UNIVERSIDADE FEDERAL DO AMAZONAS - UFAM

INSTITUTO DE COMPUTAÇÃO - ICOMP

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Um Modelo Baseado em Aprendizagem de Máquina para Apoiar o Aumento da Autonomia das Baterias de Smartphones

Rômulo José Pereira da Costa Júnior

Manaus - AM

Julho - 2025

Rômulo José Pereira da Costa Júnior

Um Modelo Baseado em Aprendizagem de Máquina para Apoiar o Aumento da Autonomia das Baterias de Smartphones

Monografia de Graduação apresentada ao Instituto de Computação da Universidade Federal do Amazonas como requisito parcial para a obtenção de título de bacharel em Ciência da Computação.

Orientador

Prof. Dr. Raimundo da Silva Barreto

Manaus - AM

Julho - 2025

Um Modelo Baseado em Aprendizagem de Máquina para Apoiar o Aumento da Autonomia das Baterias de Smartphones

Autor: Rômulo José Pereira da Costa Júnior

Orientador: Prof. Dr. Raimundo da Silva Barreto

RESUMO

Machine Learning (ML) ou aprendizagem de máquina, é o estudo de algoritmos e modelos estatísticos que os computadores usam para realizar tarefas específicas sem serem explicitamente programados para isso, incluindo realizar previsões sobre o estado de carga (State Of Charge - SoC), que está relacionado com a quantidade de carga realmente disponível em uma bateria. Atualmente, a demanda energética dos aplicativos aumenta, contudo, as capacidades das baterias ficaram estagnadas por muito tempo e não conseguem acompanhar essa necessidade. Então, até o momento, muitas abordagens foram propostas para resolver o problema, incluindo promover uma medição precisa do SoC. Nesse cenário, com o intuito de auxiliar na resolução dos problemas energéticos em smartphones, a partir de boas previsões da bateria dos dispositivos, esta monografia consiste no planejamento e criação de um modelo de ML para estimar o SoC a partir do nível da bateria. Para realizar essa tarefa, uma Deep Neural Network (DNN) foi projetada e testada em três cenários de previsão do nível da bateria (10, 30 e 60 minutos). Três versões do modelo utilizando 3, 4, 5, 6 e 7 camadas ocultas foram experimentadas. Os resultados demonstraram que o desempenho do modelo foi ótimo no cenário de 10 minutos (até 96,83% de precisão), mas reduziu conforme o tempo de previsão aumentava (78,94% de precisão no pior caso).

Palavras-chave: aprendizagem de máquina; redes neurais profundas; state of charge; eficiência energética.

Lista de Figuras

Lista de Tabelas

Sumário

1 Introdução

Segundo [MAHESH 2019], Machine Learning (ML) ou aprendizagem de máquina, é o estudo de algoritmos e modelos estatísticos que os computadores usam para realizar tarefas específicas sem serem explicitamente programados para isso. Ou seja, atualmente o objetivo de diversas indústrias é normalmente ensinar as máquinas a aprender e lidar com dados, ou extrair informações. O ML depende de vários algoritmos para alcançar seu objetivo, e o uso de cada algoritmo depende do tipo de problema a ser resolvido, número de variáveis, etc [MAHESH 2019]. Nesse contexto, diversas subáreas do ML que estudam e usam algoritmos específicos foram criadas, como o Deep Learning (DL), Reinforcement Learning (RL), entre outros.

Na área de dispositivos móveis, o State Of Charge (SoC) ou estado da bateria está relacionado com a quantidade de carga realmente disponível em uma determinada bateria [GARCHER et al. 2009]. E, pode-se afirmar atualmente que a estimação precisa do SoC é essencial para gerenciar e otimizar o desempenho da bateria dos dispositivos [WU et al. 2025]. Nesse contexto, técnicas de previsão do SoC através do consumo de energia ou bateria dos dispositivos e que utilizam ML passaram a ser desenvolvidas e testadas, com resultados efetivos. O rápido avanço do DL também criou novas oportunidades para a estimação do SoC, pois suas técnicas possuem capacidades excelentes de ajuste não linear [WU et al. 2025]. Essas habilidades permitem que o DL possa automaticamente aprender representações de features, com resultados fortes em SoC [WU et al. 2025].

1.1 Contexto e Problemática

A cada dia que passa, aplicativos cada vez mais ricos e complexos são desenvolvidos, muitos deles demandando computação intensa e transmissão de dados em alta velocidade [SHAIKH et al. 2019]. Consequentemente, a demanda energética aumentou significativamente. Entretanto, as capacidades das baterias ficaram estagnadas por muito tempo e não conseguem acompanhar as necessidades modernas [SHAIKH et al. 2019]. Além disso, na atualidade, não existe um entendimento claro sobre como resolver problemas energéticos na engenharia de softwares [ULLAH et al. 2022].

Até o momento, muitas abordagens foram propostas para resolver o problema, como analisar e reduzir o consumo de energia de um hardware específico e restringir o

uso de recursos desnecessários [LI et al. 2018]. Infelizmente, grande parte dessas técnicas visam estender a vida da bateria ao custo de comprometer a experiência do usuário [LI et al. 2018].

Porém, uma medição precisa do SoC a partir do nível da bateria, consumo de energia ou similares é uma tática que vem ganhando bastante destaque, pois, pode aumentar a eficiência da bateria, prolongar a vida útil e reduzir os potenciais riscos de sobrecarga ou descarga excessiva [LI et al. 2018]. No entanto, [LI et al. 2018] observaram que até mesmo as alternativas que utilizam essa tática possuem limitações que as fazem ser primitivas e pouco práticas: (i) ausência de dados refinados, em virtude da origem dos dados da maioria dos estudos ser laboratorial, os quais não representam uso real dos smartphones; (ii) e os próprios dados, em razão dos conjuntos de dados coletados serem compostos por features superficiais e com granularidade grosseira (por exemplo, coletas a cada 1 minuto).

Além disso, foi realizado um mapeamento sistemático (detalhado na Seção 2) sobre o estado da arte da predição do consumo e nível da bateria dos smartphones. Esse mapeamento permitiu que novas lacunas e déficits fossem observados: (iii) alta concentração de uso do ML, com poucas tentativas em DL ou outras subáreas; (iv) falta de padronização quanto os critérios e métricas de avaliação dos modelos, e unidades de medida de algumas métricas; (v) omissão e pouca clareza de detalhes sobre o conjunto de dados e coleta; (vi) tamanho dos conjuntos de dados relativamente pequenos.

1.2 Objetivos

O objetivo desta monografia é criar um modelo preditivo baseado em DL, mais especificamente em redes neurais profundas, para predizer o nível da bateria de smartphones no futuro após certo tempo de uso. Esse modelo também busca superar as principais limitações citadas identificadas em estudos anteriores. Para tanto, os objetivos específicos são:

- Gerar um conjunto de dados que supere os problemas (i), (ii), (v) e (vi), ou seja, que contenha dados de uso reais em smartphones, apresente baixa granularidade na coleta, seja o mais claro quanto às informações dos dados, e possua um tamanho consideravelmente superior aos utilizados nos estudos encontrados;

- Modelar e implementar um modelo de DL baseado em redes neurais profundas que

utilize o menor número de features possível, a fim de minimizar a complexidade e o custo temporal do treinamento; e

- Avaliar o desempenho do modelo a partir das métricas apropriadas, aplicadas em um estudo de caso.

1.3 Organização do Trabalho

O restante do trabalho está organizado da seguinte forma. Na Seção 2, o referencial teórico usado nesta monografia é explicado. Na Seção 3, o mapeamento sistemático que foi realizado para esse estudo é detalhado, incluindo as conclusões tomadas. Na Seção 4, é realizado o detalhamento dos trabalhos correlatos retirados a partir do mapeamento. A Seção 5 detalha o método proposto, abordando o estudo e o planejamento dos dados, bem como o desenvolvimento do modelo de previsão. A Seção 6 apresenta os resultados obtidos e discute suas implicações. Na Seção 7, as considerações finais são feitas.

2 Referencial Teórico

2.1 Deep Neural Networks

Segundo [ABABEI e MOGHADDAM 2018], uma Deep Neural Network (DNN), ou Rede Neural Profunda, é estruturalmente um Percepton Multicamadas (ou Multilayer Percepton) (MLP), que simplificadamente é uma Feedforward Neural Network com várias camadas ocultas. Nessas redes, a informação flui na direção da camada de entrada para a saída. A possibilidade de contar com várias camadas ocultas é a principal diferença entre as DNN e as Neural Networks comuns, o que permite que as DNN capturem padrões de relacionamento não lineares [ABABEI e MOGHADDAM 2018] através das funções de ativação divididas em camadas na rede, o que é particularmente efetivo para a previsão do SoC [WU et al. 2025].

O processo de aprendizagem em MLPs consiste, de forma geral, na minimização do erro entre as saídas previstas pelo modelo e os valores reais [POPESCU et al. 2009]. Para alcançar esse objetivo, primeiramente os dados passam pelas camadas da rede, gerando uma previsão. Então, essa previsão é comparada com o valor real para gerar o erro do modelo. Por fim, o erro é repassado para as camadas através do ajuste dos pesos dos neurônios e das camadas de entrada e saída [POPESCU et al. 2009].

2.2 Random Forest

Random Forest (RF), ou Floresta Aleatória, é um algoritmo de ML baseado em *ensemble*, o que significa que ele consiste na agregação dos resultados de um conjunto mais simples de estimadores. Esse algoritmo é construído a partir de árvores de decisão, que são uma forma simples de realizar classificações a partir de perguntas com resposta binária [VANDERPLAS 2016].

Cada árvore é treinada com uma amostra aleatória dos dados e considera apenas um subconjunto das features. Após o treinamento, todas as árvores votam juntas para a previsão final, que é eleita pelo voto da maioria [BALADRAM 2024]. Uma boa RF possui nós que dividem os dados em dois grupos usando um valor de corte em cada uma das features utilizadas. Nesse contexto, é essencial definir as perguntas que devem ser feitas [VANDERPLAS 2016]. As principais vantagens do algoritmo incluem a flexibilidade do modelo, aliada à rapidez no treinamento e na execução de previsões [VANDERPLAS 2016]. A Figura 1 [BALADRAM 2024], apresentada a seguir, ilustra o funcionamento básico do algoritmo RF.

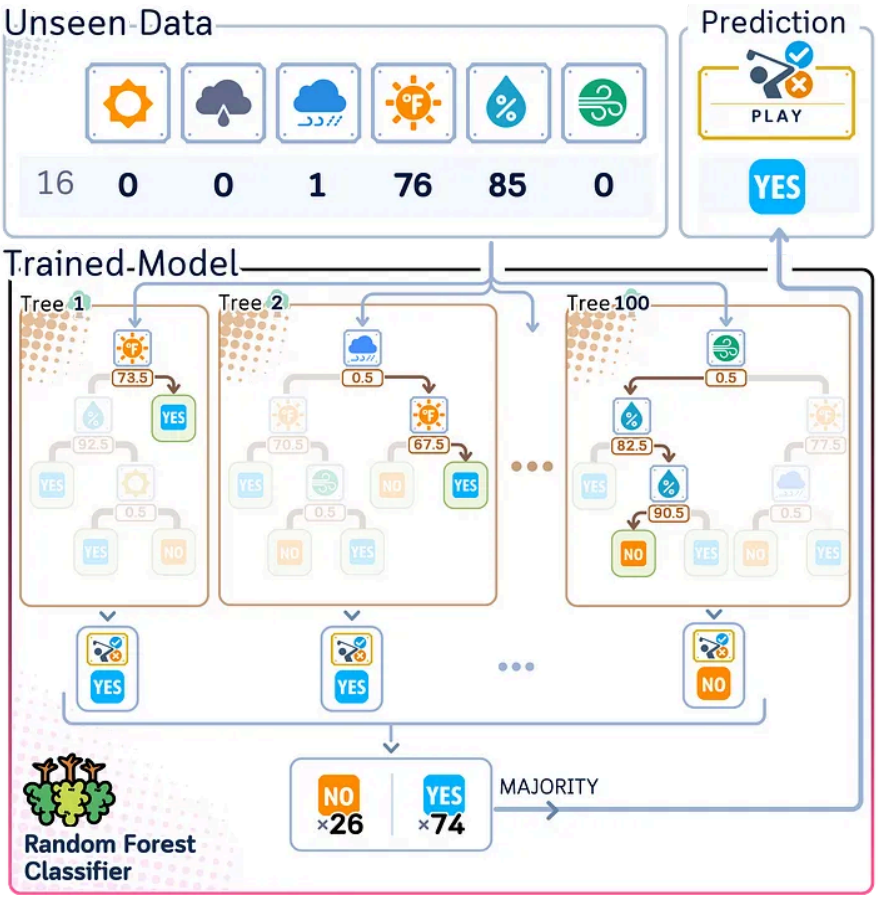

Figura 1 – Funcionamento de uma Random Forest [BALADRAM 2024]

2.3 One Hot Encoding

Em muitos casos, é necessário representar atributos categóricos de forma apropriada para evitar que o modelo interprete erroneamente relações numéricas entre valores que, na realidade, são apenas categorias [GOOGLE 2024a]. Por exemplo, tratar uma variável "codigo_postal" (CEP) como numérica fará com que o modelo acredite que o CEP 20002 seja duas vezes maior que o CEP 10001. Além disso, é necessário fazer a conversão de variáveis em texto para numérico, uma vez que a maioria dos modelos não podem ser treinados com dados em formato textual [GOOGLE 2024a].

Quando uma variável possui baixa dimensionalidade, ou seja, um baixo número de categorias, a abordagem geralmente utilizada é a *one hot encoding*. Neste método, primeiramente, as categorias são convertidas para números de índice, que são como um identificador das categorias [GOOGLE 2025a], veja a Figura 2.

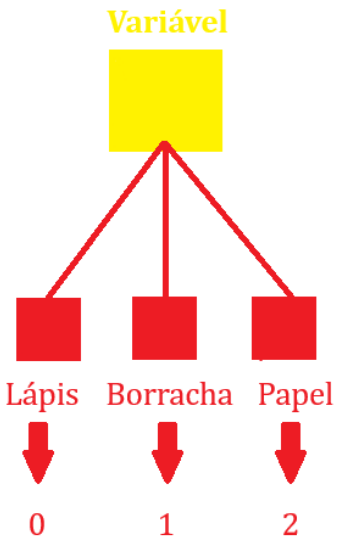

Figura 2 – Exemplo de criação de números de índice
Fonte: o autor

Em seguida, esses índices são convertidos em vetores binários de dimensão N, sendo N o número total de categorias da variável. Cada vetor possui exatamente um elemento com valor 1 (indicando a categoria correspondente) e todos os demais elementos com valor 0. Esse vetor, denominado vetor one-hot, é então utilizado como entrada para o modelo de aprendizado de máquina.

2.4 Categorical Embedding

A técnica *one hot encoding* é viável quando a variável categórica possui baixa dimensionalidade, dado que um grande número de categorias gera alguns problemas nessa representação: a alta dimensionalidade de categorias acarreta em um grande vetor *one hot*, o que implica em um grande número de pesos. Consequentemente, a quantidade de dados, computação e memória necessários para o treinamento aumentarão [GOOGLE 2024b]. Nesse cenário, a técnica *embedding*, ou *categorical embedding*, pode solucionar esses problemas.

Uma *embedding* é uma representação vetorial dos dados em um espaço vetorial, em que cada componente do vetor representa alguma informação. Um modelo encontra uma *embedding* projetando o espaço de alta dimensão dos vetores de dados iniciais em um espaço de baixa dimensão, porém esse processo muitas vezes não é interpretável para humanos [GOOGLE 2025b].

Há muitas formas de se capturar as características importantes em um espaço de alta dimensionalidade em um espaço de baixa dimensionalidade, uma delas é utilizando redes neurais [GOOGLE 2025c]. Nessa abordagem, é criada a camada de *embedding*, que é uma camada oculta responsável por aprender gradualmente a representação vetorial dos dados categóricos. A Figura 4 mostra um exemplo de uso de *embedding* em uma rede neural para recomendação de comidas, cuja entrada do modelo é uma codificação *one-hot*, que é processada pela camada de embedding.

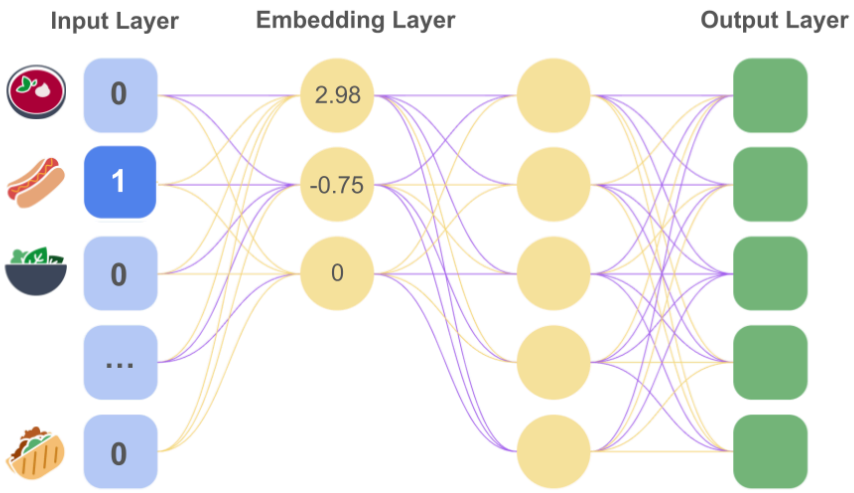

Figura 3 – Exemplo de rede neural com uma camada de *embedding* [GOOGLE 2025c]

2.5 Métricas de Avaliação de ML

O número de aplicações baseadas em ML cresceu significativamente nas últimas décadas, o que impulsionou o desenvolvimento de novos modelos. Nesse cenário, torna-se essencial avaliar e comparar o desempenho dos modelos propostos com os já existentes, tanto para identificar soluções com desempenho insuficiente quanto para selecionar aquelas com maior potencial de otimização [RAINIO et al. 2024].

Em problemas de regressão, nos quais o objetivo é prever valores flutuantes ao invés de categorias, existem várias formas de se avaliar o desempenho do modelo. Uma dessas formas são as métricas baseadas em erro, que mensuram a distância da previsão para o valor real [RAINIO et al. 2024]. Dois exemplos comuns altamente utilizados são o erro médio ao quadrado (MSE) e o erro médio absoluto (MAE) [RAINIO et al. 2024]. O MSE é utilizado para quantificar quanto a equação de regressão aprendida pelo modelo se ajusta às variáveis conhecidas, bem como mensurar quanto os valores previstos diferem dos valores reais [HANDELMAN et al. 2019]. Essa métrica é calculada pela fórmula:

$$\textbf{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y - yi)^2$$

Onde “y” é o valor real e “yi” o valor previsto. Como os dados são elevados ao quadrado, o MSE não possui a mesma unidade dos dados, o que dificulta sua interpretação. Portanto, a raiz do erro médio ao quadrado (RMSE) pode ser utilizada para contornar esse problema. Sua fórmula é:

$$\textbf{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y - yi)^2}$$

Já o MAE, por sua vez, é uma variação do MSE que calcula a média das diferenças absolutas dos dos erros pela fórmula:

$$\textbf{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y - yi|$$

Por não elevar os erros ao quadrado, o MAE é menos sensível a valores extremos e mantém a mesma unidade das variáveis previstas.

3 Mapeamento Sistemático

Devido à importância e abrangência do tema, foi realizado um mapeamento sistemático a fim de traçar um mapa sobre o estado da arte do tópico e evidenciar as lacunas ou déficits existentes nos trabalhos anteriores e corrigi-los. Esta seção detalha a construção e os resultados.

3.1 Questões de Pesquisa

Dado o objetivo do mapeamento sistemático, a pergunta de pesquisa geral formulada foi: *"De que forma o Machine Learning foi usado para mensurar/prever o nível ou consumo da bateria dos smartphones?".* Para responder a essa questão de forma mais abrangente, ela foi subdividida em:

- (SQ1) *"Quais técnicas foram utilizadas para mensurar/prever o nível ou consumo da bateria dos smartphones?"*
- (SQ2) *"Quais técnicas foram utilizadas para mensurar/prever o nível ou consumo da bateria de íons de lítio, usada nos smartphones?"*
- (SQ3) *"Qual tem sido o desempenho das técnicas utilizadas para mensurar/prever o nível ou consumo da bateria dos smartphones, ou da bateria de íons de lítio?"*
- (SQ4) *"Qual é o tamanho dos dados usados nos modelos e de que forma esses dados são coletados?*

3.2 Critérios de Inclusão e Exclusão

Os critérios de inclusão e exclusão para os estudos definidos foram:

Código	Tipo do critério	Descrição
CI01	**inclusão**	O estudo contém uma aplicação de machine learning para prever/medir o **consumo ou nível da bateria** de Smartphones.
CI02	**inclusão**	O estudo contém uma aplicação de machine learning para prever/medir o **consumo ou nível da bateria** de baterias de Íon de Lítio.
CI03	**inclusão**	O estudo contém informações que podem ser usadas como referências complementares.
CE01	**exclusão**	O estudo **não tenta prever/medir** o consumo ou nível da

		bateria dos Smartphones.
CE02	**exclusão**	A técnica usada no estudo **não é relacionada** à machine learning.
CE03	**exclusão**	Os resultados do estudo **não se aplicam** a smartphones.
CE04	**exclusão**	O estudo **não está disponível** na íntegra.
CE05	**exclusão**	O estudo **não está escrito** em português ou inglês.

3.3 Busca Por Estudos

Inicialmente, foi feita uma busca manual por estudos do tópico, com o intuito de entender os principais termos a serem utilizados e selecionar alguns estudos-chave para validação da string final.

A busca pelos estudos foi realizada no portal de periódicos da CAPES, IEE Xplore, ACM Digital Library e Google Scholar. A string de busca utilizada foi: *"("battery consumption" OR "energy consumption" OR "battery prediction" OR "State-of-Charge" OR "State of Charge" OR "SoC") AND ("smartphone" OR "smart phone") AND ("machine learning" OR "ml")"*. No filtro 1, 26 artigos foram selecionados, dos quais 17 foram incluídos após o filtro 2. Após o último filtro, 12 artigos foram englobados (9 aplicações de ML e 3 estudos de suporte iniciais para a escritura do documento).

3.4 Respostas das Questões de Pesquisa

3.4.1 Subquestão de Pesquisa 1

- **(SQ1)** "Quais técnicas foram utilizadas para *mensurar/prever o nível ou consumo da bateria dos Smartphones?*"

Dos estudos analisados, 11,1% utilizaram técnicas de DL, mais especificamente Deep Neural Network (DNN) e Recurrent Neural Network (RNN) para resolver o problema. Os 88,9% restantes utilizaram técnicas clássicas de ML, sendo comum o uso e comparação de mais de uma técnica por estudo. A partir da Figura 4, foi identificado que as principais técnicas utilizadas são: Random Forest (RF) (20%), Linear Regression (12%), e Neural Network/ Support Vector Machine (SVM)/ Long Short-Term Memory (LSTM)/ Gradient Boosted Regression Tree (GBRT)/ AdaBoosted Decision Tree (ADT) (8%).

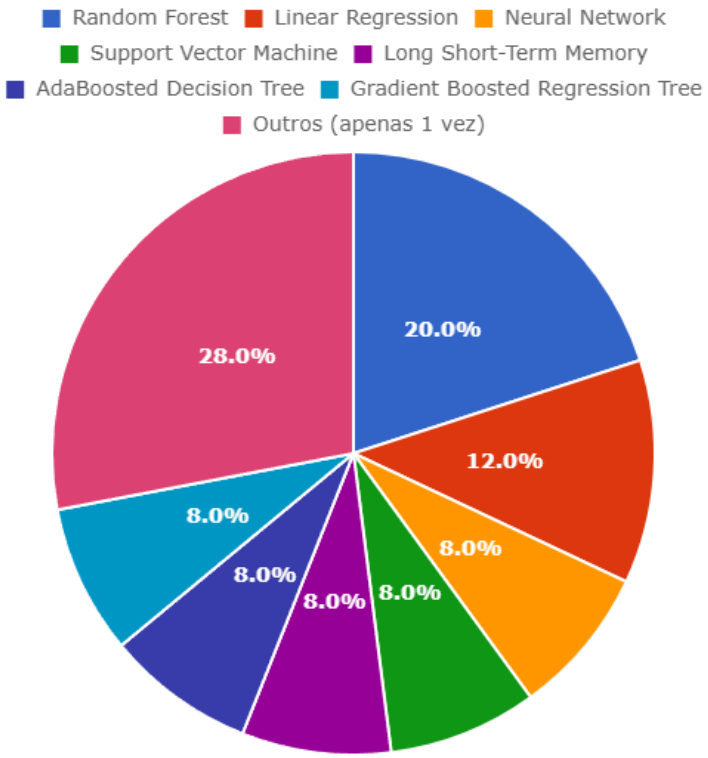

Figura 4 – Técnicas de ML utilizadas nos estudos incluídos no mapeamento sistemático

3.4.2 Subquestão de Pesquisa 2

- **(SQ2)** *"Quais técnicas foram utilizadas para mensurar/ prever o nível ou consumo da bateria de íons de lítio, usada nos Smartphones?"*

Não foram encontrados estudos que tentaram mensurar/ prever o nível ou consumo da bateria de íons de lítio explícitamente.

3.4.3 Subquestão de Pesquisa 3

- **(SQ3)** *"Qual tem sido o desempenho das técnicas utilizadas para mensurar/prever o nível ou consumo da bateria dos smartphones, ou da bateria de íons de lítio?"*

Devido ao uso de diferentes critérios e métricas de avaliação adotados pelos estudos analisados, não é possível quantificar de forma objetiva quais técnicas apresentaram melhor desempenho de forma geral. Além disso, os estudos possuem diferentes alvos de previsão, o que faz com que certos critérios de avaliação tenham diferentes unidades de medida. Contudo, como comentado anteriormente, muitos estudos fizeram comparações internas entre múltiplas técnicas. A partir dessas comparações, é possível apontar as

técnicas com melhores desempenhos dentro de cada estudo, que foram: RF (3 vezes), LSTM (2 vezes), Extreme Gradient Boosting (XGBoost) (1 vez), Neural Network (1 vez), Support Vector Regressor (SVR) (1 vez), AdaBoosted Decision Tree (ADT) (1 vez) e SVM (1 vez). Esses resultados indicam um relativo equilíbrio entre as técnicas avaliadas, sugerindo que não há uma técnica definitiva para os diversos cenários de previsão realizados.

3.4.4 Subquestão de Pesquisa 4

- **(SQ4)** *"Qual é o tamanho dos dados usados nos modelos e de que forma eles são coletados?"*

Em grande parte dos estudos analisados, informações essenciais sobre os dados utilizados não foram disponibilizadas explicitamente. No entanto, entre aqueles que apresentaram essas informações de forma clara, observou-se que o tamanho dos conjuntos de dados era relativamente limitado: 62,5% utilizaram 1400 observações ou menos nos modelos. E mesmo entre os 37,5% restantes, o maior conjunto possui menos de 50 mil observações. Essas informações revelam uma lacuna de modelos treinados e testados com grandes conjuntos de dados.

Em relação ao intervalo de coleta utilizado, verificou-se uma grande variação entre os estudos. Porém, o preocupante é que apenas 42,8% realizaram coletas com intervalo de 1 minuto ou menos, e destes, apenas 14,2% utilizaram coletas a cada segundo. Por outro lado, 57,2% aderiram intervalos maiores que variam de 5 a 20 minutos. Esse cenário evidencia um déficit de estudos com baixa granularidade na coleta.

3.5 Conclusões do Mapeamento Sistemático

O mapeamento sistemático realizado possibilitou a identificação de algumas lacunas e déficits na subárea de previsão do nível ou consumo da bateria dos Smartphones. Além das duas lacunas anteriores observadas por [LI et al. 2018] na Seção 1.1, as demais são: (iii) escassez de estudos que utilizam técnicas de DL para resolver o problema; (iv) falta de padronização quanto os critérios e métricas de avaliação dos modelos, e unidades de medida de algumas métricas; (v) omissão ou pouca clareza em detalhes sobre o conjunto de dados e coleta; (vi) uso de conjuntos de dados de tamanhos relativamente pequenos.

Nesse cenário, o modelo de DL criado para esta monografia foi concebido tendo em vista tais questões, de modo a não perpetuar os problemas identificados.

4 Trabalhos Correlatos

Se tratando de prever o nível ou consumo de bateria dos Smartphones, algumas tentativas já foram feitas. [LI et al. 2018] propuseram uma abordagem baseada em quatro técnicas clássicas de ML para prever a duração da bateria até um nível crítico: Linear Regression, Random Forest Regression, Gradient Boosted Regression Tree (GBRT) e Extreme Gradient Boosting (XGBoost). Os autores dividiram as features em quatro categorias e as utilizaram em conjunto para as previsões, com intervalos de coleta variando entre 5 segundos e 1 minuto. Entretanto, como não aplicaram técnicas de seleção de features, enfrentaram dificuldades em definir o conjunto ideal, o que pode ter impactado a precisão do modelo. O melhor resultado foi alcançado com o XGBoost, que obteve um Root Mean Square Error (RMSE) de 1 hora e 46 minutos (106 minutos).

Flores-Martín et al. (2024) utilizam DL, especificamente uma Deep Neural Network (DNN) e uma Recurrent Neural Network (RNN), para prever o nível da bateria. Os dados foram coletados a cada 20 minutos por um aplicativo desenvolvido pelos autores para coletar as features necessárias. Foram realizados treinamentos, testes e validações sobre dados de 5 dispositivos com diferentes padrões de uso e tempo de coleta. Os resultados obtidos foram semelhantes para ambas as redes, evidenciando uma tendência de degradação nas métricas de desempenho à medida que o tamanho do conjunto de dados aumentava. Por exemplo, em um dispositivo com 120 linhas coletadas, a acurácia e Mean Square Error (MSE) foram 94,74% e 16,31 (DNN), respectivamente. Já um dispositivo com 1600 linhas, as métricas foram 60% e 196,11 (DNN).

Chantrapornchai e Nusawat (2016) desenvolveram dois modelos de ML para prever o descarregamento da bateria de smartphones: Neural Networks e SVM, com algumas variações. Com o auxílio de aplicações externas, os autores coletaram conjuntos de dados contendo valores de features pré-definidos. Esses conjuntos de dados foram coletados a cada 5 minutos em um período de 1 hora, totalizando 320 a 480 linhas de três categorias de aplicação. Como melhores resultados, foram encontrados valores muito baixos de Mean Absolute Error (MAE) e RMSE utilizando redes neurais, como 0,0007 e 0,001, respectivamente. Além disso, os autores obtiveram bons resultados utilizando SVM, como 0,0005 e 0,0019 de MAE e RMSE.

Chen et al. (2015) desenvolveram um método de previsão de energia que analisa o consumo de aplicações com respeito às chamadas de sistema, por meio de uma Neural Network com lógica Fuzzy. Foram realizados testes limitados, porém em cenários reais. O

resultado foi uma diferença máxima de 1,2% entre os valores reais e previstos.

Mehrotra et al. (2021) construíram uma abordagem de classificação multiclasse baseada em aprendizado supervisionado para prever o consumo energético dos Smartphones, categorizando-os em três níveis distintos: baixo, médio e alto. A metodologia adotada possui duas etapas principais: inicialmente, os aplicativos foram agrupados utilizando o algoritmo K-Nearest Neighbor (KNN) para análise preliminar do impacto energético das diferentes categorias; em seguida, foram implementadas e comparadas cinco técnicas de ML: Naive Bayes, KNN, árvore de decisão, RF e J48. Em qualquer cenário, todas as técnicas eram avaliadas de forma comparativa, e aquela que apresentasse o melhor desempenho era escolhida. Os resultados demonstraram que a técnica RF foi superior às demais, com 97% de acurácia, 96,7% de precisão e 0,967 de F1-score.

Neto et al. (2021) desenvolveram modelos para estimativa do consumo energético em dispositivos móveis mediante um processo automatizado capaz de analisar e preparar dados para geração de modelos personalizados. O estudo abrangeu a construção de 30 modelos utilizando quatro técnicas distintas: Multilayer Perceptron (MLP), SVR, RF e LSTM. Para a seleção do conjunto ótimo de features dentre as 30 inicialmente monitoradas, os pesquisadores empregaram adicionalmente as técnicas RF e SVR. A coleta de dados foi realizada em dois dispositivos distintos, com durações de 47.728 e 32.063 segundos, respectivamente. A avaliação dos modelos considerou três cenários em múltiplos dispostivos: (i) utilização de todas as features disponíveis; (ii) uso exclusivo de features numéricas; e (iii) aplicação apenas das features numéricas selecionadas via técnicas de feature selection. Os resultados mostraram diferenças significativas entre os dispositivos e cenários. No dispositivo 1, a técnica LSTM obteve os melhores desempenhos nos cenários (i) e (ii), com erros médios absolutos (MAE) de 158,5 mW e 188,9 mW, respectivamente, enquanto no cenário (iii) a RF se destacou com um MAE de 429,9 mW. Já no dispositivo 2, o SVR apresentou os melhores resultados em todos os cenários, com MAEs de 268,0 mW, 276,0 mW e 256,0 mW, respectivamente.

Qian et al. (2019) apresentam alguns modelos de predição do nível da bateria de smartphones, a fim de entender o comportamento dos usuários e o consumo de energia. A abordagem consiste em selecionar eventos relacionados à energia, extrair as features de entrada desses eventos para os modelos, e então aplicar uma série de modelos de ML: Linear Regression, ADT, GBRT e RF. O processo de coleta de dados estendeu-se por 80 dias, com particularidades na frequência de aquisição: enquanto algumas features eram

registradas imediatamente, métricas como o nível da bateria eram armazenadas em intervalos de 5 minutos. Para a avaliação dos modelos, os autores selecionaram 6 cenários de uso dos dispositivos e aplicaram as técnicas. No treinamento, as técnicas ADT e RF obtiveram os melhores resultados, com RMSE mínimo de 0,2153 e 0,3966, respectivamente. Enquanto nos testes, apesar do desempenho geral inferior das técnicas, a RF foi a melhor em 5 dos 6 cenários, com RMSE mínimo de 0,5624.

Ullah et al. (2022) testaram a arquitetura de rede neural LSTM para prever a energia que seria consumida pelas aplicações no smartphone. Nos testes, diferentes intervalos de coleta que variam de 15 segundos a 4 minutos foram utilizados. O modelo contou com uma rede neural simples e poucas features de entrada, os melhores resultados foram um MAE mínimo de 21,8, além de 83% e 84% de acurácia no treinamento e teste, respectivamente.

Por fim, Gaska et al. (2018) criaram uma abordagem altamente escalável ao desenvolver um sistema que utiliza ML para estimar a eficiência energética de aplicativos usando apenas informações textuais e meta-informações disponíveis na Google Play Store. As bases de treino e teste do modelo consistem em 5 features coletadas de 1400 aplicativos. Foram testadas 3 técnicas não-lineares (SVM não linear, ADT e RF) e 2 lineares (SVM linear e Linear Regression). O resultado foi que o MSE geral das técnicas totalizou 0,625 (de uma classificação 1-5), porém, com um desempenho relativamente superior das técnicas lineares: 0,927 utilizando o SVM linear e 0,940 utilizando a Linear Regression. As métricas indicam que a performance foi relativamente boa, com o SVM linear possuindo erros de classificação próximos de 0. Além disso, ficou evidente que o modelo colocou mais importância nas features que impactam negativamente na vida útil da bateria do que as impactam positivamente.

5 Método Proposto

Neste estudo, a tarefa de predizer o nível da bateria de smartphones foi realizada por uma DNN baseada no comportamento do usuário. Esse comportamento foi representado por um conjunto de características (ou *features*) coletadas a partir de um aplicativo chamado Tucandeira [SWPERFI 2024], desenvolvido em uma parceria UFAM-Motorola no projeto SWPERFI pelo subgrupo IS (*Intelligent Software*). O objetivo do Tucandeira é ajudar a detectar que informações do smartphone podem influenciar diretamente no desempenho e contribuir com o desenvolvimento de soluções para melhorar a experiência do usuário

[SWPERFI 2024]. O aplicativo funciona como um coletor ininterrupto e silencioso de uma série de informações do smartphone a cada segundo, como a bateria, conexão com a internet, aplicações em primeiro e segundo plano, entre outras funcionalidades [SWPERFI 2024].

A metodologia foi dividida em duas etapas principais: estudo e planejamento dos dados; e desenvolvimento do modelo. Essas etapas serão detalhadas a seguir.

5.1 Estudo e Planejamento dos Dados

A primeira etapa de estudo e planejamento dos dados foi dividida em três sub-etapas: desenvolvimento do estudo de caso; análise exploratória dos dados; e preparação dos dados.

Na primeira etapa, uma experimentação prática do modelo foi planejada com o intuito de avaliar o mesmo em um cenário real de uso de um smartphone. Em seguida, uma análise exploratória de dados foi realizada, a fim de entender o comportamento dos dados, bem como constatar ou não a presença de problemas na coleta. Por fim, foi feito um trabalho de análise das correlações entre as features selecionadas e a aplicação da técnica RF para verificar a importância de cada feature. O objetivo por trás dessa etapa era encontrar um balanço entre a minimização da correlação entre as features, para minimizar o tempo de treinamento, e a maximização das suas importâncias, a fim de maximizar o desempenho pelo uso correto das características relevantes.

5.1.1 Experimentação Prática

O objetivo da experimentação é demonstrar o desempenho do modelo na previsão do nível de descarga da bateria em um cenário real de uso para um perfil de uso do usuário. Esse estudo de caso foi desenvolvido tendo em vista os problemas comentados na Seção 1.1. A Tabela 1 resume as diretrizes da experimentação.

Tabela 1 – Resumo das diretrizes da experimentação do Modelo

ID	Diretriz
D1	Os dados coletados serão sobre um cenário de uso real, portanto, não serão impostas restrições sobre o uso do dispositivo.

D2	As features deverão ser escolhidas com embasamento teórico.
D3	A granularidade da coleta será de no mínimo 1 segundo.
D4	As métricas de avaliação do modelo foram escolhidas com base nas principais utilizadas nos trabalhos correlatos.
D5	Após o término da coleta, serão fornecidos detalhes sobre os dados.
D6	O tamanho da coleta deverá ser maior que o observado nos trabalhos correlatos.

A diretriz D1 tem como objetivo garantir a qualidade dos dados coletados, evitando que apresentem características artificiais ou laboratoriais. Já a diretriz D2 busca assegurar que as *features* utilizadas sejam relevantes e baseadas em evidências, evitando abordagens superficiais ou arbitrárias, como observado em [LI et al. 2018]; para isso, foram aplicadas técnicas de seleção de atributos, conforme detalhado na Seção 5.1.3. A diretriz D3 visa superar a granularidade grosseira comentada por [LI et al. 2018] e observada nos trabalhos correlatos. Nesse contexto, o aplicativo escolhido foi o Tucandeira, pois o mesmo pode garantir que a coleta seja confiável e realizada a cada 1 segundo. Quanto à diretriz D4, as métricas de avaliação escolhidas foram o MAE, RMSE e precisão, devido a relevância dessas métricas no contexto de regressão [RAINIO et al. 2024], bem como uma grande frequência de uso observada nos estudos analisados. A diretriz D5 tem como foco evitar a omissão ou falta de clareza sobre o conjunto de dados, então, o detalhamento do conjunto de dados utilizado nesta monografia será feito na Seção 5.1.2. Finalmente, a diretriz D6 visa superar a tendência de uso de um conjunto de dados pequeno no modelo. Para isso, os dados foram coletados ao longo de aproximadamente quatro dias de uso contínuo, resultando em um conjunto de dados significativamente maior do que os analisados na literatura.

5.1.2 Análise Exploratória de Dados

A análise exploratória foi feita sobre o conjunto de dados que é composto por coletas a cada 1 segundo de um dispositivo: o Samsung Galaxy A54 5G. O total de dados coletados foi 348635 segundos (aproximadamente 97 horas de coleta ininterrupta), esses dados são específicos para um usuário. Inicialmente, uma busca por valores ausentes foi aplicada, mas nenhuma ausência foi identificada. Adicionalmente, uma análise sobre as lacunas na coleta de dados indicou que apenas 143 (0,04%) segundos de lacuna ocorreram, destes o valor máximo foi 27 segundos.

Visando superar o problema (v), referente à omissão e pouca clareza de detalhes sobre o conjunto de dados, observado na maioria dos estudos, o comportamento do usuário foi analisado. Após a finalização da etapa de preparação dos dados detalhada na Seção 5.1.3 e a escolha do conjunto final de características ter sido finalizado, foi constatado que:

- O usuário utilizou diretamente em 7,38% do tempo apps baixados por ele (WhatsApp, entre outros), e os 92,62% foram utilizados aplicativos do sistema;
- Em uma escala inteira de 0 à 255, a média do nível de brilho utilizado foi 55. Além disso, pela Figura 5 é possível notar que o uso do nível de brilho pelo usuário possui distribuição aproximadamente simétrica, e que o segundo (86) e terceiro (137) quartis são valores intermediários, ou seja, em 75% do tempo o usuário utilizou níveis baixos ou intermediários de brilho.

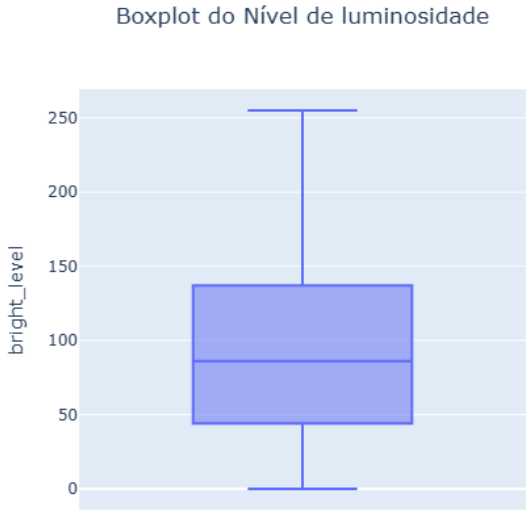

Figura 5 – Boxplot do nível de brilho utilizado pelo usuário

- Em um intervalo inteiro de 0 à 15, o nível de som do dispositivo foi em média 3, e a sua distribuição é assimétrica para os menores valores. Ou seja, o nível do som utilizado pelo usuário foi consideravelmente baixo.
- Durante o período de coleta, o usuário passou em média 253 segundos com a tela ligada ininterruptamente. Adicionalmente, o usuário passou 60% do tempo da coleta com a tela desligada.

- O usuário não reiniciou o dispositivo nenhuma vez durante a coleta, mas vale destacar que no início da coleta o dispositivo já estava há aproximadamente 346 horas ligado de forma incessante.

5.1.3 Preparação dos Dados

Para encontrar o conjunto de features que balanceia a correlação e importância, primeiramente foi realizada uma filtragem manual de características que foram julgadas com alta probabilidade de serem irrelevantes para o problema. Por exemplo, o operador de rede (Vivo, Tim, etc), disponibilidade da impressão digital, entre outras. Em seguida, um modelo que utiliza a técnica RF foi treinado baseado no nível da bateria para mensurar a importância das features restantes. Além disso, a correlação entre essas características foi também contabilizado. Com todas as correlações entre duas features e as importâncias individuais calculadas, foram gerados todos os conjuntos com 5 features possíveis (valor experimental escolhido), e a correlação média entre todas as features do conjunto e o somatório das importâncias foi calculado. Na Figura 3, todos os conjuntos gerados estão visíveis.

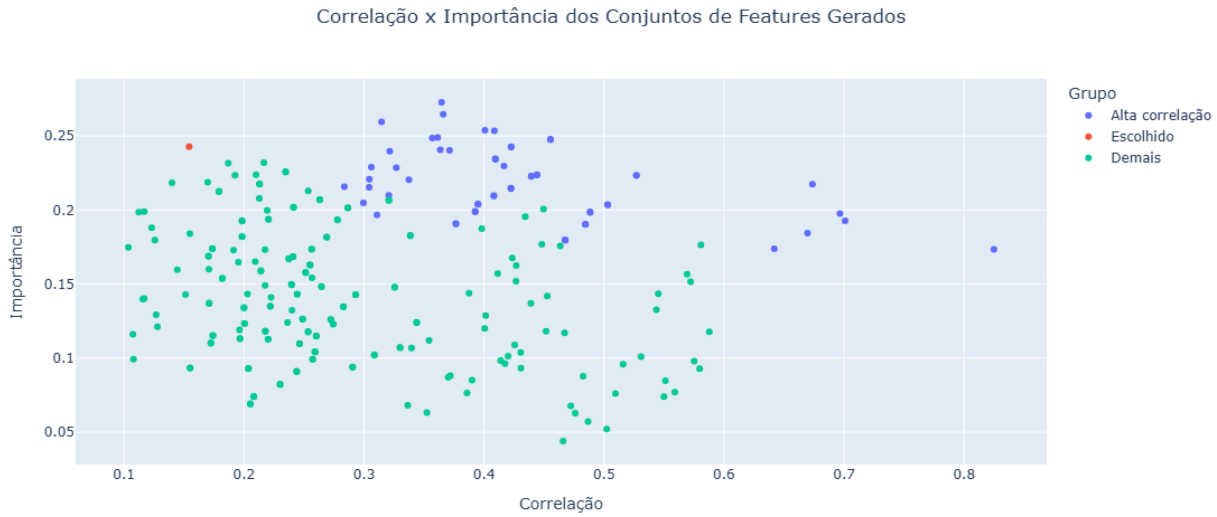

Figura 6 – Conjuntos de Features Gerados

Durante a análise dos subconjuntos gerados, observou-se que embora alguns apresentassem correlação média razoável, certas features possuíam alta correlação entre si (indicado por pontos azuis na Figura 3). Nesse cenário, o conjunto foi desconsiderado da escolha final. Então, além da feature correspondente ao nível da bateria, o conjunto de características do smartphone escolhido (indicado pelo ponto vermelho na Figura 3) é composto pelo: aplicativo que está sendo executado em primeiro plano, nível de brilho,

nível de som, tempo em segundos que o dispositivo está ligado sem ser reiniciado e o tempo em segundos que a tela está ligada ininterruptamente.

A feature que representa o estado da bateria (carregando, descarregando, não carregando, completamente carregado) foi incluída posteriormente, pois, como não foram impostas restrições na coleta, o modelo não conseguiu diferenciar se o dispositivo estava carregando ou descarregando durante o treinamento do modelo. Portanto, foram utilizadas sete features no total.

5.2 Desenvolvimento do Modelo

No desenvolvimento do modelo, a conversão das variáveis categóricas foi a primeira etapa realizada. Das sete features utilizadas, apenas duas eram categóricas: o aplicativo que está sendo executado em primeiro plano (*foreground_app*), e a feature que representa o estado da bateria (*battery_charging_status*). Para a feature *foreground_app*, optou-se pela técnica *categorical embedding*, uma vez que a quantidade de aplicativos do dispositivo é consideravelmente alta. Então, uma rede neural simples foi construída e utilizada para aprender e retornar as *embeddings* dessa categoria. Já para a característica *battery_charging_status*, que possui apenas quatro valores possíveis, a técnica *one hot encoding* foi suficiente. Nesse caso, quatro novas colunas sempre serão geradas como conversão da feature.

Posteriormente, foi necessário armazenar em uma coluna nova no conjunto de dados os valores reais da bateria após um intervalo de tempo definido, com o intuito dessa coluna ser utilizada como a feature alvo da previsão. Como o objetivo desta monografia é criar um modelo inicial capaz de realizar previsões da bateria, as previsões realizadas nas etapas de treinamento, teste, e validação eram realizadas a cada linha do conjunto de dados (que equivale a 1 segundo). Todavia, em um cenário real, o mais adequado seria sempre adicionar um intervalo de tempo entre as previsões.

Além disso, foi feita uma divisão de dados com 80% para treinamento, 10% para testes e 10% para validação. Vale destacar que os dados de validação foram separados do treinamento e testes, a fim de proporcionar uma maior confiabilidade sobre a avaliação dos desempenhos do modelo.

Após as etapas anteriores, o modelo foi de fato construído. Uma rede neural profunda *feedforward* foi utilizada para realizar a previsão do nível da bateria. O modelo foi treinado, testado e validado para prever a bateria em três cenários: após 10, 30, e 60 minutos. O intuito de utilizar mais de um cenário é analisar como o desempenho do

modelo se comportaria com o aumento do tempo de previsão.

Além disso, foi avaliado se o modelo conseguiria se adaptar melhor ao padrão complexo de descarga do dispositivo por meio da adição de camadas ocultas. Contudo, não há quase nada na literatura que afirma exatamente quantas camadas ocultas devem ser utilizadas [THOMAS et al. 2017]. Consequentemente, diversas quantidades de camadas e neurônios são testadas na prática. Porém, nos estudos analisados por [UZAIR e JAMIL 2020], foi constatado que o uso de menos de 3 camadas ocultas resultou em uma perda de desempenho. Então, por esses motivos, as quantidades de camadas ocultas testadas foram 3, 4, 5, 6 e 7. Todas as versões do modelo foram construídas utilizando a arquitetura de funil, e as quantidades de neurônios eram potências de dois até o valor 16. A taxa de aprendizado utilizada foi fixada em 0,001. A Figura 7 ilustra um exemplo de rede com três camadas ocultas, estruturadas em arquitetura de funil, ou seja, a quantidade de neurônios reduz a cada camada.

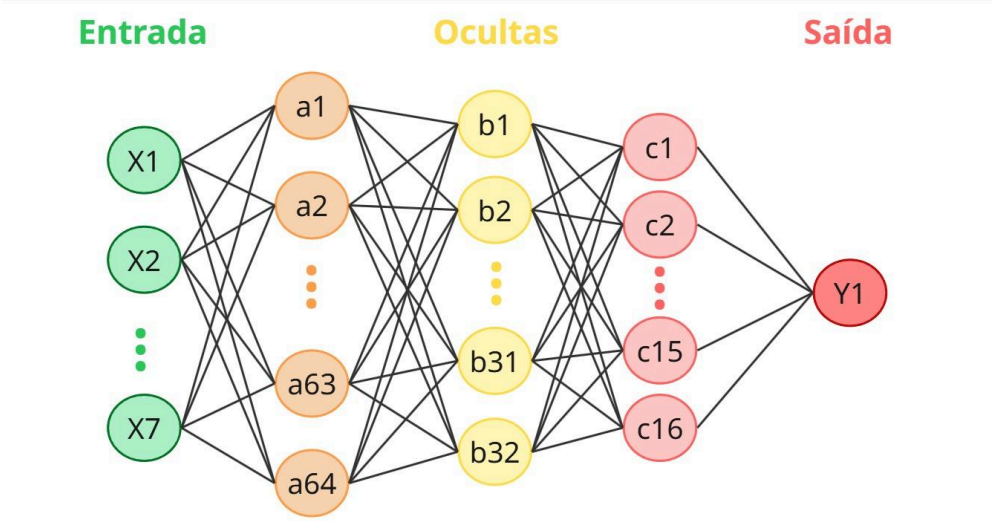

Figura 7 – Esquema da primeira rede neural testada
(autoria própria)

6 Resultados

A Seção 6.1 mostra os resultados das seis versões do modelo, com 3, 4, 5, 6 e 7 camadas ocultas, testadas nos cenários de previsão de 10, 30 e 60 minutos. Em seguida, uma discussão dos resultados é apresentada na Seção 6.2.

6.1 Resultados da Experimentação

A Tabela 2 traz os resultados dos modelos para o cenário de 10 minutos:

Tabela 2 – Resultado da validação do modelo para previsões de 10 minutos

	Métrica		
Nro. de Camadas	**MAE**	**RMSE**	**Precisão**
3	0,68	2,02	95,01%
4	0,60	1,96	**96,83%**
5	0,60	1,96	95,35%
6	**0,55**	**1,93**	96,56%
7	0,66	1,96	95,02%

A Tabela 2 indica que a versão do modelo com 6 camadas possui o menor MAE (0,55) e RMSE (1,93), enquanto a versão com 4 camadas possui a melhor precisão (96,83%). Todavia, ao realizar uma comparação entre a versão com 3 camadas e as demais versões, constatou-se que as métricas diferiram da melhor versão em média aproximadamente: 0,08 no MAE; 0,07 no RMSE; e 0,9% na precisão. Além disso, os valores do MAE das versões ficou na casa dos 0,6, o que é um bom resultado. Por outro lado, o RMSE, que penaliza previsões *outliers*, apresentou valores em torno de 2,0 nas três versões. Essa métrica indica que houve uma certa parcela relativa de previsões ruins. Diante desses indícios, é possível concluir que o desempenho de todas as versões não foi significativamente diferente, e que todas as versões obtiveram um bom desempenho geral. Nesse cenário, os resultados indicam que mesmo o modelo com 3 camadas é suficiente para realizar previsões de 10 minutos, e que não há diferenças significativas em aumentar o número de camadas ocultas.

A Tabela 3 traz os resultados dos modelos para o cenário de 30 minutos:

Tabela 3 – Resultado da validação do modelo para previsões de 30 minutos

	Métrica		
Nro. de Camadas	**MAE**	**RMSE**	**Precisão**
3	1,23	2,91	83,30%

4	1,19	2,89	84,75%
5	1,23	2,85	84,79%
6	1,07	2,98	88,40%
7	**0,94**	**2,75**	**91,85%**

A Tabela 3 evidencia que, no cenário de previsão de 30 minutos, não houve diferenças tão acentuadas entre as versões com 3, 4 e 5 camadas. Contudo, a versão com **7 camadas** se mostrou ser razoavelmente superior às demais: MAE de 0,94, RMSE de 2,75, e 91,85% de precisão. Esses valores superaram as outras versões em pelo menos 0,13 (MAE), 0,23 (RMSE), e **3,45%** (Precisão). Adicionalmente, podemos observar a partir da Figura 8 que quando triplicamos o tempo de previsão em relação ao cenário de 10 minutos, as consequências foram as seguintes:

- O MAE de todas as versões praticamente dobrou, com exceção da versão de 7 camadas. O aumento médio foi de 0,51;
- O RMSE das versões aumentou em média 0,86;
- A precisão de todas as versões caiu em média 6,78 , sendo que essa redução reduz quanto maior for o número de camadas;

Esses resultados sugerem que, à medida que a tarefa de previsão se torna mais complexa (devido ao aumento do horizonte temporal), há uma redução geral no desempenho dos modelos. Contudo, modelos com maior profundidade demonstram maior capacidade de adaptação, indicando que o aumento do número de camadas pode, de fato, trazer ganhos significativos em cenários mais complicados.

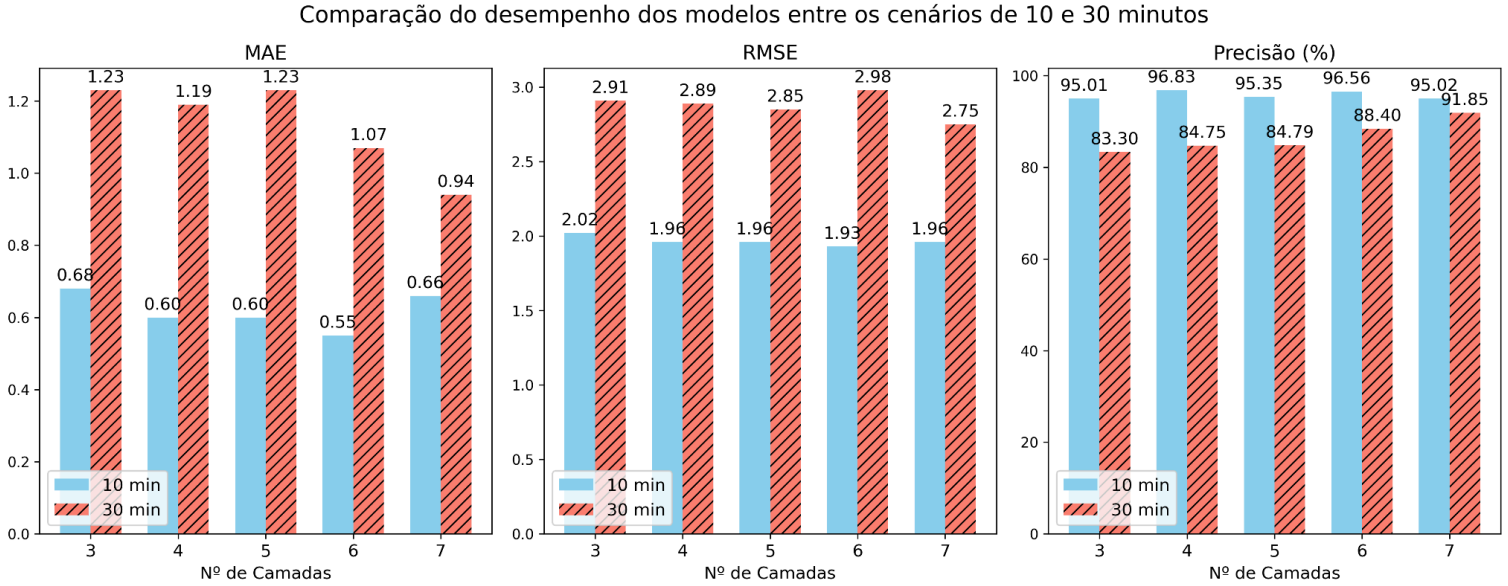

Figura 8 – Gráfico de barras comparativo entre os resultados das métricas para previsões de 10 e 30 minutos.

A Tabela 4 traz os resultados dos modelos para o cenário de 60 minutos:

Tabela 4 – Resultado da validação do modelo para previsões de 60 minutos

	Métrica		
Nro. de Camadas	**MAE**	**RMSE**	**Precisão**
3	1,22	2,46	78,94%
4	1,20	3,05	85,64%
5	0,88	2,54	90,28%
6	**0,77**	**2,06**	**91,10%**
7	1,10	2,43	85,00%

A Tabela 4 aponta que a versão com **6 camadas** é a melhor entre todas, com uma boa vantagem. Essa versão possui desempenho superior em pelo menos 0,11 de MAE, 0,48 de RMSE, e 0,82% de precisão. E, esse resultado fica ainda mais evidente quando a melhor versão é comparada com o desempenho médio das demais versões: o MAE e RMSE são menores em 0,33 (redução de 30%) e 0,56 (redução de 21%), respectivamente; enquanto a precisão é maior em 6,13 (aumento de 7,21%). Além disso, os dados da Tabela 4 sugerem que 5 camadas ocultas é a quantidade mínima para se obter resultados aceitáveis, enquanto utilizar mais que 6 camadas não traz benefícios. Adicionalmente, os impactos de dobrar o

tempo de previsão em relação ao cenário de 30 minutos são surpreendentes:

- Com exceção das versões de 7 camadas, o MAE de todas as versões no cenário de 60 minutos foi praticamente igual ou inferior às versões do cenário de 30 minutos. Mesmo com o aumento do MAE de 0,16 entre as versões de 7 camadas, é notório que o modelo conseguiu obter um desempenho semelhante ou superior em quase todos os casos, mesmo com o aumento da complexidade do problema;
- Em média, todos os RMSE foram menores em 0,36 , o que significa que o modelo teve uma redução na penalidade de previsões *outliers*;
- Houve uma redução média de 5,6 na precisão das versões com 3 e 7 camadas e um aumento médio de 3,03 das versões com 4, 5 e 6 camadas;

De forma geral, esses resultados mostram que, embora o desempenho em 60 minutos seja naturalmente inferior ao cenário de 10 minutos (como era esperado devido à maior complexidade), o modelo conseguiu manter um nível de desempenho satisfatório, especialmente nas versões com 5 e 6 camadas ocultas.

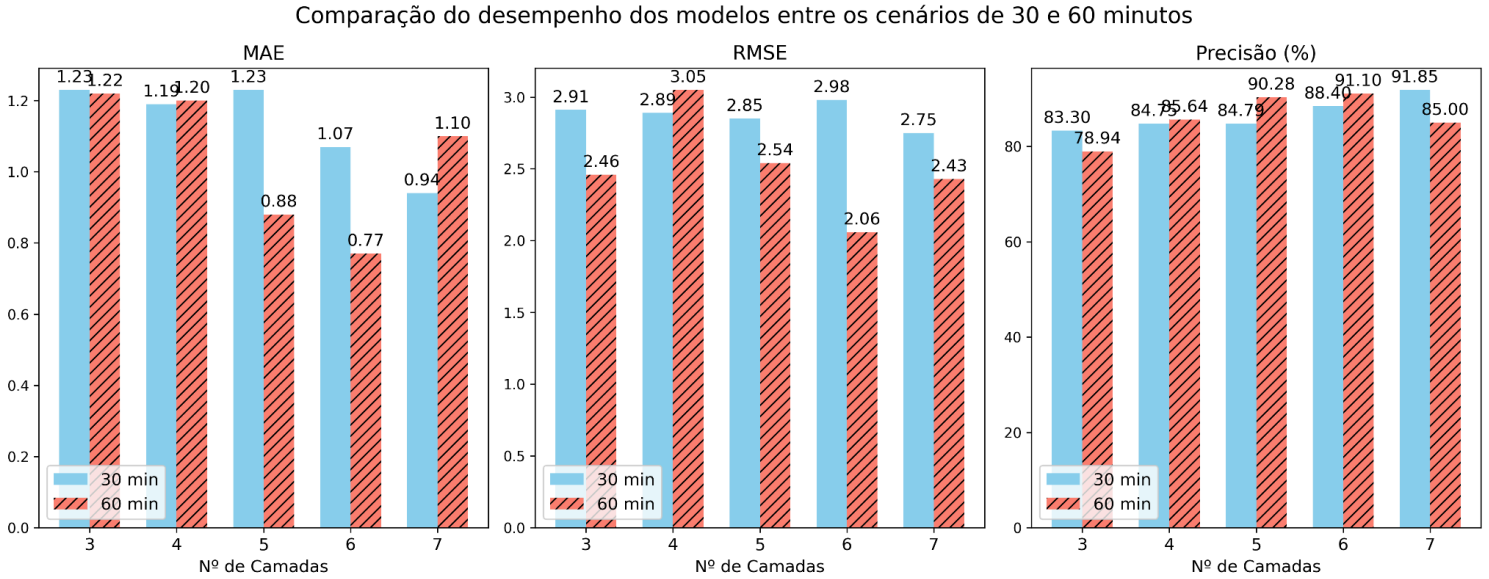

Figura 9 – Gráfico de barras comparativo entre os resultados das métricas para previsões de 30 e 60 minutos.

Abaixo, para facilitar a visualização dos dados, todos os resultados são reunidos na Tabela 5.

Tabela 5 – Todos os resultados de validação do modelo

		Métrica		
Nro. de Camadas	**Cenário de Previsão**	**MAE**	**RMSE**	**Precisão**
3	10 minutos	0,68	2,02	95,01%
	30 minutos	1,23	2,91	83,30%
	60 minutos	1,22	2,46	78,94%
4	10 minutos	0,60	1,96	96,83%
	30 minutos	1,19	2,89	84,75%
	60 minutos	1,20	3,05	85,64%
5	10 minutos	0,60	1,96	95,35%
	30 minutos	1,23	2,85	84,79%
	60 minutos	0,88	2,54	90,28%
6	10 minutos	0,55	1,93	96,56%
	30 minutos	1,07	2,98	88,40%
	60 minutos	0,77	2,06	91,10%
7	10 minutos	0,66	1,96	95,02%
	30 minutos	0,94	2,75	91,85%
	60 minutos	1,10	2,43	85,00%

6.2 Discussão dos Resultados da Experimentação

Atualmente, apesar de várias técnicas terem sido propostas para encontrar um número adequado de camadas ocultas e seus neurônios, essa questão ainda é um problema que confunde os pesquisadores [UZAIR, JAMIL 2020]. Quando o número de camadas ocultas de um modelo é muito superior quando comparado com a complexidade do problema, o super-ajuste (ou *overfitting*) ocorre. Quando isso acontece, dá-se início a um processo de supertreinamento do modelo, o que acarreta no aumento da complexidade temporal e perda da generalização [UZAIR e JAMIL 2020]. Porém, quando o número de camadas ocultas de um modelo é inferior à complexidade do problema, o sub-ajuste (ou *underfitting*) acontece. Nesse caso, a eficiência do modelo é fortemente afetada negativamente [UZAIR e JAMIL 2020].

No estudo conduzido por [UZAIR e JAMIL 2020], uma vasta gama de artigos foram

investigados. E, mesmo com uma variedade de problemas enfrentados e camadas ocultas utilizadas, foi possível notar que poucas camadas ocultas tendem a gerar resultados ruins quando problemas complexos e grandes conjuntos de dados são introduzidos. Ou seja, é necessário em alguns casos aumentar o número de camadas ocultas até uma quantidade aceitável.

Com base nessas informações, é possível chegar nas possíveis causas dos resultados apresentados na Seção 6.1. O cenário de 10 minutos é aquele com complexidade razoavelmente inferior aos demais, consequentemente, mesmo um baixo número de camadas ocultas é suficiente para alcançar resultados altamente satisfatórios. Todavia, quando aumentamos o tempo de previsão, estamos aumentando a profundidade do problema. Dessa forma, essa elevação na complexidade gera maior propensão a erros e pode exigir mais camadas ocultas para alcançar bons resultados. Por isso, conforme o problema dificultava, o mesmo número de camadas ocultas enfrentou um aumento nas métricas de erro, além de uma redução na precisão. Veja esse fenômeno na Figura 10, que compara o desempenho do modelo com 3 camadas ocultas nos três cenários de previsão.

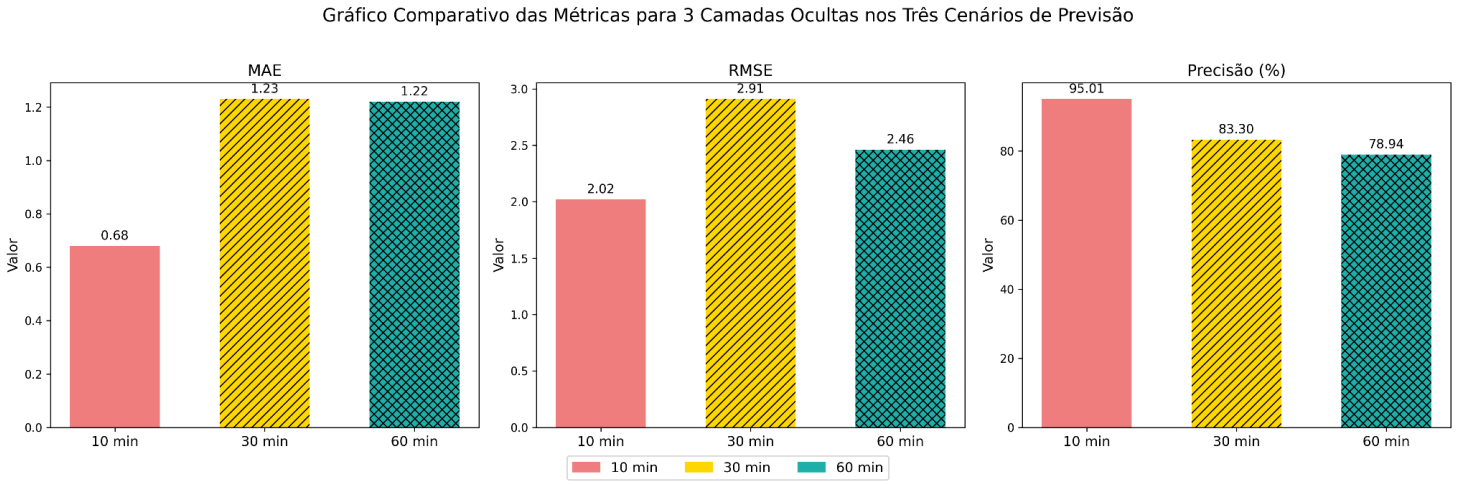

Figura 10 – Gráfico de barras comparativo dos resultados do modelo com 3 camadas ocultas entre os cenários de previsão.

Além disso, é possível notar variações de desempenho das versões do modelo em um mesmo cenário, principalmente nos cenários de 30 e 60 minutos. Observe na Figura 11 que as métricas de erro do modelo no cenário de 60 minutos tendem a cair, e a precisão tende a subir conforme o número de camadas ocultas se aproxima do valor 6. No entanto, esse comportamento se inverte quando se utiliza 7 camadas. Uma provável explicação para esse comportamento é que 3 camadas ocultas não são adequadas para a complexidade do cenário, resultando em *underfitting*. Todavia, conforme o número de camadas é aumentado, mais se aproxima do valor ideal (que nesse caso é 6) e melhores são os resultados. A partir

desse ponto, adicionar mais camadas significa se distanciar da complexidade do problema. Nesse caso, o *overfitting* ocorre e o desempenho tende a piorar.

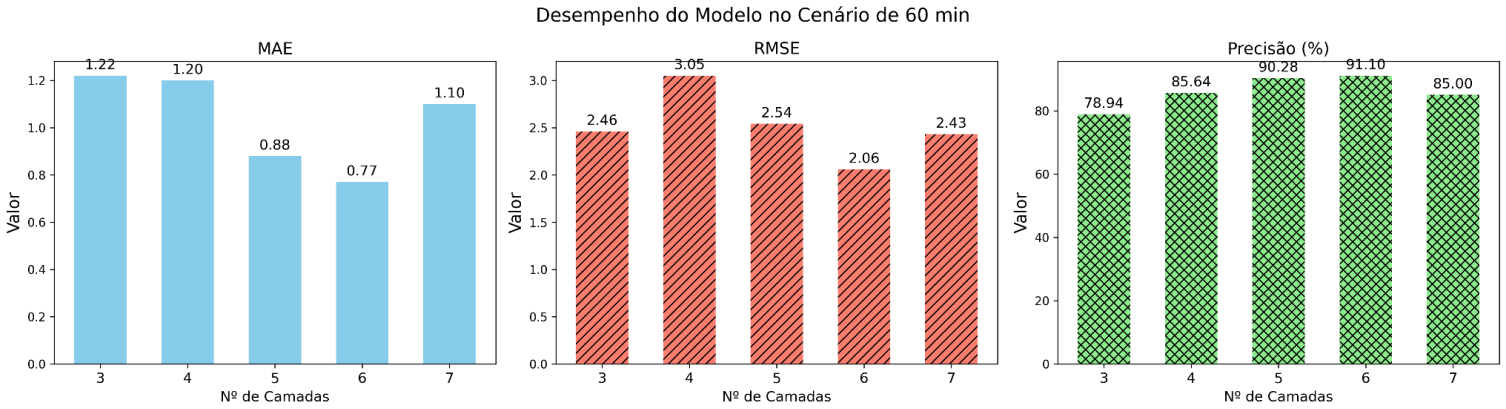

Figura 11 – Gráfico de barras comparativo dos resultados do modelo apenas para o cenário de 60 minutos.

7. Considerações Finais

Esta monografia apresentou o desenvolvimento e a avaliação de um modelo baseado em Deep Learning (DL) para apoiar a previsão do nível da bateria de smartphones, visando contribuir para o aumento da autonomia desses dispositivos. Os resultados obtidos revelam que, mesmo com arquiteturas relativamente simples de redes neurais profundas, é possível alcançar bons resultados na previsão do nível da bateria em diferentes cenários de tempo (como 10, 30 e 60 minutos). Além disso, de modo geral, foi observado que aumentar o número de camadas ocultas até certo ponto proporcionou uma melhora no desempenho em um mesmo cenário, e também garantiu que a piora dos não acentuasse ainda mais conforme o tempo de previsão aumentava.

Diferentes cenários foram utilizados visando observar o impacto do aumento da complexidade do problema sob o modelo. Contudo, independente do modelo possuir um número de camadas ocultas ideal em cada cenário, o melhor cenário a ser utilizado é aquele que atende melhor às necessidades do usuário. Por exemplo, caso o usuário esteja com a bateria razoavelmente cheia, 60 minutos são suficientes para ele se planejar. No entanto, se a bateria estiver praticamente vazia, não faz sentido saber quanto será a descarga em 1 hora, mas sim em 10 minutos.

Vale destacar que, apesar das melhorias feitas para preencher as lacunas identificadas, este estudo também possui algumas limitações. A coleta foi realizada a partir de um único dispositivo e padrão de uso, o que pode limitar a generalização dos

resultados. Futuros trabalhos podem expandir esta abordagem ao:

- Utilizar diversos dispositivos e perfis de usuários;
- Explorar arquiteturas mais complexas de redes neurais, como a LSTM, que podem capturar ainda melhor as dependências temporais do consumo energético;
- Expandir esta abordagem para um sistema de recomendação de ações automáticas para economizar energia com base na previsão realizada;
- Evoluir esta abordagem com a introdução de técnicas de IA explicável, contribuindo para uma melhor compreensão das previsões ou sugestões de ações pelos usuários;
- Adaptar a abordagem para um aplicativo que realiza previsões ou sugestões periódicas, baseadas em um intervalo de tempo definido;

Agradecimentos Especiais

Gostaria de fazer os devidos agradecimentos a três pessoas pela grande ajuda no desenvolvimento desta monografia: ao Prof. Dr. Raimundo Barreto, meu orientador; ao Prof. Dr. Moisés Gomes de Carvalho, que é especialista na área de inteligência artificial e me deu dicas valiosas para a construção deste trabalho; e à Mayane Maia, que me auxiliou na revisão do texto e em outros pequenos detalhes quando necessário.

Referências

Ababei, C., & Moghaddam, M. G. (2018). A survey of prediction and classification techniques in multicore processor systems. IEEE Transactions on Parallel and Distributed Systems, 30(5), 1184-1200.

Baladram, S. (2024). Random Forest, Explained: A Visual Guide with Code Examples. Medium. https://medium.com/data-science/random-forest-explained-a-visual-guide-with-code-examples-9f736a6e1b3c

Chantrapornchai, C. & Nusawat, P.. (2016). Two Machine Learning Models for Mobile Phone Battery Discharge Rate Prediction Based on Usage Patterns. Journal of Information Processing Systems. 12. 436-454. 10.3745/JIPS.03.0048.

Chen, D. R., Chen, Y. S., Chen, L. C., Hsu, M. Y., & Chiang, K. F. (2015). A Machine Learning Method for Power Prediction on the Mobile Devices. Journal of medical systems, 39(10), 126. https://doi.org/10.1007/s10916-015-0320-5

Flores-Martin, D., Laso, S., & Herrera, J. L. (2024). Enhancing Smartphone Battery Life: A Deep Learning Model Based on User-Specific Application and Network Behavior. Electronics, 13(24), 4897. https://doi.org/10.3390/electronics13244897

Garche, J., Dyer, C., Moseley, P., Ogumi, Z., Rand, D., & Scrosati, B. (Eds.). (2009). Encyclopedia of Electrochemical Power Sources. Elsevier.

Gaska, B., Gniady, C., & Surdeanu, M. (2018). MLStar: machine learning in energy profile estimation of android apps. In Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (pp. 216-225).

Google. (2024 A). Como trabalhar com dados categóricos. Curso Intensivo de ML. https://developers.google.com/machine-learning/crash-course/categorical-data?hl=pt-br

Google. (2024 B). Embeddings. Curso Intensivo de ML. https://developers.google.com/machine-learning/crash-course/embeddings?hl=pt-br

Google. (2025 A). Dados categóricos: vocabulário e codificação one-hot. Curso Intensivo de ML. https://developers.google.com/machine-learning/crash-course/categorical-data/one-hot-encoding?hl=pt-br

Google. (2025 B). Embeddings: espaço de embedding e embedding estáticos. Curso Intensivo de ML. https://developers.google.com/machine-learning/crash-course/embeddings/embedding-space?hl=pt-br

Google. (2025 C). Embeddings: como receber embeddings. Curso Intensivo de ML. https://developers.google.com/machine-learning/crash-course/embeddings/obtaining-embeddings?hl=pt-br

Handelman, G. S., Kok, H. K., Chandra, R. V., Razavi, A. H., Huang, S., Brooks, M., ... & Asadi, H. (2019). Peering into the black box of artificial intelligence: evaluation metrics of machine learning methods. *American Journal of Roentgenology*, *212*(1), 38-43.

Li, H., Liu, X., & Mei, Q. (2018). Predicting smartphone battery life based on comprehensive and real-time usage data. arXiv preprint arXiv:1801.04069.

Mahesh, Batta. (2019). Machine Learning Algorithms - A Review. International Journal of Science and Research (IJSR). 9. 10.21275/ART20203995.

Mehrotra, D., Srivastava, R., Nagpal, R., & Nagpal, D. (2021). Multiclass classification of mobile applications as per energy consumption. *Journal of King Saud University - Computer and Information Sciences, 33*(6), 719-727. https://doi.org/10.1016/j.jksuci.2018.05.007

Neto, A. S. B., Farias, F., Mialaret, M. A. T., Cartaxo, B., Lima, P. A., & Maciel, P. (2021). Building energy consumption models based on smartphone user's usage patterns. Knowledge-Based Systems, 213, 106680.

Popescu, M. C., Balas, V. E., Perescu-Popescu, L., & Mastorakis, N. (2009). Multilayer perceptron and neural networks. WSEAS Transactions on Circuits and Systems, 8(7), 579-588.

Qian, W., Gechter, F., & Lauri, F. (2019). Comparison of Machine Learning Algorithms on Smartphone Energy Consumption Modeling Issue Based on Real User Context Data. eKNOW 2019, 14

Rainio, O., Teuho, J., & Klén, R. (2024). Evaluation metrics and statistical tests for machine learning. *Scientific Reports*, *14*(1), 6086.

Shaikh, F. K., Zeadally, S., & Exposito, E. (2019). Power consumption analysis, measurement, management, and issues: A state-of-the-art review of smartphone battery and energy usage. IEEE Access, 7, 182113-182172. https://doi.org/10.1109/ACCESS.2019.2960833

SWPERFI. (2024). TucdAndroidDataCollector-app-ext [Repositório de código-fonte]. GitHub.

Thomas, A. J., Petridis, M., Walters, S. D., Gheytassi, S. M., & Morgan, R. E. (2017). Two hidden layers are usually better than one. In Engineering Applications of Neural Networks: 18th International Conference, EANN 2017, Athens, Greece, August 25–27, 2017, Proceedings (pp. 279-290). Springer International Publishing.

Ullah T., Siraj A. H., Andrabi U. M., Nazarov A. Approximating and Predicting Energy Consumption of Portable Devices. (2022). VIII International Conference on Information Technology and Nanotechnology (ITNT), Samara, Russian Federation, 2022, pp. 1-7, doi: 10.1109/ITNT55410.2022.9848659.

Uzair, M., & Jamil, N. (2020). Effects of hidden layers on the efficiency of neural networks. In 2020 IEEE 23rd international multitopic conference (INMIC) (pp. 1-6). IEEE.

Vanderplas, J. (2016). Python data science handbook: Essential tools for working with data. " O'Reilly Media, Inc.".

Wu, Y.; Bai, D.; Zhang, K.; Li, Y.; Yang, F. Advancements in the estimation of the state of charge of lithium-ion battery: a comprehensive review of traditional and deep learning approaches. J. Mater. Inf. (2025, 5, 18). http://dx.doi.org/10.20517/jmi.2024.84