
Universidade Federal de Pernambuco
Centro de Tecnologia e Geociências
Departamento de Energia Nuclear



Curso de Graduação:
Engenharia de Energia

Disciplina:
EN248 Projeto de Graduação (TCC)
Responsável pela Disciplina:
Prof. Alexandre Costa
Período: 2020.2
Local e Data:
Recife
30 de agosto de 2021

Trabalho de Conclusão de Curso

Estratégias de Controle Preditivo de Sistemas Híbridos de Energia com Redes Neurais

Aluno: Leonardo Teixeira Peregrino

Orientador: Alexandre Carlos Araújo da Costa
Departamento de Energia Nuclear, UFPE

Resumo

Um sistema híbrido é composto por múltiplas fontes de energia. Aquelas que possuem dependência meteorológica constituem incerteza no suprimento da demanda, podendo comprometer o funcionamento dos serviços abastecidos. Por isso, o operador do sistema precisa estar apto a prever o balanço energético e decidir a melhor estratégia de controle.

A previsão de diversas séries temporais é uma tarefa difícil de ser feita analiticamente por isso métodos envolvendo inteligência artificial tem sido abordados. Dados históricos são necessários para serem processados mas os modelos possuem grande capacidade de generalização, adequando-se ao cenário.

O presente trabalho teve como objetivo estudar a viabilidade de análise preditiva em sistemas híbridos utilizando aprendizado de máquina. As fontes abordadas foram eólica, solar e diesel, juntamente com reserva de banco de baterias. Os dados utilizados foram retirados da rede SONDA, fornecidos pelo INPE, na estação meteorológica de Petrolina.

Foi utilizado aprendizado supervisionado através de rede neurais. O algoritmo aplicado foi uma *Recurrent Neural Network*. Foi usada a linguagem Python com auxílio de bibliotecas como *numpy*, *sklearn* e *tensorflow*.

As redes neurais apresentaram bom desempenho em horizontes de curto prazo mas degeneraram com passar dos timesteps. A estratégia preditiva teve acurácia de 73% e foi capaz de obter resultados similares às tradicionais. Ainda há espaço para melhora em trabalhos futuros.

Palavras-chave: Série Temporal, Aprendizado de Máquina, Sistemas Híbridos, Redes Neurais

Abstract

A hybrid system is composed of multiple energy sources. Those with weather dependency represent uncertainty in demand provision, which could compromise the functioning of the supplied services. Therefore, the system operator needs to be able to predict the energy balance and decide the best control strategy.

Predicting multiple time series is a difficult task to be done analytically, hence artificial intelligence has been used as alternative. Historical data has to be processed but the models have great generalization capacity, adapting to the scenario.

The present work aimed to study the feasibility of predictive analysis in hybrid systems using machine learning. Wind, solar and diesel sources were addressed along with energy storage. The data used was taken from the SONDA network, provided by INPE, at the Petrolina meteorological station.

Neural Network was used as supervised learning. The algorithm applied was the *Recurrent Neural Network*. Python language was used with the help of libraries like *numpy*, *sklearn* and *tensorflow*.

Neural networks performed well in short-term horizons but degraded with longer times steps. The predictive strategy had accuracy of 73% and was able to obtain similar results to traditional ones. There is still space for improvement in future work.

Keywords: Time Series, Machine Learning, Hybrid Systems, Neural Networks

Sumário

1	Introdução	1
1.1	Eletricidade no Meio Urbano e Rural	1
1.2	Sistemas Híbridos	2
1.3	Inteligência Artificial	2
1.4	Apresentação e Objetivos	2
2	Conceitos Preliminares	4
2.1	Perceptron	4
2.2	Gradiente Descendente	5
2.3	Redes Neurais	6
2.4	Redes Neurais Recorrentes	7
3	Revisão Bibliográfica e Enunciado do Problema	9
3.1	Enunciado	13
4	Metodologia	15
4.1	Solar	16
4.2	Eólica	16
4.3	Diesel	18
4.4	Bateria	18
4.5	Rede Neural	19
4.6	Sistema Híbrido	20
5	Estudo de Caso	22
5.1	Resultados	23
6	Conclusão e Perspectivas	31
	Referências	32
	Apêndice A Rede Neural	34
	Apêndice B Sistema Híbrido	38
	Apêndice C Estratégias	44

Lista de Figuras

1	Visualização da eletrificação no mundo. Fonte: NASA (2012)	1
2	Representação de um Perceptron com n entradas. Fonte: própria.	5
3	Gradiente descendente em uma função custo quadrática. Fonte: própria. . .	6
4	Representação de uma Rede Neural de 3 camadas, 1 oculta. Fonte: própria. .	7
5	BP através da regra da cadeia. Fonte: própria.	7
6	Representação de uma Rede Neural recorrente. Fonte: própria.	8
7	Perfil de carga de exemplo. Fonte: própria. .	15
8	Curva de potência do aerogerador <i>Bergey Excel-10</i> . Fonte: própria.	17
9	Esquema de várias redes para previsão individual de timesteps. Fonte: própria.	19
10	Histograma da velocidade do vento .	22
11	RMSE final de cada rede .	23
12	Previsão de irradiação nos dados de teste .	24
13	Previsão de velocidade do vento nos dados de teste	25
14	Previsão final comparada aos dados reais .	26
15	Comparação da oscilação do SOC .	28
16	Resultados para previsão .	29

Lista de Tabelas

1	Metodologia de dimensionamento de um sistema híbrido	9
2	Estratégias de controle combinadas	11
3	Softwares disponíveis	12
4	Painel YGE 60 Cell Series 2	16
5	Gerador diesel	18
6	Bateria Trojan SPRE 12 225	19
7	Modelo treinado	20
8	Sistema híbrido abordado	21
9	Quantidade de NaNs	22
10	Comparativo de estratégias	29

Siglas

ANN Artificial Neural Network.

API Application Programming Interface.

BP Backpropagation.

BPTT Backpropagation Through Time.

CC Cycle Charging.

COE Cost of Energy.

EWT Empirical Wavelet Transform.

GHI Global Horizontal Irradiation.

GPU Graphical Processing Unit.

HOMER Hybrid Optimization Model for Multiple Energy Resources.

HRES Hybrid Renewable Energy System.

IA Inteligência Artificial.

INPE Instituto Nacional de Pesquisas Espaciais.

KiBaM Kinetic Battery Model.

LCE Levelized Cost of Energy.

LF Load Following.

LPSP Loss of Power Suply Probability.

LSTM Long Short Term Memory.

MLP Multi Layer Perceptron.

MPC Model Predictive Control.

NaN Not a Number.

NPV Net Present Value.

ReLU Rectified Linear Unit.

RMSE Root Mean Squared Error.

RNN Recurrent Neural Network.

SOC State of Charge.

SONDA Sistema de Organização Nacional de Dados Ambientais.

WS_10m Wind Speed at 10 meters.

Nomenclatura

η	Taxa de aprendizado
\mathbf{w}	Matriz de pesos
θ	Característica do modelo
$\varphi(\cdot)$	Função não linear de ativação
W	Matriz de pesos com viés
X	Matriz de entradas

1 Introdução

1.1 Eletricidade no Meio Urbano e Rural

Atualmente, o perímetro urbano costuma ter disponibilidade elétrica de forma ininterrupta. Entretanto, a medida em que afasta-se dos centros urbanos, o acesso à energia se torna cada vez mais difícil. Em 2018, 668 milhões de pessoas vivem no campo sem acesso à eletricidade, de acordo com IEA et al. (2020, pág. 4) . A disparidade pode ser percebida quando vista por imagens de satélite, como na Figura 1.

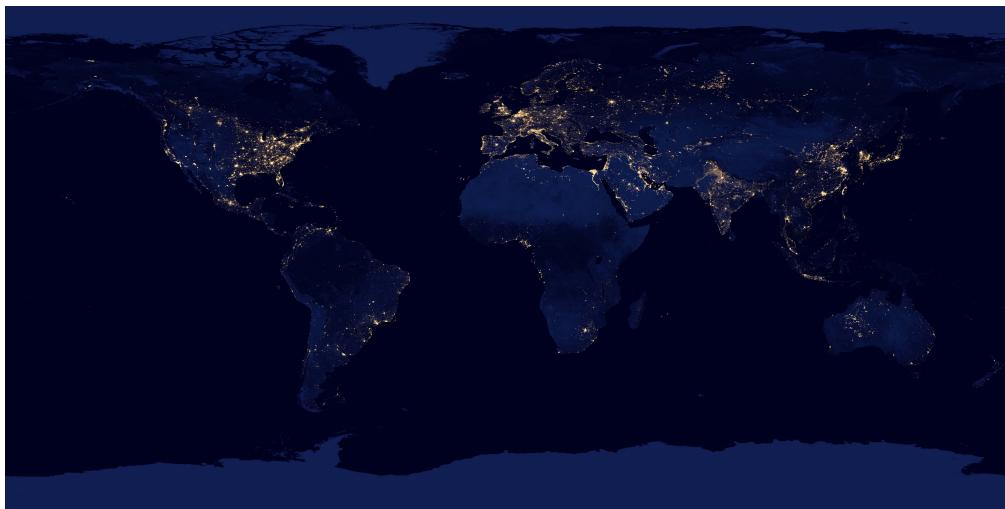


Figura 1: Visualização da eletrificação no mundo. Fonte: NASA (2012)

De forma a contornar tais limitações é comum a prática de geração de energia localmente. A aplicação pode ser feita de forma isolada — nenhuma conexão com rede externa é feita — ou com auxílio parcial da rede. Quando feita desconectada, também conhecida como *off-grid*, é comum em localizações remotas com difícil acesso à linhas de transmissão. Atualmente no Brasil, existem 212 localidades isoladas, a maioria na região Norte segundo ONS (2017) , visto que é a mais densamente vegetada.

Também pode ser feito o abastecimento parcial do consumo. No Brasil, a resolução 786 institui a geração distribuída, com visto em ANEEL (2012) . De acordo com a resolução, é possível converter geração excedente em créditos na concessionária respectiva. Dessa forma, é possível tornar rentável aplicações menores que não fazem uso de baterias. De acordo com o Balanço Energético Nacional em EPE (2020) , mini e micro geração distribuída tiveram um crescimento de 169% em relação ao ano anterior.

1.2 Sistemas Híbridos

Para tornar-se sustentável fora da rede convencional, *off-grid*, diversas fontes podem ser usadas em conjunto. Um grupo de tais fontes constitui um sistema híbrido. As suas aplicações vão além de residenciais, sendo usadas também para serviços de baixo custo de operação e manutenção. Sistemas eletrônicos embarcados como sinalização de trânsito, iluminação pública ou pequenas redes de rádio-telefonia; bombeamento e dessalinização de água por parte de moradores de zonas pobres são alguns exemplos como mostrado em Kaldellis (2010, cap. 1.4).

Sistemas híbridos são comumente compostos por fonte eólica e solar e dependem de questões meteorológicas sendo, portanto, inadequados para encarregar-se de manter oferta constante de serviços como hospitais. A diversificação é usada para mitigar possíveis intermitências no fornecimento, ao aproveitar a complementariedade entre as fontes. Como resguardo, as deficiências são contornadas com banco de baterias ou outra fonte estável, feito diesel.

Fontes intermitentes, isto é, variam no tempo, são séries temporais com vários parâmetros. Como o conjunto, além da complexidade, possui extrema importância para o abastecimento elétrico, faz-se necessário modelar e prever seu comportamento, mitigando possíveis interrupções.

1.3 Inteligência Artificial

Na última década, pôde-se perceber um crescimento acentuado do uso de inteligência artificial. Em especial, as chamadas redes neurais artificiais, Artificial Neural Network (ANN), foi um dos métodos que mais se destacaram. O modelo possui esse nome pois assemelha-se ao cérebro humano, composto por inúmeros neurônios interconectados capazes de aprender a partir do ambiente. O algoritmo baseia-se no classificador Perceptron de Rosenblatt (1958).

ANN por ser computacionalmente custosa é dependente do desenvolvimento de máquinas capazes de realizar. Recentemente, com a expansão da computação em nuvem, *cloud computing*, e grandes *data centers*, ANN tornaram-se acessíveis. O algoritmo tem alta capacidade de generalização e, por isso, é usado em situações de alta complexidade, como: preços de ações, meteorologia, demanda por energia, entre outros. A desvantagem de métodos baseados em redes neurais é devido à dificuldade de interpretação sobre os parâmetros aprendidos pelo algoritmo.

1.4 Apresentação e Objetivos

As métricas para atuar no controle do sistema requer a avaliação de possíveis estados futuros. As sobrecargas e déficits precisam ser manejadas com antecedência devido à inércia mecânica do sistema diesel ou indisponibilidade do banco de bateria. Por isso, análise

preditiva é utilizada para mitigar essas incertezas.

Inteligência artificial tem adquirido relevância na área de geração de energia, com diversos artigos lançados sobre o tema recentemente.

O trabalho propõe como objetivo geral estudar o uso de redes neurais na previsão de múltiplas séries temporais: eólica e solar. Através disso, investigar a aplicabilidade de análise preditiva para estratégias de controle de um sistema híbrido.

2 Conceitos Preliminares

Segundo Haykin et al. (2009) , no campo de Inteligência Artificial (IA), uma rede neural é um modelo de processamento paralelo massivo constituído por uma unidade base, chamada de neurônio, capaz de armazenar, computar e comunicar informação à seus pares.

Uma ANN é dividida em camadas¹ de neurônios onde cada uma conecta a anterior com a seguinte. Quando formado por várias camadas, é chamado de aprendizado profundo², em referência à distância entre camada de *input* e de *output*.

As ANNs possuem várias vantagens em matéria de aprendizado de máquina. Na forma de aprendizado supervisionado — apresentadas à dados anteriores — são capazes de modelagem preditiva com grande acurácia. Quando novos dados são inseridos no modelo, a rede ajusta seus parâmetros se adaptando. Devido à função de ativação presente no algoritmo, a rede pode mapear relações não lineares de alta complexidade.

2.1 Perceptron

Um Perceptron é a forma mais simples de uma rede neural. É um algoritmo para classificação binária, usado para distinguir se uma entrada pertence a um grupo ou não.

O algoritmo consiste em um neurônio com duas características: uma matriz de peso, representado por \mathbf{w} , e um valor de viés, representado por b . As entradas, x_n , do modelo são multiplicadas por seus respectivos pesos, somadas entre si e com o viés. A operação até aqui consiste em uma aplicação linear e pode ser matematicamente representada como na Equação 1.

$$f(x) = \mathbf{w} \cdot \mathbf{x} + b \quad (1)$$

Em seguida, o resultado é passado para uma função ativadora, no caso do Perceptron é usado a função degrau³, Equação 2. Essa ativação funciona como separador de positivos e negativos.

$$H(x) = \begin{cases} 1 & x > 0, \\ 0 & \text{caso contrário} \end{cases} \quad (2)$$

O processo pode ser resumido na Equação 3. O resultado é o valor previsto \hat{y} , onde 0 e 1 significam cada classe distinta a ser separada.

¹layers

²Deep Learning

³função de Heavside

$$\hat{y} = H \left(\sum_{i=1}^N w_i \cdot x_i + b \right) \quad (3)$$

A Figura 2 representa visualmente o processo descrito.

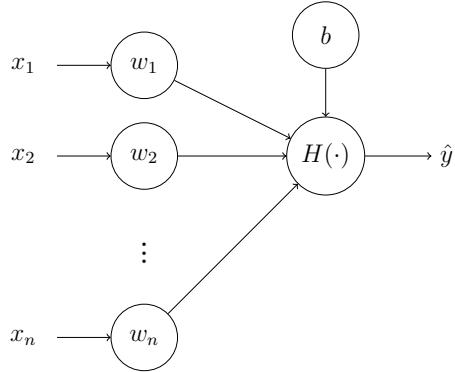


Figura 2: Representação de um Perceptron com n entradas. Fonte: própria.

Inicialmente, são atribuídos valores aleatórios aos pesos e ao viés. É preciso ajustar os parâmetros de forma que eles mapeiem adequadamente o conjunto entrada com o saída. A solução mais usada é aplicar a aproximação numérica por gradiente descendente.

2.2 Gradiente Descendente

O método consiste em alterar os pesos a cada iteração de modo a minimizar uma função custo pré-definida. A função custo mede o quanto distante os pesos atuais mapearam a entrada do resultado esperado. Comumente é usado o desvio quadrático médio, representado na Equação 4, onde \hat{y}_i é o valor previsto e y_i é o valor real em uma i -ésima entrada.

$$J = \frac{1}{2n} \sum_{i=0}^n (\hat{y}_i - y_i)^2 \quad (4)$$

É desejado minimizar o valor de J que, nesse caso, é uma função de \mathbf{w} e b . De forma a simplificar os parâmetros, pode-se considerar o viés como parte de \mathbf{w} e sempre o multiplicar por um. Em notação matricial fica representado como na Equação 5.

$$W^T X = \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 & \dots & \theta_{n_x} \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_{n_x} \end{bmatrix} \quad (5)$$

Na Equação 5, θ são os parâmetros a serem ajustados e θ_0 é equivalente ao viés.

É sabido que a derivada de uma função é equivalente a inclinação da reta tangente e quando positiva indica crescimento. Logo, o problema em questão precisa diminuir W quando a derivada do custo for positiva e aumentar quando ela for negativa, pois em ambas situações o custo é minimizado.

A situação é ilustrada na Figura 3. A linha vermelha representa o caminho das iterações e a preta as derivadas pontuais.

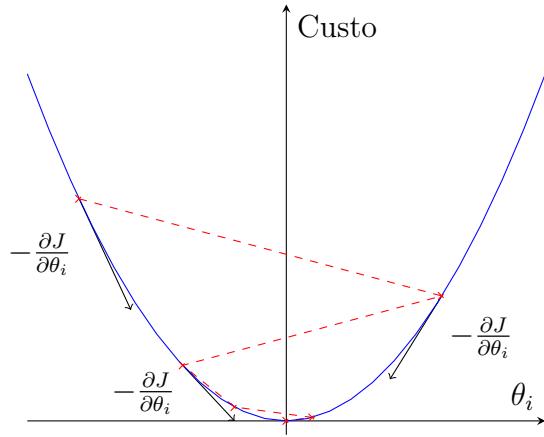


Figura 3: Gradiente descendente em uma função custo quadrática. Fonte: própria.

A Equação 6 representa a generalização do cálculo numérico para o parâmetro θ_i de uma função custo qualquer. Cada iteração atualiza a matriz de pesos W de acordo com sua direção de decrescimento. O parâmetro η é inserido para regular a intensidade da alteração, chamada taxa de aprendizado⁴.

$$\theta_{i+1} \leftarrow \theta_i - \eta \frac{\partial J}{\partial \theta_i}(\theta_0, \dots, \theta_n) \quad (6)$$

2.3 Redes Neurais

Uma rede neural é a generalização do Perceptron para uma quantidade qualquer de neurônios e uma função ativadora qualquer. Por esse motivo, a rede pode também ser chamada de Perceptron Multicamada ou MLP⁵.

O objetivo da função ativadora no MLP é introduzir não linearidade no algoritmo. Caso não fosse aplicada entre os neurônios, a rede seria uma composição de aplicações lineares, o que é equivalente a uma única aplicação. Funções ativadoras comuns são a função logística, tangente hiperbólica e a Rectified Linear Unit (ReLU).

⁴Learning Rate

⁵Multi Layer Perceptron

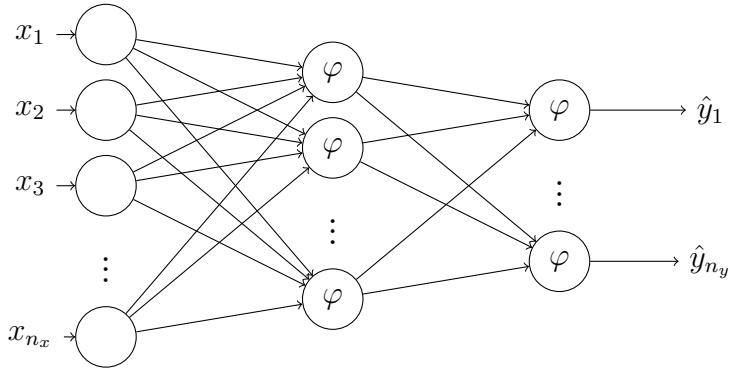


Figura 4: Representação de uma Rede Neural de 3 camadas, 1 oculta. Fonte: própria.

O processo de treinamento é composto por duas fases, a de *Feedforward* e a de *Backpropagation* (BP). A primeira, é apenas calcular aplicação linear dos pesos e ativação da entrada até a saída da rede. A segunda consiste em propagar de forma reversa os gradientes, da saída até a entrada.

Para realizar a BP, é preciso encontrar a derivada do custo em relação cada nodo rede. Como cada neurônio é uma aplicação do anterior, a taxa de variação é calculada através da regra da cadeia, como visto na Figura 5.

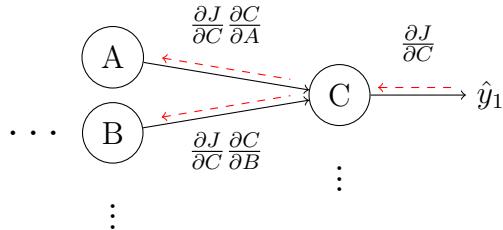


Figura 5: BP através da regra da cadeia. Fonte: própria.

Propagando todas as variações, os pesos da rede são atualizados de acordo com a Equação 6.

2.4 Redes Neurais Recorrentes

Em séries temporais, o valor a ser previsto num determinado tempo t possui algum grau de dependência com os valores anteriores, não sendo apenas função do valor t . Não é esperado que a irradiação ou a velocidade do vento mudem bruscamente, mesmo que variações aconteçam.

Uma rede neural comum apenas identifica padrões pelo valor da entrada — o instante de tempo nesse caso — sendo portanto inadequada para situações onde a proximidade de cada entrada é relevante.

Em tal situação, é preciso implementar uma forma de dependência entre entradas próximas. A solução é feita na forma de redes neurais recorrentes. Uma rede recorrente passa o cálculo de uma predição em t para o próximo cálculo em $t + 1$. A RNN pode ser representada como na Figura 6.

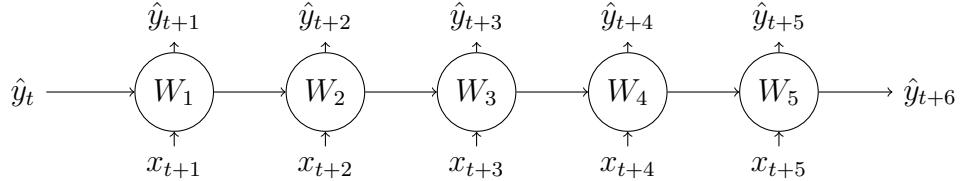


Figura 6: Representação de uma Rede Neural recorrente. Fonte: própria.

O gradiente descendente passa a ser aplicado ao longo de *timesteps* diferentes: BP através do tempo ou Backpropagation Through Time (BPTT). Para redes muito profundas, o processo de treinamento com BP acaba por gerar um problema chamado de desaparecimento de gradiente⁶. Calcular vários gradientes seguidos o faz tender a 0, visto que o valor entre cada camada possui pequena diferença. Por isso, treinar tais redes se torna inviável. Alguns modelos propõem soluções na forma de células que carregam informação entre vários *timesteps* como a LSTM, desenvolvida por Hochreiter et al. (1997) .

⁶vanishing gradient

3 Revisão Bibliográfica e Enunciado do Problema

Uma das maiores dificuldades de implementação de um sistema híbrido de energia é a insegurança de abastecimento. Como as fontes renováveis dependem de efeitos meteorológicos, é comum o super dimensionamento da aplicação. Tal prática resulta em um desenho de sistema custoso mas com vista a garantir a disponibilidade elétrica, como visto em Deshmukh et al. (2008) .

Ainda em Deshmukh et al. (2008) , é visto que abordagem mais frequente para estimar um sistema híbrido é avaliar cada fonte separadamente para depois as analisar em conjunto. Caso as previsões individuais sejam acuradas o suficiente, a previsão conjunta será a ótima.

Kaldellis (2010, cap. 3) propõe que o problema de desenho de um sistema híbrido ideal pode ser dividido em 4 etapas: síntese, design, operação e otimização. A Tabela 1 sintetiza cada uma das etapas propostas. A otimização pode ser aplicada em todas as outras etapas de forma a melhorar o resultado. O problema de operação é crítico, dado que existem muitos modos alternativos de operar um sistema e satisfazendo diferentes métricas operacionais.

Tabela 1: Metodologia de dimensionamento de um sistema híbrido

Problema	Entrada/dado	Resultado
Síntese/ Configuração	Requerimentos do sistema. Disponibilidade de recurso. Considerações alternativas. Especificações básicas.	Componentes incluídos Topologia
Design/ Dimensionamento	Configuração do sistema Resultados do problema de síntese. Requisitos detalhados do sistema. Dados básicos	Tamanho dos componentes Custo de investimento
Análise de operação	Projeto de sistema. Modo operacional. Requisitos, dados técnicos. Dados de custo.	Cálculo de fluxos, eficiências, custos, etc
Otimização	Critérios a serem otimizados. Restrições operacionais Restrições técnicas Restrições ambientais	solução para o problema, cumpridas as restrições

Como visto em Kusakana et al. (2013) existem vários métodos de dimensionamento de sistemas híbridos solar-eólico tais como: suprir a média mensal anual; considerar o mês mais

desfavorável; evitar a probabilidade de perda no fornecimento de energia (LPSP); métodos de dimensionamento usando software; entre outros.

Em Deshmukh et al. (2008) também é feita uma revisão bibliográfica sobre a modelagem de componentes, desenho e métricas de Hybrid Renewable Energy System (HRES). Foi observado que aproximadamente 90% dos estudos realizados são sobre aspectos de modelagem ou econômicos. Em relação às fontes presentes, energia fotovoltaica com eólica representam 56% do número de publicações, seguido por HRES de fotovoltaica com 23% em seguida de eólica com 21%. Outras combinações híbridas revisadas em fase de pesquisa incluem hidro, biomassa, célula de combustível e lixo municipal.

O consumo de combustível é um dos componentes mais onerosos da vida útil de um gerador a diesel. Portanto, determinar o melhor momento para iniciar e parar o diesel é um fator crucial para a otimização. De acordo com Ashari et al. (1999) , essas operações são geralmente feitas com base em certa porcentagem de demanda do sistema ou o estado de carga da bateria (SOC). No estudo, para determinar o valor ótimo de arranque do diesel, realizou-se a comparação dos custos operacionais do gerador diesel e da bateria. O autor concluiu que operar o sistema em uma estratégia que considere o custo de uso da bateria, o custo de consumo do combustível diesel e o perfil de carga fornece um menor custo de operação.

No estudo paramétrico de Elhadidy et al. (2000) , ficou evidente que a geração de energia a diesel é consideravelmente menor com inclusão banco de baterias. Verificou-se que, em média, com a presença de 3 dias de autonomia da bateria, cerca de 27% da carga anual é alimentada a partir do sistema diesel. Com a eliminação do armazenamento da bateria, cerca de 48% da carga anual precisa ser fornecido pelo sistema diesel.

Gupta et al. (2011) propõe um algoritmo de controle para operação otimizada de com estratégias de controle combinadas. Cinco estratégias podem ser usadas, são elas: carga da bateria, descarga da bateria, Load Following, Cycle Charging e Peak Shaving. A Tabela 2 descreve cada uma das estratégias. O algoritmo apresentado foi capaz de projetar com eficiência um sistema de eletrificação ótimo. Apesar das flutuações de radiação solar, o gerador a diesel é capaz de manter a potência constante.

Segundo HOMER Energy by UL (2021) , a estratégia de Cycle Charging consiste em operar o gerador com toda potência disponível e escoar a energia líquida para cargas de menor prioridade, como baterias. A estratégia de Load Following consiste em colocar o gerador diesel para apenas acompanhar a carga primária, apenas produzindo o suficiente para o abastecimento. As cargas de menor prioridade nessa situação ficam a cargo das fontes renováveis.

Em Tazvinga et al. (2014) é empregado o Model Predictive Control⁷ (MPC), um processo

⁷Modelo de Controle Preditivo

Estratégia	Caracterização
Carregamento da Bateria	A bateria carrega até o SOC máximo for atingido.
Descarregamento da Bateria	A bateria descarrega até um dos casos: O SOC mínimo para descarga for atingido. A energia renovável é suficiente para atender a carga. A carga líquida é igual ou maior que a potência mínima de operação do diesel.
Load Following	O gerador a diesel funciona para seguir a carga líquida, sem carga ou descarga para a bateria. Um tempo mínimo de execução de diesel também é aplicado para evitar frequência alta de partida/parada.
Cycle Charging	O gerador a diesel funciona para cobrir a carga líquida e carregar a bateria. O diesel continua durante o tempo mínimo de execução definido; depois disso, o diesel continua funcionando até que uma das condições seja atendida: O SOC desejado foi atingido. A energia renovável é suficiente para atender a carga, carregando ou não as baterias.
Peak Shaving	O diesel operara com potência total. A Bateria é usada apenas para atender às flutuações momentâneas em torno da carga líquida

Tabela 2: Estratégias de controle combinadas

de controle que satisfaz um grupo de restrições do sistema usando uma função de custo definida explicitamente pelo usuário. O trabalho faz uma comparação entre modelos de *loop* fechados e abertos. O modelo de *loop* aberto não possuem um mecanismo de *feedback* entre as previsões de cada *timesteps*, ou seja: \hat{y}_t e \hat{y}_{t+1} não são relacionados. A ausência de *feedback* pode tornar o sistema vulnerável a perturbações nas entradas.

Um MPC em *loop* fechado é proposto para um sistema híbrido solar-eólica-diesel-bateria, com as seguintes restrições:

- demanda de carga em cada momento é satisfeita
- energia fornecida pelo gerador diesel é minimizada
- o sistema de *loop* fechado é robusto com relação a distúrbios na demanda de carga e saída de energia renovável

Duas simulações foram feitas em cada modelo de *loop*, uma com perturbações outra sem. No cenário sem perturbações, a performance dos dois modelos é muito similar, o consumo de diesel é aproximadamente igual. No outro cenário é suposto que o sistema encontra um condição ruim: a demanda de carga é 20% maior que o esperado e a energia eólica e solar são, cada uma, 20% menor que o esperado. Foi percebido que o desempenho do sistema de *loop* fechado é geralmente melhor, indicando que sua robustez com relação a distúrbios é superior ao sistema de *loop* aberto. A razão é que o MPC é capaz de prever estados futuros com base em *feedback* dos estados atuais, influenciados por distúrbios. Em contraste, o controle de *loop* aberto é incapaz de responder a perturbações imprevisíveis e simplesmente começa o gerador diesel quando a demanda de carga é maior do que o esperado. Embora MPC pode ser sofisticada para aplicações domésticas individuais, ainda pode ser benéfico para aplicações industriais.

De forma a automatizar o processo de dimensionamento, alguns softwares foram desenvolvidos, como HOMER, IHOGA e o Hybrid2. Upadhyay et al. (2014) faz um comparativo de entradas e saídas de cada programa. A Tabela 3 apresenta as diferenças verificadas.

Tabela 3: Softwares disponíveis

Software	Entrada	Saída
HOMER	Demand	Dimensionamento ideal
	Recursos	COE
	Detalhes dos componentes	NPV
	Controle do sistema	Percentual renovável
HYBRID2	Demand	Dimensionamento com otimização
	Recursos	Emissão percentual de gases poluentes
	Investimento inicial	COE
	Detalhes dos componentes	Payback
RET Screen	Demand	Custos
	Tamanho do array solar	Redução de emissão
	Database climática	Viabilidade financeira
	Detalhes dos componentes	Análise de risco
IHOGA	Demand	Otimização multi-objetivo
	Recursos	COE
	Detalhes dos componentes	Emissão na vida útil
		Análise de compra e venda de energia

Em matéria de inteligência artificial, Upadhyay et al. (2014) sumariza métodos usados na literatura para o dimensionamento de um HRES. Todos estudos abordaram a partir de

uma perspectiva econômica, com otimização de indicadores como LCE, custo de operação total, custo total do sistema ou custo de energia. As entradas de cada modelo limitam-se majoritariamente a dados periódicos de radiação solar e velocidade do vento. Segundo o autor, essa abordagem através de IA pode ser programada para convergir para a melhor solução mas pode tornar-se ineficiente com o aumento de parâmetros do sistema.

Em redes neurais, a abordagem é altamente dependente dos dados a serem processados. Além de ser necessário quantidade, a qualidade dos dados é determinante para as previsões. Séries temporais possuem características que, quando não tratadas previamente, podem enveredar o resultado. Apesar dos modelos serem muito versáteis e capazes de aprender padrões, ainda sim é benéfico realizar tratamentos para melhorar a performance de previsões temporais, de acordo com Zhang et al. (2005) . As estações do ano representam variações em intervalos de tempo fixos, chamada sazonalidade. Possuir dados durante todo o ano permite o modelo aprender esse padrão. As mudanças climáticas representam uma tendência de crescimento e decrescimento constante em alguns parâmetros.

A literatura de IA aplicada a fontes renováveis é mais extensa em cada fonte individualmente, quando comparada à disponível em HRES. Em Liu et al. (2018) é avaliada a predição de velocidade do vento através de redes neurais *Elman* e LSTM. O trabalho apresentou a decomposição do sinal do vento em frequências diferentes através da transformada empírica *wavelet*. A EWT é composta de filtros de banda da transformada de Fourier, com vista a separar padrões e comportamento estocástico. A combinação desses algoritmos mostrou resultados satisfatórios para previsão de múltiplos intervalos de tempo.

Elsheikh et al. (2019) realizou uma revisão de trabalhos usados para a modelagem solar através de redes neurais. Aplicações de ANN em energia solar foram feitas em áreas como térmica, fotovoltaicas e concentradores. Os benefícios incluem capacidade de generalização, evitar resolver modelos matemáticos complexos, reduzir tempo gasto em modelagem e reduzir gastos com experimentos.

3.1 Enunciado

Como visto, conciliar as particularidades de cada fonte em um sistema híbrido é uma tarefa sujeita a diversos fatores. Mesmo após o dimensionamento da aplicação ainda é necessária a operação adequada para manutenção de expectativa de vida dos componentes e para suprir a demanda.

A problemática a ser abordada em seguida será centrada em um consumidor com necessidade de alta disponibilidade de energia. Devido a esse cenário, será usada a métrica de probabilidade de perda de oferta (LPSP) para avaliar as estratégias de Load Following e Cycle Charging. A Loss of Power Supply Probability, como citada em Upadhyay et al. (2014) ,

pode ser descrita pela Equação 7, onde DE é o déficit energético e P_{load} é a potência demandada pela carga. O sistema híbrido usado será constituído de fontes renováveis solar e eólica, banco de baterias e gerador diesel, visto que esse HRES é amplamente explorado na literatura.

$$\text{LPSP} = \frac{\sum_{t=1}^T \text{DE}(t)}{\sum_{t=1}^T P_{\text{load}}(t) \Delta t} \quad (7)$$

Será realizada uma análise preliminar sobre os dados meteorológicos, levando em consideração a estação meteorológica de Petrolina da Rede SONDA de INPE (2021). Para as previsões, redes neurais recorrentes simples serão empregadas. Os resultados das previsões serão avaliados para decidir sobre a estratégia de controle mais adequada, Cycle Charging ou Load Following.

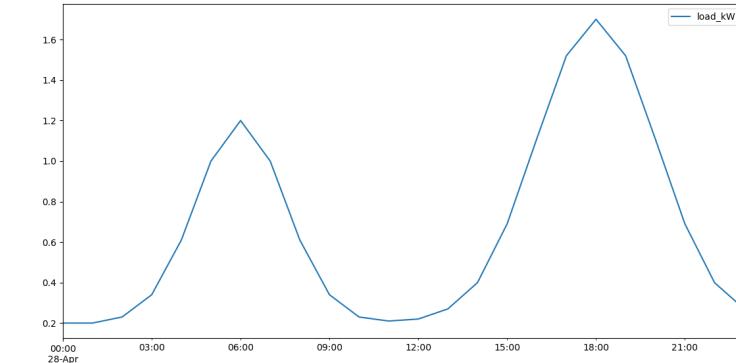
4 Metodologia

Foi considerado o sistema em regime quasi-estacionário, onde entre cada intervalo de tempo é suposto que não há alteração na configuração do HRES.

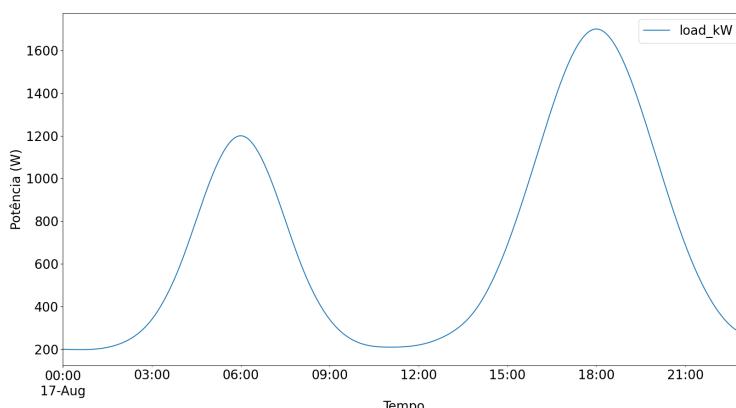
A primeira etapa foi a avaliação e tratamento dos dados, de forma a encontrar uma entrada que não tendencie a rede. Foi avaliado a quantidade de NaNs, problema de falta de dados que pode comprometer a previsão.

A carga a ser abastecida pelo sistema é suposta periódica ao longo de cada dia, com perfil de carga exibido na Figura 7. O perfil foi interpolado para garantir maior granularidade na análise da carga. Foi obtida a carga líquida renovável para cada instante da série temporal de acordo com a Equação 8.

$$E_{\text{ren}} = E_L - E_w - E_s \quad (8)$$



(a) Curva original



(b) Curva interpolada

Figura 7: Perfil de carga de exemplo. Fonte: própria.

Os modelos a seguir foram considerados para cálculo de potência cada componente do sistema.

4.1 Solar

A modelagem de um sistema fotovoltaico começa com o cálculo da irradiância incidente no plano dos painéis. A irradiância I_T é a soma da direta I_b , difusa I_d e a refletida I_r , proporcionais ao ângulo de inclinação, de acordo com a Equação 9.

$$I_T = I_b R_b + I_d R_d + (I_b + I_d) R_r \quad (9)$$

A potência dos painéis depende da área ocupada A_{PV} e a eficiência η dos módulos, de acordo com a Equação 10.

$$P_{PV} = I_T \eta A_{PV} \quad (10)$$

$$\eta_m = \eta_r [1 - \beta(T_c - T_r)] \quad (11)$$

$$T_c = T_a + \left(\frac{T_{NOCT} - 20}{800} \right) I_T \quad (12)$$

O painel considerado foi o *YGE 60 Cell Series 2*, com características na Tabela 4.

Tabela 4: Painel YGE 60 Cell Series 2

Parâmetro	Valor	Unidade
η	19.6	.
β	-0.30	T^{-1}
A_{PV}	1.63	m^2

4.2 Eólica

Para velocidade do vento medidas em uma altura diferente do cubo do aerogerador, é preciso corrigir de acordo com a lei de cisalhamento vertical. As medições mais próximas à superfície são menores devido à interação com o solo. A medida que ascende-se, a velocidade torna-se logarítmicamente maior de acordo com a Equação 13.

$$V_z = V_i \frac{Z^x}{Z_i} \quad (13)$$

A curva de potência de um aerogerador é deduzida a partir de dados de campanhas de medição do recurso eólico local. Quando os dados disponíveis para o sistema híbrido

tem frequência diferente do que foi usado na campanha de medição, a curva de potência original não é mais aplicável. Pode ser feita a correção da curva introduzindo componentes estocásticas e determinísticas de acordo com o modelo de Von-Kármán para rajadas de vento contínuas. Por motivos de simplificação, foi considerado que a curva foi deduzida na mesma frequência amostral da aplicação do sistema, dispensando correção.

A Turbina considerada foi o *Bergey Excel-10*, com curva de potência na Figura 8. A curva também foi interpolada.

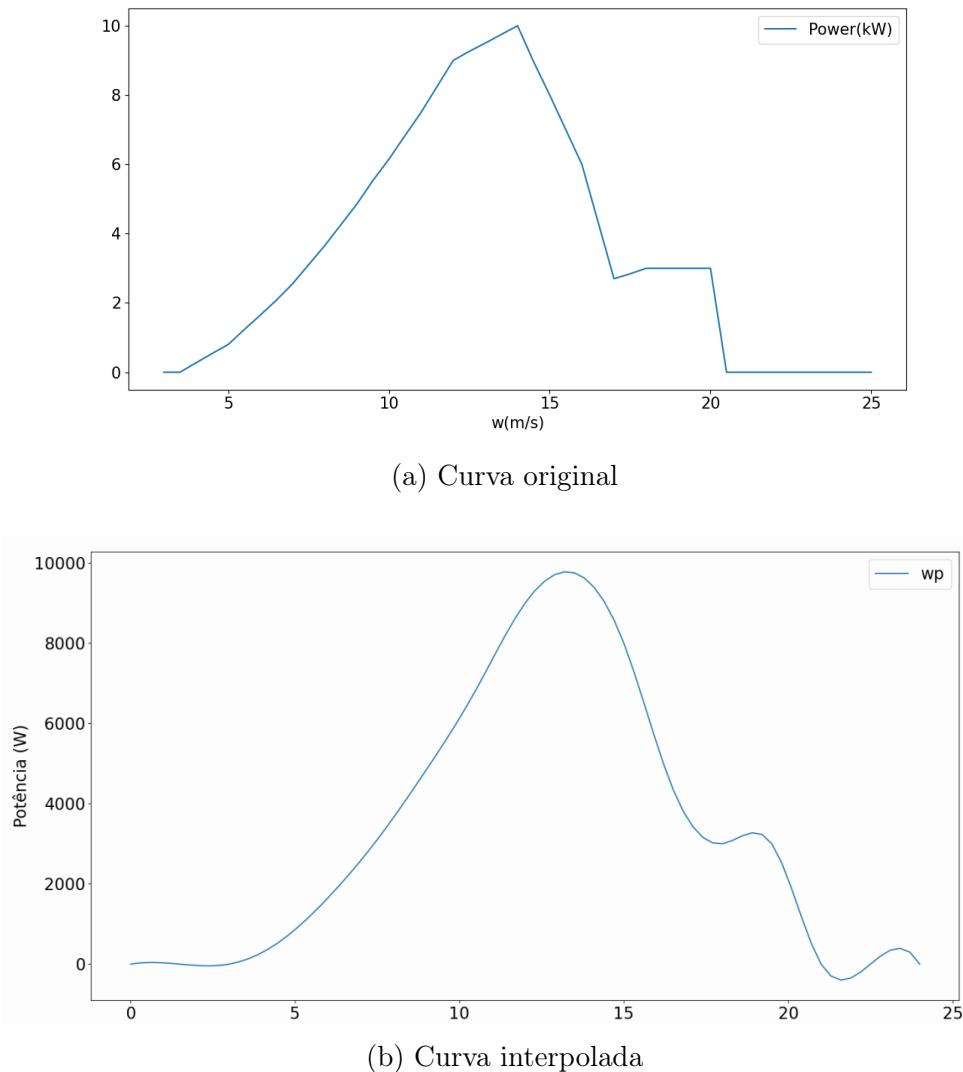


Figura 8: Curva de potência do aerogerador *Bergey Excel-10*. Fonte: própria.

4.3 Diesel

O grupo diesel ser modelado como uma função linear é uma suposição razoável de acordo com resultados experimentais, como visto em Manwell et al. (2006, cap. 6.1) . O coeficiente linear da função é o consumo sem carga, com a máquina parada. A inclinação da reta é dada pela taxa de consumo de combustível por unidade de potência de saída. De acordo com Nema et al. (2009) , para determinar a capacidade nominal do gerador diesel a ser instalado, caso estiver diretamente conectado à carga, então a capacidade nominal do gerador deve ser pelo menos igual à carga máxima.

$$F = a + bP \quad (14)$$

a = consumo sem carga

b = consumo por potência (ℓ/kWh)

O gerador diesel considerado foi o exemplo apresentado em Manwell et al., com características na Tabela 5.

Tabela 5: Gerador diesel

Parâmetro	Valor	Unidade
P_{nom}	15	kW
a	0	ℓ/h
b	1/3	kWh^{-1}

4.4 Bateria

A capacidade do banco de baterias é dimensionado em função do tempo de inatividade das outras fontes, referido como dias de autonomia. Normalmente é assumido 2 ou 3 dias de autonomia, ou seja: as baterias tem capacidade para sustentar sozinhas o consumo por esse período.

As baterias são limitadas em um estado máximo e um mínimo de carga, não podendo ultrapassá-los, como na Equação 15. O estado de carga pode ser atualizado pelas equações 16 e 17 respeitando o modelo cinético de baterias, Kinetic Battery Model.

$$\text{SOC}_{\min} \leq \text{SOC}(t) \leq \text{SOC}_{\max} \quad (15)$$

O KiBaM consiste em admitir que uma parte da capacidade está à disposição para ser consumida imediatamente (*available charge*) e outra está confinada (*bound charge*). Tal fato

decorre da inércia da bateria em transformar energia química em energia elétrica prontamente disponível para o consumo. O modelo restringe a mudança de carga de acordo com a equação 16 para a carga confinada e a equação 17 para carga disponível. Cada bateria possui um parâmetro c que representa o percentual de carga disponível e outo k que mensura a velocidade de conversão de energia.

$$q_a(t+1) = q_ar + \frac{(q_ak'c - i)(1 - r) - ic(k't - 1 + r)}{k'} \quad (16)$$

$$q_b(t+1) = q_b r + q_t(1 - c)(1 - r) - \frac{i(1 - c)k't - 1 + r}{k'} \quad (17)$$

A bateria usada foi a *Trojan Solar SPRE 12 225*, com especificações na Tabela 6.

Tabela 6: Bateria Trojan SPRE 12 225

Parâmetro	Valor	Unidade
V	12	V
E_{\max}	225	Ah
η	75%	.
SOC _{min}	40%	.
SOC _{max}	100%	.

4.5 Rede Neural

A metodologia para previsão foi usar 36 instantes passados de tempo para prever o comportamento meteorológico dos próximos 36 instantes. Para isso, 36 redes neurais foram treinadas, cada uma com o objetivo de prever um instante futuro específico. A rede 1 prevê o instante $t + 1$, a rede 2 prevê o instante $t + 2$ e assim em diante até a trigésima sexta. Todas recebem a mesma entrada, os 36 instantes passados, como ilustrado na Figura 9. As séries temporais usadas foram a de radiação solar e a velocidade do vento, cada uma com suas respectivas previsões, totalizando 72 ANNs.

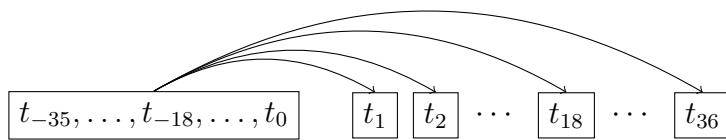


Figura 9: Esquema de várias redes para previsão individual de timesteps. Fonte: própria.

A arquitetura para todas redes foi a mesma, composta células RNN simples. O modelo foi feito utilizando o *Keras*. A ferramenta é uma API do *TensorFlow*, biblioteca de apren-

dizado de máquina desenvolvida pelo *Google*. Para treinamento e teste da rede foi usado a plataforma *Google Colab* que permite computação em nuvem com disponibilidade de GPU. Outras arquiteturas foram avaliadas como a LSTM mas, como obteve resultados semelhantes à RNN, foi escolhida a mais simples.

Os dados que entram na rede foram normalizados: subtraídos do menor valor e divididos de pela diferença entre máximo e mínimo. O processo ilustrado pela Equação 18 tem objetivo de limitar os dados no intervalo de $[0, 1]$. Dessa forma, evita-se problemas de cálculo numérico como *overflow* durante o processo de treinamento e o gradiente descendente é capaz de convergir muito mais rápido.

$$x'_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \quad (18)$$

Além das células RNN, também foi usado *dropout*. A técnica consiste em aleatoriamente desativar um percentual determinado de neurônios de uma camada. Os neurônios desativados não passam informação à próxima camada. Dessa forma, no processo de treinamento, a rede aprende a não depender de cada neurônio individualmente, reduzindo a probabilidade de um nodo enviesar toda a ANN. O sumário da rede é exibido na Tabela 7.

Tabela 7: Modelo treinado

(a) Arquitetura

Camada	Shape de saída	Parâmetros
SimpleRNN	(None, 36, 64)	4224
Dropout	(None, 36, 64)	0
SimpleRNN	(None, 32)	3104
Dropout	(None, 32)	0
Dense	(None, 1)	33

(b) Parâmetros

Total de parâmetros	Parâmetros treináveis	Parâmetros não treináveis
7,361	7,361	0

4.6 Sistema Híbrido

Foi considerado um sistema abastecido por 10 módulos fotovoltaicos, 1 aerogerador, 1 gerador diesel e banco de bateria com 1 dia de autonomia, como mostrado na Tabela 8. Dessa forma, a tensão do barramento de corrente contínua ficou em 300 volts de fotovoltaica mais 220 volts do aerogerador, totalizando 520.

Tabela 8: Sistema híbrido abordado

Equipamento	Modelo	Quantidade
Aero gerador	Bergey Excel-10	1
Painel	YGE 60 CELL Series 2	10
Gerador Diesel 15 kW		1
Baterias	Trojan Solar SPRE 12 225	20

Foi feita uma comparação considerando a LPSP para as duas estratégias: LF. O gerador diesel em LF entrará em operação caso a bateria não suprir o déficit, acompanhando a carga líquida. Em CC, quando há déficit de energia, o gerador diesel é ligado na máxima potência disponível que não gere excesso, escoando a sobra para a bateria. Para manter a vida útil do gerador, tempos mínimos de operação foram considerados.

5 Estudo de Caso

Os dados avaliados possuíam considerável quantidade de NaNs, como exposto na Tabela 9. Para evitar o comprometimento do modelo a ser treinado, o período dos 3 primeiros meses de 2013 foram escolhidos para servir de dado de entrada. O período for composto dos 3 primeiros meses do ano. Os dados foram separados em grupos: treinamento com 75% e teste com 25% do total. Dessa forma, foram 2 meses para treinamento do modelo e 1 para teste.

Tabela 9: Quantidade de NaNs

Data	WS_10m	GHI
2006	0	0
2007	58185	58185
2008	137714	137714
2009	26791	26791
2010	5651	5651
2011	3120	3120
2012	38798	38798
2013	5520	11269
2014	3	2106
2015	10278	39468
2016	13380	21841
2017	126826	126826

A velocidade do vento no local foi insuficiente para manter a operação da aerogerador por longos períodos, devido a sua velocidade de *cut-in*. O histograma da Figura 10 mostra a frequência da velocidade corrigida pela Equação 13 para uma altura de 35 metros. Percebe-se que a maioria encontra-se ao redor de 4 m/s , próximo ao começo de operação.

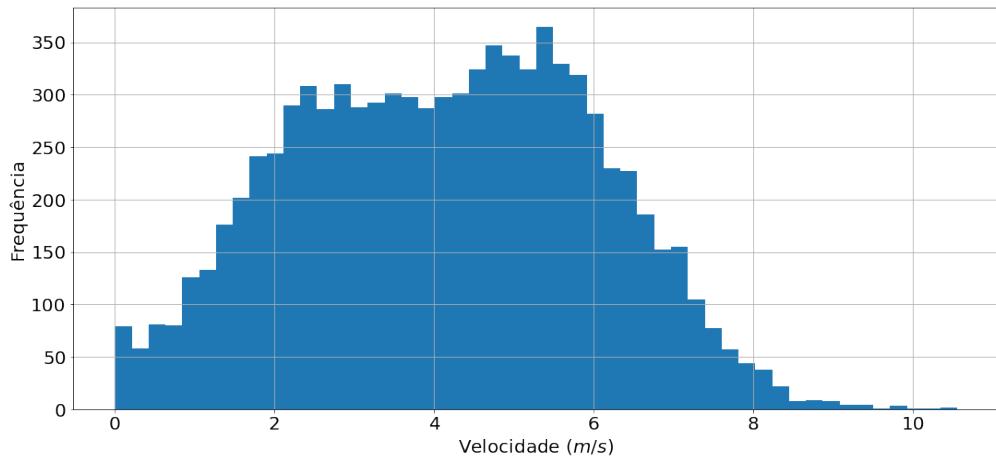


Figura 10: Histograma da velocidade do vento

5.1 Resultados

O treinamento das redes atingiu baixos valores de RMSE, permanecendo na segunda casa decimal, em parte por causa da normalização da Equação 18. Ficou perceptível que a medida em que tenta-se realizar previsões de mais longo prazo, o desempenho diminui, como exposto na Figura 11. Na figura, a linha laranja, com legenda de GHI⁸, representa as redes de previsão solar. A linha azul, WS_10m⁹, representa as redes de previsão eólica.

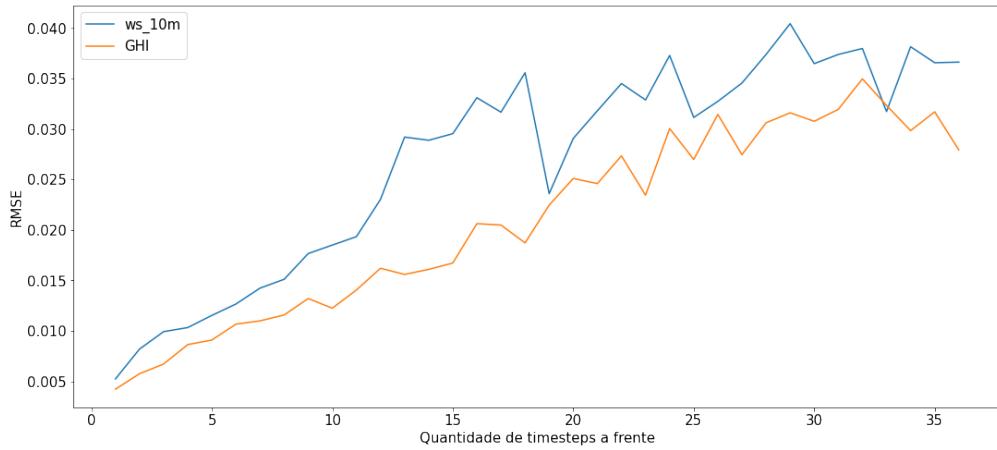
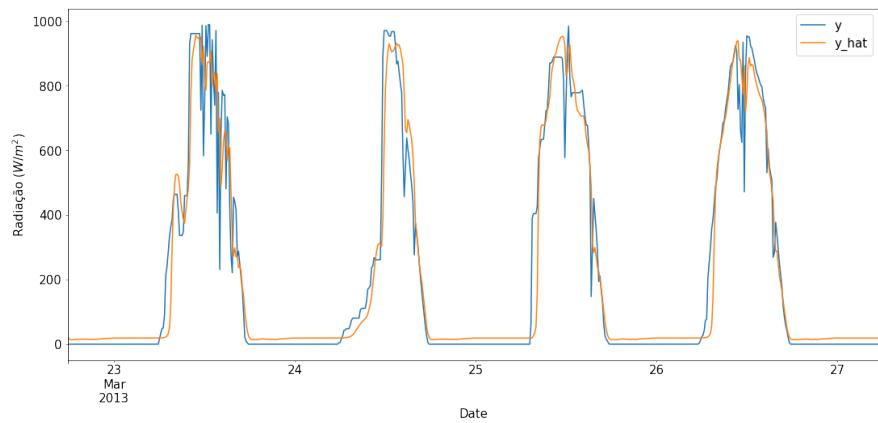


Figura 11: RMSE final de cada rede

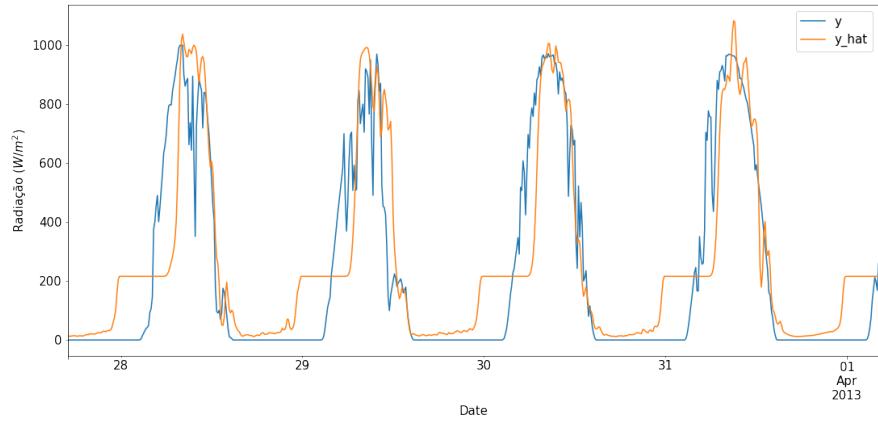
As previsões da rede eólica inicialmente seguiram os dados mas nas últimas pode se ver que o comportamento é descaracterizado. O algoritmo aprendeu a prever a média de forma a reduzir a função custo, como visto na Figura 13. A previsão solar teve melhor desempenho, com mais fidelidade aos dados. Ainda sim, os períodos noturnos provocaram platôs de inércia, vistos na Figura 12b e 12c.

⁸Global Horizontal Irradiation

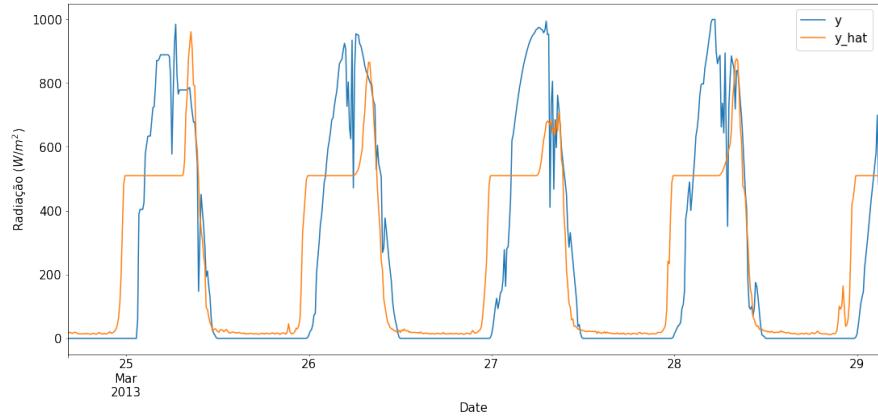
⁹Wind Speed at 10 meters



(a) Previsão realizada pela primeira rede

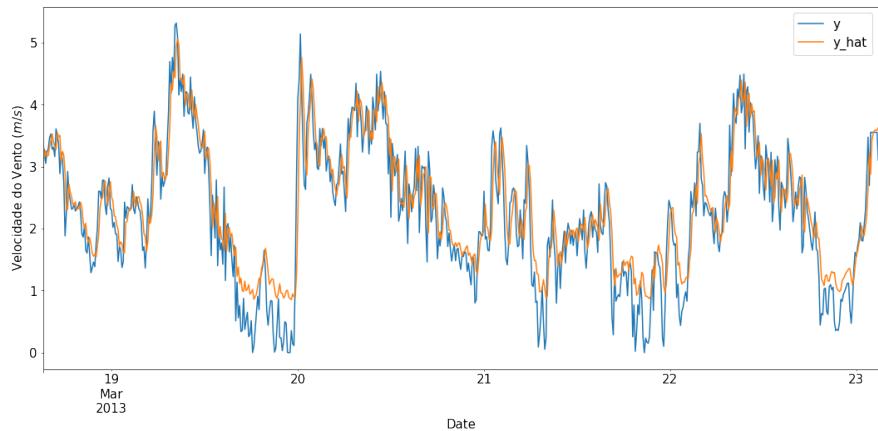


(b) Previsão realizada pela décima oitava rede

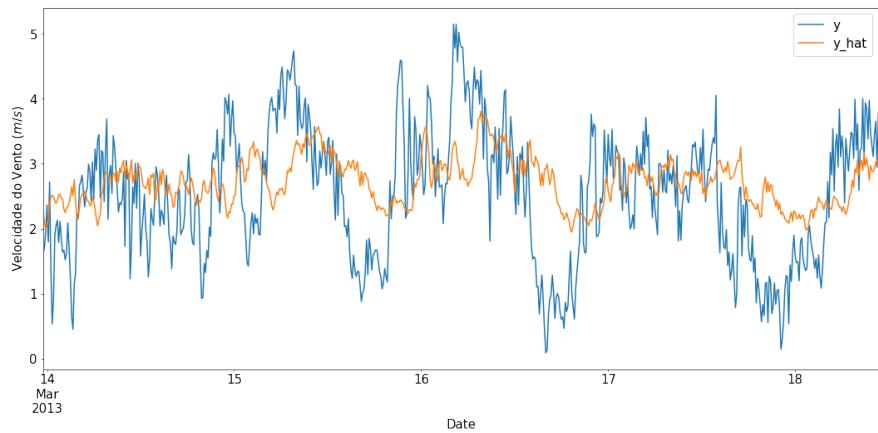


(c) Previsão realizada pela última rede

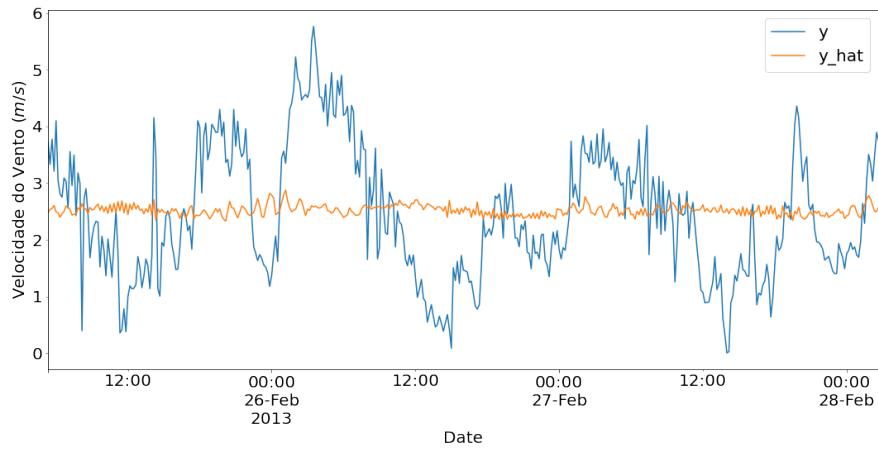
Figura 12: Previsão de irradiação nos dados de teste



(a) Previsão realizada pela primeira rede



(b) Previsão realizada pela décima oitava rede



(c) Previsão realizada pela última rede

Figura 13: Previsão de velocidade do vento nos dados de teste

Quando aplicadas em conjunto com uma previsão de cada rede, nas figuras 14b e 14a, o resultado aparentou ser satisfatório.

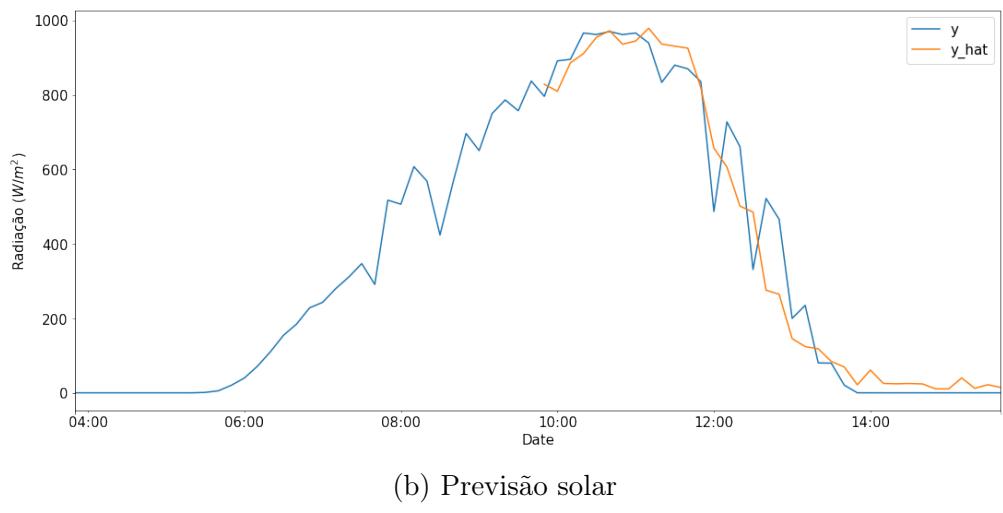
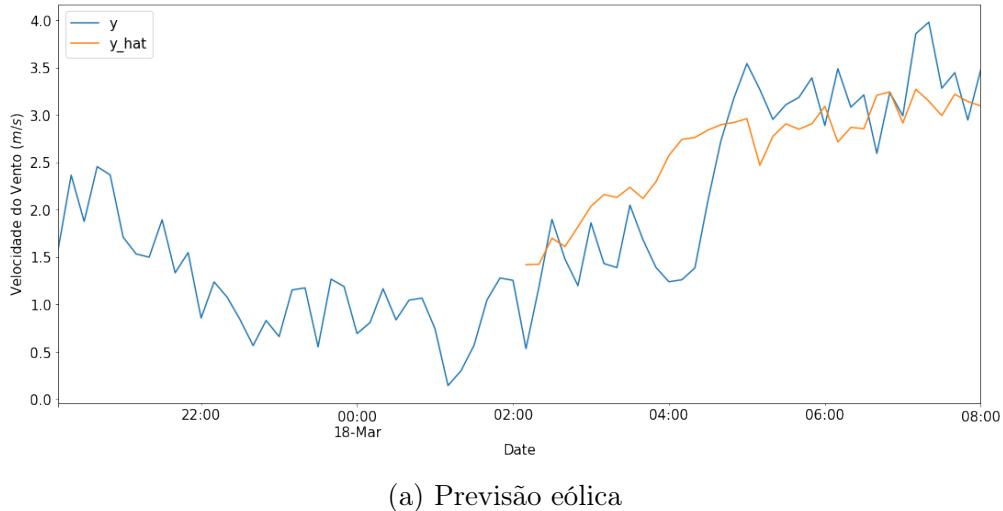


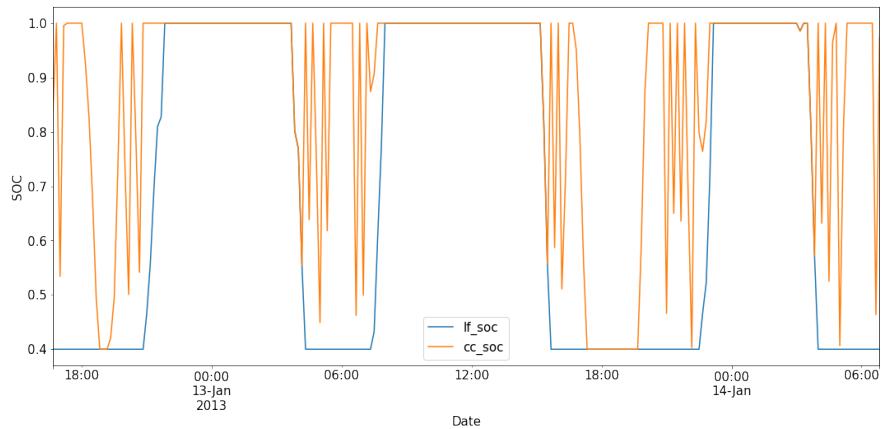
Figura 14: Previsão final comparada aos dados reais

Com apenas fontes renováveis, o sistema apresentou alto índice de Loss of Power Supply Probability, com déficit em 46% dos instantes de tempo. No cenário de uma única estratégia de controle para o sistema, a LPSP caiu consideravelmente. Diante dessas condições, quando aplicada apenas a estratégia de CC, é obtido 5,94% de LPSP, já no caso de LF o percentual é de 7,69%.

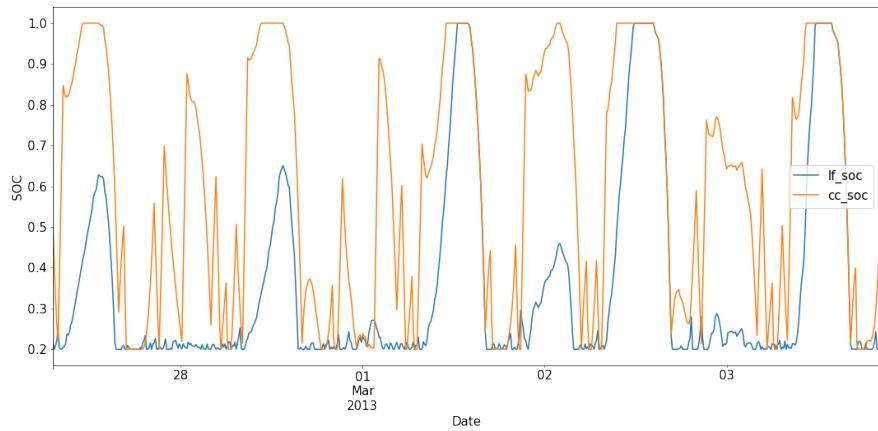
Para mensurar a aplicabilidade de estratégias preditivas, foram feitas previsões da radiação solar e velocidade do vento para todos intervalos. Em posse das previsões, métricas foram calculadas em todos os instantes da série temporal. O primeiro instante considera a

janela de t até $t + 36$, o segundo considera $t + 1$ até $t + 37$ e assim por diante. Em cada uma dessas janelas, 4 cenários de Loss of Power Supply Probability foram avaliados: apenas com LF para dados reais, apenas com CC para dados reais e o mesmo para as previsões. A melhor estratégia dos dados reais foi comparada com a melhor estratégia prevista pelo modelo. Caso os resultados fossem iguais, a previsão foi correta.

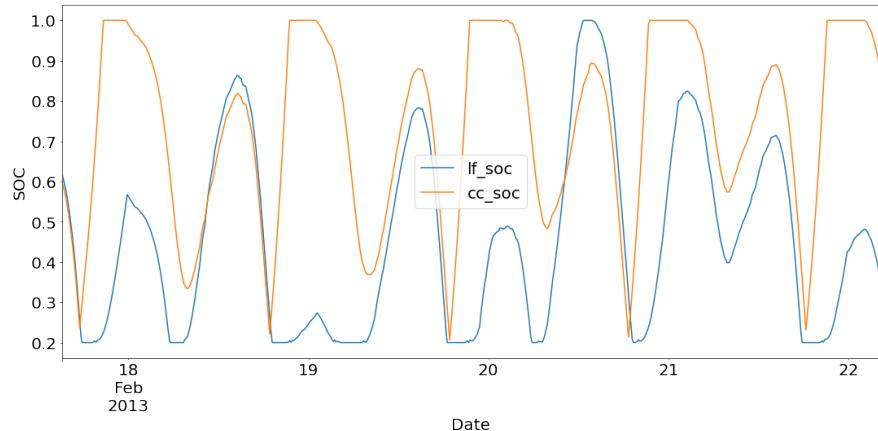
Inicialmente, o diesel poderia arrancar e parar em qualquer instante, por isso o comportamento de CC e LF foi praticamente o mesmo. Foi introduzida uma limitação ao grupo diesel de mínimo tempo de execução, uma vez iniciado. A limitação é comum em máquinas térmicas, com vista a preservar a vida útil do equipamento. Foram avaliados diferentes valores para o mínimo tempo de execução. O estado de carga da bateria exibido na Figura 15 ilustra a diferença de comportamento. Devido à CC arrancar sempre com maior potência disponível, a estratégia teve melhor LPSP.



(a) Sem tempo mínimo de funcionamento



(b) Mínima execução de 3 intervalos



(c) Mínima execução de 6 intervalos

Figura 15: Comparaçao da oscilação do SOC

Os resultados para um mínimo tempo do diesel de 6 intervalos foram traçados em uma matriz de confusão, visto na Figura 16. As previsões apresentaram cerca de 75% de acurácia, tendo previsto corretamente LF como maior LPSP.



Figura 16: Resultados para previsão

O controle preditivo conseguiu resultados intermediários como exposto na Tabela 10. O comportamento de LPSP está de acordo com o estudo de Das et al. (2019), que avalia diferentes estratégias de controle. No trabalho, é concluído que, no cenário em questão, a maior diferença entre LF e CC é a fração renovável de cada uma e, consequentemente, o consumo de diesel.

Tabela 10: Comparativo de estratégias

	LF	CC	LF e CC
LPSP	5.94%	7.69%	6.87%

O *Google Colab* mostrou-se útil para construção e treinamento de modelos em uma máquina remota. O uso da ferramenta gratuita pode facilitar investimentos com pouca disponibilidade de recursos computacionais. As redes neurais apresentaram baixo RMSE, entretanto previsões de mais longo prazo mostraram comportamentos errôneos: a solar não previu os picos diários, compensando em previsões negativas ao fim do dia e a rede eólica seguiu a tendência mas não acompanhou a amplitude do sinal. O sistema abastecido apenas por fontes renováveis apresentou alta probabilidade de perda de carga. Quando aplicadas as estratégias, tanto LF quanto CC, a LPSP diminuiu consideravelmente. Entretanto, o comportamento

preditivo das estratégias foi similar devido à liberdade de arrancar e parar o gerador diesel a qualquer momento. Quando introduzido limitações de tempo mínimo de operação, CC se tornou a estratégia mais confiável por causa de sua potência de arranque mais alta.

6 Conclusão e Perspectivas

Como visto em, o processo de aplicação de um sistema híbrido é uma problemática com diversos fatores que a influenciam, desde a idealização até operação. As fases de configuração, dimensionamento e operação passam todas por processos de otimização, cada uma com particularidades. As métricas abordadas auxiliaram em decisões de desenho melhores.

A utilização da mesma arquitetura para todas ANN pode ter sido uma das causas, visto que os fenômenos meteorológicos tem frequências diferentes, como início e fim do dia ou estações do ano. Melhorias podem ser feitas através da decomposição dos sinais, como no estudo de Liu et al. (2018) , citado anteriormente. Para as redes da irradiação solar, em estudos futuros é possível conseguir melhores resultados realizando treinamento em dados apenas do período diurno. Para determinar momentos em que há sol, o ângulo azimutal pode ser indicador, considerando limites de nascer e pôr do sol. Uma função custo diferente, como a correlação, para o treinamento poderia minimizar a diferença de amplitude de sinal causada com a RMSE.

Os resultados expuseram que, quando aplicada em contextos mais simples, as estratégias se assemelham. A aplicação pode ser estudada com maior granularidade nos critérios para uma avaliação mais diversa.

Referências

- Agência Nacional De Energia Elétrica. (2012). Resolução Normativa 482 [Online; acessado 1-Abril-2021].
- Ashari, M., & Nayar, C. (1999). An optimum dispatch strategy using set points for a photo-voltaic (PV)–diesel–battery hybrid power system. *Solar energy*, 66(1), 1–9.
- Das, B. K., & Zaman, F. (2019). Performance analysis of a PV/Diesel hybrid system for a remote area in Bangladesh: Effects of dispatch strategies, batteries, and generator selection. *Energy*, 169, 263–276.
- Deshmukh, M., & Deshmukh, S. (2008). Modeling of hybrid renewable energy systems. *Renewable and Sustainable Energy Reviews*, 12(1), 235–249. <https://doi.org/10.1016/j.rser.2006.07.011>
- Elhadidy, M., & Shaahid, S. (2000). Parametric study of hybrid (wind+ solar+ diesel) power generating systems. *Renewable energy*, 21(2), 129–139.
- Elsheikh, A. H., Sharshir, S. W., Abd Elaziz, M., Kabeel, A., Guilan, W., & Haiou, Z. (2019). Modeling of solar energy systems using artificial neural network: A comprehensive review. *Solar Energy*, 180, 622–639.
- Empresa de Pesquisa Energética. (2020). Balanço Energético Nacional 2020 [Online; acessado 31-Janeiro-2021].
- Gupta, A., Saini, R., & Sharma, M. (2011). Modelling of hybrid energy system—Part II: Combined dispatch strategies and solution algorithm. *Renewable Energy*, 36(2), 466–473.
- Haykin, S. S., et al. (2009). *Neural networks and learning machines*. New York: Prentice Hall.
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735–1780.
- HOMER Energy by UL. (2021). HOMER Pro 3.14 User Manual [Online; acessado 3-Agosto-2021].
- IEA, IRENA, UNSD, World Bank & WHO. (2020). *Tracking SDG 7: The Energy Progress Report* (rel. técn.). International Renewable Energy Agency. abu Dhabi, World Bank, Washington, DC.
- Instituto Nacional de Pesquisas Espaciais. (2021). Sistema de Organização Nacional de Dados Ambientais - Estação Petrolina [Online; acessado 2021-07-19].
- Kaldellis, J. K. (2010). *Stand-alone and hybrid wind energy systems: technology, energy storage and applications*. Elsevier.

- Kusakana, K., & Vermaak, H. J. (2013). Hybrid renewable power systems for mobile telephony base stations in developing countries. *Renewable Energy*, 51, 419–425. <https://doi.org/10.1016/j.renene.2012.09.045>
- Liu, H., Mi, X.-w., & Li, Y.-f. (2018). Wind speed forecasting method based on deep learning strategy using empirical wavelet transform, long short term memory neural network and Elman neural network. *Energy Conversion and Management*, 156, 498–514. <https://doi.org/10.1016/j.enconman.2017.11.053>
- Manwell, J., Rogers, A., Hayman, G., Avelar, C., McGowan, J., Abdulwahid, U., & Wu, K. (2006). Hybrid2—a hybrid system simulation model—theory manual. *Renewable Energy Research Laboratory, University of Massachusetts*.
- NASA. (2012). NASA-NOAA Satellite Reveals New Views of Earth at Night [Online; acessado 2021-07-19].
- Nema, P., Nema, R., & Rangnekar, S. (2009). A current and future state of art development of hybrid energy system using wind and PV-solar: A review. *Renewable and Sustainable Energy Reviews*, 13(8), 2096–2103. <https://doi.org/10.1016/j.rser.2008.10.006>
- Operador Nacional do Sistema Elétrico. (2017). Sobre o SIN - Sistemas Isolados [Online; acessado 31-Janeiro-2021].
- Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6), 386.
- Tazvinga, H., Zhu, B., & Xia, X. (2014). Energy dispatch strategy for a photovoltaic–wind–diesel–battery hybrid power system. *Solar Energy*, 108, 412–420.
- Upadhyay, S., & Sharma, M. (2014). A review on configurations, control and sizing methodologies of hybrid energy systems. *Renewable and Sustainable Energy Reviews*, 38, 47–63. <https://doi.org/10.1016/j.rser.2014.05.057>
- Zhang, G., & Qi, M. (2005). Neural network forecasting for seasonal and trend time series. *European Journal of Operational Research*, 160(2), 501–514. <https://doi.org/10.1016/j.ejor.2003.08.037>

Apêndice A Rede Neural

```
1 import tensorflow as tf
2 import pandas as pd
3 import numpy as np
4
5 import time
6
7 from tensorflow.keras.preprocessing.sequence import TimeseriesGenerator
8 from sklearn.preprocessing import MinMaxScaler
9 from random import randint
10 from tensorflow.keras.callbacks import EarlyStopping
11
12 early_stopping = EarlyStopping(monitor='val_loss', patience=3, mode='min')
13 train_features = ['ws_10m', 'GHI']
14 hours_ahead = 6
15 window_length = int(hours_ahead * 60/10)
16 batch_size = 256
17 n_features = 1
18
19 results = {feat: {'history': [], 'model': []} for feat in train_features}
20 scalers = {feat: MinMaxScaler() for feat in train_features}
21
22
23 def generate_time_series(x_train, y_train, batch_size, window_length):
24     ts_train = TimeseriesGenerator(
25         data=x_train,
26         targets=y_train,
27         batch_size=batch_size,
28         length=window_length
29     )
30
31     ts_test = TimeseriesGenerator(
32         data=x_test,
33         targets=y_test,
34         batch_size=batch_size,
35         length=window_length
36     )
37
38     return ts_train, ts_test
39
40
41 def create_model():
42     model = tf.keras.Sequential([
43         tf.keras.layers.SimpleRNN(
44             64,
```

```

45         input_shape=(window_length , n_features) ,
46         return_sequences=True
47     ) ,
48     tf.keras.layers.Dropout(0.3) ,
49     tf.keras.layers.SimpleRNN(
50         32,
51         input_shape=(window_length , n_features) ,
52         return_sequences=False
53     ) ,
54     tf.keras.layers.Dropout(0.3) ,
55     tf.keras.layers.Dense(1, activation='sigmoid')
56   ])
57
58   model.compile(
59     loss=tf.losses.MeanSquaredError() ,
60     optimizer=tf.optimizers.Adam() ,
61     metrics=['mean_absolute_error']
62   )
63
64   return model
65
66
67 def create_prediction_dataframe(ts_test , y_test , window_length ,
68 timesteps , df):
69   test_pred = model.predict(ts_test)
70
71   y = scalers[feature_name].inverse_transform(y_test)
72   y_hat = scalers[feature_name].inverse_transform(test_pred)
73
74   lag = window_length + timesteps+1
75
76   return pd.DataFrame(
77     {'y': y[lag:, 0] , 'y_hat': y_hat[timesteps+1:, 0]} ,
78     index=df.index[int(0.75*len(df))+lag:])
79
80
81 def plot_sample(nn , ylabel="" , p=0.1):
82   n = int(len(nn) * p)
83
84   ini = randint(0, len(nn)-n)
85
86   nn.iloc[ini:ini+n].plot(ylabel=ylabel)
87
88
89 def last_rmse(results):
90   rmse = {}
91

```

```

92     n = 1
93     for feature, data in results.items():
94         rmse[feature] = []
95
96         last_rmse = None
97         for history in data['history']:
98             last_rmse = history['val_loss'].iloc[-1]
99             rmse[feature].append(last_rmse)
100
101    n = len(rmse[feature]) + 1
102
103    return pd.DataFrame(rmse, index=range(1, n))
104
105
106 def compare_prediction(features_results):
107
108     window_length = len(features_results['model'])
109     sample = features_results['model'][0]
110     length = len(sample)
111     start = randint(1, length - window_length)
112     start_pred = window_length + start
113
114     values = sample[start:start_pred].copy(deep=True)
115     values['y_hat'] = None
116
117     for nn in features_results['model']:
118         values = values.append(nn.iloc[start_pred])
119
120     return values
121
122
123 if __name__ == "__main__":
124
125     for feature_name in train_features:
126         print(f'feature_name:{feature_name}')
127
128     values = df[feature_name].values.reshape(-1, 1)
129     features = scalers[feature_name].fit_transform(values)
130
131     for timesteps in range(window_length):
132         i = time.time()
133         print(f'timestep:{timesteps}')
134
135         labels = np.roll(features, -timesteps)
136
137         x_train, x_test, y_train, y_test = train_test_split(
138             features,
139             labels,

```

```

140         test_size=0.25,
141         shuffle=False
142     )
143
144     ts_train, ts_test = generate_time_series(
145         x_train,
146         y_train,
147         batch_size,
148         window_length
149     )
150
151     model = create_model()
152
153     history = model.fit(
154         ts_train,
155         validation_data=ts_test,
156         callbacks=early_stopping,
157         epochs=100,
158         shuffle=False,
159         verbose=0
160     )
161
162     loss = history.history['val_loss']
163     rmse = loss[-1]
164     print(f'RMSE: {rmse}')
165     print(f'epochs: {len(loss)})')
166
167     history_df = pd.DataFrame(history.history)
168     nn = create_prediction_dataframe(
169         ts_test,
170         y_test,
171         window_length,
172         timesteps,
173         df
174     )
175
176     results[feature_name]['model'].append(nn.copy(deep=True))
177     results[feature_name]['history'].append(history_df.copy(deep=
178         True))
179
180     f = time.time()
181     print(f'{f-i:.3}s')
182     print('-----')
183     print()

```

Apêndice B Sistema Híbrido

```
1 import numpy as np
2
3
4 def log_wind_profile(wind_speed_2, height_1, height_2, surf_roughness=1):
5     :
6         return wind_speed_2 * (
7             (np.log(height_1) - np.log(surf_roughness)) /
8             (np.log(height_2) - np.log(surf_roughness)))
9     )
10
11
12 def wind_power(speed, curve=power_curve):
13     try:
14         return curve.loc[round(speed*2)/2]
15     except KeyError:
16         return curve.loc[round(speed*2+1)/2]
17
18
19 def solar_power(irradiation,
20                 modules,
21                 ambient_temp,
22                 NOCT_temp=45,
23                 ref_efficiency=0.1711,
24                 temp_coefficient=-0.003,
25                 ref_temp=25):
26
27     pv_area = modules * 1.63
28     cell_temp = ambient_temp + ((NOCT_temp - 20) * (irradiation / 800))
29     efficiency = ref_efficiency * (1 - temp_coefficient * (cell_temp -
30                                         ref_temp)))
31
32     return irradiation * pv_area * efficiency
33
34 def net_load(solar, wind, load):
35     return solar + wind - load
36
37
38 def lolp(net_load):
39     return (net_load < 0).sum() / len(net_load)
40
41
42 def load_following(net_load,
43                     soc_previous,
44                     diesel_timeout=False,
```

```

45             diesel_running=False ,
46             soc_max=SOC_MAX,
47             dod_max=DOD_MAX,
48             diesel_power=DIESEL_POWER,
49             diesel_min=DIESEL_MIN) :
50
51     soc_min = soc_max * (1 - dod_max)
52     diesel_fuel = 0
53     dump = 0
54     fuel = 0
55     soc = soc_previous
56
57     if diesel_timeout:
58         diesel_power = 0
59
60     if diesel_running:
61         net_load += diesel_power
62         fuel = diesel_power
63         diesel_power = 0
64
65     if soc_previous < soc_min or soc_previous > soc_max:
66         raise ValueError(f'{soc_min}<!{soc_previous}<!{soc_max}')
67
68     if net_load > 0:
69         diesel_fuel = 0
70         if net_load > soc_max - soc:
71             dump = net_load - (soc_max - soc)
72             soc = soc_max
73             net_load = 0
74         else:
75             soc += net_load
76             net_load = 0
77     else:
78         if -net_load > soc - soc_min:
79             net_load += (soc - soc_min)
80             soc = soc_min
81
82         if -net_load > diesel_min:
83             diesel_fuel = min(-net_load , diesel_power)
84             net_load += diesel_fuel
85         else:
86             soc = min(soc_min+diesel_min+net_load , soc_max)
87             diesel_fuel = diesel_min
88             net_load = 0
89     else:
90         soc += net_load
91         diesel_power = 0
92         net_load = 0

```

```

93
94     if diesel_running:
95         diesel_fuel = fuel
96
97     return net_load, soc, dump, diesel_fuel
98
99
100 def cycle_charging(net_load,
101                      soc_previous,
102                      diesel_timeout=False,
103                      diesel_running=False,
104                      soc_max=SOC_MAX,
105                      dod_max=DOD_MAX,
106                      diesel_power=DIESEL_POWER):
107
108     soc_min = soc_max * (1 - dod_max)
109     dump = 0
110     fuel = 0
111     diesel_fuel = 0
112     soc = soc_previous
113
114     if diesel_timeout:
115         diesel_power = 0
116
117     if diesel_running:
118         net_load += diesel_power
119         fuel = diesel_power
120         diesel_power = 0
121
122     if soc_previous < soc_min or soc_previous > soc_max:
123         raise ValueError(f'{soc_min}<!{soc_previous}<!{soc_max}')
124
125     if net_load > 0:
126         diesel_fuel = 0
127         if net_load > soc_max - soc:
128             dump = net_load - (soc_max - soc)
129             soc = soc_max
130             net_load = 0
131         else:
132             soc += net_load
133             net_load = 0
134     else:
135         if -net_load > soc - soc_min:
136             net_load += (soc - soc_min)
137             soc = soc_min
138
139         if diesel_power > -net_load:
140             diesel_power += net_load

```

```

141         diesel_fuel = -net_load
142         soc = min(diesel_power+soc, soc_max)
143         diesel_fuel += soc - soc_min
144         net_load = 0
145     else:
146         net_load += diesel_power
147         diesel_fuel = diesel_power
148
149     else:
150         soc += net_load
151         diesel_power = 0
152         net_load = 0
153
154 if diesel_running:
155     diesel_fuel = fuel
156
157 return net_load, soc, dump, diesel_fuel
158
159
160 def apply_dispatch(df,
161                     cc_soc_start,
162                     lf_soc_start,
163                     diesel_lag,
164                     soc_max=SOC_MAX,
165                     dod_max=DOD_MAX,
166                     diesel_power=DIESEL_POWER):
167
168     d = df.copy(deep=True)
169
170     d['lf_net_load'] = 0
171     d['cc_net_load'] = 0
172     d['lf_dump'] = 0
173     d['cc_dump'] = 0
174     d['lf_diesel'] = 0
175     d['cc_diesel'] = 0
176     d['lf_diesel_start'] = False
177     d['cc_diesel_start'] = False
178     d['lf_soc'] = lf_soc_start
179     d['cc_soc'] = cc_soc_start
180     d['lf_diesel_stop'] = True
181     d['cc_diesel_stop'] = True
182     lf_diesel_running = False
183     cc_diesel_running = False
184     lf_diesel_timeout = False
185     cc_diesel_timeout = False
186
187 for i in range(1, len(d)):
188     lf_soc_previous = d['lf_soc'].iloc[i-1]

```

```

189 cc_soc_previous = d[ 'cc_soc' ].iloc [ i-1 ]
190 lf_diesel_previous = d[ 'lf_diesel' ].iloc [ i-1 ]
191 cc_diesel_previous = d[ 'cc_diesel' ].iloc [ i-1 ]
192 net = d[ 'net_load' ].iloc [ i ]
193
194 if i > diesel_lag and diesel_lag > 0:
195     lf_diesel_running = (d[ 'lf_diesel_start' ].iloc [ i-diesel_lag : i ].sum() > 0)
196     cc_diesel_running = (d[ 'cc_diesel_start' ].iloc [ i-diesel_lag : i ].sum() > 0)
197     lf_diesel_timeout = (d[ 'lf_diesel_stop' ].iloc [ i-diesel_lag : i ].sum() > 0)
198     cc_diesel_timeout = (d[ 'cc_diesel_stop' ].iloc [ i-diesel_lag : i ].sum() > 0)
199
200 ( net_lf ,
201   soc_lf ,
202   dump_lf ,
203   diesel_power_lf ) = load_following(
204     net ,
205     lf_soc_previous ,
206     soc_max=soc_max ,
207     dod_max=dod_max ,
208     diesel_timeout=lf_diesel_timeout ,
209     diesel_running=lf_diesel_running ,
210     diesel_power=lf_diesel_previous if lf_diesel_running else
211         diesel_power
212 )
213
214 ( net_cc ,
215   soc_cc ,
216   dump_cc ,
217   diesel_power_cc ) = cycle_charging(
218     net ,
219     cc_soc_previous ,
220     soc_max=soc_max ,
221     dod_max=dod_max ,
222     diesel_timeout=cc_diesel_timeout ,
223     diesel_running=cc_diesel_running ,
224     diesel_power=cc_diesel_previous if cc_diesel_running else
225         diesel_power
226 )
227
228 d[ 'lf_diesel_stop' ].iloc [ i ] = (diesel_power_lf == 0 and
229                                         lf_diesel_running)
230 d[ 'cc_diesel_stop' ].iloc [ i ] = (diesel_power_cc == 0 and

```

```

        cc_diesel_running)
230    d[ 'lf_net_load' ]. iloc [ i ] = net_lf
231    d[ 'cc_net_load' ]. iloc [ i ] = net_cc
232    d[ 'lf_soc' ]. iloc [ i ] = soc_lf
233    d[ 'cc_soc' ]. iloc [ i ] = soc_cc
234    d[ 'lf_dump' ]. iloc [ i ] = dump_lf
235    d[ 'cc_dump' ]. iloc [ i ] = dump_cc
236    d[ 'lf_diesel_start' ]. iloc [ i ] = (diesel_power_lf > 0 and not
237                                lf_diesel_running)
237    d[ 'cc_diesel_start' ]. iloc [ i ] = (diesel_power_cc > 0 and not
238                                cc_diesel_running)
238    d[ 'lf_diesel' ]. iloc [ i ] = diesel_power_lf
239    d[ 'cc_diesel' ]. iloc [ i ] = diesel_power_cc
240
241 return d

```

Apêndice C Estratégias

```
1 import pandas as pd
2 import seaborn as sn
3 import sklearn as sk
4 import warnings
5
6 from datetime import timedelta
7 from matplotlib import pyplot as plt
8 from pandas.core.common import SettingWithCopyWarning
9 warnings.simplefilter(action="ignore", category=SettingWithCopyWarning)
10
11 timestep = timedelta(minutes=10)
12 start_pred = int(len(df)*0.75)+window_length+1
13
14 diesel_timeout = 3
15 confusion_matrix = []
16
17
18 for start in d[start_pred:-window_length].index:
19
20     window = d.loc[start:start + timestep * 35]
21
22     dispatch = apply_dispatch(
23         window,
24         d.loc[start][ 'cc_soc' ],
25         d.loc[start][ 'lf_soc' ],
26         diesel_lag=diesel_timeout
27     )
28     lolp_cc = lolp(dispatch[ 'cc_net_load' ])
29     lolp_lf = lolp(dispatch[ 'lf_net_load' ])
30
31     if lolp_cc < lolp_lf:
32         real = 'cc'
33     elif lolp_cc > lolp_lf:
34         real = 'lf'
35
36     lolp_cc_predicted, lolp_lf_predicted = best_predicted(
37         results,
38         start_pred,
39         d.loc[start][ 'cc_soc' ],
40         d.loc[start][ 'lf_soc' ],
41         diesel_timeout=diesel_timeout
42     )
43
44     if lolp_cc_predicted < lolp_lf_predicted:
45         predicted = 'cc'
46     elif lolp_cc_predicted > lolp_lf_predicted:
```

```

47     predicted = 'lf'
48
49     confusion_matrix.append((real, predicted))
50
51
52 confusion_matrix = pd.DataFrame(confusion_matrix)
53 confusion_matrix = sk.metrics.confusion_matrix(
54     confusion_matrix[0],
55     confusion_matrix[1]
56 )
57
58 df_cm = (
59     pd.DataFrame(
60         confusion_matrix,
61         index=['lf', 'cc'],
62         columns=['lf', 'cc']
63     )
64     /
65     confusion_matrix.sum()*100
66 )
67
68 ax = sn.heatmap(df_cm, annot=True, fmt='g')
69 plt.figure(figsize=(10, 7))
70 ax.set_title('Matrix de Confusao', y=1.08)
71 ax.set_ylabel('Real')
72 ax.set_xlabel('Previsto')
73 ax.xaxis.set_ticks_position('top')
74 ax.xaxis.set_label_position('top')

```