Universidade da Beira Interior
Departamento de Informática

Relatório do Projeto Prático 2

CLASSIFICAÇÃO

Elaborado por:

TASSIA DA SILVA DE CARVALHO

MESTRADO EM ENGENHARIA INFORMÁTICA

UC.14469 APRENDIZAGEM AUTOMÁTICA

Professor Doutor HUGO PEDRO PROENÇA

Capítulo

1

Introdução

1.1 Objetivo do Relatório

O presente trabalho tem como objetivo desenvolver uma rede neural para resolver o problema de classificação de dígitos manuscritos utilizando o dataset MNIST. Este dataset é amplamente utilizado como referência para algoritmos de reconhecimento visual, especialmente para tarefas de classificação de imagens. Ele é citado em diversos livros e artigos renomados, como *Deep Learning* de Ian Goodfellow, Yoshua Bengio e Aaron Courville, e *Pattern Recognition and Machine Learning* de Christopher M. Bishop, onde é apresentado como um benchmark fundamental para validar modelos de aprendizado de máquina. Desde sua criação por Yann LeCun e colaboradores, o MNIST tornou-se um dos conjuntos de dados mais populares para testar e comparar a performance de diferentes arquiteturas de redes neurais.

Cada imagem contém um dígito de 0 a 9 representado por uma matriz de 28x28 pixels que, quando convertida para valores de escala de cinza, resulta em 784 pixels para cada exemplo. A tarefa principal é, portanto, classificar corretamente estas imagens, identificando o dígito presente em cada uma.

Para esta tarefa, foi configurada uma rede neural feedforward, que é o tipo mais básico de rede neural, onde as informações fluem em uma única direção, a qual processa os dados em camadas sequenciais de neurónios, extraindo características que permitem a classificação entre as diferentes classes de dígitos.

A correta escolha e configuração dos parâmetros do modelo têm um impacto direto no desempenho e na eficácia da rede. Parâmetros como a quantidade de camadas, o número de neurónios em cada camada, a taxa de aprendizagem (learning rate), o tamanho dos lotes (batch size), a taxa de dropout e o uso de regularização influenciam fortemente a capacidade da rede em aprender padrões significativos e generalizáveis sem incorrer em overfitting. Além disso, o uso de técnicas como validação cruzada e early stopping ajuda a garantir que o modelo atinja uma boa performance e apresente robustez perante novos dados. Neste relatório, será explorada a importância de cada um desses parâmetros e o impacto das escolhas feitas na precisão e estabilidade do modelo final.

Capítulo

2

Conceitos e Metodologias

2.1 Introdução

O dataset MNIST (Modified National Institute of Standards and Technology) trata-se de um conjunto de dados que contém imagens de dígitos manuscritos, variando de 0 a 9, com uma ampla diversidade de estilos e formas devido às variações naturais na escrita humana.

Cada amostra no dataset é representada por uma linha com 785 valores. O primeiro valor de cada linha corresponde ao *label*, ou rótulo, que identifica o dígito presente na imagem, variando entre 0 e 9. Os 784 valores restantes representam os pixels da imagem de 28x28 pixels, com valores que variam de 0 (preto) a 255 (branco). Essa estrutura transforma cada imagem num vetor unidimensional com 784 características, permitindo que seja processada pela rede neural.

O dataset MNIST está dividido em dois conjuntos principais:

- **Conjunto de Treino**: composto por 60.000 exemplos, utilizado para ensinar a rede neural a reconhecer padrões.
- **Conjunto de Teste**: composto por 10.000 exemplos, utilizado para avaliar a capacidade do modelo de generalizar para novos dados.

A tarefa de classificar dígitos manuscritos não é tão simples quanto parece, de facto, ele é composto por imagens de dígitos escritos à mão, ou seja, por pessoas. Originalmente, o dataset foi criado a partir de uma coleta de números escritos por diferentes pessoas para capturar uma grande variedade de estilos de escrita. Cada dígito, de 0 a 9, tem pequenas variações, já que ninguém escreve exatamente da mesma forma. Por isso, o MNIST reflete essas variações naturais na escrita humana, o que torna o problema de classificação mais realista e desafiador para algoritmos de reconhecimento de padrões. Com este conjunto de dados, é possível avaliar se o modelo consegue aprender a reconhecer padrões de forma eficaz e generalizar para imagens novas, independentemente das variações naturais na escrita. (Bishop, pag 677, 2006)

2.2 Redes Neurais Artificiais

Redes Neurais Artificiais (RNAs) são modelos computacionais inspirados na estrutura e funcionamento do cérebro humano, sendo compostas por neurônios artificiais interligados. Elas são especialmente eficazes em resolver problemas complexos, onde a relação entre as variáveis de entrada e saída não é linear.
Cada neurônio realiza operações matemáticas simples, consistindo em uma função linear seguida de uma função de ativação não linear. A combinação dessas funções permite a captura de padrões complexos. “Essas redes são ferramentas particularmente poderosas para reconhecimento de padrões devido à sua capacidade de aprender relações complexas e não lineares.” (Bishop, 2006, p. 225)

2.3 Estrutura do Modelo (Topologia)

A topologia de um modelo refere-se a quantas camadas de neurónios ele possui (camadas de entrada, ocultas e de saída) e como essas camadas estão organizadas e interconectadas. Em redes *feed-forward*, são constituídas por uma série de camadas de neurónios que estão conectados entre si, e cada neurónio em uma camada é ligado a todos os neurónios da camada seguinte, por exemplo, a informação passa sequencialmente da camada de entrada para as camadas ocultas e finalmente para a camada de saída (Bishop, 2006, p. 227). Esses neurónios processam informações de forma semelhante aos neurónios biológicos, aplicando uma função matemática (função de ativação) aos dados de entrada para transformar as informações e "aprender" os padrões.

Cada camada possui um certo número de neurónios, que são as unidades de processamento da rede. A quantidade de neurónios afeta a capacidade do modelo de "aprender" e pode impactar diretamente sua performance e complexidade. A escolha do número de neurónios em cada camada faz parte do design da topologia do modelo.

A topologia inclui também as funções de ativação e conexões entre camadas, que são componentes fundamentais numa rede neural, pois determinam como a informação é processada e transformada à medida que passa de uma camada para outra.

As funções de ativação são operações matemáticas aplicadas ao output de cada neurónio em uma camada. Elas têm o papel de introduzir não-linearidade no modelo, o que permite à rede aprender e representar relações complexas entre os dados de entrada. Sem funções de ativação, a rede se comportaria como uma combinação linear e, portanto, não seria capaz de resolver problemas complexos, as mais utilizadas são ReLU(Rectified Linear Unit), Softmax, Sigmoid e Tanh.

As conexões entre camadas de uma rede *feedforward* são organizadas de maneira que cada neurónio de uma camada esteja ligado a todos os neurónios da camada seguinte. Esse tipo de estrutura é conhecido como **conexão densa ou totalmente conectada** (fully connected layer), pois todos os neurónios de uma camada se comunicam com todos os neurónios da próxima camada. Em síntese, Bishop aborda esses conceitos sobre redes neurais no capítulo 5, fundamentando alguns dos conceitos apresentados neste relatório.

Uma topologia mais complexa pode incluir muitas camadas e neurónios, aumentando a capacidade de aprendizado, mas também pode levar a *overfitting* (Bishop, 2006, p. 144). Esta seção explica como o número de camadas e neurónios foi escolhido para equilibrar o aprendizado e a generalização.

2.3.1 Arquitetura da Rede Neural para processar o MNIST

O modelo implementado é uma rede neural *feedforward*. Este tipo de arquitetura é ideal para tarefas de classificação como a do dataset MNIST, onde o objetivo é reconhecer padrões visuais (os dígitos) e associá-los a uma classe específica (0 a 9).

2.3.2 Número de Camadas e Número de Neurónios em Cada Camada

O modelo foi configurado com três tipos de camadas principais:

1. **Camada de Entrada**: Esta camada recebe os dados iniciais. No caso do MNIST, como cada imagem é representada por **784 valores de pixels**, a camada de entrada é formada por **784 neurónios**. Cada neurónio representa um pixel da imagem, o que permite que a rede capture a informação visual básica de cada dígito.
2. **Camadas Ocultas**: São as camadas intermediárias onde ocorre o aprendizado mais complexo. No modelo, foram escolhidas **duas ou três camadas ocultas**, com um número significativo de neurónios **(512, 256) e (512, 512, 256)**. As camadas ocultas permitem que a rede aprenda a combinar e abstrair características dos dados de entrada, de forma a criar representações úteis para a tarefa de classificação. Camadas com muitos neurónios permitem ao modelo capturar interações complexas entre os pixels, o que é essencial para distinguir entre diferentes dígitos, isso para os testes e verificação dos parâmetros, o modelo final foi ajustado para receber o número de neurônios e camadas ocultas do modelo por parâmetros.
    - **Justificação para o Número de Camadas**: A escolha de duas ou três camadas ocultas permitiu equilibrar a complexidade do modelo. Uma única camada pode ser insuficiente para capturar padrões complexos dos dígitos manuscritos, enquanto muitas camadas podem levar a uma maior complexidade e riscos de *overfitting* (quando o modelo se adapta tanto aos dados de treino que não consegue generalizar bem para novos dados). Esse número moderado de camadas ajuda a capturar informações relevantes sem aumentar demasiadamente a complexidade computacional.
    - **Relação entre Capacidade do Modelo e Risco de Overfitting**: Camadas com mais neurónios geralmente proporcionam ao modelo uma maior "capacidade de aprendizado", ou seja, ele consegue aprender padrões mais detalhados e complexos. No entanto, isso aumenta o risco de *overfitting*, pois a rede pode acabar aprendendo os detalhes específicos dos dados de treino em vez de generalizar bem para novos dados. Para mitigar esse risco, foram aplicadas técnicas como *dropout* (que desativa neurónios aleatoriamente durante o treino) e regularização L2 (que penaliza pesos elevados), ajudando a rede a focar nos padrões mais gerais.
3. **Camada de Saída**: Esta é a última camada, que dá a resposta final do modelo. No caso do MNIST, a camada de saída possui 10 neurónios, um para cada dígito possível (de 0 a 9). Cada neurónio da camada de saída indica a probabilidade de a imagem pertencer a uma das classes. A função de ativação usada aqui é a ***softmax***, que converte as saídas em

probabilidades e permite que o modelo classifique a imagem para a classe com a maior probabilidade.

4. **Funções de Transferência (Funções de Ativação)**: As funções de ativação ou funções de transferência transformam a saída dos neurónios para que a rede possa capturar relações complexas nos dados e produzir resultados adequados à tarefa. Essas funções ajudam a introduzir não-linearidade na rede, o que é essencial para resolver problemas complexos.
    1. **ReLU (Rectified Linear Unit)**:
        Utilizada nas camadas ocultas, a função ReLU transforma os valores negativos em zero, mantendo os valores positivos inalterados. Esta operação simples permite que a rede aprenda padrões mais profundos, sem que a informação "desapareça" durante o processo de propagação de volta.
        Justificação: A função ReLU é amplamente usada em redes neurais feedforward devido à sua eficiência computacional e capacidade de reduzir problemas como o *vanishing gradient*. É ideal para extrair características em camadas ocultas, o que é crucial para a classificação de imagens complexas como as do MNIST.
    2. **Softmax**:
        A função *softmax* é aplicada na camada de saída, onde converte os outputs dos neurónios em probabilidades para cada uma das classes (neste caso, de 0 a 9).
        Justificação: Como o objetivo é classificar cada imagem em uma das 10 classes, o *softmax* permite que o modelo expresse a probabilidade de cada classe, facilitando a tarefa de escolha da classe com maior probabilidade e, portanto, a classificação final.

2.4 Parâmetros de Treinamento

Os parâmetros de treinamento definem a forma como a rede neural aprende e convergem para uma solução. São ajustados para garantir que o modelo aprenda de maneira eficiente e generalize bem para novos dados.

No presente trabalho, o modelo de rede neural foi inicialmente configurado para testar diferentes combinações de hiperparâmetros, com o objetivo de otimizar o desempenho na tarefa de classificação de dígitos manuscritos. Estes parâmetros de treinamento foram escolhidos com base nos usos mais comuns, e depois foram ajustados com base nas métricas de performance (acurácia média entre as folds) utilizando validação cruzada K-Fold com 5 divisões. Abaixo estão as descrições detalhadas dos principais parâmetros de treinamento.

2.4.1 Otimização com Validação Cruzada K-Fold

Para assegurar que os melhores hiperparâmetros sejam escolhidos, o modelo foi treinado e avaliado com uma estratégia de validação cruzada K-Fold, com k=5k = 5k=5 divisões. Em cada fold, o modelo é treinado em uma parte dos dados e avaliado na outra parte, alternando-se as divisões para cobrir todo o conjunto de treino. Esta abordagem permite calcular uma acurácia média para cada combinação de hiperparâmetros, proporcionando uma estimativa robusta do desempenho e ajudando a evitar que o modelo se adapte excessivamente aos dados de treino (overfitting).

2.4.2 Batch Size

O *batch size*, ou tamanho do lote, refere-se ao número de amostras que a rede processa antes de atualizar os pesos. Em vez de calcular os ajustes de peso para cada amostra individualmente (o que seria lento e instável) ou de processar todas as amostras de uma só vez (o que exigiria muita memória), o *batch size* permite um equilíbrio entre velocidade e precisão, processando grupos menores de amostras.

- **Papel do Batch Size**: Em cada iteração, o modelo calcula o erro com base em um lote específico de amostras e ajusta os pesos com base nessa informação. Lotear as amostras desta forma ajuda a equilibrar a eficiência computacional e a estabilidade do aprendizado, uma vez que processar cada amostra individualmente seria computacionalmente caro e processar o conjunto de dados inteiro de uma só vez poderia exigir muita memória.
- **Escolha do Valor de Batch Size**: Os valores comuns de *batch size* variam entre 32 e 128. Tamanhos maiores podem estabilizar o treinamento, mas exigem mais memória e podem demorar a processar cada iteração. Por outro lado, tamanhos menores podem levar a um aprendizado instável e mais lento para chegar a um ponto de convergência. Neste modelo, foram testados valores de **32, 64 e 128**, procurando um equilíbrio entre eficiência computacional e estabilidade na atualização dos pesos. Esses valores foram escolhidos para garantir que o modelo tenha uma base ampla de dados para ajustar os pesos de forma consistente, sem comprometer a performance.

2.4.3 Learning Rate (Taxa de Aprendizagem)

A *learning rate*, ou taxa de aprendizagem, determina o tamanho dos passos que o modelo dá ao ajustar os pesos. Este é um dos parâmetros mais críticos no treinamento de redes neurais, pois afeta diretamente a velocidade e a qualidade da convergência.

- **Papel da Learning Rate**: A taxa de aprendizagem controla o quanto os pesos da rede são ajustados em resposta ao erro observado em cada atualização. Uma taxa de aprendizagem muito alta pode fazer com que o modelo "salte" para longe do mínimo desejado, enquanto uma taxa muito baixa pode tornar o processo de aprendizagem lento demais, demorando muito para alcançar um ponto satisfatório ou até mesmo caindo num mínimo local.

- **Escolha do Valor de Learning Rate**: Foram testadas várias taxas de aprendizagem, incluindo **1e-5, 1e-4 e 1e-3** Para este modelo, optou-se por valores menores para garantir uma convergência mais estável.

2.4.4 Número de Épocas

O **número de épocas** (*epochs*) é um hiperparâmetro importante no treinamento de redes neurais, pois define quantas vezes o modelo verá o conjunto completo de dados de treino durante o processo de aprendizado. Em cada época, o modelo passa por todas as amostras de treino, ajustando os pesos com base no erro calculado a partir dos lotes (*batches*) processados.

- **Papel do Número de Épocas no Treinamento**
  - **Épocas Insuficientes**: Se o número de épocas for baixo, o modelo pode não ter tempo suficiente para aprender os padrões presentes nos dados, resultando em um subajuste (*underfitting*), ou seja, o modelo não captura adequadamente a complexidade dos dados.
  - **Épocas Excessivas**: Se o número de épocas for muito alto, o modelo pode começar a memorizar os detalhes específicos dos dados de treino, o que leva ao sobreajuste (*overfitting*). Neste caso, o modelo fica "excessivamente adaptado" aos dados de treino e perde a capacidade de generalizar para novos dados.
- **Escolha do Número de Épocas**
  O número de épocas é ajustado com base em experimentação, observando-se o desempenho do modelo no conjunto de validação. O valor ideal do número de épocas varia conforme a complexidade do modelo, a quantidade de dados e a taxa de aprendizagem, no modelo foram utilizadas **50 épocas**, na tentativa de reduzir e agilizar os custos operacionais, porém, para evitar o sobreajuste, foi utilizada técnica ***early stopping***, que é o valor de paciência (patience), que define o número de épocas que o modelo pode continuar sendo treinado após a última melhoria no desempenho de validação antes de o treinamento ser interrompido. Esse valor é essencial para encontrar um ponto de equilíbrio: interromper o treinamento antes que o modelo comece a ajustar-se demais aos dados de treino, mas sem finalizá-lo prematuramente. Um valor de paciência moderado permite ao modelo fazer pequenos ajustes adicionais nos pesos após alcançar um bom desempenho. No presente trabalho, o **valor de paciência foi definido como 15**, pois esse valor permite ao modelo realizar ajustes necessários sem prolongar o treinamento além do necessário. Se o modelo não apresentar melhorias no conjunto de validação dentro desse número de épocas, o treinamento é interrompido automaticamente, prevenindo o ajuste excessivo aos dados de treino.

2.5 Função de Perda e Regularização

2.5.1 Binary Cross-Entropy com Regularização

Para tarefas de classificação, a função de perda escolhida desempenha um papel fundamental no aprendizado do modelo, pois define a forma como os erros são calculados e os pesos ajustados em cada iteração. Neste caso, a função de perda **Binary Cross-Entropy** foi utilizada com a técnica de regularização L2 para evitar o *overfitting*.

A função de perda *Binary Cross-Entropy* mede a diferença entre as probabilidades previstas pelo modelo e os valores reais, sendo ideal para problemas de classificação binária. Embora o problema com o dataset MNIST seja de classificação multiclasse, a configuração de cada *output* como uma unidade binária permite aplicar a *Binary Cross-Entropy* a cada classe individual. Essa função calcula a penalidade baseada na discrepância entre a previsão do modelo e o rótulo verdadeiro, incentivando o modelo a maximizar a probabilidade das classes corretas e minimizar a das incorretas.

A *Binary Cross-Entropy* incentiva o modelo a reduzir as diferenças entre a previsão e os rótulos reais, ajudando a encontrar a configuração de pesos que minimiza os erros de classificação.

2.5.2 Regularização L2: Prevenindo o Overfitting

Overfitting ocorre quando o modelo aprende padrões específicos dos dados de treino, perdendo a capacidade de generalizar. A regularização adiciona penalidades na função de custo para modelos excessivamente complexos. Além da função de perda, uma técnica de **regularização L2** foi aplicada ao modelo. A regularização é uma estratégia essencial para controlar o *overfitting*, que ocorre quando o modelo se adapta excessivamente aos dados de treino e perde a capacidade de generalizar para novos dados. A regularização L2 adiciona uma penalidade ao valor da função de perda, baseada na magnitude dos pesos do modelo.

O objetivo da regularização L2 é manter os pesos do modelo em valores menores, o que impede que ele dependa excessivamente de características específicas dos dados de treino. Valores elevados de pesos podem indicar que o modelo está ajustado de forma muito precisa aos dados de treino, capturando até ruídos indesejados. Ao minimizar os valores dos pesos, a regularização L2 incentiva o modelo a aprender padrões mais gerais e robustos, melhorando sua capacidade de generalização, os pesos utilizados para L2 foram **0.001, 0.01, 0.05.**

2.5.3 Dropout: Melhorando a Robustez do Modelo

Além da regularização L2, foi utilizada a técnica adicional **Dropout** ao modelo. O *dropout* é uma técnica de regularização que ajuda a reduzir o *overfitting* desativando, de forma aleatória, uma fração dos neurónios em cada camada durante o treinamento. Em cada iteração, o *dropout* seleciona neurónios que serão temporariamente "desligados", o que significa que esses

neurónios não participarão do processo de ajuste dos pesos. Essa prática obriga o modelo a aprender representações mais robustas, já que não pode depender exclusivamente de neurónios específicos para realizar previsões.

- **Funcionamento e Vantagens do Dropout**

  A técnica de *dropout* contribui para a generalização do modelo ao forçá-lo a criar redundância nos padrões aprendidos. Como o modelo não sabe quais neurónios estarão ativos em uma determinada iteração, ele precisa distribuir o aprendizado entre os diferentes neurónios, garantindo que múltiplos neurónios possam captar características importantes dos dados. Isso ajuda a evitar que o modelo memorize os dados de treino, aumentando sua capacidade de generalizar para novos dados.

  - **Escolha da Taxa de Dropout**: Durante o treinamento, foram testadas taxas de *dropout* de **0.1, 0.25 e 0.5**, que representam a fração de neurónios desativados em cada camada durante cada iteração. Taxas mais altas aumentam a robustez do modelo, mas podem prejudicar a capacidade de aprendizado caso a quantidade de neurónios ativos se torne insuficiente. Assim, foi necessário equilibrar a taxa de *dropout* com a estrutura e profundidade do modelo.

2.5.4 Combinação de Dropout e Regularização L2

O uso combinado de *dropout* e regularização L2 reforça a capacidade de generalização do modelo. Enquanto a regularização L2 limita o crescimento dos pesos, evitando que o modelo se concentre demais em características específicas, o *dropout* distribui o aprendizado entre os neurónios, forçando o modelo a construir representações redundantes e robustas. Essa combinação ajuda a minimizar o risco de *overfitting*, assegurando que o modelo aprenda padrões relevantes e generalizáveis, ao invés de memorizar detalhes dos dados de treino.

Em conjunto, as técnicas de *Binary Cross-Entropy*, regularização L2 e *dropout* formam uma abordagem de regularização eficaz, garantindo que o modelo seja capaz de manter uma boa performance ao lidar com novos dados sem comprometer a precisão nas previsões.

Capítulo

3

Análise e Resultados

Para avaliar o desempenho do modelo e otimizar os Hiper parâmetros, foram realizadas várias experimentações com diferentes configurações de parâmetros. Essas variações permitiram observar como o modelo reage a ajustes em cada parâmetro, possibilitando a escolha da melhor combinação para alcançar uma alta performance e evitar o *overfitting*.

3.1 Configurações Experimentais

Para automatizar os testes, inicialmente foi desenvolvido um algoritmo de verificação com passagem de parâmetros:

```
# Definir intervalos de hiperparâmetros para teste
learning_rates = [1e-5, 1e-4, 1e-3]
batch_sizes = [32, 64, 128]
l2_regs = [0.001, 0.01, 0.05]
dropout_rates = [0.1, 0.25, 0.5]
topologies = [(512, 256), (512, 512, 256)]

# Loop para testar diferentes combinações de hiperparâmetros
for lr in learning_rates:
    for bs in batch_sizes:
        for reg in l2_regs:
            for dr in dropout_rates:
                for topo in topologies:
                    print(f"\n[INFO] Testando configuração: LR={lr}, Batch Size={bs}, L2={reg}, Dropout={dr}, Topologia={topo}")
                    train_and_evaluate(trainX, trainY, topo[0], topo[1], reg, dr, lr, bs, topo)

```

Essas variações foram aplicadas em diferentes combinações para observar o efeito de cada configuração e encontrar a melhor arquitetura para o conjunto de dados MNIST. O objetivo desta análise é identificar a combinação ideal de Hiper parâmetros que maximize a acurácia do modelo no conjunto de dados MNIST, minimizando o risco de *overfitting*.

Apesar de desejar que todas essas combinações fossem executadas o custo computacional estava sendo muito alto, portanto foram executas, 28 Combinações, o que já forneceu a possibilidade de analise para chegar a parâmetros mais precisos:

Nº	Learning Rate	Batch Size	L2 Regularization	Dropout	Topology	Acurácia Média	Desvio Padrão
0	1E-05	32	0.001	0.1	(512, 256)	96.95	0.14
1	1E-05	32	0.001	0.1	(512, 512, 256)	97.38	0.08
2	1E-05	32	0.001	0.25	(512, 256)	96.76	0.2
3	1E-05	32	0.001	0.25	(512, 512, 256)	97.27	0.09
4	1E-05	32	0.001	0.5	(512, 256)	96.13	0.23
5	1E-05	32	0.001	0.5	(512, 512, 256)	96.55	0.19
6	1E-05	32	0.01	0.1	(512, 256)	94.2	0.16
7	1E-05	32	0.01	0.1	(512, 512, 256)	94.73	0.3
8	1E-05	32	0.01	0.25	(512, 256)	94.16	0.23
9	1E-05	32	0.01	0.25	(512, 512, 256)	94.6	0.2
10	1E-05	32	0.01	0.5	(512, 256)	93.6	0.26
11	1E-05	32	0.01	0.5	(512, 512, 256)	94.2	0.15
12	1E-05	32	0.05	0.1	(512, 256)	89.66	0.38
13	1E-05	32	0.05	0.1	(512, 512, 256)	88.39	0.26
14	1E-05	32	0.05	0.25	(512, 256)	89.58	0.38
15	1E-05	32	0.05	0.25	(512, 512, 256)	88.02	0.21
16	1E-05	32	0.05	0.5	(512, 256)	89.29	0.04
17	1E-05	32	0.05	0.5	(512, 512, 256)	87.07	0.26
18	1E-05	64	0.001	0.1	(512, 256)	96.43	0.13
19	1E-05	64	0.001	0.1	(512, 512, 256)	97.01	0.07
20	1E-05	64	0.001	0.25	(512, 256)	96.22	0.23
21	1E-05	64	0.001	0.25	(512, 512, 256)	96.77	0.15
22	1E-05	64	0.001	0.5	(512, 256)	95.45	0.14
23	1E-05	64	0.001	0.5	(512, 512, 256)	95.78	0.15
24	1E-05	64	0.01	0.1	(512, 256)	93.63	0.17
25	1E-05	64	0.01	0.1	(512, 512, 256)	94.03	0.12
26	1E-05	64	0.01	0.25	(512, 256)	93.51	0.23
27	1E-05	64	0.01	0.25	(512, 512, 256)	93.89	0.12
28	1E-05	64	0.01	0.5	(512, 256)	92.96	0.23

3.2 Resultados das Métricas de Avaliação

Na tabela, todas as combinações utilizam uma Learning Rate de 1e-5, resultando em boas acurácias, mas em alguns casos levando a uma acurácia de validação ligeiramente menor (ex.: linhas com maior "Desvio Padrão", sugerindo variabilidade entre os folds), indicando que uma Learning Rate mais alta poderia capturar mais padrões sem risco de overfitting, acelerando a convergência.

Quanto aos Batch Sizes de 32 e 64, o tamanho 64 apresentou uma acurácia média ligeiramente superior em alguns casos, com menor desvio padrão (indicando melhor estabilidade entre folds); aumentar esse parâmetro pode melhorar a estabilidade do treinamento e verificar se uma maior atualização dos pesos a cada época ajuda a capturar melhor os padrões nos dados.

A Regularização L2 de 0.001 foi associada a melhores resultados médios de acurácia (ex.: entradas 0, 1, 3, 18 e 21), equilibrando bem a capacidade de aprendizado e a regularização para evitar overfitting.

Os valores de Dropout de 0.1 e 0.25 mostraram uma combinação adequada de acurácia média e desvio padrão baixo na tabela (ex.: linhas 1, 3, 5, 18 e 20), sugerindo que a regularização via dropout é importante para evitar a memorização dos dados de treinamento, especialmente em redes maiores.

Na tabela, observa-se que redes com topologias (512, 256) e (512, 512, 256) apresentam melhores resultados de acurácia média, indicando que redes mais profundas capturam melhor os padrões.

Para uma melhoria adicional, os testes seguiram com uma topologia mais profunda: (512, 512, 512, 256). A adição de uma terceira camada de 512 neurônios foi inspirada nos bons resultados das duas camadas de 512 na tabela (ex.: linha 3, com acurácia média de 97.27% e desvio padrão de 0.09), numa tentativa de capturar ainda mais padrões sutis nos dados sem overfitting.

A última camada de 256 neurônios é usada para uma redução gradual antes da camada de saída, o que ajuda a rede a focar nas características mais relevantes nos estágios finais do aprendizado.

3.3 Teste Individual

Após analise dos dados anteriores, observa-se:

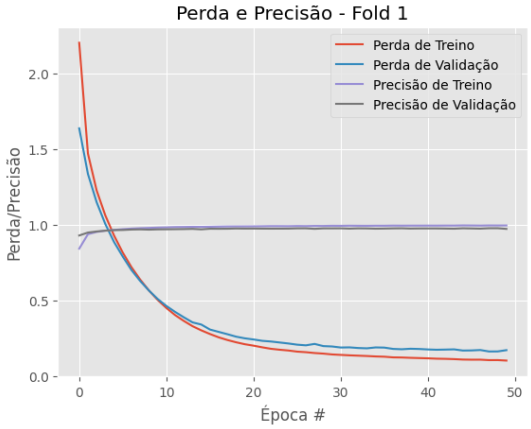
- Que as combinações com taxa de **dropout** de 0.1 tendem a ter uma acurácia levemente superior às de taxas mais altas. Uma taxa de dropout ainda menor, pode permitir ao modelo aprender mais com cada época, embora mantendo algum nível de regularização
- **A taxa de aprendizado** de 1e-5 foi utilizada, mas pode ser muito baixa para uma convergência mais rápida e efetiva. Foi utiliza a taxa ligeiramente maior, como **1e-4**
- Configurações mais complexas, como a topologia **(512, 512, 256)**, demonstraram uma acurácia melhor em relação a **(512, 256)**. Por isso o teste com uma topologia ainda mais profunda, como **(512, 512, 512, 256)** regularização leve 0.001, sendo assim foi pensado o modelo abaixo.

Primeira Combinação Individual (Melhor Desempenho):

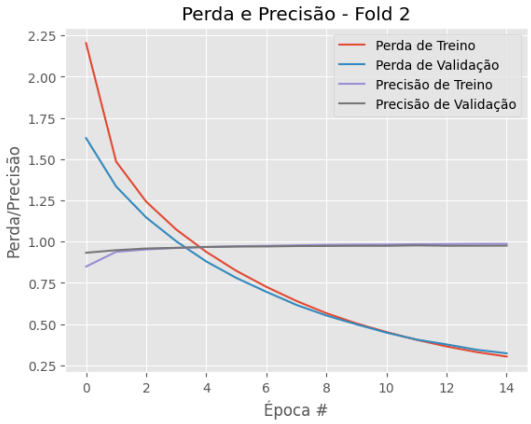
- **Learning Rate = 1e-4**
- **Batch Size = 128**
- **L2 Regularization = 0.001**
- **Dropout Rate = 0.05**
- **Topology = (512, 512, 512, 256)**

Obtendo:

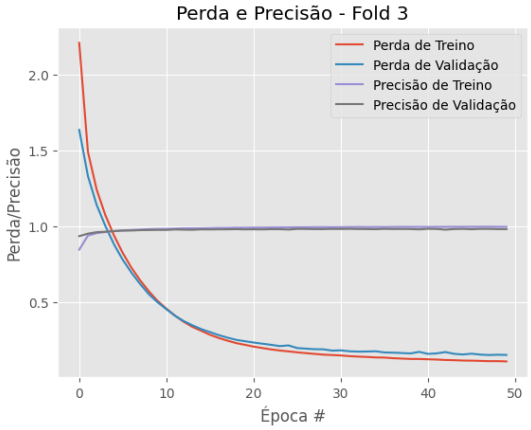
[INFO] Acurácia de treinamento na última época do fold 1: 99.64%
[INFO] Acurácia de validação no fold 1: 97.82% | Tempo de execução: 81.41 segundos



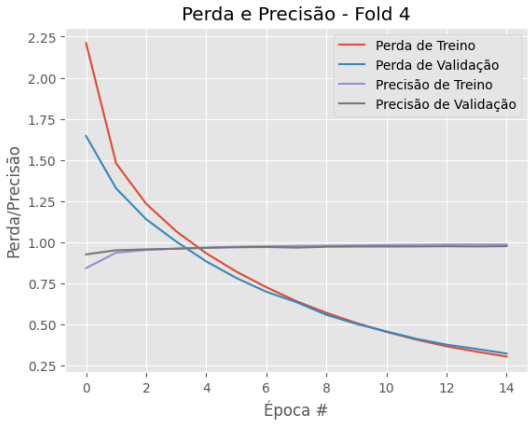
[INFO] Acurácia de treinamento na última época do fold 2: 98.70%
[INFO] Acurácia de validação no fold 2: 93.30% | Tempo de execução: 31.44 segundos



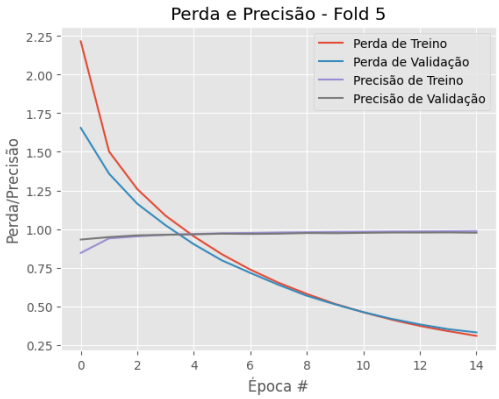
[INFO] Acurácia de treinamento na última época do fold 3: 99.57%
[INFO] Acurácia de validação no fold 3: 98.22% | Tempo de execução: 94.48 segundos



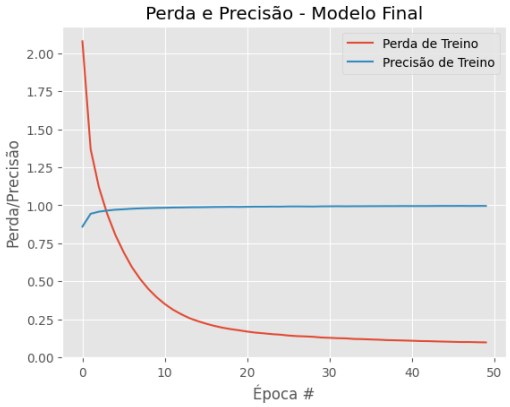
[INFO] Acurácia de treinamento na última época do fold 4: 98.67%
[INFO] Acurácia de validação no fold 4: 92.61% | Tempo de execução: 30.71 segundos



[INFO] Acurácia de treinamento na última época do fold 5: 98.72%
[INFO] Acurácia de validação no fold 5: 93.17% | Tempo de execução: 31.34 segundos



[INFO] Treinando modelo final em todos os dados de treino...
[INFO] Acurácia final no conjunto de teste: 98.23%
[INFO] Acurácia final no conjunto de validação: 99.81%



Os resultados indicam uma **alta acurácia de treinamento**, com valores variando entre **98.70% e 99.64%** nos diferentes *folds*, enquanto a **acurácia de validação** se manteve entre **92.61% e 98.22%**. Embora haja alguma variação entre os *folds*, o modelo demonstrou uma excelente capacidade de generalização ao alcançar uma **acurácia final de 98.23% no conjunto de teste** e **99.81% no conjunto de validação** após o treinamento completo.

Este desempenho confirma que a configuração manual trouxe melhorias significativas, servindo como uma base sólida para futuras otimizações. Outras combinações foram testadas, por exemplo está abaixo:

- **Learning Rate** = 5e-4
- **Batch Size** = 64
- **L2 Regularization** = 0.001
- **Dropout Rate** = 0.05
- **Topology** = (512, 256, 128)

[INFO] Acurácia de treinamento na última época do fold 1: 98.03%
[INFO] Acurácia de validação no fold 1: 97.78% | Tempo de execução: 97.61 segundos
[INFO] Acurácia de treinamento na última época do fold 2: 98.15%
[INFO] Acurácia de validação no fold 2: 97.97% | Tempo de execução: 79.70 segundos
[INFO] Acurácia de treinamento na última época do fold 3: 97.84%
[INFO] Acurácia de validação no fold 3: 95.11% | Tempo de execução: 37.97 segundos
[INFO] Acurácia de treinamento na última época do fold 4: 97.85%
[INFO] Acurácia de validação no fold 4: 95.58% | Tempo de execução: 39.32 segundos
[INFO] Acurácia de treinamento na última época do fold 5: 97.93%
[INFO] Acurácia de validação no fold 5: 95.22% | Tempo de execução: 38.04 segundos
[INFO] Acurácia média entre 5 folds (dados de validação): 96.33% ± 1.27%
[INFO] Treinando modelo final em todos os dados de treino...
[INFO] Acurácia final no conjunto de teste: 98.16%
[INFO] Acurácia final no conjunto de validação: 98.95%

Os resultados dessa configuração específica mostraram alta acurácia tanto no treinamento quanto na validação, com a acurácia de treinamento nos diferentes folds variando entre 97.84% e 98.15% e a acurácia de validação entre 95.11% e 97.97%. Apesar disso, esses valores não superaram os melhores resultados obtidos anteriormente, que apresentaram uma acurácia de validação final mais alta, de 99.81%.

Outras combinações foram testadas com resultados parecidos, mas em um determinado momento o modelo foi testado com apenas uma camada, e ajustando os demais Hiper parâmetros, destacado nos tópicos abaixo, o resultando surpreendeu em um primeiro momento, pois se equiparou a um modelo mais complexo:

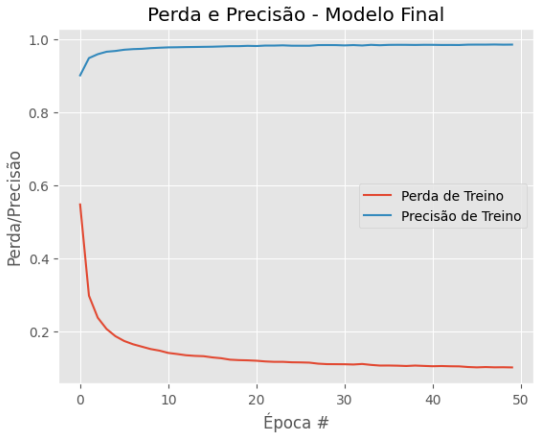
- **Learning Rate**: 5e-4
- **Batch Size**: 128
- **L2 Regularization**: 0.001
- **Dropout Rate**: 0.15
- **Topology**: (256)

Resultados Obtidos:

[INFO] Acurácia de treinamento na última época do fold 1: 98.56%
[INFO] Acurácia de validação no fold 1: 97.90% | Tempo de execução: 114.52 segundos
[INFO] Acurácia de treinamento na última época do fold 2: 97.84%
[INFO] Acurácia de validação no fold 2: 93.76% | Tempo de execução: 35.91 segundos

[INFO] Acurácia de treinamento na última época do fold 3: 97.79%
[INFO] Acurácia de validação no fold 3: 93.67% | Tempo de execução: 34.36 segundos
[INFO] Acurácia de treinamento na última época do fold 4: 97.87%
[INFO] Acurácia de validação no fold 4: 94.25% | Tempo de execução: 35.86 segundos
[INFO] Acurácia de treinamento na última época do fold 5: 97.92%
[INFO] Acurácia de validação no fold 5: 93.50% | Tempo de execução: 36.39 segundos
[INFO] Acurácia média entre 5 folds (dados de validação): 94.61% ± 1.66%
[INFO] Treinando modelo final em todos os dados de treino...
[INFO] Acurácia final no conjunto de teste: 98.11%
[INFO] Acurácia final no conjunto de validação: 99.30%

Acurácia de Treinamento: Variou entre 97.79% e 98.56% nos cinco folds.
Acurácia de Validação: Variou entre 93.50% e 97.90%, com uma média de 94.61% e desvio padrão de 1.66%.
Tempo de Execução: Tempo por fold variando entre 34 e 114 segundos.
Acurácia Final no Conjunto de Teste: 98.11%
Acurácia Final no Conjunto de Validação: 99.30%



- A acurácia de treinamento está relativamente alta, o que indica que o modelo está conseguindo aprender bem com os dados de treinamento.
- A acurácia de validação apresenta uma variabilidade significativa entre os folds (93.50% a 97.90%), com uma média razoável (94.61%), mas um desvio padrão alto (1.66%). Esse desvio padrão pode indicar uma instabilidade na generalização entre os folds, sugerindo que a configuração pode ser mais sensível à divisão dos dados.
- O tempo de execução é um pouco alto em comparação com outras configurações, especialmente no primeiro fold, que levou mais de 100 segundos. Isso pode indicar que o tamanho do lote (Batch Size) ou a topologia mais simples (apenas 256 neurônios) não estão totalmente otimizados para eficiência.
- A acurácia final no conjunto de validação (99.30%) é muito alta, mas ainda ligeiramente inferior à primeira configuração manual

3.3.1 Conclusão Comparativa

Primeira Combinação Individual (Melhor Desempenho):

- **Learning Rate = 1e-4**
- **Batch Size = 128**
- **L2 Regularization = 0.001**
- **Dropout Rate = 0.05**
- **Topology = (512, 512, 512, 256)**

Acurácia de Treinamento:
Variou entre 98.70% e 99.64%.
Acurácia de Validação:
Variou entre 92.61% e 98.22%.
Acurácia Final no Conjunto de Teste: 98.23%
Acurácia Final no Conjunto de Validação: 99.81%

Combinação com o segundo Melhor Desempenho:

- **Learning Rate: 5e-4**
- **Batch Size: 128**
- **L2 Regularization: 0.001**
- **Dropout Rate: 0.15**
- **Topology: (256)**

Acurácia de Treinamento:
Variou entre 97.79% e 98.56%.
Acurácia de Validação:
Variou entre 93.50% e 97.90%
Acurácia Final no Conjunto de Teste: 98.11%
Acurácia Final no Conjunto de Validação: 99.30%

Performance Geral:

- Ambas as configurações apresentaram alto desempenho no conjunto de treinamento e validação, mas a **primeira configuração** mostrou resultados superiores, tanto na acurácia final no conjunto de teste quanto no conjunto de validação.

Estabilidade e Generalização:

- A segunda configuração, com uma topologia mais simples e menor capacidade de rede (apenas 256 neurônios), conseguiu acurácias razoáveis, mas apresentou maior variabilidade entre os folds. A primeira configuração obteve melhor acurácia final, sugerindo uma capacidade de generalização ligeiramente superior e mais estável.

O que chama a atenção é que uma rede de apenas uma camada tem um resultado expressivamente parecido com uma rede mais complexa, isso pode ocorrer por alguns motivos elencados abaixo.

3.3.1.1 Simplicidade Pode Favorecer Generalização

- M**ais simples** nem sempre significa **pior**. Redes neurais mais complexas (com mais camadas e parâmetros) têm uma maior capacidade de aprendizado, mas também um maior risco de **overfitting** (memorização dos dados de treinamento sem capturar os padrões generalizáveis).
- Com uma topologia de uma única camada (ex.: 256 neurônios), a rede é forçada a **capturar apenas os padrões mais essenciais** dos dados. Isso pode resultar em uma melhor generalização, especialmente se o problema não exigir uma estrutura muito complexa.

3.3.1.2 Problema Pode Ser "Facilmente Linearizável"

- Redes de uma camada atuam quase como uma forma de **classificação linear**, especialmente se o número de neurônios for suficiente para o problema. Redes mais profundas são geralmente necessárias para capturar **hierarquias complexas de características**. No caso do MNIST, uma camada foi capaz de encontrar padrões suficientes para uma boa acurácia.

3.3.1.3 Regularização e Dropout Auxiliam a Rede Simples

- A **regularização L2** e o **dropout** ajudam a evitar que a rede se torne excessivamente dependente de padrões específicos dos dados de treinamento, mesmo em arquiteturas mais simples.nNa configuração de uma camada, o uso de regularização e dropout forçou o modelo a se concentrar nos padrões que aparecem de forma consistente nos dados, ignorando ruídos ou peculiaridades dos dados de treinamento. Isso também ajudou a melhorar a generalização.

3.3.1.4 O Tamanho da Camada Única é Comparável em Número de Parâmetros

- Uma camada de 256 neurônios ainda tem um número significativo de parâmetros para aprendizado, o que permitiu a rede capturar uma quantidade razoável de padrões.
- Em termos práticos, uma camada única com 256 neurônios tem cerca de 200.000 parâmetros (considerando 784 entradas e 10 saídas), o que é grande o suficiente para capturar padrões relevantes do MNIST, mas não tão grande a ponto de **saturar** ou **superaquecer** o modelo com informações irrelevantes.

3.3.1.5 A Estrutura dos Dados Pode Não Necessitar de Hierarquias Complexas

- MNIST é um dataset que contém imagens de baixa resolução (28x28 pixels) e é relativamente simples. Em datasets com mais detalhes e variações (ex.: imagens coloridas complexas). Para o MNIST, a rede de uma camada conseguiu capturar características simples como contornos de dígitos, curvas e linhas horizontais/verticais assim como em uma arquitetura profunda.

Conclusão

Para tarefas simples como a classificação de dígitos manuscritos em MNIST, a complexidade adicional de uma rede de três camadas pode não trazer uma melhoria substancial e pode até dificultar a generalização. A rede de uma camada é capaz de capturar os padrões necessários de forma eficiente, beneficiando-se de:

- Uma estrutura simples que evita o overfitting.
- Uma boa capacidade de generalização auxiliada por regularização e dropout.
- A ausência de problemas como desvanecimento de gradiente, que pode ocorrer em redes mais profundas.

Essa análise reforça que, ao projetar redes neurais, **mais profundo nem sempre é melhor**. A escolha do número de camadas e neurônios deve sempre ser balanceada com a complexidade dos dados e o objetivo da tarefa. Em tarefas mais simples, arquiteturas menores e mais simples muitas vezes superam as mais complexas em termos de eficiência e generalização.