Classificação com MLP

Gabriel Anselmo Ramos[1]

[1]Centro de Ciências Tecnológicas – Universidade do Estado de Santa Catarina (UDESC)
R. Paulo Malschitzki, 200 Zona Industrial Norte – 89219-710 – Joinville – SC – Brazil

[2]Departamento de Ciência da Computação
Universidade do Estado de Santa Catarina (UDESC) – Joinville, SC – Brazil

{anselmo,gabriel}421@gmail.com,anselmogabriel421@gmail.com

Abstract. *This work presents an alternative to classical methods, proposing a model based on an Artificial Neural Network (ANN) for classifying and identifying breast cancer. Artificial Neural Networks have become widely disseminated, standing out for their remarkable adaptability. By opting for an ANN-based model, there is the possibility of training the network for accurate disease detection. Therefore, by employing an ANN model based on a Multi Layer Perceptron (MLP), which is specialized in data classification, we can conclude that data segmentation is a viable task using this approach, although there are limitations. This study explores the implementation and training of an MLP-type neural network for data classification, using the TensorFlow and scikit-learn libraries. The dataset chosen for this analysis is Breast Cancer Wisconsin (Original).*

Resumo. *Este trabalho apresenta uma alternativa aos métodos clássicos, propondo um modelo baseado em Rede Neural Artificial (RNA) para classificação e identificação de câncer de mama. As Redes Neurais Artificiais têm se tornado amplamente difundidas, destacando-se pela sua notável adaptabilidade. Ao optar por um modelo baseado em RNA, há a possibilidade de treinar a rede para a detecção precisa da doença. Dessa forma, ao empregar um modelo de RNA fundamentado em um Multi Layer Perceptron (MLP), que é especializado em classificação de dados, podemos concluir que a segmentação dos dados é uma tarefa viável por meio desse enfoque, embora haja limitações. Este estudo explora a implementação e treinamento de uma rede neural do tipo MLP para a classificação de dados, utilizando as bibliotecas TensorFlow e scikit-learn. O conjunto de dados escolhido para essa análise é o Breast Cancer Wisconsin (Original).*

1. Introdução

O treinamento de redes neurais utilizando TensorFlow representa uma abordagem notavelmente poderosa e eficaz no âmbito da inteligência artificial (IA). Desenvolvido pela Google, o TensorFlow é uma biblioteca de código aberto que oferece uma estrutura robusta para a criação e treinamento de modelos de aprendizado de máquina. Seu destaque é evidente em tarefas complexas, incluindo, mas não se limitando a, reconhecimento de padrões, processamento de linguagem natural e visão computacional. Este texto visa desbravar os conceitos fundamentais subjacentes ao treinamento de redes neurais usando TensorFlow, sendo direcionado também aos leitores com pouca ou nenhuma experiência prévia no campo da inteligência artificial.[Elesbão et al. 2023]

1.1. Entendendo o Treinamento de Redes Neurais

Desenvolver um modelo baseado em Rede Neural Artificial (RNA) utilizando TensorFlow com o objetivo de detectar padrões indicativos de câncer de mama. Em que sua essência, a rede neural representa um modelo inspirado do cérebro humano, composto por camadas interconectadas de neurônios. Esses neurônios, também chamados de nós, processam informações e aprendem padrões à medida que são expostos a conjuntos de dados. O processo de treinamento de uma rede neural no TensorFlow implica na adaptação dos pesos dessas conexões, permitindo que o modelo execute tarefas específicas com maior precisão.

1.2. TensorFlow: A Ferramenta Facilitadora

TensorFlow simplifica a criação e treinamento de redes neurais, fornecendo uma abstração eficiente para manipulação de tensores, que são essencialmente arranjos multidimensionais de dados. Com o TensorFlow, os desenvolvedores podem construir modelos de aprendizado profundo de forma flexível, seja para classificar imagens, traduzir idiomas ou tomar decisões complexas e permite implementar facilmente o algoritmos com TensorFlow [Bagby et al. 2018]

- Os Componentes do Treinamento: uma rede neural no TensorFlow envolve vários componentes essenciais. Primeiro, há a definição da arquitetura da rede, especificando o número de camadas, o tipo de neurônios e as conexões entre eles. Em seguida, os dados de treinamento são alimentados ao modelo, permitindo que ele ajuste seus pesos durante várias iterações ou ”épocas”. Um otimizador, como o Adam, ajuda a ajustar os pesos para minimizar a diferença entre as previsões do modelo e os rótulos reais dos dados de treinamento.
- Ajuste Fino: o sucesso de um modelo de rede neural muitas vezes depende do ajuste fino de vários hiperparâmetros durante o treinamento. A taxa de aprendizado, o tamanho do lote e o número de épocas são exemplos de hiperparâmetros que influenciam diretamente na eficácia do modelo. Encontrar a combinação certa pode exigir experimentação e compreensão detalhada do problema em questão.
- Avaliação e Visualização: após o treinamento, é crucial avaliar o desempenho do modelo em dados não vistos. Métricas como precisão, matriz de confusão e características da curva ROC ajudam a quantificar a capacidade do modelo de generalizar para novos dados. O TensorFlow facilita a visualização dessas métricas, permitindo que os desenvolvedores entendam como o modelo está performando e identifiquem áreas de melhoria, como foi feito neste trabalho.

O treinamento de redes neurais com TensorFlow ofereceu uma abordagem acessível para resolver problemas complexos de aprendizado de máquina. [Verma et al. 2021] Esta apresentação fornece uma visão geral para leitores iniciantes, destacando os principais conceitos e passos envolvidos no treinamento de modelos. À medida que a inteligência artificial continua a evoluir, compreender o funcionamento interno dessas redes neurais torna-se uma habilidade valiosa para quem busca explorar as fronteiras desse campo dinâmico e promissor. [Rosa et al. 2022]

Figure 1. Tecnologias usadas

2. Metodologia de Desenvolvimento

Iremos carregar o conjunto de dados, separar os conjuntos de treinamento e teste, definir a arquitetura da rede neural usando TensorFlow, treinar o modelo com o algoritmo de otimização Adam (usado para calcular os pesos e que trabalha em conjunto com o backpropagation visto em sala) e por fim avaliar o modelo.

No decorrer do processo, realizaremos diversas etapas essenciais para a construção e avaliação do modelo. Primeiramente, faremos a carga do conjunto de dados, uma etapa crucial para garantir que o modelo seja alimentado com informações relevantes. Em seguida, procederemos à divisão dos conjuntos de treinamento e teste, uma prática padrão para avaliar o desempenho do modelo em dados que não foram utilizados durante o treinamento. E para isso foi incorporada uma etapa essencial: a Normalização dos Dados, prática que desempenha um papel significativo na garantia de consistência nos resultados do modelo. A normalização é um processo pelo qual os dados são ajustados para que possuam uma escala uniforme, evitando disparidades que podem surgir devido às diferentes magnitudes das características.

Ao normalizar os dados, asseguramos que todas as variáveis contribuam de maneira equitativa para o aprendizado do modelo, proporcionando estabilidade e coerência ao processo de treinamento. Essa prática é particularmente relevante em modelos baseados em redes neurais, onde a escala dos dados pode impactar diretamente o desempenho do modelo. Portanto, no decorrer do nosso processo, antes da separação dos conjuntos de treinamento e teste para definir a arquitetura da rede neural, incorporaremos a etapa de Normalização dos Dados no qual contribuirá para a robustez e generalização do modelo, preparando-o para realizar previsões precisas e confiáveis.

A fase subsequente envolverá a definição da arquitetura da rede neural utilizando TensorFlow, onde determinaremos a disposição das camadas, o número de neurônios em cada camada e as funções de ativação. Esta etapa é fundamental para configurar a estrutura do modelo, garantindo que ele seja capaz de aprender padrões de maneira eficaz. O treinamento propriamente dito será conduzido através do algoritmo de otimização Adam. Esse algoritmo desempenha um papel crucial no cálculo dos pesos

da rede neural, colaborando em conjunto com o processo de retropropagação (backpropagation), que é uma técnica vital de ajuste dos pesos durante o treinamento, conforme discutido em sala e a taxa de aprendizado definida como 0.01.

Por fim, avaliaremos o desempenho de dois modelos. Essa fase incluirá a análise de métricas específicas, como precisão e recall, para compreender quão bem o modelo está realizando as previsões, especialmente em dados não utilizados anteriormente. Um procedimento avaliativo fundamental para determinar a eficácia e a confiabilidade do modelo desenvolvido. No qual ambos usam a função de ativação 'relu' para as camadas ocultas, a função de ativação 'sigmoid' na camada de saída, indicando uma tarefa de classificação binária e o otimizador Adam com a mesma taxa de aprendizado.

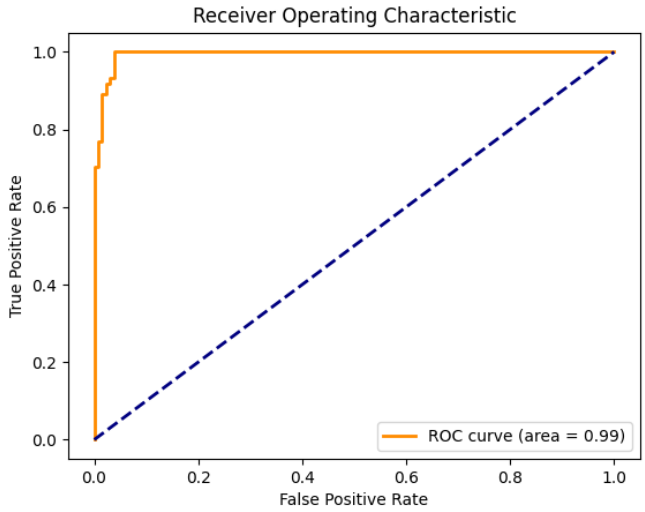

Figure 2. Características operacionais do receptor - Modelo 1

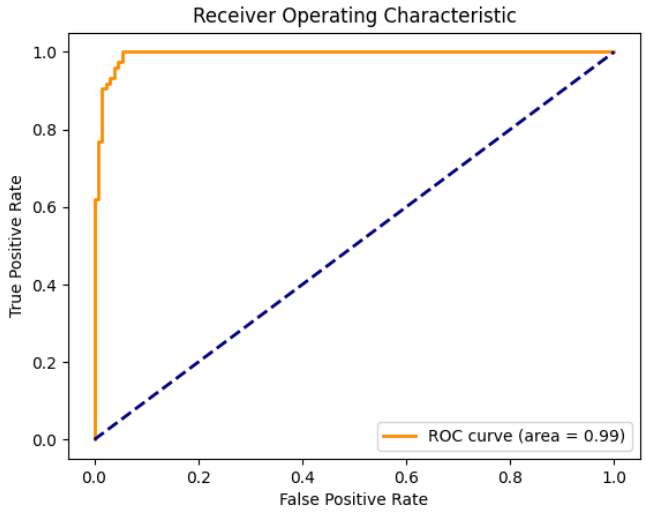

Figure 3. Características operacionais do receptor - Modelo 2

- **Semelhanças:**
    1. Ambos os modelos utilizam a mesma função de perda (binary_crossentropy) e otimizador (Adam) com a mesma taxa de aprendizado.
    2. Ambos os modelos utilizam ativação 'relu' para as camadas ocultas e 'sigmoid' para a camada de saída.
- **Diferenças:**
    1. O segundo modelo possui mais camadas (três em comparação com duas do primeiro modelo).
    2. Utiliza dropout para mitigar overfitting, enquanto o primeiro modelo não possui camadas de dropout.
    3. Possui mais unidades em suas camadas ocultas (64 e 32 em comparação com 10 do primeiro modelo).

Análise Geral: Se o conjunto de dados é pequeno ou há risco de overfitting, o segundo modelo pode ser mais robusto devido à adição de dropout. O primeiro modelo pode ser mais rápido de treinar, já que possui menos camadas e unidades. A escolha entre os modelos pode depender da natureza específica do problema, tamanho do conjunto de dados e recursos computacionais disponíveis.

3. Análise da RNA

Análise Comparativa dos Modelos de Redes Neurais Artificiais para Detecção de Câncer de Mama precoce é vital para o tratamento eficaz e a melhoria das taxas de sobrevivência. Técnicas de aprendizado de máquina, em particular, redes neurais artificiais, têm sido aplicadas com sucesso para essa tarefa. Neste estudo, exploramos duas arquiteturas de redes neurais para classificar amostras de câncer de mama como malignas ou benignas, destacando seus processos de treinamento e performance.

3.1. Pré-processamento de Dados

Iniciamos carregando e pré-processando dados de amostras de câncer de mama. Removemos instâncias com valores faltantes e normalizamos as características para garantir que cada atributo contribua igualmente para o modelo. Os dados foram divididos em conjuntos de treinamento e teste para avaliar a capacidade de generalização dos modelos.

3.2. Primeiro Modelo de Rede Neural

O primeiro modelo é uma rede neural simples com uma camada oculta de 10 neurônios usando a função de ativação ReLU. A camada de saída tem um único neurônio com uma função de ativação sigmoide, tornando-o adequado para tarefas de classificação binária. O otimizador Adam com uma taxa de aprendizado de 0.001 foi escolhido para treinar o modelo. Onde durante o treinamento, monitoramos a perda no conjunto de treinamento e validação ao longo das épocas. Os resultados mostraram uma precisão promissora no conjunto de teste.

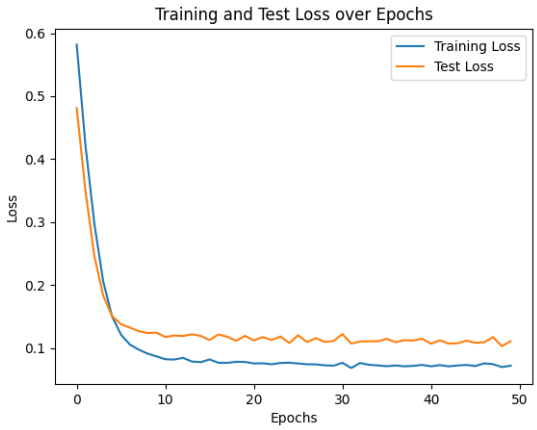

Figure 4. Perda de treinamento e teste ao longo das épocas - Modelo 1

O segundo modelo de rede neural apresenta uma arquitetura diferente para explorar possíveis melhorias. As mesmas configurações básicas foram mantidas, mas pequenas alterações, como o número de neurônios na camada oculta, podem levar a variações significativas no desempenho.

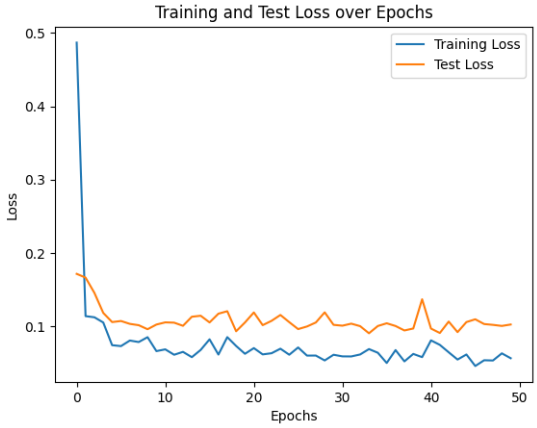

Figure 5. Perda de treinamento e teste ao longo das épocas - Modelo 2

Avaliando ambos os modelos foram indetificados no conjunto de teste usando métricas comuns de classificação binária, como precisão e matriz de confusão. A Área sob a Curva ROC (ROC AUC) foi calculada para avaliar a capacidade discriminativa dos modelos mostrados acima e nas Figuras 1 e 2. Os resultados mostram que ambos os modelos alcançaram boa precisão na tarefa de detecção de câncer de mama. No entanto, diferenças nas curvas de aprendizado e desempenho indicam nuances na eficácia de cada modelo.

O primeiro modelo demonstrou rápida convergência e desempenho robusto, enquanto o segundo modelo, embora também eficaz, pode ter benefícios em cenários específicos ou com mais ajustes de hiperparâmetros. Gráficos como a curva ROC e matrizes de confusão forneceram insights visuais sobre o desempenho dos modelos. A comparação direta desses gráficos permitiu uma avaliação holística, destacando áreas em que um modelo pode superar o outro.

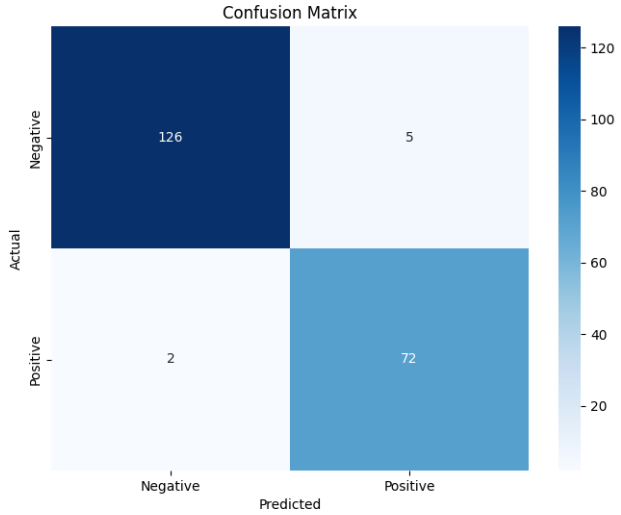

Figure 6. Matriz de confusão - Modelo 1

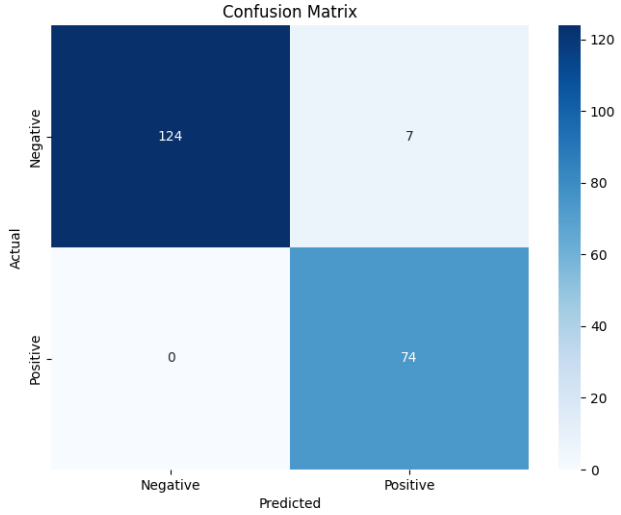

Figure 7. Matriz de confusão - Modelo 2

4. Considerações Finais

A escolha entre os modelos pode depender de requisitos específicos, como a ênfase na precisão versus a sensibilidade. Este estudo fornece uma estrutura para a análise comparativa de modelos de redes neurais, destacando a importância de entender não apenas métricas globais, mas também as nuances específicas de cada modelo. [de Assis et al. 2021] A aplicação de redes neurais para detecção de câncer de mama é promissora, e uma abordagem comparativa pode orientar a escolha do modelo mais adequado para uma determinada aplicação.

self.model.add(Dense(10, input_dim=input_dim, activation='relu'))

Nessa linha, você está adicionando uma camada densa (totalmente conectada) à sua rede neural. O primeiro argumento, 10, representa o número de neurônios ou unidades nessa camada. Alterando esse valor para outro número, você está modificando a arquitetura da rede, o que pode ter um impacto significativo no desempenho do modelo.

Por exemplo, se você alterar 10 para 20, a camada oculta terá o dobro de neurônios, o que pode levar a uma capacidade de aprendizado maior, mas também pode aumentar a possibilidade de overfitting, especialmente se você não ajustar outros hiperparâmetros adequadamente. Por outro lado, diminuir o número de neurônios pode levar a uma capacidade de aprendizado reduzida. Por exemplo:

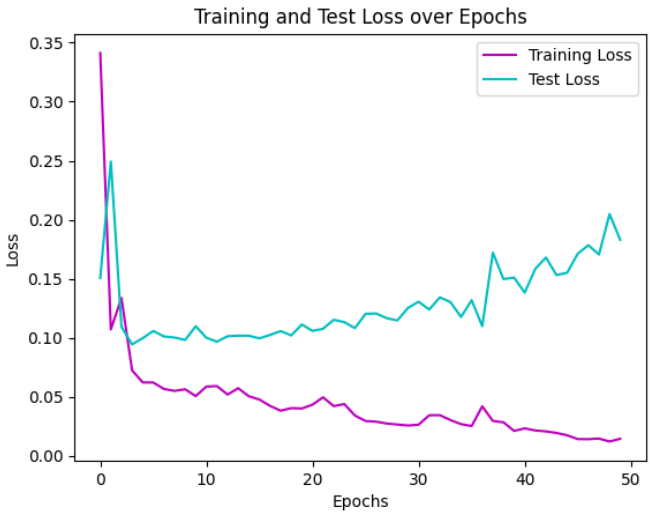

Figure 8. Possibilidade de Overfitting

Ajustar o número de neurônios na camada oculta é uma das maneiras de controlar a complexidade do modelo e, portanto, sua capacidade de generalização para dados não vistos. Essa é uma das decisões de design que os cientistas de dados frequentemente precisam tomar ao construir redes neurais.

O overfitting é um fenômeno comum em machine learning que ocorre quando um modelo se ajusta demais aos dados de treinamento. Em outras palavras, o modelo aprende não apenas os padrões fundamentais dos dados, mas também os ruídos e detalhes específicos desses dados, prejudicando sua capacidade de generalizar para novos dados não vistos anteriormente. Entender e lidar com o overfitting é crucial para o desenvolvimento de modelos de machine learning robustos e generalizáveis. A escolha entre esses modelos dependerá do conjunto de dados específico e do equilíbrio desejado entre simplicidade e capacidade de aprendizado. Em geral, é uma prática comum experimentar diferentes arquiteturas para encontrar aquela que se adapta melhor aos dados em questão.

Monitore o desempenho do modelo em um conjunto de validação durante o treinamento e pare quando o desempenho começar a se deteriorar, optar por modelos mais simples, evita complexidade desnecessária e utilize técnicas como validação cruzada para avaliar o desempenho do modelo em diferentes subconjuntos de dados.

[Cesari et al. 2019]

Ao lidar com um modelo mais complexo como o Modelo 2, é crucial monitorar o desempenho em um conjunto de validação durante o treinamento. Se a precisão no conjunto de treinamento continuar a aumentar enquanto a precisão no conjunto de validação diminuir, é um sinal claro de overfitting. [Vashisth et al. 2013] Pode ser útil experimentar com diferentes taxas de dropout e arquiteturas de rede para encontrar o equilíbrio certo entre complexidade e generalização. Isso é conhecido como ajuste fino do modelo.

O Modelo 2 é mais complexo devido às camadas ocultas adicionais e ao dropout. Essa complexidade adicional pode ser benéfica para melhorar a capacidade de generalização, especialmente em conjuntos de dados grandes ou complexos. O Modelo 1 é mais simples e pode funcionar bem em conjuntos de dados menores ou mais simples. A escolha entre esses modelos dependerá do conjunto de dados específico e do equilíbrio desejado entre simplicidade e capacidade de aprendizado. Em geral, é uma prática comum experimentar diferentes arquiteturas para encontrar aquela que se adapta melhor aos dados em questão.

Exemplo de Comparação de perda de teste entre modelos analisados

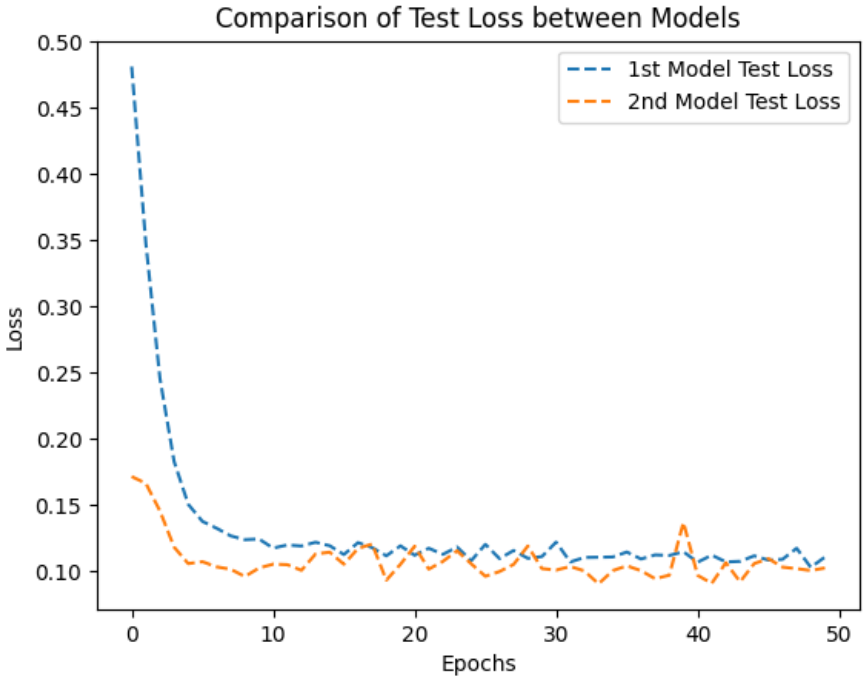

Figure 9. Comparação de perda de teste

O gráfico ”Comparison of Test Loss between Models” representa a comparação da perda nos dados de teste entre dois modelos de aprendizado de máquina. Neste gráfico, as perdas nos dados de teste de ambos os modelos são traçadas ao longo das épocas ou iterações do treinamento.

Cada curva no gráfico representa a evolução da perda nos dados de teste para um modelo específico. A posição relativa das curvas indica como o desempenho de cada

modelo se compara em termos de minimização da perda durante o treinamento.

O eixo horizontal representa as épocas ou iterações do treinamento, enquanto o eixo vertical representa a perda nos dados de teste. O objetivo é observar como as curvas se comportam ao longo do tempo. Se uma curva estiver diminuindo, isso indica que o modelo está melhorando em termos de generalização para dados não vistos. Se uma curva começar a subir, pode indicar que o modelo está começando a sofrer de overfitting, perdendo sua capacidade de generalização.

Comparar as curvas de perda nos dados de teste entre modelos fornece insights sobre quão bem cada modelo se adapta aos dados de teste e como eles podem se comportar em situações do mundo real. O modelo com uma curva de perda mais baixa nos dados de teste é geralmente considerado mais eficaz, desde que essa melhoria não seja devida a overfitting nos dados de treinamento.

References

Bagby, T., Rao, K., and Sim, K. C. (2018). Efficient implementation of recurrent neural network transducer in tensorflow. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 506–512. IEEE.

Cesari, A., Bottaro, S., Lindorff-Larsen, K., Banas, P., Sponer, J., and Bussi, G. (2019). Fitting corrections to an rna force field using experimental data. *Journal of chemical theory and computation*, 15(6):3425–3431.

de Assis, F. M., Coutinho, M. A., da Silva Filho, J. B., Macedo, E. L., and de Moraes, L. F. (2021). Iptraf: Coleta e detecção de anomalias em fluxos de rede. In *Anais do XXVI Workshop de Gerência e Operação de Redes e Serviços*, pages 96–109. SBC.

Elesbão, I. S. et al. (2023). Uso de redes neurais artificiais para determinação de pixels indicando água em uma imagem de satélite.

Rosa, E. N. et al. (2022). Utilização de redes neurais para reconhecimento de doenças foliares em culturas agrícolas.

Vashisth, H., Skiniotis, G., and Brooks III, C. L. (2013). Enhanced sampling and overfitting analyses in structural refinement of nucleic acids into electron microscopy maps. *The Journal of Physical Chemistry B*, 117(14):3738–3746.

Verma, R., Wagemans, J., Dahal, P., and Elfrink, A. (2021). Explaining low dimensional representation, a reproduction. In *ML Reproducibility Challenge 2020*.