Previsão de Temperatura: Uma Abordagem Baseada em Redes Neurais

Gabriel Antonio da Silva[1]

[1]FIEG – SENAI
Casa da Indústria - Vila Nova – 74.645-070 – Goiânia – GO – Brazil

Abstract. *This work presents the implementation of a Multi-Layer Perceptron (MLP) for predicting the next day's hourly temperature. The model uses historical meteorological data from Goiânia - Goiás, Brazil, provided by INMET. The methodology includes preprocessing, feature engineering, and a neural architecture with dense hidden layers and ReLU activation. The results show that this MLP effectively captures short-term temperature trends, demonstrating its suitability for time series forecasting tasks.*

Resumo. *Este trabalho apresenta a implementação de um Multi-Layer Perceptron (MLP) para previsão da temperatura horária do dia seguinte. O modelo utiliza dados meteorológicos históricos de Goiânia - Goiás, fornecidos pelo INMET. A metodologia abrange preprocessamento, criação de features, além de uma arquitetura neural composta por camadas densas e função de ativação ReLU. Os resultados mostram que o MLP captura eficazmente tendências de temperatura de curto prazo, comprovando sua aplicabilidade em séries temporais.*

1. Introdução

A previsão de temperatura é uma tarefa de grande relevância científica e social, com impacto significativo em diversas áreas, como agricultura, saúde pública, planejamento urbano, transportes e gestão de recursos energéticos. Decisões estratégicas em setores como irrigação agrícola, prevenção de doenças relacionadas ao clima e balanço energético dependem de previsões precisas de temperatura. Além disso, o aquecimento global e as mudanças climáticas intensificam a necessidade de ferramentas mais eficazes para prever padrões de temperatura em diferentes escalas temporais e geográficas.

Este trabalho foca em *Short Term Forecasting* (previsão de curto prazo) [Marino et al. 2017], utilizando um *Multi-Layer Perceptron* (MLP) [Hornik et al. 1989], uma arquitetura de rede neural artificial, para prever a temperatura horária das próximas 24 horas com base em dados meteorológicos históricos. O *Short Term Forecasting* é especialmente desafiador devido à limitação de dados históricos relevantes e à alta variabilidade de fenômenos atmosféricos em períodos curtos, exigindo modelos capazes de capturar relações complexas entre variáveis.

A escolha do MLP é motivada por sua capacidade de modelar relações não-lineares e interdependências entre variáveis meteorológicas, como pressão atmosférica, umidade relativa, velocidade do vento e radiação solar. Este estudo busca explorar a eficácia do uso de redes neurais para melhorar a precisão de previsões de curto prazo, considerando a complexidade inerente das variáveis e a natureza dinâmica das séries temporais.

O objetivo principal é avaliar o desempenho da abordagem proposta em cenários de previsão de curto prazo, contribuindo para o avanço do uso de redes neurais em problemas de previsão climática. Esta pesquisa também destaca os desafios e limitações dessa abordagem, fornecendo *insights* para estudos futuros no domínio de *Short Term Forecasting*.

2. Metodologia

A metodologia proposta neste trabalho foi estruturada em um pipeline de aprendizado de máquina, composto por etapas sequenciais que envolvem desde o pré-processamento dos dados até a avaliação do modelo e a realização de previsões, conforme descrito nas subseções seguintes.

2.1. Carregamento e Pré-processamento de Dados

Os dados meteorológicos utilizados neste estudo foram extraídos de um arquivo CSV coletado nos dados históricos do INMET [Instituto Nacional de Meteorologia (INMET) 2024], contendo registros climáticos da cidade de Goiânia, que abrangem o período de 1º de janeiro de 2015 a 29 de agosto de 2024. A planilha de dados possui as seguintes colunas:

- Data
- Hora UTC
- PRECIPITAÇÃO TOTAL, HORÁRIO (mm)
- PRESSÃO ATMOSFÉRICA AO NÍVEL DA ESTAÇÃO, HORÁRIA (mB)
- PRESSÃO ATMOSFÉRICA MAX. NA HORA ANTERIOR (AUT) (mB)
- PRESSÃO ATMOSFÉRICA MIN. NA HORA ANTERIOR (AUT) (mB)
- RADIAÇÃO GLOBAL (Kj/m²)
- TEMPERATURA DO AR - BULBO SECO, HORÁRIA (°C)
- TEMPERATURA DO PONTO DE ORVALHO (°C)
- TEMPERATURA MÁXIMA NA HORA ANTERIOR (AUT) (°C)
- TEMPERATURA MÍNIMA NA HORA ANTERIOR (AUT) (°C)
- TEMPERATURA ORVALHO MÁX. NA HORA ANTERIOR (AUT) (°C)
- TEMPERATURA ORVALHO MÍN. NA HORA ANTERIOR (AUT) (°C)
- UMIDADE REL. MÁX. NA HORA ANTERIOR (AUT) (%)
- UMIDADE REL. MÍN. NA HORA ANTERIOR (AUT) (%)
- UMIDADE RELATIVA DO AR, HORÁRIA (%)
- VENTO, DIREÇÃO HORÁRIA (gr) (°)
- VENTO, RAJADA MÁXIMA (m/s)
- VENTO, VELOCIDADE HORÁRIA (m/s)

A variável alvo para o modelo é a temperatura do ar (*TEMPERATURA DO AR - BULBO SECO, HORÁRIA (°C)*), e o objetivo final é prever a temperatura horária para o próximo intervalo de 24 horas, configurando um *Short Term Forecast*.

A função *load_data* foi responsável por carregar e organizar as informações em um formato estruturado. O pré-processamento envolveu as seguintes etapas:

- **Seleção de Colunas:** Foram selecionadas 17 colunas como *features*, incluindo medições de temperatura, pressão atmosférica, precipitação e outros fatores climáticos.

- **Tratamento de Valores Faltantes:** Entradas com valores ausentes foram removidas para garantir a consistência dos dados.
- **Separação de *Features* e *Target*:** As colunas selecionadas foram armazenadas em uma matriz `X`, enquanto a variável *target* foi armazenada em *y*, representando a temperatura horária.
- **Normalização:** As *features* foram normalizadas utilizando a classe *StandardScaler* da biblioteca *scikit-learn*.

2.2. Modelo e Camadas

O modelo utilizado neste trabalho é um *Multi-Layer Perceptron* (MLP) [Hornik et al. 1989], implementado em *PyTorch* [Paszke et al. 2019]. A arquitetura do modelo implementado é definida a seguir:

- **Camadas Fully Connected (FC):** O modelo é composto por cinco camadas densamente conectadas, configuradas da seguinte forma:
  - *fc1:* Mapeia o número de entradas para 128 neurônios.
  - *fc2:* Reduz de 128 para 64 neurônios (saída).
  - *fc3:* Reduz de 64 para 32 neurônios.
  - *fc4:* Reduz de 32 para 16 neurônios.
  - *fc5:* Reduz de 16 para 1 neurônio (saída).
- **Função de Ativação:** A função ReLU (*Rectified Linear Unit*) [Nair and Hinton 2010] é usada após cada camada oculta para introduzir não-linearidade.
- **Regularização:** A técnica de *dropout* [Srivastava et al. 2014] é empregada com uma taxa de 20% nas duas primeiras camadas para reduzir o sobreajuste (*overfitting*).
- **Propagação Direta:** O método *forward* [Srivastava et al. 2014] implementa a propagação dos dados de entrada pelas camadas do modelo.

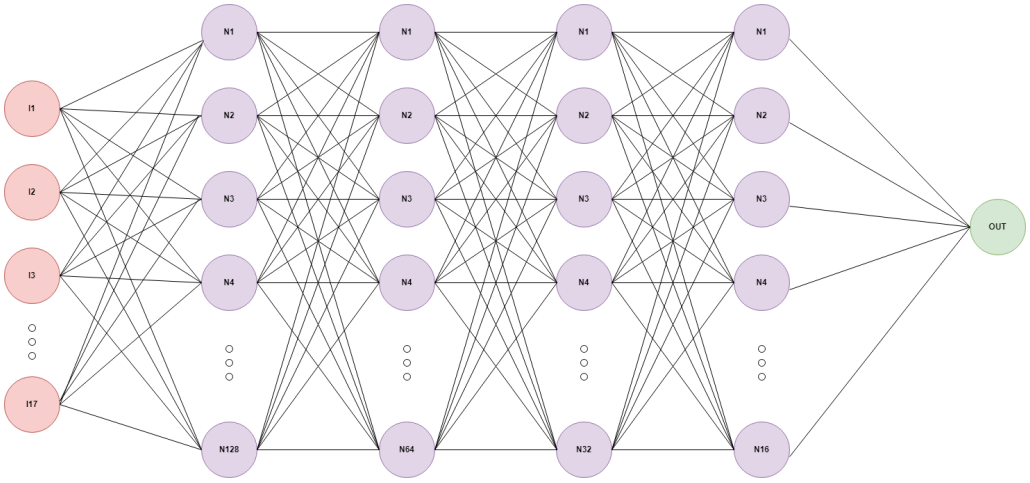

Figure 1. Estrutura do Modelo MLP

2.2.1. ReLU (*Rectified Linear Unit*)

Uma função de ativação é um componente fundamental em redes neurais que introduz não-linearidade no modelo, permitindo que ele aprenda padrões complexos nos dados. Ela transforma a saída linear de um neurônio em uma forma não-linear, essencial para resolver problemas não-lineares. Matematicamente, a função de ativação $f(x)$ é aplicada ao somatório ponderado das entradas de um neurônio ($W \cdot x + b$), gerando a saída final y. Sem uma função de ativação, a rede seria equivalente a um modelo linear, incapaz de resolver problemas complexos.

A função de ativação ReLU [Nair and Hinton 2010] é uma das mais populares em redes neurais, especialmente em modelos de aprendizado profundo. Ela é definida como:

$$\text{ReLU}(x) = \max(0, x)$$

A função ReLU foi escolhida pela sua capacidade de superar os problemas associados a funções de ativação como a sigmoide e a tangente hiperbólica, que podem levar ao *vanishing gradient problem*, onde os gradientes se tornam extremamente pequenos durante o retropropagação, retardando o aprendizado. Essa função, por ser linear para valores positivos, mantém gradientes mais consistentes e facilita a propagação de informações através da rede.

Além disso, a função ReLU é computacionalmente eficiente, pois envolve apenas uma comparação simples, o que a torna vantajosa em termos de tempo de processamento.

2.2.2. Regularização e *Dropout*

A regularização é uma técnica utilizada para prevenir o sobreajuste (*overfitting*) em modelos de aprendizado de máquina, onde o modelo se ajusta tão bem aos dados de treinamento que perde a capacidade de generalizar para novos dados.

Neste trabalho, foi utilizada a técnica de *dropout* [Srivastava et al. 2014], que consiste em desativar aleatoriamente uma fração dos neurônios durante o treinamento. A formulação matemática para o *dropout* é:

$$y = \frac{x \cdot m}{p}$$

Onde:

- x é a saída original dos neurônios.
- m é uma máscara binária gerada aleatoriamente.
- p é a taxa de probabilidade de manter os neurônios ativos.

A utilização do *dropout* oferece as seguintes vantagens:

- **Redução de *overfitting*:** Evita que o modelo dependa excessivamente de combinações específicas de neurônios.

- **Melhoria do aprendizado:** Força o modelo a aprender representações mais generalizáveis.

Foi aplicada uma taxa de *dropout* de 20% nas duas primeiras camadas, promovendo maior diversidade nas representações aprendidas.

2.2.3. Propagação Direta

A propagação direta (*forward propagation*) [Hornik et al. 1989] é o processo pelo qual os dados de entrada percorrem as camadas da rede neural, desde a entrada até a saída. O objetivo é calcular a predição do modelo com base nos pesos e *biases* aprendidos. A formulação geral é:

$$y = f(W \cdot x + b)$$

Onde:

- W são os pesos da camada.
- x é o vetor de entrada.
- b é o vetor de *bias*.
- f é a função de ativação.

A propagação direta é essencial por diversas razões:

- **Cálculo da perda:** Permite calcular a diferença entre a saída prevista e o valor esperado.
- **Base para o treinamento:** Os resultados obtidos na propagação direta são usados na retropropagação para ajustar os pesos e *biases*.
- **Eficiência:** As operações matriciais realizadas na propagação direta são otimizadas por *frameworks* como *PyTorch*, tornando o processo rápido e escalável.

Neste modelo, a propagação direta foi implementada no método *forward*, garantindo a passagem sequencial das entradas pelas camadas da rede.

2.3. Treinamento do Modelo

O treinamento do modelo foi realizado com os seguintes componentes:

- **Função de Perda:** A *MSELoss* (Erro Médio Quadrático) [Paszke et al. 2017] foi utilizada para calcular a discrepância entre as previsões e os valores reais.
- **Otimizador:** O algoritmo Adam foi empregado para ajustar os pesos do modelo, com taxa de aprendizado configurada para 0.01.
- **Épocas de Treinamento:** O treinamento foi conduzido por 100 épocas, com ajustes de pesos realizados a cada iteração.
- **Divisão de Dados:** O conjunto de dados foi dividido em 80% para treinamento e 20% para teste.

2.3.1. Função de Perda

A função de perda é um componente essencial em modelos de aprendizado de máquina. Ela mede a discrepância entre as previsões do modelo e os valores reais, fornecendo um sinal que orienta o ajuste dos pesos da rede.

A função de perda escolhida foi a *MSELoss* (Erro Médio Quadrático) [Paszke et al. 2017], definida como:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

onde y_i são os valores reais e $\hat{y}_i$ as previsões do modelo. O *MSELoss* foi selecionado por sua simplicidade e eficácia em capturar diferenças absolutas entre previsões e valores reais, especialmente em problemas de regressão como o abordado neste trabalho.

Além disso, a MSE atribui maior penalidade a erros maiores, incentivando o modelo a minimizar discrepâncias significativas. Esse comportamento ajuda a obter previsões mais precisas em cenários onde pequenas variações podem ter impacto relevante.

2.3.2. Otimizador

O otimizador é responsável por ajustar os pesos da rede com base nos gradientes calculados durante a retropropagação, minimizando a função de perda.

Neste trabalho, utilizamos o otimizador *Adam* (*Adaptive Moment Estimation*) [Kingma and Ba 2014], que combina as vantagens do SGD (*Stochastic Gradient Descent*) com adaptações automáticas de taxas de aprendizado. O *Adam* calcula taxas de aprendizado adaptativas para cada parâmetro, utilizando momentos de primeira ordem (média) e segunda ordem (variância) dos gradientes. Sua fórmula é dada por:

$$\theta_{t+1} = \theta_t - \eta\frac{\hat{m_t}}{\sqrt{\hat{v_t}} + \epsilon}$$

onde θ_t representa os pesos, η é a taxa de aprendizado, $\hat{m_t}$ e $\hat{v_t}$ são as médias ajustadas e ϵ é um valor pequeno para evitar divisão por zero.

O *Adam* foi escolhido por sua eficiência em problemas com grande quantidade de dados e parâmetros. Com uma taxa de aprendizado de 0.01, ele garante uma convergência estável e relativamente rápida.

2.3.3. Épocas de Treinamento

Uma época de treinamento consiste em uma passagem completa por todo o conjunto de dados de treinamento. Durante uma época, o modelo ajusta seus pesos para minimizar a função de perda.

A escolha de 100 épocas reflete um equilíbrio entre tempo de treinamento e a possibilidade de capturar padrões nos dados sem sobreajuste (*overfitting*). Testes preliminares indicaram que 100 épocas eram suficientes para a convergência do modelo, garantindo que ele aprenda sem superestimar ruídos nos dados.

2.3.4. Divisão de Dados

A divisão dos dados é fundamental para avaliar a capacidade do modelo de generalizar para dados novos. O conjunto de dados foi dividido em:

- 80% para treinamento: Utilizado para ajustar os pesos do modelo.
- 20% para teste: Reservado para avaliar a performance do modelo em dados não vistos.

Essa proporção é amplamente utilizada por equilibrar a quantidade de dados para aprendizado e validação. A parte de teste permite estimar o desempenho do modelo em cenários reais, evitando que ele se adapte apenas aos dados de treinamento. Divisões mal equilibradas podem levar a um modelo subtreinado (poucos dados de treinamento) ou avaliações imprecisas (poucos dados de teste).

2.4. Avaliação e Predição

A avaliação do modelo foi realizada utilizando o conjunto de teste, com o cálculo do Erro Médio Quadrático (MSELoss) [Paszke et al. 2017]. Duas funções foram implementadas para predizer a temperatura:

- *predict_day_after_final_data:* Realiza previsões para cada hora do dia seguinte ao final do conjunto de dados. Utilizado principalmente para testes com o modelo após seu treinamento.
- *predict_next_day:* Prevê a temperatura horária para o próximo dia, considerando a data e hora atuais. Esse é o objetivo principal do modelo, prever a temperatura horária das 24 horas seguintes.

Os resultados das previsões foram apresentados com precisão decimal e comparados com os valores reais registrados no conjunto de teste.

3. Resultados

Os resultados mostram que o MLP [Hornik et al. 1989] captura padrões diários e sazonais da temperatura com bom desempenho preditivo. A análise gráfica confirma a proximidade entre previsões e valores reais, destacando a capacidade do modelo de lidar com séries temporais curtas. A Tabela 1 apresenta uma comparação entre as previsões e os valores reais dos últimos dados válidos no conjunto de dados. Nota-se que o modelo chegou bem perto do valor real da temperatura horária no período escolhido e possui um Erro Médio Quadrático (MSE) [Paszke et al. 2017] de 1.3169.

O segundo conjunto de dados gerados pelo modelo é uma previsão baseada no último dia com dados válidos no conjunto de dados, apontado pelas colunas *Data*, *Hora* e *Previsões* da Tabela 2.

Data	Hora	Previsões (°C)	Reais (°C)
28/08/2024	10:00	17.22	17.90
28/08/2024	11:00	19.63	22.30
28/08/2024	12:00	25.07	27.10
28/08/2024	13:00	28.74	29.20
28/08/2024	14:00	30.74	31.20
28/08/2024	15:00	32.08	32.30
28/08/2024	16:00	32.21	32.70
28/08/2024	17:00	32.36	33.50
28/08/2024	18:00	32.37	33.50
28/08/2024	19:00	31.59	32.60
28/08/2024	20:00	30.20	31.40
28/08/2024	21:00	28.13	29.10
29/08/2024	10:00	20.90	21.30
29/08/2024	11:00	22.61	23.50
29/08/2024	12:00	25.14	25.60
29/08/2024	13:00	27.56	28.00
29/08/2024	14:00	29.39	28.90
29/08/2024	15:00	30.85	30.70
29/08/2024	16:00	30.98	31.20
29/08/2024	17:00	31.78	32.30
29/08/2024	18:00	31.61	31.40
29/08/2024	19:00	31.20	32.00
29/08/2024	20:00	29.64	30.40
29/08/2024	21:00	27.78	28.70

Table 1. Previsões e Valores Reais de Temperatura

O modelo utiliza os dados do dia anterior para fazer a previsão do próximo dia, a última entrada é de 29/08/2024, nota-se que a previsão para o dia 30/08/2024 é bem próxima ao seu dia anterior. No entanto, a coluna *Reais* da Tabela 2 indica uma diferença clara entre a previsão para o dia 30/08/2024 e sua temperatura horária real. Esse resultado configura uma limitação do modelo.

Data	Hora	Previsões (°C)	Reais (°C)
30/08/2024	00:00	17.22	24.8
30/08/2024	01:00	19.63	24.5
30/08/2024	02:00	25.07	23.5
30/08/2024	03:00	28.74	24.5
30/08/2024	04:00	30.74	21.4
30/08/2024	05:00	32.08	20.1
30/08/2024	06:00	32.21	19.5
30/08/2024	07:00	32.36	18.2
30/08/2024	08:00	32.37	17.5
30/08/2024	09:00	31.59	16.7
30/08/2024	10:00	30.20	17.6
30/08/2024	11:00	28.13	23.3
30/08/2024	12:00	20.90	25.7
30/08/2024	13:00	22.61	27.9
30/08/2024	14:00	25.14	29.0
30/08/2024	15:00	27.56	29.7
30/08/2024	16:00	29.39	31.1
30/08/2024	17:00	30.85	31.5
30/08/2024	18:00	30.98	32.7
30/08/2024	19:00	31.78	32.3
30/08/2024	20:00	31.61	31.0
30/08/2024	21:00	31.20	28.8
30/08/2024	22:00	29.64	26.0
30/08/2024	23:00	27.78	24.8

Table 2. Previsões e Dados Reais para o Dia 30/08/2024

O último resultado apresentado pelo modelo é a previsão das 24 horas seguintes, como apresentado na Tabela 3, os valores apresentados correspondem à temperatura

horária do dia seguinte à execução do modelo. Vale destacar que os dados utilizados se encerram em Agosto/2024, o que pode influenciar na precisão da previsão de curto prazo, que gera outro ponto importante a ser discutido.

Data	Hora	Temperatura (°C)
30/11/2024	20:36	20.57
30/11/2024	21:36	24.47
30/11/2024	22:36	26.03
30/11/2024	23:36	28.02
01/12/2024	00:36	30.32
01/12/2024	01:36	30.63
01/12/2024	02:36	29.31
01/12/2024	03:36	27.28
01/12/2024	04:36	28.74
01/12/2024	05:36	30.39
01/12/2024	06:36	25.84
01/12/2024	07:36	25.11
01/12/2024	08:36	24.28
01/12/2024	09:36	21.44
01/12/2024	10:36	21.66
01/12/2024	11:36	22.00
01/12/2024	12:36	22.90
01/12/2024	13:36	22.30
01/12/2024	14:36	21.11
01/12/2024	15:36	20.78
01/12/2024	16:36	22.01
01/12/2024	17:36	23.65
01/12/2024	18:36	24.60
01/12/2024	19:36	25.80

Table 3. Previsões de Temperatura para o Dia 30/11/2024 e 01/12/2024

4. Conclusão

Este trabalho demonstrou a viabilidade e a eficácia do uso de um *Multi-Layer Perceptron* (MLP) [Hornik et al. 1989] na previsão de temperatura diária em cenários de curto prazo, utilizando dados meteorológicos públicos. O modelo apresentou resultados promissores ao capturar relações não-lineares entre variáveis climáticas, reforçando o potencial das redes neurais para problemas de *Short Term Forecasting*.

Uma das principais vantagens do MLP está na sua simplicidade de implementação e flexibilidade em modelar interações complexas entre variáveis. Além disso, o MLP é uma solução viável devido ao seu desempenho computacional em comparação com modelos mais sofisticados, tornando-o adequado para aplicações onde recursos computacionais são limitados.

No entanto, alguns desafios foram identificados. O modelo demonstrou uma relativa imprecisão, possivelmente devido à limitação dos dados utilizados. Para previsão imediata utiliza-se os dados do dia anterior como base, o modelo com seus parâmetros atuais não conseguiu prever dias atípicos com exatidão, que foi o caso do dia 29/08/2024 em que tivemos maiores flutuações de temperatura na cidade de Goiânia. Nesse ponto, o modelo apresentou para o dia 30/08/2024 valores próximos do dia anterior, que foi diferente da temperatura horária real.

Outra questão a ser considerada foi a previsão das próximas 24 horas, observando que utilizamos dados que acabaram em 29/08/2024 e utilizamos o modelo treinado com esses dados para prever 30/11/2024. Isso foge da previsão de curto prazo mas é uma questão a ser observada, se houver quantidade limitada de dados, seja por imprecisão das informações ou ausência de coleta, o modelo vai ter dificuldades de fazer previsões com variação aceitável, mesmo que tenha muitos dados históricos à disposição.

Além disso, a definição da estrutura da rede e a escolha dos parâmetros de treinamento ainda representam uma dificuldade significativa, dado que esses aspectos não podem ser facilmente otimizados em uma pesquisa de pequeno porte.

Como trabalhos futuros, propõe-se:

- Inclusão de variáveis climáticas adicionais, como índices de vegetação, dados geográficos e outros fatores sazonais, para enriquecer o conjunto de dados e reduzir os riscos de *overfitting*;
- Uso de técnicas de *ensemble* [Dietterich 2000], combinando diferentes modelos para aumentar a precisão das previsões;
- Comparação com modelos mais avançados, como LSTMs [Hochreiter and Schmidhuber 1997] e *Transformer-based architectures* [Vaswani et al. 2017], que podem capturar dependências temporais de forma mais eficaz;
- Realização de estudos aprofundados para melhorar a arquitetura do MLP, incluindo experimentos com camadas adicionais, funções de ativação alternativas e métodos de regularização mais sofisticados;
- Exploração de técnicas automatizadas de busca de hiperparâmetros, como *grid search* [Och 2003] ou *Bayesian optimization* [Snoek et al. 2012], para obter configurações de treinamento mais adequadas.

Apesar das limitações identificadas, o MLP mostrou-se uma solução viável, de fácil implementação e com grande potencial para previsão de curto prazo, especialmente em contextos os recursos computacionais são restritos. Este estudo contribui para o avanço das aplicações de redes neurais em previsão climática e colabora para investigações futuras neste campo desafiador.

References

Dietterich, T. G. (2000). Ensemble methods in machine learning. *Multiple Classifier Systems*, 1857:1–15.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Instituto Nacional de Meteorologia (INMET) (2024). Dados históricos. `https://portal.inmet.gov.br/dadoshistoricos`. Accessed: 2024-11-26.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Marino, D. L., Amarasinghe, K., and Manic, M. (2017). Short-term load forecasting with deep neural networks. *IEEE Transactions on Industrial Informatics*, 13(1):311–320.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. pages 807–814.

Och, F. J. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Paszke, A., Gross, S., Lerer, A., Chintala, S., et al. (2017). *Deep Learning with PyTorch: A 60 Minute Blitz*. Available at https://pytorch.org/.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. 25.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. 30.