IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Sistema de Localização para ambientes fechados baseado nos sinais de Wif-Fi utilizando Machine Learning

Leonardo Fachetti Jovêncio
Engenharia Eletrônica e de Computação
Universidade Federal do Rio de Janeiro

Trabalho teórico final da disciplina de redes neurais ministrada pelo professor Rogério Pinto Espindola no período de 2024.1.

I. INTRODUÇÃO

O Sistema de Posicionamento Global (GPS) enfrenta grandes limitações em ambientes fechados ou cobertos, como no interior de edifícios, túneis ou locais subterrâneos. Esses ambientes possuem estruturas feitas de materiais como concreto, vidros e metais que obstruem e/ou geram interferências destrutivas em sinais eletromagnéticas, comprometendo a comunicação entre dispositivos no interior desses ambientes e os satélites GPS orbitando ao redor da Terra. Devido a obstrução desses sinais, o cálculo da posição é comprometido, resultando em uma estimação de localização muito imprecisa.

Para lidar com essa limitação, técnicas alternativas têm sido desenvolvidas [1]-[3], como sistemas de posicionamento baseados nos sinais de Wi-Fi, Bluetooth, infravermelho ou ultrassom. Esses sistemas utilizam pontos de acesso instalados ao longo do ambiente fechado para localizar os dispositivos receptores. Através da troca de sinais entre esses pontos de acesso e os dispositivos receptores, é feito um cálculo de estimativa da localização.

Essas soluções são frequentemente utilizadas em sistemas de navegação interna, como em aeroportos, shopping centers, hospitais, fábricas, dentre outros, e oferecem uma alternativa viável para localização precisa em ambientes fechados onde o GPS não é eficaz.

II. OBJETIVO DO TRABALHO

Será desenvolvido um sistema de localização para ambientes fechados utilizando as intensidades dos sinais de Wi-Fi emitidos por diversos roteadores distribuídos ao longo do ambiente. Para isso, as intensidades dos sinais de Wi-Fi captadas por um dispositivo receptor no interior do ambiente serão usadas como dados de entrada para uma rede neural do tipo MLP totalmente conectada com saídas regressivas. A rede irá computar esses dados e predizer a longitude, latitude e altitude que o dispositivo receptor está localizado dentro do ambiente avaliado. A figura 1 ao lado mostra a relação entrada-saída da rede neural a ser implementada.

III. FONTE DOS DADOS

Será utilizada a base de dados *UJIIndoorLoc*, criada por [4] e obtida do repositório da UC Irvine [5]. Essa base contém informações acerca das intensidades do sinal Wi-Fi emitidos por 520 roteadores distintos distribuídos ao longo do interior de três edifícios da Universidade Jaume I, localizado na Espanha, abrangendo uma área total de aproximadamente

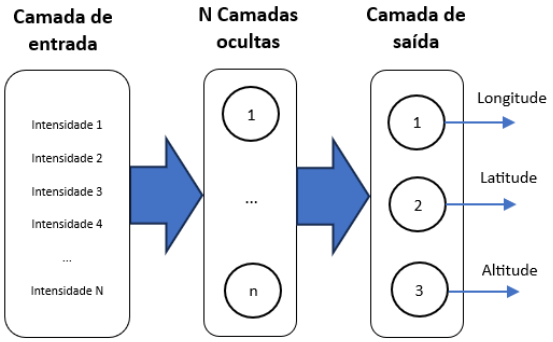

FIGURA 1 – Relação entrada-saída da rede neural MLP a ser implementada. A rede neural terá N entradas referente as intensidades dos sinais de Wi-Fi emitidos por N roteadores distribuídos ao longo do ambiente fechado, e terá 3 saídas regressivas, informando a longitude, latitude e altitude.

110.000 m^2. As leituras dos sinais Wi-Fi foram capturadas utilizando diferentes dispositivos receptores, incluindo smartphones, tablets, dentre outros, enquanto os coletores caminhavam por diferentes regiões dos edifícios. A base de dados possui 21.048 registros e 529 atributos contendo as intensidades das leituras dos sinais de Wi-Fi, além de informações de localização, como latitude, longitude e andar. Não há informações acerca da altitude na base de dados, mas, a partir da informação do andar, o valor da altitude pode ser calculado (cada andar possui aproximadamente 3,5 metros). A tabela 1 abaixo mostra algumas informações acerca da base de dados utilizada neste trabalho.

Coluna	Descrição	Valor mínimo	Valor máximo
WAP001 a WAP520	Intensidade do sinal de Wi-Fi captado do roteador WAPXXX	-104	100
LONGITUDE	Longitude do ponto de medição realizado.	-7695.9	-7299.8
LATITUDE	Latitude do ponto de medição realizado.	4864745.7	4865017.4
FLOOR	Andar do ponto de medição realizado.	0	4
Dimensão: 21.048 x 529			

Tabela 1 – Informações sobre a base de dados *UJIIndoorLoc*.

Existem outros atributos de localização na base de dados escolhida além dos mencionados na tabela 1, no entanto, não serão utilizados neste trabalho. As colunas WAP (Wireless Access Point) guardam as intensidades dos sinais de Wi-Fi captados de um roteador específico do ambiente, sendo que WAP001 guarda a intensidade do roteador 1, WAP002 do roteador 2 e assim se sucessivamente. O valor de intensidade está na escala de dB e assume valores inteiros negativos de -104 dB a 0 dB, informando o nível de atenuação do sinal. Quanto mais próximo do valor -104 dB, menor a intensidade do sinal Wi-Fi captado. Além disso, caso o sinal não seja captado, seu valor será +100 dB (escolha dos autores em [4]).

IV. DESENVOLVIMENTO DO SISTEMA

A. PREPARAÇÃO DOS DADOS

Antes de iniciarmos o desenvolvimento do sistema de localização proposto, devemos avaliar a qualidade da base de dados escolhida, removendo outliers e tratando os valores ausentes. O método *isna ( ).sum ( ),* da biblioteca pandas da linguagem de programação python, retorna à quantidade de **valores ausentes** em cada uma das colunas do objeto DataFrame informado. No entanto, nenhum valor ausente foi encontrado, como pode ser observado no retorno do método mencionado na figura 3 abaixo.

```
2 print (db.isna().sum())

WAP001       0
WAP002       0
WAP003       0
WAP004       0
WAP005       0
            ..
WAP519       0
WAP520       0
LONGITUDE    0
LATITUDE     0
ALTITUDE     0
Length: 523, dtype: int64
```

FIGURA 3 – Retorno da função isna ( ).sum ( ) da biblioteca pandas. Nota-se que há a presença de valores ausentes.

Para avaliar a presença de outliers, traçaremos os gráficos de boxplot. A partir dos gráficos de boxplot da figura 4 abaixo, observa-se que não há **outliers** presentes na base de dados.

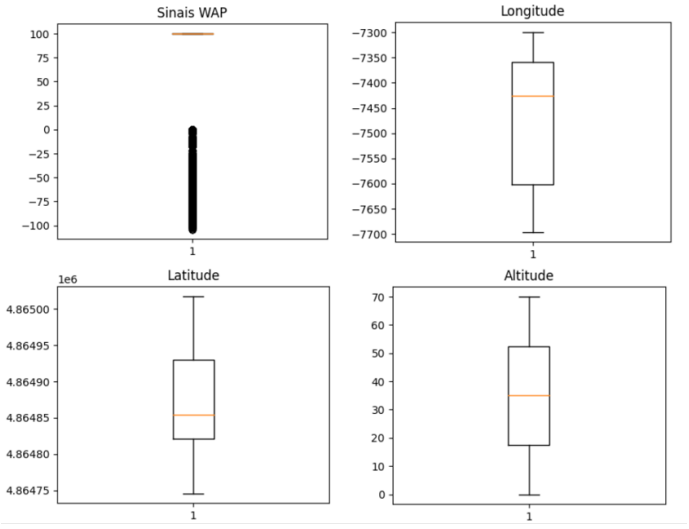

FIGURA 4 – Boxplot contendo as entradas WAP e as informações de longitude, latitude e altitude. Nota-se que há a presença de outliers.

Em relação a **padronização**, foi adotada a padronização valor máximo para as entradas WAP e a padronização gaussiana para as informações de latitude, longitude e altitude. A tabela 2 abaixo mostra os intervalos dos atributos da base de dados escolhida após a padronização adotada.

Coluna	Tipo de padronização adotada	Valor mínimo	Valor máximo
WAP001 a WAP520	Valor máximo	-1.04	1.00
LONGITUDE	Gaussiana	-1.839	1.353
LATITUDE		-1.879	2.147
ALTITUDE		-1.376	1.922

Tabela 2 – Informações sobre a base de dados após as padronizações adotadas. O atributo ALTITUDE foi obtida a partir do atributo FLOOR da base de dados original multiplicando o andar por 3.5 metros.

B. ANÁLISE EXPLORATÓRIA DOS DADOS

A figura 5 a abaixo mostra um histograma com os valores das intensidades WAP das colunas WAP001 a WAP520. Observa-se, a partir do histograma, que na maioria das vezes o sinal de Wi-Fi emitido pelos roteadores não é captado (representado pelo valor WAP = 1). Esse efeito pode fornecer informações valiosas a cerca da localização do dispositivo receptor, já que é esperado que, quanto mais distante o roteador estiver do dispositivo receptor, menor será a intensidade do sinal Wi-Fi captado.

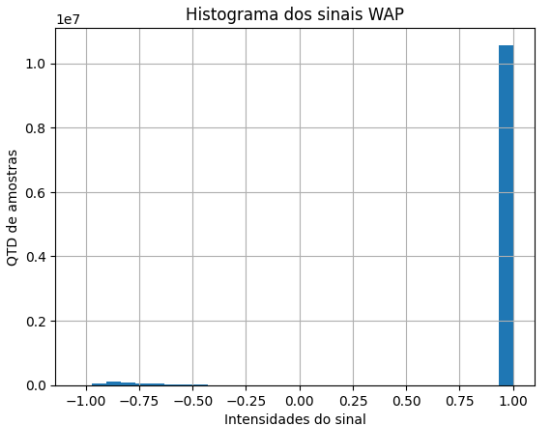

FIGURA 5 – Histograma dos valores das intensidades WAP das colunas WAP001 a WAP520. Nota-se uma quantidade maior de ocorrências do valor 1, casos em o sinal de Wi-Fi não é captado. Nesses cenários, o sinal de Wi-Fi proveniente do roteador WAP em questão naquela localização medida é muito fraco e, por tanto, não é captado pelo dispositivo receptor.

A figura 6 abaixo mostra um histograma com os valores das intensidades WAP das colunas WAP001 a WAP520 eliminando os casos em que o sinal de Wi-Fi não foi captado. Observa-se uma distribuição variada das amostras, sendo que, na maioria dos casos, a intensidade do sinal Wi-Fi é captada com uma intensidade média ou baixa, variando principalmente em torno de -100 dB a -40 dB. Espera-se que a combinação das intensidades dos 520 valores WAP traga padrões fortes de localização acerca do dispositivo receptor, fornecendo detalhes relevantes para que a rede neural consiga aprender os padrões de localização envolvidos no problema.

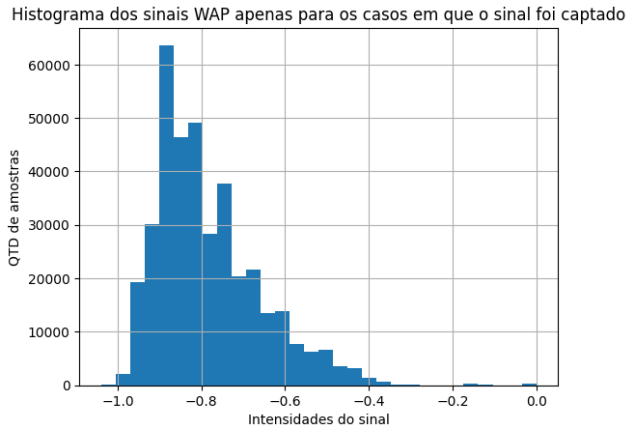

FIGURA 6 – Histograma dos valores das intensidades WAP das colunas WAP001 a WAP520 eliminando os casos em que o sinal não foi captado (WAP = 1). Nota-se uma distribuição variada, mas com uma maior concentração no extremo esquerdo do intervalo.

Como última análise, as figuras 7, 8 e 9 a seguir mostram os histogramas das informações de localização Longitude, Latitude e Altitude.

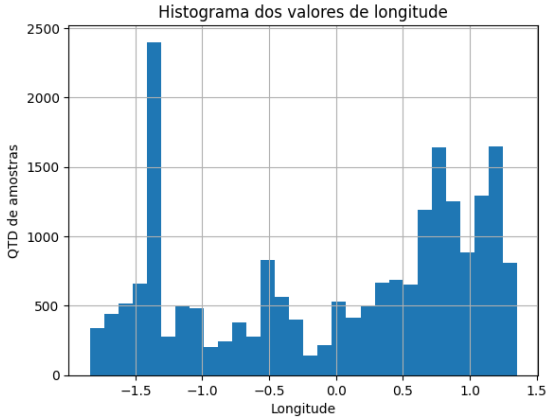

FIGURA 7 – Histograma dos valores de longitude.

A partir do histograma da figura 7 acima, observa-se uma certa variabilidade para os valores de longitude, apresentando maiores ocorrências próximas aos extremos do intervalo.

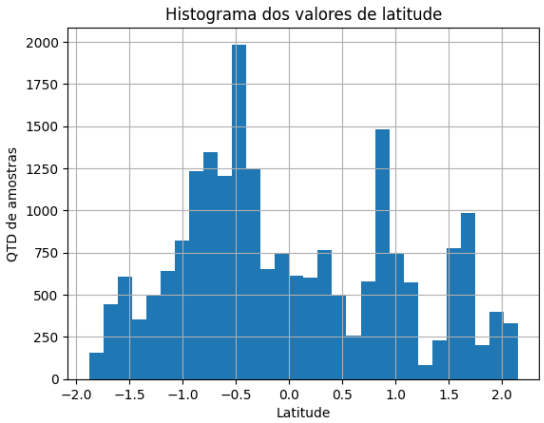

FIGURA 8 – Histograma dos valores de latitude.

A partir do histograma da figura 8 acima, observa-se uma certa variabilidade para os valores de latitude, apresentando maiores ocorrências principalmente na região entre o extremo esquerdo do intervalo e o valor médio.

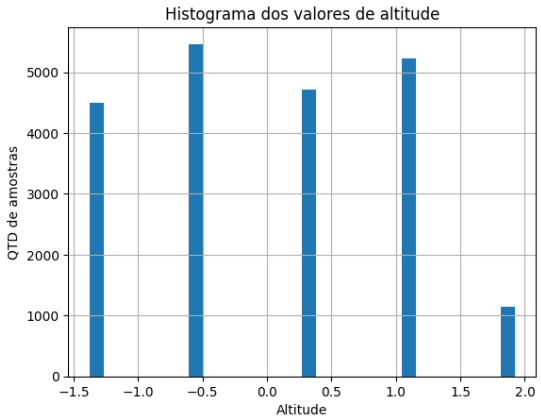

FIGURA 9 – Histograma dos valores de altitude.

Por fim, a partir do histograma da figura 9 acima, observa-se que os valores de altitude assumem valores discretos, apresentando uma baixa variabilidade.

C. MODELAGEM DA REDE

Será implementada uma rede neural MLP totalmente conectada com saídas regressivas. Para isso, a rede terá 520 entradas (WAP001 a WAP520), referente as intensidades dos sinais de Wi-Fi emitido pelos 520 roteadores distribuídos ao longo do ambiente, e 3 saídas regressivas, informando os atributos de localização Latitude, Longitude e Altitude, nessa ordem. A figura 10 abaixo mostra um diagrama da rede neural a ser implementada.

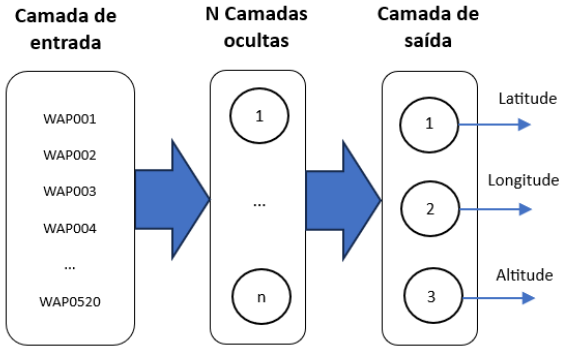

FIGURA 10 – Diagrama da rede neural a ser implementada. A rede terá 520 neurônios na camada de entrada e 3 neurônios na camada de saída, sendo que a saída será regressiva.

Foi feito uma exploração de hiperparâmetros, variando o número de camadas ocultas e a quantidade de neurônios em cada camada oculta. Ao todo, foram avaliadas 6 arquiteturas distintas, sendo elas mostradas na tabela 3 abaixo.

Arquiteturas	Camadas ocultas		
	1º	2º	3º
Arquitetura 1	128	-	-
Arquitetura 2	256	-	-
Arquitetura 3	128	64	-
Arquitetura 4	256	128	-
Arquitetura 5	128	64	32
Arquitetura 6	256	128	64

Tabela 3 – Diferentes arquiteturas de redes neurais avaliadas. Ao todo, 6 arquiteturas foram avaliadas, variando entre 1 a 3 camadas, além do número de neurônios em cada camada oculta.

A rede neural foi implementada utilizando a biblioteca torch, da linguagem de programação python. Essa biblioteca contem diversas funcionalidades úteis para modelagem, treinamento e avaliação de redes neurais. A tabela 4 abaixo mostra alguns hiperparâmetros utilizados na modelagem da rede neural proposta.

Hiperparâmetro	Descrição
Taxa de aprendizado	0.001
Função custo	MSE (Mean Squared Error)
Otimizador	ADAM (Adaptive Moment Estimation)
Função de ativação das camadas ocultas	ReLU
Função de ativação da camada de saída	Nenhuma
Número de épocas	300

Tabela 4 – Hiperparâmetros utilizados na modelagem da rede neural. Esses hiperparâmetros foram os mesmos para as 6 arquiteturas exploradas (descritas na tabela 3).

O otimizador ADAM utiliza diversas boas práticas recomendadas no processo de aprendizagem, como ajuste dinâmico da taxa de aprendizado, uso do termo momento no processo de ajuste dos pesos sinápticos, regularização L2, dentre outros, sendo um dos otimizadores mais utilizados atualmente. A função custo MSE foi escolhida por ser a mais adequada para problemas de regressão. A imagem 11 abaixo mostra a implementação da rede neural da arquitetura 6 proposta utilizando na linguagem de programação python.

```python
# Configurando alguns hiperparâmetros da rede
learning_rate = 0.001
num_epochs = 300

# Estrutura da rede neural
class NeuralNetwork(nn.Module):
    def __init__(self):
        super(NeuralNetwork, self).__init__()
        self.layer1 = nn.Linear(520, 256)  # 1 camada oculta
        self.layer2 = nn.Linear(256, 128)  # 2 camada oculta
        self.layer3 = nn.Linear(128, 64)   # 3 camada oculta
        self.output = nn.Linear(64, 3)     # Camada de saída

    def forward(self, x):
        x = torch.relu(self.layer1(x))  # ReLU na 1 camada oculta
        x = torch.relu(self.layer2(x))  # ReLU na 2 camada oculta
        x = torch.relu(self.layer3(x))  # ReLU na 3 camada oculta
        x = self.output(x)              # Sem função de ativação
        return x

# Inicializando o modelo, a função de perda e o otimizador
model = NeuralNetwork()
criterion = nn.MSELoss()  # Mean Squared Error para regressão
optimizer = optim.Adam(model.parameters(), learning_rate) # Adam
```

FIGURA 11 – Implementação da rede neural da arquitetura 6 proposta utilizando a linguagem de programação python e a biblioteca Torch.

D. TREINAMENTO DA REDE

A base de dados foi dividida em treinamento, validação e teste, conforme descrito na tabela 5 a seguir. Além disso, afim de evitar a ocorrência do overfitting e melhorar a capacidade de generalização da rede, foi adotado um critério de parada antecipada do aprendizado, de modo a interromper o treinamento da rede assim que o seu desempenho comece a piorar no conjunto de validação. Para isso, foi estabelecido um intervalo de 40 épocas de espera, isto é, caso não houver melhoria no aprendizado no intervalo de 40 épocas sucessivas, o aprendizado será interrompido.

Conjunto	Porcentagem	Quantidade de amostras
Treinamento	70%	14733
Validação	15%	3157
Teste	15%	3158
Total	**100%**	**21048**

Tabela 5 – Porcentagem e quantidade de amostras dos conjuntos de treinamento, validação e teste adotado neste trabalho.

No encerramento do processo de aprendizado, é extraído o melhor modelo obtido, isto é, aquele que apresentou o menor erro para o conjunto de validação. Esse erro é calculado utilizando a função custo MSE, que realiza o cálculo apresentado em (1).

$$erro_{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left[\left(\hat{y}_{1_i} - y_{1_i}\right)^2 + \left(\hat{y}_{2_i} - y_{2_i}\right)^2 + \left(\hat{y}_{3_i} - y_{3_i}\right)^2\right] \quad (1)$$

Da equação (1) acima, $\hat{y}_1$, $\hat{y}_2$ e $\hat{y}_3$ representam as 3 saídas preditas pela rede, enquanto que y_1 , y_2 e y_3 representam a saída real presente no conjunto de validação. Além disso, N é a quantidade total de amostras do conjunto de validação e i uma amostra específica desse conjunto. A tabela 6 abaixo mostra algumas informações acerca do processo de aprendizado para cada uma das 6 arquiteturas proposta.

Arquiteturas	Época da parada antecipada	Época do melhor modelo	Tempo de treinamento (minutos)
Arquitetura 1	185	144	5.6
Arquitetura 2	281	240	18.5
Arquitetura 3	217	176	8.8
Arquitetura 4	237	196	18.7
Arquitetura 5	264	223	12.6
Arquitetura 6	138	97	8.0

Tabela 6 – Informações acerca do processo de aprendizado para cada uma das 6 arquiteturas propostas.

As imagens 12 a 17 a seguir mostram a evolução do erro da rede no conjunto de treino e validação durante o processo de aprendizado, bem como a época da parada antecipada.

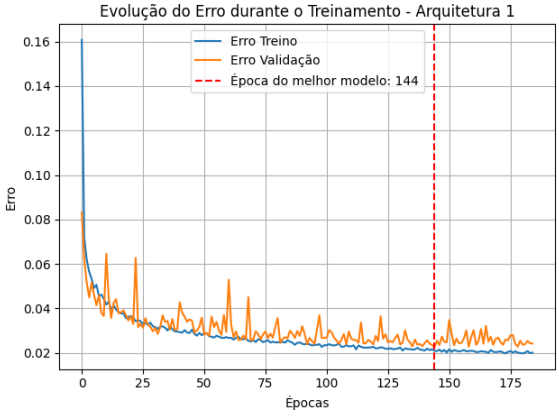

FIGURA 12 – Evolução do erro da rede no conjunto de treino e validação durante o processo de aprendizado para a arquitetura 1 proposta.

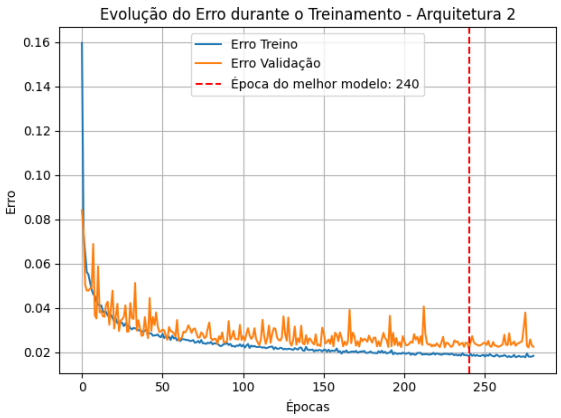

FIGURA 13 – Evolução do erro da rede no conjunto de treino e validação durante o processo de aprendizado para a arquitetura 2 proposta.

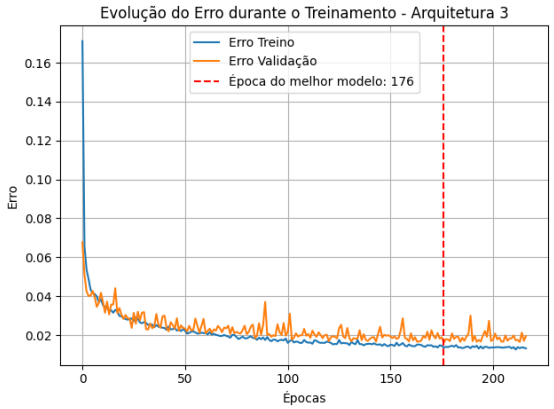

FIGURA 14 – Evolução do erro da rede no conjunto de treino e validação durante o processo de aprendizado para a arquitetura 3 proposta.

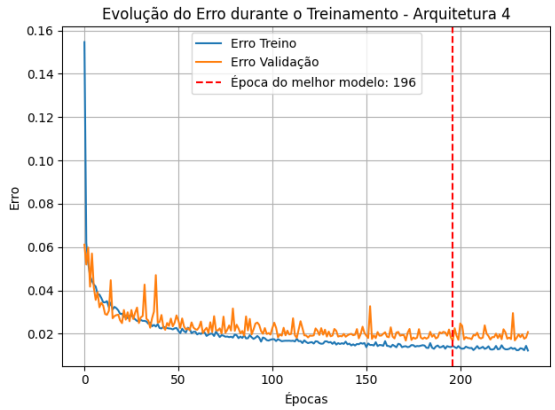

FIGURA 15 – Evolução do erro da rede no conjunto de treino e validação durante o processo de aprendizado para a arquitetura 4 proposta.

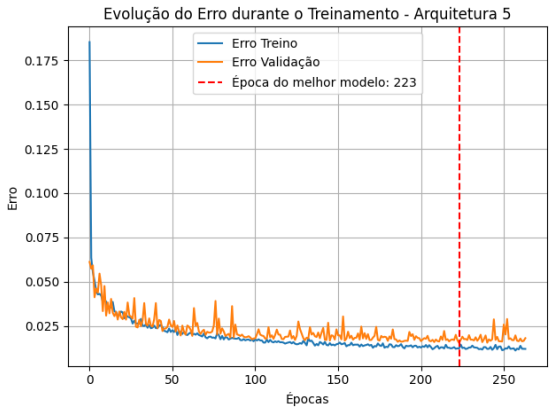

FIGURA 16 – Evolução do erro da rede no conjunto de treino e validação durante o processo de aprendizado para a arquitetura 5 proposta.

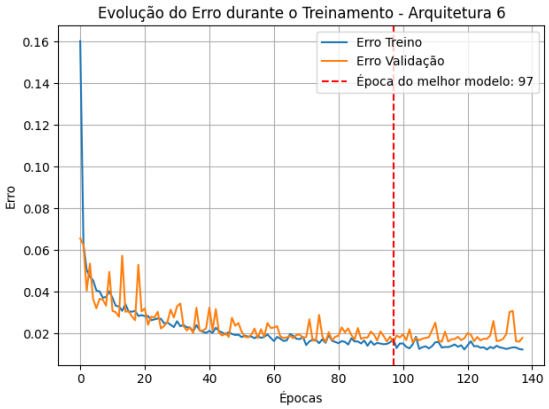

FIGURA 17 – Evolução do erro da rede no conjunto de treino e validação durante o processo de aprendizado para a arquitetura 6 proposta.

D. AVALIAÇÃO DA REDE

A tabela 7 abaixo mostra o desempenho de cada uma das arquiteturas de rede proposta avaliada no conjunto de teste. Para o cálculo do erro individual das saídas Latitude, Longitude e Altitude, foi utilizado a função custo MSE descrita em (2) abaixo no qual foi considerado apenas a saída avaliada, sendo $\hat{y}$ a saída predita pela rede e y a saída real presente no conjunto de teste. Além disso, para o cálculo do erro da rede, foi utilizado a função custo MSE considerando as 3 saídas preditas pela rede, conforme descrito na equação (1) anteriormente.

Arquiteturas	Erro médio da rede			
	Saída Latitude	Saída Longitude	Saída Altitude	Erro da rede
Arquitetura 1	0.01603	0.00900	0.04605	0.02369
Arquitetura 2	0.01596	0.00851	0.04302	0.02250
Arquitetura 3	0.01355	0.00736	0.03109	0.01733
Arquitetura 4	0.01315	0.00740	0.03166	0.01740
Arquitetura 5	0.01347	0.00720	0.03173	0.01747
Arquitetura 6	0.01310	0.00714	0.02932	0.01652

Tabela 7 – Erros da rede avaliada no conjunto de teste para cada uma das 6 arquiteturas proposta. Nota-se que a rede com o melhor desempenho foi a de arquitetura 6, com 3 camadas ocultas e 256, 128 e 64 neurônios por camada oculta, respectivamente.

$$erro_{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}_{1_i} - y_{1_i}\right)^2 \quad (2)$$

A partir da tabela 7 acima, observa-se que a rede de arquitetura 6 apresentou o menor erro médio, sendo este de 0.01652. A tabela 8 abaixo mostra o erro percentual das saídas dessa rede, sendo esse erro calculado a partir da expressão descrita em (3).

Dados	Latitude	Longitude	Altitude
Padronizados	0.33%	0.22%	0.89%

Tabela 8 – Erros percentuais para cada uma das saídas da rede de arquitetura 6. Esse erro foi calculando usando a expressão descrita em (3).

$$erro_{\%} = \frac{erro_{médio_{saída}}}{|valor_{saída_{max}} - valor_{saída_{min}}|} * 100 \quad (3)$$

VI. CONCLUSÕES

Os resultados obtidos neste trabalho se mostraram promissores e confirmam a viabilidade da solução proposta. A rede da arquitetura 6 apresentou baixos erros percentuais nas estimativas de latitude, longitude e altitude, com erros de 0.33%, 0.22% e 0.89%, respectivamente. A confirmação da viabilidade da solução proposta abre portas para futuras aplicações e melhorias na metodologia, permitindo evoluções contínuas e adaptações conforme o necessário.

O trabalho aqui proposto não teve o objetivo de contribuir com avanços científicos relevantes a cerca da problemática abordada, mas sim permitir um aprendizado a cerca do tema redes neurais. Nesse sentido, o trabalho desenvolvido pouco foi guiado por artigos científicos e o estudo foi conduzido de forma independente e de autoria própria.

Bibliografia

[1] - N. Singh, S. Choe and R. Punmiya, "Machine Learning Based Indoor Localization Using Wi-Fi RSSI Fingerprints: An Overview," in IEEE Access, vol. 9, pp. 127150-127174, 2021, doi: 10.1109/ACCESS.2021.3111083.

[2] - Qin, Feng & Zuo, Tao & Wang, Xing. (2021). CCpos: WiFi Fingerprint Indoor Positioning System Based on CDAE-CNN. Sensors. 21. 1114. 10.3390/s21041114.

[3] - M. Ibrahim, M. Torki and M. ElNainay, "CNN based Indoor Localization using RSS Time-Series," 2018 IEEE Symposium on Computers and Communications (ISCC), Natal, Brazil, 2018, pp. 01044-01049, doi: 10.1109/ISCC.2018.8538530.

[4] - J. Torres Sospedra et al., "UJIIndoorLoc: A new multi-building and multi-floor database for WLAN fingerprint-based indoor localization problems," 2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN), Busan, Korea (South), 2014, pp. 261-270, doi: 10.1109/IPIN.2014.7275492.

[5] – UJIIndoorLoc, UC Irvine, 2014. Disponível em https://archive.ics.uci.edu/dataset/310/ujiindoorloc. Acesso em: 07 de jun. de 2024.