000 para 2.000. Conforme a resolução é alterada, diferentes observações e informações são consideradas.

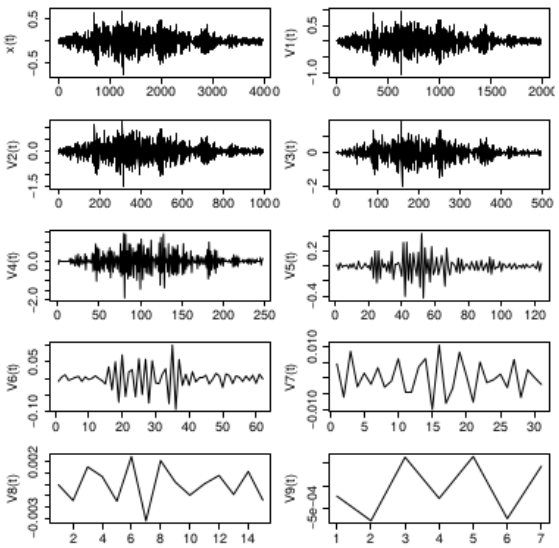

Figura 3.9 $x(t)$ é um sinal TR e $V1(t) - V9(t)$ representa diferentes resoluções da wavelet.

3.3.2 Concept Drift para Internet das Coisas

Os dados produzidos pelos devices da IoT são considerados não estacionários e provenientes de ambientes dinâmicos. Sendo assim, a distribuição desses dados podem mudar ao longo do tempo, fenômeno denominado *concept drift*. O *concept drift* é relacionado a mudanças na distribuição condicional da saída de uma variável alvo, dada a entrada. A ocorrência do *concept drift* geralmente é inesperada e imprevisível, embora em algumas situações do mundo real podemos conhecer a mudança antecipadamente para ambientes específicos (GAMA et al., 2014). A Figura 3.10 apresenta um exemplo de como uma mudança de conceito pode ocorrer em um fluxo de dados.

Formalmente, o *concept drift* entre dois pontos no tempo t_0 e no tempo t_1 pode ser definido da seguinte maneira:

$$\exists X : p_0(X, y) \neq p_1(X, y) \tag{3.2}$$

O p_{t0} é a distribuição coexistente da variável de entrada X e a variável alvo y no tempo t_0. As mudanças nos dados podem ser entendidas como mudanças nos componentes dessa relação. Existem diferentes tipos de mudanças na distribuição dos dados. A mudança pode ser repentina/abrupta, onde ocorre a mudança de um conceito para o outro, ao contrário da incremental que alterna entre vários conceitos intermediários. Outro tipo de mudança é a gradual, cujo conceito dos dados muda lentamente ao decorrer do tempo. O último tipo de *concept drift* é o recorrente, no qual conceitos que foram percebidos anteriormente podem voltar a ocorrer. A seguir, alguns algoritmos de *concept drift* são

brevemente apresentados.

3.3.2.1 Cumulative Sum O CUSUM é uma das técnica de detecção de *concept drift* aplicável nos dispositivos limitados da IoT. Essa técnica pode ser utilizada nos dispositivos da IoT por causa da sua baixa complexidade computacional exigindo pouco recursos para seu processamento. O CUSUM ou Cumulative Sum é um método de análise sequencial baseado no teste de razão de probabilidade sequencial (SPRT) (GAMA et al., 2014). O SPRT é definido da seguinte forma: seja X_1^n uma sequência de dados, onde um subconjunto desses dados seja X_1^w e $1 < w < n$. Seja X_1^n gerada por uma distribuição desconhecida P_0, sendo o subconjunto X_w^n gerado a partir de uma outra distribuição desconhecida P_1. Então, quando ocorrer a mudança na distribuição dos dados de P_0 para P_1 no ponto w, seria esperado que a probabilidade de observar certas subsequências em P_1 seja maior do que em P_0. Sendo assim, pode-se observar que a razão das duas probabilidades não é menor que um limite. Assumindo que as observações X_i são independentes, a estatística para testar a hipótese de que um ponto de mudança ocorreu no tempo w é dada por:

$$T_w^n = \log \frac{P(x_w ... x_n | P1)}{P(x_w ... x_n | P0)} = \sum_{i=w}^{n} \log \frac{P1[x_i]}{P0[x_i]} = T_w^{n-1} + \log \frac{P1[x_n]}{P0[x_n]} \tag{3·3}$$

A mudança é detectada se $T_w^n > L$, onde L é o limite definido pelo usuário. De acordo com Bifet (BIFET; READ, 2018), a técnica CUSUM define o teste para detecção de *concept drift* da seguinte maneira: $g_0 = 0$ e $g_t = \max(0, g_t - 1 + (x_t - \delta))$. A mudança é detectada quando $g_t > h$, após a detecção $g_t = 0$. O x_t é a entrada atual, δ corresponde à magnitude das alterações permitidas, t é tempo atual e h é o limite definido pelo usuário. Para detecção de mudanças negativas, a função utilizada detecta quando o valor de g_t está abaixo do limite negativo. O algoritmo do CUSUM é considerado sem memória e tem como ideia geral identificar a mudança quando a média dos dados de entrada é significativamente diferente de zero.

3.3.2.2 Page-Hinkley O algoritmo Page-Hinkley (PHT) é uma técnica de análise sequencial utilizada para monitorar detecção de mudanças nos dados (PAGE, 1954). Esse algoritmo foi desenvolvido para detectar mudança na média de um sinal Gaussiano (Mouss et al., 2004). O teste aplicado no PHT considera uma variável cumulativa U_T definida como a diferença acumulada entre os valores observados e sua média até o momento atual:

$$U_T = \sum_{t=1}^{T} (x_t - \bar{x}_T - \delta) \tag{3·4}$$

Onde $x_t = 1/T \sum_{t=1}^{t} x_t$ e δ corresponde à magnitude das mudanças que são permitidas. Para detectar crescimento, PHT calcula o valor mínimo de $U_t : m_T = \min(U_t, t = 1...T)$ e será monitorada a diferença entre U_t e $m_T : PH_T = U_t - m_T$. Após isso, é verificado quando PH_T é maior que um determinado limite (λ). Então, uma mudança na distribuição é confirmada. O limite λ configura a taxa admissível de falsos alarmes. Aumentando λ implicará em menos alarmes falsos, mas poderá perder ou atrasar algumas alterações na distribuição dos dados. O controle deste parâmetro de limite de detecção torna possível estabelecer uma compensação entre os alarmes falsos e o atraso nas detecções de mudança de conceito.

3.3.2.3 Exponentially Weighted Moving Average O algoritmo EWMA é aplicado primariamente para detectar um aumento na média de uma sequência de variáveis aleatórias (ROSS et al., 2012). A partir da observação das variáveis aleatórias independentes $X_1, ..., X_n$ que têm uma média comum μ_0 antes do ponto de mudança e μ_1 após a mudança. A variável μ_t é definida para a média no tempo t. Por enquanto, assume-se que tanto μ_0 quanto σ_X, o desvio padrão do fluxo de dados, são conhecidos. O estimador μ_t é definido como:

$$\begin{aligned} Z_0 &= \mu_0 \\ Z_t &= (1-\lambda)Z_t + \lambda X_t, t > 0 \end{aligned} \tag{3.5}$$

Essa formulação do EWMA tem o objetivo de criar uma estimativa recente de μ_t, levando em consideração que os dados mais antigos vão sendo progressivamente reduzidos. O parâmetro λ controla quanto peso é dado aos dados mais recentes em comparação com os dados mais antigos. Diante disso, é possível entender que independente da distribuição das variáveis X_t, a média e o desvio padrão de Z_t são:

$$\begin{aligned} \mu_{Z_t} &= \mu_t \\ \sigma_{Z_t} &= \sqrt{\frac{\lambda}{2-\lambda}(1-(1-\lambda)^{2_t})\sigma_X} \end{aligned} \tag{3.6}$$

Enquanto não é detectada a mudança nos dados, verifica-se que $\mu_t = \mu_t$ e o estimador Z_t irá permanecer em torno deste valor. Quando ocorre uma mudança o valor de μ_t muda para μ_1, Z_t reagirá divergindo de μ_0 em direção a μ_1. Isso pode ser usado para detecção de mudança, sinalizando que uma mudança ocorreu quando:

$$Z_t > \mu_0 + L\sigma_{Z_t} \tag{3.7}$$

Nessa última equação, o parâmetro L é denominado o limite de controle e determina o quando Z_t deve diferenciar de μ_0 para que uma alteração seja sinalizada. O valor de L é normalmente escolhido para garantir que o detector atinja algum nível predefinido de

desempenho.

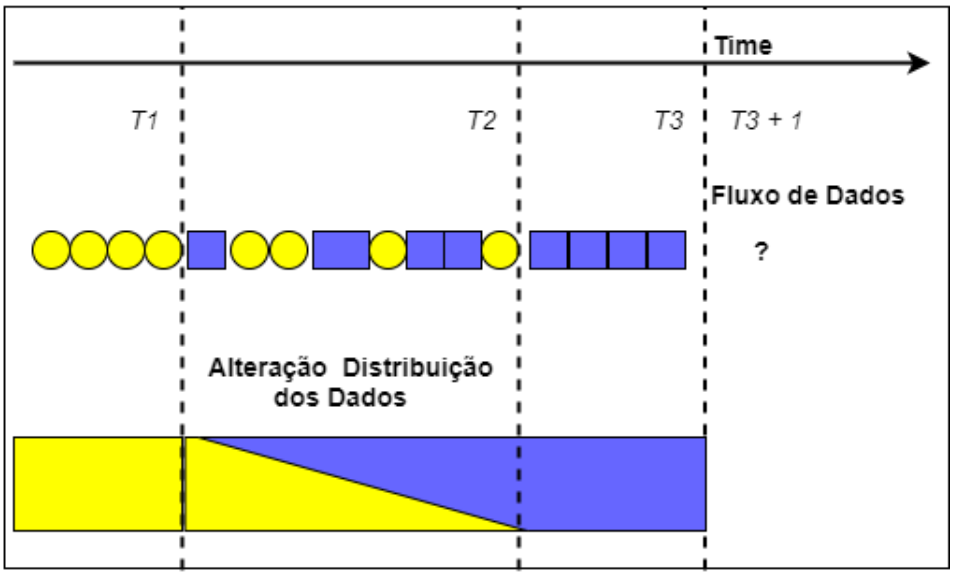

Figura 3.10 Exemplo de *Concept Drift*, adaptado de: (KAMATH; CHOPPELLA, 2017).

A partir dessas técnicas de *concept drift* é possível determinar instâncias representativas de dados para treinamento de uma modelo de predição. Dentre os modelos disponíveis para predição que têm sido amplamente utilizadas estão as Redes Neurais Artificiais. Esses modelos estão sendo aplicados em diversos domínios do conhecimento, assim como na Internet das Coisas. Diante disso, nas próximas seções descreveremos as redes neurais artificiais, mais especificamente uma arquitetura de rede que permite predição em sequência de dados temporais, que é o caso dos dados IoT, conhecida como LSTM.

3.3.3 Redes Neurais Artificiais

Uma Rede Neural Artificial (ANN - *Artificial Neural Network*) é inspirada no sistema nervoso humano. Uma ANN é um modelo matemático orientado a dados que tem a capacidade de resolver problemas através de uma rede de neurônios (HAYKIN, 2005). Uma das principais características de uma ANN é a capacidade de identificar relações complexas não lineares entre a entrada e a saída sem conhecimento prévio da organização dos dados (HAYKIN, 2005). A relação entre a entrada e a saída é especificada da seguinte maneira:

$$Y = f(X^n). \tag{3.8}$$

Sendo X^n o vetor de entrada n-dimensional que consiste em variáveis $X_1, ..., X_i, ..., X_n$, Y é a saída do vetor e $f(.)$ uma função (Equação 3.8).

Uma das configurações mais comuns das ANNs consiste de 3 camadas: entrada, oculta e saída. A camada de entrada e saída são independentes. Entre as camadas de entrada

e saída podem existir uma ou mais camadas ocultas que geralmente são conectadas por matrizes de pesos, viés e funções de ativação.

A principal diferença entre os diferentes tipos de ANNs está relacionada com a arquitetura da rede. A complexidade da arquitetura de uma ANN está relacionada ao número de camadas ocultas, assim como ao número de neurônios em cada camada oculta. O número de camadas ocultas depende das características do conjunto de dados de entrada, como, por exemplo, tipo ou tamanho dos dados. Em geral, não há regra específica para selecionar esses parâmetro, eles podem ser determinados com base no processo de treinamento e avaliados através da otimização de erros.

O treinamento, geralmente, é feito utilizando o algoritmo gradiente descendente e os pesos podem ser atualizados através do algoritmo de retropropagação (backpropagation) com base nos erros calculados a partir da função de custo. Na maioria das arquiteturas de redes, a função de custo da rede é calculada pela diferença entre a saída atual e a predição feita pela rede. O algoritmo de treinamento de retropropagação é um algoritmo de aprendizado supervisionado que permite que a rede atualize os pesos para otimizar a função de custo. O erro atual é propagado para uma camadas anteriores, calculando gradientes da função de custo. Isso significa que os gradientes tendem a se tornar menores à medida que o cálculo continua a retroceder na rede.

Além disso, o modelo tradicional de ANN tem uma limitação na solução de problemas de dados sequenciais, como, por exemplo, problemas de predição em séries temporais ou completar sentenças. A rede neural recorrente é uma classe de ANN introduzida para resolver os problemas acima mencionados.

3.3.4 Rede Neural Recorrente

Rede Neural Recorrente (Recurrent Neural Network - RNN) foi desenvolvida pela primeira vez nos anos 80 (ELMAN, 1990). Sua estrutura consiste em uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. As RNNs têm estruturas semelhantes a cadeias de módulos repetidos com a ideia de usar esses módulos como uma memória para armazenar informações das etapas de processamento anteriores. Ao contrário das redes neurais *feedforward*, as RNNs incluem um *loop* de *feedback* que permite que a rede neural aceite uma sequência de entradas. Isso significa que a saída da etapa $t-1$ é retornada à rede para influenciar o resultado da etapa t e para cada etapa subsequente. Portanto, as RNNs podem ser mais eficazes em aprender sequências.

Durante o processo de treinamento, a RNN utiliza um algoritmo de retropropagação aplicado no cálculo de gradientes e no ajuste de matrizes de pesos na ANN. No entanto, os pesos serão atualizados após a modificação do processo de *feedback*. Esse processo é comumente referido como retropropagação no tempo (*Backpropagation Through Time* - BPTT). O processo BPTT utiliza uma abordagem de retrocesso, camada por camada, a partir da saída final da rede, ajustando os pesos baseado no erro total de saída. Os loops de informações se repetem, resultando em enormes atualizações nos pesos dos modelos da rede e levam a uma rede instável devido ao acúmulo de gradientes de erro durante o processo de atualização. Portanto, o BPTT não é eficiente para aprender uma padrão com dependências a longo prazo (Bengio; Simard; Frasconi, 1994).

3.3.5 Redes Neurais Long Short-Term Memory (LSTM)

A rede neural LSTM é uma evolução da RNN e foi introduzida por Hochreiter e Schmidhuber (1997) para resolver os problemas da RNN mencionadas acima. O LSTM é um tipo especial de RNN, capaz de aprender dependências de longo prazo e lembrar informações por períodos prolongados. O modelo LSTM é organizado na forma de uma estrutura de cadeia. No entanto, o módulo de repetição possui uma estrutura diferente. Em vez de uma única rede neural como uma RNN padrão, ela possui quatro camadas de interação com um método único de comunicação.

Uma rede LSTM típica é composta por blocos de memória chamados células. Dois estados são transferidos para a próxima célula, o estado da célula e o estado oculto. O estado da célula é a principal cadeia de fluxo de dados, o que permite que os dados fluam adiante essencialmente inalterados. No entanto, algumas transformações lineares podem ocorrer. Os dados podem ser adicionados ou removidos do estado da célula através de portas sigmoides. Uma porta é semelhante a uma camada ou a uma série de operações de matriz, que contem diferentes pesos individuais. O LSTM é projetado para evitar o problema de dependência de longo prazo, porque usam portas para controlar o processo de memorização.

A estrutura de uma rede neural LSTM é apresentada na Figura 3.11, já a Figura 3.12 apresenta como ocorre a composição de células na montagem de uma rede LSTM. A primeira parte na construção de uma rede LSTM identifica informações desnecessárias que serão retiradas da célula na primeira etapa. Esse processo é decidido pela função sigmoide. Essa função tem como entrada a saída da última célula LSTM ($H_t - 1$) no tempo $t - 1$. Essa etapa também recebe a entrada atual (X_t) no tempo t. Além disso, a função sigmoide determina qual a parte da saída antiga deve ser eliminada. Esse processo é realizado pelo registro de esquecimento f_t, onde f_t é um vetor com valores variando de 0 até 1, correspondendo a cada número no estado da célula C_{t-1}.

$$f_t = \sigma(W_f[h_{t-1}, X_t] + b_f). \tag{3.9}$$

Na Equação 3.9, σ é a função sigmoide. O W_f e b_f são as matrizes de pesos e vieses (bias), respectivamente, do registro de esquecimento.

A próxima etapa da criação de uma rede LSTM é decidir e armazenar as informações da nova entrada (X_t) no estado da célula, bem como atualizar o estado da célula. Esta etapa contém duas partes, a camada sigmoide e a camada tanh. Primeiramente, a camada sigmoide decide se a nova informação deve ser atualizada ou ignorada (0 ou 1). No próximo passo, a função tanh dá peso aos valores que passaram, decidindo seu nível de importância (-1 a 1). Os dois valores são multiplicados para atualizar o novo estado da célula. Essa nova memória é então adicionada na velha memória C_{t-1}, resultando em C_t.

$$i_t = \sigma(W_i[h_{t-1}, X_t] + b_i), \tag{3.10}$$

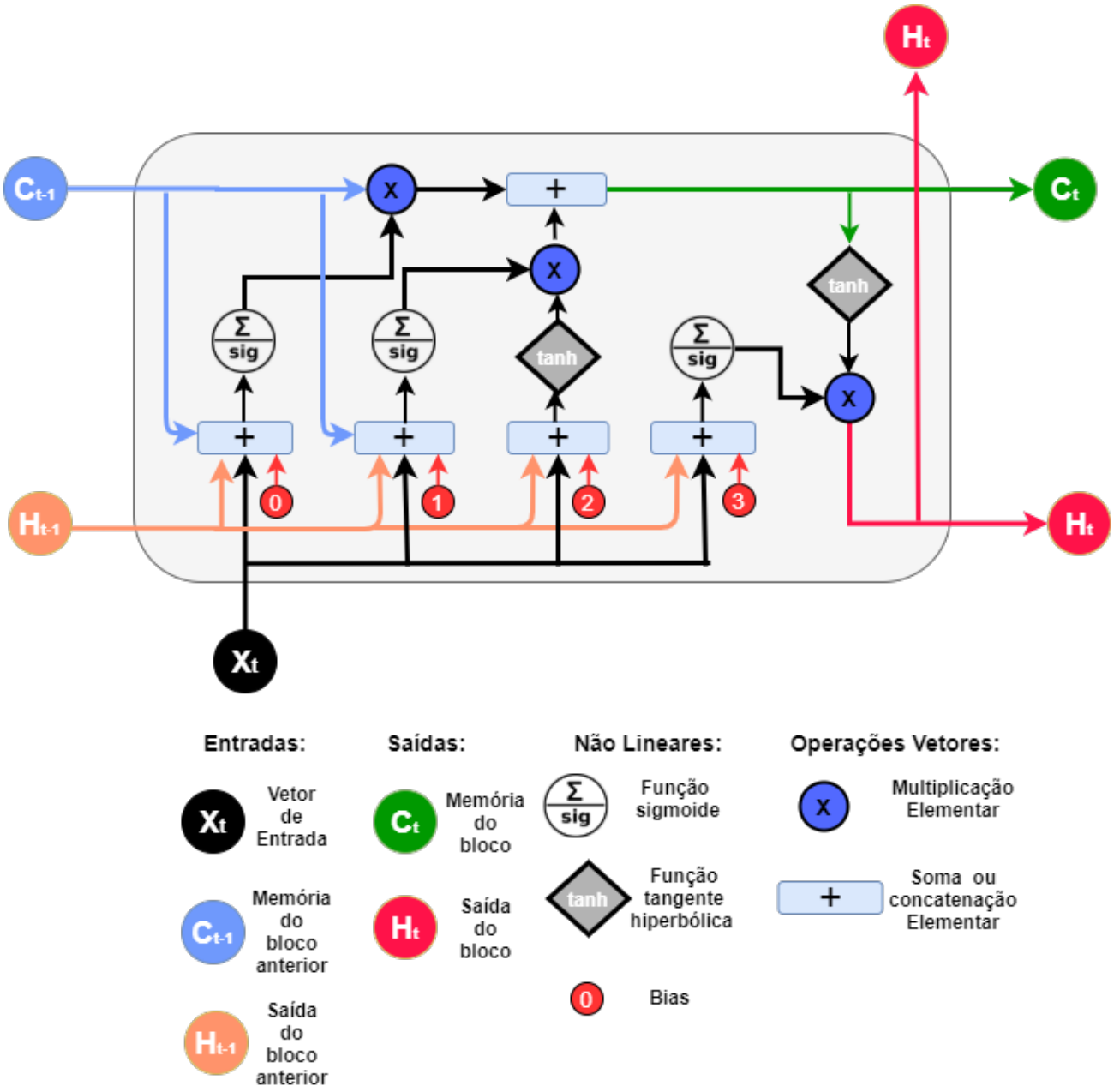

Figura 3.11 Estrutura da rede neural Long Short-Term Memory LSTM.

$$N_t = \tanh(W_n[h_{t-1}, X_t] + b_n), \tag{3.11}$$

$$C_t = C_{t-1} f_t + N_t i_t. \tag{3.12}$$

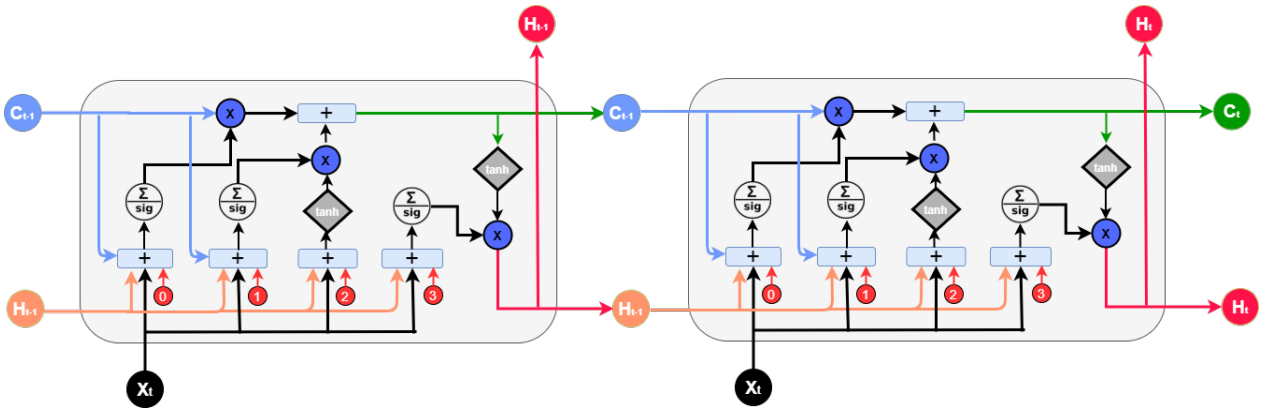

Figura 3.12 Composição da estrutura da rede neural Long Short-Term Memory LSTM.

Nessas Equações 3.10, 3.11 e 3.12, C_{t-1} e C_t são os estados da célula no tempo $t-1$ e t. Enquanto que W e b são as matrizes de pesos e bias, respectivamente, do estado da célula.

Na etapa final, os valores de saída (h_t) são baseados no estado da célula de saída (O_t), porém, é uma versão filtrada. Inicialmente, uma camada sigmoide decide quais partes do estado da célula chegam à saída. Em seguida, a saída da parte sigmoide (O_t) é multiplicada pelos novos valores criados pela camada tanh do estado da célula (C_t), com o valor variando entre -1 e 1.

$$O_t = \sigma(W_0[h_{t-1}, X_t] + b_0), \tag{3.13}$$

$$h_t = O_t \tanh(C_t). \tag{3.14}$$

Nas Equações 3.13 e 3.14, W_0 e b_0 são as matrizes de pesos e bias, respectivamente, do registro de saída.

3.4 CONSIDERAÇÕES FINAIS

A pesquisa abordada neste trabalho está posicionada na utilização de *Fog Computing* com Data Stream Mining para tratar o Big Data de *stream* dos ambientes IoT. Por esse motivo, este Capítulo tem o objetivo de oferecer os principais conceitos, características, padrões e tecnologias necessários para o entendimento deste trabalho, envolvendo os temas: (i) Internet das Coisas; (ii) *Fog Computing* (iii) *Fog of Things* (iv) *Big Data Stream Mining* (v) redes neurais artificiais.

A Seção 3.1 apresenta os principais conceitos e características relacionadas ao termo Internet das Coisas, abordando, entre outras coisas, os requisitos, domínios de aplicações e tecnologias importantes neste trabalho. As Seções 3.2 e 3.2.1 descrevem os aspectos relacionados a *Fog* e o paradigma *Fog of Things*, tratando como essas tecnologias são

utilizadas em conjunto com a Internet das Coisas. Na Seção 3.3 os conceitos e algoritmos aplicados no Big Data Stream Mining são discutidos. Por fim, a Seção 3.3.3 apresenta uma visão conceitual de como as redes neurais artificiais funcionam. No próximo Capítulo será descrita a revisão sistemática da literatura que permitiu levantar os trabalhos que fundamentam esta proposta.

Capítulo

4

PROCESSAMENTO E ANÁLISE DE FLUXO DE DADOS NA COMPUTAÇÃO EM NÉVOA

Este capítulo apresenta uma abordagem implementada para minerar de fluxo de dados produzidos na IoT, tendo como objetivo utilizar toda capacidade computacional dos dispositivos empregados na *Fog Computing* para o processamento e análise de dados, sem a necessidade de utilização constante da *Cloud Computing*. Diante disso, esta proposta visa minimizar o volume de dados que precisam ser encaminhados para a nuvem, melhorando tempo de resposta e custos de transmissão dos dados na rede.

4.1 ARQUITETURA DE FOG PARA PROCESSAMENTO E ANÁLISE DE FLUXO DE DADOS IOT

Plataformas centralizadas na nuvem, em geral, suportam a análise de fluxos de dados, conforme exemplificado na Figura 4.1. Nesse exemplo, o *gateway* IoT constantemente requisita e recebe dados dos dispositivos IoT em curto espaço de tempo. Os dispositivos IoT também podem enviar dados para o *gateway* IoT sem serem requisitados. Além disso, dados podem ser enviados diretamente para nuvem sem a utilização do *gateway* IoT. Como apresentado na Figura 4.1, a cada segundo, novos dados são produzidos pelos sensores e enviados para a nuvem por intermédio do *gateway* IoT. Nessa perspectiva, alguns sensores podem gerar milhares de dados sendo transferidos para nuvem por segundo. Sendo assim, essa arquitetura apresenta problemas de latência e sobrecarga da rede devido ao volume e a distância entre os dispositivos e a nuvem. Além disso, esse tipo de proposta parte do princípio que o *gateway* IoT ou sensor sempre possuirá conectividade com a Internet.

Visando superar esses problemas, a Figura 4.2 apresenta a visão geral da arquitetura proposta, a qual é dividida em três camadas: (i) camada de dispositivos; (ii) camada da borda da rede; (iii) camada da nuvem. A camada de dispositivos é onde os aparelhos tecnológicos que realizam a integração entre o mundo real físico e digital da Internet estão empregados. Nessa camada estão sensores e atuadores como, por exemplo, sensores de

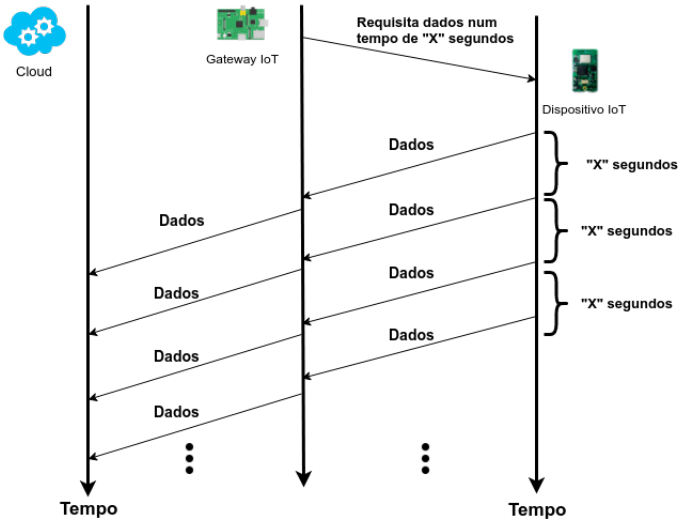

Figura 4.1 Fluxo de dados arquitetura centralizada.

temperatura ou atuadores servo motor para abrir e fechar cortinas. Os dispositivos desta camada são gerenciados pelo *gateway* IoT e, com isso, não necessitam constantemente da conexão com a Internet, pois as trocas de mensagens são realizadas diretamente com o *gateway* da camada da borda. A camada da borda permite que parte do processamento de dados seja realizada através servidores locais e com menor poder de processamento. O processamento que demanda maior poder computacional pode ocorrer na camada da nuvem. A camada da borda também disponibiliza serviços que são executados localmente nos servidores da borda. Neste trabalho, utiliza-se diretamente os seguintes componentes na borda da rede: *gateway* IoT; aplicação de processamento de fluxo de dados; servidor de borda. Na camada da nuvem, ocorre o processamento dos dados centralizados. Nessa camada, pode-se realizar análise de dados em *batch* nos dados devido a maior capacidade de processamento e armazenamento em *Data Centers* mais robustos do que os dispositivos da borda.

Nesta arquitetura, a abordagem proposta (Seção 4.3) atua principalmente na borda da rede para processar e analisar fluxos de dados, reduzindo o tráfego da rede e detectando mudança de comportamento nos dados ao longo do tempo. Com a abordagem proposta, é possível obter informações do comportamento dos sensores à medida que os dados estão sendo produzidos em cada componente. Por exemplo, no cenário de um campus universitário, o *gateway* IoT processa dados de sensores individuais como um sensor de temperatura de uma sala. Já nos servidores da borda, é possível obter o comportamento dos sensores de temperatura de um bloco de aulas. Consequentemente, na nuvem, pode-se saber como estão se comportando os sensores de temperatura de todo o campus. Na arquitetura apresentada neste trabalho, os dados trafegados entre os dispositivos da arquitetura serão apenas aqueles que representam mudança de comportamento do fluxo de dados produzidos pelos sensores.

A Figura 4.3 apresenta a visão geral da proposta como descrito no exemplo anterior. Nesta figura, existem dispositivos distribuídos em ambientes de *Fog* monitorando, por

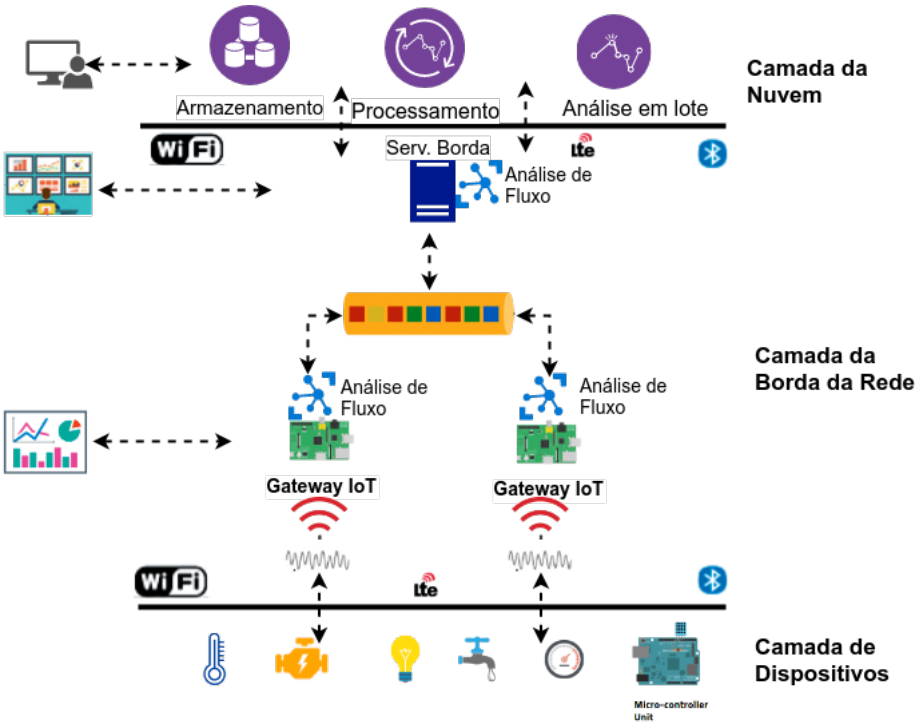

Figura 4.2 Arquitetura da proposta.

exemplo, som, temperatura e luminosidade. Existem quatro ambientes de *Fog* (A - D) projetados para gerenciar o conjunto de dispositivos. Além disso, esses ambientes possuem *gateways* IoT responsáveis por analisar os fluxos de dados produzidos, com o objetivo de detectar desvios de conceito no nível do sensor. Todo *gateway* IoT está conectado aos Servidores de Borda, que também são responsáveis por analisar os fluxos de dados. No entanto, a análise realizada pelos servidores tem uma visão de alto nível, ou seja, eles analisam apenas as saídas produzidas por cada *gateway* IoT. Ao considerar todos os *gateway* de IoT monitorados por um servidor, se um novo comportamento (mudança de conceito) for detectado, o fluxo de dados analisado será enviado para a *Cloud Computing* para (re)modelar todo o ambiente.

Um cenário que pode exemplificar a arquitetura apresentada na Figura 4.3 é o de plataformas de petróleo. Nessas plataformas existe a necessidade de monitoramento constante da quantidade de gás sulfídrico no ambiente, pois, a exposição dos engenheiros a uma grande quantidade desse tipo de gás pode causar danos à saúde. Considerando esse cenário, as *Fogs* A e B da Figura 4.3 podem ser módulos da plataforma de petróleo. Diante disso, os *gateways* IoT gerenciam cada *Fog* individualmente, tendo como objetivo detectar mudanças em tempo real da quantidade de gás em cada módulo. Caso encontre mudanças acima do normal, alertas devem ser emitidos. Apenas mudanças no comportamento dos dados devem transmitidas para o servidor de borda. O servidor de borda fica responsável por encontrar mudanças nos fluxos de dados do monitoramento do gás produzidos pelos *gateways* de cada módulo. No exemplo da Figura 4.3, o servidor de borda A/B é responsável por analisar os dados produzidos pelas *Fogs* A e B. Essa ar-

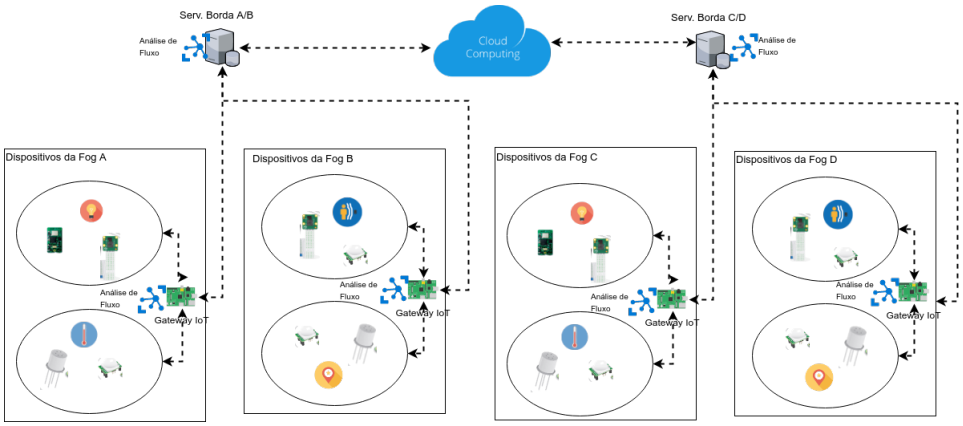

Figura 4.3 Visão geral da arquitetura da proposta.

quitetura implantada no cenário de plataforma de petróleo permitiria o monitoramento constante da quantidade de gás sulfídrico, tanto nos módulos quanto no ambiente mais geral composto por vários módulos. Além disso, levando em consideração plataformas que estão no oceano e possuem conexão de alta latência com a Internet, essa arquitetura permitiria o monitoramento do ambiente produzindo respostas com menos latência e utilizando menos a nuvem.

Complementarmente, a Figura 4.4 apresenta o funcionamento desta proposta numa perspectiva orientada pelo tempo, apresentando a colaboração entre os dispositivos da *Fog* para o processamento de todo fluxo de dados. Por exemplo, o *gateway* IoT requisita dados para os dois sensores: TemperatureSensorA e TemperatureSensorB. Então, esses sensores começam enviar suas leituras para o *gateway* IoT. Na primeira vez, as novas informações são transmitidas da *Edge* até a *Cloud*. À medida que novos dados são recebidos pelo *gateway* IoT, ele analisa se um novo conceito (*concept drift*) foi detectado. Neste exemplo, os dois próximos dados enviados pelo sensor são semelhantes aos primeiros. Então, esses dados não são enviados para as camadas superiores. Finalmente, a quarta temperatura enviada pelos sensores é diferente do modelo atual. Essa diferença é detectada pelo método de desvio de conceito e, então, os novos dados são transmitidos para a próxima camada. Se a camada *Edge* também detectar uma alteração geral causada por esses últimos dados de temperatura, eles também serão enviados para a nuvem. É importante destacar que esta Figura é apenas um exemplo ilustrativo. Além da detecção de mudança, a arquitetura proposta utiliza transformadas de Wavelet para subamostrar os dados produzidos e transmitido entre os dispositivos sem perder informação do comportamento geral. Em seguida, se uma alteração for detectada pelos métodos de desvio de conceito, os dados serão transmitidos entre as camadas (Dispositivos→Camada de Borda→Nuvem).

4.2 IMPLEMENTAÇÃO DA ARQUITETURA NA PLATAFORMA SOFT-IOT

A arquitetura proposta neste trabalho foi implementada a partir da plataforma SOFT-IoT apresentada na Seção 3.2.1. Resumidamente, a plataforma SOFT-IoT é composta por:

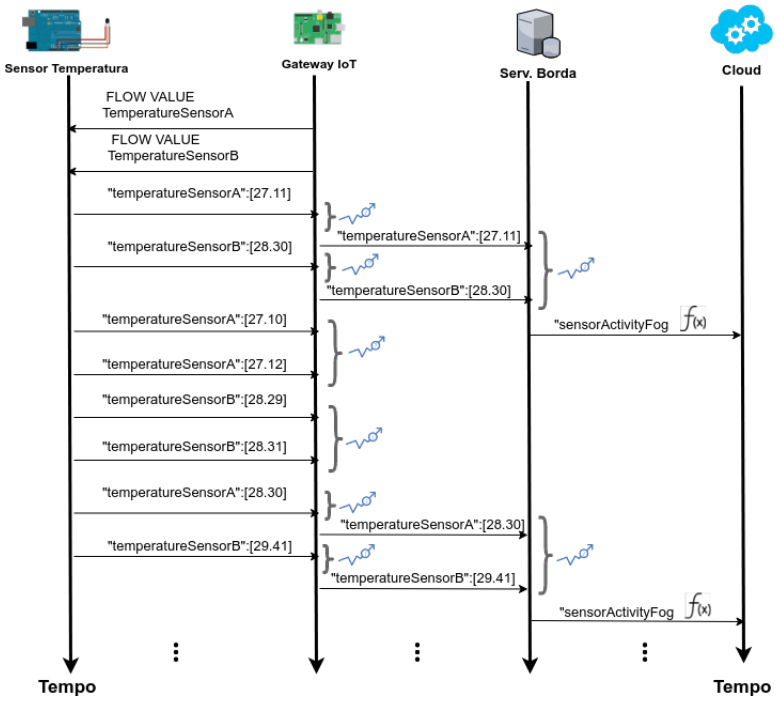

Figura 4.4 Análise de fluxo de dados IoT.

Application; *FoT-Device*; *FoT-Gateway*; *FoT-Server*; *Security Provider*. Este trabalho reutiliza algumas das aplicações definidas na SOFT-IoT, porém, implementa no *FoT-Server* e *FoT-Gateway* o suporte para analises de fluxos de dados, denominando o gateway IoT de FoT-StreamGateway e o servidor de borda FoT-StreamServer. O *FoT-Gateway* é um dispositivo que executa: um servidor MQTT; uma versão reduzida do sistema operacional Linux; uma implementação da especificação OSGI; uma máquina virtual Java.

A Figura 4.5 apresenta os módulos do FoT-StreamGateway implementados (destacados em vermelho) tendo como base o *FoT-Gateway* da SOFT-IoT. A principal alteração está na camada *Application*, a qual agora tem implementações para processamento e análise de fluxo de dados: *Stream Processing*; *Stream Analytics*; Kafka Connector; Edgent. O *Stream Processing* e *Stream Analytics* fornecem implementações para detectar mudança de conceito e reduzir a quantidade de dados enviados pelos dispositivos usando Wavelets. O Edgent[1] é um modelo de programação e um ambiente de execução de código aberto para dispositivos da borda da rede, ele permite processar dados de fluxo contínuos locais em tempo real dos diversos dispositivos da IoT. O Kafka Connector é uma aplicação que permite a conexão do FoT-StreamGateway com a aplicação Kafka utilizada no FoT-StreamGateway. O *Reactive MQTT Broker* é uma implementação reativa de um broker MQTT para IoT proposto por Santana, Alencar e Prazeres (2019). Além de fornecer características tradicionais dos *brokers* IoT em uma arquitetura modular, também permite dimensionar agentes MQTT reativos de acordo com o número de núcleos disponíveis no

[1]http://edgent.apache.org/docs/edgent-getting-started

sistema. Com o objetivo de tornar os resultados apresentados nesta dissertação reproduzíveis, o códigos-fonte da implementação está disponível no repositório do Github[2].

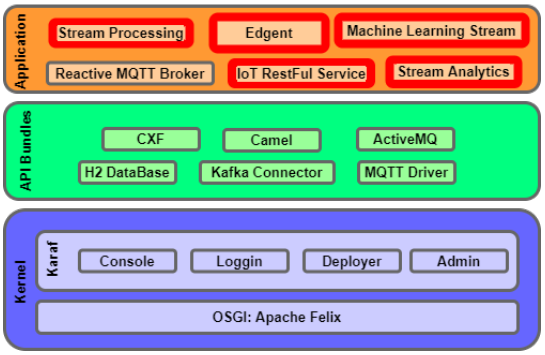

Figura 4.5 Arquitetura FoT-StreamGateway para Fluxo de Dados.

Similarmente, a Figura 4.6 apresenta o FoT-StreamServer implementado a partir do *FoT-Server* de gerenciamento, os módulos desenvolvidos neste trabalho estão destacados em vermelho. O FoT-StreamServer segue a mesma estrutura tecnológica do *FoT-Gateway* e também utiliza a implementação dos módulos *Stream Processing* e *Stream Analytics*. Um outro módulo importante é o *Machine Learning Stream*, o qual implementa técnicas de aprendizado de maquinas para fluxo de dados. Além desses módulos, o Apache Kafka faz parte da pilha de tecnologias. O Apache Kafka[3] é uma plataforma para processamento distribuído de *streaming*, permitindo a publicação e inscrição em fluxos, armazenamento de fluxos de registros duráveis e tolerante a falhas, também pode processar os registros dos fluxos conforme eles chegam. O Kafka se diferencia do Edgent porque é uma implementação mais robusta para trabalhar com *streaming*, por isso é aplicada no servidor de borda e utilizada para processar uma maior quantidade de dados.

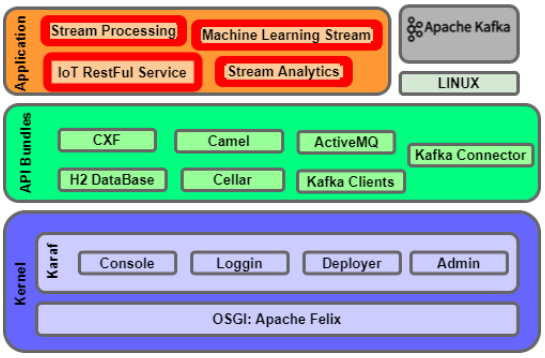

Figura 4.6 Arquitetura FoT-StreamServer para Fluxo de Dados.

[2]https://github.com/WiserUFBA/FoT-StreamGateway-Soft_IoT
[3]https://kafka.apache.org/intro.html

4.3 ABORDAGEM DE ANÁLISE DE FLUXO DE DADOS NA FOG COMPUTING

Através da arquitetura apresentada, uma implementação para tratar o fluxo de dados da IoT na *Fog of Things*, reutilizando os componentes da SOFT-IoT, foi implementada. A abordagem, em linhas gerais, consiste em implementar nos componentes da arquitetura algoritmos que utilizem amostragem de dados e detecção de mudança de conceito em janelas de dados dos sensores IoT. Essa abordagem é apresentada nos Algoritmo 1 e 2. O Algoritmo 1 tem o propósito de tratar dados dos sensores nos FoT-StreamGateways. O Algoritmo 2 trata no servidor de borda (FoT-StreamServer) os dados dos sensores que foram processados pelos FoT-StreamGateways. Os algoritmos desenvolvidos tem como base o protocolo MQTT. Sendo assim, são utilizados os conceitos da arquitetura pub/sub que define tópico como endereço para o qual os dados serão encaminhados. Além disso, é utilizada a função 'on_connect' que define a inicialização no protocolo MQTT e a 'on_message' que é a função executada para todas mensagens recebidas nos tópicos assinados. O Algoritmo 1 modela o fluxo de dados nos FoT-StreamGateways da seguinte maneira:

(i) **linhas 1-2:** função chamada na inicialização do algoritmo para a configuração do cliente MQTT;

(ii) **linhas 3-6:** nessa função é inicializado o cliente MQTT responsável por gerenciar o tópico onde os sensores, que estão conectados ao FoT-StreamGateways, enviam seus dados. Nessa função de configuração o cliente MQTT define duas funções: a que inicializa o tópico que será gerenciado 'on_connect' e a que trata os dados recebidos dos sensores 'on_message';

(iii) **linhas 7-8:** essa função é a que subscreve o tópico em que os dados dos sensores são recebidos pelo FoT-StreamGateways;

(iv) **linhas 9-13:** essa função tem a responsabilidade de tratar as mensagens recebidas pelo FoT-StreamGateways. Inicialmente, ela transforma a mensagem para o formato Json e extrai da mensagem uma lista com os dados e o nome do sensor. Após isso, é chamada a função 'insert_window' para processar a janela de dados do sensor;

(v) **linhas 14-21:** essa função trata da janela de dados dos sensores. Caso o sensor que enviou a mensagem já tenha dados inseridos na janela, os novos dados são armazenados na janela atual. Caso seja a primeira mensagem com dados enviados pelo sensor, uma nova janela vazia é criada. Além disso, é inicializado um detector de *concept drift* para a janela de dados do sensor. Por fim, é marcada como falso a inicialização do *concept drift*, pois essa variável serve para controlar quando é a primeira janela de dados enviada por um sensor;

(vi) **linhas 22-38:** essa função tem como responsabilidade verificar se existem janelas que devem ser processadas levando em consideração o tamanho máximo definido. Caso existam janelas que devem ser processadas, o processamento ocorre da seguinte maneira: **linha 26:** a janela de dados é amostrada a partir de uma função Wavelet;

linhas 29-32: os dados amostrados são processados por um detector de *concept drift*. **linhas 34-37:** Se na janela de dados ocorrer a detecção de *concept drif* ou essa for a primeira janela de dados do sensor, a janela é enviada para o servidor de borda, pelo broker Kafka, permitindo a inicialização do modelo de previsão do servidor. **linha 38:** após o processamento da janela, os dados são descartados e uma nova janela vazia é criada para o sensor.

A partir do primeiro processamento dos dados com Algoritmo 1 no FoT-StreamGateway, o Algoritmo 2 recebe esses dados no servidor de borda. Com isso, o Algoritmo 2 realiza amostragem e detecção de *concept drift*. Além disso, esse algoritmo implementa uma abordagem de previsão que é (re)treinada a partir da detecção de mudança de conceito nas janelas enviadas pelo FoT-StreamGateway. Essa abordagem possibilita prever com precisão os dados dos sensores que estão distribuídos pelo ambiente IoT, pois é natural que a taxa de erro de um modelo de previsão seja alta com a mudança de conceito nos dados. O Algoritmo 2 é responsável por tratar o fluxo de dados nos servidores de borda da seguinte maneira:

(i) **linhas 1-3:** essa é a função de inicialização do algoritmo. Primeiro, é inicializado o cliente que consome o tópico do Kafka onde os FoT-StreamGateways enviam seus dados. Após isso, é chamada a função 'run';

(ii) **linhas 4-11:** nessa função é onde as mensagens com os dados dos sensores são recebidas pelo servidor de borda. Inicialmente, o tópico é subscrito pelo cliente do Kafka. Após isso, na **linhas 7** o algoritmo fica aguardando o recebimento de novas mensagens. Quando uma mensagem chega com erro ela é descartada nas **linhas 8-9**. Caso seja uma mensagem sem erro, ela é transformada em Json e enviada para função 'process_messages';

(iii) **linhas 12-21:** essa função tem a responsabilidade de fazer o primeiro tratamento das mensagens recebidas pelo servidor. Se o FoT-StreamGateway que enviou a mensagem já tiver uma janela sendo gerenciada pelo servidor, os novos dados são inseridos na janela atual. Caso não exista a janela, uma nova janela é criada. Além disso, também é inicializado um detector de *concept drift* e um algoritmo de previsão nas **linhas 18-19**. Após isso, na **linha 21** é executada a função 'check_windows';

(iv) **linhas 22-35:** nessa função as janelas de dados dos sensores enviadas pelos FoT-StreamGateway são analisadas. Inicialmente, a janela de dados é amostrada a partir de uma função Wavelet. Depois disso, nas **linhas 29-32** os dados amostrados são analisados a partir de um detector de *concept drift*. Caso tenha ocorrência de *concept drift* na janela, na **linha 35** é treinado ou atualizado um modelo de previsão a partir dos dados e na **linha 36** a janela de dados é enviada para a aplicação de *Cloud*, caso exista. Não ocorrendo *concept drif* e nem sendo a primeira janela de dados, na **linha 38** é calculado o erro de predição dos dados da janela. Por fim, na **linha 39** uma nova janela vazia é inicializada.

Data: **listMessageKafka**: Lista dos dados do sensor que podem ser enviador para o FoT-StreamServer;
sensoresData: Lista que armazena os dados da janela dos sensores conectados no FoT-StreamServer;
dicDetectors: Lista que armazena os algoritmos para detecção de concept drift nos dados;

```
1  Function main()
2      config_mqtt()
3  Function config_mqtt()
4      Mqttclient.on_connect = on_connect;
5      Mqttclient.on_message = on_message;
6      Mqttclient.connect(brokerMQTT, portBrokerMQTT, keepAliveBrokerMQTT)
7  Function on_connect()
8      Mqttclient.subscribe(topico=dev/)
9  Function on_message()
10     dataRaw = json.loads(msg);
11     value = parser_msg_value(dataRaw);
12     nameSensor = parser_sensor_name(dataRaw);
13     insert_window(nameSensor, value);
14 Function insert_window(sensor, data)
15     if sensor in sensoresData then
16         sensoresData[sensor].append(data)
17     else
18         sensoresData[sensor] = [ ];
19         dicDetectors[sensor] = objConceptDrift.initDetectorDrift();
20         initConceptDrift[sensor] = False
21     check_windows()
22 Function check_windows()
23     foreach indexSensor in sensoresData do
24         if size(sensoresData[indexSensor]) >= threshold=30 then
25             detectChange = False;
26             dataWavelet = objWavelet.pyramid(sensoresData[indexSensor], level=1);
27             listMessageKafka = [ ];
28             sendMessaKafka = False;
29             foreach data in dataWavelet do
30             listMessageKafka.append(data);
31             warning_status, detectChange = dicDetectors[indexSensor].run(data);
32             if detectChange == True then
33                 sendMessaKafka = True;
34             if sendMessaKafka == True or initConceptDrift[indexSensor] == False
                then
35                 initConceptDrift[indexSensor] = True;
36                 json_message = {gatewayID:args.name, nameSensor:indexSensor,
                    typeSensor:type_Sensor, values:listMessageKafka};
37                 producerKafka.send(topic=dev, json_message);
38             sensoresData[indexSensor] = [ ];
```

Algoritmo 1: Algoritmo do FoT-StreamGateway para Fluxo de Dados da Névoa das Coisas.

Data: **gatewaySensoresData**: Lista de armazenamento dos dados recebidos dos *FoT-Gateways* gerenciados pelo *FoT-Server*;
dicDetectors: Lista que armazena os algoritmos para detecção de concept drift nos dados;
modelPrediction: Lista que armazena os modelos para previsões nos dados;

```
1  Function main()
2      kafka_consumer = Consumer(conf);
3      run();
4  Function run()
5      while True do
6          kafka_consumer.subscribe(topic=dev);
7          message = kafka_consumer.poll();
8          if message is None or message.error() then
9              continue;
10         jsonData = json.loads(message.value());
11         process_messages(jsonData);
12 Function process_messages(jsonData)
13     if jsonData[gatewayID] in gatewaySensoresData then
14         foreach value in jsonData[values] do
15             gatewaySensoresData[jsonData[gatewayID]].[jsonData[typeSensor]].append(value)
16     else
17         gatewaySensoresData[jsonData[gatewayID]] = [ ];
18         dicDetectors[jsonData[gatewayID]] = objConceptDrift.initDetectorDrift();
19         modelPrediction[jsonData[gatewayID]] =
            series.modelPredictions(jsonData[gatewayID]);
20         initModel[jsonData[gatewayID]] = False;
21     check_windows()
22 Function check_windows()
23     foreach indexGateway in gatewaySensoresData do
24         if size(gatewaySensoresData[indexGateway]) >= threshold=50 then
25             dataWavelet = objWavelet.pyramid(gatewaySensoresData[indexGateway],
                level);
26             listMessage = [ ];
27             trainModel = False;
28             detectChange = False;
29             foreach data in dataWavelet do
30                 warning_status, detectChange = dicDetectors[indexGateway].run(data);
31                 if detectChange == True then
32                     trainModel = True;
33             if (trainModel == True or initModel[indexGateway] == False) then
34                 initModel[indexGateway] = True;
35                 modelPrediction[indexGateway].train_model(dataWavelet);
36                 sendMessageCloud(dataWavelet)
37             else
38                 modelPrediction[indexGateway].calc_error(dataWavelet);
39             gatewaySensoresData[indexGateway] = [ ];
```

Algoritmo 2: Algoritmo do FoT-StreamServer para Fluxo de Dados da Névoa das Coisas.

Entre os componentes da proposta são trocadas mensagens. No Código 4.1 é apresentado exemplos de como são estruturadas essas mensagens. As mensagens dos sensores são formatadas de acordo com o protocolo TATU. No exemplo do Código 4.1, o sensor é de temperatura e está utilizando o método FLOW que define tempo de coleta dos dados (collect) e de publicação (publish) dos dados no FoT-StreamGateway. Outro tipo de mensagem utilizada é a que o FoT-StreamGateway envia para o servidor de borda. Nessa mensagem são enviadas informações sobre o sensor, por exemplo, nome e tipo do sensor. Além disso, são enviados os dados da janela que apresentam *concept drift*. Por fim, o FoT-StreamServer, caso exista aplicação na *Cloud*, envia para *Cloud* dados dos sensores e informações do FoT-StreamGateway, por exemplo, nome do FoT-StreamGateway e tipo do sensor. Além disso, pode ser enviado para *Cloud* informações sobre as métricas de avaliação da predição feita pelo algoritmo utilizado, bem como os valores atuais e os valores preditos da janela. Apesar dessa definição de troca de mensagens, as informações que devem fazer parte das mensagens podem ser alteradas de acordo com a necessidade do ambiente que a proposta seja aplicada.

```
//Exemplo de mensagens do sensores enviadas para os gateways
{"CODE":"POST","METHOD":"FLOW","HEADER":{"NAME":"ufbaino08"},
"BODY":{"temperature":["19.2338", "19.2436", "19.2534",
↪  "19.2632"],"FLOW":{"publish":4000,"collect":1000}}}
//Exemplo de mensagens do gateway enviadas para o servidor de borda
   {"sensor_name": "ufbaino08", "typeSensor": "temperature", "data":
    ↪  [19.2338, 19.2436, 19.2534, 19.2632, 19.2828, 19.3024, 19.322,
    ↪  19.3612, 19.3612, 19.371, 19.3808, 19.4004, 19.4004, 19.4592,
    ↪  19.4788, 19.4788, 19.4984, 19.4984, 19.4984, 19.518, 19.5376,
    ↪  19.5376, 19.5474, 19.5768, 19.616, 19.616, 19.6258, 19.6552,
    ↪  19.6748, 19.6748], "size": 352}
//Exemplo de mensagens do servidor de borda
{"name_gateway": "h1", "typeSensor": "temperature", "rmse":
↪  0.30322635899912215, "mae": 0.2624377647073324, "mse":
↪  0.0919462247918645, "raw_values": [17.72, 17.7, 17.7, 17.71,
↪  18.55, 19.34, 19.31, 19.31, 19.26, 19.23, 19.21, 19.2],
↪  "predictions": [17.873683109879494, 17.894994464814662,
↪  17.875971026271582, 17.88817790314555, 17.89628034606576,
↪  18.82590137332678, 19.44347089856863, 19.129150803014635,
↪  19.072905498966573, 18.988083418495954, 18.947033732421694,
↪  18.92768905390054]}
```

Código 4.1 Exemplos das mensagens trocadas entre os componentes da proposta.

4.4 CONSIDERAÇÕES FINAIS

Este trabalho propõe a utilização de uma abordagem de *Fog* combinada com a mineração de fluxo de dados para tratar o cenário do Big Data Stream da Internet das Coisas. Por conta disso, neste Capítulo foram apresentados os detalhes da modelagem e implementação das soluções propostas. Na Seção 4.1 é apresentada a arquitetura de como abordagem é estrutura, assim como na Seção 4.2 é descrita a implementação tecnológica dos componentes da arquitetura. Por fim, na Seção 4.3 é descrita a abordagem aplicada nos componentes em formato de algoritmos. No próximo Capítulo o planejamento, execução e análise dos resultados dos experimentos é descrito.

Capítulo

5

PLANEJAMENTO DOS EXPERIMENTOS E AVALIAÇÃO DOS RESULTADOS

As aplicações desenvolvidas para suportar o grande fluxo de dados da Internet das Coisas têm como requisitos principais tratar as variáveis throughput e tempo de atraso na análise dos dados. Por outro lado, as infraestruturas que suportam as aplicações IoT necessitam tratar do desafio da grande quantidade de dados trafegados na rede. Para isso, este Capítulo apresenta os experimentos para avaliar a proposta deste trabalho seguindo esses requisitos. Os experimentos neste Capítulo consistem em duas etapas: experimento com sensores reais IoT e experimento em ambiente simulado da IoT. Na primeira etapa, analisou-se o funcionamento da abordagem a partir de ambiente real da IoT. Na segunda etapa, experimentos foram executados em um simulador de rede para avaliar a abordagem com maior produção de dados e replicação dos componentes da proposta.

5.1 EXPERIMENTOS CENÁRIO REAL DA IOT

Neste experimento, a proposta deste trabalho foi avaliada no gerenciamento de sensores em um cenário de edifício inteligente (*Smart Building*). Um *Smart Building* permite que todos os sistemas (e.g medidores de água, bombas, alarmes de incêndio, medidores de energia, iluminação) dentro dele se comuniquem entre si. Esses sistemas são compostos por sensores que desempenham um papel importante na coleta de dados para permitir as decisões no ambiente. Sendo assim, os dados lidos dos sensores devem ser analisados pelos sistemas constantemente e em tempo real. Esse monitoramento contínuo permite ajustes automatizados que podem controlar as condições em todo um edifício. *Smart Buildings* geram um grande volume de dados sobre seu próprio uso, algo que os prédios comuns simplesmente não fazem.

A Figura 5.1 apresenta a arquitetura do experimento realizado. Utilizamos a camada da borda (Gateway IoT) e a camada de nuvem (Servidor de Nuvem). Para este experimento utiliza-se os seguintes dispositivos: sensores IoT; Gateway IoT e plataforma de processamento de fluxo de dados. Neste experimento não foi utilizado o servidor de borda.

Como dispositivo IoT, foi utilizado o Sonoff SC[1] para monitoramento do edifício. Esse dispositivo permite o monitoramento das seguintes variáveis: temperatura, umidade, luminosidade, qualidade do ar e intensidade de sons. O *gateway* IoT FoT-StreamGateway foi utilizado na borda de rede para processamento e análise do fluxo de dados. Para gerenciamento do fluxo de dados na nuvem o Servidor de Nuvem foi implementado em Python. Para aplicação de processamento de fluxo de dados foi utilizado o Kafka, pois é uma plataforma para fluxo de dados mais robusta comparado com o Edgent utilizado na borda da rede. O *gateway* IoT foi implantando no Apache ServiceMix 7.0.0 com Java 1.8, o hardware utilizado foi o Raspberry Pi 3 Model B+ 64bits ARM Quad-Core com clock 1.4 GHz e memória RAM de 1GB. O Servidor de Nuvem e o Kafka foram implantados em um servidor Amazon EC2 (Intel Xeon com CPU de 2.40GHz e 1GB de memória RAM) com servidor Linux/Ubuntu.

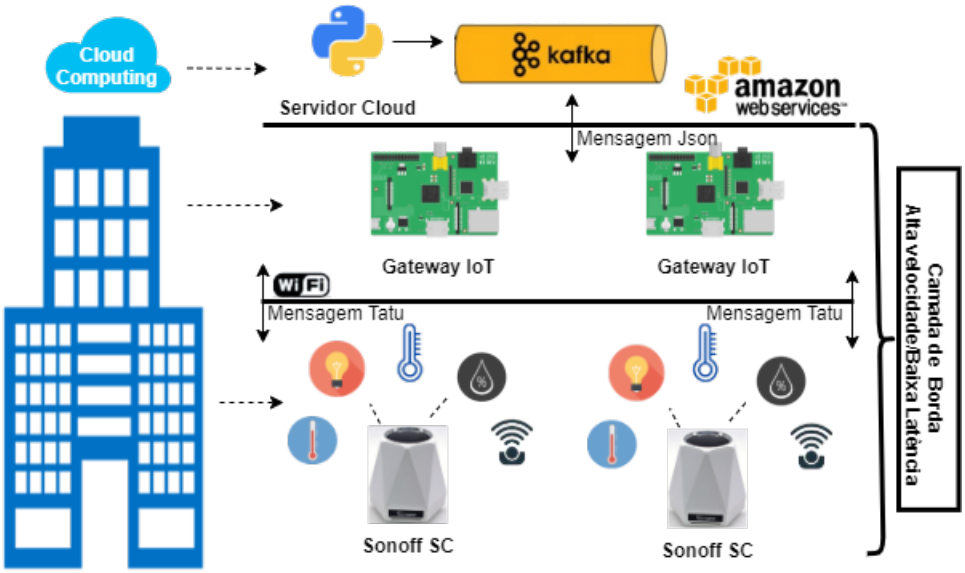

Figura 5.1 Modelagem do Experimento.

A abordagem implementada no FoT-StreamGateway para análise de fluxo de dados é a apresentada no Algoritmo 1. Inicialmente são criadas janelas ajustáveis pela quantidade de dados recebidos dos sensores. Neste experimento, foram definidas janelas de 20 leituras. A partir desses dados, a transformada discreta wavelet Haar foi aplicada com a resolução de 1 nível. Depois de aplicar a transformada discreta, utilizou-se o processamento dos dados com algoritmo CUSUM para identificação de *concept drift*. Em testes preliminares, verificou-se que as mudanças de conceito no fluxo de dados dos sensores foram detectados com um *threshold* de 50. Então, esse valor foi definido como padrão para este experimento. Por fim, caso alguma mudança de conceito seja detectada, a janela de dados é enviada para o Apache Kafka na *Cloud* e inserida no banco de dados H2[2] na *Fog*. Caso não tenha mudança de conceito identificada, a janela é descartada. Se no momento de enviar a

[1]https://www.itead.cc/sonoff-sc.html
[2]https://www.h2database.com/html/main.html

janela algum erro ocorrer, essa janela também é inserida no H2 para posteriormente ser enviada para *Cloud*.

Os resultados deste experimento foram medidos com execuções de 12 horas de funcionamento seguindo a arquitetura da Figura 5.1. Os Sonoffs foram configurados através do protocolo TATU, o tempo de coleta dos dados foi de 8 segundos e o tempo de envio foi de 9 segundos. Neste experimento, 2 Sonoffs foram configurados com 5 sensores e 10 sensores cada. Para uma maior precisão na avaliação das técnicas utilizadas, os seguintes experimentos foram executados: somente com detecção de *concept drift* e combinando a transformada Wavelet com a detecção de *concept drift*.

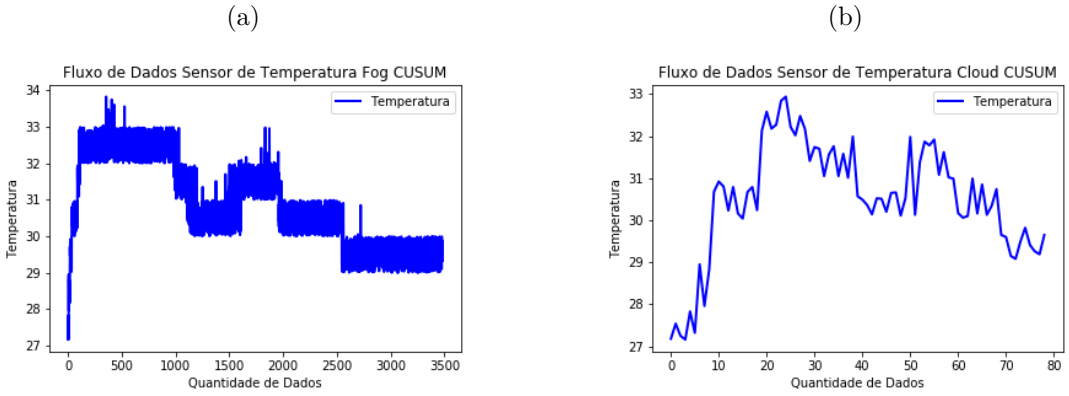

Figura 5.2 Fluxo de dados de temperatura na Fog e Cloud analisados pelo CUSUM.

A Figura 5.2 apresenta os resultados da execução do experimento para o sensor de temperatura utilizando CUSUM. A Figura 5.2 (a) mostra o comportamento dos valores de temperatura na *Fog*. Nessa execução, uma quantidade de 3500 dados foram lidos. Na Figura 5.2 (b), tem-se os valores que foram enviados da *Fog* para *Cloud*. Na Figura (b), é possível visualizar como foi a variação da temperatura ao longo da execução do experimento. Porém, comparando com a Figura (a) da *Fog*, pode-se uma quantidade menor de dados trafegados, com 3500 na *Fog* e 80 leituras na *Cloud*.

A Figura 5.3 apresenta o experimento utilizando a combinação da Wavelet Haar com CUSUM. A vantagem de usar essa combinação é notada comparando as Figuras 5.3 (a) e 5.2 (a) que mostram a redução dos dados analisados de 3500 para 2000. A Figura 5.3 (b) apresenta que, na *Cloud*, depois de usar o método de *concept drift*, a quantidade de dados transmitidos foi ainda menor (cerca de 70). O resultado mais importante em ambos os experimentos é a redução dos dados transmitidos e analisados sem perder o comportamento principal do sistema, como pode ser percebido ao visualizar o comportamento dos dados nos dois gráficos mais à direita.

Os experimentos descritos anteriormente foram repetidos para os 5 tipos de sensores disponíveis no Sonoff SC. As Figuras 5.4, 5.6, 5.8 e 5.10 apresentam os resultados do experimento utilizando apenas CUSUM. A partir dessas figuras é possível ter a mesma conclusão do experimento com o sensor de temperatura descrito anteriormente, podemos visualizar como foi a variação dos dados ao longo da execução do experimento. Porém, comparando as Figuras (a) e (b) temos uma quantidade menor de dados trafegados da

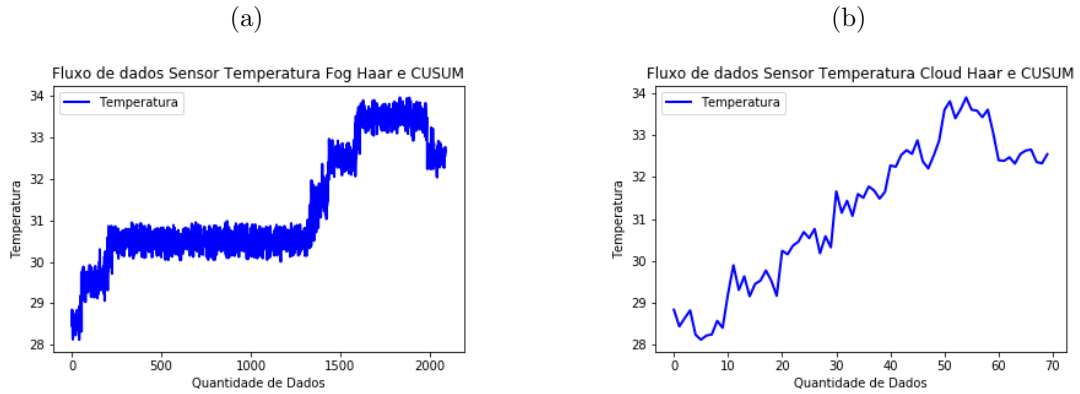

Figura 5.3 Fluxo de dados Temperatura na Fog e Cloud. Experimento realizado utilizando Haar and CUSUM.

Fog para *Cloud*. As Figuras 5.5, 5.7, 5.9 e 5.11 apresentam o resultado do experimento utilizando a combinação da Wavelet Haar com CUSUM. Como explicado anteriormente, os resultado representados nas figuras mostram que existe a redução de dados analisados na *Fog* comparado com os resultados da utilização somente do CUSUM. Além disso, existe a redução dos dados enviados da *Fog* para *Cloud*.

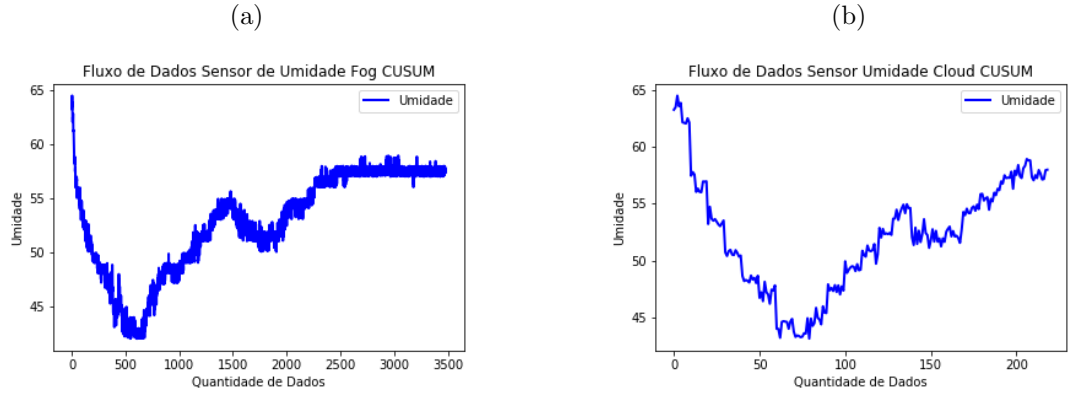

Figura 5.4 Fluxo de dados de umidade na Fog e Cloud analisados pelo CUSUM.

A Figura 5.12 mostra a quantidade de kilobytes processados e trafegados na rede durante a execução do experimento para os cinco sensores do Sonoff, sendo utilizada somente a técnica CUSUM. Pode-se observar que para cada sensor, foram utilizadas a quantidade de dados por volta de 70 KB na *Fog*. Entretanto, a quantidade de dados enviados para *Cloud* é reduzida, por volta de 20 KB (máximo) para o sensor de poeira e próximo de 5 KB (mínimo) para o sensor temperatura.

A Figura 5.13 também apresenta a quantidade de kilobytes processados e trafegados na rede para os cinco sensores, porém, utilizando a Wavelet Haar com CUSUM. Nessa combinação, é possível observar a redução nos dados tanto na *Fog* quanto na *Cloud*. Por exemplo, a quantidade de dados na *Fog* ficou em torno de 40 KB. No teste somente com CUSUM (Figura 5.12), a quantidade de dados na *Fog* é de 70 KB. Na *Cloud* o sensor de

(a) (b)

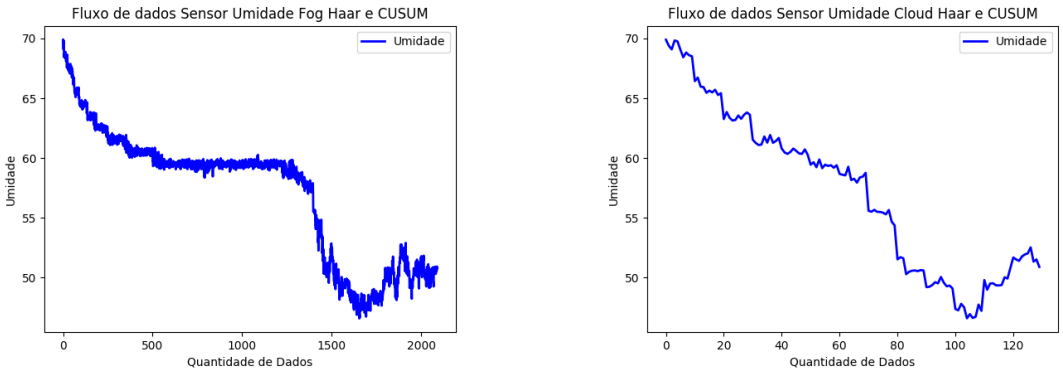

Figura 5.5 Fluxo de dados Umidade na Fog e Cloud. Experimento realizado utilizando Haar and CUSUM.

(a) (b)

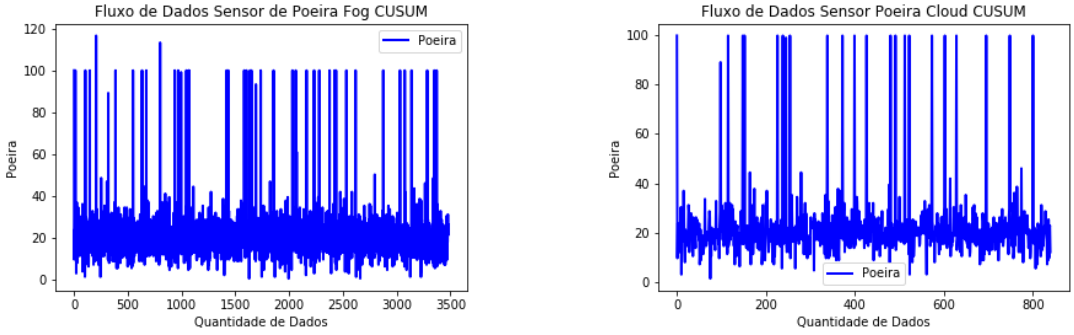

Figura 5.6 Fluxo de dados de poeira na Fog e Cloud analisados pelo CUSUM.

(a) (b)

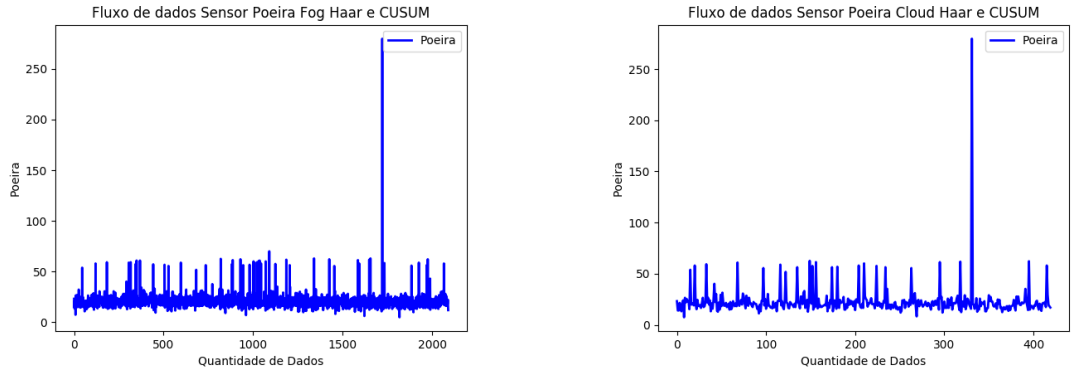

Figura 5.7 Fluxo de dados Poeira na Fog e Cloud. Experimento realizado utilizando Haar and CUSUM.

(a) (b)

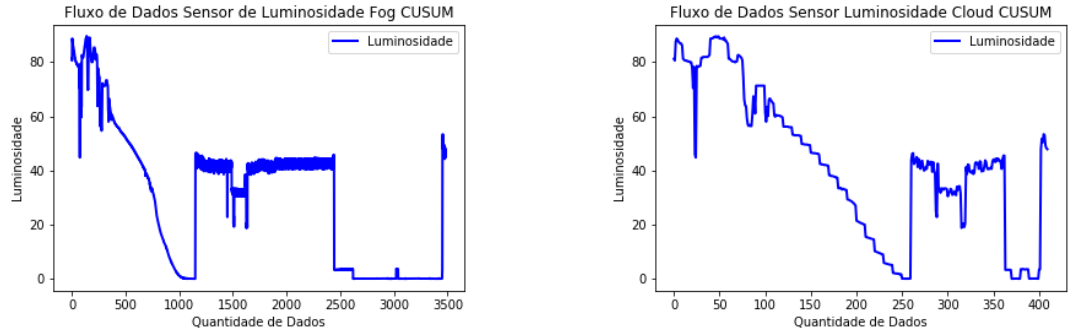

Figura 5.8 Fluxo de dados de luminosidade na Fog e Cloud analisados pelo CUSUM.

(a) (b)

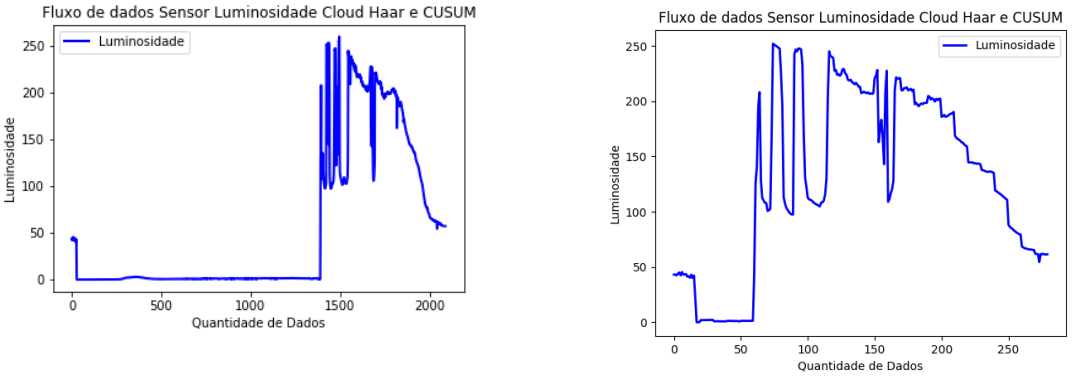

Figura 5.9 Fluxo de dados Luminosidade na Fog e Cloud. Experimento realizado utilizando Haar and CUSUM.

(a) (b)

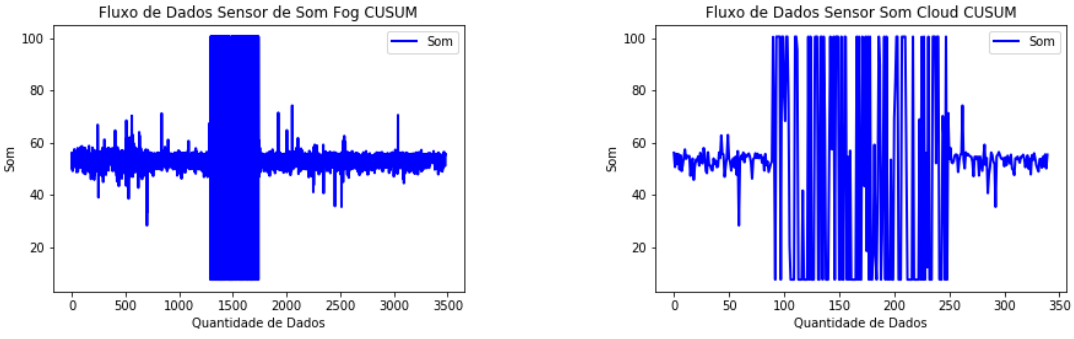

Figura 5.10 Fluxo de dados de som na Fog e Cloud analisados pelo CUSUM.

poeira ficou com menos de 10 KB (máximo) recebidos, também houve uma redução nos dados do sensor de umidade (5 KB) e som (5 KB).

As Figuras 5.14 e 5.15 apresentam o *throughput* por minuto das mensagens recebidas

(a) (b)

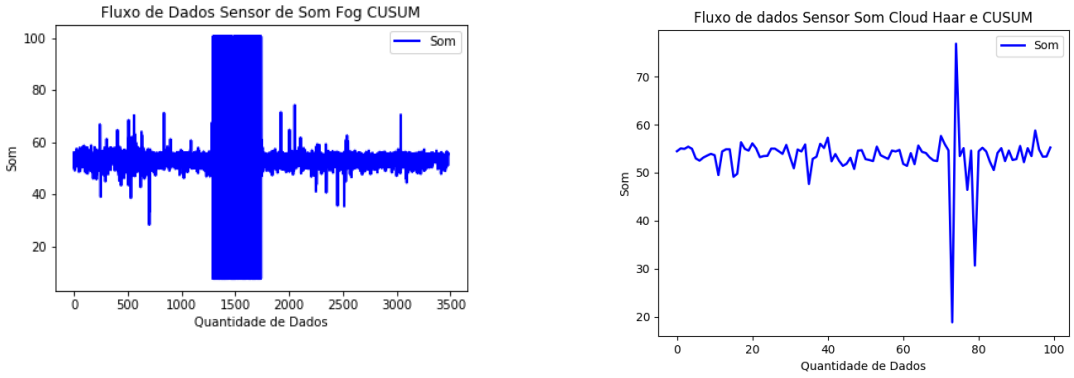

Figura 5.11 Fluxo de dados Som na Fog e Cloud. Experimento realizado utilizando Haar and CUSUM.

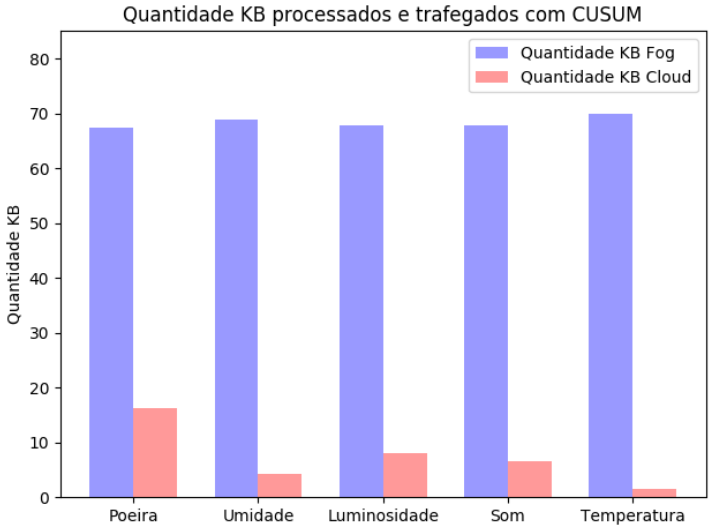

Figura 5.12 Dados trafegados entre a Fog e a Cloud. Experimento realizado utilizando somente CUSUM.

por sensor. O *throughput* por minuto leva em consideração o tempo de envio de mensagens que os sensores foram configurados. Neste experimento, o tempo de envio foi a cada 9 segundos. Na Figura 5.14, foram processados dados de 5 sensores e a Figura 5.15 apresenta dados de 10 sensores. De acordo com os resultados, pode-se notar que a taxa de mensagens bem-sucedidas entregues na rede permaneceu estável após a adição de mais sensores. O objetivo ao apresentar esses números é mostrar que as análises de fluxo de dados não estão afetando a quantidade de mensagens que podem ser processadas pela implementação atual do *gateway*.

A conclusão anterior sobre o *throughput* é ratificada analisando tempo de processamento gasto pelas técnicas utilizadas. A Figura 5.16 apresenta o resultado do *delay* de processamento somente da transformada Wavelet e CUSUM para os dados de 5 sensores

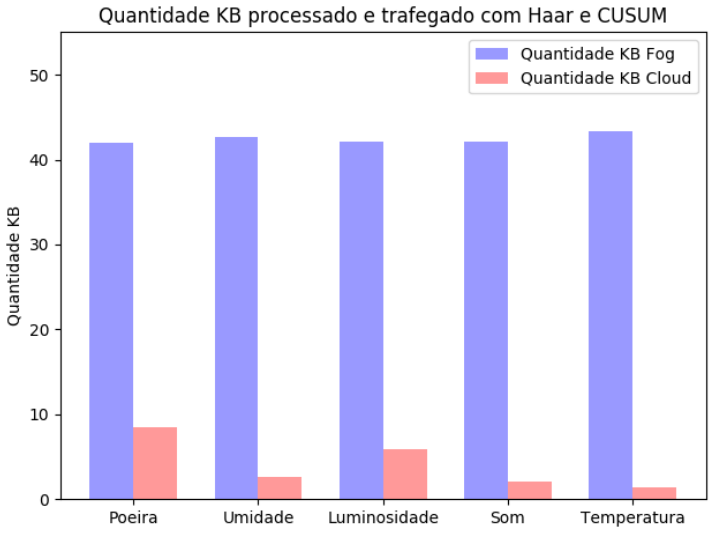

Figura 5.13 Dados trafegados entre a Fog e a Cloud. Experimento realizado utilizando Haar e CUSUM.

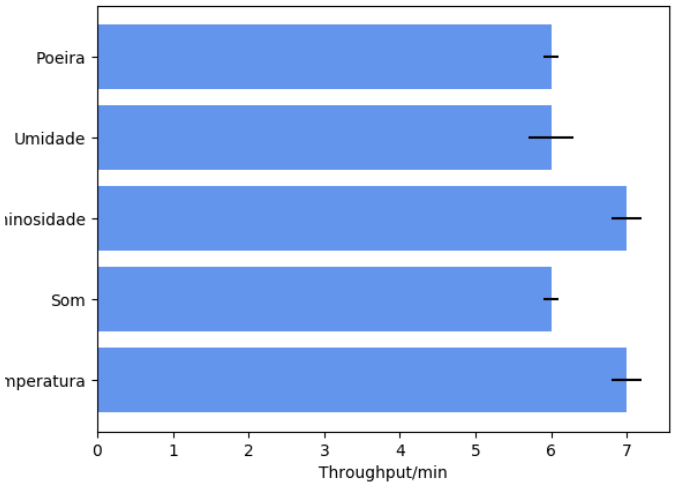

Figura 5.14 Throughput do gateway com 5 sensores.

na *Fog*. A quantidade de dados das janelas são 20 instâncias. Os resultados da Figura 5.16 enfatizam, para cada tipo de sensor, que o tempo médio de execução gasto (em milissegundos) para aplicar Haar e CUSUM nas janelas do fluxo de dados não representa um alto custo para o *gateway*.

Na Figura 5.17, apresenta-se o *delay* das janelas processadas. O início do processamento é feito nos dispositivos da *Fog* até os dados serem processados na *Cloud*. O *delay* das janelas é o tempo médio de processamento da abordagem utilizando Haar e CUSUM. Esse tempo leva em consideração o tempo de chegada entre o primeiro e o último dado

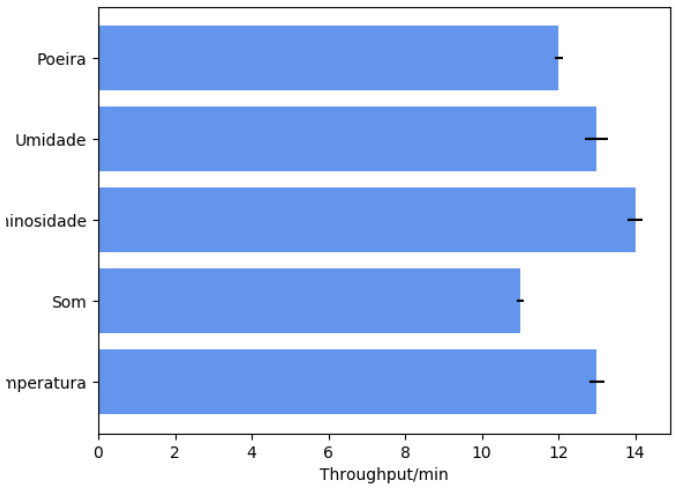

Figura 5.15 Throughput do gateway com 10 sensores.

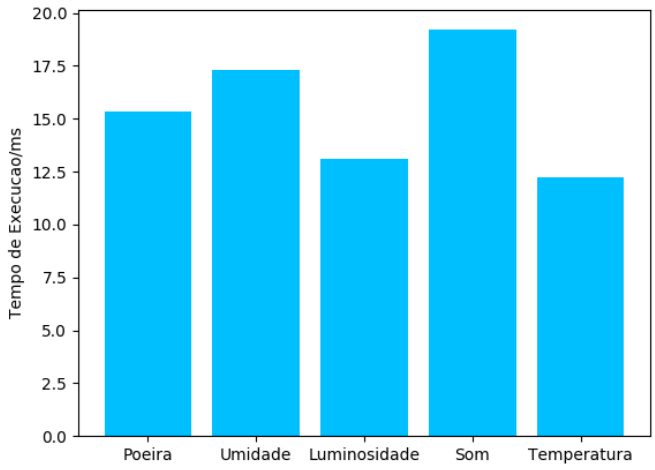

Figura 5.16 Tempo médio de processamento calculado para Haar e CUSUM no gateway.

coletado em uma janela, mais o tempo gasto para processar esse fluxo de dados. Diante dessas analises, é possível confirmar que o atraso de processamento é influenciado pelo processo de processamento dos dados pela abordagem. Apesar disso, concluímos que essa proposta apresentou os seguintes resultados: i) reduziu a quantidade de dados transmitidos na rede; ii) reduziu processamento em camadas superiores (como a *Cloud*), uma vez que todo o processamento é distribuído em vários dispositivos espalhados no ambiente da IoT.

Para avaliar a proposta com outras métricas foi desenvolvido um estudo experimental emulado em um cenário mais robusto. Esse cenário apresenta a necessidade de aplicação

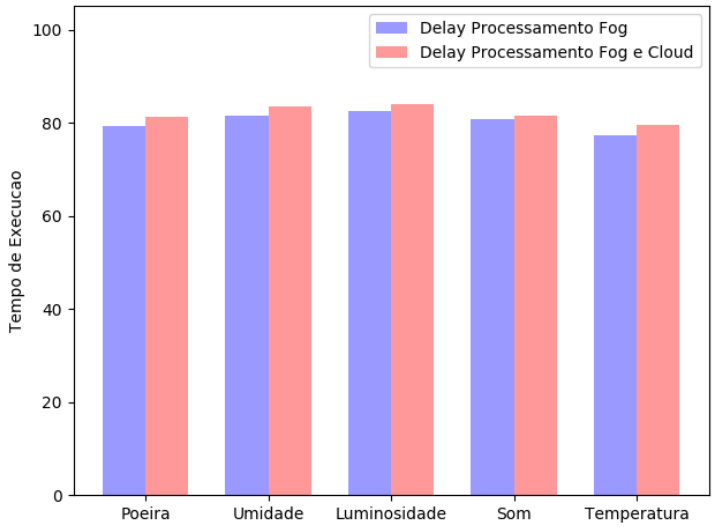

Figura 5.17 Tempo médio de processamento para janela de dados processados com Haar e CUSUM no gateway.

de toda arquitetura proposta. Além de uma análise visual dos dados transmitos, o próximo experimento permite avaliar o impacto da detecção de *concept drift* em cascata para o treinamento online de técnicas de Aprendizagem de Máquina. Por fim, foi desenvolvida uma emulação da plataforma para permitir a execução de experimentos com os todos os componentes da proposta. Nas próximas seções são mostradas uma descrição com mais detalhes desse experimento.

5.2 EXPERIMENTOS CENÁRIO EMULADO DA IOT

O experimento emulado foi executado com dados reais do *dataset* disponibilizado pelo Intel Berkeley Research Lab disponível em: http://db.lcs.mit.edu/labdata/labdata.html. O *dataset* é composto por dados de 54 sensores Mica2Dot com placas meteorológicas. Cada sensor coletou informações das seguintes variáveis: umidade, temperatura, luz, tensão e *timestamp* a cada 31 segundos. Cada Mica2Dot possui um identificador associado (i.e.moteid), as Figuras 5.18 e 5.19 apresentam como foram distribuídos os sensores dentro do laboratório Intel Berkeley Research.

Os arquivos do *dataset* incluem dados de cerca de 2,3 milhões de leituras coletadas dos Mica2Dots. Os dados coletados de temperatura foram medidos em graus Celsius e a umidade relativa foi corrigida pela temperatura, variando de 0 a 100%. A luminosidade está em Lux e a tensão é expressa em volts variando entre 2 a 3.

As Figuras 5.18 e 5.19 também apresentam quais sensores foram utilizados para execução dos experimentos. A primeira configuração foi modelada como apresentado na Figura 5.18. Nessa configuração foram utilizados os dados dos sensores de temperatura que estão fisicamente próximos para serem gerenciados por um *gateway*. O Gateway 1 do experimento é responsável por tratar os dados dos sensores: 1, 2, 3, 35, 33. O Gateway 2 do experimento é responsável por gerenciar os sensores: 7, 6, 8, 9, 10.

A Figura 5.19 apresenta o segundo tipo de configuração para os experimentos. A segunda configuração tem uma quantidade maior de sensores e *gateways*. O Gateway 1 do segundo experimento é responsável por gerenciar os sensores: 1, 2, 3, 35, 33. O Gateway 2 gerencia os sensores: 7, 6, 8, 9, 10, enquanto que o Gateway 3 é responsável pelos sensores: 19, 20, 21, 22, 23. Por fim, o Gateway 4 trata os dados dos sensores: 43, 44, 45, 46, 47. Os dois tipos de configuração utilizam 1 servidor de borda para gerenciamento dos dados dos *gateways* apresentado nas Figuras 5.18 e 5.19.

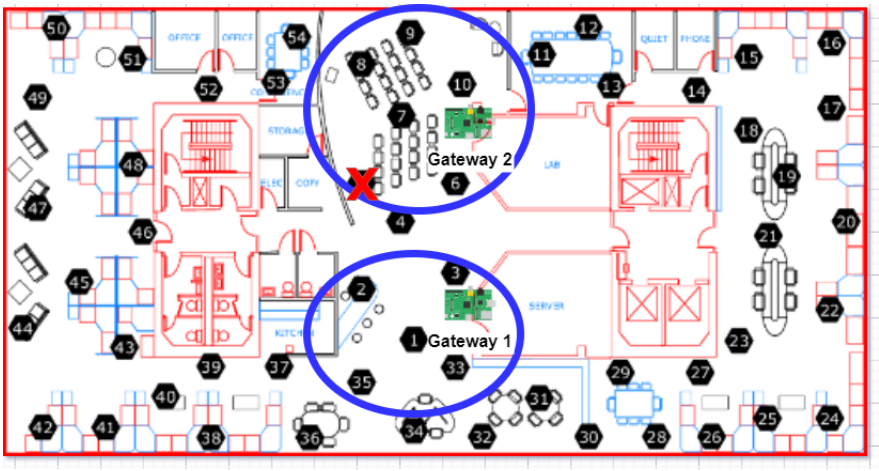

Figura 5.18 Configuração 1 da Emulação.

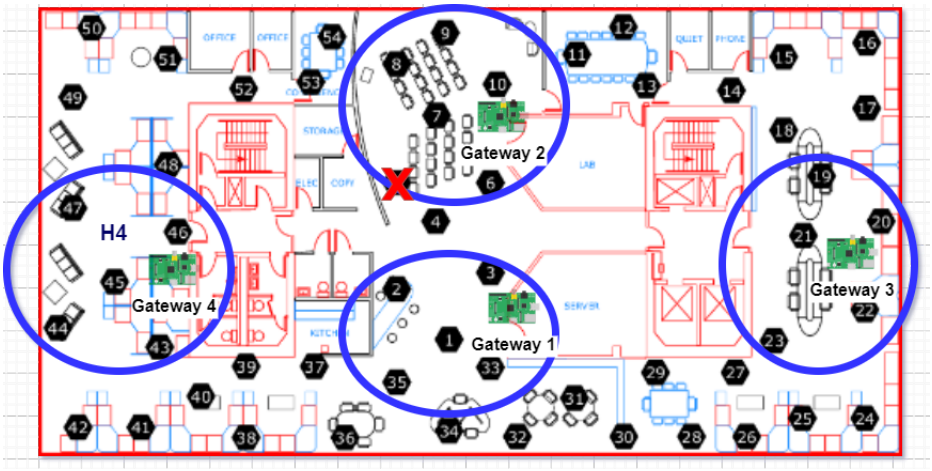

Figura 5.19 Configuração 2 da Emulação.

Para emulação dos componentes da proposta e sensores com os dados produzidos pelo Intel Lab, foi utilizada a ferramenta Mininet. O Mininet permite a criação de *hosts* virtuais com capacidades de conexões em vários tipos de topologias com programação personalizada a partir de sua API. Na próxima seção será apresentado com mais detalhes o Mininet.

5.2.1 Emulador de Redes Mininet

O Mininet[3] é um emulador de redes que permite criar *hosts* virtuais, *switches*, controladores SDN/OpenFlow e links. O emulador Mininet é um projeto[4] de código aberto e disponibilizado de forma gratuita. Por causa dessas características o Mininet é considerado um facilitador na prototipagem, desenvolvimento, aprendizagem, teste e depuração das mais variadas soluções em redes. A utilização do Mininet apresenta as seguintes vantagens: (i) permite o teste de infraestruturas de redes de uma maneira simples e de baixo custo; (ii) possibilita testes complexos de topologia sem a necessidade de equipamentos de redes físicos; e (iii) possui API na linguagem de programação Python para criação e experimentação de infraestruturas de redes.

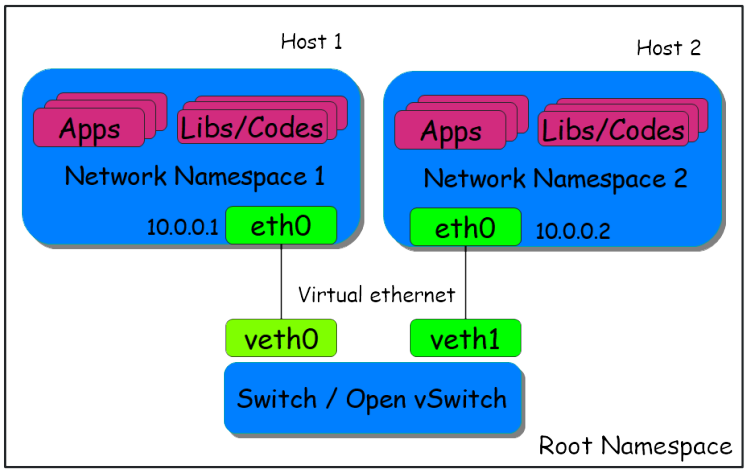

Figura 5.20 Rede Mininet.

O Mininet permite a virtualização baseada em processos para a execução e criação dos componentes da rede. Apesar de compartilhar o mesmo *kernel* linux, todos os dispositivos de rede possuem recursos individuais. A utilização de mecanismos de virtualização, chamados de "linux *network namespaces*", possibilita ao Mininet criar interfaces de redes e tabelas de encaminhamento/roteamento individuais, sendo essas características as principais para a modelagem e implementação dos dispositivos de rede. As interfaces de rede no Mininet são conectadas através de links ethernets virtuais para as diversas instâncias possíveis de *switches* (ver Figura 5.20) como, por exemplo, o *switch* padrão do Mininet e o *Open vSwitch* [5] (*switch* de software multicamada adequado para funcionar como virtual *switch* em ambientes de máquinas virtuais).

[3]disponível em: http://mininet.org/
[4]disponível em: https://github.com/mininet/mininet
[5]disponível em: http://docs.openvswitch.org/en/latest/intro/what-is-ovs/

Por causa dessas características, o Mininet oferece uma maneira simples para modelar infraestrutura de redes. Os *hosts* da rede no Mininet executam código real, incluindo bibliotecas e aplicativos feitas para Linux, assim como a pilha do *kernel* e rede. Diante disso, o código desenvolvido para ser executado em instâncias no Mininet pode ser transferido para um dispositivo real com pequenos ajustes. Por causa dessas característica o Mininet é considerado como um emulador apropriado para implementação de ambientes de *Fog Computing* da IoT. A partir da infraestrutura disponibilizada pelo Minitet os componentes da proposta foram implementados. Essa modelagem é apresentada na próxima seção.

5.2.2 Modelagem da Emulação no Mininet

A emulação do experimento no Mininet foi definida de acordo com a modelagem da Figura 5.21. Como apresentado, foram criados *hosts* virtuais emulando sensores IoT, *host* para executar o FoT-StreamGateway e FoT-StreamServer. A emulação começa com os sensores enviando dados para serem tratados pelo *host gateway*. No *gateway* é executado o Algoritmo 1. A partir das análises no *gateway* os dados são enviados para processamento no *host* servidor que executa o Algoritmo 2.

A partir da especificação do ambiente, os componentes e comunicações foram modelados de acordo com as Figuras 5.22 e 5.23. O *host* sensor, na Figura 5.22, é composto por: serviço que emula o sensor (IoT_Device.py); servidor MQTT; módulos para tratar TATU e Json; e *dataset* dos dados utilizados na simulação. A Figura 5.22 também apresenta o *host gateway*, ele possui um serviço chamado de Stream_Gateway.py onde é executada a emulação do *gateway*. Outros componentes são: servidor MQTT; Api Kafka para comunicação com *host* servidor; módulos para tratar TATU e Json. A comunicação entre os sensores emulados e o *gateway* ocorre na Etapa 1 com o serviço Stream_Gateway.py requisitando no tópico MQTT do IoT_Device.py com uma requisição Flow do TATU. Após essa requisição, na Etapa 2, a emulação do sensor inicia o envio dos dados do *dataset* previamente configurado. Nas simulações executadas neste experimento os tempos de coleta e envio dos dados foram de 1 leitura por segundo.

A modelagem do *host* servidor e sua comunicação com o *host gateway* são apresentadas na Figura 5.23. O *host gateway*, na Figura 5.23, segue a mesma modelagem apresentada anteriormente. O servidor é composto por: Stream_Server.py que é a execução principal do servidor; a biblioteca de aprendizado de máquina Scikit-Learn; Api do kafka para comunicação assíncrona *publish/subscribe*; e a biblioteca do Keras e TensorFlow para implementação das redes neurais. A comunicação entre esses dois componentes ocorre, inicialmente, na Etapa 1, com o envio das janelas de dados tratadas pelo Stream_Gateway.py através do Kafka para o servidor. Na Etapa 2, essas janelas de dados são novamente tratadas pelo servidor e, a partir dessas análises, podem ser enviados para uma aplicação na *Cloud*.

O *host servidor*, além das implementações propostas na Seção 4.3, possui um algoritmo de treinamento de redes neurais. Neste experimento emulado, foi utilizado o Algoritmo 3, apresentado mais adiante, para treinamento do modelo de rede neural e para predição. A arquitetura de rede neural utilizada foi a *Long Short-Term Memory* implementada nas

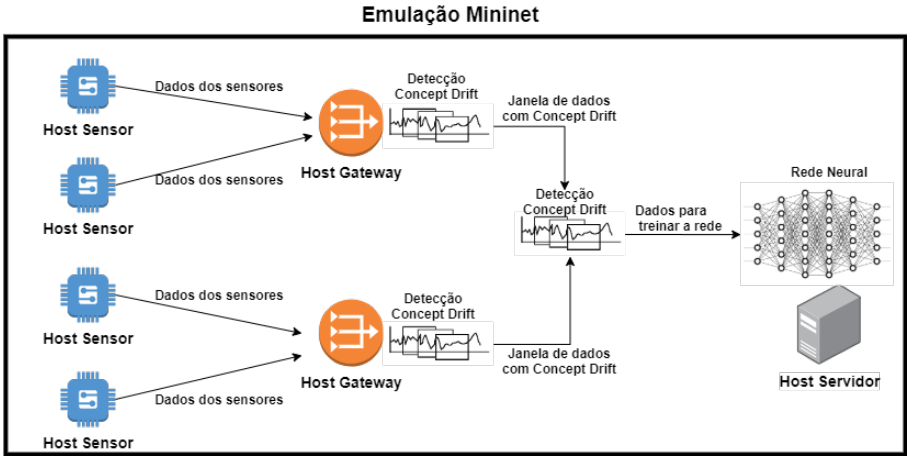

Figura 5.21 Visão geral do experimento no Mininet.

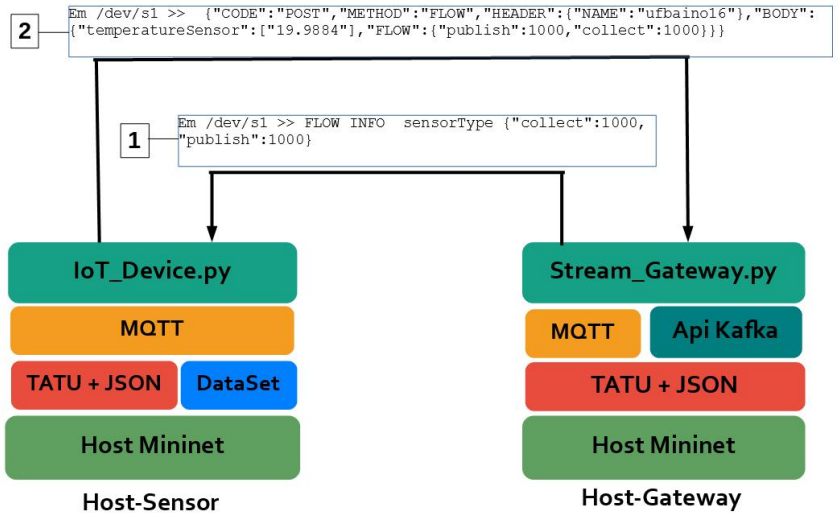

Figura 5.22 Comunicação entre Sensor e Gateway emulados no Mininet.

bibliotecas Keras[6] e TensorFlow[7]. O TensorFlow e Keras foram escolhidos por serem uma plataforma de código aberto para Aprendizado de Máquina. Além disso, possuem um ambiente abrangente e flexível de ferramentas, bibliotecas e recursos da comunidade que permitem que os pesquisadores desenvolvam e implantem facilmente aplicativos com Aprendizado de Máquina. O Algoritmo 3 apresenta a modelagem e as configurações utilizadas:

(i) **Linhas 1-5:** função chamada na inicialização do algoritmo. Nessa função, são

[6]https://keras.io/getting_started/
[7]https://www.tensorflow.org/

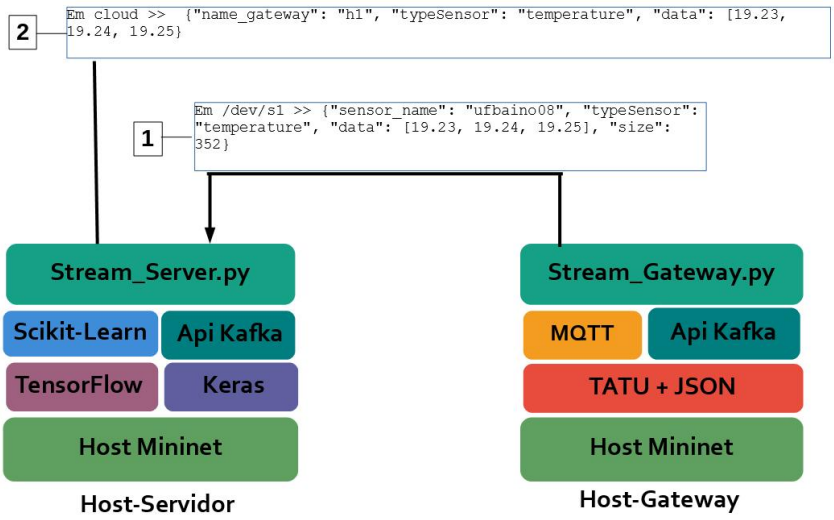

Figura 5.23 Comunicação entre Servidor e Gateway emulados no Mininet.

inicializadas as variáveis utilizadas como configuração da rede neural. Por fim, a função de criação das camadas da rede é executada;

(ii) **Linhas 6-11:** nessa função é executada a configuração da rede. Nesse experimento, a rede é composta por duas camadas (Linhas 9 e 10) com a função de custo Mean Squared Error e a de otimização adam (Linha 11);

(iii) **Linhas 12-16:** essa primeira parte da função realiza as transformações nas janelas de dados para o treinamento. Inicialmente os os dados são convertidos em um *data frame*[8]. A partir desses valores, é calculada a diferença entre os dados no tempo $t-1$ com a observação atual no tempo t (Linha 15). Após isso, a série de diferenças é transformada no formato para treinamento supervisionado, onde a observação no tempo $t-1$ é entrada para a observação do tempo t (Linha 16);

(iv) **Linhas 17-24:** em seguida das transformações executadas nas Linhas 12-16 a janela de dados é dividida em dados de treinamento e testes (Linha 17). A Linha 18 converte os dados numa escala entre -1 e 1 para melhor adaptação da função de ativação da rede LSTM. Nas Linhas 19-24, é testado se o modelo já foi inicializado, caso tenha sido inicializado o modelo é recuperado e retreinado com os novos dados. Caso seja o primeiro treinamento o modelo é criado e depois é realizado o treinamento.

(v) **Linhas 25-35:** nessas linhas são executadas as operações de validação das predições do modelo. Inicialmente, na Linha 28, os dados de treinamento são divididos em

[8]https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html

dados para predição (X) e dados para validação (y). Na Linha 29, é realizada a predição e, nas Linhas 30-31, os dados são remodelados removendo a escala e revertendo os valores da diferença para os valores reais. Nas Linhas 32-34, são criadas as lista com os dados preditos pelo modelo e os que são esperados. Por fim, na Linha 35 são calculadas as métricas de erro.

Data: **predictions**: Lista com os dados de predições realizadas pela rede;
expectations: Lista de dados reais esperados como predição;
input_data: Lista que representa a janela de dados para treinamento do modelo;

```
1  Function __init__()
2      neurons = 4;
3      batch_size = 1;
4      nb_epoch = 100;
5      model = create_model_denses();
6  Function create_model_denses()
7      model = Sequential();
8      model.add(LSTM(neurons, batch_input_shape=(batch_size,1,1), stateful=True));
9      model.add(Dense(2));
10     model.add(Dense(1));
11     model.compile(loss=mean_squared_error, optimizer=adam);
12 Function create_model(input_data)()
13     series = DataFrame(input_data);
14     raw_values = series.values;
15     diff_values = difference(raw_values, 1);
16     supervised_values = timeseries_to_supervised(diff_values, 1);
17     train, test = supervised_values[0:-12], supervised_values[-12:];
18     scaler, train_scaled, test_scaled = scale(train, test);
19     if init_model_status == True then
20         model = open_model(gateway);
21         model = fit_lstm(train_scaled);
22     else
23         model = create_model_denses();
24         model = fit_lstm(train_scaled);
25     predictions = [ ];
26     expectations = [ ];
27     foreach i in range(len(test_scaled)) do
28         X, y = test_scaled[i, 0:-1], test_scaled[i, -1];
29         yhat = forecast_lstm(X);
30         yhat = invert_scale(scaler, X, yhat);
31         yhat = inverse_difference(raw_values, yhat, len(test_scaled)+1-i);
32         predictions.append(yhat[0]);
33         expected = raw_values[len(train) + i + 1];
34         expectations.append(expected[0]);
35     permance_calc(expectations, predictions);
```

Algoritmo 3: Algoritmo de treinamento da rede neural utilizado no servidor de borda.

O Algoritmo 4 é utilizado quando não é necessário realizar treinamento no modelo LSTM. Essa situação ocorre quando não é detectado *concept drift* na janela de dados de treinando. Sendo assim, apenas transformações nos dados para testes de predições são realizadas. Além disso, nesse algoritmo também são executados os cálculos das métricas

de validação do modelo. O algoritmo funciona da seguinte maneira:

(i) **Linhas 1-7:** a primeira parte realiza as transformações nas janelas de dados (input_data) para o treinamento. Os dados são convertidos em um *data frame* [9]. Com esses dados, é calculada a diferença entre a observação no tempo $t-1$ com a observação atual no tempo t. Em seguida, com os dados das diferenças, é gerada uma nova série em que a observação no tempo $t-1$ é entrada para a observação do tempo t. Após isso, os dados são divididos em dados de treinamento e testes. Por fim, os dados de treinamento e testes são convertidos numa escala entre -1 e 1 para melhorar execução da função de ativação aplicada no modelo.

(ii) **Linhas 10-18:** como nesse algoritmo não é realizado treinamento, nessas linhas são executadas apenas a predição e validação do modelo. Primeiramente, os dados de treinamento são dividos em dados para predição (X) e dados para validação (y). Na Linha 12, é realizada a predição. Nas Linhas 13-14 a escala e os valores da diferença são convertidos para os valores reais. Nas Linhas 15-17 são preenchidas as lista com os dados preditos pelo modelo e os que são esperados. Por fim, na Linha 18, as métricas de erro são medidas.

Data: **predictions**: Lista com os dados de predições realizadas pela rede;
expectations: Lista de dados reais esperados como predição;
input_data: Lista que representa a janela de dados para treinamento do modelo;

```
1  Function calc_error(input_data)
2      series = DataFrame(input_data);
3      raw_values = series.values;
4      diff_values = difference(raw_values, 1);
5      supervised_values = timeseries_to_supervised(diff_values, 1);
6      train, test = supervised_values[0:-12], supervised_values[-12:];
7      scaler, train_scaled, test_scaled = scale(train, test);
8      predictions = [ ];
9      expectations = [ ];
10     foreach i in range(len(test_scaled)) do
11         X, y = test_scaled[i, 0:-1], test_scaled[i, -1];
12         yhat = forecast_lstm(X);
13         yhat = invert_scale(scaler, X, yhat);
14         yhat = inverse_difference(raw_values, yhat, len(test_scaled)+1-i);
15         predictions.append(yhat[0]);
16         expected = raw_values[len(train) + i + 1];
17         expectations.append(expected[0]);
18     permance_calc(expectations, predictions);
```

Algoritmo 4: Algoritmo de cálculo do erro das predições no servidor de borda.

[9] https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html

5.2.3 Configurações do Ambiente e Experimentos

Este experimento foi executado utilizando a plataforma de nuvem Google Cloud. A seguinte configuração foi utilizada para conduzir os experimentos: sistema operacional Ubuntu 14.04.4 server amd64 LTS; máquina n1-highmem-2 com 2 vCPUs e 13 GB de memória. Para gerar os resultados deste experimento, foi realizado para cada configuração da Tabela 5.1 a emulação da proposta com duração de 5 horas. A Tabela 5.1 especifica o que foi utilizado em cada execução: quantidade de sensores, quantidade de *gateways*, tipo de algoritmo de *concept drift* e configuração da emulação de acordo com a Seção 5.2. Na execução dos experimentos, foi utilizada uma topologia linear do Mininet com 7 *switches*. A emulação possui conexões de 50 Mb/s entre *switches* e, entre os *hosts* e os *switches*, conexões foram de 3 Mb/s. Os algoritmos utilizados também tiveram alguns parâmetros de configuração. No algoritmo Cumulative Sum (CUSUM), foi definida como 50 a quantidade mínima de instâncias, o delta aplicado foi de 0.0001 e o lambda de 50. No algoritmo Exponentially Weighted Moving Average (EWMA) foi utilizado o número mínimo de instâncias de 50 e o lambda de 1. Por fim, no algoritmo Page Hinkley (PH) a quantidade mínima de instâncias foi de 30, o delta de 0.005 e o alpha de 1. O algoritmo Wavelet de redução de dados aplicado foi Haar com a utilização do nível 1.

Tabela 5.1 Configurações utilizadas nos experimentos.

	Sensores	Gateways	Algoritmo Concept Drift	Configuração
Execução 1	10	2	CUSUM	1
Execução 2	20	4	CUSUM	2
Execução 3	10	2	Page Hinkley	1
Execução 4	20	4	Page Hinkley	2
Execução 5	10	2	EWMA	1
Execução 6	20	4	EWMA	2

5.2.4 Avaliação dos Resultados

A partir da execução dos experimentos, através da modelagem e configurações apresentadas anteriormente, esta seção mostra uma análise sobre como a abordagem proposta neste trabalho influenciou as variáveis de resposta definidas. Para avaliar o desempenho das predições do modelo foram utilizadas as seguintes métricas: raiz do erro médio quadrático (RMSE), erro quadrático médio (MSE), e erro médio absoluto (MAE). Outras avaliações medidas foram: o tráfego de dados na rede e a quantidade de dados processados nos *hosts* das execuções.

Nas métricas de avaliação do modelo, o RMSE, Equação 5.1, mede o erro existente entre dois conjuntos de dados, ou seja, ele faz a comparação entre o valor previsto e um valor observado ou conhecido. A interpretação do RMSE é que quanto menor for seu valor, mais próximos estão os valores preditos e observados. O MSE, Equação 5.2, calcula a média do quadrado dos erros das observações preditas e observadas, em outras

palavras, calcula a diferença quadrática média entre o valor estimado e o real. Por fim, o MAE (Equação 5.3) calcula a magnitude média dos erros entre conjunto de dados de previsões. Esta métrica pode ser definida como a média das amostras de teste das diferenças absolutas entre a previsão e a observação, onde todas as diferenças individuais têm o mesmo peso.

$$RMSE = \sqrt{(\frac{1}{n})\sum_{i=1}^{n}(y_i - x_i)^2} \tag{5.1}$$

$$MSE = (\frac{1}{n})\sum_{i=1}^{n}(y_i - x_i)^2 \tag{5.2}$$

$$MAE = (\frac{1}{n})\sum_{i=1}^{n}|y_i - x_i| \tag{5.3}$$

As Tabelas 5.2, 5.3, 5.4 apresentam os resultados das métricas RMSE, MAE, MSE médio para as execuções do experimento. A Tabela 5.2 exibe os resultados para as execuções com 2 *gateways*. Como pode ser observado, a Execução 3, com o algoritmo Page Hinkley, tive os melhores desempenhos nas métricas RMSE, MAE e MSE, apresentando o menor valor tanto no Gateway 1 quanto no Gateway 2. Para os resultados no Gateway 1, os melhores resultados são apresentados pela Execução 1, com algoritmo CUSUM. Por outro lado, nos resultados do Gateway 2, os melhores resultados são da Execução 3, com algoritmo EWMA. Diante dos resultados, também é possível perceber que os resultados das métricas não ultrapassam o valor 0.5, mostrando a eficácia das predições feitas nas execuções dos experimentos com a proposta deste trabalho.

Tabela 5.2 Resultados médios das métricas dos Execução 1, 3, 5.

	Gateway 1			Gateway 2		
	RMSE	MAE	MSE	RMSE	MAE	MSE
Execução 1	0.47	0.32	0.35	0.36	0.26	0.39
Execução 3	0.46	0.29	0.31	0.28	0.19	0.14
Execução 5	0.58	0.36	0.59	0.33	0.21	0.26

As Tabelas 5.3 e 5.4 apresentam os resultados com a execução para 4 *gateways*. Na Tabela 5.3, é possível destacar que a Execução 6 obteve os melhores resultados tanto nos dados do Gateway 1 quanto do Gateway 2, nessa execução foi utilizado o algoritmo EWMA. Nesses mesmos resultados da tabela, a Execução 2 com algoritmo CUSUM obteve melhores resultados comparado com a Execução 4 do algoritmo Page Hinkley. Na Tabela 5.4, a Execução 6 do algoritmo EWMA obteve os melhores resultados de predição. Nas duas Tabelas 5.3 e 5.4 o maior valor de métrica de erro foi o MSE do Gateway 3, na Execução 4, com valor de 2.01, em todas as outras execuções as métricas ficaram com valor inferior a 2. Diante desses dados, é possível concluir que para predições com 10 sensores a aplicação do algoritmo Page Hinkley apresenta os melhores resultados. Por

outro lado, aumentando a quantidade de sensores para 20, o algoritmo EWMA permite melhores resultados nas predições.

Tabela 5.3 Resultados médios das métricas dos Execução 2, 4, 6.

	Gateway 1			Gateway 2		
	RMSE	MAE	MSE	RMSE	MAE	MSE
Execução 2	0.82	0.53	1.09	0.42	0.26	0.29
Execução 4	1.02	0.85	1.69	0.47	0.31	0.37
Execução 6	0.63	0.38	0.54	0.35	0.22	0.21

Tabela 5.4 Resultados médios das métricas dos Experimentos 2, 4, 6.

	Gateway 3			Gateway 4		
	RMSE	MAE	MSE	RMSE	MAE	MSE
Execução 2	1.01	0.62	1.47	0.77	0.56	1.21
Execução 4	1.17	0.84	2.01	0.66	0.40	0.70
Execução 6	0.97	0.56	1.24	0.51	0.31	0.46

As Figuras 5.24 e 5.25 mostram o gráfico da comparação da temperatura observada e predita pelo algoritmo LSTM com a abordagem proposta neste trabalho. A Figura 5.25 apresenta as predições de temperatura para Execução 1. Diante do gráfico, é possível perceber que em mais de 1000 predições realizadas pelo modelo a diferença entre o valor real e o predito não é grande, pois as linhas que representam essas informações estão sobrepostas. Também é possível extrair outras informações visuais como, por exemplo, nas primeiras 200 predições nas Figuras 5.24 (a) e (b), as linhas que representam as predições não estão totalmente sobrepostas. Isso ocorre devido ao modelo não ter sido ainda treinado com a quantidade de dados que permitam fazer predições mais acertadas. A partir das 200 predições o modelo se ajusta melhor aos dados e permite predições mais precisas. Assim como na figura anterior, a Figura 5.25 apresenta, pela sobreposição das linhas, que o modelo treinado para a Execução 2 (20 sensores e 4 *gateways*) consegue realizar as melhores predições a partir das 200 predições realizadas. Nesse sentido, com os gráficos das Figuras 5.25 (a), (b), (c) e (d), é possível concluir que para mais de 500 predições um modelo com algoritmo LSTM e a abordagem proposta consegue obter resultados satisfatórios nas predições. Os gráficos de todas as execuções podem ser encontrados no repositório do Colab[10].

Uma outra avaliação realizada neste experimento está relacionada com a quantidade de dados processados nas execuções. Essa avaliação permite perceber que os resultados das métricas da avaliação do modelo LSTM foram obtidos com menos dados comparado ao que foi gerado pelo *dataset* original da emulação. Esse fato pode ser percebido nas

[10]https://colab.research.google.com/drive/1L9_oZ7Ni8IsyoUWm7jTmkhy0yVJncMC4?usp=sharing#scrollTo=L2-Vhh1kce-a&uniqifier=1

(a) (b)

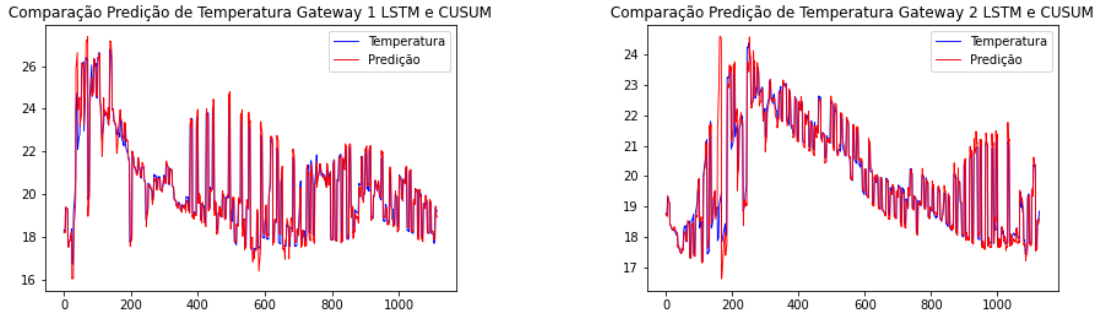

Figura 5.24 Comparação temperatura real e predita Execução 1.

(a) (b)

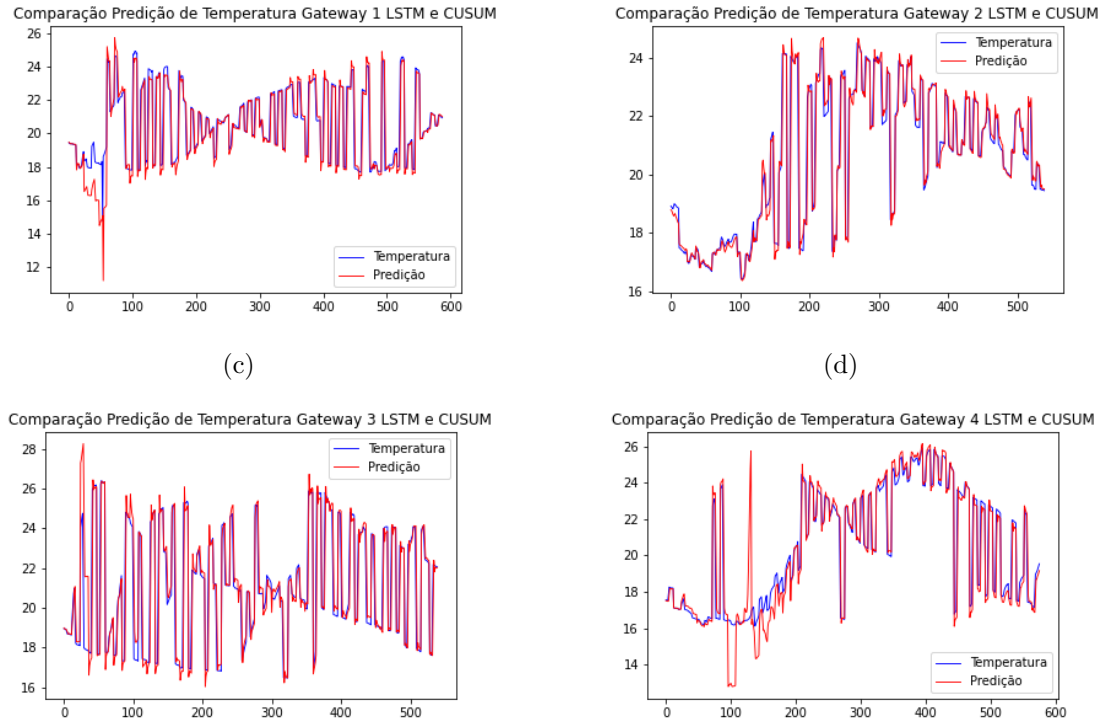

Figura 5.25 Comparação temperatura real e predita Execução 2.

Figuras 5.26 e 5.27. Na Figura 5.26 (a) é apresentado os resultados da Execução 5, nos *gateways* 1 e 2 são recebidos por volta de 229 KB de dados. Por outro lado, após processamento, os dados recebidos pelo servidor para o *gateway* 1 foram de 72 KB e do *gateway* 2 foram 50 KB. Na Figura 5.26, Execução 6, os *gateways* 1, 2, 3, e 4 produziram mais de 155 KB. Entretanto, os dados que foram enviados e processados pelo servidor para todos os *gateways* foram pouco mais de 20 KB.

A Figura 5.27 apresenta a quantidade de dados processados nas Execuções 1, 2, 3 e

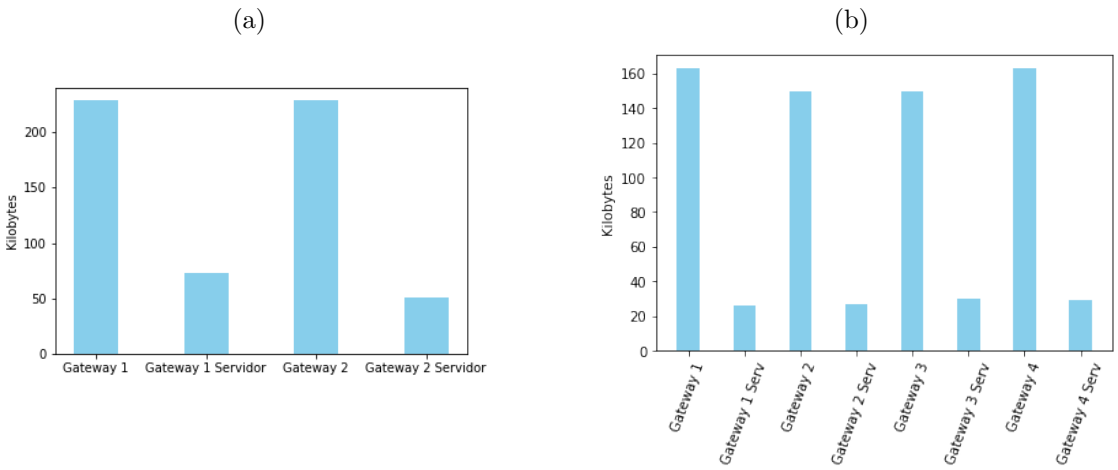

Figura 5.26 Quantidade de dados processados nas Execuções 5, 6.

4. A Figura 5.27 (a) e (c) são os resultados das Execuções 1 e 3, onde ambas contêm 2 *gateways*. Nas duas execuções, nota-se redução dos dados que são processados no servidor. Do mesmo modo, as Figuras 5.27 (b) e (d) apresentam as Execuções 2 e 4 onde ambas contém a redução dos dados processados através da aplicação da abordagem. De modo geral, nas Execuções, a redução de dados processados não aconteceu de maneira diferente para os diferentes algoritmos aplicados nas execuções, eles tiveram resultados parecidos nessa métrica de avaliação.

Neste experimento, também foi realizada a avaliação da quantidade de pacotes de dados que foram trafegados na rede entre os componentes utilizados na emulação. A Figura 5.28 (a) exibe os resultados para Execução 5. Pode-se notar que nos 2 *gateways* os pacotes de dados recebidos dos sensores estão próximos dos 1000 KB, já no servidor o tráfego de pacotes ficou em torno de 27 KB. A Figura 5.28 (b) é a Execução 6 com 4 *gateways*, cujos pacotes enviados dos sensores para os *gateways* também estão próximos 1000 Kb, contudo, no servidor são recebidos 51 KB. Entre a Execução (a) e (b) existe uma aumento de pacotes enviados para o servidor, mas isso é causado pelo incremento da quantidade de sensores e *gateways* nas execuções que consequentemente geram mais dados.

Por fim, a Figura 5.29 apresenta os resultados do tráfego de dados das Execuções 1, 2, 3 e 4. As Figuras 5.29 (a) e (b) exibem os resultados para as Execuções 1 e 2 que são configuradas com 2 *gateways*. Nessas figuras, é possível perceber a redução de pacotes de dados nos servidores, com os *gateways* recebendo por volta de 1000 KB e na Execução 1 o servidor com 27 KB e o servidor da Execução 3 com 20 KB. Por outro lado, as Figuras (a) e (d) exibem os resultados das Execuções 2 e 4 com 4 *gateways*. Através desses gráficos, é possível perceber que os dados trafegados para os *gateways* estão em torno de 1000 KB, já no servidor da Execução 1, Figura 5.29 (b), tem o valor de 45 KB. A Execução 4, Figura 5.29 (d), tem a quantidade de pacotes de dados trafegados para o servidor com 58 KB.

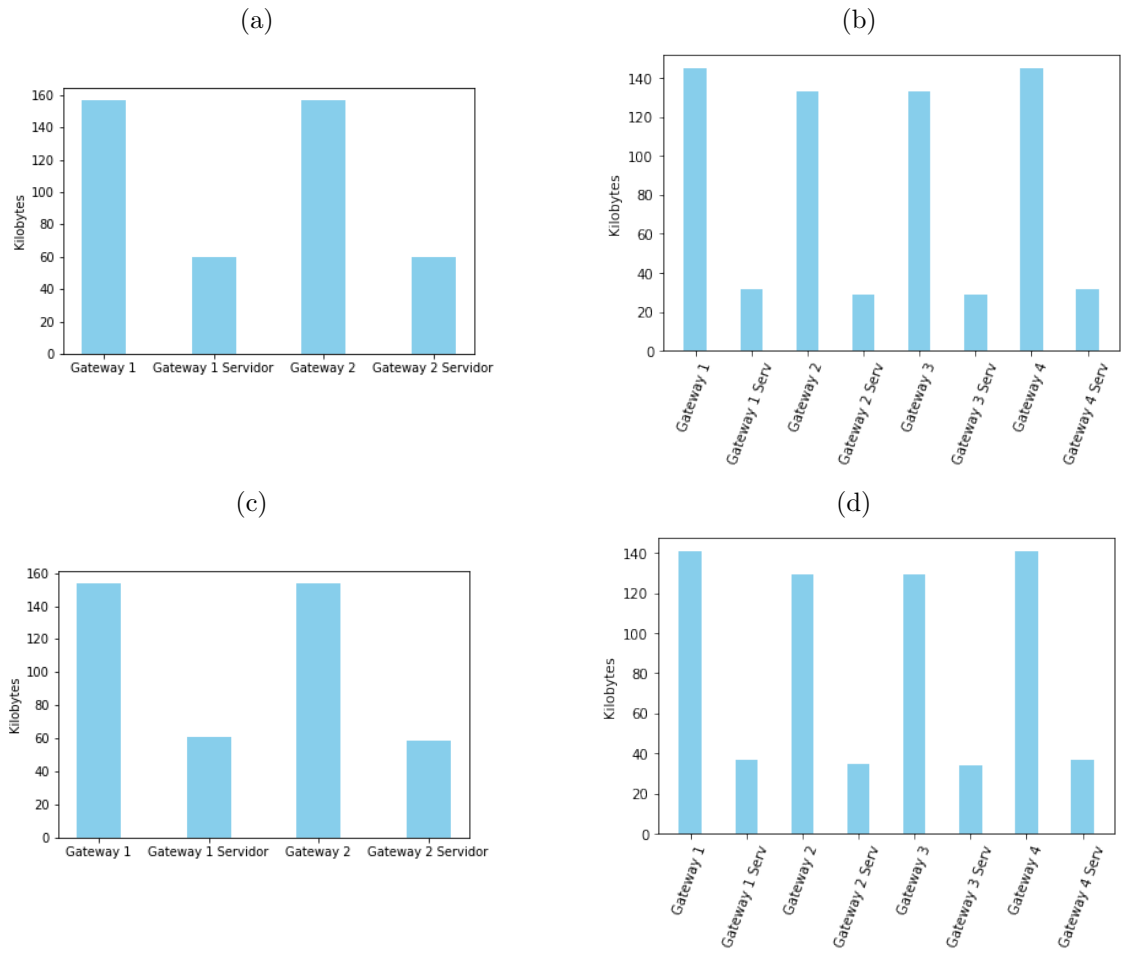

Figura 5.27 Quantidade de dados processados nas Execuções 1, 2, 3, 4.

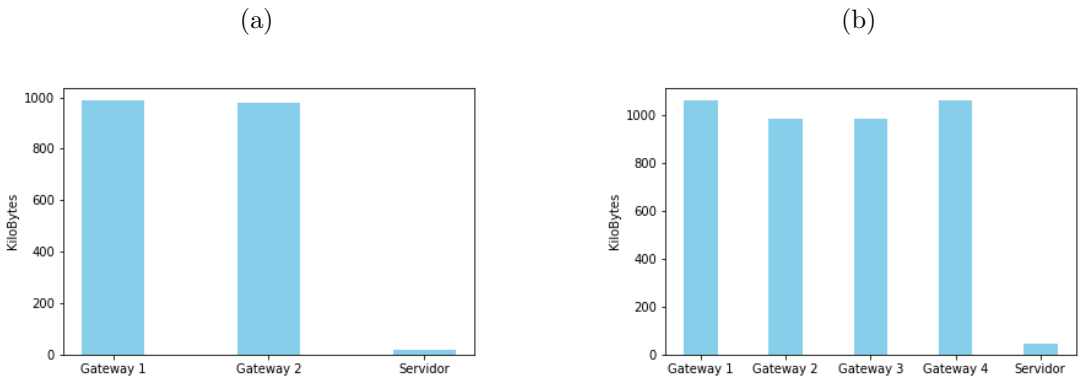

Figura 5.28 Tráfego de dados nas Execuções 5, 6.

5.3 CONSIDERAÇÕES FINAIS

Este Capítulo apresentou os experimentos da solução proposta avaliando as métricas: tráfego de dados na rede, quantidade de dados processados, RMSE, MAE, MSE, tempo

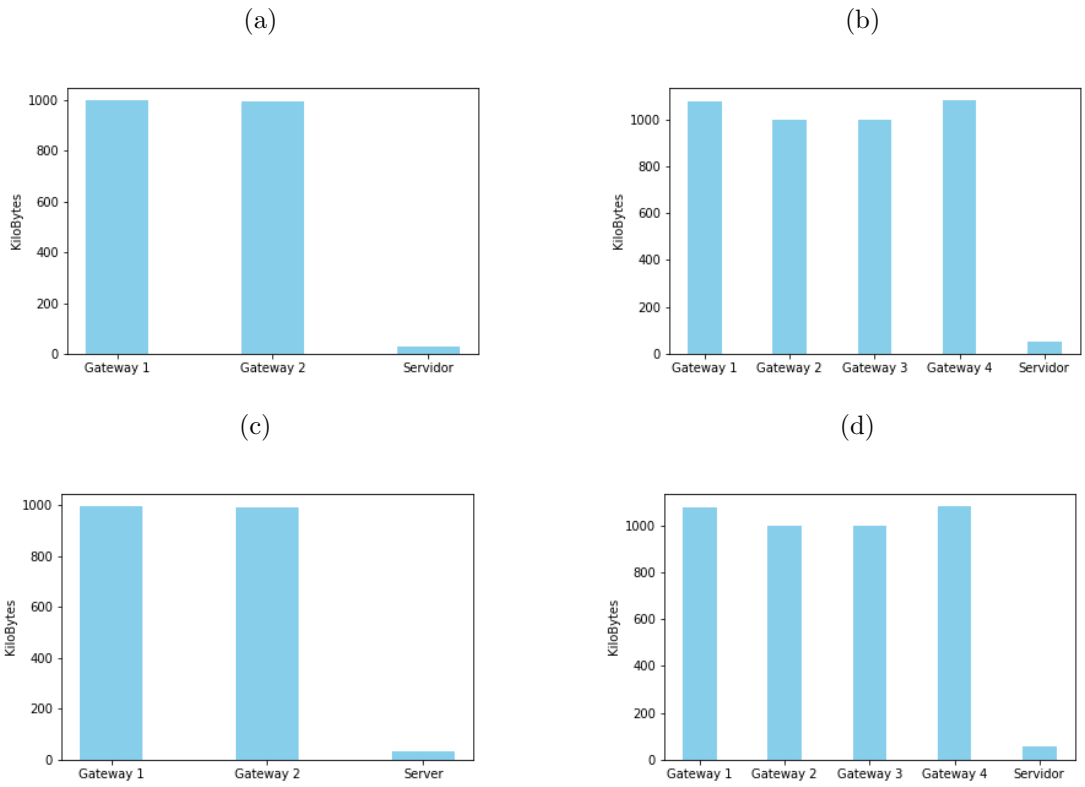

Figura 5.29 Tráfego de dados nas Execuções 1, 2, 3, 4.

de atraso no processamento e *throughput*. A partir das análises dos resultados, é possível perceber que a aplicação da abordagem proposta possibilita uma redução de processamento e tráfego de dados na rede. Essa redução não elimina a representatividade do fluxo de dados nos níveis da *Fog Computing*. Pode-se perceber que existe um padrão comum nos experimentos entre os dados trafegados nos níveis mais baixos da *Fog* até o mais alto. Além disso, o modelo de predição pode ser treinado com menos dados, diminuindo o consumo de rede no tráfego de dados e processamento de entrada no algoritmo. Apesar disso, as predições avaliadas a partir das métricas não apresentam resultados de predições aleatórias. Por fim, o *throughput* e tempo de atraso na aplicação da abordagem não influenciam diretamente no *delay* do tráfego do fluxo de dados entre os componentes.

Capítulo 6

CONCLUSÃO

Esta dissertação de mestrado apresenta uma abordagem de *Fog Computing* que propõem a utilização das técnicas de mineração de fluxo de dados wavelet e *concept drift* para aprimorar o processamento e análise dos dados na borda da rede. A partir disso, foram realizados experimentos que demonstram a eficácia da proposta em dois cenários de experimentos (Capítulo 5). Além disso, um ambiente de simulação para abordagem proposta foi desenvolvido. Com isso, é possível replicar e estender a execução da simulação nos mais variados cenários da *Fog Computing* e Internet das Coisas. Esses resultados foram produzidos baseados nos conceitos apresentados no Capitulo 2 e a revisão sistemática da literatura do Capítulo 3 que dão suporte para fundamentação da proposta.

6.1 CONTRIBUIÇÕES DO TRABALHO REALIZADO

Esse trabalho é motivado pelo problema inicial de tratar o *Big Data Stream* gerado pelo grande número de dispositivos da Internet das Coisas. Esse sensores devem produzir dados constantemente e em curto espaço de tempo. Por isso, pode ocorrer a sobrecarga na infraestrutura de rede e, com isso, consumir grande parte dos recursos das infraestruturas computacionais, por exemplo, banda de rede e armazenamento. A partir desse problema, a abordagem proposta neste trabalho, torna-se ferramenta simples e realista que permite atacar diretamente o problema levantado pelo *Big Data Stream*. O realismo da aplicabilidade da abordagem pode ser constatado a partir dos cenários de experimentos que foram realizados. Além disso, através da disponibilização da simulação em um repositório livre, é possível realizar a replicação do trabalho. Dessa forma, as principais contribuições referentes ao trabalho apresentado nesta dissertação estão resumidas a seguir:

1. Evolução da plataforma de Internet das Coisas e *Fog Computing* SOFT-IoT para suportar processamento e análise de fluxo de dados, servindo de infraestrutura para utilização das técnicas de mineração de fluxo de dados;

2. Definição e implementação de uma abordagem de análise de fluxo de dados da Internet das Coisas na *Fog Computing*, isso utilizando as técnicas de mineração de fluxo de dados;

3. Aplicação de algoritmo de predição na *Fog Computing* para realizar predições no fluxo de dados através da abordagem proposta;

4. Implementação de um ambiente de simulação com os principais componentes da proposta;

5. Realização da integração entre os paradigmas de *Fog of Things* e mineração em *Big Data Stream*.

6.2 LIMITAÇÕES

As limitações deste trabalho estão relacionadas, principalmente, as métricas de desempenho que não foram avaliadas nos experimentos. Os resultados do primeiro experimento não levou em consideração medir a utilização da CPU e memória dos dispositivos. Outra limitação identificada é que na emulação, os experimentos são executados na mesma máquina, não é avaliado o possível impacto que a sobrecarga de processamento, provocada pela execução dos experimentos, causa no desempenho dos algoritmos executados. Nos experimentos também podem ser avaliados a variação dos tamanhos das janelas. Com isso, é possível identificar a qualidade das predições e a quantidade de dados que são reduzidos para diferentes tamanhos de janelas. Outro tipo de experimento que pode ser realizado é a utilização de fluxos maiores e menores com padrões variáveis da distribuição dos dados. Diante isso, é possível identificar qual a eficácia da abordagem para diferentes distribuições de dados. Por fim, a abordagem proposta não foi implantada e testada a partir de dispositivos mais limitados, como, por exemplo, sensores embutidos com microcontroladores.

6.3 TRABALHOS FUTUROS

O conteúdo apresentado nesta dissertação permite identificar alguns trabalhos futuros, tanto para dar continuidade à pesquisa aqui reportada, como para avaliar as limitações apresentadas na Seção 6.2. A seguir uma lista de oportunidades para trabalhos futuros:

- Realizar implementações para abordagem ser eficiente para ser aplicada em dispositivos com menor poder de processamento, por exemplo, microcontroladores;

- Realizar novos experimentos em cenários reais e emulados para verificar o consumo de recursos computacionais pela abordagem, como, por exemplo, CPU e memória;

- Realizar testes em larga escala para algoritmos que utilizem modelos predição sofisticados com análises a partir dos sensores.

REFERÊNCIAS BIBLIOGRÁFICAS

AL-FUQAHA, A. et al. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, IEEE, v. 17, n. 4, p. 2347–2376, 2015.

ANAND, N. et al. Practical edge analytics: Architectural approach and use cases. In: IEEE. *Edge Computing (EDGE), 2017 IEEE International Conference on*. [S.l.], 2017. p. 236–239.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. *Computer networks*, Elsevier, v. 54, n. 15, p. 2787–2805, 2010.

BANDYOPADHYAY, S. et al. Sensipro: Smart sensor analytics for internet of things. In: IEEE. *Computers and Communication (ISCC), 2016 IEEE Symposium on*. [S.l.], 2016. p. 415–421.

BAUER, M. et al. Iot reference architecture. In: *Enabling Things to Talk*. [S.l.]: Springer, 2013. p. 163–211.

BECKEL, C. et al. Improving device-level electricity consumption breakdowns in private households using on/off events. *ACM SIGBED Review*, ACM, v. 9, n. 3, p. 32–38, 2012.

BELLI, L. et al. Applying security to a big stream cloud architecture for the internet of things. *International Journal of Distributed Systems and Technologies (IJDST)*, IGI Global, v. 7, n. 1, p. 37–58, 2016.

Bengio, Y.; Simard, P.; Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, v. 5, n. 2, p. 157–166, March 1994. ISSN 1941-0093.

BHARGAVA, K. et al. Using edge analytics to improve data collection in precision dairy farming. In: IEEE. *Local Computer Networks Workshops (LCN Workshops), 2016 IEEE 41st Conference on*. [S.l.], 2016. p. 137–144.

BIFET, A. Mining big data in real time. *Informatica (Slovenia)*, v. 37, n. 1, p. 15–20, 2013.

BIFET, A.; KIRKBY, R. *Data stream mining: a practical approach*. [S.l.], 2009.

BIFET, A.; READ, J. Ubiquitous artificial intelligence and dynamic data streams. In: *Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems*. New York, NY, USA: ACM, 2018. (DEBS '18), p. 1–6. ISBN 978-1-4503-5782-1. Disponível em: ⟨http://doi.acm.org/10.1145/3210284.3214345⟩.

BISWAS, A. R.; DUPONT, C.; PHAM, C. Iot, cloud and bigdata integration for iot analytics. *Building Blocks for IoT Analytics*, River Publishers, p. 11, 2016.

BONOMI, F. et al. Fog computing: A platform for internet of things and analytics. In: *Big Data and Internet of Things: A Roadmap for Smart Environments*. [S.l.]: Springer, 2014. p. 169–186.

BONOMI, F. et al. Fog computing and its role in the internet of things. In: ACM. *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. [S.l.], 2012. p. 13–16.

CANARIO, J. P. et al. In-depth comparison of deep artificial neural network architectures on seismic events classification. *Journal of Volcanology and Geothermal Research*, v. 401, p. 106881, 2020. ISSN 0377-0273. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0377027319306171⟩.

CARVALHO, A. et al. Inteligência artificial–uma abordagem de aprendizado de máquina. *Rio de Janeiro: LTC*, 2011.

CONSORTIUM, O. et al. Openfog reference architecture for fog computing. *Architecture Working Group*, p. 1–162, 2017.

DAUTOV, R. et al. Pushing intelligence to the edge with a stream processing architecture. In: IEEE. *Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2017 IEEE International Conference on*. [S.l.], 2017. p. 792–799.

DOHR, A. et al. The internet of things for ambient assisted living. In: IEEE. *Information technology: new generations (ITNG), 2010 seventh international conference on*. [S.l.], 2010. p. 804–809.

DUNKELS, A.; VASSEUR, J. *IP for Smart Objects, Internet Protocol for Smart Objects (IPSO) Alliance*. 2008.

DYBA, T.; DINGSOYR, T.; HANSSEN, G. K. Applying systematic reviews to diverse study types: An experience report. In: IEEE. *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*. [S.l.], 2007. p. 225–234.

ELMAN, J. L. Finding structure in time. *Cognitive Science*, v. 14, n. 2, p. 179 – 211, 1990. ISSN 0364-0213. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/036402139090002E⟩.

GAMA, J. *Knowledge discovery from data streams*. [S.l.]: CRC Press, 2010.

GAMA, J.; RODRIGUES, P. P. An overview on mining data streams. In: *Foundations of Computational, IntelligenceVolume 6*. [S.l.]: Springer, 2009. p. 29–45.

GAMA, J. et al. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, ACM, v. 46, n. 4, p. 44, 2014.

GIMENEZ, R. et al. The safety transformation in the future internet domain. In: SPRINGER. *The Future Internet Assembly*. [S.l.], 2012. p. 190–200.

HARTH, N.; DELAKOURIDIS, K.; ANAGNOSTOPOULOS, C. Convey intelligence to edge aggregation analytics. In: *New Advances in the Internet of Things*. [S.l.]: Springer, 2018. p. 25–44.

HAYKIN, S. *Redes neurais: Principios e prática*. 2nd. ed. Bookman, 2005. ISBN 9788573077186. Disponível em: ⟨http://gen.lib.rus.ec/book/index.php?md5=8b97272ccefcbd87bf869fd0c292d7c2⟩.

HE, J. et al. Multi-tier fog computing with large-scale iot data analytics for smart cities. *IEEE Internet of Things Journal*, IEEE, 2017.

HEUVELDOP, N. et al. Ericsson mobility report. *Ericsson AB, Technol. Emerg. Business, Stockholm, Sweden, Tech. Rep. EAB-17*, v. 5964, 2017.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Comput.*, MIT Press, Cambridge, MA, USA, v. 9, n. 8, p. 1735–1780, nov. 1997. ISSN 0899-7667. Disponível em: ⟨https://doi.org/10.1162/neco.1997.9.8.1735⟩.

KAMATH, U.; CHOPPELLA, K. *Mastering Java Machine Learning*. [S.l.]: Packt Publishing Ltd, 2017.

KITCHENHAM, B.; CHARTERS, S. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. 2007.

LAVASSANI, M. et al. Combining fog computing with sensor mote machine learning for industrial iot. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 18, n. 5, p. 1532, 2018.

LEE, J. et al. Integrating machine learning in embedded sensor systems for internet-of-things applications. In: IEEE. *Signal Processing and Information Technology (ISSPIT), 2016 IEEE International Symposium on*. [S.l.], 2016. p. 290–294.

LOU, P. et al. Agile supply chain management over the internet of things. In: IEEE. *2011 International Conference on Management and Service Science*. [S.l.], 2011. p. 1–4.

LUJIC, I.; MAIO, V. D.; BRANDIC, I. Efficient edge storage management based on near real-time forecasts. In: IEEE. *Fog and Edge Computing (ICFEC), 2017 IEEE 1st International Conference on*. [S.l.], 2017. p. 21–30.

MA, B. B.; FONG, S.; MILLHAM, R. Data stream mining in fog computing environment with feature selection using ensemble of swarm search algorithms. In: IEEE. *Information Communications Technology and Society (ICTAS), 2018 Conference on*. [S.l.], 2018. p. 1–6.

MANYIKA, J. et al. Unlocking the potential of the internet of things. *McKinsey Global Institute*, 2015.

MARJANI, M. et al. Big iot data analytics: Architecture, opportunities, and open research challenges. *IEEE Access*, IEEE, v. 5, p. 5247–5261, 2017.

MINERVA, R.; BIRU, A.; ROTONDI, D. Towards a definition of the internet of things (iot). *IEEE Internet Initiative*, n. 1, 2015.

MOHAMMADI, M. et al. Deep learning for iot big data and streaming analytics: A survey. *IEEE Communications Surveys & Tutorials*, IEEE, v. 20, n. 4, p. 2923–2960, 2018.

Mouss, H. et al. Test of page-hinckley, an approach for fault detection in an agro-alimentary production system. In: *2004 5th Asian Control Conference (IEEE Cat. No.04EX904)*. [S.l.: s.n.], 2004. v. 2, p. 815–818 Vol.2.

NGUYEN, H.-L.; WOON, Y.-K.; NG, W.-K. A survey on data stream clustering and classification. *Knowledge and information systems*, Springer, v. 45, n. 3, p. 535–569, 2015.

OYEKANLU, E. Predictive edge computing for time series of industrial iot and large scale critical infrastructure based on open-source software analytic of big data. In: IEEE. *Big Data (Big Data), 2017 IEEE International Conference on*. [S.l.], 2017. p. 1663–1669.

PAGE, E. S. Continuous inspection schemes. *Biometrika*, [Oxford University Press, Biometrika Trust], v. 41, n. 1/2, p. 100–115, 1954. ISSN 00063444. Disponível em: ⟨http://www.jstor.org/stable/2333009⟩.

PATEL, P.; ALI, M. I.; SHETH, A. On using the intelligent edge for iot analytics. *IEEE Intelligent Systems*, IEEE, v. 32, n. 5, p. 64–69, 2017.

PECORI, R. A virtual learning architecture enhanced by fog computing and big data streams. *Future Internet*, Multidisciplinary Digital Publishing Institute, v. 10, n. 1, p. 4, 2018.

PRANATA, A. A.; LEE, J. M.; KIM, D. S. Towards an iot-based water quality monitoring system with brokerless pub/sub architecture. In: IEEE. *Local and Metropolitan Area Networks (LANMAN), 2017 IEEE International Symposium on*. [S.l.], 2017. p. 1–6.

PRAZERES, C.; SERRANO, M. Soft-iot: Self-organizing fog of things. In: IEEE. *Advanced Information Networking and Applications Workshops (WAINA), 2016 30th International Conference on*. [S.l.], 2016. p. 803–808.

QAISAR, S. B.; USMAN, M. Fog networking for machine health prognosis: A deep learning perspective. In: SPRINGER. *International Conference on Computational Science and Its Applications*. [S.l.], 2017. p. 212–219.

RAAFAT, H. M. et al. Fog intelligence for real-time iot sensor data analytics. *IEEE Access*, IEEE, v. 5, p. 24062–24069, 2017.

ROSS, G. J. et al. Exponentially weighted moving average charts for detecting concept drift. *Pattern recognition letters*, Elsevier, v. 33, n. 2, p. 191–198, 2012.

SANTANA, C. J. L. de; ALENCAR, B. de M.; PRAZERES, C. V. S. Reactive microservices for the internet of things: A case study in fog computing. In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. New York, NY, USA: ACM, 2019. (SAC '19), p. 1243–1251. ISBN 978-1-4503-5933-7. Disponível em: ⟨http://doi.acm.org/10.1145/3297280.3297402⟩.

SOLDATOS, J. *Building Blocks for IoT Analytics:*. River Publishers, 2016. (River Publishers Series in Signal, Image and Speech Processing). ISBN 9788793519039. Disponível em: ⟨https://books.google.com.br/books?id=svQRMQAACAAJ⟩.

TSAI, P.-H. et al. Distributed analytics in fog computing platforms using tensorflow and kubernetes. In: IEEE. *Network Operations and Management Symposium (APNOMS), 2017 19th Asia-Pacific*. [S.l.], 2017. p. 145–150.

WHITMORE, A.; AGARWAL, A.; XU, L. D. The internet of things—a survey of topics and trends. *Information Systems Frontiers*, Springer, v. 17, n. 2, p. 261–274, 2015.

YANG, S. Iot stream processing and analytics in the fog. *IEEE Communications Magazine*, IEEE, v. 55, n. 8, p. 21–27, 2017.

ZOU, Z. et al. Edge and fog computing enabled ai for iot-an overview. In: IEEE. *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. [S.l.], 2019. p. 51–56.

Apêndice

A

RESULTADOS DA REVISÃO SISTEMÁTICA

Este apêndice apresenta os principais resultados obtidos na execução da revisão sistemática da literatura. A Tabela A.1 apresenta os detalhes dos artigos depois de lidos, são analisados 4 aspectos dos trabalhos de acordo com as perguntas definidas na Seção 2.5. Os aspectos são: nível da *Fog/Edge*; técnicas; contexto; avaliação.

Tabela A.1: Artigos selecionados na revisão sistemática.

ID	Título	Autores	Nível da Fog/Edge	Técnicas	Contexto	Avaliação
EP1	Efficient Edge Storage Management Based on Near Real-Time Forecasts	(LUJIC; MAIO; BRANDIC, 2017)	Utilização da camada de borda (EDGE LAYER)	Análise de dados de séries temporais provenientes de sensores IoT para melhorar a eficiência no gerenciamento de dados na borda da rede. Utiliza um algoritmo adaptativo com métodos de previsão (ETS ou ARIMA) .	Os conjuntos de dados são obtidos pelo UMass Trace Repository que são dados provenientes do projeto Smart* que tem a finalidade de criar casas sustentáveis	O algoritmo adaptativo proposto foi avaliado realizando simulações com o pacote de previsão do R
EP2	Practical Edge Analytics: Architectural Approach and Use Cases	(ANAND et al., 2017)	Apresenta uma arquitetura para implantação de uma estrutura prática de análise de dados na borda da rede	Abordagens tecnológicas que permitem análise de dados na Edge Computing	O artigo apresenta a implantação contínua do framework de análise de dados na Fog para uma grande empresa de petróleo e gás	Estudo de caso numa empresa de petróleo e gás
EP3	IoT Stream Processing and Analytics in the Fog	(YANG, 2017)	Proposta de arquitetura para processamento de Stream de dados IoT e análises em todos os níveis da Fog	Direcionamentos sobre tecnologias que dão suporte para o processamento de fluxo de dados na borda da rede	Cenário de energia solar dinâmica	Não informada
EP4	Fog Intelligence for Real-Time IoT Sensor Data Analytics	Raafat et al. (2017)	Análise de dados no nível dos sensores	Neste artigo, sete recursos foram extraídos e alimentados numa rede neural para classificação de sinal e detecção de 30 eventos no nível da Fog	Dados coletados dos seguintes sensores: luminosidade, temperatura, CO2 e umidade, com objetivo de prever a ocupação de uma sala de escritório	As métricas utilizadas foram: sensibilidade, especificidade, precisão, acurácia, F1-score, G-mean

Tabela A.1: Artigos selecionados na revisão sistemática.

ID	Título	Autores	Nível da Fog/Edge	Técnicas	Contexto	Avaliação
EP5	Using Edge Analytics to Improve Data Collection in Precision Dairy Farming	(BHARGAVA et al., 2016)	A aplicação é executada nos sensores	O artigo utiliza L-SIP sobre ClassAct e Bare Necessities, uma das técnicas Edge Mining que fornece feedbacks em tempo real orientado para eventos, permitindo a reconstrução precisa dos dados originais do sensor	O sistema deste artigo é um protótipo baseado em WSN (Wireless Sensor Networks) para a coleta de dados em uma fazenda de vacas leiteira	A avaliação é feita no desempenho de L-SIP em termos da Raiz do Erro Médio Quadrático (RMSE) e ganhos de memória usando a análise em R.
EP6	Multi-tier Fog Computing with Large-scale IoT Data Analytics for Smart Cities	(HE et al., 2017)	O trabalho está relacionado a todo o ecossistema da Fog Computing	O artigo apresentou experimentos com regressão logística (LR) e máquina de vetor de suporte (SVM) apenas para fins de demonstração	O artigo propõe um serviço analítico baseado em modelos de computação em nevoeiro de várias camadas para aplicações de cidades inteligentes	Neste artigo foi realizado um benchmarking sobre A-Fogs, são utilizados cinco conjuntos de dados com diferentes tamanhos de dados
EP7	On Using the Intelligent Edge for IoT Analytics	(PATEL; ALI; SHETH, 2017)	Define uma arquitetura de *Fog* para Internet das Coisas	Define uma arquitetura para análises de dados IoT na *Fog Computing*	Não informado	Não informada
EP8	Fog networking for machine health prognosis: A deep learning perspective	(QAISAR; USMAN, 2017)	Propõe uma arquitetura de Fog atuando no nível do gateway	Artigo trata do uso da Fog para suportar o aprendizado profundo ao executar diferentes redes neurais na névoa para facilitar o usuário final e a utilização da Cloud. As técnicas são: Convolutional Neural Network e Recurrent Neural Network	A pesquisa mostrou a importância do modelo da CNN (Convolutional Neural Network) em classificação de imagens e também no prognóstico de saúde	Foram realizadas classificações com os algoritmos propostos

Tabela A.1: Artigos selecionados na revisão sistemática.

ID	Título	Autores	Nível da Fog/Edge	Técnicas	Contexto	Avaliação
EP9	SensIPro: Smart sensor analytics for Internet of things	(BANDYOPADHYAY et al., 2016)	Apresenta um novo modelo agnóstico de sensores chamado de SensIPro para realizar uma análise robusta e não supervisionada de dados de sensores para suportar análises escaláveis	Análise de séries temporais utilizando a técnica Dynamic Time Warping (DTW)	Dados de accelerometer, Smart Meter, PPG, ECG	O SensIPro foi avaliado considerando sinais heterogêneos de sensores de séries temporais com dinâmica de sinal diversificada e natureza agnóstica de sensores estabelecida no modelo
EP10	Convey intelligence to edge aggregation analytics	(HARTH; DELAKOURIDIS; ANAGNOSTOPOULOS, 2018)	A análise de dados é realizada nos sensores e *nodes* da fog computing	Análise preditiva dos dados na rede de sensores e agregação de dados na borda da rede	CO, PT08.S1 (óxido de estanho), HydroCarbons Não Metânicos, Benzeno, PT08.S2 (titânia), NOx, PT08.S3 (óxido de tungstênio), NO2, PT08.S4 PT08.S5 (óxido de índio), temperatura, umidade relativa e humidade absoluta. Parâmetros utilizados para medir a poluição do ar de uma área específica.	Não informada
EP11	Distributed analytics in fog computing platforms using tensorflow and kubernetes	(TSAI et al., 2017)	A arquitetura proposta abrange todos níveis da Fog Computing	Aplica uma rede neural numa imagem inteira e executa classificação e localização. Técnica YOLO (Real-Time Object Detection)	Detecção de Poluição do Ar e aplicação para classificação e localização de imagens	A plataforma proposta foi testada num cenário real e avaliada de acordo com as métricas de utilização do CPU, network overhead e número de processamento de imagens por minuto

Tabela A.1: Artigos selecionados na revisão sistemática.

ID	Título	Autores	Nível da Fog/Edge	Técnicas	Contexto	Avaliação
EP12	Integrating machine learning in embedded sensor systems for Internet-of-Things applications	(LEE et al., 2016)	Análise embarcada nos sensores	O processo de aprendizagem de máquina utilizado nesse artigo depende de um modelo de mistura gaussiana (GMM) que usa o algoritmo de maximização da expectativa (EM) com o critério de comprimento mínimo de descrição (MDL)	O sistema integrado de sensores e ML proposto no artigo foi utilizado no gerenciamento de um fogão inteligente e sistema de recirculação de água, onde o sistema de sensores integrados monitora várias condições	A implementação embutida é comparada com uma implementação offline no MATLAB
EP13	Predictive Edge Computing for Time Series of Industrial IoT and Large Scale Critical Infrastructure based on Open-source Software Analytic of Big Data	(OYEKANLU, 2017)	Utiliza a camada do paradigma da *Edge Computing*	Comparação de série temporal de referência com a produzida pelos dispositivos, utiliza as técnicas estatísticas: análise de magnitude, assimetria, raiz quadrada média e fatores de crista	O cenário utilizado neste trabalho tem o objetivo de interpretar os dados da vibração de máquinas industriais e análise de máquinas rotativas	Não definida no artigo