Ariel A. O. Menezes

# Relatório Redes Perceptron

Este projeto foi desenvolvido em Python 3.11.12 em um sistema com as seguintes configurações:

- **OS**: NixOS 24.11
- **Placa-Mãe**: Gigabyte B550M K
- **CPU**: AMD Ryzen 5 5500 (12) 4.2
- **GPU**: NVIDIA GeForce RTX 4060

## Exercício 1: Iris Data Set

Este exercício utilizou o conjunto de dados Iris, composto por 150 amostras de flores divididas entre três espécies: setosa, versicolor e virginiana.

A tarefa consiste em classificar corretamente a espécie da flor com base nesses atributos, utilizando uma rede neural do tipo perceptron com as seguintes configurações:

- Uma camada de entrada com 4 entradas, correspondendo aos quatro atributos do conjunto.
- Uma camada de saída com 3 neurônios, representando as três classes alvo.
- Com 100 épocas de treinamento.

A variável de classe foi transformada para o formato one-hot encoded com três posições, permitindo que cada neurônio de saída modele a probabilidade de uma das espécies. Para isso, a camada de saída utilizou a função de ativação softmax, gerando uma distribuição de probabilidade entre as três classes. A função de perda adotada foi a entropia cruzada categórica.

O conjunto de dados foi particionado da seguinte forma: 70% das amostras foram utilizadas para treinamento, 15% para validação durante o treino, e os 15% restantes para o teste final. O processo de inferência da rede foi realizado via propagação direta (forward), sem camadas ocultas.

A metodologia experimental foi dividida em dois testes principais. Cada teste foi projetado para isolar e avaliar o impacto de um único fator: no primeiro, a influência dos pesos iniciais; no segundo, o efeito da taxa de aprendizado. Ambos os testes mantiveram controle sobre as demais variáveis, e cada configuração foi executada 10 vezes para permitir análise estatística confiável dos resultados.

## Teste 1: O efeito dos pesos iniciais sobre o modelo

Para investigar a influência dos valores iniciais dos pesos no desempenho final da rede, foi conduzido um experimento sistemático em que esses valores foram fixados em diferentes constantes e avaliados sob as mesmas condições de treinamento.

Toda a estrutura de execução foi organizada no diretório **Reports/test_1/**. Dentro dele, foram criadas subpastas com o padrão **weight_N/**, onde N representa o valor específico de inicialização dos pesos, variando de 0.1 a 1.0, em incrementos de 0.1. Além disso, foi criado uma subpasta adicional chamada **weight_random/**, representando o cenário em que os pesos iniciais foram definidos com valores aleatórios para cada peso de cada entrada.

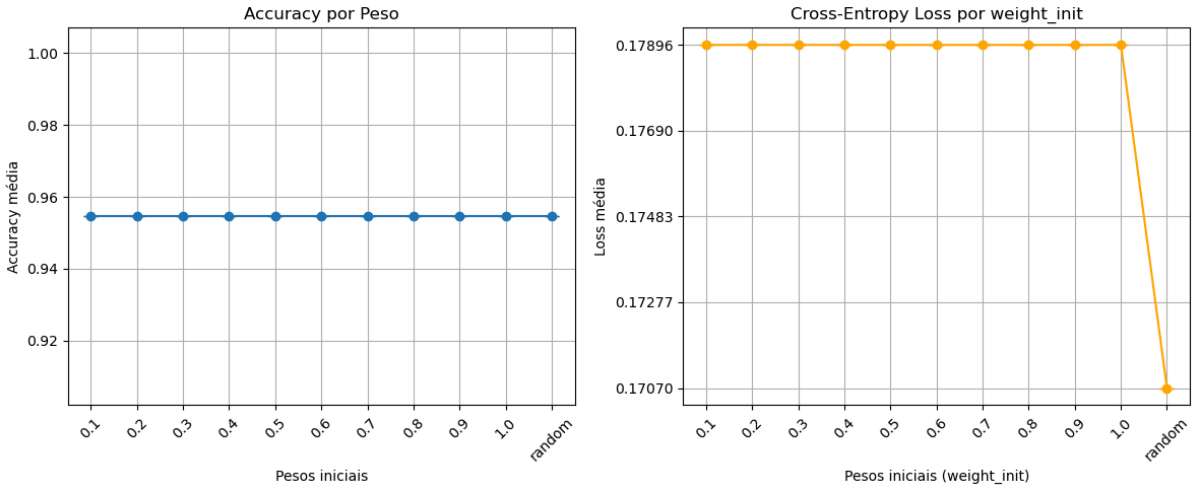
Em cada pasta **weight_N/** e na pasta **weight_random/**, foram executadas 10 rodadas independentes de treinamento e teste, armazenadas como **run_0/** até **run_9/**. Cada uma dessas rodadas contém os seguintes arquivos json:

- Um arquivo contendo o modelo completo ao final de todas as épocas de treinamento.
- Um arquivo com o modelo da época considerado ideal com base no desempenho de validação.
- Um relatório com as métricas de desempenho sobre o conjunto de teste, contendo valores médios de perda (usando entropia cruzada e erro quadrático médio), acurácia e total de amostras válidas, além das predições realizadas – incluindo a classe real, a classe predita e as probabilidades atribuídas a cada classe.
- Um log das validações realizadas ao longo do treinamento, contendo, para cada época, os valores de perda por entropia cruzada, erro quadrático médio e perda total.

Os arquivos contendo os modelos (tanto o final quanto o da melhor época) registram a estrutura da rede, incluindo número de perceptrons, número de entradas, presença de bias, e os pesos finais após o treinamento.

Essa organização metodológica permitiu a coleta sistemática de dados para posterior análise do impacto da inicialização dos pesos no processo de aprendizado e na capacidade de generalização de redes.

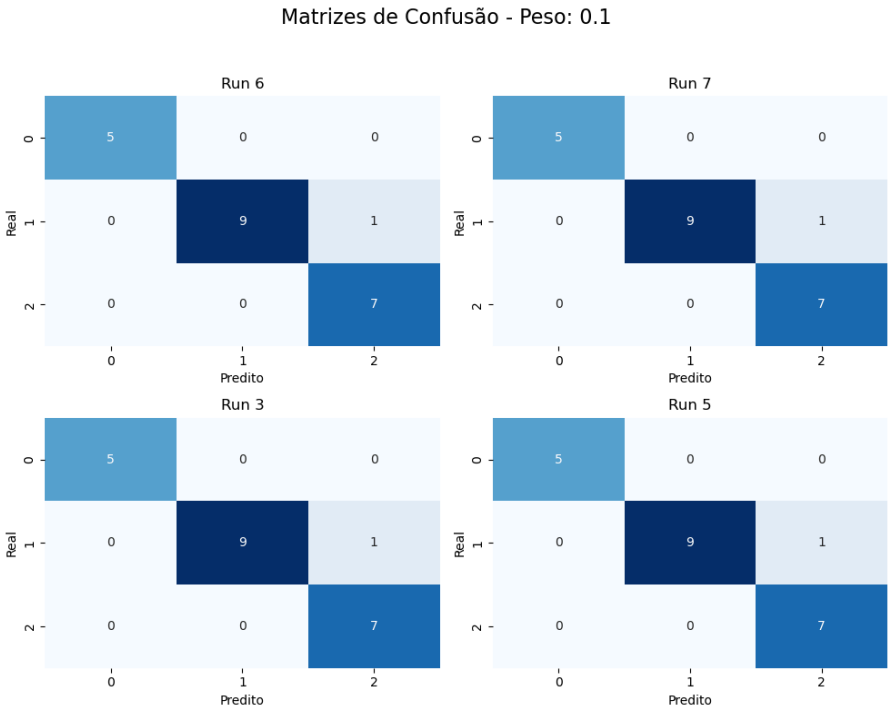
Com a taxa de aprendizado fixa em 0.1 para todas as iterações do experimento podemos concluir com base nos dados contidos no gráfico abaixo que o valor inicial para os pesos não possui diferença significativa na acurácia do modelo final.



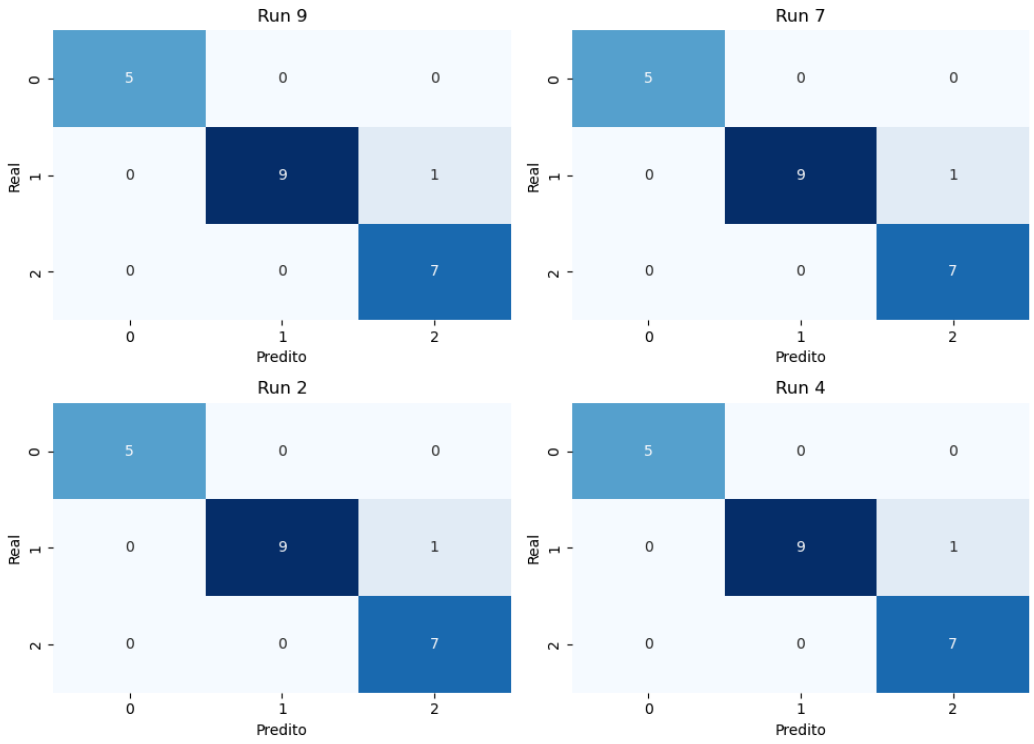

O modelo apresentou uma acurácia constante de 0.954545 em todas as 10 execuções para cada valor de peso inicial, tanto nos pesos definidos quanto no aleatoriamente gerado.

Em relação à perda por entropia cruzada (cross-entropy loss), o peso inicial aleatório obteve uma leve vantagem, com um valor médio ligeiramente menor. No entanto, essa diferença não é estatisticamente significativa, já que os valores se mantêm dentro de um intervalo estreito de 0.170 a 0.179, variando apenas 0.009 no total.

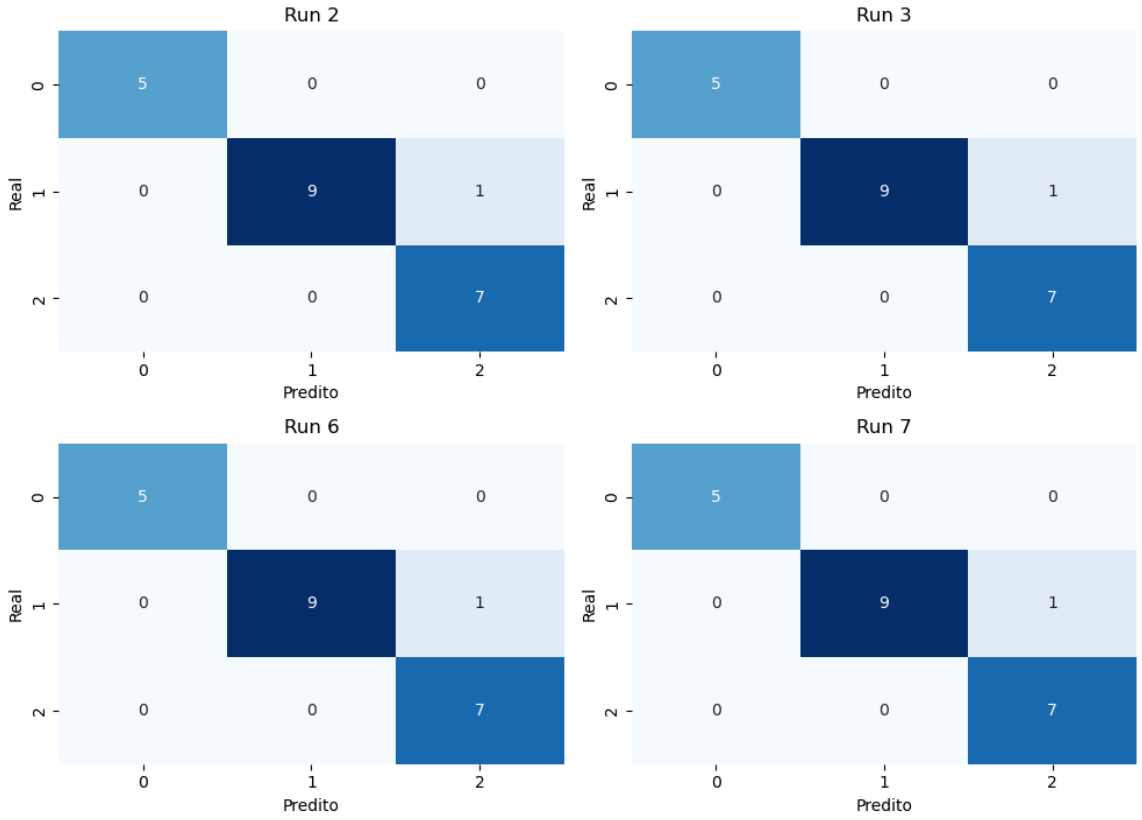
As imagens a seguir apresentam as matrizes de confusão de 4 execuções selecionadas aleatoriamente para cada valor de peso inicial utilizado no experimento. Observa-se que o modelo manteve um bom desempenho preditivo, independentemente da variação dos pesos iniciais ou do fato de cada configuração ter sido executada 10 vezes. Isso sugere uma boa estabilidade e robustez do modelo frente às variações na inicialização.



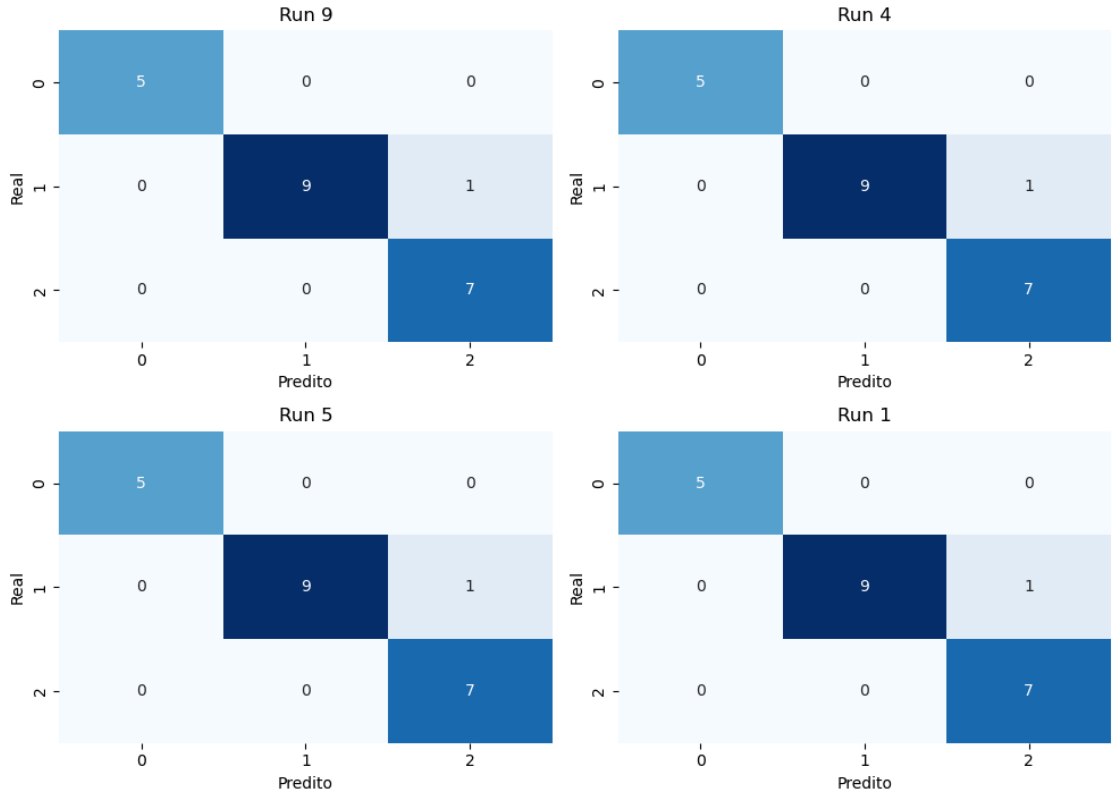
## Matrizes de Confusão - Peso: 0.2



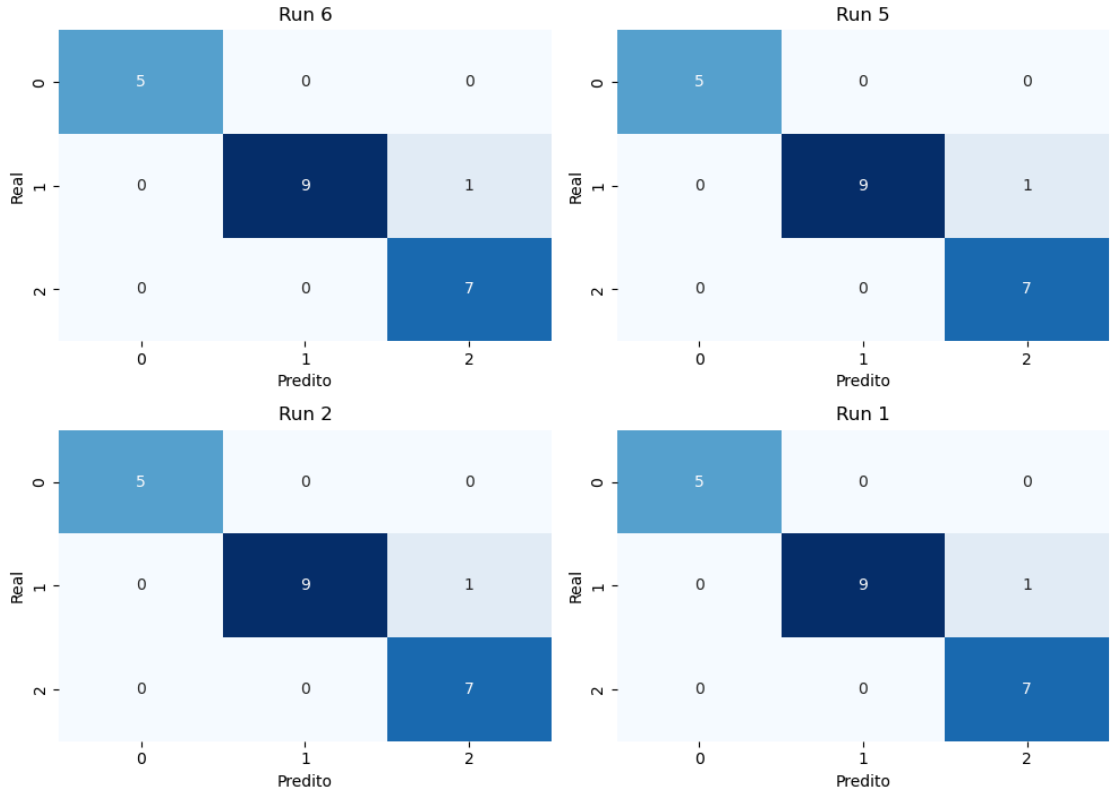

## Matrizes de Confusão - Peso: 0.3



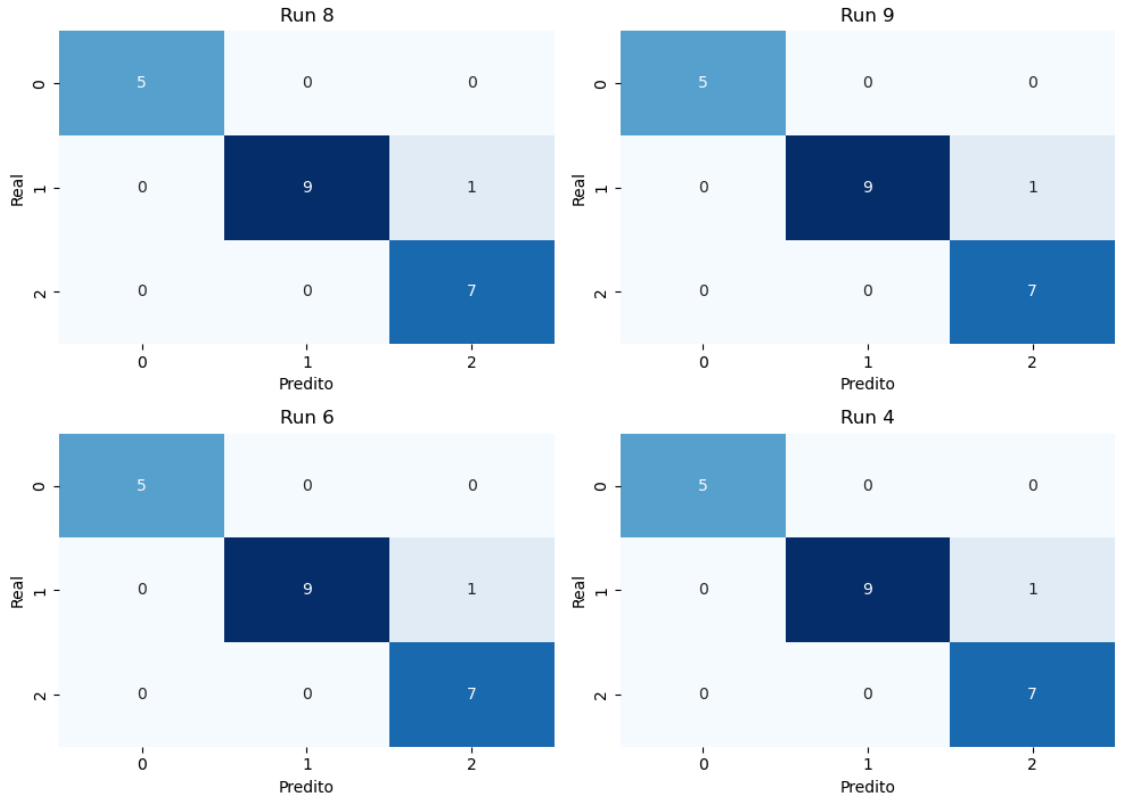
## Matrizes de Confusão - Peso: 0.4



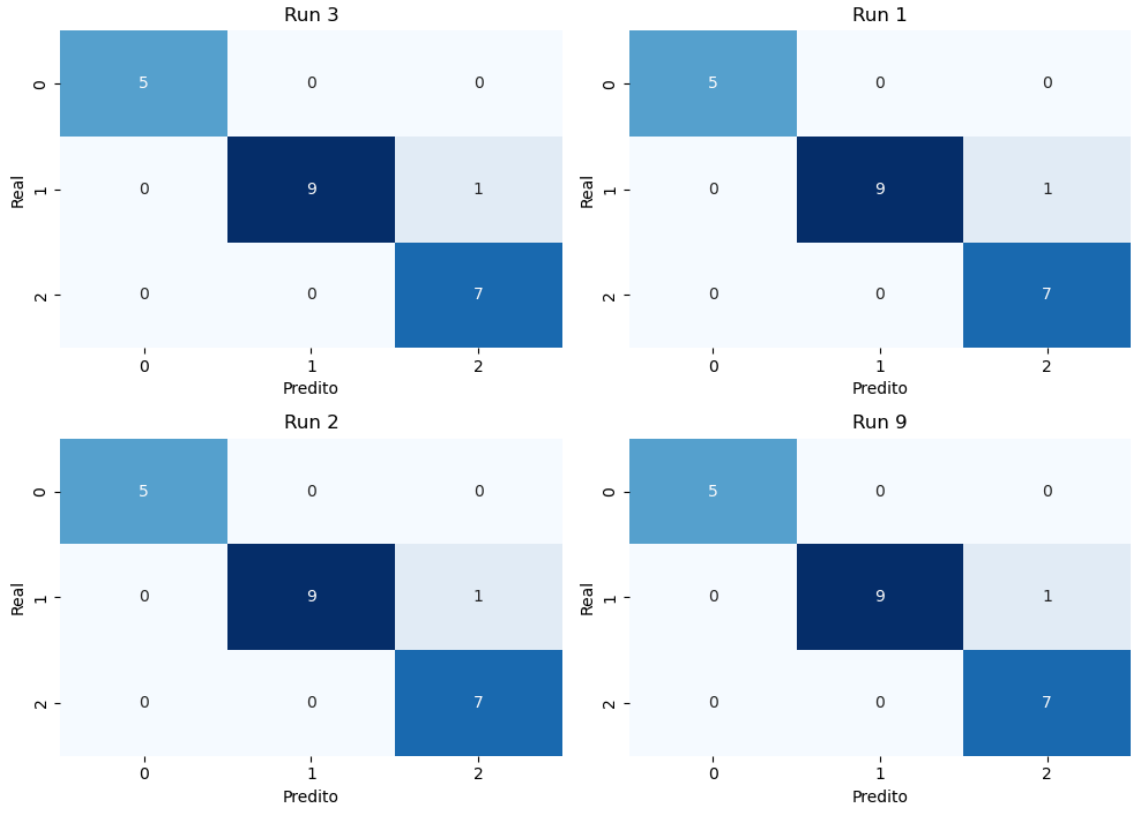

## Matrizes de Confusão - Peso: 0.5



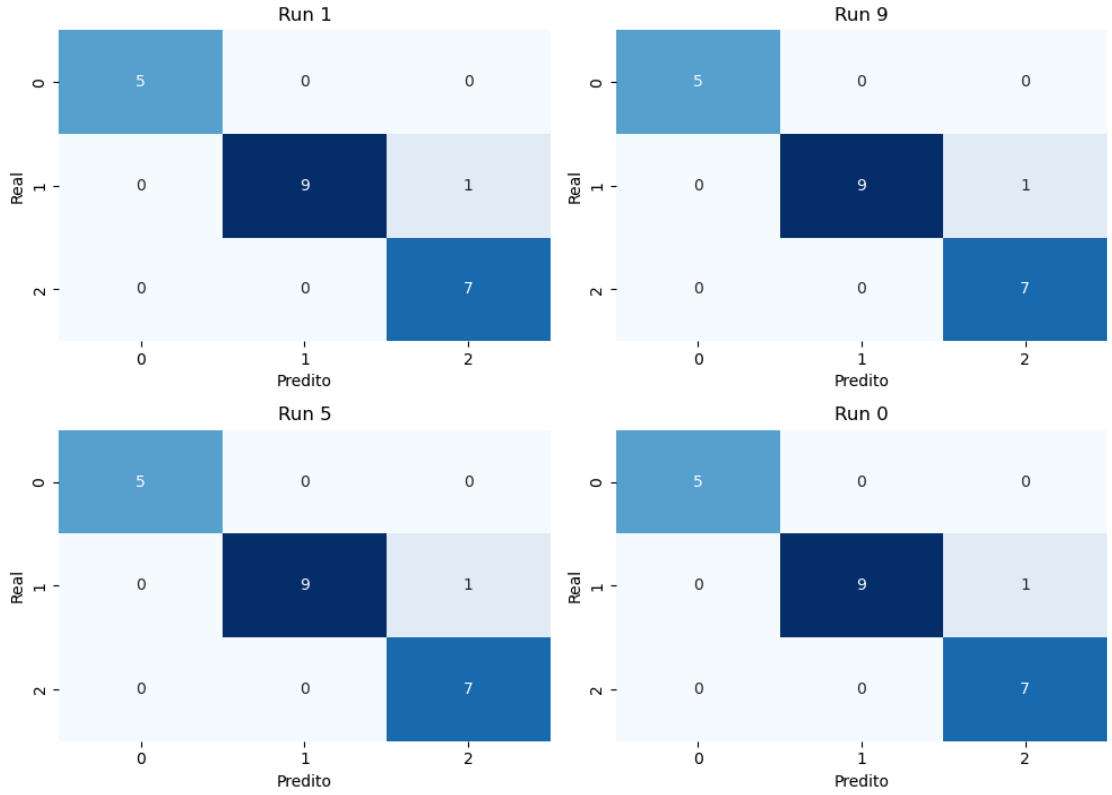
## Matrizes de Confusão - Peso: 0.6



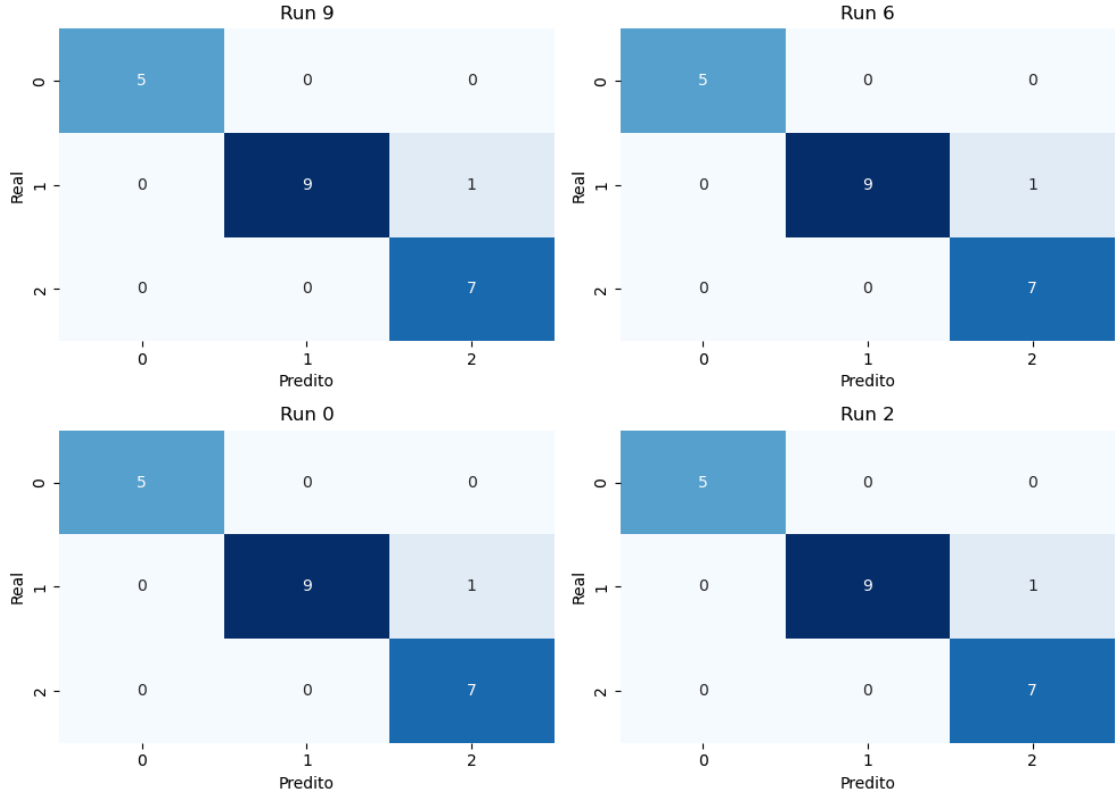

## Matrizes de Confusão - Peso: 0.7



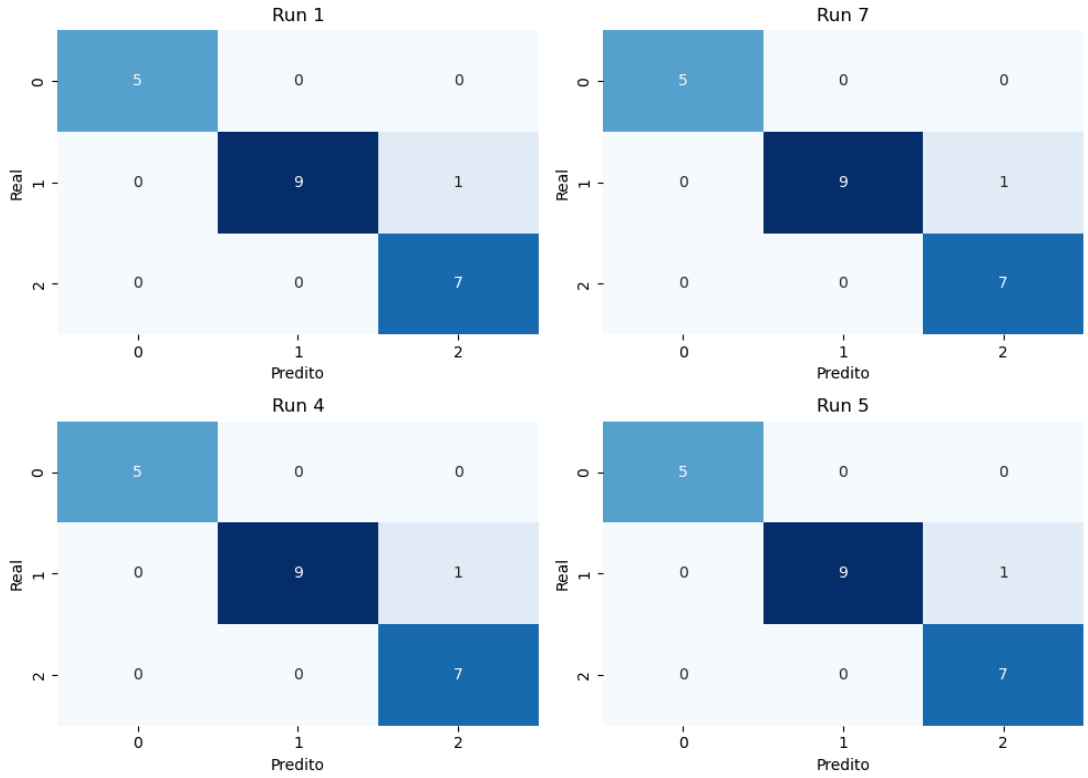
## Matrizes de Confusão - Peso: 0.8



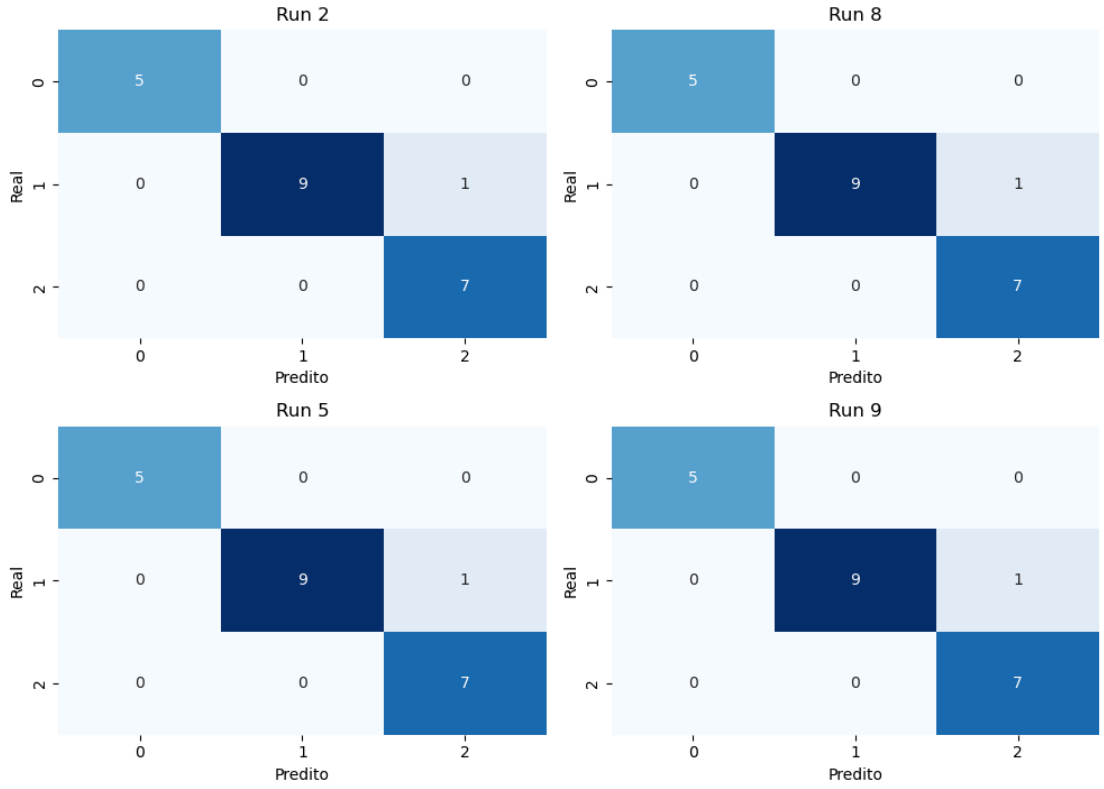

## Matrizes de Confusão - Peso: 0.9



## Matrizes de Confusão - Peso: 1.0




## Matrizes de Confusão - Peso: random



## Teste 2: O efeito da taxa de aprendizado sobre o modelo

Para investigar a influência da taxa de aprendizado no desempenho final da rede, foi conduzido um experimento sistemático em que os pesos iniciais foram mantidos fixos em 0.5, e diferentes valores de taxa de aprendizado foram testados sob as mesmas condições de treinamento.

Toda a estrutura de execução foi organizada em diretório **Reports/test_2/.** Dentro dele, foram criadas subpastas com o padrão **lr_X_Y/** onde **X_Y** representa o valor específico da taxa de aprendizado, variando de 0.01 a 0.1 com incrementos de 0.01 e depois variando de 0.1 a 1.0 com incrementos de 0.1. Cada pasta corresponde a um experimento com uma configuração distinta de taxa de aprendizado aplicada ao modelo.

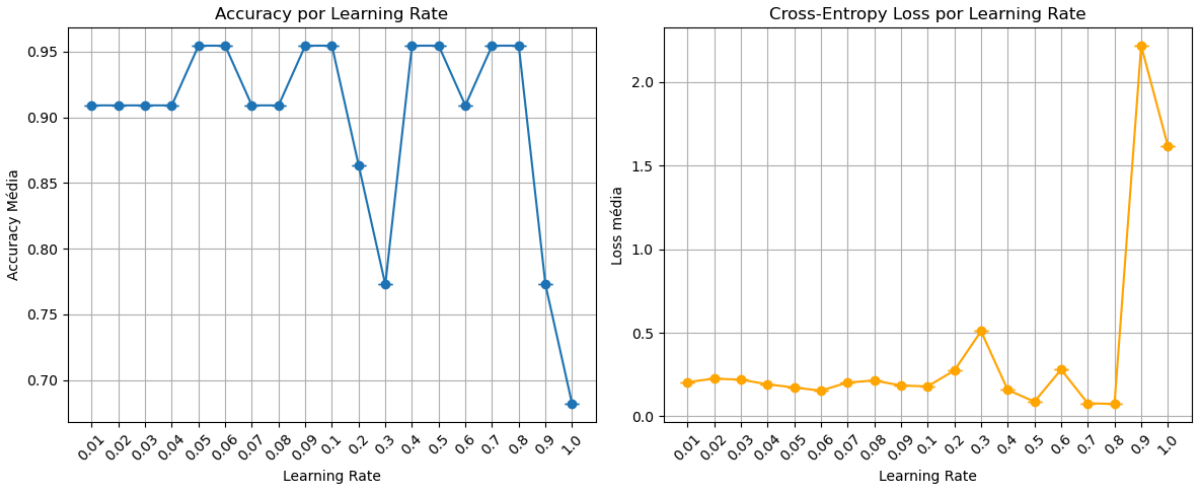
Em cada uma dessas pastas (**lr_X_Y/**), foram executadas 10 rodadas independentes de treinamento e teste, armazenadas como **run_0/** até **run_9/**. Cada uma dessas rodadas contém os seguintes arquivos **.json**:

- Um arquivo contendo o modelo completo ao final de todas as épocas de treinamento.
- Um arquivo com o modelo da época considerado ideal com base no desempenho de validação.
- Um relatório com as métricas de desempenho sobre o conjunto de teste, contendo valores médios de perda (usando entropia cruzada e erro quadrático médio), acurácia e total de amostras válidas, além das predições realizadas - incluindo a classe real, a classe predita e as probabilidades atribuídas a cada classe.
- Um log das validações realizadas ao longo do treinamento, contendo, para cada época, os valores de perda por entropia cruzada, erro quadrático médio e perda total.

Os arquivos contendo os modelos (tanto o final quanto o da melhor época) registram a estrutura da rede, incluindo o número de perceptrons, número de entradas, presença de bias, e os pesos finais após o treinamento.

Essa organização metodológica permitiu a coleta de dados para posterior análise do impacto da taxa de aprendizado no processo de aprendizado e na capacidade de generalização das redes.

A figura abaixo mostra a relação entre a taxa de aprendizado e a acurácia e a taxa de aprendizado e a perda por entropia cruzada.



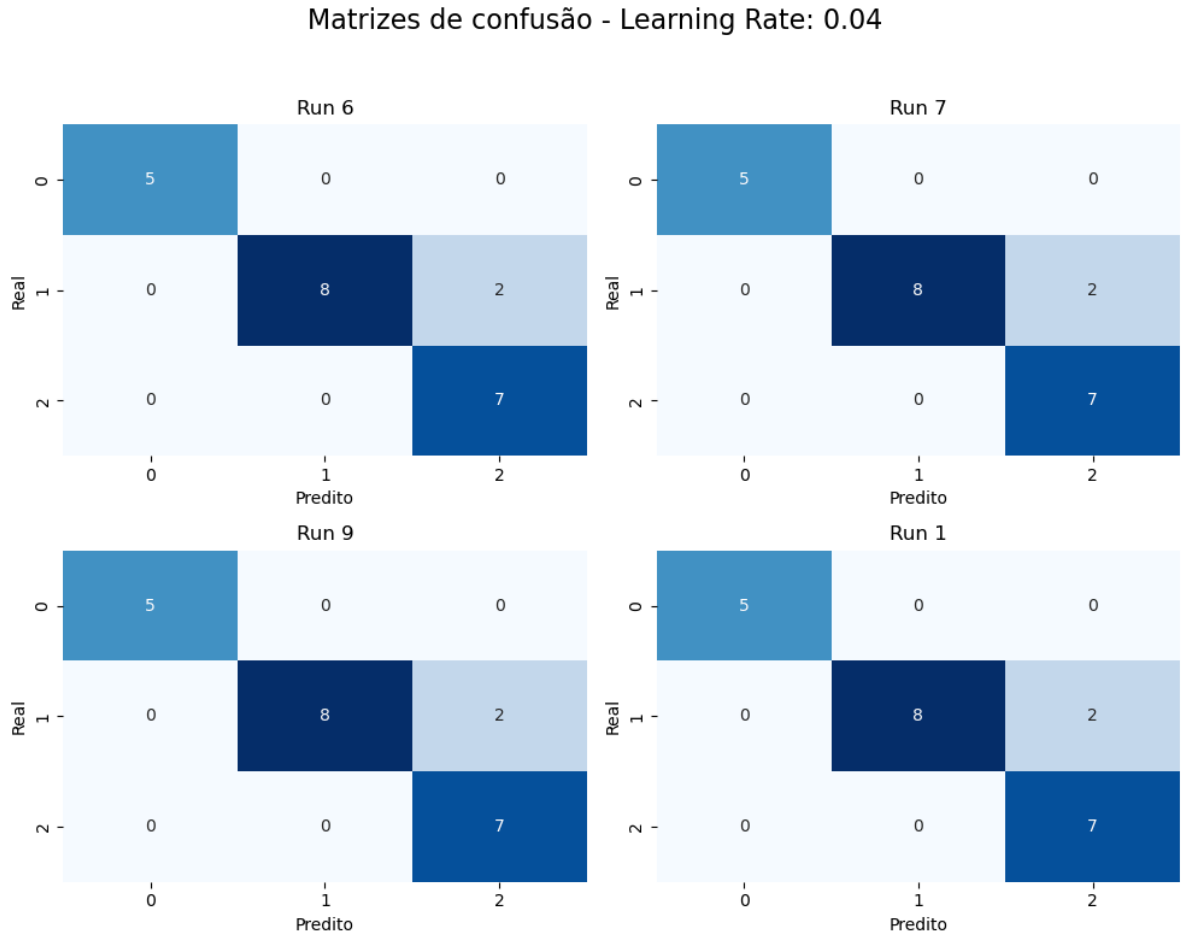

A figura apresenta a acurácia média e a perda média por entropia cruzada do modelo em função de diferentes valores de taxa de aprendizado (**learning rate**), variando de 0.01 a 0.1 em acréscimos de 0.01 e depois de 0.1 a 1.0 em acréscimos de 0.1. Observe-se que, para valores de **learning rate** entre 0.01 e 0.01, o modelo apresenta um desempenho estável e satisfatório, com acurácias em torno de 91% e perdas consistentemente baixas. Esses resultados indicam que, dentro dessa faixa, a rede consegue aprender de forma eficiente, com atualizações de pesos suficientemente pequenas para garantir a convergência adequada do modelo.

A partir de **lr = 0.1,** observa-se um ligeiro aumento na variabilidade dos resultados. Em **lr=0.3**, por exemplo, há uma queda abrupta na acurácia (cerca de 77%) acompanhada de um aumento considerável na perda, sugerindo instabilidade no processo de aprendizado. Essa instabilidade pode estar relacionada ao fenômeno de overshooting, em que os passos de atualização dos pesos se tornam grandes demais e passam a oscilar ao redor do mínimo da função de erro, dificultando a convergência. No entanto, de maneira inesperada, o desempenho volta a melhorar em **lr=0.4** e permanece elevado em **lr=0.6, 0.7 e 0.8**, com acurácias superiores a 95%e perda significativamente baixas. Isso indica que existem múltiplas regiões inicial de que o desempenho cairia de forma progressiva a partir de um determinado ponto.

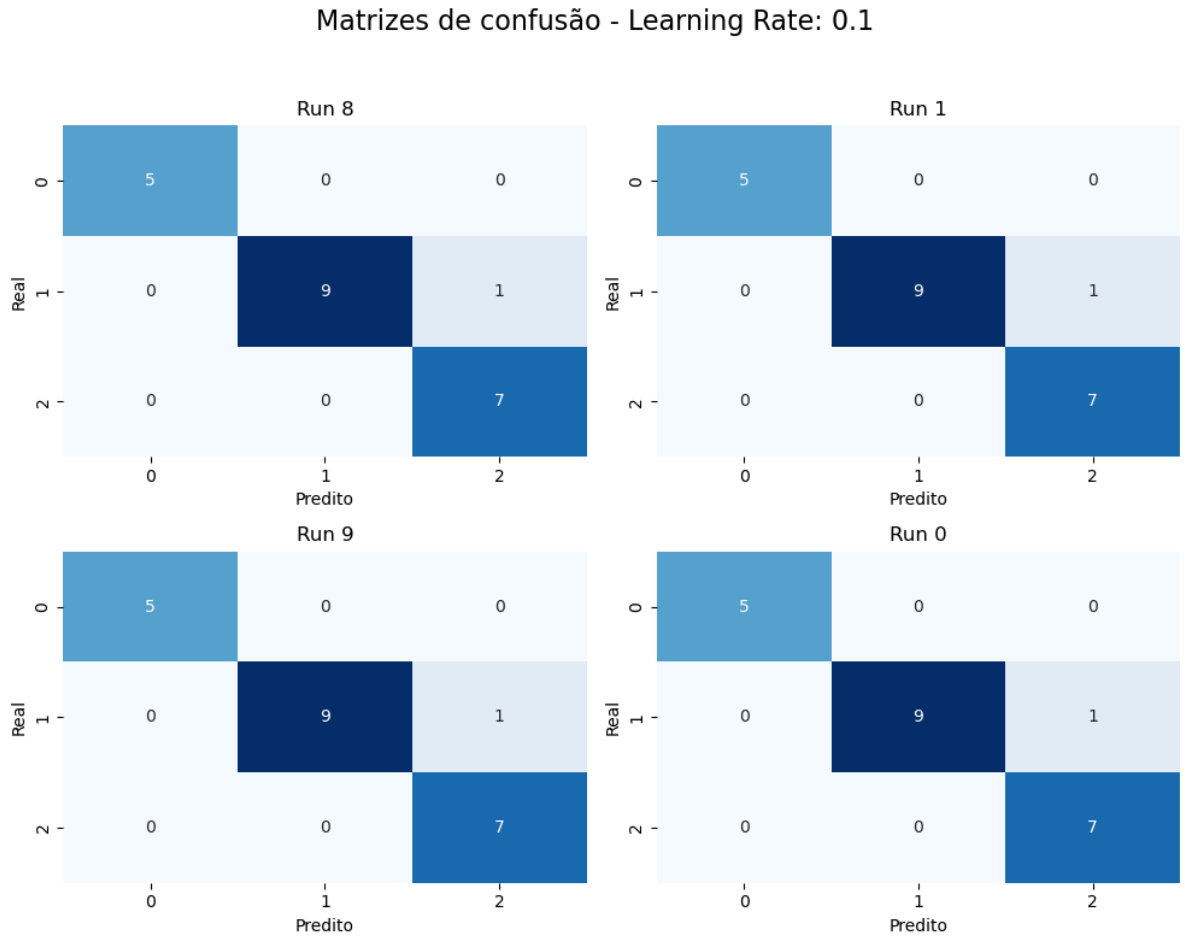
Por outro lado, taxas de aprendizado muito elevadas, como 0.9 e 1.0, levam a um comportamento divergente do modelo. A acurácia nesses pontos cai acentuadamente (abaixo de 80% e 70%, respectivamente), e a perda média apresenta crescimento abrupto, ultrapassando 2.0 em **lr=0.9.** esse padrão indica falhas de convergência durante o treinamento, possivelmente devido a passos de atualização demasiadamente agressivos, que impedem que o modelo se estabilize em uma solução minimamente aceitável.

Esses resultados reforçam a importância de uma escolha criteriosa da taxa de aprendizado. Embora valores muito baixos possam levar a um aprendizado lento, valores excessivamente altos podem comprometer completamente a capacidade da rede de generalizar.

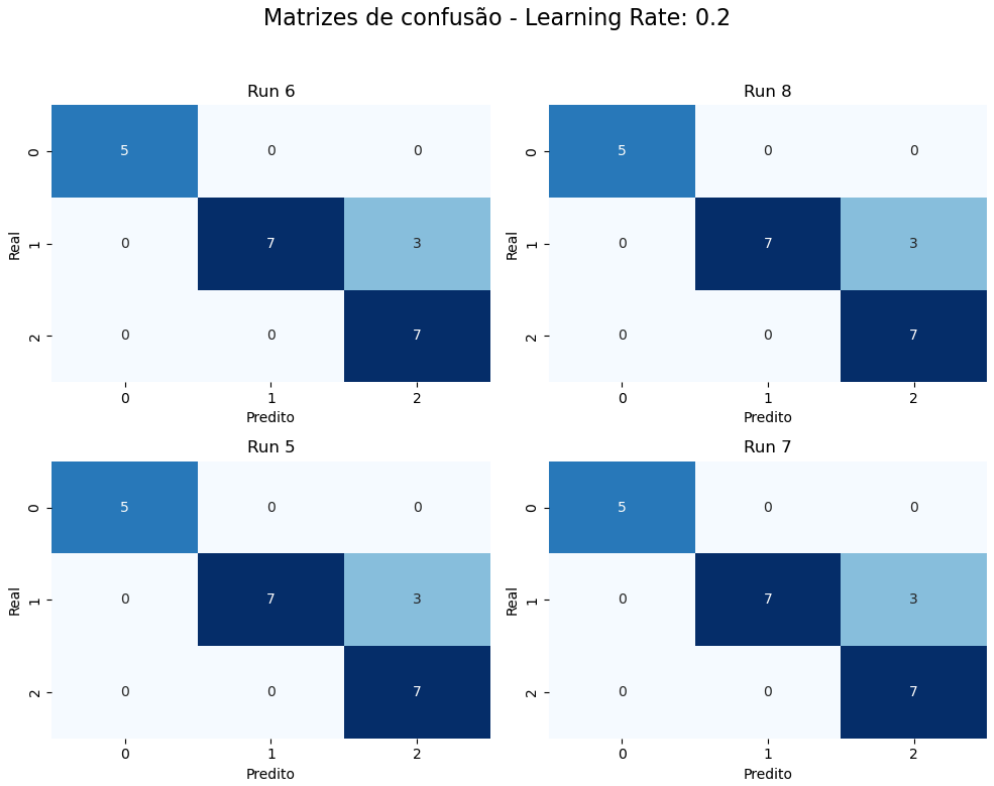
Abaixo temos as matrizes de confusão de algumas taxas de aprendizado. Foi selecionado aleatoriamente runs para compor cada imagem, demonstrando as matrizes de 4 runs aleatórias para cada matriz de confusão. Por exemplo, a imagem abaixo demonstra a capacidade do modelo para uma taxa de aprendizado de 0.04, mas ela também representa todos os modelos que ficaram em um limite de acurácia de 0.90 a 0.95.

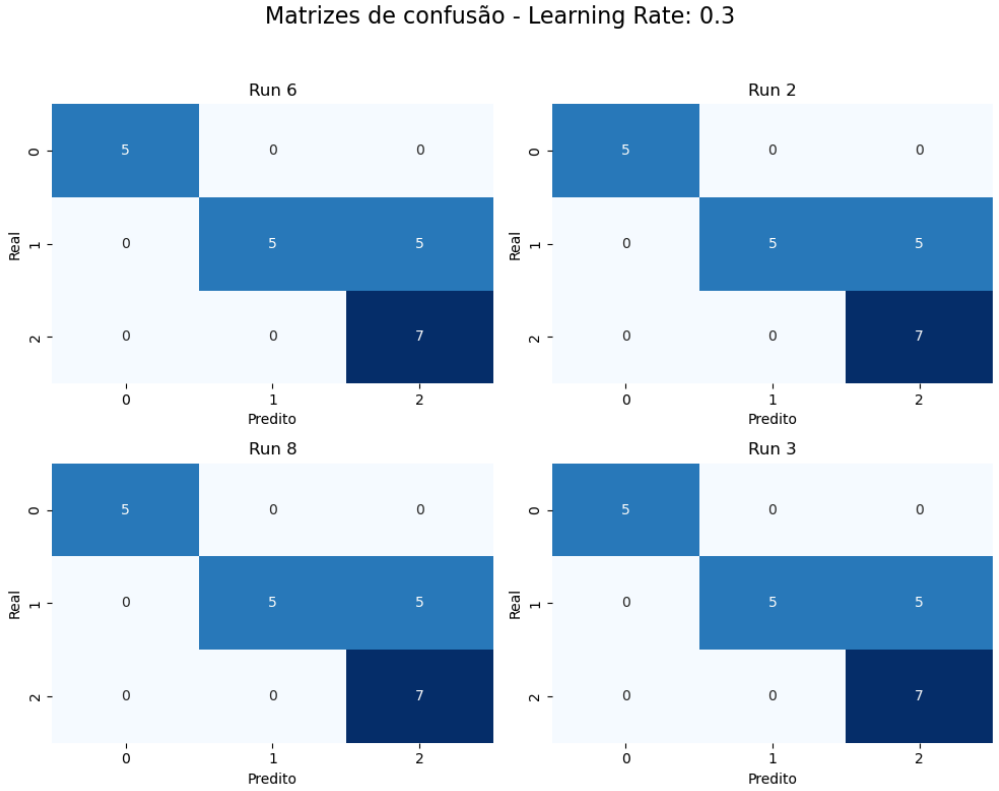


Já a imagem que se segue demonstra as capacidades do modelo para uma taxa de aprendizado de 0.1, mas pode facilmente representar as taxas de aprendizado que obtiveram uma acurácia acima de 0.95 como demonstra a imagem de acurácia e perda por entropia cruzada.

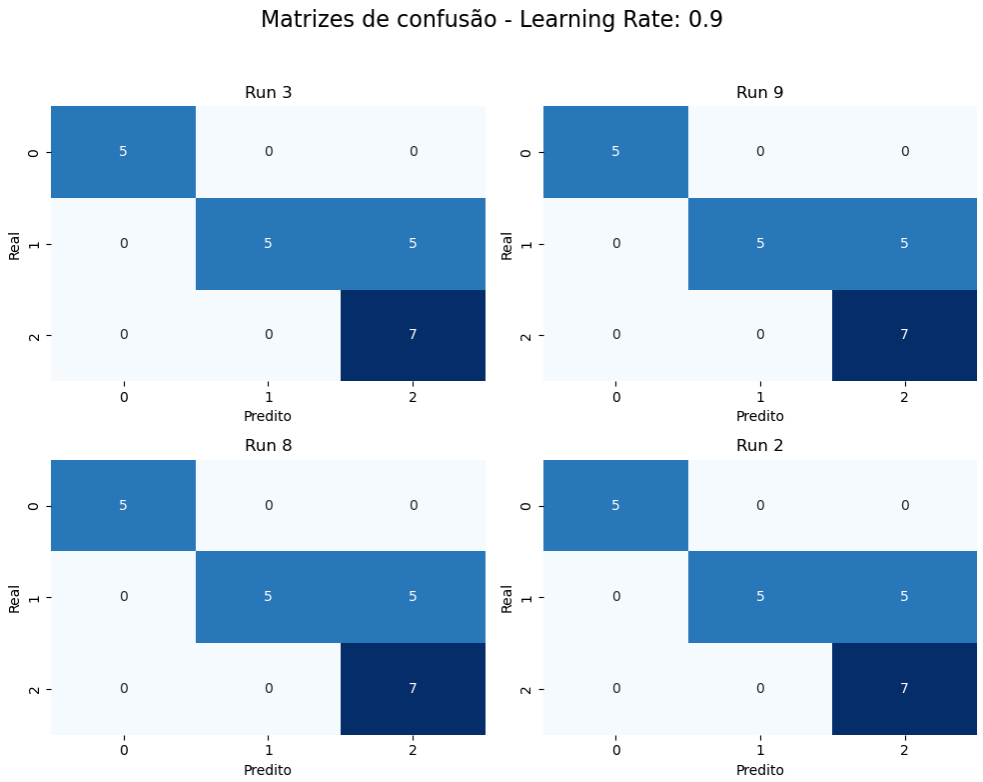


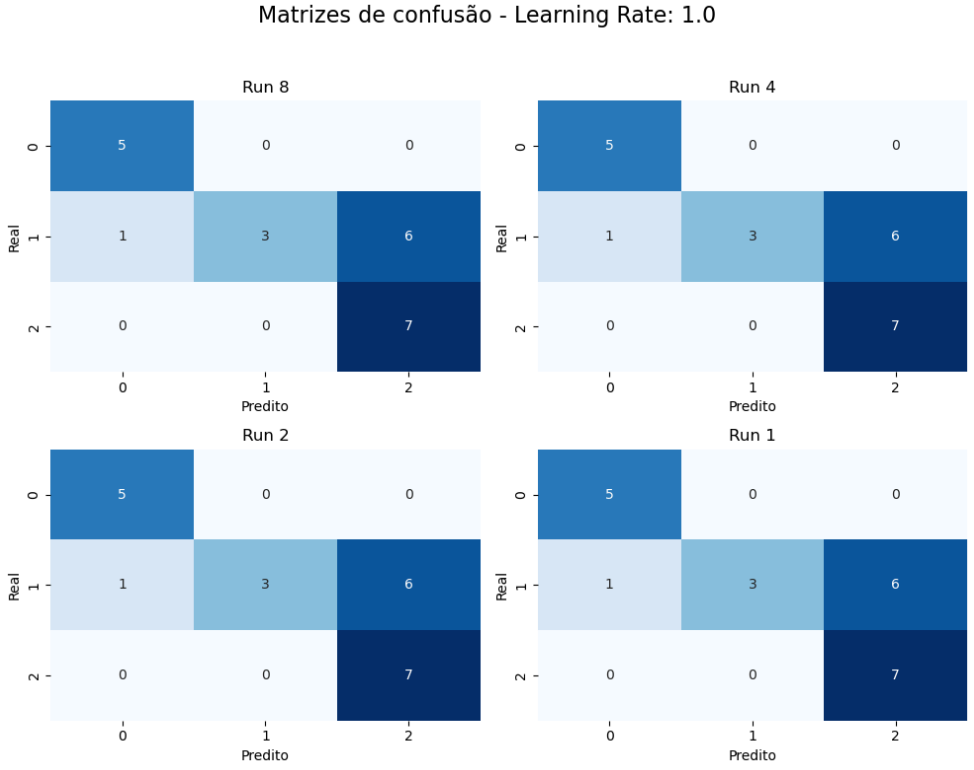
Já as imagens a seguir são referentes às matrizes de confusão dos modelos que utilizaram as taxas de aprendizado de 0.2 e 0.3, que ocorreu uma queda de acurácia de 0.86 para 0.76 respectivamente. Podemos observar que o modelo teve uma performance um pouco pior que os anteriormente apresentados.



Por fim, temos aqui a matriz de confusão dos modelos que utilizaram taxa de aprendizado de 0.9 e 1.0. Podemos observar que um passo muito grande no gradiente entre iterações não colaborou para uma boa convergência, fazendo com que o modelo tivesse uma acurácia pobre de respectivamente 0.76 e 0.66.



Por fim, podemos concluir que mesmo alguns modelos tendo acurácias abaixo de 0.85, as predições erradas feitas ocorreram em relação às Iris virginianas e versicolor que possuem parâmetros parecidos entre elas e que se diferem mais da Iris setosa.

# Exercício 2: Wine Data Set

Este exercício utilizou o conjunto de dados Wine, composto por 178 amostras de vinhos, divididas entre três classes, cada uma representando uma cultura diferente de uva (classe 1, 2 e 3), originárias da região da Itália.

Cada amostras é descrita por 13 atributos contínuos que representam medidas químicas e físicas do vinho, como teor alcoólico, acidez, cor e concentração de fenóis.

A tarefa consiste em classificar corretamente o tipo de vinho com base nesses atributos, utilizando uma rede neural do tipo perceptron com as seguintes configurações:

- Uma camada de entrada com 13 entradas, correspondendo aos treze atributos do conjunto.
- Uma camada de entrada com 13 entradas, correspondendo aos treze atributos do conjunto.
- Uma camada de saída com 3 neurônios, representando as três classes alvos.
- Com 100 épocas de treinamento.

A variável de classe foi convertida para o formato one-hot encoded com três posições, permitindo que cada neurônio de saída modelasse a probabilidade de uma das classes. Para isso, a camada de saída utilizou a função de ativação softmax, gerando uma distribuição de probabilidade entre três classes. A função de perda adotada foi a entropia cruzada categórica.

O conjunto de dados foi particionado da seguinte forma: 70% das amostras foram utilizadas para treinamento, 15% para validação durante o treinamento, e os 15% restantes para o teste final. O processo de inferência da rede foi realizado via propagação direta (forward), sem camadas ocultas.

A metodologia experimental foi dividida em dois testes principais. Cada teste foi projetado para isolar e avaliar o impacto de um único fator: no primeiro, a influência dos pesos iniciais; no segundo, o feito da taxa de aprendizado. Ambos os testes mantiveram controle sobre as demais variáveis, e cada configuração foi executada 10 vezes para permitir análise estatísticas confiáveis dos resultados.

O ponto divergente deste experimento para o anterior, com o conjunto de dados Iris, foi a normalização dos dados entre os limites -1 e 1. O objetivo foi verificar se essa normalização impactaria diretamente os experimentos, tanto o teste com pesos iniciais variados, quanto o teste com taxas de aprendizado diferentes.

## Teste 1: O efeito dos pesos iniciais sobre o modelo

Para investigar a influência dos valores iniciais dos pesos no desempenho final da rede, foi conduzido um experimento sistemático em que esses valores foram fixados em diferentes constantes e avaliados sob as mesmas condições de treinamento.

Toda a estrutura de execução foi organizada no diretório **Reports/test_1/**. Dentro dele, foram criadas subpastas com o padrão **weight_N/**, onde N representa o valor específico de inicialização dos pesos, variando de 0.1 a 1.0, em incrementos de 0.1. Além disso, foi criado uma subpasta adicional chamada **weight_random/**, representando o cenário em que os pesos iniciais foram definidos com valores aleatórios para cada peso de cada entrada.

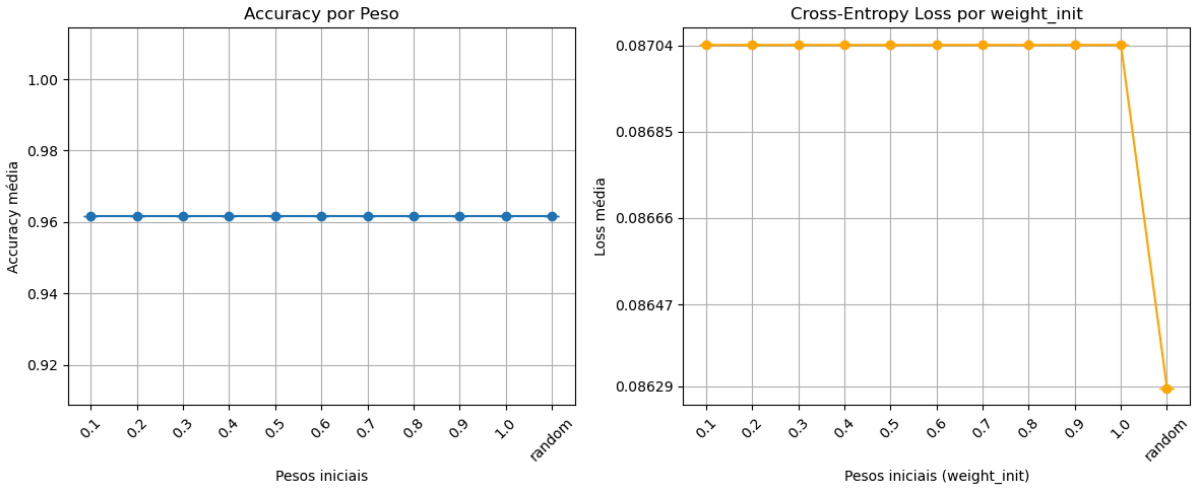
Em cada pasta **weight_N/** e na pasta **weight_random/**, foram executadas 10 rodadas independentes de treinamento e teste, armazenadas como **run_0/** até **run_9/**. Cada uma dessas rodadas contém os seguintes arquivos json:

- Um arquivo contendo o modelo completo ao final de todas as épocas de treinamento.
- Um arquivo com o modelo da época considerado ideal com base no desempenho de validação.
- Um relatório com as métricas de desempenho sobre o conjunto de teste, contendo valores médios de perda (usando entropia cruzada e erro quadrático médio), acurácia e total de amostras válidas, além das predições realizadas – incluindo a classe real, a classe predita e as probabilidades atribuídas a cada classe.
- Um log das validações realizadas ao longo do treinamento, contendo, para cada época, os valores de perda por entropia cruzada, erro quadrático médio e perda total.

Os arquivos contendo os modelos (tanto o final quanto o da melhor época) registram a estrutura da rede, incluindo número de perceptrons, número de entradas, presença de bias, e os pesos finais após o treinamento.

Essa organização metodológica permitiu a coleta sistemática de dados para posterior análise do impacto da inicialização dos pesos no processo de aprendizado e na capacidade de generalização de redes.

Com a taxa de aprendizado fixa em 0.1 para todas as iterações do experimento podemos concluir com base nos dados contidos no gráfico abaixo que o valor inicial para os pesos não possui diferença significativa na acurácia do modelo final mesmo que tenhamos mais entradas do que o experimento anterior. Mas podemos verificar que normalizar os dados interfere positivamente na acurácia e perda do modelo.

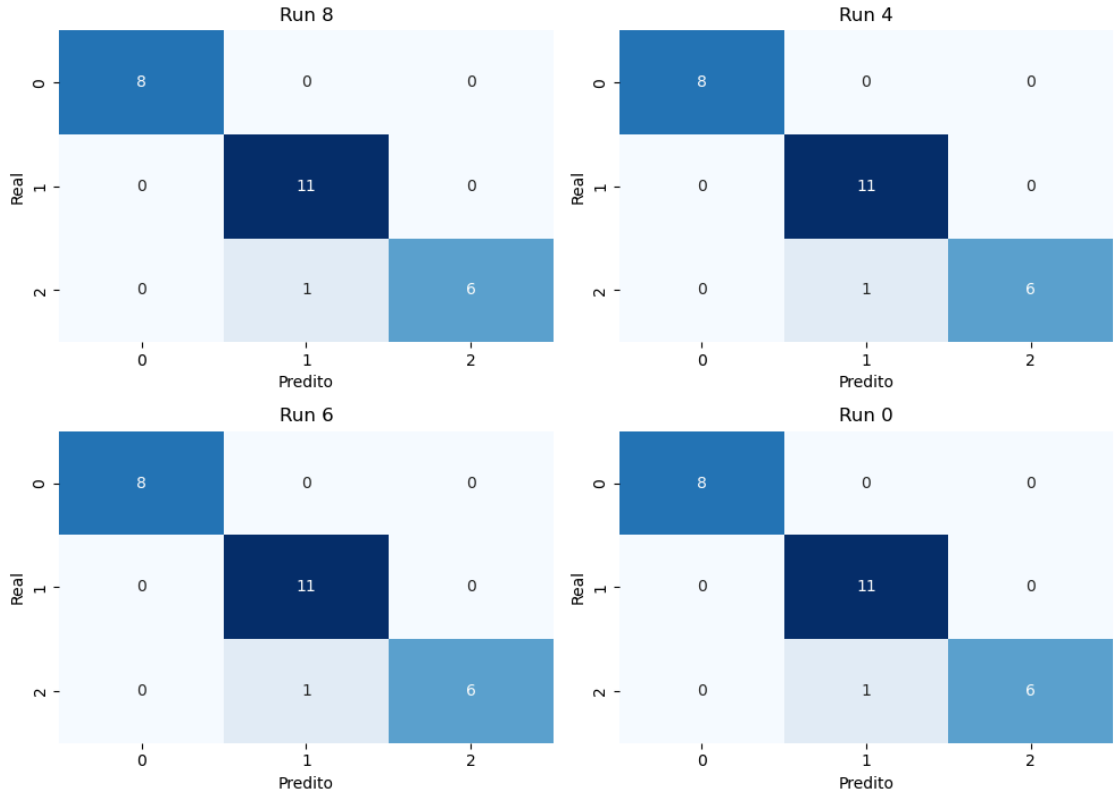



O modelo apresentou uma acurácia constante de 0.96 em todas as 10 execuções para cada valor de peso inicial, tanto nos pesos definidos manualmente quanto gerados aleatoriamente. Este comportamento indica que a normalização dos dados contribuiu para uma melhor performance e estabilidade do modelo, especialmente quando comparado ao experimento anterior com o conjunto de dados Iris, onde os dados não foram normalizados e a acurácia média ficou em 0.95. A padronização dos dados no intervalo de -1 a 1 parece ter reduzido o impacto das variações nos pesos iniciais, favorecendo uma convergência mais estável durante o treinamento.

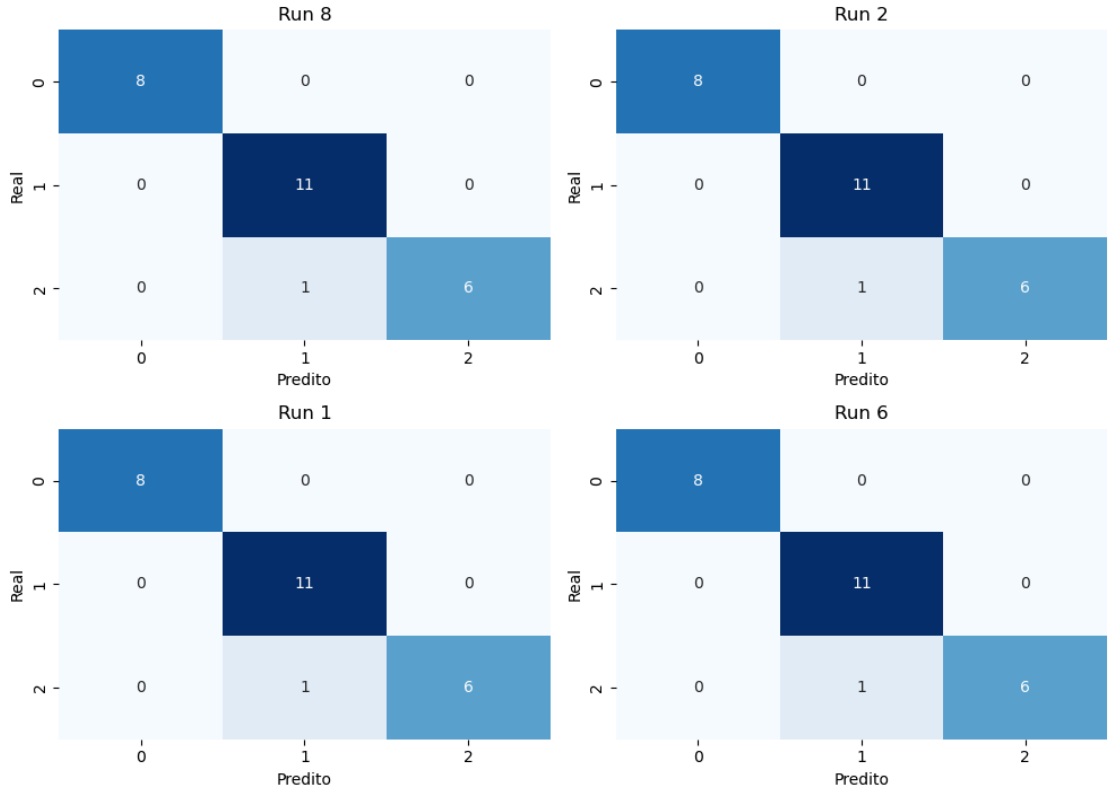
Em relação à perda por entropia cruzada (cross-entropy loss), os modelos com pesos aleatórios apresentaram uma leve vantagem, com perda média marginalmente inferior. Contudo, essa diferença não é estatisticamente significativa, já que os valores médios oscilaram em um intervalo bastante estreito, de 0.087 a 0.086 – uma variação de apenas 0.001. Importante notar que mesmo o maior valor de perda nesse experimento (0.087) ficou abaixo do menor valor observado no experimento anterior com o conjunto de dados Iris, o que reforça que, ainda que a configuração dos pesos iniciais tenha influência limitada, a normalização dos dados de entrada exerce um impacto expressivo na qualidade da convergência e nos desempenhos final do modelo.

As imagens a seguir apresentam as matrizes de confusão de 4 execuções selecionadas aleatoriamente para alguns valores de peso inicial utilizados no experimento. Observa-se que o modelo manteve um bom desempenho preditivo, independentemente da variação dos pesos iniciais ou do fato de cada configuração ter sido executada 10 vezes. Isso sugere uma boa estabilidade e robustez do modelo frente às variações na inicialização.

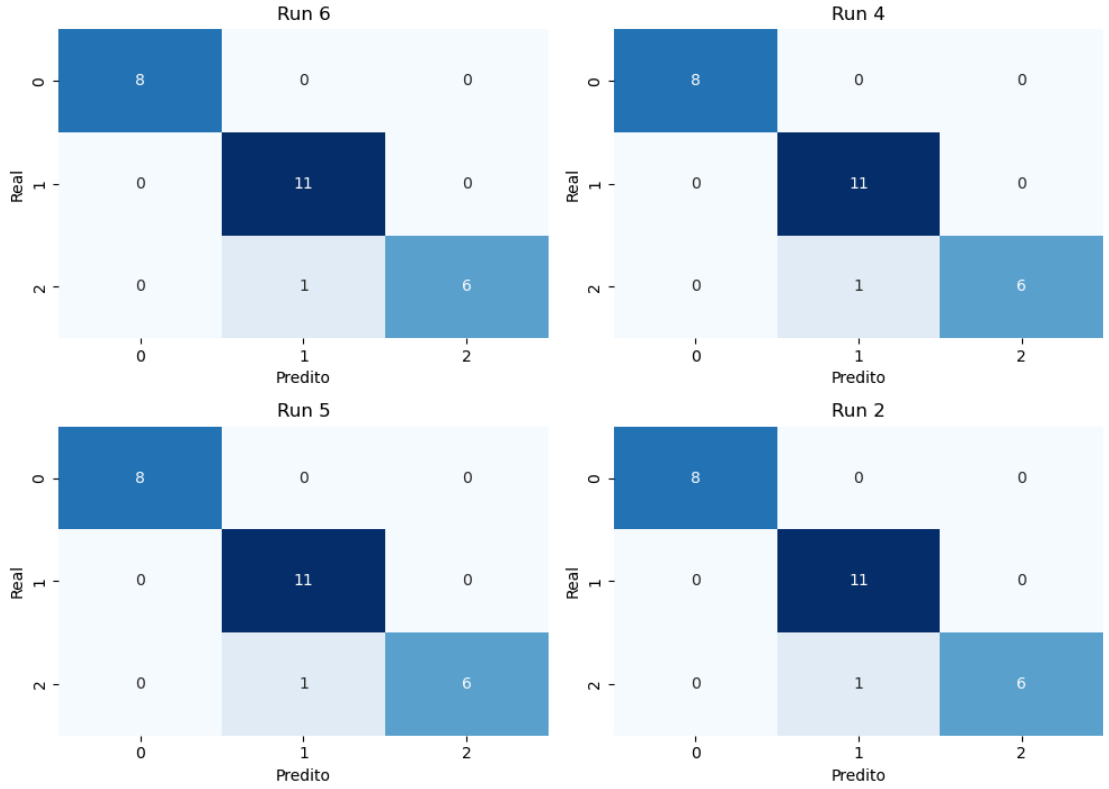
## Matrizes de Confusão - Peso: 0.1



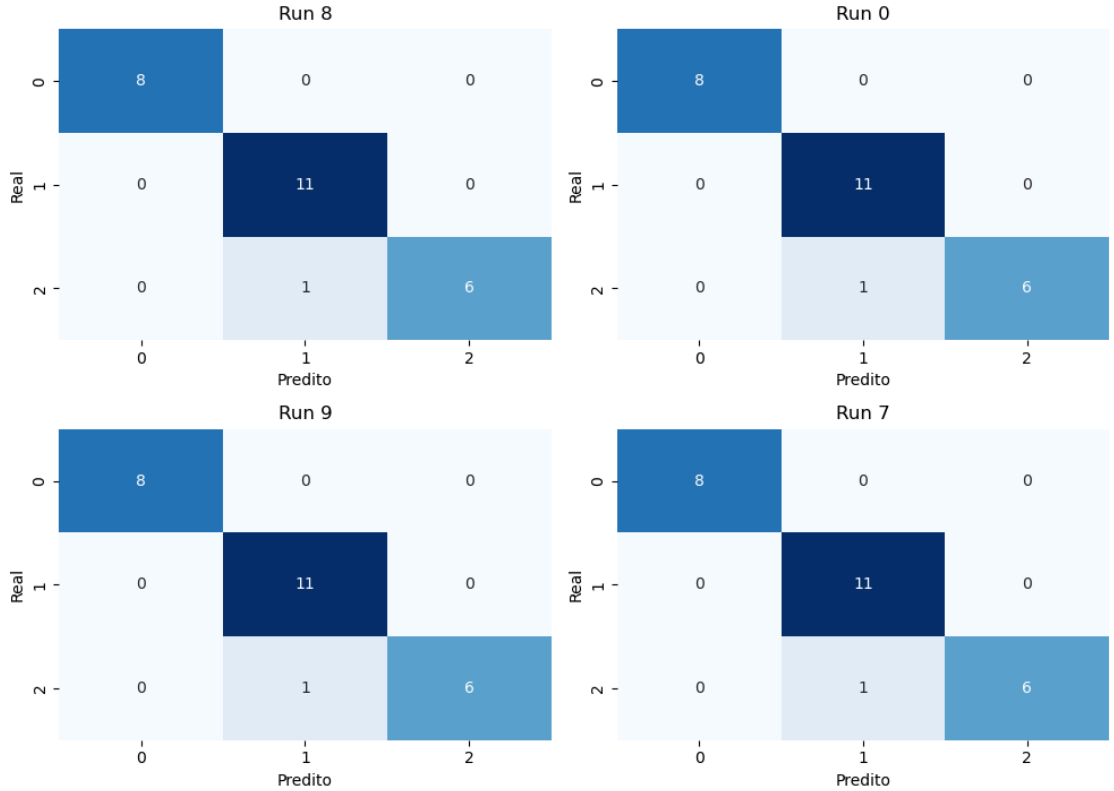

## Matrizes de Confusão - Peso: 0.4

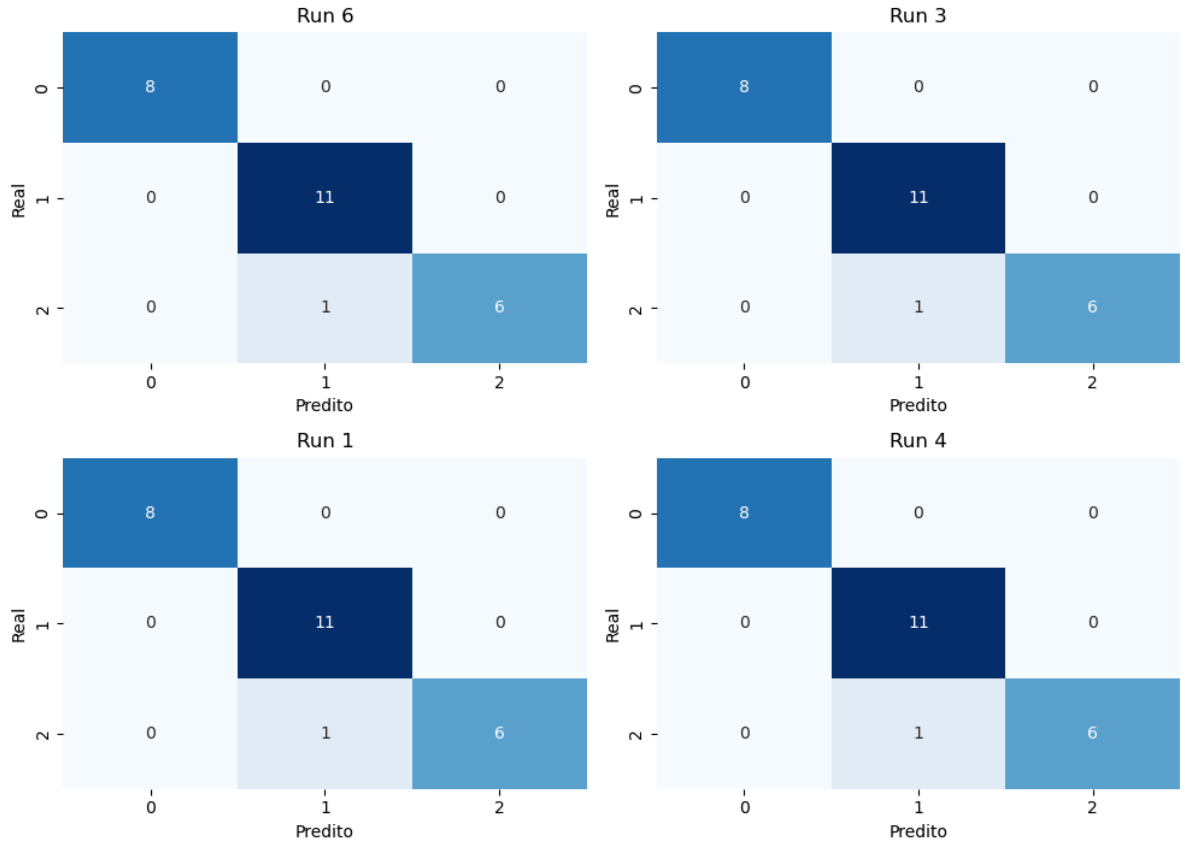


## Matrizes de Confusão - Peso: 0.7




## Matrizes de Confusão - Peso: 1.0




Matrizes de Confusão - Peso: random
Run 6
Run 3
Run 1
Run 4
Real
Predito
0
1
2
8
0
0
0
11
0
0
1
6

## Teste 2: O efeito da taxa de aprendizado sobre o modelo

Para investigar a influência da taxa de aprendizado no desempenho final da rede, foi conduzido um experimento sistemático em que os pesos iniciais foram mantidos fixos em 0.5, e diferentes valores de taxa de aprendizado foram testados sob as mesmas condições de treinamento.

Toda a estrutura de execução foi organizada em diretório **Reports/test_2/.** Dentro dele, foram criadas subpastas com o padrão **lr_X_Y/** onde **X_Y** representa o valor específico da taxa de aprendizado, variando de 0.01 a 0.1 com incrementos de 0.01 e depois variando de 0.1 a 1.0 com incrementos de 0.1. Cada pasta corresponde a um experimento com uma configuração distinta de taxa de aprendizado aplicada ao modelo.

Em cada uma dessas pastas (**lr_X_Y/**), foram executadas 10 rodadas independentes de treinamento e teste, armazenadas como **run_0/** até **run_9/**. Cada uma dessas rodadas contém os seguintes arquivos **.json**:

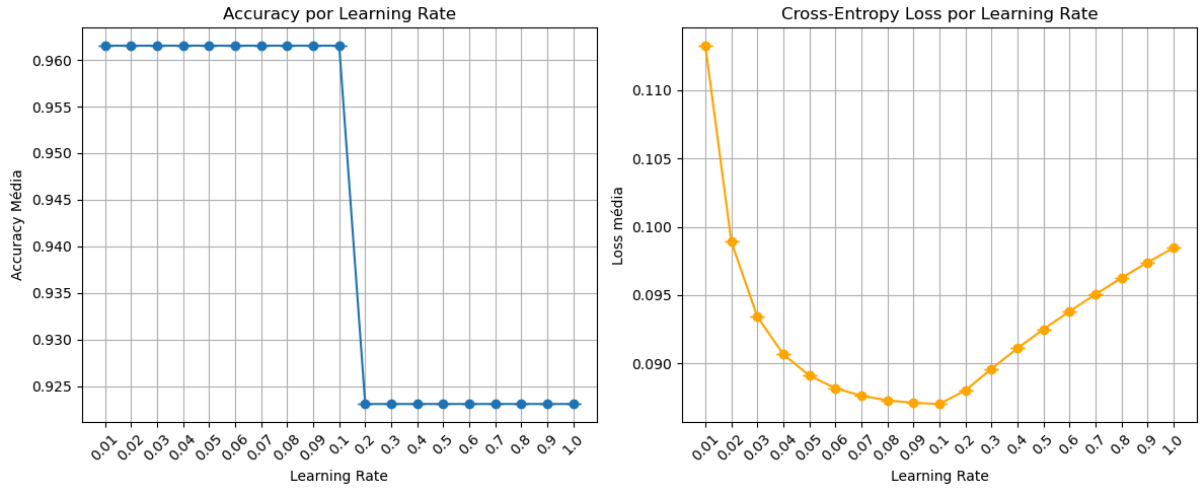
- Um arquivo contendo o modelo completo ao final de todas as épocas de treinamento.
- Um arquivo com o modelo da época considerado ideal com base no desempenho de validação.
- Um relatório com as métricas de desempenho sobre o conjunto de teste, contendo valores médios de perda (usando entropia cruzada e erro quadrático médio), acurácia e total de amostras válidas, além das predições realizadas - incluindo a classe real, a classe predita e as probabilidades atribuídas a cada classe.
- Um log das validações realizadas ao longo do treinamento, contendo, para cada época, os valores de perda por entropia cruzada, erro quadrático médio e perda total.

Os arquivos contendo os modelos (tanto o final quanto o da melhor época) registram a estrutura da rede, incluindo o número de perceptrons, número de entradas, presença de bias, e os pesos finais após o treinamento.

Essa organização metodológica permitiu a coleta de dados para posterior análise do impacto da taxa de aprendizado no processo de aprendizado e na capacidade de generalização das redes.

A figura abaixo ilustra a relação entre a taxa de aprendizado e duas métricas fundamentais: acurácia e perda por entropia cruzada (cross-entropy loss). É possível observar que a normalização dos dados de entrada no intervalo de -1 a 1 teve um papel significativo na estabilização do comportamento do modelo, especialmente sob diferentes taxas de aprendizado. O gráfico resultante apresenta uma distribuição mais regular e interpretável, evidenciando de forma clara como o ajuste da taxa de aprendizado impacta o desempenho de um modelo treinado com dados normalizados. Essa regularidade contribui

para uma análise mais objetiva do efeito isolado dessa hiperparâmetro na performance da rede.




A figura apresenta, à esquerda, a acurácia média e, à direita, a perda média por entropia cruzada do modelo normalizado em função da taxa de aprendizado, variando de 0.01 a 0.1 (passos de 0.01) e de 0.1 a 1.0 (passos de 0.1).

Podemos observar que, para learning rates entre 0.01 e 0.1, a acurácia se mantém altamente estável em torno de 96.1%, enquanto a perda decresce continuamente de cerca de 0.113 até um mínimo aproximado de 0.087. Esse comportamento indica que, dentro dessa faixa, as atualizações de peso são suficientemente pequenas para garantir convergência eficiente e baixo erro, resultando em ótimo equilíbrio entre rapidez de aprendizado e estabilidade.

A partir de lr = 0.2, nota-se um declínio abrupto de acurácia, que cai para aproximadamente 92.3%, permanecendo nesse patamar para taxas maiores, enquanto a perda começa a subir lentamente (atingindo cerca de 0.098 em lr = 1.0). Essa queda e posterior estabilidade em nűeis inferiores sugerem que o modelo passa a experimentar um leve overshooting ou uma atualização de pesos demasiadamente agressiva já a partir de 0.2, comprometendo a capacidade de ajuste fino dos parâmetros.

Em suma, os resultados reafirmam que existe uma janela ótima de learning rate (aproximadamente entre 0.04 e 0.1), na qual o modelo atinge seu pico de desempenho: perda mínima e acurácia máxima. Fora desse intervalo, especialmente acima de 0.2, há perda de performance – ainda que moderada – evidenciando a necessidade de selecionar criteriosamente a taxa de aprendizado para redes treinadas com dados normalizados.

# Exercício 3: Classificação de Dígitos com MNIST

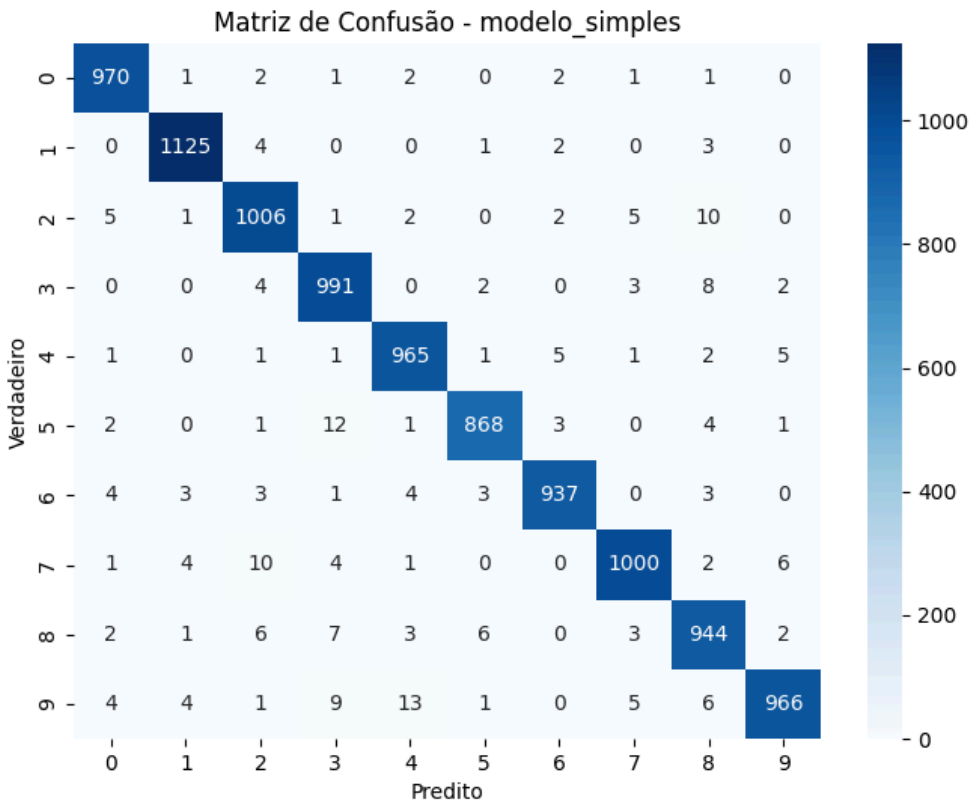
Neste experimento, utilizamos o conjunto de dados MNIST, composto por 70.000 imagens de dígitos manuscritos (60.000 para treinamento e 10.000 para teste), para avaliar distintas arquiteturas de redes neurais em tarefa de classificação multiclasses. O objetivo principal foi comparar o desempenho de uma rede perceptron simples (uma única camada oculta) com arquiteturas mais profundas, variando hiperparâmetros como número de neurônios, taxa de dropout e profundidade da rede.

Para garantir reprodutibilidade, todas as configurações (arquitetura, taxa de aprendizado, função de ativação, número de épocas, tamanho de batch e divisão entre treino e validação) foram documentadas e salvas em arquivos JSON na pasta ./Reports. As métricas monitoradas incluíram precisão (accuracy), perda (loss), matriz de confusão, bem como medidas de precisão (precision), revocação (recall) e F1-score por classe.

## Teste 1: Modelo Simples

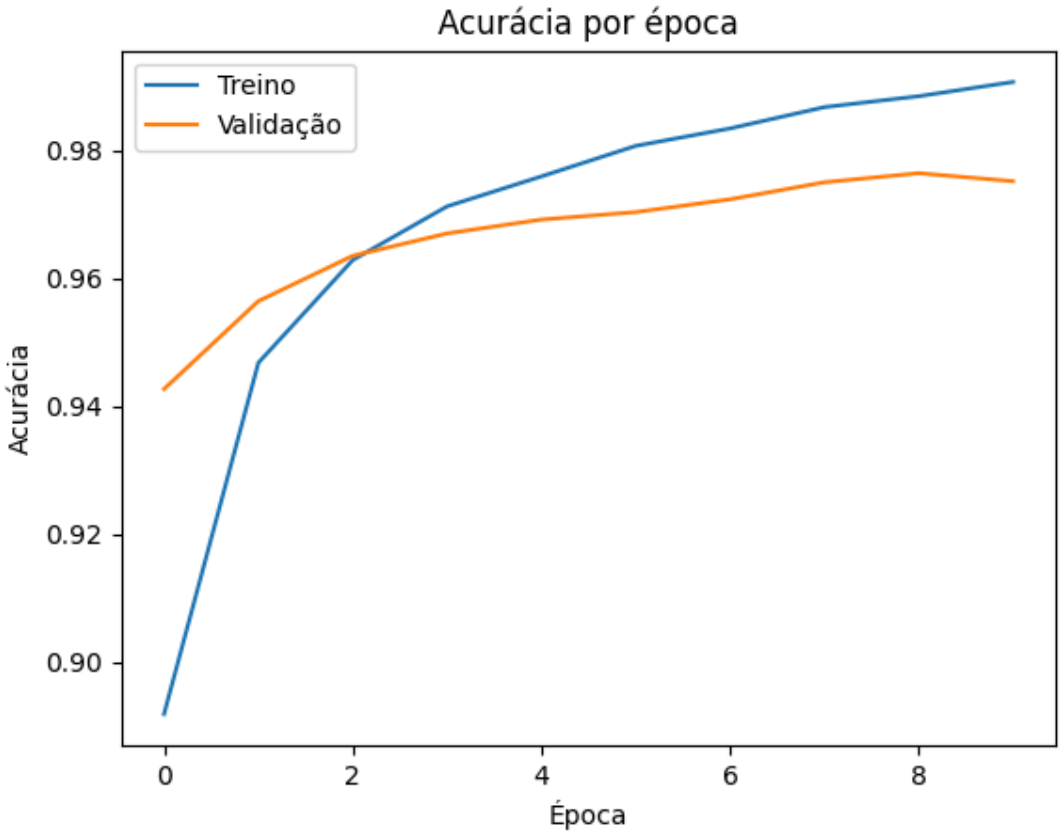
A seguir apresentamos as plotagens de métricas referentes ao modelo simples (uma camada oculta com 128 neurônios e ativação ReLU):

**Matriz de Confusão**



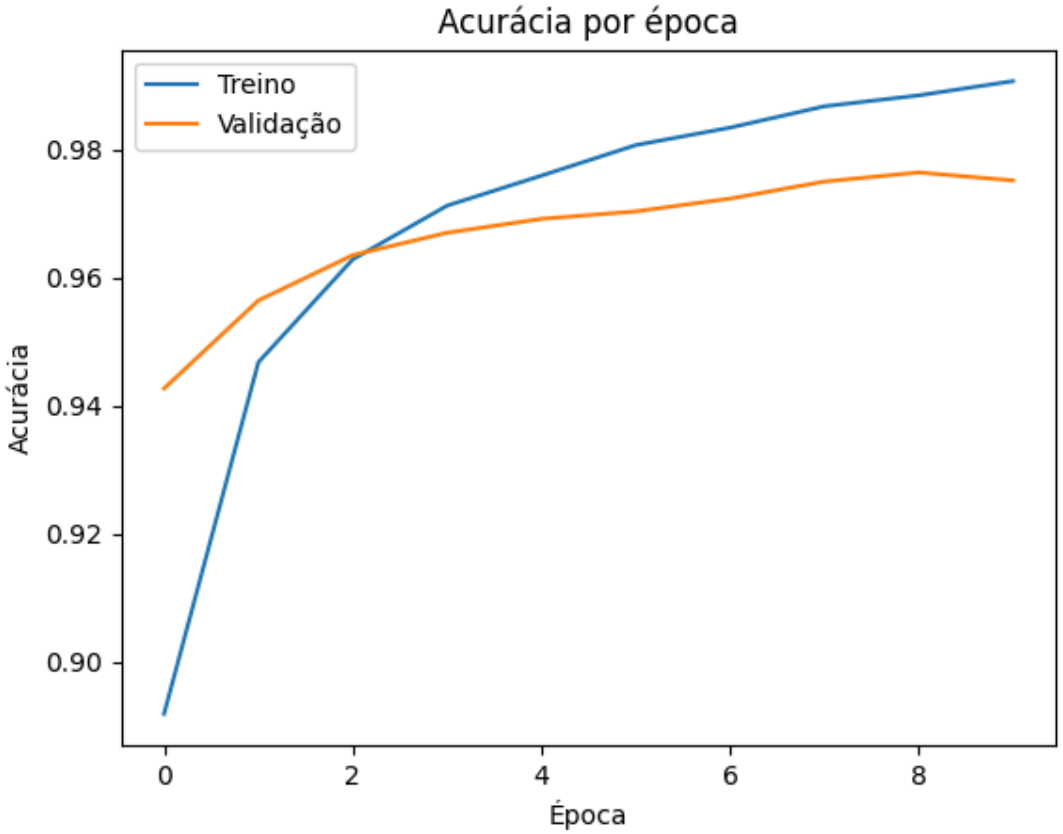
A matriz de confusão ilustra a distribuição de acertos e erros por classe. Observa-se que a maior parte dos dígitos é classificada corretamente, com valores na diagonal principal próximos a 970–1.125 amostras. As principais confusões ocorrem em pares de dígitos visualmente semelhantes (por exemplo, 5 vs. 3 e 4 vs. 9), mas em frequência limitada.

**Evolução da Acurácia por Época**




A curva de acurácia mostra rápido ganho de performance nas primeiras 3 épocas, atingindo aproximadamente 96% de acurácia de validação. Após a época 3, a melhora torna-se mais suave, convergindo para cerca de 97,5% em validação e 99% em treino, indicando leve overfitting ao final de 10 épocas.

**Evolução da Loss por Época**



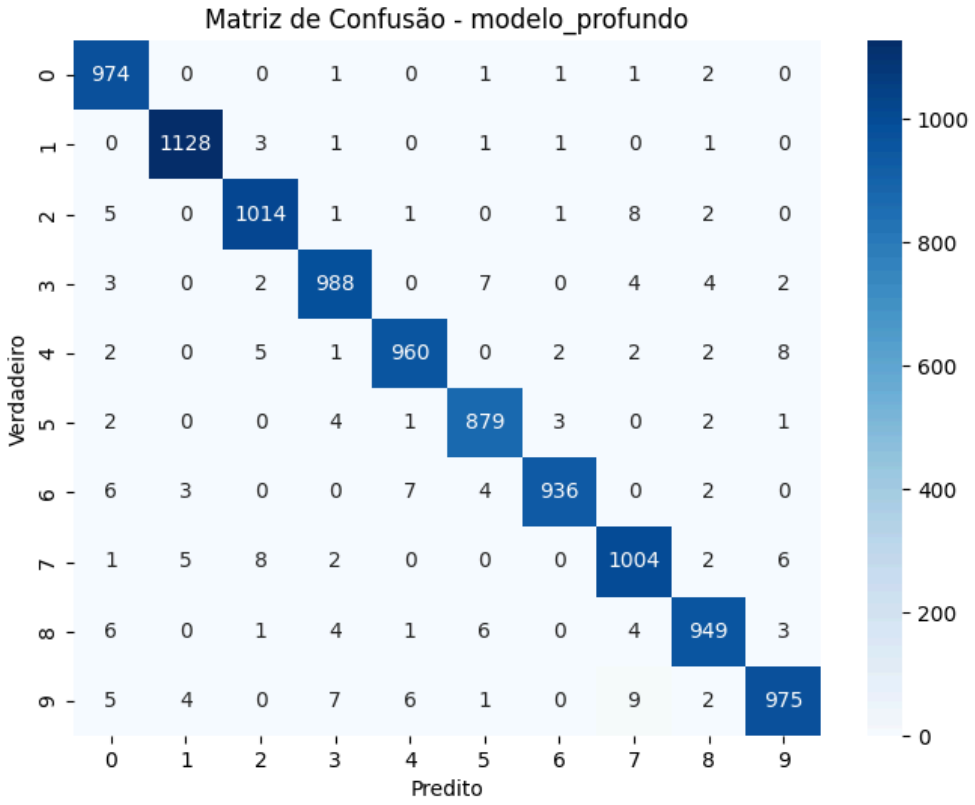

A perda (loss) diminui de forma acentuada nas primeiras 2 épocas e depois decresce de maneira gradual. A discrepância crescente entre as curvas de treino e validação reforça o leve overfitting observado.

Em resumo, o modelo simples alcança bom desempenho com mínima arquitetura, mas já apresenta indícios de overfitting que poderão ser atenuados ao comparar com a arquitetura profunda.

## Teste 2: Modelo Profundo

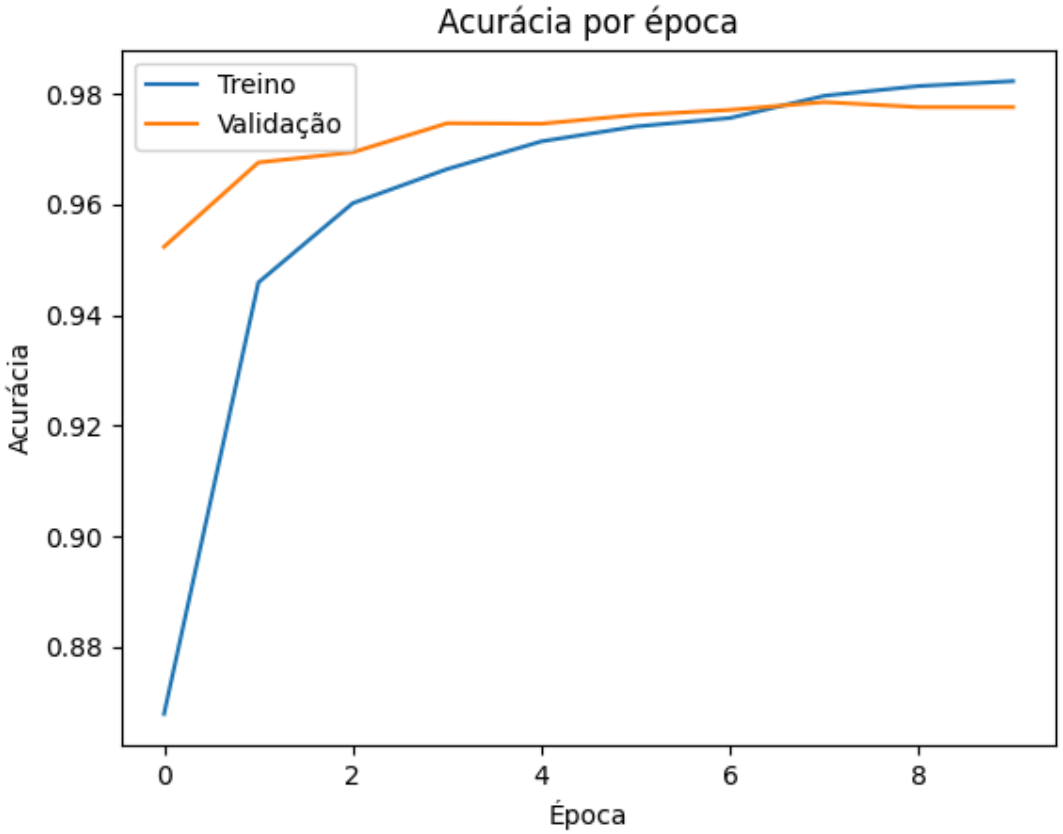
A arquitetura profunda testada consiste em duas camadas ocultas (256 e 128 neurônios, ReLU) com dropout de 30% entre elas. As métricas e plotagens a seguir destacam seu desempenho:

**Matriz de Confusão do Modelo Profundo**



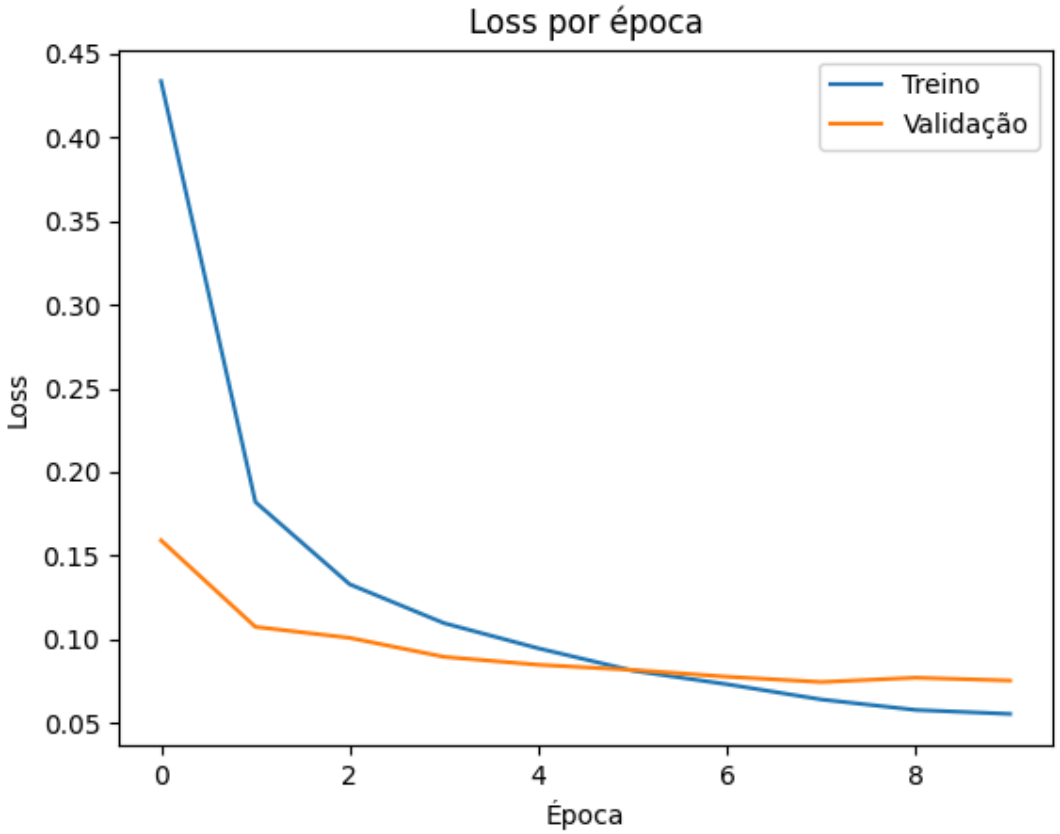

A diagonal principal mostra contagens ainda maiores de acertos (até 1.128 para a classe '1'), indicando leve ganho em precisão geral. Observa-se redução nas confusões mais frequentes do modelo simples, especialmente em pares como 4 vs. 9.

**Evolução da Acurácia por Época**




A curva de validação inicia mais alta (~95%) e atinge ~98% já na 4ª época, demonstrando maior capacidade de aprendizado, sem sinais prematuros de overfitting. Ao final de 10 épocas, o modelo profundo apresentou acurácia de validação próxima de 98% e acurácia de treino de 99,8%.

**Evolução da Loss por Época**




A perda de validação diminui rapidamente nas primeiras 2 épocas e então entra em um platô levemente acima de 0,07. A menor diferença entre curvas de treino e validação indica melhor generalização em comparação ao modelo simples.

Em suma, a arquitetura profunda trouxe ganhos de precisão e robustez, reduzindo o overfitting observado no modelo mais raso.