UFRPE

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO
DEPARTAMENTO DE ESTATÍSTICA E INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA APLICADA

Investigação de técnicas de localização de dispositivos IoT em redes LoRa

Daniel José de Carvalho

Recife - PE, Fevereiro 2024

Daniel José de Carvalho

Investigação de técnicas de localização de dispositivos IoT em redes LoRa

Dissertação submetida à Coordenação do Programa de Pós-Graduação em Informática Aplicada do Departamento de Estatística e Informática - DEINFO - Universidade Federal Rural de Pernambuco, como parte dos requisitos necessários para obtenção do grau de Mestre.

Universidade Federal Rural de Pernambuco – UFRPE

Departamento de Estatística e Informática

Programa de Pós-Graduação em Informática Aplicada

Orientador: Prof. Dr. Victor Wanderley Costa de Medeiros

Coorientador: Prof. Dr. Rodrigo Gabriel Ferreira Soares

Recife - PE

Fevereiro 2024

Dados Internacionais de Catalogação na Publicação
Universidade Federal Rural de Pernambuco
Sistema Integrado de Bibliotecas
Gerada automaticamente, mediante os dados fornecidos pelo(a) autor(a)

C331i Carvalho, Daniel José de
Investigação de técnicas de localização de dispositivos IoT em redes LoRa / Daniel José de Carvalho. - 2024.
77 f. : il.

Orientador: Victor Wanderley Costa de Medeiros.
Coorientador: Rodrigo Gabriel Ferreira Soares.
Inclui referências.

Dissertação (Mestrado) - Universidade Federal Rural de Pernambuco, Programa de Pós-Graduação em Informática Aplicada, Recife, 2024.

1. geolocalização. 2. redes neurais. 3. machine learning. 4. redes sem fio. 5. lora. I. Medeiros, Victor Wanderley Costa de, orient. II. Soares, Rodrigo Gabriel Ferreira, coorient. III. Título

CDD 004

Dedico a todas as pessoas que direta ou indiretamente contribuíram por toda a minha vida para construir minha formação até então, para que dessa forma eu conseguisse atingir mais esse objetivo.

Agradecimentos

Gostaria de agradecer primeiramente a minha família, especialmente minha esposa Rafaela por ter entendido que por muitas vezes eu precisava de um tempo a mais para desenvolver este trabalho. Houveram algumas situações na minha família durante esse trabalho, mas consegui superar pelo apoio da própria família e amigos, e assim dar seguimento ao projeto.

Agradecer aos amigos, desde os de longa data até os mais recentes, pois com eles eu conseguia relaxar, conversar assuntos diversos e me divertir um pouco, para que assim conseguisse reduzir minha fadiga e cansaço mental durante esse período.

Agradecer aos meus professores das aulas do mestrado, em especial meus orientadores Victor e Rodrigo, que embarcaram juntos comigo na ideia deste projeto. Tiveram a paciência e disposição para me ajudar nos estudos dos assuntos que eu ficava com mais dúvidas, e também ajudando a buscar alternativas quando me encontrava em uma parte complexa do projeto.

Agradecer à equipe do JuáLabs, que por muito tempo fazíamos as reuniões semanais, e que ficava atento aos trabalhos dos colegas e os conselhos dos professores, aprendendo e contribuindo sempre que possível, tudo isso com um clima bem legal em que todos buscavam se ajudar.

"Nós temos que sempre navegar adiante. Longe das sombras, para a luz mais além.
Nunca feche seus olhos cansados..."
(Sail On - Masterplan)

Resumo

Obter a localização de um dispositivo remoto com maior eficiência energética é um desafio que vem sendo enfrentado com o avanço da Internet das Coisas dentro da sociedade, seja no ambiente doméstico ou profissional. Soluções tradicionais, como o *Global Positioning System* (GPS), são largamente utilizadas para resolver problemas de geolocalização, porém apresentam um consumo de energia alto para dispositivos que funcionam por meio de baterias e precisam preservar ao máximo a energia acumulada para prolongar seu tempo de atividade, além de terem um alto custo de implementação. O surgimento de protocolos e tecnologias de comunicação de baixo consumo, como o LoRa/LoRaWAN, trouxe a necessidade do uso mais eficiente da energia, inclusive para manter o baixo de custo de implementação da sua infraestrutura de comunicação. Esta dissertação busca avaliar métodos de localização usando dados coletados dos pacotes enviados dentro de uma rede LoRa/LoRaWAN, fazendo uso de aprendizagem de máquina para superar impedimentos que métodos analíticos como o *Time Difference of Arrival* (TDoA), que requer pelo menos três estações receptoras para ter um resultado satisfatório. Utilizando os dados de *Received Signal Strength Indicator* (RSSI) para os modelos RBF e kNN, e imagens com as estações receptoras plotadas utilizando a técnica de Estimativa de Densidade de Kernel (EDK) de acordo o RSSI para o modelo CNN. Foram utilizados dois cenários para a execução dos modelos, um deles contendo dados de 27 estações e outro com dados de apenas duas estações. Os resultados foram comparados com os resultados do TDoA e RSSI *fingerprinting*. O modelo RBF apresentou o melhor desempenho no cenário com 27 estações, enquanto o RSSI *fingerprinting* superou os modelos no cenário de 2 estações, com RBF ficando em segundo lugar.

Palavras-chaves: geolocalização, redes sem fio, lora, redes neurais, machine learning.

Abstract

Obtaining the location of a remote device with high energy efficiency is a challenging task faced with the advancement of the Internet of Things within society, whether in the domestic or professional environment. Traditional solutions, such as the Global Positioning System (GPS), are widely used to solve geolocation problems, however, they present high power consumption for devices that run on batteries and need to save the accumulated energy as much as possible to prolong their uptime, in addition to having a high implementation cost. The emergence of low-power communication protocols and technologies, such as LoRa/LoRaWAN, also brought the need for more efficient use of energy, including maintaining the low cost of the communication infrastructure implementation. This work seeks to evaluate localization methods using data collected from packets sent within a network LoRa/LoRaWAN, making use of machine learning to overcome impediments to analytical methods such as Time Difference of Arrival (TDoA), which requires at least three receiving stations to achieve a satisfactory result. Using the data from Received Signal Strength Indicator (RSSI) for the RBF and kNN models, and images with the stations receivers plotted using the Kernel Density Estimation (KDE) technique under the RSSI for the CNN model. Two scenarios were used to execute the models: one containing data from 27 stations and the other from only two stations. The results were compared with the results of TDoA and RSSI fingerprinting. The model RBF presented the best performance in the scenario with 27 stations, while RSSI fingerprinting outperformed the models in the 2-station scenario, with RBF coming second.

Keywords: geolocation, wireless networks, LoRa, neural networks, machine learning.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ADR	*Adaptive Data Rate*
AM	Aprendizado de Máquina
AP	*Access Point*
API	*Application Programming Interface*
Chirp	*Compressed High Intensity Radar Pulse*
CNN	*Convolutional Neural Network*
CSS	*Chirp Spread Spectrum*
dBm	decibel miliwatt
EDK	Estimativa de Densidade por Kernel
FDP	Função Densidade de Probabilidade
GPS	*Global Positioning System*
GPU	*Graphics Processing Unit*
HTTP	*Hypertext Transfer Protocol*
IoT	*Internet of Things*
KDE	*Kernel Density Estimation*
kNN	*k Nearest Neighbors*
LoRa	*Long Range*
LOS	*Line Of Sight*
LPWA	*Low Power Wide Area*
LPWAN	*Low Power Wide Area Network*
MAC	*Media Access Control*
MAE	*Mean Absolute Error*
MHz	MegaHertz

MLP	*Multilayer Perceptron*
NLOS	*Non-LOS*
OSI	*Open System Interconnections*
PER	*Periodic Exponential Regression*
RBF	*Radial Basis Function*
RBFN	Redes de funções de base radial
RNN	*Recurrent Neural Network*
RSSI	*Received Signal Strength Indicator*
SE	*Squared Exponential*
SF	*Spreading Factor*
SNR	*Signal-to-Noise Ratio*
SVR	*Support Vector Regression*
TDoA	*Time Difference Of Arrival*
ToA	*Time Of Arrival*

Sumário

1 Introdução

A localização de pessoas, animais e objetos é um requisito em diversas situações, seja para localizar um indivíduo perdido em uma floresta, um veículo roubado, ou simplesmente para rastrear objetos em serviços de entrega. Como dito em Anagnostopoulos e Kalousis (2019) e Zhang et al. (2019), o avanço da Internet das Coisas (*Internet of Things* ou IoT) aumentou a necessidade de localizar dispositivos remotos, tanto em ambientes externos como internos. Já existem soluções para resolver a maioria desses problemas, como por exemplo o uso de *Global Positioning System* (GPS), recurso comum até em dispositivos de uso pessoal, como *smartphones*, e utilizado por diversos aplicativos. Contudo, em alguns cenários os dispositivos precisam prolongar ao máximo a carga de sua bateria, e o alto consumo energético do GPS pode inviabilizar a execução do sistema, além de não ser muito eficaz em ambientes internos, sendo necessário buscar alternativas.

Em tecnologias *wireless*, de acordo com Sand, Dammann e Mensing (2014) e Anjum et al. (2020), as técnicas de localização baseiam-se principalmente nos princípios temporais e de intensidade do sinal. Utilizando a característica temporal, a localização é estimada baseando-se nos instantes de tempo envolvidos na comunicação entre dispositivo e estações receptoras do sinal, sendo a técnica *Time Difference of Arrival* (TDoA) uma das mais utilizadas nesse contexto, a qual utiliza a diferença de tempo de recepção do pacote entre as estações receptoras. A intensidade de um sinal também permite estimar a localização, sendo a técnica denominada *fingerprinting* uma das mais conhecidas, que consiste em utilizar valores de intensidade de sinal para identificar o posicionamento em uma determinada região, de modo a buscar padrões que permitam estimar a posição do objeto. Conforme Daramouskas, Kapoulas e Pegiazis (2019), uma característica de sinal bastante utilizada para se tentar localizar um objeto é o *Received Signal Strength Indicator* (RSSI), que indica a força do sinal recebido, onde quanto mais forte ele for, melhor a conexão entre o dispositivo e a estação.

De acordo com Dieng, Pham e Thiare (2019) e Janssen, Berkvens e Weyn (2020), técnicas como TDoA requerem alta precisão na sincronia dos relógios entre antenas para obter resultados em um nível de resolução de 10 metros, em que um *time stamping* de 0,3 nanossegundos é necessário. O GPS é utilizado para conseguir um *time stamping* dessa precisão. Segundo Niitsoo et al. (2019), a técnica TDoA não apresenta um bom desempenho em cenários com muitos objetos metálicos ao redor, o que pode causar problemas como reflexões de sinal e obstrução, acarretando em estimativas imprecisas. Para corrigir esses problemas, uma solução comum é instalar muitas antenas para cobrir o máximo da área desejada, o que resulta em um alto custo de implementação da infraestrutura, pois pode acontecer de instalar mais antenas que o necessário. Já a técnica de *Fingerprinting* pode

requerer uma infraestrutura mais onerosa, pois como utiliza os padrões de cada ponto da área que atua, sua performance vai depender da quantidade de *access points* (APs) instalados. Além disso, se há mudanças frequentes de organização de objetos dentro da área, pode influenciar nos padrões utilizados pela técnica, dificultando seu uso.

Para obter melhores resultados, técnicas de inteligência artificial podem ser utilizadas para auxiliar os sistemas de localização, e alguns exemplos podem ser vistos nos trabalhos Kim, Lee e Huang (2018) e Song et al. (2019), apresentando resultados de estimativa até 12,25% melhor que as outras soluções testadas, como kNN (*k-Nearest Neighbours*) e *Weighted Centroid*. Modelos de Aprendizado de Máquina (*Machine Learning*) ou Aprendizado Profundo (*Deep Learning*) podem se adaptar facilmente a mudanças no cenário que atua, pois é possível aprender novos padrões com os dados que eles são treinados. Situações em que hajam obstáculos físicos podem servir de aprendizado para os modelos, e assim aprenderem como lidar nessas circunstâncias. Outra questão importante é a quantidade de características que podem ser utilizadas para as estimativas, pois além dos dados de intensidade de sinal, quaisquer outros parâmetros das transmissões dos pacotes podem ser utilizadas para melhorar o aprendizado dos modelos de inteligência artificial.

Além das características físicas das comunicações *wireless*, também é possível utilizar outras formas de dados para treinar modelos, como imagens. Por exemplo, pode-se utilizar uma imagem que mostra o mapa da área que se deseja estimar a localização, com informações de coordenadas geográficas, e partir delas treinar modelos com esse recurso. No contexto de dispositivos e estações receptoras, é essencial saber quais delas receberam a mensagem, para assim possibilitar a estimativa da localização do dispositivo emissor. Para isso, essas imagens podem ser geradas usando técnicas como a Estimativa de Densidade por Kernel (EDK) ou *Kernel Density Estimation* (KDE), que ao ser aplicada nos dados, ajuda a destacar os pontos de maior relevância, ou *hot spots*, que podem indicar, por exemplo, com qual intensidade de RSSI a estação recebeu o pacote.

1.1 Motivação

Conforme dito em Devalal e Karthikeyan (2018), há bilhões de dispositivos IoT em funcionamento no mundo, os quais utilizam diferentes tecnologias de comunicação. Uma tecnologia *wireless* que vem ganhando espaço em projetos, segundo Gu, Jiang e Tan (2019), é a LoRa, ou *Long Range*. LoRa é uma tecnologia do tipo *Low Power Wide Area Network* (LPWAN), tecnologias de comunicação atuantes em grandes áreas com baixo consumo de energia, que utiliza um tipo de sinal, denominado *Chirp*, para determinar padrões que possam ser utilizados para codificar e decodificar as mensagens enviadas. Sua popularidade se deve principalmente pela simplicidade e baixo custo na implementação e eficiência energética.

Funcionando junto com o LoRa existe o LoRaWAN, o qual define elementos como um protocolo de comunicação para o LoRa e uma arquitetura de rede. Com isso, os projetos dispõem de referências e padronizações para seu desenvolvimento. Tipicamente, de acordo com Semtech (2024), uma comunicação LoRa tem um alcance de 5 quilômetros em áreas urbanas e de 15 quilômetros em áreas rurais (campos abertos sem grandes obstáculos). Um recorde[1] de distância ao enviar uma mensagem utilizando LoRa/LoRaWAN foi de 730.360 km, em uma comunicação entre o planeta Terra e a Lua ocorrido em 2021. Para mensagens entre elementos na superfícies terrestre, um recorde[2] de distância em uma transmissão bem sucedida foi de 832 km entre a estação e o dispositivo, registrado em 2020.

Segundo Groombridge (2022), tecnologias *wireless* são uma tendência tecnológica estratégica para as organizações no atual contexto, envolvendo viabilizar, dentre outros aspectos, o fornecimento de dados de localização em tempo real. Como dito em Schieltz et al. (2017), campos como o de rastreio animal demanda bastante dos serviços de localização remoto. Seja em uma situação de rastreio de animais com risco de extinção, de espécies terrestres, aéreas ou aquáticas, os ambientes que esses animais vivem podem ser hostis a um nível que demande um sistema mais resiliente. Logo, desenvolver projetos que consigam manter sua eficácia em cenários adversos é deveras importante para a continuidade dessas ações que visam preservar a natureza.

Outra situação é a de manejo de animais com finalidade pecuária, por exemplo, em que os grupos de animais precisem ser monitorados, como visto em Dieng, Pham e Thiare (2019). O uso de soluções tradicionais, como GPS, tem um custo elevado para implementar, pois um rebanho pode conter centenas de animais. O LoRa se mostra como uma solução viável, tanto no contexto financeiro quanto no tecnológico, pois com baixo custo de implementação é possível desenvolver um sistema para que se possa rastrear, em tempo real, grandes grupos de animais criados a pasto por exemplo. Em conjunto, modelos de inteligência artificial podem contribuir nas estimativas de localização com a análise dos dados e padrões, indo além do que métodos analíticos conseguem prover, ao mesmo tempo em que não requer um alto custo de implementação e apresenta boa escalabilidade.

1.2 Objetivos

O objetivo deste trabalho é de propor e avaliar o desempenho de modelos baseados em aprendizado de máquina para a localização de dispositivos em redes LoRa e comparar com métodos analíticos clássicos.

E ainda como objetivos específicos, pretende-se:

[1] Recorde Lua - https://www.landmobile.co.uk/news/new-world-record-for-lora-comms/
[2] Recorde Terrestre - https://www.thethingsnetwork.org/article/lorawan-world-record-broken-twice-in-single-experiment-1

- Elaborar um conjunto de dados e imagens adequados para a avaliação dos modelos;
- Analisar as estruturas e fundamentos dos modelos selecionados;
- Avaliar o impacto de técnicas de seleção de valores de hiperparâmetros na qualidade dos modelos utilizados;
- Comparar os resultados obtidos dos modelos de aprendizagem de máquina com técnicas clássicas para solução do problema de localização;

1.3 Organização do trabalho

Este trabalho está organizado em seis Capítulos, sendo o primeiro de introdução. O Capítulo 2 aborda os trabalhos relacionados. O Capítulo 3 apresenta o referencial teórico sobre princípios básicos de localização de dispositivos em redes *wireless*, LoRA e LoRaWAN, Aprendizado de Máquina, Aprendizado Profundo, Busca Aleatória, kNN (*k-Nearest Neighbors*), RBFN (*Radial Basis Function Networks* ou Redes de Função de Base Radial) e CNN (*Convolutional Neural Networks* ou Redes Neurais Convolucionais). O Capítulo 4 descreve os materiais e métodos utilizados no desenvolvimento do projeto, incluindo o preprocessamento dos dados, modelos para a busca aleatória, métricas de avaliação de resultados e o formato das imagens geradas. O Capítulo 5 apresenta e discute os resultados obtidos. No Capítulo 6 tem-se as conclusões e considerações finais sobre este trabalho, e também são apresentadas ideias para trabalhos futuros.

2 Referencial Teórico

Neste Capítulo serão demonstrados os conceitos para a devida compreensão dessa dissertação. Na Seção 2.1 são explicados os fundamentos do posicionamento *wireless*, aprofundando com técnicas baseadas nos princípios de Tempo e Sinal, respectivamente, nas Seções 2.1.1 e 2.1.2. Na Seção 2.2 é detalhada a estrutura e funcionamento da tecnologia LoRa e também do LoRaWAN.

Em seguida são apresentados os modelos de AM, começando pelo modelo k-*Nearest Neighbors* na Seção 2.3. Os princípios de redes neurais é abordado na Seção 2.4, servindo como base para o melhor entendimento do modelo de Redes de Funções de Base Radial, apresentado na Seção 2.4.1. A Seção 2.4.2 demonstra o conceito de Aprendizado Profundo, auxiliando na apresentação do modelo de Redes Neurais Convolucionais, explicada na Seção 2.4.2.1.

Finalmente, as técnicas que auxiliam tanto na execução dos modelos quanto para a geração de imagens são abordadas. Na Seção 2.4.3 são demonstrados os principais otimizadores de pesos de redes neurais. Já na Seção 2.5 é explicado a técnica de otimização de hiperparâmetros chamada de Busca Aleatória. A Seção 2.6 encerra o capítulo ao explicar o funcionamento da Estimativa de Densidade de Kernel, utilizada na geração das imagens que serviram de entrada para o modelo CNN.

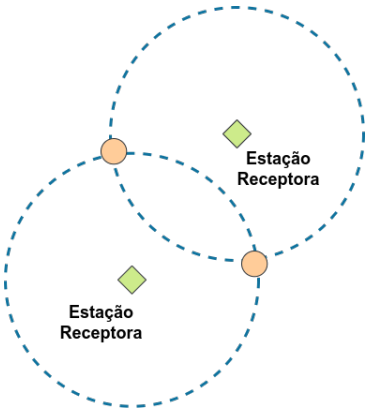
2.1 Fundamentos de localização espacial com tecnologias *wireless*

Estimar a posição de algo no espaço requer observar algumas grandezas físicas. No contexto de tecnologias *wireless*, a principal grandeza a ser observada são formas de ondas eletromagnéticas, e também alguns parâmetros que possam influenciá-las. De acordo com Sand, Dammann e Mensing (2014), o atraso na propagação das ondas, a distância do objeto a ser localizado e os obstáculos presentes no trajeto das ondas são algumas características que podem influenciar na precisão da localização.

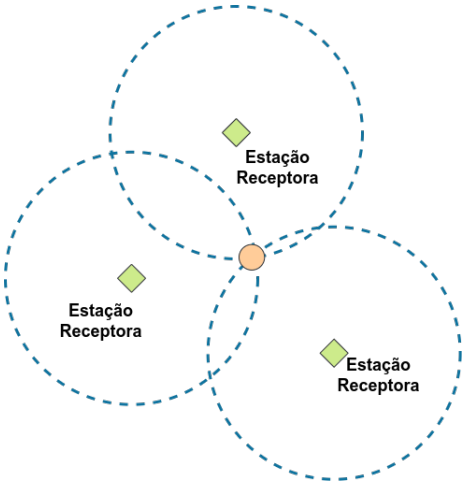
A Equação 2.1, como visto em Sand, Dammann e Mensing (2014), demonstra uma relação funcional entre a forma de onda s, tempo t e posição no espaço $\mathbf{x}$. Como a observação r é influenciada pelos ruídos inerentes do ambiente, que geralmente é um processo randômico gaussiano, a posição $\mathbf{x}$, que é inicialmente desconhecida, necessita ser estimada.

$$f : (s, \mathbf{x}, t) \mapsto r \tag{2.1}$$

(a) TDoA com 1 estação



(b) TDoA com 2 estações



(c) TDoA com 3 estações

Figura 1 – Operação do TDoA com diferentes quantidades de estações receptoras.

A complexidade da função f vai depender do ambiente em que está o dispositivo a ser localizado. Em um cenário LOS (*Line of Sight*), onde não há obstáculos para a propagação das ondas, a função pode ser mais simples. No entanto, é necessária atenção em regiões com padrão de radiação omnidirecional ao utilizar o TDoA, pois como demonstrado na Figura 1, é perceptível as ambiguidades existentes quando há menos de 3 estações. Com apenas 1 estação, a distância do objeto para a estação é a mesma em qualquer ponto situado na linha tracejada. Com 2 estações, existem dois pontos de interseção que relaciona as 2 estações, podendo o objeto estar em um deles.

Já em um cenário NLOS (*Non-LOS*), o qual contempla alguns obstáculos na propagação das ondas, a complexidade da função aumenta. Nesse caso, é essencial saber as posições e propriedades de elementos como bases transmissoras, pois eles guardam dados sobre as transmissões realizadas. Esses dados funcionam como um *fingerprint* das distintas posições, ou seja, uma combinação única de dados para representar cada posição, e servem como parâmetros para técnicas de *fingerprinting*.

De forma mais abstrata, há dois métodos de localização nesse cenários. Um deles é Auto-localização, em que o receptor de posicionamento capta medições de sinal adequadas de transmissores espalhados geograficamente e, em seguida, usa essas medições para determinar sua própria localização. Um exemplo de Auto-localização é o GPS, onde comumente se busca determinar a localização que o dispositivo se situa. Outro é o Posicionamento de rede, onde um sinal é emitido por um dispositivo cuja posição deseja-se estimar, e é recebido por uma estação, de localização conhecida. Nesse caso, a posição é calculada em uma das estações da rede. Exemplos de técnicas para Posicionamento de rede é o TDoA e *Fingerprinting*.

2.1.1 Tempo de propagação

O princípio de tempo baseia os métodos que utilizam o tempo de propagação para estimar a posição de um dispositivo, os quais utilizam a distância geométrica entre as estações receptoras e o dispositivo em si. A troca de dados entre esses elementos permite calcular o tempo de propagação do sinal de envio e de retorno, e juntando os tempos de mais estações receptoras para o mesmo dispositivo, é possível inferir a posição dele. Para isso, há duas premissas, sendo uma delas a de que a velocidade de propagação do sinal é conhecida, e de um cenário LOS, ou seja, não haver obstáculos no caminho do sinal que o desviem, como objetos refletores ou refratores.

Como visto em Daramouskas, Kapoulas e Pegiazis (2019), um método bastante utilizado é o *Time of Arrival* (ToA), que utiliza o tempo de propagação do sinal entre a estação receptora e o dispositivo. Ele utiliza um sistema de equações não lineares para determinar a posição de um dispositivo na rede, como visto nas Equações 2.2, onde n é o número de estações receptoras, (x, y, z) são as coordenadas do dispositivo e (x_i, y_i, z_i), onde $i = 1 \dots n$, são as coordenadas de cada estação receptora em um cenário com três dimensões, $T_n - T_0$ é o tempo de propagação do sinal correspondente a distância d_n percorrida e c é a velocidade da luz.

$$\begin{aligned}
\sqrt{(x-x_1)^2+(y-y_1)^2+(z-z_1)^2} &= c(T_1-T_0) = d_1,\\
\sqrt{(x-x_2)^2+(y-y_2)^2+(z-z_2)^2} &= c(T_2-T_0) = d_2,\\
&\vdots\\
\sqrt{(x-x_n)^2+(y-y_n)^2+(z-z_n)^2} &= c(T_n-T_0) = d_n
\end{aligned} \tag{2.2}$$

Para uma estimativa mais precisa, é necessário pelo menos três equações, o que significa que pelo menos três estações conseguiram se comunicar com o dispositivo. Contudo, não existe uma solução para esse sistema, pois em cenários reais é comum haver ruídos, fazendo com que haja um valor de erro a se considerar na posição estimada. Esses ruídos

podem ser obstáculos no ambiente que atrapalhem a propagação do sinal, reflexões do sinal causando múltiplas chegadas do sinal na estação, ocasionando interferência. Para o ToA, garantir a sincronia de relógios entre as estações e os dispositivos é mandatório, sendo esse um requisito desafiador para garantir, pois é possível que, por exemplo, algum desses elementos seja ineficiente para manter seu relógio ajustado.

Outro método, que tem relação com o ToA, é o *Time Difference of Arrival* (TDoA). Seu objetivo é medir a diferença nos tempos de propagação do sinal vindos das estações base. Os sinais, quando emitidos, chegam a um dispositivo receptor cada um a seu tempo, e a diferença entre eles irá ser utilizada para inferir a posição deste dispositivo. Uma vantagem é a de que as estações e o dispositivo não precisam estar sincronizados, apenas as estações entre elas. Isso porque a diferença dos tempos será feita no dispositivo, então estarão na mesma base temporal. Na Equação 2.3 é demonstrado o cálculo da diferença das distâncias de propagação entre duas estações base i e j, onde T_i e T_j, respectivamente, são os tempos de chegada no dispositivo, e T_0 é o tempo de início do envio do sinal. O TDoA não compartilha de algumas vulnerabilidades do ToA, como por exemplo o problema de múltiplas chegadas, dado que o TDoA usa apenas a diferença no tempo de chegada entre as estações, e mesmo que o sinal chegue por mútiplos caminhos, essa diferença permanece a mesma. Contudo, alguns ruídos que afetam o ToA podem afetar também o TDoA, como obstáculos que atrapalham a propagação do sinal.

$$\Delta d_{i,j} = d_i - dj = c(T_i - T_0) - c(T_j - T_0) = c(T_i - T_j) = c * \Delta T_{i,j} \tag{2.3}$$

2.1.2 *Fingerprinting*

Esse método parte de uma premissa de que os sinais transmitidos ou recebidos apresentam propriedades únicas em cada posição dentro do ambiente, de modo que é possível analisar tais propriedades para inferir essa posição. Para isso, é comum criar um banco de dados mapeando as características dos sinais para cada posição do ambiente, onde cada mapeamento desse é chamado de *fingerprint*. Logo, quando se almeja localizar um dispositivo, o sinal é analisado, e os dados são comparados com os *fingerprints* previamente armazenados para predizer a localização com base na similaridade entre eles. Esse banco de dados pode ser criado usando características como o RSSI.

A Equação 2.4 demonstra a relação de proporcionalidade entre a potência de recebimento P_{RX} e a potência de envio P_{TX}, ganho de envio G_{TX} e de recebimento G_{RX}. O d é a distância entre o transmissor e receptor, e o d_0 é uma distância de referência. O β é conhecido como o fator de decaimento, e ele vai depender do ambiente. Segundo Sand, Dammann e Mensing (2014), em ambientes livres, o β é pelo menos 2, enquanto em ambientes NLOS é comum ele assumir valores entre 3,5 e 3,8.

$$P_{RX} \ \alpha \ P_{TX} G_{TX} G_{RX} (\frac{d}{d_0})^{-\beta} \tag{2.4}$$

2.2 LoRa

A tecnologia LoRa (*Long Range*) foi desenvolvida pela empresa Semtech[1], tendo como principal característica ter um longo alcance de transmissão consumindo poucos recursos. Ela consiste em uma camada física de comunicação, de acordo com o modelo *Open System Interconnections* (OSI), que utiliza padrões nos sinais transmitidos para conseguir enviar e receber dados. Isso é possível graças a uma técnica baseada em *Spread Spectrum*, chamada de *Chirp Spread Spectrum* (CSS).

Como dito em Montagny (2022), o *Spread Spectrum* é um método de transmissão em que os dispositivos enviam os dados ao mesmo tempo no mesmo canal, e para isso, utiliza códigos como símbolos para que o receptor consiga decodificar a mensagem. Na maioria dos seus métodos de modulação, ele utiliza códigos para desempenhar suas funções. Esses códigos são sequências de números, ortogonais entre si, e são utilizados para codificar e decodificar os dados enviados.

O LoRa utiliza uma técnica chamada de *Chirp Spread Spectrum*, o qual utiliza *Chirps* como símbolos em vez de códigos nas transmissões. O Chirp (*Compressed High Intensity Radar Pulse*), como visto em Devalal e Karthikeyan (2018), é um sinal que aumenta e diminui sua frequência com o tempo, e esses estados são chamados de *up-chirp* e *down-chirp*, respectivamente. Na Figura 2 é mostrado um exemplo gráfico de um sinal gerado por um Chirp, onde em 10 segundos o sinal passou da frequência de $10Hz$ para $1Hz$.

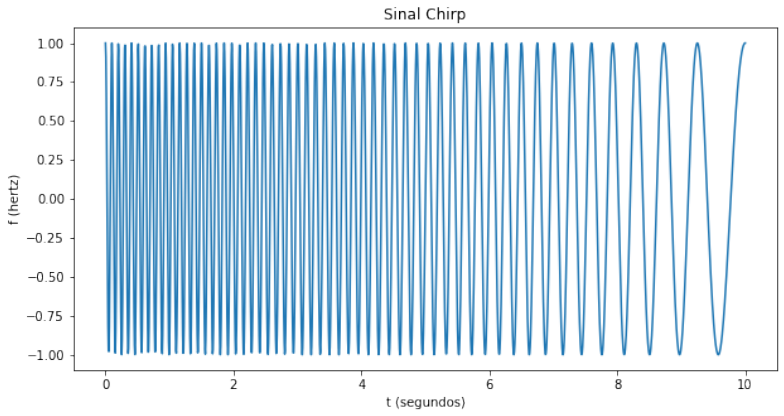


Figura 2 – Representação gráfica de um exemplo de sinal de Chirp.

[1] Semtech - https://www.semtech.com/lora

Cada chirp, também chamados de símbolos, iniciam de um valor de frequência distinto. A frequência que um Chirp pode assumir depende da frequência do canal e da largura de banda, podendo iniciar de $F_{canal} - \frac{BW}{2}$ e terminar até $F_{canal} + \frac{BW}{2}$, onde F_{canal} é a frequência de canal utilizada e BW é a largura de banda escolhida. A faixa de frequências permitidas no Brasil é de $902MHz$ a $907{,}5MHz$ e de $915MHz$ a $928MHz$, determinada pela Agência Nacional de Telecomunicações (ANATEL) em seu ato de número 14.448 de 2017[2]. Na Tabela 1 estão alguns exemplos de faixas de frequências[3] utilizadas em diferentes países. Os valores de largura de banda utilizados pelo LoRa são de $125kHz$, $250kHz$ e $500kHz$.

Código	**Faixa de frequência (MHz)**	**Exemplos de países que adotam**
AU915-928	915 a 928	Austrália e Brasil*
US902-928	902 a 928	EUA e Canadá
EU863-870	863 a 870	Países da Europa

Tabela 1 – Frequências utilizadas em diferentes regiões.
* Brasil dispõe também das frequências 902-907,5 MHz

A quantidade de bits transmitidos em cada Chirp vai depender do valor de um parâmetro chamado Fator de Espalhamento ou *Spreading Factor* (SF). O LoRa suporta seis níveis de SF, do SF7 ao SF12, e esses números determinam também quantas formas diferentes um símbolo pode assumir, que é 2^{SF}. O SF também influencia no tempo da transmissão e no alcance. Quanto menor o SF, menor o alcance, contudo a transmissão se dá de forma mais rápida e consumindo menos energia. Um SF maior consome mais energia e tem uma transmissão mais lenta, porém o alcance é maior. Logo, a decisão por qual SF usar vai depender da situação da área em que a rede LoRa esteja operando. Seu valor também pode ser determinado automaticamente utilizando o *Adaptive Data Rate* (ADR), um mecanismo que verifica o valor da potência que o receptor registrou ao receber os dados, e com isso configura um valor adequado para o SF.

2.2.1 LoRaWAN

O LoRaWAN é um protocolo de comunicação de rede *Low Power Wide Area* (LPWA) que funciona sobre a tecnologia LoRa. Ele é padronizado pela LoRa Alliance[4], que também define como deve ser a arquitetura dos projetos que venham a utilizar esse protocolo. São quatro elementos básicos nesse tipo de rede, sendo eles os dispositivos, gateways, servidor de rede e servidores de aplicação. De acordo com o modelo OSI, o

[2] Ato 14.448/2017 da Anatel - https://informacoes.anatel.gov.br/legislacao/atos-de-certificacao-de-produtos/2017/1139-ato-14448
[3] Frequências LoRa - https://www.thethingsnetwork.org/docs/lorawan/frequency-plans/
[4] LoRa Alliance - https://lora-alliance.org/about-lorawan/

LoRaWAN está localizado na subcamada *Media Access Control* (MAC), dentro da camada de enlace. Na Figura 3 é demonstrado um exemplo de arquitetura de uma rede LoRaWAN.

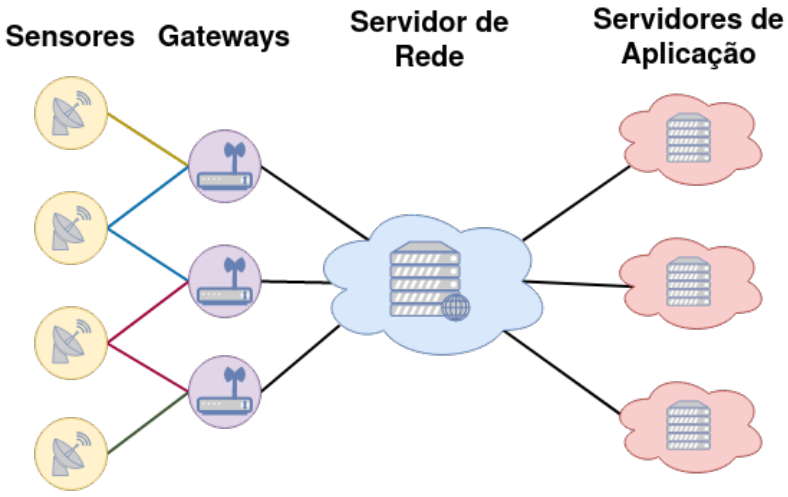


Figura 3 – Exemplo de arquitetura LoRaWAN.

A topologia utilizada para o LoRaWAN é do tipo estrela, em que vários nós da rede estão conectados a um ponto comum, o qual seria um subconjunto dos gateways disponíveis. Os dispositivos sempre estarão conectados com todos os gateways que estiverem a seu alcance, e assim conseguirem enviar seus dados. São definidas três classes diferentes para os dispositivos, sendo elas Classe A, Classe B e Classe C.

Os dispositivos Classe A (ou *All*) são os mais econômicos em relação a energia, e todos os dispositivos LoRa se enquadram nessa categoria, pelo menos. Seu modo de operar é, logo após enviar os dados, duas janelas de tempo são acionadas para que o dispositivo possa receber dados. Os intervalos das janelas, segundo Devalal e Karthikeyan (2018), devem ser longos o suficiente para detectar o preâmbulo da mensagem enviada, e são esses os únicos momentos em os gateways podem enviar dados para o dispositivo. Caso não consiga, deve esperar o próximo envio de dados por parte do dispositivo para que as janelas se abram novamente. Nos dispositivos Classe B (ou *Beacon*), além das janelas de recebimento definidas pela Classe A, mais janelas são criadas, sendo essas agendadas. Com as janelas agendadas, o servidor consegue saber quando o dispositivo está pronto para receber dados. Os da Classe C (ou *Continuous*), por fim, estão sempre preparados para receber dados, com exceção de quando estão enviando os dados. Por isso, dispositivos Classe C têm baixa eficiência energética, pois aguardar a recepção de dados é um estado que consome mais energia.

Os gateways são os elementos que estão diretamente conectados com os dispositivos, e também com o servidor de rede. Do lado dos dispositivos, os gateways estabelecem uma conexão LoRa para a troca de dados, e também dispõem de conexão com a internet, onde os dados são repassados para um servidor de rede, geralmente funcionando na nuvem.

O servidor de rede é o elemento em que está toda a inteligência do sistema. Por

exemplo, é nele que funciona o ADR, repassando para os gateways a necessidade ou não de alterar o SF. Ele também vai verificar os dados recebidos e repassar para os servidores de aplicação pertinentes. Os servidores de aplicação recebem os dados e suprem os sistemas que os utilizam.

2.3 k-*Nearest Neighbors* (kNN)

O k-*Nearest Neighbors* (kNN) é um algoritmo de AM não paramétrico bastante utilizado em casos de classificação e regressão. Ele se baseia na proximidade dos exemplos de treinamento no espaço de características. Segundo Bishop (2006), o kNN é um dos métodos não paramétricos mais simples, pois não requer treinamento. O seu funcionamento se baseia em determinar a classe de um dado elemento de acordo com as classes mais comuns entre seus vizinhos mais próximos, onde o valor k determina a quantidade desses vizinhos.

Uma das principais considerações ao usar o kNN é a escolha adequada do valor de k. Um k muito pequeno pode levar a uma classificação ou previsão instável, pois estaria muito suscetível a ruídos nos dados. Por outro lado, um k muito grande pode levar a uma perda de sensibilidade a padrões locais. A escolha do valor de k depende do conjunto de dados e do problema específico. A proximidade dos vizinhos é calculada de acordo com uma métrica, a ser escolhida no momento da execução do modelo.

O kNN também pode ser utilizado em casos de regressão. Diferente dos problemas de classificação, as entradas para problemas de regressão não possuem "classes", mas sim cada amostra representa um conjunto de valores (ou características) e um valor alvo associado. Essas características, por exemplo, podem ser o valor do RSSI, e o valor alvo a coordenada geográfica, trazendo para o contexto deste trabalho. O processo consiste em obter as k menores distâncias entre os vizinhos e calcular uma média dos valores alvo respectivos de cada um deles. O resultado da média será a estimativa. A fórmula da média é mostrada na Equação 2.5. O y_i é a distância calculada entre o dado e um vizinho e o k é a quantidade de vizinhos considerados para o cálculo.

$$\hat{y} = \frac{1}{k} \sum_{i=1}^{k} y_i \tag{2.5}$$

2.4 Redes Neurais

Conforme Glassner (2021), "neurônios" ou simplesmente "unidades", são os nós presentes em uma rede neural. Na Figura 4 é mostrada a estrutura de um neurônio. Dispondo de múltiplas entradas, para cada uma delas haverá um peso correspondente, e na soma desses resultados é acrescido um valor chamado viés ou *bias*, que corresponde a

um erro sistemático. Esse valor é então submetido a uma função, chamada de "função de ativação", que proverá o resultado da saída do respectivo neurônio.

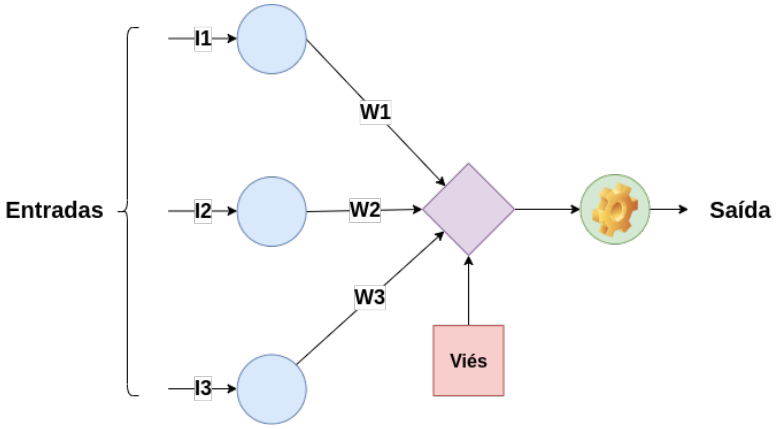


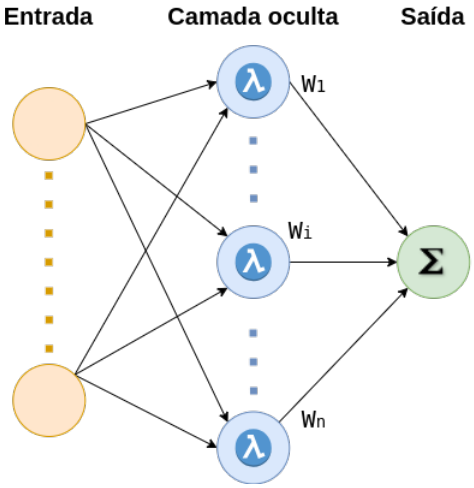
Figura 4 – Estrutura de um neurônio.

Neurônios podem ser conectados entre si, de modo que a saída de um pode ser a entrada de outro. Com um conjunto de neurônios conectados se forma uma rede neural. Uma forma comum é organizá-los em camadas, de modo que a saída de uma camada serve como entrada da outra seguinte. No contexto de redes neurais, os "pesos" são valores internos da rede e são reajustados durante o treinamento do modelo devido ao seu aprendizado, como por exemplo os pesos das conexões entre os neurônios, os quais determinam a força de influência entre eles.

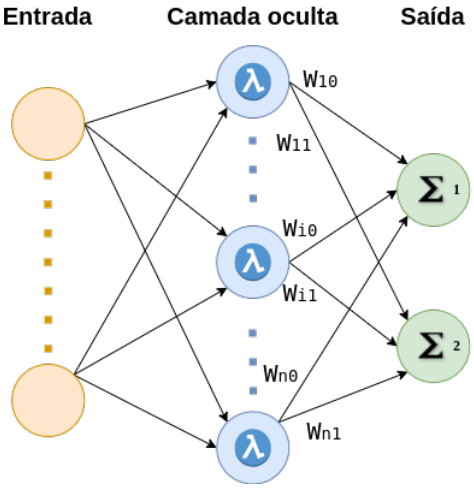
2.4.1 Redes de Funções de Base Radial (RBFN)

Em Stella, Russo e Šarić (2014), é dito que Redes de funções de base radial, ou redes RBF, é um tipo de rede neural *feedforward*, treinada com um algoritmo de treinamento supervisionado. Redes *feedforward* tem como característica ter todas suas camadas conectadas, porém sem um caminho de volta, ou seja, todas as conexões tem a mesma direção, partindo da camada de entrada até a camada de saída. Com relação à camada de saída, essa pode conter desde apenas um neurônio ou vários, com cada um deles dispondo de seus respectivos pesos em relação à camada oculta que lhe antecede. A principal diferença de uma RBFN para outros modelos é a presença da camada oculta com funções de base radial. Além disso, as RBFN pertencem ao grupo de redes de função de kernel, ao passo que suas entradas são submetidas a uma função de kernel, limitando suas saídas a regiões específicas do espaço de sua entrada, sendo comum o uso da função Gaussiana.

Conforme Liu (2013), RBFN são constituídas de uma camada de entrada, camadas ocultas, e uma camada de saída, como é visto na representação da Figura 5. Para realizar a extração de características, na camada oculta os neurônios são ativados por uma função de base radial não linear, e cada um deles possui um vetor central c de mesma dimensão



(a) RBF saída única



(b) RBF saída dupla

Figura 5 – Representações gráficas de RBFN.

que o vetor de entrada x. O intuito é verificar a similaridade entre os valores de ambos os vetores, e quanto maior a similaridade, maior o peso atribuído àquele neurônio, auxiliando assim na tarefa de interpretar os padrões existentes. Em uma rede *Multilayer Perceptron* (MLP), por exemplo, a extração de características é executada reajustando os pesos durante o treinamento, realizando uma combinação linear das entradas ponderadas pelos pesos, seguida pela aplicação de uma função de ativação. Enquanto uma MLP têm maior flexibilidade para aprender representações complexas e não lineares, uma RBFN é mais interpretável e geralmente aplicada a tarefas específicas de reconhecimento de padrões.

Os pesos da camada de saída podem ser determinados usando técnicas como o Método dos Mínimos Quadrados ou Mínimos Quadrados Recursivos, como visto em Bishop (2006). A Equação 2.6 demonstra como é a camada de saída de uma RBFN, uma combinação linear das funções de base radial, onde x é um vetor de entrada, x_n é um vetor central, n é o tamanho do vetor de entrada, w_n é o respectivo peso, e $h()$ a função de base radial utilizada. Dado um vetor de valores alvo t, a intenção é encontrar um $f(\mathrm{x})$ para que $f(x_n) = t_n$.

$$f(\mathrm{x}) = \sum_{n=1}^{N} w_n h(\|\mathrm{x} - \mathrm{x}_n\|) \tag{2.6}$$

Na Equação 2.7 é demonstrado como é gerada a saída da camada oculta, onde b_i é a largura da função gaussiana do neurônio i e n o número de neurônios da camada oculta. A expressão $\| \mathrm{x} - \mathrm{c}_i(t) \|$ calcula a distância euclidiana entre o vetor c e o vetor de entrada x. O Algoritmo 1 representa o pseudocódigo do processo de treinamento de uma RBFN clássica.

$$h_i(t) = \exp(-\frac{\| \mathrm{x}(t) - \mathrm{c}_i(t) \|}{2b_i^2}), \ i = 1, ..., n \tag{2.7}$$

Algorithm 1 Treinamento de RBFN

1: **Entrada:** Conjunto de dados $\mathrm{X} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, número de centros K, função de base radial h
2: Inicializar os centros $\mathbf{c}_k$ aleatoriamente ou usando um método específico
3: Calcular a matriz de ativação $\Phi = h(\mathbf{x}_i, \mathbf{c}_k)$
4: Resolver o sistema linear $\Phi\mathbf{w} = \mathrm{T}$ para obter os pesos $\mathbf{w}$

2.4.2 Aprendizado Profundo

Um modelo de rede neural que possua muitas camadas conectadas entre si formam uma estrutura denominada de rede profunda, surgindo assim o conceito de Aprendizado Profundo ou *Deep Learning*, segundo Glassner (2021). O Aprendizado Profundo permite analisar os dados de forma hierárquica, onde cada camada avalia de forma crescente os conceitos presentes nos dados. Por exemplo, ao analisar uma imagem de um automóvel por completo, as camadas iniciais podem tentar identificar primeiro linhas, depois curvas e assim sucessivamente, até poder inferir que uma parte da imagem é um pneu.

2.4.2.1 Redes Neurais Convolucionais

As Redes Neurais Convolucionais, ou *Convolutional Neural Network* (CNN), são um tipo de rede neural de aprendizado profundo apropriada para aprender com dados de entrada na forma de matrizes, sendo comum seu uso em processamento de imagens, porém pode ser utilizada com outros tipos de dados, como dados espaciais ou temporais, como dito em Aggarwal (2018). Vale ressaltar a operação matemática chamada de "convolução", que no contexto de uma CNN consiste em realizar um produto escalar entre uma matriz de pesos e as matrizes de dados, que são as entradas. As entradas de uma CNN seguem o formato "Largura x Altura x Profundidade", onde a profundidade, por exemplo, costuma indicar, no tocante a imagens, os canais de cores.

Na Figura 6 é demonstrada uma arquitetura básica de uma CNN. Alguns tópicos são essenciais para entender o processo, como *Pooling*, *Stride*, *Padding* e Filtros de *kernel*. O *Pooling* é definido como uma estrutura em matriz, no qual ele vai percorrer os dados do início ao fim, coletando a média dos valores (*average pooling*), ou apenas o maior (*max pooling*), da área que ele está cobrindo no momento. Esse valor será utilizado para compor a estrutura de matriz de saída da camada. O tipo do *pooling* é determinado pelo usuário. Sua principal função é reduzir o mapa de características da rede, generalizando os dados e permitindo a rede obter resultados satisfatórios mesmo com leves alterações nas entradas.

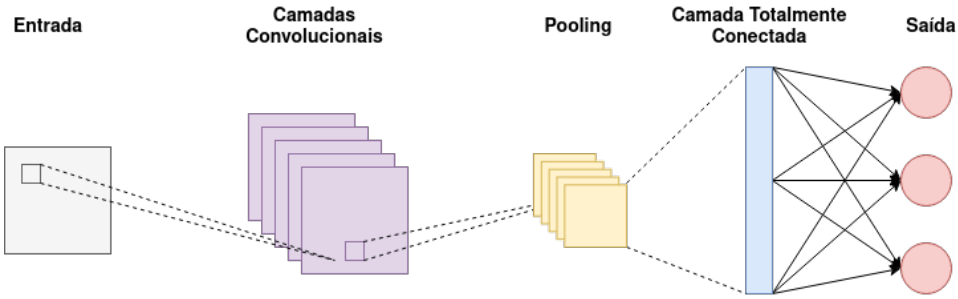


Figura 6 – Exemplo de uma arquitetura de CNN.

Na Figura 7 é demonstrado um exemplo de como é realizado um *max pooling*. O deslocamento do *Pooling* é definido pelo *Stride*, indicando quantas células das matrizes de dados a matriz do *Pooling* vai percorrer a cada iteração. O *Padding* adiciona valores nas bordas das entradas de dados, pois a cada operação de convolução, as estruturas dos dados vão ficando menores, e com isso perdendo informação.

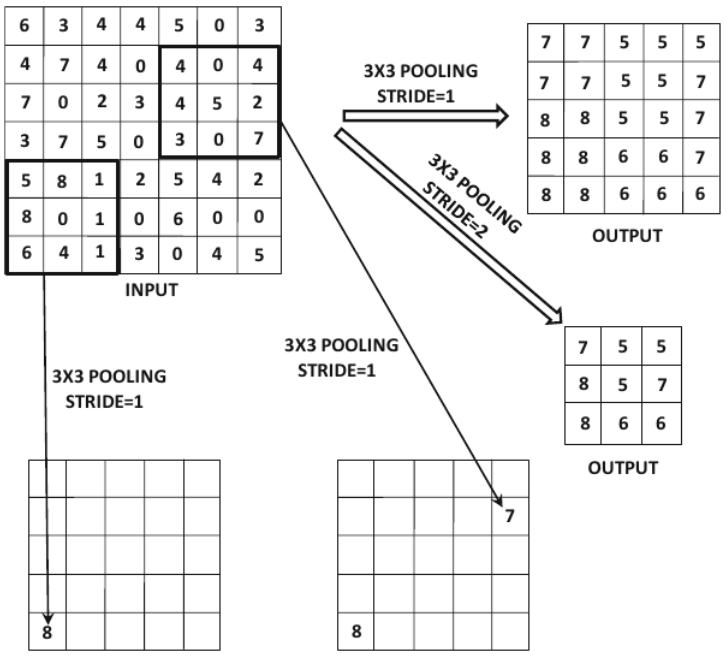


Figura 7 – Exemplo de *Max Pooling* com diferentes valores de *Stride*. Fonte: (AGGARWAL, 2018).

Os Filtros de *kernel* são os pesos da CNN, matrizes geralmente quadradas, com dimensões menores que as camadas da rede, porém com profundidade de mesmo valor. Seu tamanho é definido pelo usuário, e é responsável por realizar as convoluções na rede. Na Equação 2.8 é demonstrado a relação dos elementos para gerar a saída de uma camada, onde P é o *Padding*, F é o Filtro de *kernel*, S é o *Stride* e N_O e N_I são os tamanhos das matrizes de saída e entrada, respectivamente.

$$N_O = \left\lfloor \frac{N_I + 2P - F}{S} + 1 \right\rfloor \tag{2.8}$$

A rede pode conter várias sequências de camadas convolucionais e *Pooling* em sua composição, onde a saída de uma parte serve como entrada para outra. A camada Totalmente Conectada é uma rede *feedforward*, podendo ser apenas uma ou várias camadas do tipo. A saída de uma CNN vai depender do tipo do problema, se é classificação ou regressão, e vai ser composto de quantas unidades forem precisas para a resolução do problema em questão.

2.4.3 Otimizadores

Um otimizador é um algoritmo que treina os pesos da rede para minimizar o erro, pois no caso de uma regressão ele aprende a função-alvo subjacente aos dados. O primeiro otimizador proposto foi o Gradiente Descendente, que utiliza gradientes para encontrar o menor erro. Esse e outros serão abordados nesta seção.

2.4.3.1 Gradiente Descendente

Conforme Glassner (2021), gradiente é uma generalização de uma derivada em três ou mais dimensões, e com ele podemos identificar mínimos na sua superfície, que é bastante útil para treinamentos de modelos em redes neurais. A função de erro apresenta uma forte dependência não linear em relação aos pesos, logo irão existir muitos pontos no espaço de pesos em que o gradiente irá "sumir" (será muito próximo de 0), e isso dificulta a busca pelos mínimos. Fazendo uma analogia visual, seria como se o gradiente estivesse em um platô, sem subidas ou descidas, onde o gradiente considerasse que aquele local é o mínimo que ele consegue encontrar. Contudo, os algoritmos recentes dispõem de recursos para resolver essa situação caso ela ocorra.

Ao utilizar os pontos do vetor de pesos para encontrar o valor mínimo da função de erro, caso esse valor seja o menor de todos os pontos, ele é considerado o mínimo global. Consequentemente, todos os outros mínimos são considerados mínimos locais. Segundo Bishop (2006), uma aplicação com redes neurais não precisa encontrar o mínimo global para ser considerada eficaz, mas é necessário comparar vários mínimos locais para escolher o menor. Essa busca pelo menor erro também pode-se dizer que é o aprendizado.

Para utilizar esse conhecimento no contexto de redes neurais, pode-se utilizar uma técnica como a de Gradiente Descendente. Essa técnica busca avançar com pequenos passos em direção contrária ao gradiente de uma função de erro, onde a cada iteração do algoritmo, o vetor de pesos atual é reavaliado com o novo vetor calculado, repetindo novamente o processo, sempre movendo o vetor de pesos para a maior taxa de decrescimento da função de erro. Na Equação 2.9 é demonstrado o cálculo de cada iteração do gradiente descendente, onde w é vetor de pesos, τ é o passo de iteração, η é a taxa de aprendizagem e $E()$ é a função de erro escolhida. O sinal negativo denota que o algoritmo segue a direção

de maior queda do erro, além também de demonstrar o que ocorre caso o gradiente suma, que torna a influência do aprendizado quase nula.

$$\mathrm{w}^{(\tau+1)} = \mathrm{w}^{(\tau)} - \eta \nabla E(\mathrm{w}^{(\tau)}) \tag{2.9}$$

2.4.3.2 Gradiente Descendente Estocástico

Baseado no Gradiente Descendente, uma técnica chamada de Gradiente Descendente Estocástico busca estimar o vetor gradiente da função, ∇C, calculando um vetor gradiente para um subconjunto de entradas, ∇C_x, escolhidas de forma aleatória. Ele é estocástico pois não tem acesso a todo o conjunto de treinamento, logo o aprendizado é executado com pequenos subconjuntos dos dados para ajustar os pesos de forma incremental. Esses subconjuntos são chamados de *mini-batch*. É pressuposto que a média do *mini-batch* possua um valor equivalente ao do vetor gradiente da função, tornando o aprendizado mais rápido. A Equação 2.10 demonstra a equivalência, onde m é o tamanho do subconjunto aleatório de entradas.

$$\nabla C \approx \frac{1}{m} \sum_{j=1}^{m} \nabla C_{x_j} \tag{2.10}$$

2.4.3.3 Adam

O Adam é um método otimização estocástico que calcula taxas de aprendizagem adaptativas individuais para diferentes pesos, a partir de estimativas do gradiente, a um menor custo computacional, como dito em Kingma e Ba (2017). Esse método busca agregar as vantagens de outros, como a capacidade tanto de otimizar taxas de aprendizagem individuais em gradientes esparsos, típico do AdaGrad, como de também otimizar as taxas de aprendizagem com base nas médias recentes das magnitudes dos gradientes, característica do RMSProp.

A um dado instante de tempo t, o algoritmo do Adam busca atualizar com as médias móveis exponenciais do gradiente e do quadrado do gradiente, respectivamente m_t e v_t. Dois hiperparâmetros importantes do Adam são o β_1 e β_2, que controlam o decaimento exponencial das médias calculadas. Nas Equações 2.11 estão demonstrados os cálculos, onde $\nabla_\theta f_t(\theta_{t-1})$ são os gradientes da função objetivo no instante t, $\lambda \cdot \theta_{t-1}$ representa a regularização L2, θ é o vetor de pesos, β_1 e β_2 são os hiperparâmetros que controlam o decaimento exponencial, m_t é a média móvel exponencial do gradiente no instante t e v_t é a média móvel exponencial do quadrado do gradiente no instante t.

$$\begin{aligned} g_t &= \nabla_\theta f_t(\theta_{t-1}) + \lambda \cdot \theta_{t-1} \\ m_t &= \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \\ v_t &= \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \end{aligned} \tag{2.11}$$

A Equação 2.12 representa o momento que os pesos são atualizados. Os elementos $\hat{m}_t$ e $\hat{v}_t$ são as estimativas com correção de viés, calculadas por $\hat{m}_t = m_t/(1-\beta_1^t)$ e $\hat{v}_t = v_t/(1-\beta_2^t)$, respectivamente. O ϵ é um valor positivo e pequeno com a finalidade de impedir uma divisão por 0, sendo comum atribuir 10^{-8}. O α é o *stepsize*.

$$\theta_t = \theta_{t-1} - \alpha \cdot \hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon) \tag{2.12}$$

Uma modificação do método Adam foi proposto por Loshchilov e Hutter (2017). Chamado de AdamW, a diferença está no desacoplamento entre o decaimento do peso e a atualização do gradiente, acrescentando o peso apenas depois do processo, o que faz com que o mesmo não entre no cálculo da média móvel do processo do Adam original quando se está calculando os gradientes. A Equação 2.13 demonstra esse procedimento, evidenciando que o decaimento encontra-se fora da divisão ($\lambda \cdot \theta_{t-1}$), onde λ é a taxa de decaimento do peso por *step*, θ é vetor de pesos em um dado instante, α é o *stepsize*, η_t é o fator de escalonamento, que tem como função considerar um escalonamento de α e λ.

$$\theta_t = \theta_{t-1} - \eta_t(\alpha \cdot \hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon)) + \lambda \cdot \theta_{t-1} \tag{2.13}$$

No Algoritmo 2 é mostrado o pseudocódigo do AdamW, baseado em Loshchilov e Hutter (2017). Na linha 9 o η_t é definido por um ajuste da taxa de aprendizado a ser definido pelo usuário, que pode ser um valor fixo ou fazer decair com o tempo, sendo aplicado a cada iteração da execução do otimizador.

Algorithm 2 AdamW - Baseado em Loshchilov e Hutter (2017).

1: **Entrada:** Parâmetros θ, taxa de aprendizado α, coeficientes de decaimento β_1, β_2, ϵ, parâmetro de decaimento de peso λ
2: Inicializar $m_0 = 0$, $v_0 = 0$
3: **for** $t = 1, 2, \ldots$ **do**
4: $\quad g_t \leftarrow \nabla f_t(\theta_{t-1})$
5: $\quad m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$
6: $\quad v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$
7: $\quad \hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$
8: $\quad \hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$
9: $\quad \eta_t \leftarrow SetScheduleMultiplier(t)$
10: $\quad \theta_t \leftarrow \theta_{t-1} - \eta_t(\frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}) + \lambda \theta_{t-1}$
11: **end for**

2.5 Busca Aleatória

Em modelos de AM, segundo Aggarwal (2018), os hiperparâmetros são valores configurados antes da etapa de treinamento do modelo, e após fixados os valores esses

não serão mais alterados, tendo como exemplo taxa de aprendizado, número de camadas ocultas e até o valor de k do modelo kNN. Os valores dos hiperparâmetros podem ser configurados manualmente, com base em testes empíricos, ou por alguma técnica de *tuning*, que visa testar diversos combinações de valores para avaliar qual deles melhor se adequa ao experimento. Uma dessas técnicas se chama Busca Aleatória ou *Random Search*. A Busca Aleatória consiste em uma técnica de *tuning* de hiperparâmetros que utiliza uma distribuição de probabilidade para determinar os valores a serem testados, gerando combinações aleatórias de valores a cada etapa do seu processo.

De acordo com Bergstra e Bengio (2012), o *Random Search* é um método mais eficiente que outros também utilizados, como por exemplo o *GridSearch*. É perceptível que à medida que mais hiperparâmetros são adicionados, o total de combinações aumenta de forma exponencial, aumentando o custo de processamento necessário para executar o *GridSearch*. Além disso, verificar todas as combinações possíveis faz com que se gaste tempo de processamento com valores pouco relevantes para o experimento.

Com o *Random Search*, a busca se mostra mais eficiente, pois devido a sua aleatoriedade, consegue testar diversas combinações de valores distintos em um menor período de tempo, possibilitando o encontro de bons valores mais rapidamente. Além disso, é possível paralelizar a busca, pois cada uma é feita de forma independente, ou seja, havendo disponibilidade de mais máquinas para processar, acelera ainda mais a obtenção dos resultados. Outro fator importante é quantas iterações o *Random Search* irá executar, tornando-o mais eficiente em relação ao custo computacional, pois independe da quantidade de hiperparâmetros. A aleatoriedade se dá por conta de uma distribuição probabilística, como por exemplo a Distribuição Gaussiana.

2.6 Estimativa de densidade de kernel

Estimativa de Densidade de Kernel (EDK), em inglês *Kernel Density Estimation* ou KDE, é uma forma não-paramétrica de estimar a Função densidade de probabilidade (FDP) de uma variável aleatória. Kernel, por sua vez, é uma função que, de acordo com Bishop (2006) e Gramacki (2018), satisfaz as características demonstradas nas equações 2.14.

$$\begin{aligned}
&\int K(x)dx = 1, \\
&\int x^j K(x)dx = 0 \text{ para j = 1,...,k-1}, \\
&\int x^k K(x)dx \neq 0, \\
&K(x) = K(-x), \text{ para todo } x \\
&K(x) \geq 0, \text{ para todo } x
\end{aligned} \tag{2.14}$$

A função de kernel deve ser simétrica e não negativa para todo x. Esse x, que é

o valor onde a função de kernel é computada, vem de uma série de valores linearmente espaçados que contém todos os dados observados. $K(x)$ será calculado para cada elemento dessa série em relação a cada dado observado. Além disso, a sua área deve ser sempre igual a 1. Para obter a EDK dos dados, ele vai somar todos os resultados das aplicações da função de kernel em cada valor e normalizar o resultado (dividir pelo tamanho da amostra). Na Figura 8 contém um exemplo didático de um EDK calculado, em uma série com valores entre 1 e 100 usando a função de kernel Gaussiana.

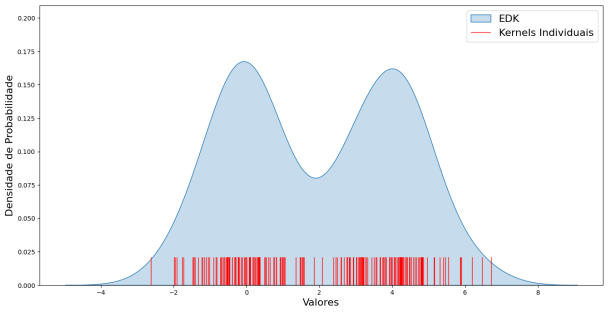


Figura 8 – Exemplo didático de EDK com valores entre 1 e 100.

Um dos principais parâmetros do EDK é a largura de banda, ou *bandwidth*, que define o quão suave é a curva da função, representando o seu desvio padrão. Uma baixa largura de banda vai considerar apenas os pontos mais próximos, e também uma maior variância, e a medida que aumenta, dados mais distantes passam a serem considerados na estimativa, favorecendo um maior viés. Esse parâmetro vai influenciar a acurácia da estimativa de densidade de kernel. Existem diferentes funções de kernel para serem utilizadas, de acordo com Gramacki (2018). Neste trabalho, foi utilizada a função Gaussiana, mostrada na Equação 2.15, onde h é a largura de banda, x é o valor que a função foi aplicada e x_i é um parâmetro de localização, que determina a posição central da curva. Essa função é mais utilizada em aplicações práticas com EDK.

$$K(x) = \frac{1}{h\sqrt{2\pi}} e^{-0.5(\frac{x-x_i}{h})^2} \tag{2.15}$$

3 Trabalhos Relacionados

O trabalho de Aernouts e Berkvens (2018) foi responsável por criar a primeira versão do conjunto de dados utilizado no presente experimento, e além disso, no mesmo artigo, também foi demonstrada uma tentativa de localização de dispositivos, utilizando o próprio conjunto de dados, com a técnica kNN. Com os subconjuntos de treino, validação e teste, foi criada uma matriz de distâncias contendo as distâncias euclidianas entre os valores de RSSI de treino e validação. Cada linha dessa matriz foi testada com diferentes valores de k, comparando com as coordenadas reais do subconjunto de validação, e no final foi estimado o respectivo erro médio. Com isso, foi encontrado um valor ótimo de k, o qual foi utilizado nas predições com o subconjunto de teste. Com relação aos resultados da rede LoRa, o erro médio foi de $398,4$ metros, e a mediana do erro foi de $273,03$ metros. O presente trabalho busca testar outros métodos para estimar a localização do dispositivo, utilizando uma versão mais recente do conjunto de dados destes autores, com mais registros de pacotes.

Também utilizando a mesma base de dados de Aernouts e Berkvens (2018), em Anagnostopoulos e Kalousis (2019) foi realizado um experimento a fim de comparar diferentes métodos para localizar dispositivos em uma rede LoRa/LoRaWAN. Os métodos utilizados foram o kNN, *Extra Trees* e MLP. Foi utilizado como dados de entrada os valores do RSSI preprocessados de acordo com quatro métodos: O método Positivo, em que os valores negativos do RSSI são transformados em valores positivos, subtraindo cada valor por um limiar, sendo esse limiar o valor do menor RSSI subtraído de 1; o método Normalizado, em que os valores de RSSI são escalonados em valores entre 0 e 1; os métodos Exponencial e *Powed*, transformações não-lineares, dado que o RSSI tem uma escala logarítmica. O melhor resultado encontrado foi com a MLP, obtendo um erro médio de 357 metros e mediana de 206 metros.

Modelos de redes RBF foram utilizadas no experimento de Yang, Yu e Li (2021), visando estimar a localização de dispositivos em ambientes internos. Os conjuntos de dados utilizados continham valores de RSSI de redes wireless em prédios de múltiplos andares, e serviram para que as redes propostas pudessem estimar o andar e a posição do dispositivo no respectivo pavimento. Para isso, foram executadas redes RBF dos tipos Classificativa e Regressiva, dado que andares são considerados valores discretos, enquanto a localização no andar são valores contínuos, como mencionado no trabalho. O sistema proposto no experimento inclui três etapas: inicialização para preparar a rede para o ambiente específico, calibração para otimizar os parâmetros da rede e maximizar a precisão da localização, e atualização para permitir a adaptação contínua às mudanças das condições do ambiente. Analisando a média de erro na localização do dispositivo, comparando com kNN, SVM

e DNN, apresentou um desempenho melhor de 21%, 14% e 15% respectivamente. Já na estimativa do andar do prédio, analisando a taxa de acerto, comparando com as mesmas técnicas e na mesma ordem, a melhora foi de 21,6%, 16,6% e 12,7%. O erro médio apresentado foi de 6,55 metros e a mediana foi de 4,55 metros.

O trabalho de Zhang et al. (2019) utilizou um modelo CNN em conjunto com um processo de regressão gaussiano (GPR) para estimar localização em ambientes internos usando dados de RSSI de redes Wi-Fi. Utilizando o conjunto de dados disponibilizado por Mendoza-Silva et al. (2018), para cada coordenada foi atribuída um agrupamento de vetores contendo valores de sinal RSSI, onde cada vetor são os valores em um instante de tempo. Esse agrupamento, enfim, é interpretado como uma imagem, e cada vetor como um *pixel* multicanal. Os valores de RSSI foram normalizados, enquanto valores inferiores a -105 dbm foram considerados como 0. Os modelos CNN foram executados com e sem o GPR, e as versões com GPR variou entre 3 funções de kernel, sendo elas a *Matern*, *Periodic Exponential Regression* (PER) e *Squared Exponential* (SE). Os resultados foram comparados com um modelo kNN, e considerando a média dos erros de cada modelo CNN testado, apresentou um desempenho 61,8% melhor que o kNN, com um MAE de 1 metro e o percentil 75 do erro foi de 1,35 metros, não sendo divulgada a mediana.

O trabalho de Magsi et al. (2023) utilizou a técnica de *Support Vector Regression* (SVR) junto com dados de transmissão de pacotes LoRa/LoRaWAN, em um cenário montado pelos autores, a fim de estimar a localização de dispositivos remotos. Um *gateway* foi colocado em uma posição 74 metros acima do nível do mar, e na área ao redor, foram selecionados de forma arbitrária 14 pontos dentro da área utilizada de onde o dispositivo iria enviar os pacotes. Todos esses pontos estavam situados em área externa. A menor e maior distância verificada entre o *gateway* e os pontos foram de 17 metros e 1330 metros, respectivamente. Em cada um dos 14 pontos foram coletadas 21 leituras de valor de RSSI, esses dados passaram por uma etapa de preprocessamento e padronização usando *z-score*, para enfim aplicar a técnica de SVR. Obteve um erro médio de 171,59 metros nas estimativas, e de acordo com os valores de erro em cada ponto demonstrados no trabalho, a mediana calculada a partir deles foi de 84,76 metros.

A Tabela 2 traz as características de cada trabalho mencionado, permitindo avaliar de forma mais objetiva as diferenças. Por exemplo, alguns trabalhos utilizaram bases de dados em ambientes internos, abrigados, e os valores do erro de estimativa desses demonstraram serem consideravelmente menores que os trabalhos com dados de transmissões em ambientes externos. Além disso, esses trabalhos em ambientes internos utilizaram dados de redes WiFi. O projeto aqui proposto utiliza uma base de dados de transmissões Lora/LoRaWAN realizadas em ambiente externo e urbano. Todos os trabalhos relacionados utilizaram seus modelos no contexto de regressão, com apenas um que utilizou também um modelo classificador, porém o presente trabalho optou por utilizar modelos

de regressão apenas devido aos resultados verificados na literatura. Já dentre os modelos utilizados, o trabalho aqui proposto apresentou ideias diferentes, onde no modelo CNN foram utilizadas como dados de entrada imagens das estações plotadas com aplicação da técnica de EDK para identificar a estação com maior RSSI daquele pacote, em detrimento de valores de RSSI. Já o modelo RBF foi escolhido devido à uma pressuposição de que a similaridade entre a estrutura dos dados utilizados e a estrutura clássica de uma rede RBF pudesse favorecer resultados mais precisos. Nos modelos RBF e kNN, além do RSSI padrão, também foi utilizado o RSSI *Powed*, buscando avaliar o impacto nos resultados desse valor transformado. Além disso, os resultados foram comparados com técnicas tradicionais utilizadas para localização remota, como o TDoA e RSSI *fingerprinting*, e não somente entre modelos de inteligência artificial, possibilitando verificar o desempenho dos modelos propostos em relação à essas técnicas, identificando as vantagens e desvantagens.

Trabalho	**Ano de publicação**	**Técnicas utilizadas**	**Ambiente***	**Dados utilizados**	**Tipo do problema**	**Erro médio (metros)**	**Mediana (metros)**
(YANG; YU; LI, 2021)	2021	RBF	I	RSSI WiFi	Classificação e Regressão	6,55	4,55
(ZHANG et al., 2019)	2019	CNN	I	RSSI WiFi	Regressão	0,95	Não informado
(AERNOUTS; BERKVENS, 2018)	2018	kNN	E	RSSI LoRa	Regressão	398,4	273,03
(ANAGNOSTOPOULOS; KALOUSIS, 2019)	2019	kNN, MLP e Extra Trees	E	RSSI LoRa	Regressão	357	206
(MAGSI et al., 2023)	2023	SVR	E	RSSI LoRa	Regressão	171,59	84,76

Tabela 2 – Descrição dos trabalhos relacionados.
* I = Interno, E = Externo

4 Materiais e Métodos

Este Capítulo irá apresentar os materiais e métodos empregados na realização dos experimentos propostos, incluindo o conjunto de dados, os métodos selecionados e as métricas adotadas para avaliação do desempenho dos modelos.

Foi utilizado um conjunto público de dados[1], o qual contém registros de transmissões de pacotes LoRa em um ambiente urbano. Esses dados passaram por uma etapa de preprocessamento a fim de obter os valores formatados e compatíveis para uso com os métodos e cenários executados. Na etapa de formatação, foi condensado em apenas um arquivo CSV todos os dados do arquivo original do conjunto e as coordenadas das estações (disponibilizadas em arquivo separado), incluindo posteriormente as coordenadas transformadas em *pixels*, processo esse detalhado na Seção 4.1.3.

Foram testados dois cenários, onde um deles contém dados de pacotes recebidos por 27 estações, quantidade essa encontrada após filtrar as estações menos relevantes, e o outro cenário contempla apenas 2 estações, com o intuito de testar situações que os métodos analíticos tradicionais não conseguem estimar com precisão satisfatória.

Os métodos utilizados são o kNN, RBF e CNN, representando o conjunto de modelos de AM, além dos métodos analíticos TDoA e RSSI *fingerprinting*, que foram usados como referência para a avaliação da qualidade dos modelos de AM testados, visto que são os métodos tradicionalmente empregados em problemas deste tipo, como visto em Wu et al. (2019) e Gioia, Sermi e Tarchi (2020). Os resultados obtidos foram comparados utilizando a métrica de Erro Médio Absoluto (*Mean Absolute Error* ou MAE).

Esse projeto foi executado dentro da plataforma Google Colab[2], utilizando um sistema operacional com kernel Linux, entre os anos de 2022 e 2023. Durante esse período, mas não de forma contínua, a plataforma foi utilizada com os recursos disponibilizados pela assinatura Colab PRO, que fornecia unidades computacionais que permitia acesso a GPUs, variando entre os modelos T4, A100 e V100 da fabricante NVIDIA, de acordo com a disponibilidade da plataforma, sendo útil principalmente para o modelo CNN.

4.1 Conjunto de dados

O conjunto de dados foi compilado por Aernouts e Berkvens (2018), contendo dados referentes a transmissão de pacotes utilizando o protocolo LoRaWAN na cidade da Antuérpia, Bélgica. Para envio dos pacotes e registro da localização, os autores utilizaram

[1] Conjunto Público de dados LoRa - https://zenodo.org/records/3904158
[2] Google Colab - https://colab.research.google.com/

três módulos *OCTA-Connect*, sendo eles do modelo LoRaWAN, Sigfox e GPS. Estes dispositivos estão apresentados na Figura 9. Para gerar os dados, 20 carros do serviço postal da cidade foram equipados com esses dispositivos, percorrendo toda a cidade entre as datas de 17 de novembro de 2017 até 5 de fevereiro de 2018, contendo no total 123.529 registros de pacotes.

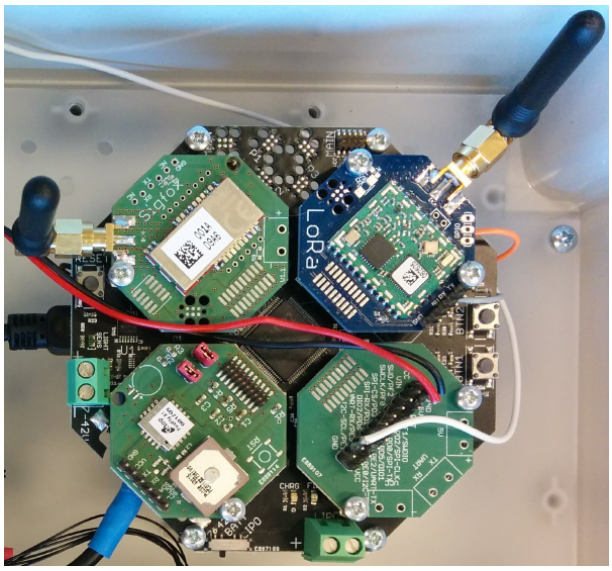


Figura 9 – Módulos OCTA-Connect. Fonte: (AERNOUTS; BERKVENS, 2018).

No entanto, para o presente trabalho, foi utilizada uma versão atualizada desse conjuntos de dados, coletados 1 ano depois do original, encontrada em Aernouts et al. (2019), contendo mais registros de pacotes, totalizando 130.430. Outros dados encontrados são as localizações das estações base, em coordenadas de latitude e longitude. Esses dados foram essenciais para o trabalho, principalmente para a geração das imagens utilizadas nas redes CNN. Ambos os conjuntos de dados estão disponíveis em formato JSON. Na Figura 10 estão os dados presentes no conjunto original.

Dos dados referentes à transmissão dos pacotes, para os modelos propostos foi utilizado apenas o RSSI. O RSSI é uma medida que mensura a potência do sinal recebido por uma estação receptora, expresso em dBm (decibel miliwatt). Essa medida pode indicar a qualidade do sinal de uma conexão, mas não pode ser considerado uma referência absoluta, pois outros fatores, como interferências, podem afetar a comunicação. A relação entre o sinal e o ruído, conhecido também pela sigla em inglês SNR e expresso em decibel, presente na base de dados, foi utilizada apenas nos métodos de TDoA e RSSI *fingerprinting*. As coordenadas geográficas originais estavam em pares de latitude e longitude, expressos em graus.

- **HDOP**: Horizontal Dilution of Precision
- **dev_addr:** LoRaWAN device address
- **dev_eui:** LoRaWAN device EUI
- **sf:** Spreading factor
- **channel:** TX channel (EU region)
- **payload:** application payload
- **adr:** Adaptive Data Rate (1 = enabled, 0= disabled)
- **counter:** device uplink message counter
- **latitude:** Groundtruth TX location latitude
- **longitude:** Groundtruth TX location longitude
- **airtime:** signal airtime (seconds)
- **gateways:**
  - **rssi:** Received Signal Strength
  - **esp:** Estimated Signal Power
  - **snr:** Signal-to-Noise Ratio
  - **ts_type:** Timestamp type. If this says "GPS_RADIO", a nanosecond precise timestamp is available
  - **time:** time of arrival at the gateway
  - **id:** gateway ID

Figura 10 – Dados contidos no conjunto de dados original. Fonte: (AERNOUTS et al., 2019).

4.1.1 Seleção das estações para os cenários de experimentação

A seleção das estações considerou experimentos de outros trabalhos e também para se adequar aos recursos computacionais disponíveis. Foi visto que o trabalho de Anagnostopoulos e Kalousis (2022) utilizou apenas pacotes recebidos por pelo menos 3 estações receptoras, devido a técnicas como TDoA e Multilateração requererem esse mínimo em seus cálculos, ficando apenas com 55.375 pacotes do conjunto de dados original. Nos cenários experimentados foram utilizados todos os 130.430 pacotes do conjunto de dados.

Com relação às localizações de estações base, os identificadores das estações presentes no conjunto de dados dos pacotes foram comparados com os identificadores do conjunto de dados das localizações das estações. O cruzamento desses dados resultou em 39 estações com localização conhecida, pois o conjunto de dados de localização das estações dispõe de uma quantidade maior de estações comparado com o outro conjunto, e também algumas estações do conjunto de dados dos pacotes não tem dados de localização.

Com relação às estações receptoras, foi aplicado um filtro a fim de remover estações que receberam menos de 1% da quantidade total de pacotes, ou seja, menos de 1304 pacotes. Isso fez reduzir de 39 para 27 estações. Isso foi feito para se adaptar a infraestrutura disponível para o experimento, dado que modelos mais complexos, como a CNN, demandou bastante recursos computacionais, então estações que pouco receberam foram eliminadas do conjunto.

Para selecionar duas estações dentre as 27 selecionadas, considerou-se o valor de correlação mais negativo entre as estações. A intenção foi de obter um ambiente mais adverso e conseguir testar o desempenho dos modelos nessa situação. Ao utilizar o valor do RSSI no cálculo da matriz de correlação, quanto mais negativa a correlação, maiores os casos em que uma estação recebeu e a outra não. Logo, ao executar os modelos, em várias ocorrências eles terão que estimar a localização de origem do transmissor utilizando o RSSI de pacotes recebidos por apenas uma estação, testando assim seu desempenho nesse ambiente ainda mais complexo, considerando soluções tradicionais que necessitam de pelo menos 3 estações. Na Figura 11 é demonstrada a matriz de correlação calculada, e nela as estações 080E00B9 e FF0107C9 apresentaram uma correlação de $-0,34$, sendo essa a maior correlação negativa.

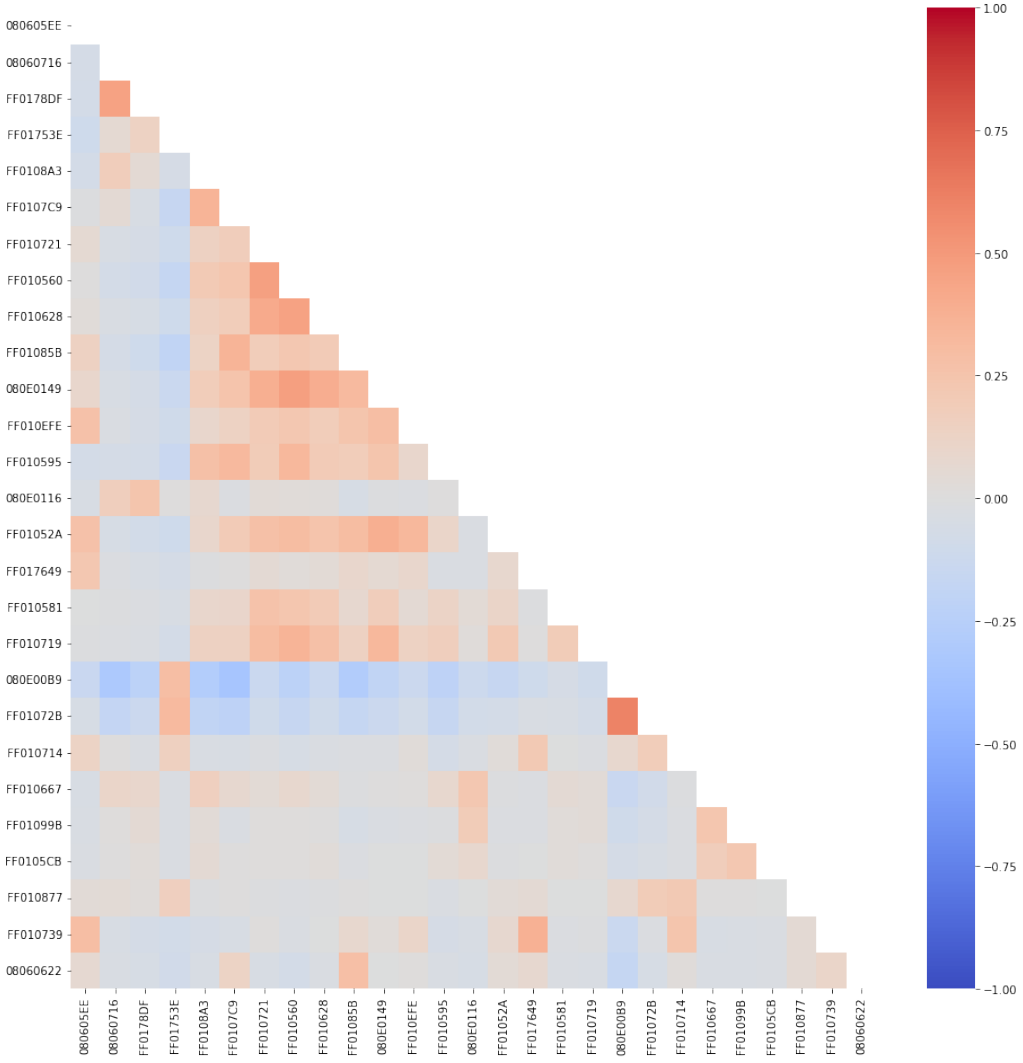


Figura 11 – Matriz de correlação entre as estações em relação ao recebimento de pacotes.

Utilizando a linguagem Python com a biblioteca Pandas[3], foi criado um único arquivo contendo todos os dados interessantes para a pesquisa, no formato CSV. Esses dados

[3] Biblioteca Pandas - https://pandas.pydata.org

consistem nos valores de RSSI e coordenadas geográficas de cada estação, as coordenadas geográficas da origem do respectivo pacote, o valor de SNR, além das coordenadas em pixels das estações e da origem do pacote (representados pelo X e Y) decorrente de um mapeamento aplicado nas coordenadas geográficas, cujo processo é explicado na Seção 4.1.3.

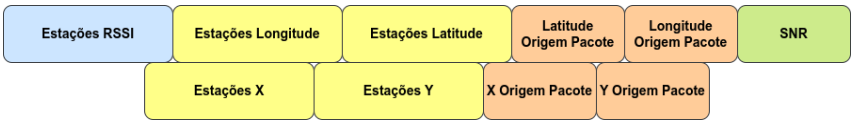


Figura 12 – Dados originais utilizados no tratamento prévio dos dados.

4.1.2 Caracterização da base de dados

O conjunto de dados utilizado no presente trabalho, após as manipulações descritas no Capítulo 4, foi submetido a uma análise exploratória para a melhor compreensão de suas características. A Figura 13 traz o total de pacotes recebidos por cada uma das 27 estações da base de dados preprocessada, ordenado-as de forma decrescente.

Esses dados evidenciaram que nenhuma das estações, de forma isolada, recebeu sequer um terço do total de pacotes, inclusive excetuando-se as duas que mais receberam, o restante não recebeu um quarto do total. Também é possível perceber uma tendência de queda no recebimento de pacotes para as estações que encontram-se mais afastadas do centro da cidade, como pode-se verificar ao comparar as posições com os IDs na Figura 14. Este fato parece estar de acordo com a natureza da captura realizada utilizando-se os dispositivos emissores em carros do serviço postal da cidade. É esperado que haja uma maior frequência de presença desses carros nas proximidades do centro da cidade onde há também uma maior concentração populacional.

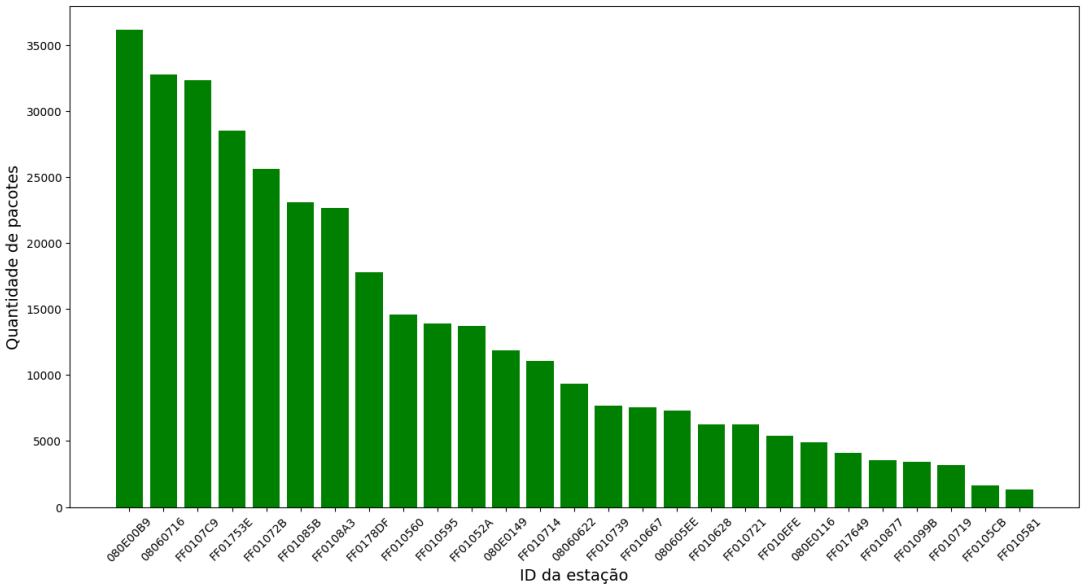


Figura 13 – Quantidade de pacotes recebidos por cada estação.

Na Figura 14 é possível perceber que as estações que mais receberam pacotes estão localizadas mais ao centro do mapa, que coincide com a região central da cidade. Elas estão destacadas com os pinos nas cores verde, rosa e laranja, respectivamente em relação a Tabela 3. Na Tabela 3 estão as quantidades de pacotes recebidos por essas 3 estações. De acordo com a matriz de correlação da Seção 4.1.1, as duas estações que apresentaram a maior correlação negativa foram as estações $080E00B9$ e $FF0107C9$, a primeira e terceira, respectivamente, que mais receberam pacotes. O interesse pela maior correlação negativa é a de identificar um contexto em que o maior número de pacotes possível seja recebido por uma ou outra estação, e não ambas para o mesmo pacote, evidenciando um cenário complexo para estimar a localização. Na Figura 15 é mostrado um mapa de calor da origem de transmissão dos pacotes, onde percebe-se uma concentração relevante na área central da cidade.

ID da Estação	Quantidade de pacotes recebidos
080E00B9	36185
08060716	32812
FF0107C9	32353

Tabela 3 – Estações que mais receberam pacotes.

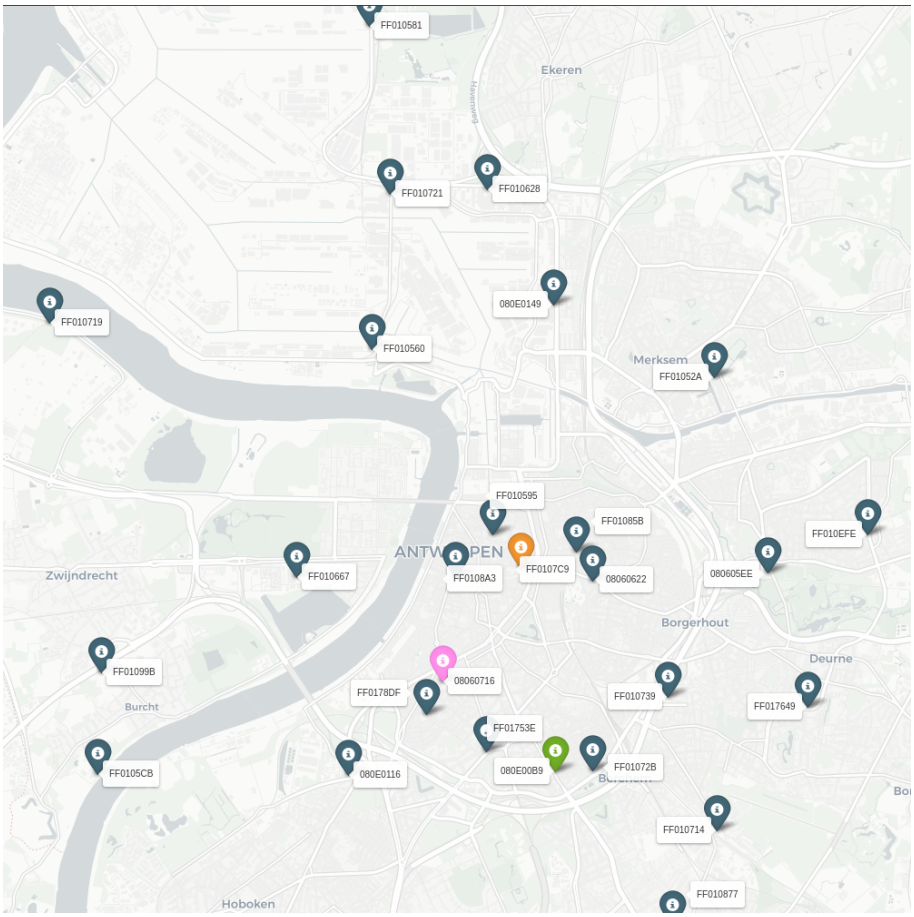


Figura 14 – Plot das estações na cidade da Antuérpia com seus IDs.

Figura 15 – Mapa de calor da origem de transmissão dos pacotes.

Na Figura 16 mostra a relação de quantos pacotes foram recebidos por um número de estações simultaneamente. É visível que a maior parte dos pacotes foi recebida por até 3 estações. Essa informação, aliada ao fato de haver muitas estações que receberam proporcionalmente poucos pacotes, reforça a relevância dos testes com cenários de apenas 2 estações.

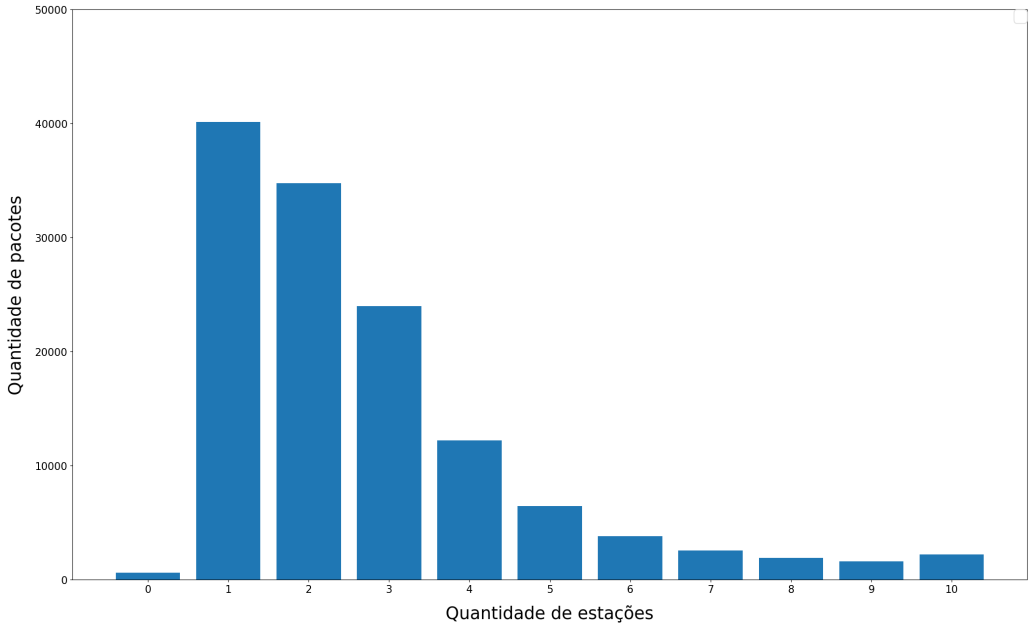


Figura 16 – Levantamento de quantidade de estações que receberam cada pacote.

4.1.3 Transformação das coordenadas geográficas em *pixels*

Visando utilizar valores menos complexos de coordenadas geográficas para os experimentos, os dados de latitude e longitude, tanto para as estações quanto para os pacotes, foram convertidos para um grid de *pixel*. Com isso, cada célula desse grid representa uma parte da área total, facilitando assim os cálculos dos modelos.

O primeiro passo foi de converter as coordenadas geográficas expressas como latitude e longitude para valores em metros, utilizando a fórmula de *haversine*, demonstrada na Equação 4.1. Na fórmula, d é distância esférica entre os dois pontos, r é o raio da esfera (nesse caso, é o da Terra), ϕ_1 e ϕ_2 são as latitudes dos dois pontos, λ_1 e λ_2 são as longitudes dois pontos.

Neste trabalho, utilizando como ponto de origem um ponto arbitrário que possibilitasse o cálculo das distâncias, demonstrado na Figura 17, onde esse ponto foi considerado a coordenada cartesiana $(0, 0)$, cada coordenada geográfica foi convertida para uma coordenada cartesiana equivalente, no formato (x, y). Para calcular o x, é calculada a distância do ponto de origem $(0, 0)$ para um ponto $(0, LONGITUDE)$, onde a $LONGITUDE$ seria o valor da longitude original do elemento. O y, por sua vez, é obtido ao calcular a distância do ponto de origem $(0, 0)$ para um ponto $(LATITUDE, 0)$, onde a $LATITUDE$ seria o valor da latitude original do elemento.

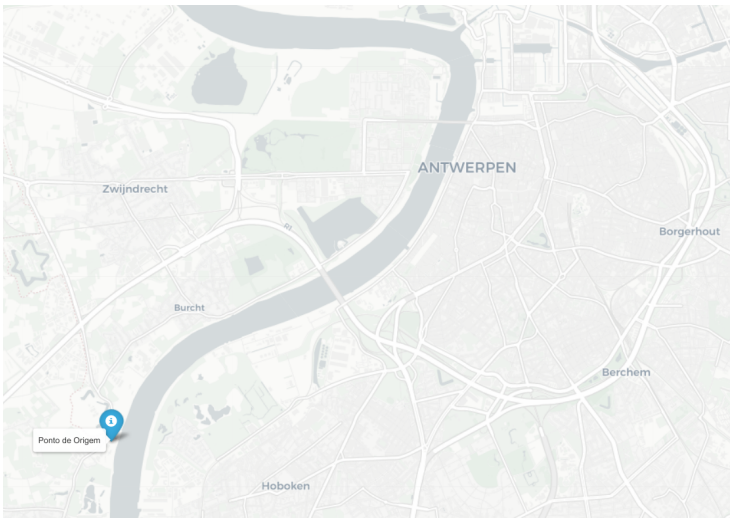


Figura 17 – Ponto arbitrário de origem para cálculo das distâncias.

$$haversine(\frac{d}{r}) = haversine(\phi_2 - \phi_1) - cos(\phi_1) \cdot cos(\phi_2) \cdot haversine(\lambda_2 - \lambda_1) \quad (4.1)$$

Em seguida, esses valores em metros foram escalonados para uma matriz de *pixels* de tamanho 128 x 128. Isso fez com que cada *pixel* representasse uma área de aproximadamente $0{,}01km^2$.[4]

[4] Cálculo de área - largura × altura = 95m × 110m = $10450m^2 \approx 0{,}01km^2$

4.1.4 Transformação *Powed*

Como visto no trabalho de Anagnostopoulos e Kalousis (2019), seus melhores resultados tiveram como entrada os valores de RSSI transformados com a função *Powed*, que realiza uma transformação não-linear, respeitando a característica de escala logarítmica do RSSI e demonstrada no trabalho de Torres-Sospedra et al. (2015). Na Equação 4.2 é mostrado como é realizado o cálculo, onde o i é o número da estação, x é o valor do RSSI padrão, min é o menor valor de RSSI de todo o conjunto de dados (desconsiderando o valor padrão de -200 dos que não receberam) subtraído de 1, $Positive_i(x)$ é o valor positivo do RSSI obtido pela Fórmula 4.3 e o β é um parâmetro baseado em casos, como descrito em Torres-Sospedra et al. (2015). Para este trabalho, o parâmetro β foi definido como $1,1$, que é o utilizado no trabalho de Anagnostopoulos e Kalousis (2019).

$$Powed_i(x) = \frac{(Positive_i(x))^\beta}{(-min)^\beta} \tag{4.2}$$

$$Positive_i(x) = \begin{cases} (RSSI_i - min) \text{ se a estação recebeu o pacote} \\ \\ 0 \text{ se a estação não recebeu} \end{cases} \tag{4.3}$$

4.1.5 Geração das imagens para a CNN

Para servir como dados de entrada para o modelo CNN, foram geradas imagens para os 2 cenários, o de 27 e o de 2 estações. As imagens possuem as estações posicionadas de acordo com sua coordenada geográfica, aplicando a técnica de Estimativa de Densidade de Kernel (EDK), abordada na Seção 2.6, em que o valor da intensidade de RSSI vai servir como "peso", e vai determinar o quão excitado estará a respectiva estação, quanto mais forte o sinal, mais próximo da cor branca estará o ponto. O tamanho definido para as imagens foi de 128 x 128 pixels, proposital para coincidir com o grid de pixels mencionado na Seção 4.1.3, com coloração em escala de cinza, onde cada pixel apresenta um valor para sua cor entre 0 (preto) e 255 (branco).

Foi utilizada a biblioteca Scikit-learn[5] versão 1.2.2 para a execução do EDK, e a sua função apenas aceitou valores maiores que 0 como pesos. Como o RSSI foi escolhido como peso para o EDK, e seu valor é negativo, houve a necessidade de realizar uma transformação nesses valores, sendo escolhida a transformação Positiva modificada, explicada na Equação 4.4. A diferença é que estações que não receberam o pacote foi atribuído o valor 1, e as que receberam foi feito o cálculo da subtração do valor mínimo,

[5] Scikit-learn - https://scikit-learn.org/

somando 1 ao resultado, satisfazendo assim o requisito da função do EDK. Não foram geradas imagens utilizando o RSSI transformado com a função *Powed*.

$$Positive_i(x) = \begin{cases} (RSSI_i - min + 1) \text{ se a estação recebeu o pacote} \\ \\ 1 \text{ se a estação não recebeu} \end{cases} \tag{4.4}$$

Na Figura 18 é mostrado um exemplo de imagem gerada para ser utilizada como dado de entrada no modelo CNN. Esse exemplo demonstra 4 estações que receberam o pacote, visto que existem 4 pontos excitados na imagem.

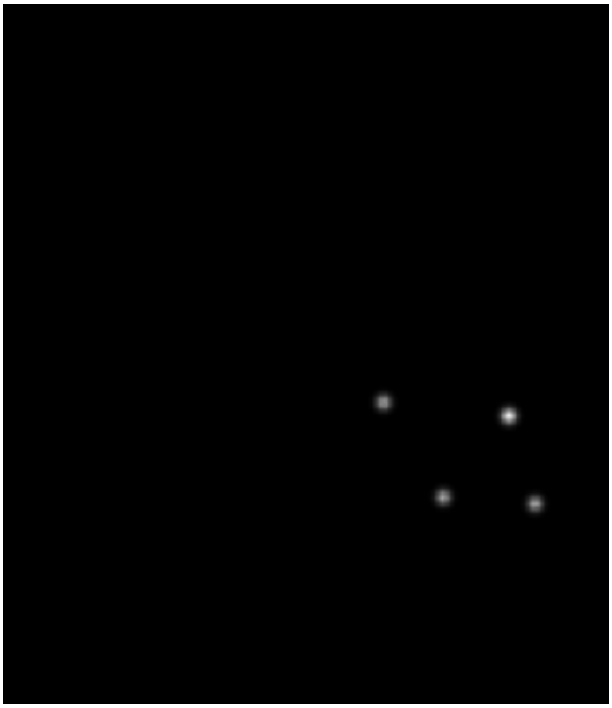


Figura 18 – Exemplo de imagem gerada para ser usada como entrada no modelo CNN.

4.1.6 Divisão do conjunto

Para a execução dos modelos de redes neurais utilizados neste trabalho, desde a parte de treino até as estimativas de localização, o conjunto de dados foi dividido em 3 subconjuntos, sendo eles treino, validação e teste. Na Tabela 4 estão os valores proporcionais de cada subconjunto, em relação ao conjunto de dados original. A divisão foi feita utilizando o arquivo gerado na explicação da Seção 4.1.1, reservando 10% do conjunto original para testes, e dos 90% restantes, 10% deles foram reservados para validação, chegando assim à proporção demonstrada. Essas divisões foram feitas uma versão para cada cenário (27 e 2 estações) e salvas em arquivos independentes, de formato CSV, para facilitar o uso direto nos métodos executados. No caso do cenário com 2 estações, foram

utilizados os arquivos do cenário de 27 estações e retirados os dados das outras 25 estações não selecionadas. O conjunto de testes foi o mesmo para todos os métodos e modelos executados.

Subconjunto	**Proporção**
Treino	81%
Validação	9%
Teste	10%

Tabela 4 – Quantidade proporcional de registros nos subconjuntos de dados referentes à base de dados original após o tratamento prévio.

4.2 Métodos analíticos

Com a finalidade de ter uma referência para comparar os resultados dos modelos propostos por este trabalho, foi obtida a localização dos pacotes utilizando TDoA, um método clássico para esse tipo de problema que utiliza o tempo de propagação das ondas, um dos princípios de posicionamento dito por Sand, Dammann e Mensing (2014). Para isto, foi utilizado um serviço de geolocalização para redes LoRa oferecido pela Semtech[6], no primeiro semestre de 2023.

Nesse serviço, os usuários podem fazer requisições HTTP para uma API, usando um JSON contendo dados como coordenadas geográficas das estações, RSSI, além de outros parâmetros pertinentes a comunicações sem fio, e obtém como resposta um JSON com as localizações estimadas, caso o sistema consiga fazer a predição. Essa API também permite escolher se a predição vai ser realizada usando o TDoA ou RSSI *fingerprinting*, escolha essa podendo ser feita com um dos parâmetros no JSON enviado da requisição. É possível configurar para o caso o TDoA não consiga estimar, utilizar o método do RSSI *fingerprinting* como *fallback*, mas para as estimativas deste trabalho isso foi desativado, para que os valores estimados sejam de fato do método escolhido.

Para os testes deste trabalho, foram realizadas requisições em diferentes cenários. Para os testes com todas as estações, foram feitas requisições de localização para pacotes recebidos por no mínimo 3 estações, e foram testadas as estimativas com TDoA e o RSSI *fingerprinting*. Para os testes com apenas duas estações, foram utilizadas as estações selecionadas de acordo com o descrito em 4.1.1, e foram testadas as predições apenas com o RSSI *fingerprinting*, pois o TDoA não consegue estimar com menos de 3 estações, limitação essa da própria técnica. Em todos os cenários foram enviados no JSON de requisição os dados de RSSI, Latitude, Longitude, Tempo de chegada do pacote e o SNR (*Signal-to-Noise Ratio*), que expressa a qualidade do sinal enviado em decibéis.

[6] Semtech LoRa Cloud - https://www.loracloud.com/

4.3 Modelos investigados

Os modelos investigados neste trabalho foram o CNN, RBF e kNN. Para a execução dos modelos CNN e RBF foi utilizada a biblioteca Keras[7] versão 2.12.0, e para o kNN foi utilizada a biblioteca Scikit-learn. utilizando a linguagem Python versão 3.8 para todos. Sendo parte integrante da Keras, foi utilizado também o Keras Tuner[8] versão 1.1.3 para a execução da Busca Aleatória nos modelos CNN e RBF, e para o kNN foi utilizado recursos da própria biblioteca Scikit-learn. Para os modelos CNN e RBF foi utilizado o otimizador AdamW, e essa escolha foi devido aos resultados satisfatórios dos experimentos realizados por Loshchilov e Hutter (2017).

Os parâmetros da execução do treinamento do modelo foram os mesmos para a execução dos testes da Busca Aleatória e para o treinamento do modelo após encontrar os valores dos hiperparâmetros mais eficientes. Na Tabela 5 estão os valores utilizados nos modelos descritos. A técnica de *Early Stopping* foi utilizada buscando evitar o *overfitting* durante o treinamento.

Parâmetro	**RBF**	**CNN**
Epochs	2000	100
Batch size	128	128
Patience (Early Stopping)	10	5

Tabela 5 – Parâmetros para as etapas de testes da Busca Aleatória e treino.

4.3.1 Busca Aleatória

A escolha dessa técnica para escolher os hiperparâmetros se deu pela sua eficiência na busca pelos melhores valores, visando melhorar o desempenho da rede neural, como visto em Bergstra e Bengio (2012). Para cada um dos modelos, foram testadas 100 diferentes configurações de hiperparâmetros, utilizando os conjuntos de treino e validação, sendo escolhida a que obteve o menor erro segundo a métrica do Erro Médio Absoluto (MAE). Os conjuntos de valores que a Busca Aleatória poderia selecionar em cada configuração estão nas Tabelas 6, 7 e 8. No modelo CNN, a primeira camada convolucional permaneceu fixa, e a Busca Aleatória otimizou a quantidade restante de camadas, solução essa encontrada para lidar com a limitação da infraestrutura.

4.3.2 kNN

As Tabelas 9 e 10 mostram os valores dos hiperparâmetros selecionados para o modelo kNN em cada um dos cenários. Com esses valores, utilizando o regressor kNN

[7] Keras - https://keras.io/
[8] Keras Tuner - https://keras.io/keras_tuner/

Parâmetro	**Distribuição de Probabilidade**
k	Uniforme em $\{1, ..., 50\}$
métrica	Uniforme em {distância euclidiana, cosseno, l2, manhattan}

Tabela 6 – Conjunto de valores utilizados pela Busca Aleatória nos modelos kNN.

Hiperparâmetro	**Distribuição de Probabilidade**
Função de ativação	Uniforme em {Tangente Hiperbólica, ReLU}
Qtde Camadas ocultas	Uniforme em $\{1, ..., 6\}$
Neurônios camadas ocultas	Uniforme em $\{0, ..., 2000\}$
Dropout camadas oculta	Uniforme em $(0, 1, ..., 0, 5)$
Taxa aprendizagem (AdamW)	Uniforme em $(0, 0001, ..., 0, 01)$
Beta 1 (AdamW)	Uniforme em $(0, 10, ..., 0, 99)$
Beta 2 (AdamW)	Uniforme em $(0, 100, ..., 0, 999)$

Tabela 7 – Conjunto de valores utilizados pela Busca Aleatória nos modelos RBF.

Hiperparâmetro	**Distribuição de Probabilidade**
Função de Ativação	Uniforme em {Tangente Hiperbólica, ReLU}
Qtde camadas convolucionais	Uniforme em $\{1, ..., 4\}$
Neurônios camadas convolucionais	Uniforme em $\{1, ..., 100\}$
Dropout camadas convolucionais	Uniforme em $(0, 1, ..., 0, 5)$
Neurônios camada totalmente conectada	Uniforme em $\{1, ..., 100\}$
Dropout camada totalmente conectada	Uniforme em $(0, 1, ..., 0, 5)$
Taxa de aprendizagem (AdamW)	Uniforme em $(0, 0001, ..., 0, 01)$
Beta 1 (AdamW)	Uniforme em $(0, 10, ..., 0, 99)$
Beta 2 (AdamW)	Uniforme em $(0, 100, ..., 0, 999)$

Tabela 8 – Conjunto de valores utilizados pela Busca Aleatória nos modelos CNN.

da biblioteca Scikit-learn, foram realizadas 30 iterações de treinamento e estimativa da localização da origem do pacote. A Figura 19 demonstra de forma simplificada as etapas para realizar as estimativas utilizando o modelo kNN regressor. É utilizado como dados de entrada uma lista de valores de RSSI, tanto do RSSI padrão quanto do RSSI *Powed*, que é referente à um pacote. O modelo irá apresentar como resultado de sua estimativa um par de coordenadas, em pixels, que representa a localização de origem do respectivo pacote.

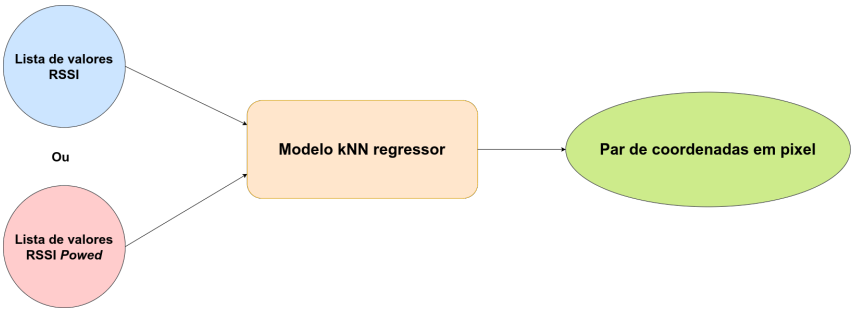


Figura 19 – Fluxograma do processo de estimativa do modelo kNN.

kNN - 27 estações		
Dados de entrada	RSSI	RSSI Powed
k	30	31
Métrica	manhattan	l2

Tabela 9 – Valores de hiperparâmetros encontrados pela Busca Aleatória para o modelo kNN usando os dados das 27 estações.

kNN - 2 estações		
Dados de entrada	RSSI	RSSI Powed
k	50	38
Métrica	cosseno	cosseno

Tabela 10 – Valores de hiperparâmetros encontrados pela Busca Aleatória para o modelo kNN usando os dados de 2 as estações.

4.3.3 RBF

A opção pelo modelo RBF se deu pela aparente similaridade da estrutura tradicional do modelo com a estrutura dos dados utilizados. O vetor central, ou centro dos neurônios da camada oculta, de uma RBFN consiste em pontos de interesse presentes no espaço da amostra dos dados de entrada, e cada neurônio possui um peso atrelado, que define a sua importância para a estimativa do resultado. Já os dados utilizados nos experimentos propostos possuem coordenadas de estações (pontos) localizadas em uma área definida (espaço), onde cada uma, para cada pacote, apresentou um valor de RSSI, que pode ser utilizado para identificar qual estação estaria mais próxima da origem do pacote, ou seja, usar o RSSI como peso. Logo, a proposta é utilizar as estações e valores de RSSI dentro de uma rede RBF, considerando o pseudocódigo da RBFN clássica demonstrado na Seção 2.4.1 e assumindo as estações como os centros dos neurônios de uma camada oculta e os valores de RSSI como os pesos de saída, a fim de verificar se essa semelhança apresentará um bom desempenho nos resultados.

Neste experimento, utilizando a biblioteca Keras com a linguagem Python, foram montadas e testadas 2 variações do modelo RBF. Por padrão, redes do tipo RBF possuem apenas uma camada oculta, além da de entrada e saída, sendo essa uma das variações. No entanto, foi testado também um cenário buscando otimizar o número de camadas ocultas, variando entre 1 a 6 camadas usando Busca Aleatória. Para cada uma das configurações foram executados 100 conjuntos diferentes de valores dos hiperparâmetros. A função de ativação da camada de saída utilizada foi uma função linear padrão da biblioteca utilizada, do tipo $f(x) = x$. No cenário que buscou otimizar o número de camadas, a quantidade de camadas ocultas selecionada foi 5, com exceção apenas do cenário de 27 estações com RSSI *Powed*, onde 3 camadas ocultas foi a quantidade considerada com melhor desempenho. O módulo Keras-Tuner foi utilizado para a execução da Busca Aleatória.

As Tabelas 11 e 12 mostram os valores dos modelos que utilizaram dados das 27

estações, enquanto as Tabelas 13 e 14 são dos modelos que utilizaram dados das 2 estações selecionadas. A Figura 20 demonstra o fluxo executado para a realização de estimativas do modelo RBF. Da mesma forma que no modelo kNN, listas de valores de RSSI (padrão e *Powed*) são utilizadas como dados de entrada para o modelo RBF, e no final é obtida a estimativa da localização de origem do pacote, em pixels.

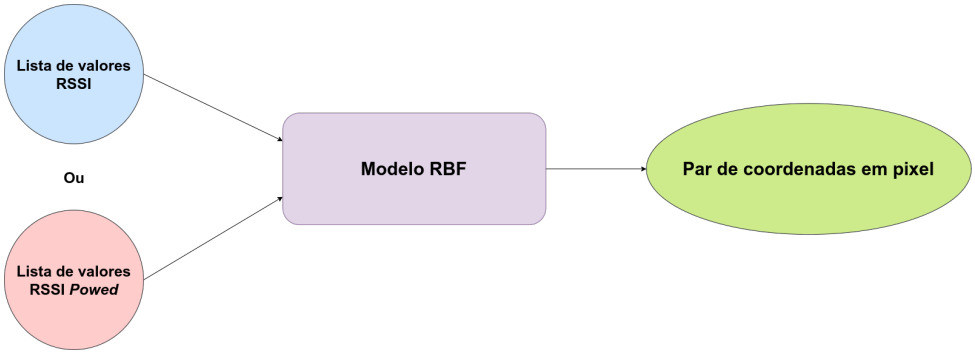


Figura 20 – Fluxograma do processo de estimativa do modelo RBF.

RBF - 27 estações		
Dados de entrada	RSSI	RSSI Powed
Função de Ativação	Tangente Hiperbólica	Tangente Hiperbólica
Neurônios camada Oculta	1712	1057
Dropout camada oculta	0,3487	0,1508
Taxa aprendizagem	0,0004	0,0046
Beta 1	0,6919	0,1903
Beta 2	0,7146	0,5306

Tabela 11 – Valores de hiperparâmetros encontrados pela Busca Aleatória para o modelo RBF usando os dados das 27 estações.

RBF com 5 camadas - 27 estações		
Dados de entrada	RSSI	RSSI Powed
Função de Ativação	Tangente Hiperbólica	ReLu
Qtde camadas ocultas	5	3
Neurônios camada oculta 1	1067	1355
Dropout camada oculta 1	0,4028	0,1442
Neurônios camada oculta 2	756	1574
Dropout camada oculta 2	0,1403	0,3798
Neurônios camada oculta 3	864	1229
Dropout camada oculta 3	0,2593	0,1110
Neurônios camada oculta 4	1482	–
Dropout camada oculta 4	0,2423	–
Neurônios camada oculta 5	541	–
Dropout camada oculta 5	0,2956	–
Taxa aprendizagem	0,0006	0,0012
Beta 1	0,5642	0,3628
Beta 2	0,8627	0,4757

Tabela 12 – Valores de hiperparâmetros encontrados pela Busca Aleatória para o modelo RBF com quantidade de camadas otimizadas usando os dados das 27 estações.

RBF - 2 estações		
Dados de entrada	RSSI	RSSI Powed
Função de Ativação	ReLu	Tangente Hiperbólica
Neurônios camada oculta	643	903
Dropout camada oculta	0,1065	0,2405
Taxa aprendizagem	0,0051	0,0038
Beta 1	0,7881	0,9311
Beta 2	0,5198	0,9829

Tabela 13 – Valores de hiperparâmetros encontrados pela Busca Aleatória para o modelo RBF usando os dados de 2 estações.

RBF com 5 camadas - 2 estações		
Dados de entrada	RSSI	RSSI Powed
Função de Ativação	ReLu	Tangente Hiperbólica
Qtde camadas ocultas	5	5
Neurônios camada oculta 1	144	1965
Dropout camada oculta 1	0,3768	0,3252
Neurônios camada oculta 2	1855	412
Dropout camada oculta 2	0,2482	0,1369
Neurônios camada oculta 3	267	1547
Dropout camada oculta 3	0,1763	0,3682
Neurônios camada oculta 4	1400	95
Dropout camada oculta 4	0,3330	0,1507
Neurônios camada oculta 5	1422	765
Dropout camada oculta 5	0,1571	0,2095
Taxa aprendizagem	0,0003	0,0015
Beta 1	0,8967	0,5275
Beta 2	0,6974	0,3683

Tabela 14 – Valores de hiperparâmetros encontrados pela Busca Aleatória para o modelo RBF com quantidade de camadas otimizadas usando os dados de 2 estações.

4.3.4 CNN

Os modelos de CNN foram configurados usando hiperparâmetros encontrados por meio da Busca Aleatória, inclusive a quantidade de camadas ocultas na rede. A Busca Aleatória testou 100 diferentes combinações de valores em cada um dos 2 cenários diferentes. Para os 2 cenários, foi utilizado o mesmo modelo com a Busca Aleatória, dispondo dos mesmos conjuntos de valores para cada hiperparâmetro. Para o modelo CNN, foi utilizada a biblioteca Keras, e a Keras-Tuner para a execução da Busca Aleatória. A Figura 21 demonstra o processo de estimar as coordenadas de localização, em pixels, utilizando o modelo CNN. Os dados de entrada utilizados foram as imagens geradas com os valores de RSSI e localização de estações que receberam cada um dos pacotes, aplicando em cada uma a técnica de EDK.

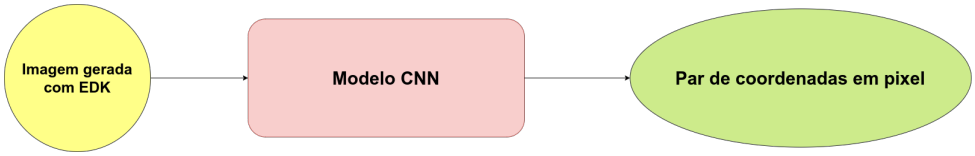


Figura 21 – Fluxograma do processo de estimativa do modelo CNN.

As imagens geradas foram carregadas usando um *Generator*, permitindo que fossem carregadas na memória de forma iterativa. Isso foi necessário pois a infraestrutura não permitia carregar todas as imagens de uma vez. Para o uso desse recurso, foi utilizada a biblioteca Tensorflow[9] na sua versão 2.11.0, fazendo uso também do recurso de GPU (*Graphics Processing Unit*) do ambiente de execução. As Tabelas 15 e 16 mostram os valores dos hiperparâmetros usados nos modelos treinados. O processo de treinamento de cada modelo e estimativa das coordenadas foi repetido por 30 vezes, com os resultados de cada uma armazenados individualmente, além de dados como tempo de treino, tempo de previsão e as métricas de desempenho.

[9] Tensorflow - https://www.tensorflow.org/

CNN - 27 estações	
Entrada	Imagens com todas as estações plotadas utilizando KDE.
Função de ativação	ReLu
Qtde camadas ocultas	5
Neurônios camada convol/pooling 1	78
Neurônios camada convol/pooling 2	65
Dropout camada convol/pooling 2	0,3992
Neurônios camada convol/pooling 3	97
Dropout camada convol/pooling 3	0,4236
Neurônios camada convol/pooling 4	93
Dropout camada convol/pooling 4	0,4671
Neurônios camada convol/pooling 5	64
Dropout camada convol/pooling 5	0,2873
Neurônios camada densa	83
Beta 1	0,6918
Beta 2	0,9167
Taxa de aprendizagem	0,0047

Tabela 15 – Valores de hiperparâmetros encontrados pela Busca Aleatória para o modelo CNN usando os dados das 27 estações.

CNN - 2 estações	
Entrada	Imagens com duas estações plotadas utilizando KDE.
Função de ativação	ReLu
Qtde camadas ocultas	5
Neurônios camada convol/pooling 1	90
Neurônios camada convol/pooling 2	21
Dropout camada convol/pooling 2	0,1575
Neurônios camada convol/pooling 3	15
Dropout camada convol/pooling 3	0,1717
Neurônios camada convol/pooling 4	86
Dropout camada convol/pooling 4	0,2217
Neurônios camada densa	96
Beta 1	0,2654
Beta 2	0,8988
Taxa de aprendizagem	0,0044

Tabela 16 – Valores de hiperparâmetros encontrados pela Busca Aleatória para o modelo CNN usando os dados de 2 estações.

4.4 Métricas de desempenho

A métrica utilizada para avaliar o desempenho foi o Erro Médio Absoluto (MAE), como também utilizada em Anagnostopoulos e Kalousis (2019) e Zhang et al. (2019). Durante o treinamento e os testes executados pela Busca Aleatória, essa foi a métrica utilizada como parâmetro e aplicada ao conjunto de validação, em que durante o processo o erro era monitorado. A métrica MAE é de fácil interpretabilidade e menos sensível a *outliers*, identificados nos modelos propostos durante a avaliação dos resultados.

Já na etapa de estimação das coordenadas, foi calculada a distância euclidiana entre o valor estimado com o valor real de cada coordenada, onde no final foi calculada a média de todas essas distâncias, obtendo o erro médio da previsão. Além do erro médio das estimativas, também foi obtido em cima da lista de distâncias euclidianas o desvio padrão e a mediana.

5 Resultados e Discussões

Neste Capítulo serão apresentados os resultados das análises comparativas entre os modelos de AM e os métodos analíticos TDoA e RSSI *fingerprinting* que serão considerados como *baseline*. Nos resultados obtidos, foram realizados testes de hipótese para avaliar a equivalência entre eles, sendo os testes de Kruskal-Wallis e Wilcoxon, ambos com nível de confiança de 95%.

Para os modelos kNN e RBF foram utilizados como dados de entrada os valores de RSSI comum e o RSSI *Powed*, valor obtido com a transformação *Powed* de Torres-Sospedra et al. (2015), enquanto o modelo CNN utilizou as imagens geradas com aplicação do EDK como dados de entrada. Para os modelos analíticos tradicionais, o tempo de chegada foi utilizado como entada para o TDoA e o RSSI comum em conjunto com o SNR serviram como entrada para o cálculo do RSSI *fingerprinting*.

O Kruskal-Wallis realiza o teste no grupo como um todo, sem entrar em detalhes de cada modelo em si. Por conta disso, foi também executado um teste que avaliasse par a par os modelos utilizados, sendo escolhido o Wilcoxon. Como este trabalho utiliza apenas 2 bases de dados diferentes, uma para cada cenário, inviabiliza testes como o ANOVA, que desempenham melhor em experimentos com mais bases de dados ou características. O teste Kruskal-Wallis foi executado usando a biblioteca SciPy[1]. Já os testes Wilcoxon foram executados usando a biblioteca scikit-posthocs[2] e tiveram seus *p-values* ajustados pelo método Bonferroni, devido aos múltiplos testes de hipótese realizados entre os resultados.

Os erros das estimativas de localização estão demonstrados em pixels. Como foi explicado na Seção 4.1.3, a área onde ocorreram as transmissões da base de dados foi transformada em um grid de tamanho 128 x 128, em que cada célula foi chamada de pixels. Então, esse pixel foi escolhido como unidade de medida para calcular o erro estimado dos modelos. Na Tabela 17 são demonstrados os termos utilizados na apresentação dos resultados para cada modelo, tanto no cenário com 27 estações quanto no cenário com 2 estações.

[1] Biblioteca SciPy - https://scipy.org
[2] Biblioteca scikit-posthocs - https://scikit-posthocs.readthedocs.io/

Termo	Modelo
RBF	RBF com 1 camada oculta e entrada RSSI
RBF-POWED	RBF com 1 camada oculta e entrada RSSI *Powed*
RBF-5	RBF com 5 camadas ocultas e entrada RSSI
RBF-3-POWED	RBF com 3 camadas ocultas e entrada RSSI *Powed*
RBF-5-POWED	RBF com 5 camadas ocultas e entrada RSSI *Powed*
KNN	kNN e entrada RSSI
KNN-POWED	kNN e entrada RSSI *Powed*
CNN	CNN
TDOA	TDoA
RSSI-FINGERPRINTING	RSSI *fingerprinting*

Tabela 17 – Termos e Modelos.

5.1 Estimativas de localização utilizando-se todas as estações

No cenário utilizando dados de todas as estações consideradas relevantes para o experimento (27 estações), os modelos RBF, CNN e kNN foram executados usando como dados de entrada tanto o RSSI inalterado quanto o RSSI com a transformação *Powed*, onde cada modelo foi executado 30 vezes para cada uma dessas entradas. O TDoA e RSSI *fingerprinting* foram executados apenas uma vez pelo fato de serem determinísticos, ou seja, seus resultados serão sempre os mesmos caso seja utilizado as mesmas entradas. Na Figura 22 estão plotadas as médias de cada uma das 30 execuções. Os resultados dos modelos propostos neste trabalho apresentaram resultados próximos, onde o modelo RBF com os dados de RSSI transformados com a função *Powed* demonstrou ter um melhor resultado. Dentre os modelos, a CNN foi a que teve maior variabilidade nos dados.

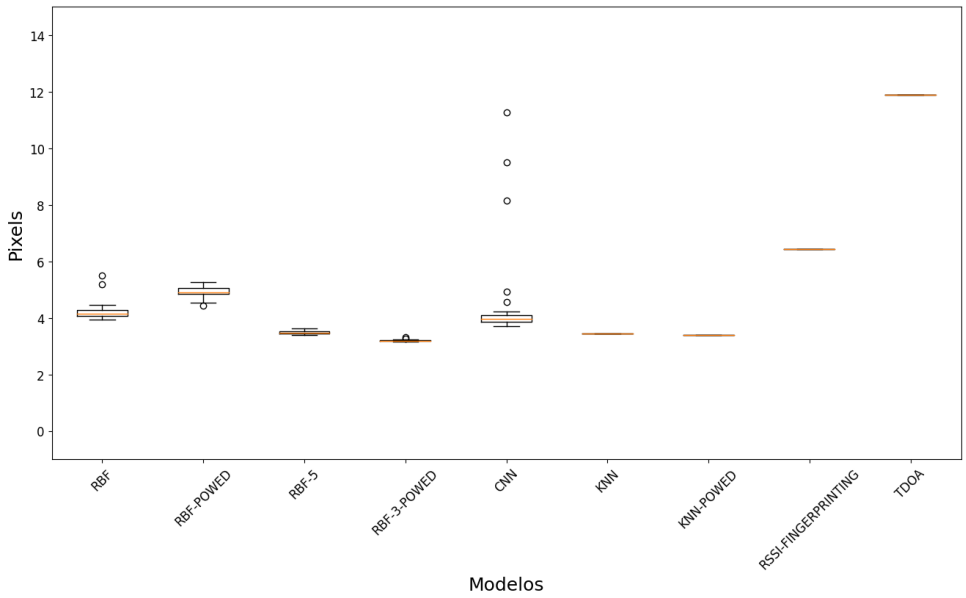


Figura 22 – Média do erro da predição das coordenadas utilizando dados das 27 estações.

O TDoA apresentou o pior desempenho nas estimativas com os dados utilizados, e

além disso, por necessitar que pelo menos 3 estações tenham recebido o pacote, ele não conseguiu estimar a localização de 7675 pacotes, visto que do total de pacotes presentes no conjunto de testes apenas 5368 foram recebidos por pelo menos 3 estações. Já para o RSSI *fingerprinting*, o requisito era ter pelo menos uma estação que recebeu o respectivo pacote, e no conjunto de testes 12982 pacotes cumpria esse requisito. Não aproveitar dados de pacotes que não foram recebidos por nenhuma estacão pode ser um desperdício, pois podem conter informações geoespaciais importantes, especialmente se estiverem relacionados a locais onde a transmissão não foi captada. Os modelos de Aprendizagem de Máquina utilizam também esses dados durante os treinos.

Com o TDoa, 1391 pacotes não puderam ter sua localização estimada, situação prevista pelo serviço da API utilizada, porém sem detalhes dos motivos. As localizações desses pacotes foram verificadas no mapa para avaliar se existia alguma característica comum a esses pacotes que pudesse explicar essa situação, porém eles estão espalhados por toda a área estudada, não possibilitando identificar, a princípio, algo que pudesse explicar esta limitação da API.

Foram realizados testes de hipótese dos tipos Kruskal-Wallis e Wilcoxon com os modelos testados, utilizando os respectivos valores das médias de erro das coordenadas estimadas. De acordo com o resultado do Kruskal-Wallis, foi possível rejeitar a hipótese nula de que as médias são iguais, com *p-value* de $7{,}123 \times 10^{-38}$. Para interpretar os resultados do Wilcoxon em ambos os cenários, foi utilizado um esquema de pontuações distribuídas em Vitórias, Derrotas e Empates. As Vitórias e Derrotas são contabilizadas quando a hipótese nula é rejeitada, ou seja, os valores são considerados estatisticamente diferentes, então um ponto de vitória é creditado no modelo que apresentar a menor média de erro, e consequentemente um ponto de derrota para o outro modelo. É considerado Empate quando a hipótese nula não é rejeitada, creditando pontos em ambos os modelos. A Tabela 18 mostra os resultados obtidos com Wilcoxon. Dentre os modelos, o RBF 3 camadas usando RSSI transformado com a função *Powed* foi o que obteve mais vitórias, sem contabilizar derrotas ou empates.

Na Tabela 19 estão as médias dos valores mostrados na Figura 22. Pelas médias é possível visualizar que o modelo RBF-3-POWED obteve uma menor média frente aos demais. Esse desempenho corrobora com a intuição de que as estações se comportariam como centros de neurônios e RSSI como pesos em uma RBFN trariam resultados satisfatórios, dada a similaridade nas suas estruturas. Não somente obteve um bom desempenho, mas que foi o melhor.

Modelos	Vitórias	Derrotas	Empates
RBF-3-POWED	8		
KNN-POWED	7	1	
RBF-5	5	2	1
KNN	5	2	1
RBF	3	4	1
RBF-POWED	2	5	1
CNN	2	4	2
RSSI-FINGERPRINTING	1	7	
TDOA		8	

Tabela 18 – Pontuação de Vitórias, Derrotas e Empates das médias de erro das estimativas do cenário de 27 estações, de acordo com o teste Wilcoxon.

Modelo	Média do erro (pixels)
RBF-3-POWED	3,20
KNN-POWED	3,41
KNN	3,46
RBF-5	3,49
RBF	4,23
CNN	4,55
RBF-POWED	4,92
RSSI-FINGERPRINTING	6,44
TDOA	11,88

Tabela 19 – Média dos erros de cada modelo no cenário usando 27 estações.

Nas Figuras 23 e 24 estão os dados referentes aos tempos de treino e predição dos modelos em cada uma das 30 execuções, respectivamente. A primeira vista, é possível visualizar que o modelo CNN demanda mais tempo no treino que os demais, os quais apresentaram tempos menores e de menor variabilidade. Contudo, durante a predição, foram os modelos kNN que apresentaram maior variabilidade e tempo, situação em que os modelos RBF tiveram melhor desempenho. Na Tabela 20 estão as médias das 30 execuções para cada modelo.

Modelo	Média do tempo de treino	Média do tempo de estimativa
RBF-3-POWED	368,39	20,05
KNN-POWED*	0,00	106,54
KNN*	0,00	122,63
RBF-5	197,07	22,41
RBF	80,33	23,82
CNN	1202,18	10,27
RBF-POWED	108,40	19,11

Tabela 20 – Média dos tempos de treino e estimativa de cada modelo no cenário usando 27 estações em segundos (*kNN não necessita treino).

Foram executados também os testes de hipótese para os tempos de treino e predição.

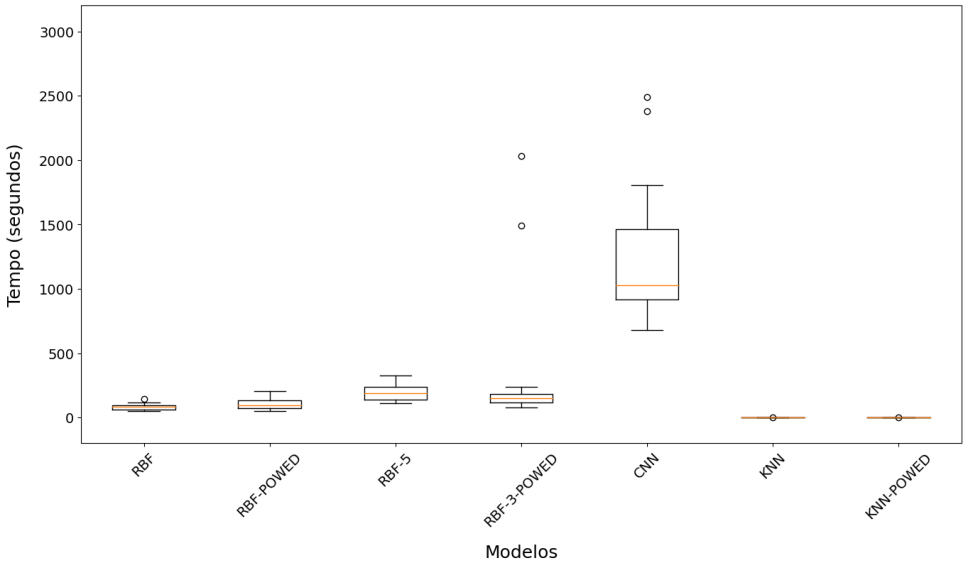


Figura 23 – Média do tempo para treinar os modelos utilizando dados 27 estações.

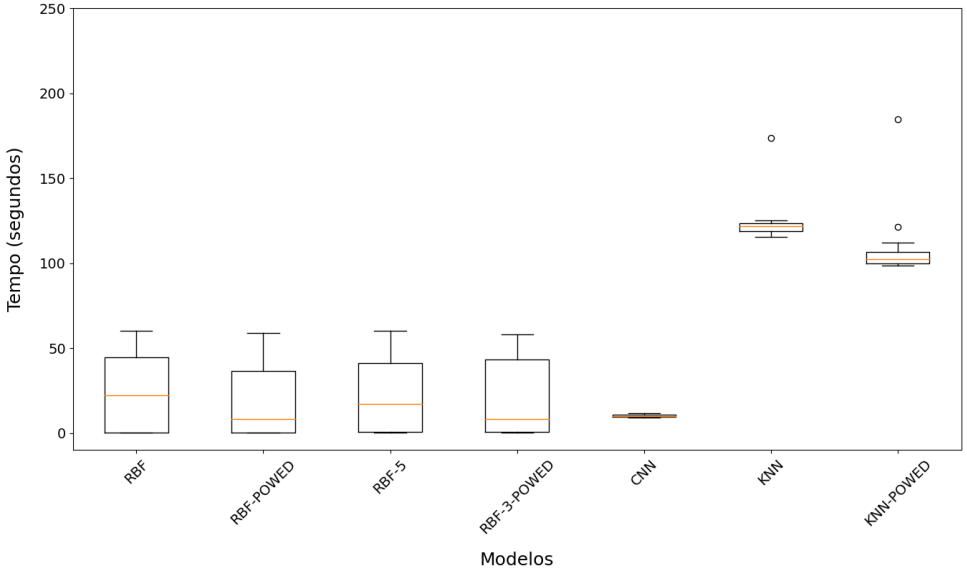


Figura 24 – Média do tempo para realizar as estimativas das coordenadas utilizando dados 27 estações.

Entre os tempos de treino, de acordo com o teste Wilcoxon, O kNN utilizando o RSSI inalterado foi o que obteve mais vitórias, sem derrotas ou empates. Já entre os tempos de predição, os resultados entre os modelos foram mais próximos, não havendo um modelo específico que se sobressaiu. Nas Tabelas 21 e 22 estão as pontuações de todos os modelos. O teste Kruskal-Wallis apresentou *p-value* de $6{,}200 \times 10^{-37}$ e $5{,}743 \times 10^{-26}$ para os tempos de treino e estimativa, respectivamente, sendo plausível rejeitar a hipótese nula de que os valores são equivalentes.

Modelos	**Vitórias**	**Derrotas**	**Empates**
KNN	6		
KNN-POWED	5	1	
RBF	3	2	1
RBF-POWED	3	2	1
RBF-5	1	4	1
RBF-3-POWED	1	4	1
CNN		6	

Tabela 21 – Pontuação de Vitórias, Derrotas e Empates dos tempos de treino do cenário de 27 estações, de acordo com o teste Wilcoxon.

Modelos	**Vitórias**	**Derrotas**	**Empates**
CNN	2		4
RBF	2		4
RBF-POWED	2		4
RBF-5	2		4
RBF-3-POWED	2		4
KNN-POWED	1	5	
KNN		6	

Tabela 22 – Pontuação de Vitórias, Derrotas e Empates dos tempos de predição do cenário de 27 estações, de acordo com o teste Wilcoxon.

Para o TDoA e RSSI *fingerprinting* não foram coletados tempos porque, sendo um serviço utilizado por meio de consultas a um servidor da internet, muitos parâmetros influenciam no tempo, como latência da rede, taxas de envio de dados e qualidade de conexão. Logo, não seria uma comparação justa com os demais, executados em máquina local ou em um servidor dedicado, sem haver a troca de dados pela internet durante a execução.

5.2 Estimativas de localização utilizando-se duas estações

Os erros de predição calculados podem ser vistos na Figura 25. O modelo CNN continuou apresentando maior variabilidade nas médias dos experimentos, em comparação ao teste com as 27 estações. Já os resultados do RBF e kNN apresentaram resultados bem próximos.

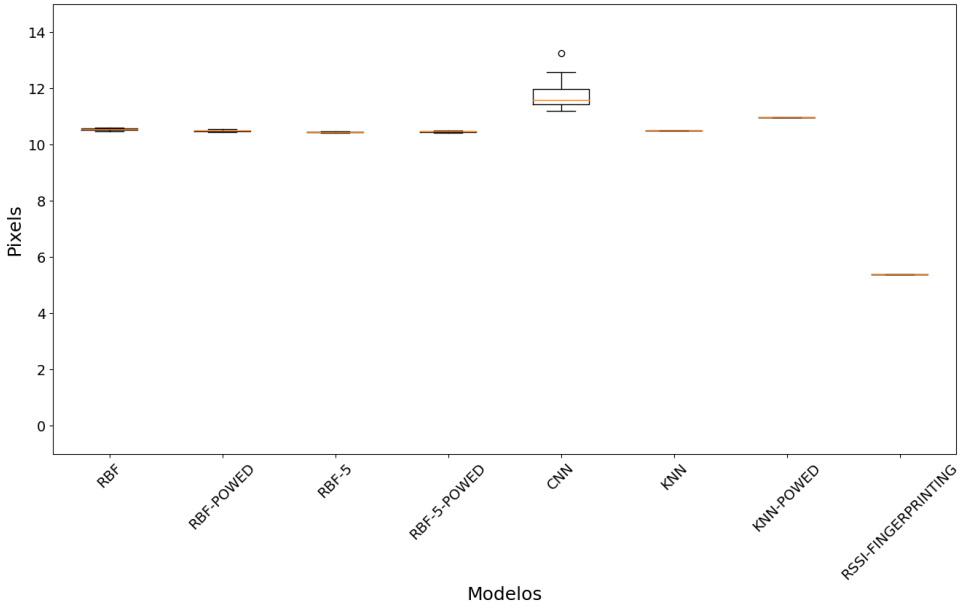


Figura 25 – Média do erro da predição das coordenadas utilizando-se duas estações.

O teste com o TDoA tiveram algumas ressalvas. Primeiro, não foi possível utilizar pacotes não recebidos por nenhuma estação, situação em que nos modelos de AM esses pacotes também foram utilizados. Isso fez com que a quantidade de pacotes reduzisse de 13043 para 6813. Segundo, utilizando o método TDoA, o serviço não conseguiu realizar nenhuma predição, muito provavelmente devido a limitação inerente à própria técnica. Contudo, utilizando o RSSI *fingerprinting* foi possível realizar a estimativa de localização de todos os 6813 pacotes.

De acordo com as pontuações obtidas dos resultados do Wilcoxon, mostradas na Tabela 23, O RSSI *fingerprinting* teve mais vitórias e nenhum empate ou derrota, seguido do RBF-5 e RBF-5-POWED. O *p-value* do Kruskal-Wallis, de valor $4{,}227 \times 10^{-38}$, permitindo rejeitar a hipótese nula, sendo os valores considerados estatisticamente diferentes.

Modelos	**Vitórias**	**Derrotas**	**Empates**
RSSI-FINGERPRINTING	7		
RBF-5	5	1	1
RBF-5-POWED	5	1	1
RBF-POWED	3	3	1
KNN	3	3	1
RBF	2	5	
KNN-POWED	1	6	
CNN		7	

Tabela 23 – Pontuação de Vitórias, Derrotas e Empates das médias de erro das estimativas do cenário de 2 estações, de acordo com o teste Wilcoxon.

Na Tabela 24 estão as médias dos valores mostrados na Figura 25. Pelas médias é possível visualizar que a melhor média foi a do RSSI *fingerprinting*, seguido do modelo RBF de 5 camadas, com ambos os tipos de entrada RSSI (inalterado e transformado *Powed*), como sugeriu os testes Wilcoxon.

Modelo	**Média do erro (pixels)**
RSSI-FINGERPRINTING	5,37
RBF-5	10,45
RBF-5-POWED	10,46
KNN	10,48
RBF-POWED	10,49
RBF	10,54
KNN-POWED	10,97
CNN	11,73

Tabela 24 – Média dos erros de cada modelo para o cenário com 2 estações.

As Figuras 26 e 27 mostram os tempos de treino e predição para os testes com duas estações. Da mesma forma, o modelo CNN apresentou tempos maiores nos treinos, e maior variabilidade, e nas predições o kNN que demandou mais tempo para concluir. O RBF apresentou tempos mais consistentes e próximos nos dois estágios. O tempo do RSSI *fingerprinting* não foi coletado pelo mesmo motivo explicado no teste com 27 estações. Na Tabela 25 estão as médias dos tempos de treino e predição das execuções para cada modelo.

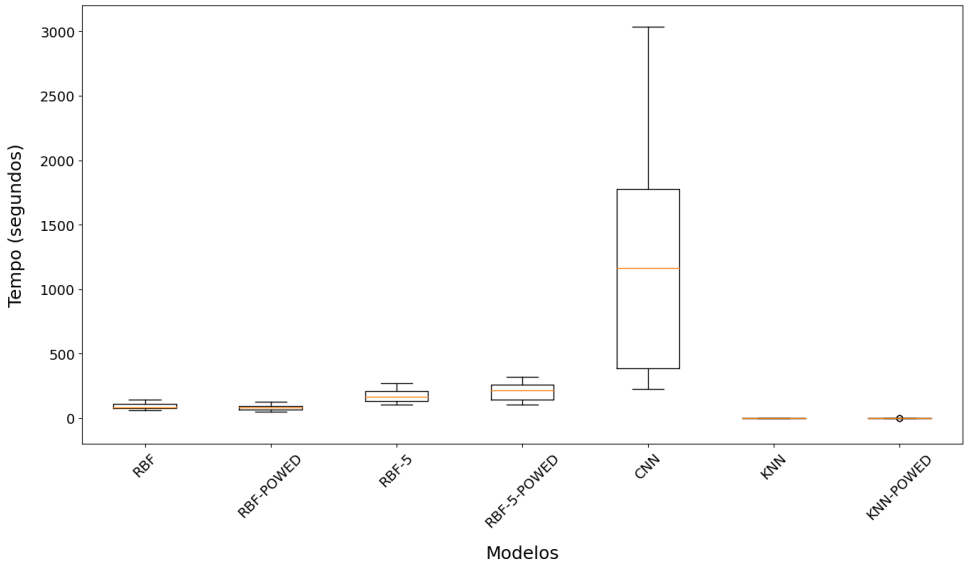


Figura 26 – Média do tempo para treinar os modelos utilizando 2 estações.

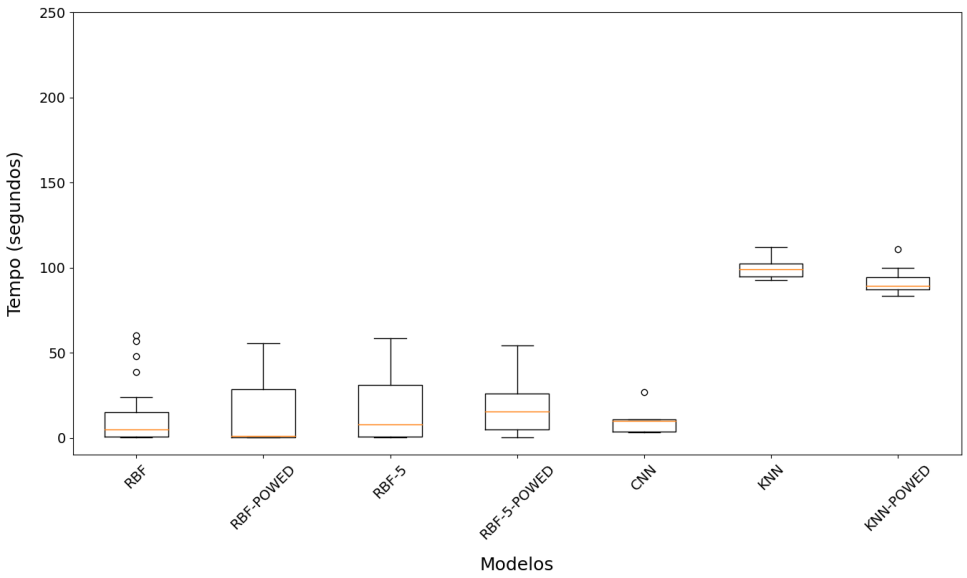


Figura 27 – Média do tempo para realizar as predições das coordenadas utilizando 2 estações.

Modelo	**Média do tempo de treino**	**Média do tempo de predição**
RBF-5	174,32	16,20
RBF-5-POWED	209,53	17,72
KNN*	0,00	99,41
RBF-POWED	82,61	15,80
RBF	95,20	11,84
KNN-POWED*	0,00	90,70
CNN	1236,76	8,12

Tabela 25 – Média dos tempos de treino e predição de cada modelo no cenário usando 2 estações em segundos (*kNN não necessita treino).

Sobre os testes Wilcoxon sobre os tempos de treino e predição, foi verificado que seguiu a mesma tendência do cenário com 27 estações, onde o kNN com RSSI inalterado obteve mais vitórias no treino, enquanto na predição os resultados foram mais aproximados, como mostra nas Tabelas 26 e 27. O Kruskal-Wallis obteve o *p-value* de $3{,}999 \times 10^{-38}$ e $2{,}921 \times 10^{-26}$ em relação aos tempos de treino e estimativa, respectivamente, podendo assim rejeitar a hipótese nula de que os valores não são estatisticamente equivalentes.

Modelos	Vitórias	Derrotas	Empates
KNN	5		1
KNN-POWED	5		1
RBF	3	2	1
RBF-POWED	3	2	1
RBF-5	1	4	1
RBF-5-POWED	1	4	1
CNN		6	

Tabela 26 – Pontuação de Vitórias, Derrotas e Empates dos tempos de treino do cenário de 2 estações, de acordo com o teste Wilcoxon.

Modelos	Vitórias	Derrotas	Empates
CNN	2		4
RBF	2		4
RBF-POWED	2		4
RBF-5	2		4
RBF-5-POWED	2		4
KNN-POWED	1	5	
KNN		6	

Tabela 27 – Pontuação de Vitórias, Derrotas e Empates dos tempos de predição do cenário de 2 estações, de acordo com o teste Wilcoxon.

5.3 Impacto do número de estações nos modelos dos algoritmos

Ao comparar os resultados dos dois cenários, o com 27 estações e o outro com 2, é perceptível que no último as médias de erro são maiores do que no primeiro. Por exemplo, o modelo RBF 3 camadas com RSSI transformado com a função *Powed*, melhor desempenho no cenário com 27 estações, teve um aumento na média de erro de estimativa em $226,87\%$ no cenário de 2 estações. Um dos fatores que sustentam esses resultados são a quantidade de dados utilizada para realizar as estimativas, pois os modelos de aprendizagem de máquina conseguem um melhor desempenho com uma base de dados mais robusta.

Mas outro ponto a considerar é o custo de implantação de uma infraestrutura, onde um cenário com 2 estações tende a custar menos que um cenário de 27 estações, e a diferença no desempenho pode ser compensada por esse custo, ou seja, aceitar um desempenho inferior por um custo menor, porém que satisfaça as necessidades do usuário.

Na Tabela 28 estão as pontuações obtidas de acordo com os testes de Wilcoxon, utilizando dados de ambos os cenários (27 e 2 estações), e o prefixo "2BS-" significa *two base stations*, representando os modelos do cenário de 2 estações. Os modelos executados no cenário com 27 estações assumem as primeiras colocações, com o RBF-3-POWED obtendo mais vitórias, sem empates ou derrotas. O TDoA não obteve nenhuma vitória, apresentando um dos piores desempenhos, junto com a CNN no cenário de 2 estações. O

RSSI *fingerprinting* teve um desempenho mediano, considerando ambos os cenários.

Modelo	**Vitórias**	**Derrotas**	**Empates**
RBF-3-POWED	16		
KNN-POWED	15	1	
RBF-5	13	2	1
KNN	13	2	1
RBF	11	4	1
RBF-POWED	10	5	1
CNN	9	4	3
2BS-RSSI-FINGERPRINTING	9	6	1
RSSI-FINGERPRINTING	8	8	
2BS-RBF-5	6	9	1
2BS-RBF-5-POWED	6	9	1
2BS-RBF-POWED	4	11	1
2BS-KNN	4	11	1
2BS-RBF	3	13	
2BS-KNN-POWED	2	14	
TDOA		15	1
2BS-CNN		15	1

Tabela 28 – Pontuação de Vitórias, Derrotas e Empates utilizando médias de erro dos 2 cenários juntos (27 e 2 estações), de acordo com o teste Wilcoxon.

No tocante ao tempo de treino e predição, os modelos de um modo geral tiveram desempenhos bem próximos na etapa de realizar as estimativas, mas a etapa de treino tiveram mais diferenças. É notório que o modelo CNN demandou mais tempo para seu período de treino, e isso pode ser justificado pelo tipo dos dados de entrada, que foram imagens de resolução 128×128 pixels, e para lidar com esse tipo de entrada é natural demandar mais tempo e recursos computacionais. Nas Tabelas 29 e 30 estão as pontuações para os tempos de treino e predição, respectivamente. Nesse quesito, os modelos demonstraram serem um elemento decisivo, dado que no treino o kNN obteve mais vitórias em ambos os cenários, tanto com RSSI inalterado quanto com RSSI *Powed*, e os executados no cenário com 2 estações ocuparam o topo da tabela. Na predição os resultados foram mais próximos, com pouca diferença entre os modelos. Uma explicação para esses resultados seria a de menos dados para serem processados durante o treino e as estimativas. No cenário com 2 estações, o RSSI *fingerprinting* apresentou a menor média de erro, todavia, vale destacar que é um serviço oferecido por uma organização privada, que inclui serviços pagos, sendo esperado que os resultados sejam satisfatórios. Outro fator também é a consideração de um parâmetro extra no cálculo da estimativa, que é o SNR, valor esse presente no conjunto de dados utilizado neste trabalho e que poderia ser utilizado pelo serviço privado para obter os resultados.

Modelos	**Vitórias**	**Derrotas**	**Empates**
2BS-KNN	12		1
2BS-KNN-POWED	12		1
KNN	11	2	
KNN-POWED	10	3	
RBF	6	4	3
2BS-RBF-POWED	6	4	3
RBF-POWED	6	4	3
2BS-RBF	6	4	3
2BS-RBF-5	2	8	3
RBF-5	2	8	3
RBF-3-POWED	2	8	3
2BS-RBF-5-POWED	2	8	3
CNN		12	1
2BS-CNN		12	1

Tabela 29 – Pontuação de Vitórias, Derrotas e Empates dos tempos de treino utilizando dados dos 2 cenários juntos (27 e 2 estações), de acordo com o teste Wilcoxon.

Modelos	**Vitórias**	**Derrotas**	**Empates**
2BS-CNN	4		9
CNN	4		9
2BS-RBF 4		9	
RBF	4		9
RBF-POWED	4		9
RBF-5	4		9
RBF-3-POWED	4		9
2BS-RBF-POWED	4		9
2BS-RBF-5	4		9
2BS-RBF-5-POWED	4		9
2BS-KNN-POWED	3	10	
2BS-KNN	2	11	
KNN-POWED	1	12	
KNN		13	

Tabela 30 – Pontuação de Vitórias, Derrotas e Empates dos tempos de predição utilizando dados dos 2 cenários juntos (27 e 2 estações), de acordo com o teste Wilcoxon.

6 Conclusões e Trabalhos Futuros

Neste trabalho foi realizado um comparativo entre modelos de Aprendizado de Máquina e Aprendizado Profundo para estimar a localização original de pacotes enviados por meio de uma rede LoRa, tendo os respectivos resultados comparados com métodos analíticos tradicionalmente utilizados, o TDoA e o RSSI *fingerprinting*. O uso de modelos de inteligência artificial ajuda a superar limitações que métodos analíticos clássicos possuem. O TDoA, por exemplo, é uma técnica bastante utilizada para resolver problemas que este trabalho buscou solucionar, no entanto, cenários com poucas estações e muitos obstáculos podem atrapalhar seu desempenho, pois depende fortemente de aspectos da propagação do sinal, especificamente o tempo. A inteligência artificial tem capacidade de superar esses obstáculos, pois os modelos identificam os padrões existentes nos dados que técnicas analíticas não conseguem, mesmo com uma infraestrutura mais modesta, usando-os para seu próprio aprendizado.

Dentre os modelos utilizados, o RBF apresentou resultados bastante satisfatórios em relação aos demais, mostrando ser uma opção viável para resolver problemas similares ao abordado neste trabalho. A suposição levantada de que as estações e RSSI se encaixariam dentro do modelo RBF, se comportando como neurônios e pesos, respectivamente, demonstrou ser bem sucedida, podendo assim considerar que uma RBFN é um modelo AM que se adequa mais naturalmente ao tipo de problema que este trabalho buscou resolver.

O uso de duas estações para tentar obter a localização mostrou potencial, sendo este um cenário bastante interessante, dado que o TDoA precisa de pelo menos três estações para seu funcionamento. Por exemplo, essa abordagem pode ser empregada em uma infraestrutura com recursos tecnológicos limitados, onde a utilização de TDoA seria impraticável, resultando em custos mais baixos para a implementação de equipamentos, mas ainda assim proporcionando resultados satisfatórios. Os modelos RBF demonstraram também um bom desempenho nesse cenário.

Os intervalos de tempo de treino dos modelos foi uma característica importante de se observar, e a CNN apresentou o maior tempo de treino dentre eles, podendo chegar a quase 1 hora na infraestrutura utilizada. Isso é um fator relevante, pois outros modelos obtiveram resultados melhores com um tempo de treino menor, comprovado inclusive pelos testes de hipótese. Já com os intervalos de tempo das predições, a CNN foi a que melhor desempenhou, seguida da RBF. Com isso, o modelo RBF se mostrou como a melhor opção para resolver o problema enfrentado neste trabalho.

Os modelos aqui avaliados são passíveis de melhorias, onde pode-se vislumbrar como trabalhos futuros o uso de mais parâmetros do protocolo LoRa além do RSSI, como o SNR

e o *Spreading Factor*, visando melhorar a acurácia dos resultados. Variar os parâmetros do EDK, tentando gerar imagens com diferentes valores de largura de banda, por exemplo, e avaliar os resultados obtidos. Outro estudo também poderia ser testar os cenários com diferentes quantidades de estações, e a partir daí avaliar o impacto no erro calculado da estimativa.

Outro ponto de melhoria seria tentar utilizar um grid de pixels com maior resolução para obter um pixel de menor área, almejando obter uma melhoria na acurácia, considerando que o erro médio se mantivesse estável comparado com os resultados encontrados neste trabalho. Isso necessitaria também de uma melhor infraestrutura, com mais recursos computacionais. Além disso, também utilizar diferentes formas de células de grid, como por exemplo a sugerida por Technologies (2024), que utiliza hexágonos para mapear as áreas urbanas devido à sua eficiência em minimizar erros de quantização durante os deslocamentos das pessoas na cidade, além de proporcionar uma fácil aproximação de raios, simplificando a análise geoespacial.

Referências

AERNOUTS, M.; BERKVENS, R. Sigfox and LoRaWAN Datasets for Fingerprint Localization in Large Urban and Rural Areas. p. 15, 2018. Disponível em: <https://www.mdpi.com/2306-5729/3/2/13>. Citado 5 vezes nas páginas 11, 22, 25, 26 e 27.

AERNOUTS, M. et al. *Sigfox and LoRaWAN Datasets for Fingerprint Localization in Large Urban and Rural Areas*. Zenodo, 2019. Disponível em: <https://doi.org/10.5281/zenodo.3904158>. Citado 3 vezes nas páginas 11, 27 e 28.

AGGARWAL, C. C. *Neural Networks and Deep Learning: A Textbook*. Cham: Springer International Publishing, 2018. ISBN 978-3-319-94462-3 978-3-319-94463-0. Disponível em: <http://link.springer.com/10.1007/978-3-319-94463-0>. Citado 4 vezes nas páginas 11, 15, 16 e 19.

ANAGNOSTOPOULOS, G. G.; KALOUSIS, A. A Reproducible Comparison of RSSI Fingerprinting Localization Methods Using LoRaWAN. In: *2019 16th Workshop on Positioning, Navigation and Communications (WPNC)*. Bremen, Germany: IEEE, 2019. p. 1–6. ISBN 978-1-72812-082-9. Disponível em: <https://ieeexplore.ieee.org/document/8970177/>. Citado 5 vezes nas páginas 1, 22, 25, 34 e 45.

ANAGNOSTOPOULOS, G. G.; KALOUSIS, A. Can I Trust This Location Estimate? Reproducibly Benchmarking the Methods of Dynamic Accuracy Estimation of Localization. *Sensors*, v. 22, n. 3, p. 1088, jan. 2022. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/22/3/1088>. Citado na página 28.

ANJUM, M. et al. RSSI Fingerprinting-based Localization Using Machine Learning in LoRa Networks. *arXiv:2006.01278 [cs, eess, math]*, jun. 2020. ArXiv: 2006.01278. Disponível em: <http://arxiv.org/abs/2006.01278>. Citado na página 1.

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. *Journal of machine learning research*, v. 13, n. 2, 2012. Citado 2 vezes nas páginas 20 e 37.

BISHOP, C. M. *Pattern recognition and machine learning*. New York: Springer, 2006. (Information science and statistics). ISBN 978-0-387-31073-2. Citado 4 vezes nas páginas 12, 14, 17 e 20.

DARAMOUSKAS, I.; KAPOULAS, V.; PEGIAZIS, T. A survey of methods for location estimation on Low Power Wide Area Networks. In: *2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)*. PATRAS, Greece: IEEE, 2019. p. 1–4. ISBN 978-1-72814-959-2. Disponível em: <https://ieeexplore.ieee.org/document/8900701/>. Citado 2 vezes nas páginas 1 e 7.

DEVALAL, S.; KARTHIKEYAN, A. LoRa Technology - An Overview. In: *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. Coimbatore: IEEE, 2018. p. 284–290. ISBN 978-1-5386-0965-1. Disponível em: <https://ieeexplore.ieee.org/document/8474715/>. Citado 3 vezes nas páginas 2, 9 e 11.

DIENG, O.; PHAM, C.; THIARE, O. Outdoor Localization and Distance Estimation Based on Dynamic RSSI Measurements in LoRa Networks: Application to Cattle Rustling Prevention. In: *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. Barcelona, Spain: IEEE, 2019. p. 1–6. ISBN 978-1-72813-316-4. Disponível em: <https://ieeexplore.ieee.org/document/8923542/>. Citado 2 vezes nas páginas 1 e 3.

GIOIA, C.; SERMI, F.; TARCHI, D. Multi-Network Asynchronous TDOA Algorithm Test in a Simulated Maritime Scenario. *Sensors (Basel, Switzerland)*, v. 20, n. 7, mar. 2020. ISSN 1424-8220. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7180892/>. Citado na página 26.

GLASSNER, A. *Deep Learning: A Visual Approach*. [S.l.: s.n.], 2021. ISBN 978-1-7185-0073-0. Citado 3 vezes nas páginas 12, 15 e 17.

GRAMACKI, A. *Nonparametric Kernel Density Estimation and Its Computational Aspects*. Cham: Springer International Publishing, 2018. v. 37. (Studies in Big Data, v. 37). ISBN 978-3-319-71687-9 978-3-319-71688-6. Disponível em: <http://link.springer.com/10.1007/978-3-319-71688-6>. Citado 2 vezes nas páginas 20 e 21.

GROOMBRIDGE, D. *Gartner Top 10 Strategic Technology Trends for 2023*. 2022. Disponível em: <https://www.gartner.com/en/articles/gartner-top-10-strategic-technology-trends-for-2023>. Citado na página 3.

GU, C.; JIANG, L.; TAN, R. LoRa-Based Localization: Opportunities and Challenges. *arXiv:1812.11481 [cs]*, jan. 2019. ArXiv: 1812.11481. Disponível em: <http://arxiv.org/abs/1812.11481>. Citado na página 2.

JANSSEN, T.; BERKVENS, R.; WEYN, M. Comparing Machine Learning Algorithms for RSS-Based Localization in LPWAN. In: BAROLLI, L.; HELLINCKX, P.; NATWICHAI, J. (Ed.). *Advances on P2P, Parallel, Grid, Cloud and Internet Computing*. Cham: Springer International Publishing, 2020. v. 96, p. 726–735. ISBN 978-3-030-33508-3 978-3-030-33509-0. Series Title: Lecture Notes in Networks and Systems. Disponível em: <http://link.springer.com/10.1007/978-3-030-33509-0_68>. Citado na página 1.

KIM, K. S.; LEE, S.; HUANG, K. A scalable deep neural network architecture for multi-building and multi-floor indoor localization based on Wi-Fi fingerprinting. *Big Data Analytics*, v. 3, n. 1, p. 4, dez. 2018. ISSN 2058-6345. Disponível em: <https://bdataanalytics.biomedcentral.com/articles/10.1186/s41044-018-0031-2>. Citado na página 2.

KINGMA, D. P.; BA, J. Adam: A Method for Stochastic Optimization. arXiv, jan. 2017. Disponível em: <http://arxiv.org/abs/1412.6980>. Citado na página 18.

LIU, J. Radial basis function (rbf) neural network control for mechanical systems. In: _____. [S.l.: s.n.], 2013. p. 339–362. ISBN 978-3-642-34815-0. Citado na página 13.

LOSHCHILOV, I.; HUTTER, F. *Decoupled Weight Decay Regularization*. arXiv, 2017. Disponível em: <https://arxiv.org/abs/1711.05101>. Citado 2 vezes nas páginas 19 e 37.

MAGSI, S. A. et al. Support Vector Regression based Localization Approach using LoRaWAN. *International Journal of Advanced Computer Science and Applications*, v. 14, n. 3, 2023. ISSN 21565570, 2158107X. Disponível em: <http://thesai.org/Publications/ViewPaper?Volume=14&Issue=3&Code=IJACSA&SerialNo=35>. Citado 2 vezes nas páginas 23 e 25.

MENDOZA-SILVA, G. M. et al. Long-term wifi fingerprinting dataset for research on robust indoor positioning. *Data*, v. 3, n. 1, 2018. ISSN 2306-5729. Disponível em: <https://www.mdpi.com/2306-5729/3/1/3>. Citado na página 23.

MONTAGNY, S. *LoRa - LoRaWAN and Internet of Things for beginners*. Savoie Mont Blanc University, 2022. 132 p. Disponível em: <https://www.univ-smb.fr/lorawan/en/free-book/>. Citado na página 9.

NIITSOO, A. et al. A Deep Learning Approach to Position Estimation from Channel Impulse Responses. *Sensors*, v. 19, n. 5, p. 1064, mar. 2019. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/19/5/1064>. Citado na página 1.

SAND, S.; DAMMANN, A.; MENSING, C. *Positioning in wireless communications systems*. Chichester, West Sussex: Wiley, 2014. ISBN 978-0-470-77064-1. Citado 4 vezes nas páginas 1, 5, 8 e 36.

SCHIELTZ, J. M. et al. Gps tracking cattle as a monitoring tool for conservation and management. *African journal of range & forage science*, Taylor & Francis, v. 34, n. 3, p. 173–177, 2017. Citado na página 3.

SEMTECH. *What are LoRa and LoRaWAN?* 2024. <https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/>. Acessado: 28-01-2024. Citado na página 3.

SONG, X. et al. CNNLoc: Deep-Learning Based Indoor Localization with WiFi Fingerprinting. In: *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. Leicester, United Kingdom: IEEE, 2019. p. 589–595. ISBN 978-1-72814-034-6. Disponível em: <https://ieeexplore.ieee.org/document/9060340/>. Citado na página 2.

STELLA, M.; RUSSO, M.; ŠARIć, M. Rbf network design for indoor positioning based on wlan and gsm. *International Journal of Circuits, Systems and Signal Processing*, v. 8, p. 116–122, 01 2014. Citado na página 13.

TECHNOLOGIES, U. *H3 Hexagonal hierarchical geospatial indexing system*. 2024. <https://h3geo.org>. Acessado: 04-03-2024. Citado na página 59.

TORRES-SOSPEDRA, J. et al. Comprehensive analysis of distance and similarity measures for Wi-Fi fingerprinting indoor positioning systems. *Expert Systems with Applications*, v. 42, n. 23, p. 9263–9278, dez. 2015. ISSN 09574174. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0957417415005527>. Citado 2 vezes nas páginas 34 e 46.

WU, P. et al. Time Difference of Arrival (TDoA) Localization Combining Weighted Least Squares and Firefly Algorithm. *Sensors*, v. 19, n. 11, p. 2554, jun. 2019. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/19/11/2554>. Citado na página 26.

YANG, L.; YU, Y.; LI, B. Multi-Floor Indoor Localization Based on RBF Network With Initialization, Calibration, and Update. *IEEE Transactions on Wireless Communications*, v. 20, n. 12, p. 7977–7991, dez. 2021. ISSN 1536-1276, 1558-2248. Disponível em: <https://ieeexplore.ieee.org/document/9465729/>. Citado 2 vezes nas páginas 22 e 25.

ZHANG, G. et al. Wireless Indoor Localization Using Convolutional Neural Network and Gaussian Process Regression. *Sensors*, v. 19, n. 11, p. 2508, maio 2019. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/19/11/2508>. Citado 4 vezes nas páginas 1, 23, 25 e 45.