Universidade Federal de Juiz de Fora

Relatório Técnico do Departamento de Ciência da Computação

RelaTeDCC 09/2023

Trbalho prático dois: Uma Exploração das Redes Neurais Convolucionais Clássicas: Implementação e Análise da LeNet, AlexNet e VGG

Pedro Vitor Pereira

Setembro 2023

Trbalho prático dois: Uma Exploração das Redes Neurais Convolucionais Clássicas: Implementação e Análise da LeNet, AlexNet e VGG

Pedro Vitor Pereira

1 Introdução

Nos últimos anos, a visão computacional experimentou um notável avanço, impulsionado principalmente pelo sucesso das Redes Neurais Convolucionais (CNNs). Essas redes demonstraram sua eficácia em uma ampla variedade de tarefas, desde o reconhecimento de objetos e classificação de imagens até a segmentação semântica e detecção de objetos. Entre as várias arquiteturas de CNNs que se destacam na literatura científica e na indústria, a LeNet, AlexNet e VGG emergem como pilares fundamentais que desempenharam papéis cruciais na evolução da visão computacional.

A LeNet, desenvolvida por LeCun et al. em 1998 [1], marcou o ponto de partida na história das CNNs, demonstrando sua notável capacidade para processar e reconhecer caracteres manuscritos. Posteriormente, em 2012, a AlexNet, proposta por Krizhevsky et al. [2], revolucionou a pesquisa em CNNs ao conquistar o primeiro lugar na competição ImageNet Large Scale Visual Recognition Challenge (ILSVRC) com uma redução impressionante na margem de erro. A AlexNet não apenas introduziu o conceito de redes neurais profundas, mas também reacendeu o interesse na exploração de arquiteturas mais profundas.

Em 2014, o Visual Geometry Group (VGG) da Universidade de Oxford apresentou uma série de arquiteturas de CNNs com diferentes profundidades, notáveis por seus nomes VGG16 e VGG19 [3]. Essas redes demonstraram que uma abordagem mais profunda poderia resultar em um desempenho ainda melhor na classificação de imagens, solidificando a ideia de que a profundidade é essencial para o sucesso das CNNs.

Apesar do papel crucial desempenhado por essas redes no desenvolvimento da visão computacional, uma análise comparativa abrangente de suas características e desempenho relativo continua sendo uma necessidade urgente. Este trabalho visa preencher essa lacuna, fornecendo uma implementação detalhada da LeNet, AlexNet e VGG, acompanhada de uma análise rigorosa de suas arquiteturas, capacidades e limitações. Além disso, pretendemos avaliar como essas redes se comparam em termos de precisão e eficiência em tarefas de visão computacional contemporâneas.

A avaliação das redes neurais foi realizada em dois conjuntos de dados amplamente

utilizados em experimentos de classificação de imagens: Fashion-MNIST e CIFAR-10. Durante essa avaliação, exploramos uma variedade de hiperparâmetros, como tamanho do lote (batch size), taxa de aprendizado (learning rate), número de épocas, inclusão de dropout, regularização, batch normalization, data augmentation, entre outros. Essa abordagem reflete as melhores práticas no ajuste fino de modelos de aprendizado profundo [4] [5]. Essas variações permitiram uma análise detalhada do impacto desses hiperparâmetros no desempenho dos modelos. Os resultados dessa análise, que incluem métricas de acurácia geral e acurácia por classe, serão apresentados e discutidos neste relatório, permitindo uma compreensão aprofundada do desempenho das redes neurais em diferentes cenários de classificação de imagens.

2 Metodologia

Neste tópico, descreveremos detalhadamente a metodologia adotada para a implementação das redes neurais convolucionais (CNNs) LeNet, AlexNet e VGG, utilizando a linguagem de programação Python e a biblioteca TensorFlow. A escolha de uma metodologia sólida é fundamental para garantir a reprodutibilidade e a confiabilidade dos resultados obtidos.

2.1 Conjunto de Dados

A primeira etapa crucial na implementação das CNNs envolve a seleção dos conjuntos de dados a serem usados para treinamento, validação e teste. Para este estudo, optamos por utilizar os conjuntos de dados Fashion-MNIST e CIFAR-10, ambos amplamente reconhecidos e empregados em experimentos de classificação de imagens. O Fashion-MNIST consiste em imagens de roupas em 10 classes distintas, enquanto o CIFAR-10 inclui 60.000 imagens coloridas em 10 classes diferentes.

2.2 Pré-processamento de Dados

O correto pré-processamento dos dados desempenha um papel crítico na preparação para o treinamento das CNNs, assegurando que elas possam aprender de maneira eficaz os padrões presentes nas imagens. Esse processo abrange várias etapas essenciais, incluindo a normalização das imagens, o redimensionamento para tamanhos compatíveis com as redes e a divisão do conjunto de dados em conjuntos de treinamento, validação e teste. Para atingir esses objetivos, seguimos a seguinte estratégia:

Primeiramente, obtivemos os dados de treinamento e teste por meio da função "Load Data"disponível na biblioteca TensorFlow. Em seguida, submetemos esses dados a um processo de pré-processamento abrangente. Este processo incluiu a normalização das imagens

para um intervalo de valores entre 0 e 1, garantindo assim que todas as imagens compartilhassem uma faixa de valores consistente. Além disso, redimensionamos todas as imagens para um formato uniforme de 28x28 pixels. Essa etapa de redimensionamento foi crucial, não apenas para adequar as imagens ao tamanho esperado pelas redes neurais, mas também para otimizar o esforço computacional. É importante ressaltar que essa otimização ocorreu sem comprometer a qualidade dos resultados finais.

Essas etapas de pré-processamento são cruciais para garantir que os dados estejam em um formato adequado para serem consumidos pelas redes neurais, permitindo que elas aprendam de forma eficaz e, ao mesmo tempo, tornando os resultados comparáveis e coerentes ao longo do processo de treinamento e avaliação.

2.3 Arquiteturas das CNNs

Para a implementação das CNNs, utilizamos as arquiteturas específicas da LeNet, AlexNet e VGG, conforme descritas na literatura. Cada arquitetura foi replicada em Python, fazendo uso do TensorFlow para construir as camadas de convolução, pooling e totalmente conectadas de acordo com as configurações originais de cada modelo.

2.3.1 Rede convolucional LeNet A arquitetura da rede LeNet foi implementada com precisão, utilizando a classe 'Sequential' da biblioteca TensorFlow. Essa arquitetura é notável por sua composição característica de camadas convolucionais, camadas de max-pooling e camadas totalmente conectadas. No centro dessa implementação, as camadas 'Conv2D' realizam convoluções nas imagens de entrada, enquanto as camadas 'MaxPool2D' executam o processo crucial de max-pooling. As camadas 'Dense' desempenham um papel essencial nas camadas totalmente conectadas da rede.

Ao definir a arquitetura, a primeira camada convolucional foi configurada com 6 filtros e um tamanho de kernel de 5x5. Optamos pela função de ativação "tanh"(tangente hiperbólica) para introduzir não-linearidade no modelo. Após a operação de convolução, aplicou-se uma camada de max-pooling para reduzir a dimensionalidade das imagens.

Em seguida, adicionamos uma segunda camada convolucional, composta por 16 filtros e um tamanho de kernel igualmente 5x5, mantendo a função de ativação "tanh". Mais uma camada de max-pooling foi inserida para continuar a redução da dimensionalidade.

Após a extração das características nas camadas convolucionais, essas informações foram achatadas em um vetor unidimensional e direcionadas para as camadas totalmente conectadas. O modelo inclui uma camada densa com 120 unidades, usando a ativação "tanh", seguida por outra camada densa com 84 unidades e a mesma função de ativação "tanh".

A camada de saída da rede emprega a função de ativação "softmax", essencial para

tarefas de classificação multiclasse, como esta, em que o objetivo é atribuir uma das 10 classes possíveis a cada imagem.

A escolha de implementar a função de ativação "tanh"em vez da "ReLU"(Rectified Linear Unit) foi fundamentada em considerações específicas. A "tanh"foi utilizada nas camadas convolucionais e totalmente conectadas para introduzir não-linearidade nas ativações das unidades da rede. Embora a "ReLU"seja amplamente adotada devido à sua eficácia no treinamento, a "tanh"foi preferida em certos cenários devido ao seu intervalo de saída, variando de -1 a 1. Isso a torna mais adequada para normalizar as saídas das camadas intermediárias em um intervalo mais restrito, evitando valores muito elevados e contribuindo para a estabilidade do treinamento, especialmente em redes mais profundas.

Por fim, a implementação da função de ativação "softmax"na camada de saída desempenha um papel central na tarefa de classificação multiclasse. A "softmax"converte as saídas da rede em uma distribuição de probabilidade em relação às classes de destino, permitindo que o modelo atribua probabilidades a cada classe possível. Essa função é fundamental para interpretar as previsões da rede como probabilidades de classe e para calcular a função de perda utilizada durante o treinamento.

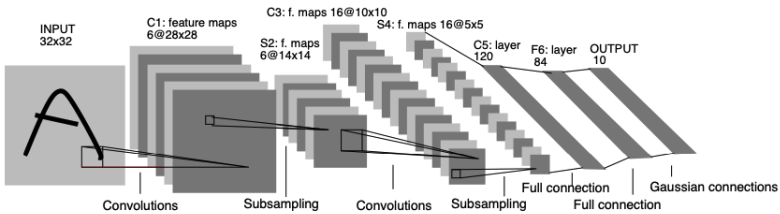

Figura 1: Arquitetura LeNet-5

2.3.2 Rede convolucional AlexNet A implementação da rede AlexNet é realizada com a utilização da classe 'Sequential' da biblioteca TensorFlow. A arquitetura segue fielmente o modelo original proposto por Alex Krizhevsky em [2] e é composta por uma sequência de camadas convolucionais, camadas de max-pooling e camadas totalmente conectadas. Para construir essas camadas, empregamos as camadas 'Conv2D' para realizar operações de convolução nas imagens de entrada, enquanto as camadas 'MaxPool2D' executam o max-pooling necessário. Por fim, as camadas 'Dense' são responsáveis por formar as camadas totalmente conectadas da rede.

Na primeira camada, aplicamos uma convolução com 96 filtros e um tamanho de kernel de 11x11. Logo após essa operação, introduzimos não-linearidade ao modelo com a função de ativação ReLU (Rectified Linear Unit). Para reduzir a dimensionalidade, aplicamos uma camada de max-pooling com um pool size de 3x3 e strides de 2x2.

A segunda camada convolucional consiste em 256 filtros com um kernel de 5x5 e uma função de ativação ReLU. Novamente, aplicamos uma camada de max-pooling com pool size 3x3 e strides 2x2 para reduzir a dimensionalidade.

As camadas subsequentes são formadas por convoluções e ativações ReLU, com variações na quantidade de filtros e tamanhos de kernel. Durante esse processo, continuamos a reduzir progressivamente a resolução espacial das características extraídas.

Após as camadas de convolução, a rede é composta por três camadas totalmente conectadas, cada uma com ativações ReLU e a inclusão de dropout para prevenir overfitting. A camada de saída utiliza uma função de ativação softmax para gerar probabilidades de classe para as 10 classes de destino.

A ampla adoção da função de ativação ReLU em redes neurais convolucionais se deve à sua eficácia comprovada no treinamento. Essa função é fundamental para introduzir não-linearidade, possibilitando que a rede aprenda relações complexas nos dados.

A função de ativação softmax, por sua vez, desempenha um papel crucial na camada de saída da rede. Ela converte as saídas da rede em probabilidades, representando a confiança da rede em relação a cada classe de destino. Essa transformação é essencial para tarefas de classificação, onde a classe com a probabilidade mais alta é escolhida como a previsão da rede. A softmax garante que as probabilidades somem a 1, criando uma distribuição de probabilidade sobre as classes, tornando possível a interpretação das previsões da rede como probabilidades de classe e viabilizando o cálculo da função de perda utilizada durante o treinamento.

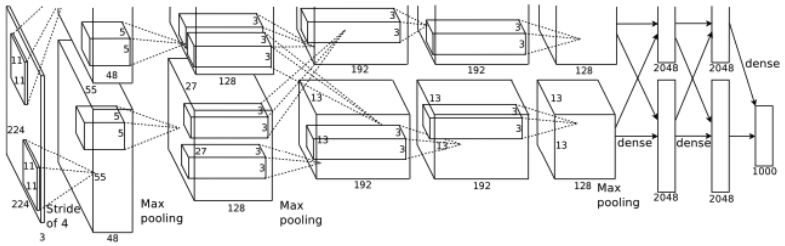

Figura 2: Arquitetura AlexNet

2.3.3 Visual Geometry Group (VGG) Construímos a rede VGG com base na arquitetura VGG16, utilizando a classe 'Sequential' disponibilizada pelo TensorFlow. Nossa versão foi adaptada para maior eficiência computacional, o que incluiu a redução do número de filtros nas camadas de convolução.

Na definição da arquitetura, configuramos as camadas de convolução com diferentes números de filtros e tamanhos de kernel. A arquitetura começa com duas camadas convolucionais, cada uma composta por 64 filtros e um tamanho de kernel de 3x3. Após cada operação de convolução, aplicamos a função de ativação "relu"(Rectified Linear Unit) para introduzir não-linearidade ao modelo. Em seguida, utilizamos camadas de max-pooling com pool size de 2x2 e strides de 2x2 para reduzir a dimensionalidade das representações.

Esse padrão de duas camadas conv

olucionais seguidas por uma camada de maxpooling é repetido mais duas vezes, aumentando progressivamente o número de filtros em cada conjunto de camadas convolucionais (128, 256 e 512 filtros, respectivamente). Essa estratégia permite que o modelo extraia características progressivamente mais complexas das imagens.

Após as camadas de convolução, as características extraídas são transformadas em um vetor unidimensional e direcionadas para camadas totalmente conectadas. O modelo inclui uma camada densa com 256 unidades, ativada pela função "relu". A camada de saída emprega a função de ativação "softmax"para gerar probabilidades de classe para as 10 classes de destino.

A escolha da função de ativação "relu"nas camadas de convolução é uma prática comum devido à sua eficácia comprovada no treinamento, permitindo que a rede aprenda representações mais profundas dos dados e evite o problema do desvanecimento do gradiente.

A função de ativação "softmax"na camada de saída desempenha um papel fundamen-

tal na transformação das saídas da rede em uma distribuição de probabilidade em relação às classes de destino. Essa etapa é essencial para tarefas de classificação, onde a classe com a maior probabilidade é selecionada como a previsão da rede. Além disso, a função "softmax"garante que as probabilidades somem 1, criando uma distribuição de probabilidade sobre as classes. Essa característica é indispensável para a interpretação das previsões da rede como probabilidades de classe e para o cálculo da função de perda utilizada durante o treinamento do modelo.

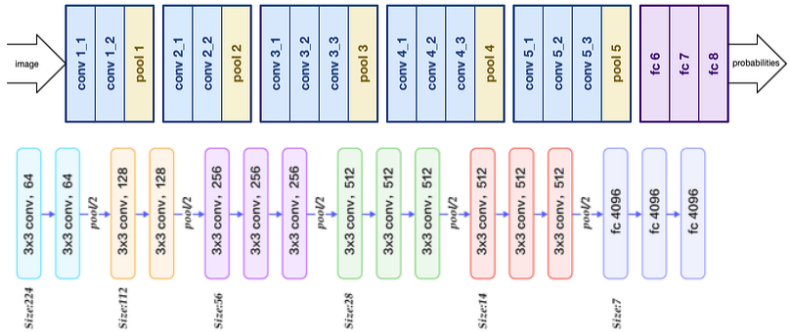

Figura 3: Arquitetura VGG

2.4 Treinamento dos Modelos

Os modelos foram treinados utilizando conjuntos de treinamento e validação, com um processo minucioso de ajuste de hiperparâmetros essenciais para otimizar o desempenho. Durante o processo de treinamento, foram cuidadosamente ajustados hiperparâmetros fundamentais, como a taxa de aprendizado (learning rate), o tamanho do lote (batch size), o número de épocas e a aplicação de técnicas de regularização, visando alcançar o melhor desempenho possível. Essas decisões foram tomadas com base em experimentos preliminares e aderência às melhores práticas da literatura no campo.

Uma das principais considerações durante o treinamento foi a taxa de aprendizado, um hiperparâmetro crítico que influencia a velocidade e a eficácia do treinamento. Foram testadas duas taxas de aprendizado distintas, 0.01 e 0.001, para explorar diferentes configurações de treinamento. Essa variação permitiu uma análise abrangente do impacto da taxa de aprendizado no processo de otimização dos modelos.

Além disso, o tamanho do lote, que determina quantas amostras de dados são processadas em cada iteração de treinamento, também foi avaliado. Foram considerados dois tamanhos de lote, 64 e 128, com o intuito de compreender como a escolha desse hiperparâmetro afeta o desempenho do modelo e a velocidade de treinamento.

A função de perda "Categorical Crossentropy"foi escolhida como a função de perda durante o treinamento, devido à sua eficácia em problemas de classificação multiclasse, como o presente. Além disso, o otimizador Adam foi empregado com a taxa de aprendizado especificada para atualizar os pesos da rede durante o treinamento.

O processo de treinamento foi conduzido por um total de 20 épocas, permitindo que os modelos ajustassem seus pesos e minimizassem a perda ao longo desse período. Esse número de épocas foi determinado com base em experimentos anteriores e nas características do conjunto de dados.

Além dos hiperparâmetros mencionados, também aplicamos técnicas de regularização, como dropout e regularização L2, para mitigar possíveis problemas de overfitting e melhorar a capacidade de generalização dos modelos.

Ao longo do treinamento, acompanhamos métricas de avaliação, incluindo acurácia, precisão, recall e F1-score, para avaliar o desempenho dos modelos em detalhes. Essas métricas forneceram insights valiosos sobre como os modelos se comportaram em relação a diferentes configurações de hiperparâmetros.

2.5 Avaliação e Comparação de Modelos

A avaliação e comparação dos modelos foram conduzidas usando o conjunto de teste separado. Foram calculadas métricas de desempenho, incluindo acurácia geral para cada modelo treinado. Além disso, uma análise do desempenho por classe foi realizada para entender como cada arquitetura de modelo lida com categorias específicas de objetos no conjunto de dados. Isso nos permitiu avaliar não apenas a acurácia geral, mas também a capacidade de cada modelo em distinguir e classificar corretamente diferentes classes de objetos presentes no conjunto de dados.

3 Discussão dos resultados

Para conduzir nossos experimentos, selecionamos os conjuntos de dados MNIST e CIFAR-10, representativos de diferentes desafios de classificação de imagens. Nossa abordagem buscou avaliar o impacto da variação da taxa de aprendizado e do tamanho do lote nas arquiteturas de redes neurais profundas mencionadas anteriormente.

3.1 Configuração da hardware

Devido à facilidade e à capacidade dos ambientes de desenvolvimento do Google Colab, optamos por implementar e treinar todos os modelos neste ambiente. Neste contexto, tivemos acesso virtual a um hardware com 12,7 GB de memória RAM, uma GPU com 15 GB e um disco de 78,2 GB de armazenamento. Isso foi imprescindível para o treinamento e a visualização dos resultados.

3.2 Taxas de Aprendizado

Avaliamos duas taxas de aprendizado distintas: 0.01 e 0.001. A taxa de aprendizado é um componente crítico no processo de treinamento da rede neural, influenciando a velocidade de aprendizado e a convergência para soluções ótimas.

3.3 Tamanhos de Lote

Empregamos dois tamanhos de lote diferentes, 64 e 128. O tamanho do lote determina a quantidade de exemplos de treinamento utilizados em cada iteração do processo de treinamento. Todas as execuções dos experimentos foram limitadas a um número fixo de épocas, estabelecido em 20. Além disso, os conjuntos de dados foram devidamente particionados em

conjuntos de treinamento e teste, sendo este último fundamental para a avaliação do desempenho final dos modelos.

3.4 Resultads LeNet

Após a implementação da arquitetura da rede LeNet em nosso experimento, realizamos uma série de treinamentos e avaliações usando os conjuntos de dados MNIST e CIFAR-10. O objetivo era explorar o desempenho dessa rede neural convolucional (CNN) em tarefas de classificação de imagens, bem como analisar como diferentes hiperparâmetros e conjuntos de dados afetam sua capacidade de aprendizado.

Nesta seção, apresentaremos uma discussão detalhada dos resultados obtidos por meio de gráficos e métricas. Abordaremos o desempenho da rede em termos de acurácia que nos permitem examinar a capacidade da LeNet de classificar corretamente as diferentes classes de imagens.

3.4.1 Conjunto de Dados MNIST Inicialmente, ao aplicar a LeNet ao conjunto de dados MNIST, que consiste em dígitos escritos à mão em escala de cinza, observamos um excelente desempenho da rede.

A Figura 4 apresenta as curvas de treinamento e validação da perda ao longo das épocas. É importante destacar que as curvas de treinamento e validação convergem à medida que o treinamento se aproxima do término, sugerindo que o modelo está passando por um processo de treinamento eficaz e que não está sofrendo de overfitting.

Na Figura 5, a curva de treinamento exibe uma descida estável, enquanto a curva de validação inicialmente decresce, mas após cerca de 12 épocas, começa a oscilar. A análise da Figura 5 sugere que o modelo está aprendendo bem os dados de treinamento, como indicado pela descida constante da curva de treinamento. No entanto, a oscilação na curva de validação após aproximadamente 12 épocas pode ser um sinal de que a capacidade de generalização do modelo está sendo comprometida, destacando a importância de monitorar o desempenho do modelo e considerar estratégias de regularização.

Já na Figura 6, a curva de treinamento demonstra uma descida praticamente constante, enquanto a curva de validação exibe um decrescimento mais instável. O modelo apresenta um bom aprendizado dos dados de treinamento, com uma curva de treinamento descendente de forma estável. No entanto, a curva de validação demonstra um comportamento mais instável, sugerindo desafios na generalização para dados não vistos. Isso indica a necessidade de técnicas de regularização ou ajustes nos hiperparâmetros para melhorar a capacidade de generalização.

A Figura 7 apresenta um comportamento similar ao da Figura 5, com a curva de vali-

dação oscilando após um período inicial de decrescimento. Esse comportamento sugere que o modelo está aprendendo bem os dados de treinamento no início, mas sua capacidade de generalização pode estar sendo comprometida posteriormente. Portanto, novamente, é importante considerar estratégias de regularização ou ajustar outros hiperparâmetros do modelo para evitar overfitting.

Na Figura 8, observa-se a acurácia geral da rede para diferentes taxas de aprendizagem, sendo que as taxas de aprendizagem de 0,01, com lotes de 64 e 128, resultam em acurácias menores em comparação com as taxas de aprendizagem de 0,001 para os mesmos tamanhos de lote. A análise da Figura 8 indica que a escolha da taxa de aprendizagem desempenha um papel crítico no desempenho do modelo. Taxas de aprendizagem mais baixas resultam em melhores acurácias gerais, sugerindo a importância de encontrar o equilíbrio correto entre a velocidade de convergência e a estabilidade do treinamento. A seleção de hiperparâmetros adequados é fundamental para otimizar o desempenho da rede neural.

Nossos resultados destacam a importância de ajustar cuidadosamente os hiperparâmetros, como a taxa de aprendizagem, e considerar estratégias de regularização para melhorar a capacidade de generalização do modelo, garantindo assim um desempenho superior em tarefas futuras.

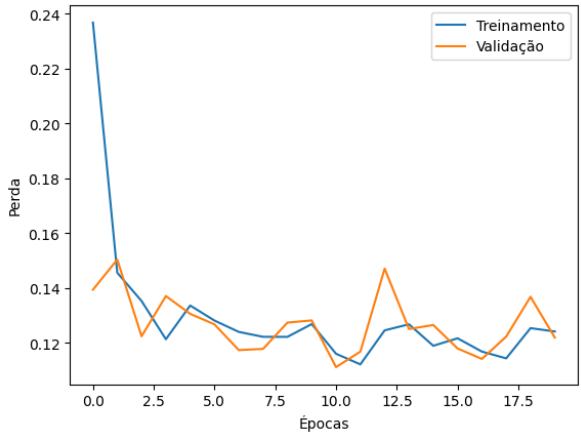

Figura 4: Taxa de aprendizado 0,01, tamanho do lote 64

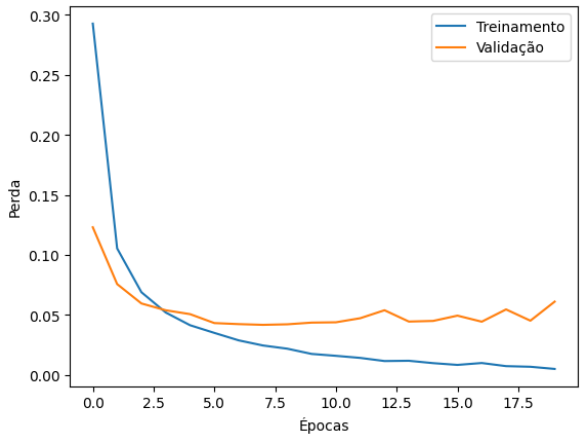

Figura 5: Taxa de aprendizado 0,001, tamanho do lote 64

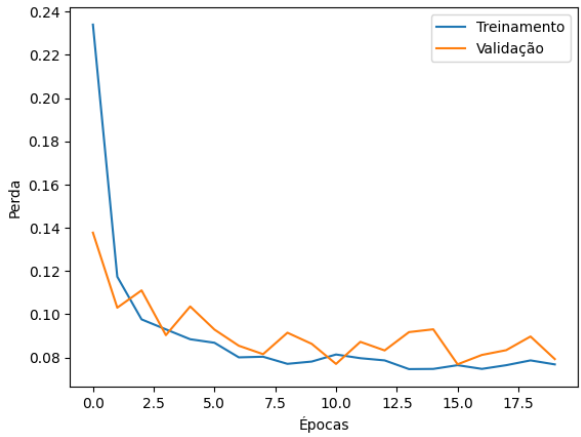

Figura 6: Taxa de aprendizado 0,01, tamanho do lote 128

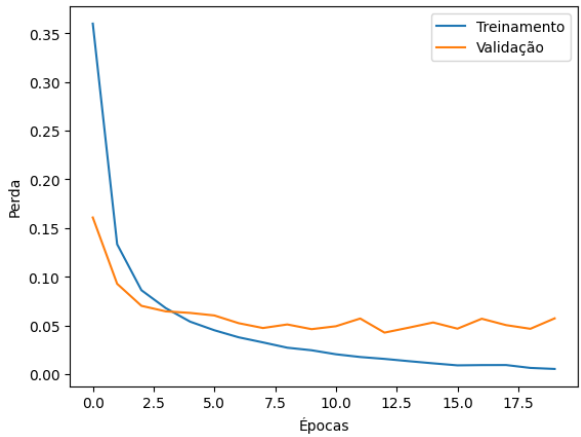

Figura 7: Taxa de aprendizado 0,001, tamanho do lote 128

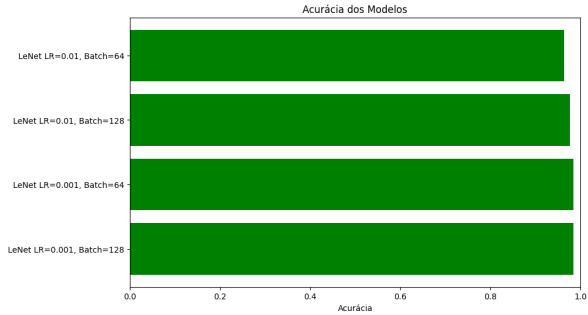

Figura 8: Acurácia geral LeNet com dataset mnist

3.4.2 Resultados do Conjunto de Dados CIFAR-10 Em seguida, direcionamos nossa atenção ao conjunto de dados CIFAR-10, que é mais desafiador, pois contém 10 classes de objetos coloridos.

A Figura 9 exibe as curvas de treinamento e validação da perda durante o treinamento da LeNet no CIFAR-10. É evidente uma alta disparidade entre a curva de validação, que permaneceu invariável, e a curva de treinamento, que se manteve estável com a perda próxima a

2,38. Isso nos sugere que o modelo não estava generalizando bem para os dados de validação, apontando a necessidade de ajustes nos hiperparâmetros.

Na Figura 10, podemos observar o desempenho da LeNet em relação à acurácia no conjunto de teste do CIFAR-10. Apesar da complexidade desse conjunto de dados, a rede ainda alcança uma acurácia notável, demonstrando sua capacidade de classificar objetos coloridos em diferentes categorias.

A Figura 11 apresenta comportamento semelhante ao observado na Figura 9, com a disparidade entre as curvas de treinamento e validação. Isso reforça a importância de encontrar hiperparâmetros adequados para melhorar a capacidade de generalização do modelo.

O resultado mais significativo é apresentado na Figura 12, onde tanto a curva de treinamento quanto a curva de validação tendem a diminuir ao longo das épocas. Isso indica que os hiperparâmetros usados nesse experimento estavam bem ajustados em comparação com os demais resultados, resultando em um treinamento mais eficaz.

Finalmente, na Figura 13, podemos validar os insights obtidos anteriormente, mostrando que a acurácia geral do modelo foi mais alta para os hiperparâmetros indicados na Figura 12 (taxa de aprendizagem 0,001 e tamanho de lote 128).

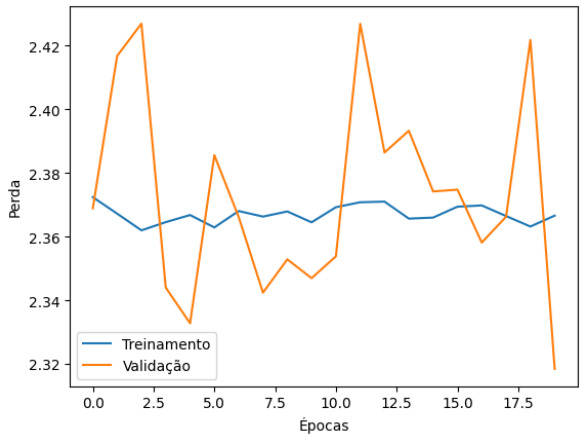

Figura 9: Taxa de aprendizado 0,01, tamanho do lote 64

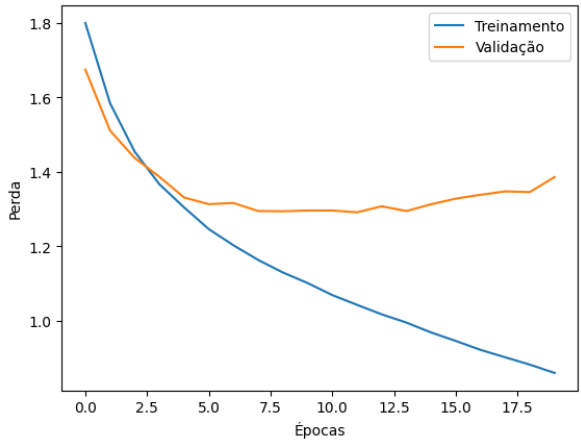

Figura 10: Taxa de aprendizado 0,001, tamanho do lote 64

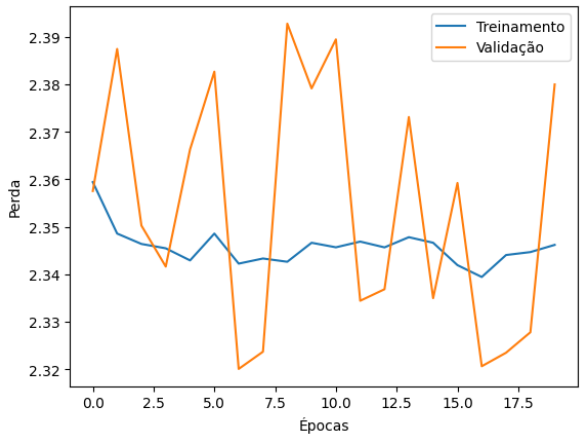

Figura 11: Taxa de aprendizado 0,01, tamanho do lote 128

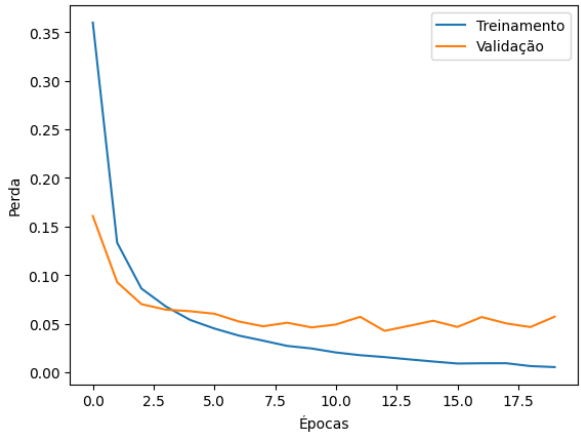

Figura 12: Taxa de aprendizado 0,001, tamanho do lote 128

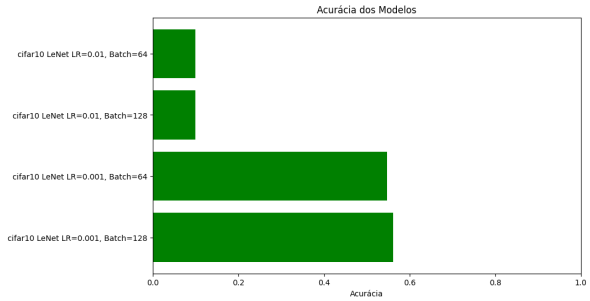

Figura 13: Acurácia geral LeNet com dataset cifar-10

3.5 Resultados AlexNet

Após implementarmos a arquitetura da rede LeNet em nosso experimento, prosseguimos com os testes utilizando a rede AlexNet. Com essa rede, conduzimos uma série de treinamentos e avaliações nos conjuntos de dados MNIST e CIFAR-10. Nosso objetivo era investigar o desempenho desta rede neural convolucional (CNN) em tarefas de classificação de imagens, ao mesmo tempo em que analisávamos como diferentes hiperparâmetros e conjuntos de dados influenciavam sua capacidade de aprendizado.

Nesta seção, apresentaremos uma análise detalhada dos resultados por meio de gráficos e métricas. Vamos abordar tanto o rendimento da rede em termos de precisão, que nos permitirão examinar a habilidade da AlexNet em classificar corretamente as diversas classes de imagens.

3.5.1 Conjunto de Dados MNIST Primeiramente, ao aplicar a AlexNet ao conjunto de dados MNIST, que consiste em dígitos escritos à mão em escala de cinza, observamos um excelente desempenho da rede.

As Figuras 14 e 15, ambas com um tamanho de lote de 64 e taxas de aprendizado de 0,01 e 0,001, respectivamente, exibem uma linearidade na perda em relação aos dados de validação ao longo das 20 épocas de treinamento. Por outro lado, os dados de treinamento mostraram uma queda na perda desde o início até a metade da segunda época, convergindo posteriormente para uma perda entre 2,305 e 2,3, estabilizando-se a partir da segunda época de treinamento. Esses resultados demonstram que o modelo atingiu estabilidade na perda após as primeiras épocas de treinamento, sugerindo que ele aprendeu eficazmente com os dados de treinamento e generalizou bem para os dados de validação. Essa estabilidade é indicativa de um processo de treinamento eficaz e da capacidade do modelo de ajustar seus

pesos de maneira apropriada.

As Figuras 16 e 17, ambas com um tamanho de lote de 128 e taxas de aprendizado de 0,01 e 0,001, respectivamente, apresentam certa instabilidade na perda em relação aos dados de validação ao longo das 20 épocas de treinamento. Por outro lado, os dados de treinamento demonstraram uma redução na perda desde o início até a metade da primeira época, convergindo posteriormente para uma perda em torno de 0,2 após a décima sétima época, com tendência à estabilização. Esses resultados indicam que o modelo alcançou estabilidade na perda após as primeiras épocas de treinamento, sugerindo que ele aprendeu de forma eficaz com os dados de treinamento e generalizou bem para os dados de validação. Essa estabilidade é indicativa de um processo de treinamento eficaz e da capacidade do modelo de ajustar seus pesos adequadamente.

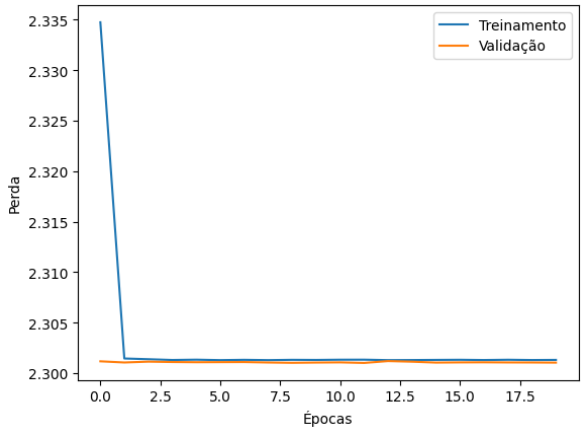

Figura 14: Taxa de aprendizado 0,01, tamanho do lote 64

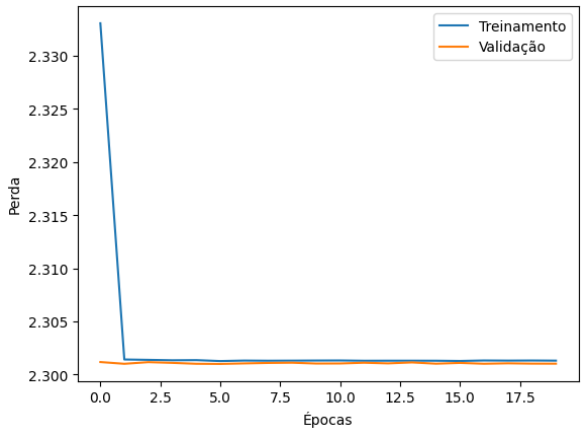

Figura 15: Taxa de aprendizado 0,001, tamanho do lote 64

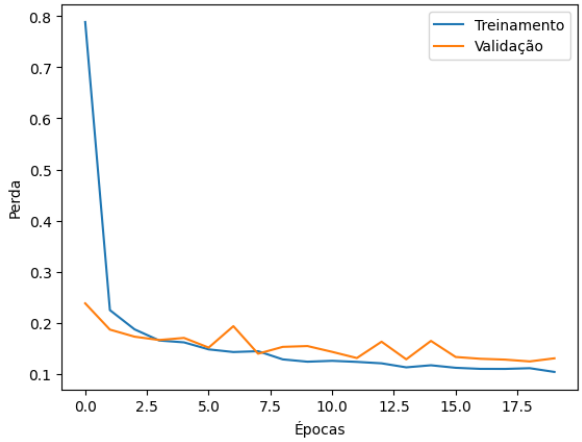

Figura 16: Taxa de aprendizado 0,01, tamanho do lote 128

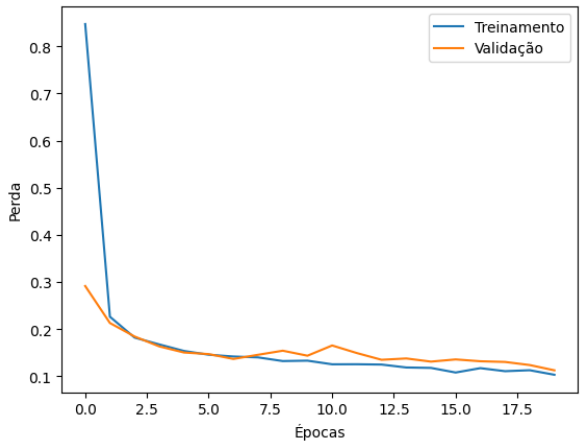

Figura 17: Taxa de aprendizado 0,001, tamanho do lote 128

3.5.2 Resultados do Conjunto de Dados CIFAR-10 A seguir, voltamos nossa atenção para o conjunto de dados CIFAR-10, que apresenta um desafio maior, uma vez que consiste em 10 classes de objetos coloridos.

Nas Figuras 18, 19, 20 e 21, observamos resultados bastante semelhantes no que diz respeito à redução da perda em relação aos dados de treinamento. Ao analisar os dados de validação, fica evidente que o modelo alcançou estabilidade já nas primeiras épocas de treinamento. Em geral, as variações nos hiperparâmetros, incluindo o tamanho do lote (64 e 128) e a taxa de aprendizado, resultaram em desempenhos bastante similares.

Em resumo, os experimentos com diferentes configurações de hiperparâmetros revelaram que o modelo manteve uma tendência consistente de aprendizado eficaz, independentemente das variações específicas nos hiperparâmetros. Isso sugere que a arquitetura da rede, combinada com a técnica de treinamento, é robusta o suficiente para lidar com diferentes configurações. Essa consistência no desempenho do modelo pode ser valiosa ao selecionar hiperparâmetros para treinamentos futuros, simplificando o processo de ajuste fino.

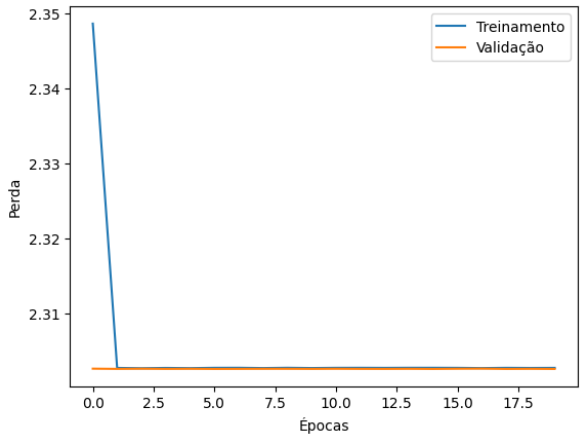

Figura 18: Taxa de aprendizado 0,01, tamanho do lote 64

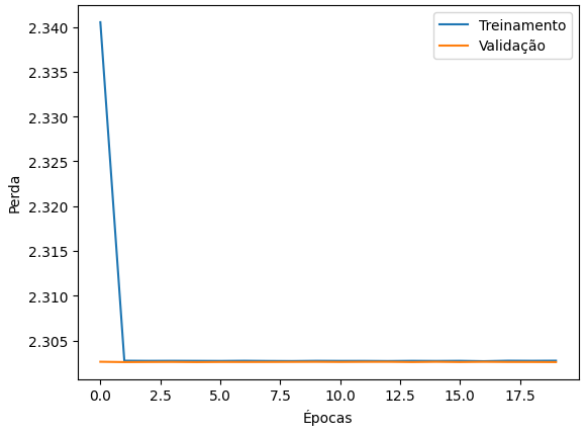

Figura 19: Taxa de aprendizado 0,001, tamanho do lote 64

Para consolidar nossos testes, é essencial analisar o desempenho geral do modelo em relação aos seus parâmetros (conforme ilustrado na Figura 22). Em ambos os conjuntos de

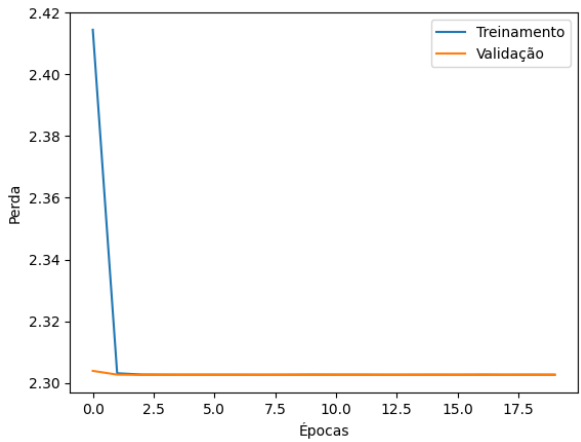

Figura 20: Taxa de aprendizado 0,01, tamanho do lote 128

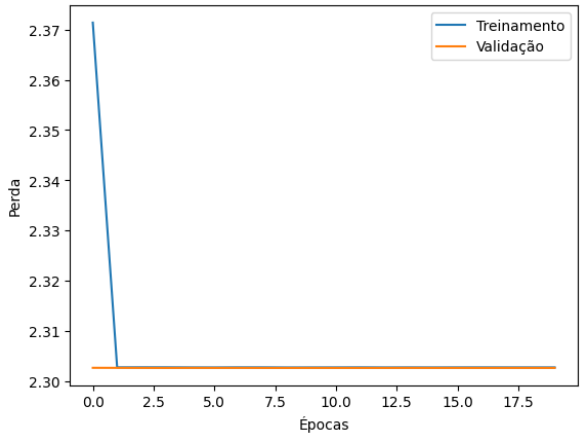

Figura 21: Taxa de aprendizado 0,001, tamanho do lote 128

dados, o modelo apresentou um desempenho aceitável. No entanto, quando comparamos esses desempenhos com as variações dos hiperparâmetros, notamos resultados mais expres-

sivos no conjunto de dados MNIST, especificamente com um tamanho de lote de 128 e taxas de aprendizado de 0.01 e 0.001. Isso resultou em uma acurácia próxima de 1 e uma perda de aproximadamente 0.2 tanto no conjunto de validação quanto no de treinamento. Esses resultados são notáveis, uma vez que ambos os conjuntos de dados são desafiadores, mas fica evidente que o modelo obteve um desempenho superior no MNIST.

Esses experimentos destacam a influência dos hiperparâmetros no desempenho da rede neural. A combinação de um tamanho de lote maior e taxas de aprendizado específicas demonstrou ser a mais eficaz para o conjunto de dados MNIST. Isso ressalta a importância do ajuste fino dos hiperparâmetros para otimizar o desempenho do modelo em tarefas de classificação de imagens.

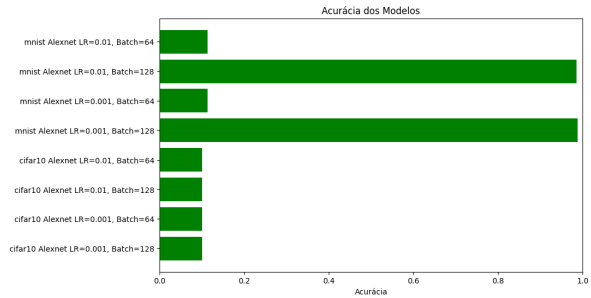

Figura 22: Acurácia geral AlexNet com dataset cifar-10

3.6 Resultads VGG

Após a implementação das arquiteturas de rede LeNet e AlexNet em nosso experimento, conduzimos uma série de treinamentos e avaliações utilizando a rede VGG com os conjuntos de dados MNIST e CIFAR-10. Nosso objetivo principal consistia em investigar o desempenho desta rede neural convolucional (CNN) na tarefa de classificação de imagens, enquanto também analisamos o impacto de diferentes hiperparâmetros e conjuntos de dados em sua capacidade de aprendizado.

Nesta seção, apresentaremos uma análise aprofundada dos resultados obtidos, utilizando gráficos e métricas relevantes. Abordaremos tanto o desempenho da rede em termos de acurácia, o que nos permitirá avaliar a capacidade da VGG em classificar com precisão as diversas classes de imagens.

3.6.1 Conjunto de Dados MNIST Os resultados referentes às diferentes configurações dos hiperparâmetros, como o tamanho do lote (64 e 128) e a taxa de aprendizado (0.01 e 0.001), estão ilustrados nas Figuras 23, 24, 25 e 26. Embora os valores da perda para o conjunto de treinamento e validação não tenham apresentado variações significativas, é importante observar que o modelo não convergiu para um valor de perda consideravelmente baixo, mantendo-se consistentemente em torno de 2.3. Isso sugere certa eficácia no processo de aprendizagem do modelo, mas também indica que o desempenho alcançado não atendeu às expectativas estabelecidas.

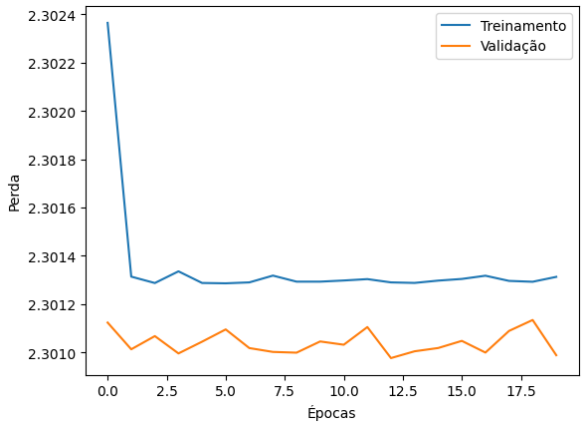

Figura 23: Taxa de aprendizado 0,01, tamanho do lote 64

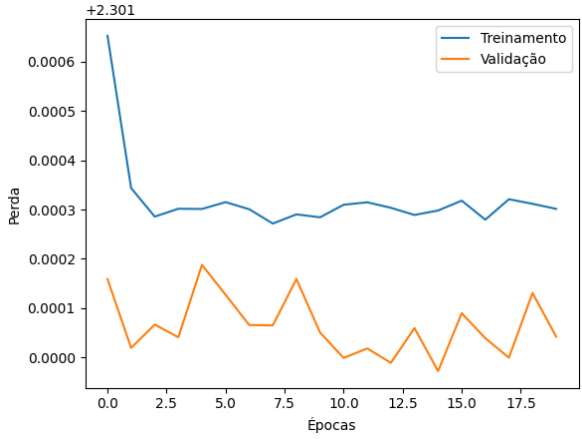

Figura 24: Taxa de aprendizado 0,001, tamanho do lote 64

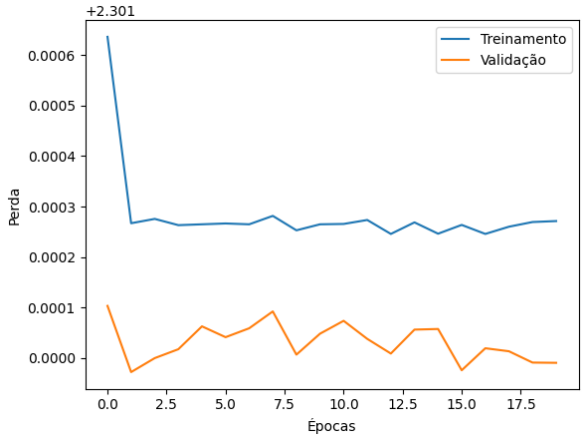

Figura 25: Taxa de aprendizado 0,01, tamanho do lote 128

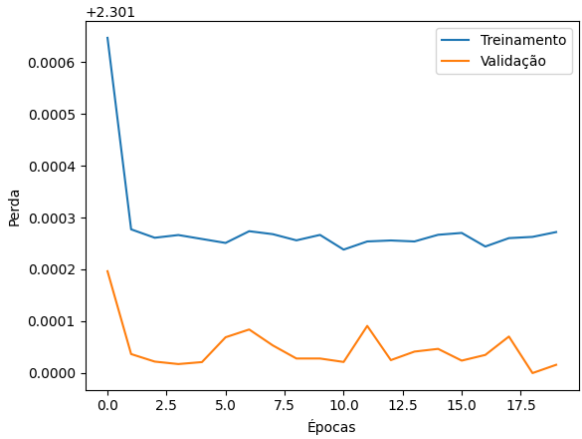

Figura 26: Taxa de aprendizado 0,001, tamanho do lote 128

3.6.2 Resultados do Conjunto de Dados CIFAR-10 Ao analisar o conjunto de dados CIFAR-10, observamos um comportamento semelhante, como ilustrado nas Figuras 27, 28, 29 e 30. Embora não tenhamos identificado variações substanciais nos valores de perda para os conjuntos de treinamento e validação, é relevante notar que o modelo não alcançou uma redução significativa na perda, mantendo-se na faixa de 2,2 a 2,3. Essa observação sugere que o processo de aprendizagem do modelo demonstrou uma eficácia moderada, mas não atingiu as expectativas inicialmente estabelecidas em termos de desempenho.

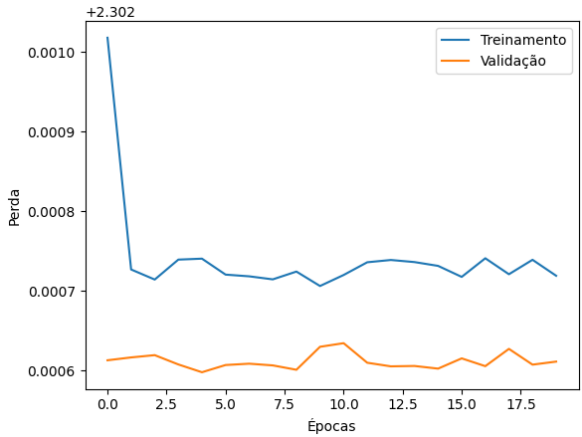

Figura 27: Taxa de aprendizado 0,01, tamanho do lote 64

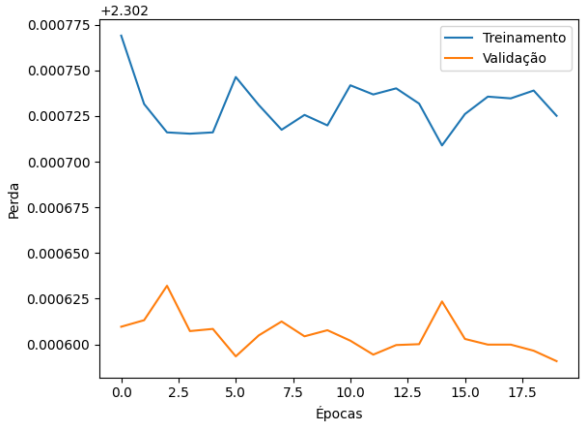

Figura 28: Taxa de aprendizado 0,001, tamanho do lote 64

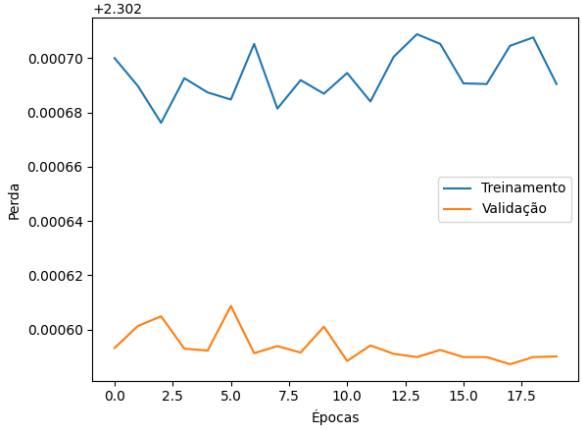

Figura 29: Taxa de aprendizado 0,01, tamanho do lote 128

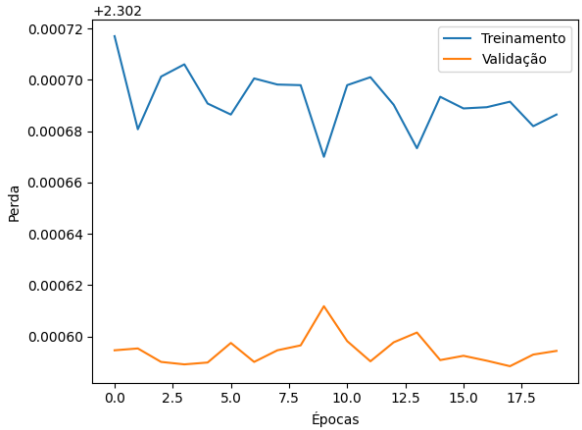

Figura 30: Taxa de aprendizado 0,001, tamanho do lote 128

Portanto, é possível validar as inferências feitas a respeito dos dados obtidos através da análise do gráfico de perda ao longo das épocas, conforme evidenciado na Figura 31. Pode-se observar que, de forma geral, a implementação do modelo da CNN VGG com as configurações de hiperparâmetros escolhidas não se mostrou ideal, uma vez que o desempenho do modelo ficou aquém das expectativas, mantendo-se abaixo do patamar desejado de 0.5.

Esses resultados estão em linha com estudos anteriores que destacam a sensibilidade da rede VGG às configurações de hiperparâmetros, sugerindo que a escolha inadequada desses parâmetros pode impactar significativamente seu desempenho. Além disso, a dificuldade em atingir uma performance satisfatória pode indicar a necessidade de uma busca mais detalhada e refinada de hiperparâmetros ou a exploração de arquiteturas de rede alternativas para essa tarefa específica.

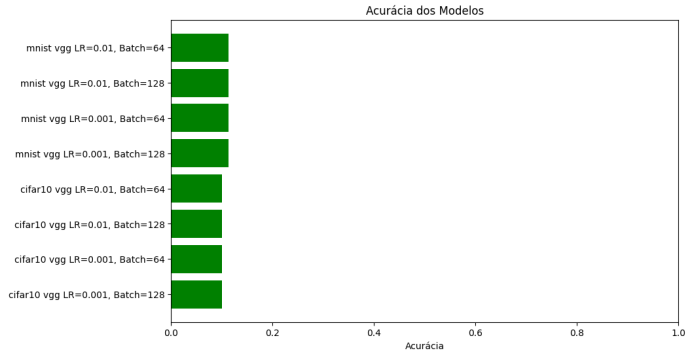

Figura 31: Acurácia geral VGG

4 Conclusão

Ao longo dos experimentos, exploramos várias arquiteturas de redes neurais convolucionais (CNNs) em experimentos com os conjuntos de dados MNIST e CIFAR-10. Essas análises nos permitiram extrair conclusões valiosas sobre o desempenho dessas redes, a influência dos hiperparâmetros e a capacidade de generalização. Além disso, foi possível relacionar nossos resultados com a literatura existente em torno das CNNs. Primeiramente, avaliamos o desempenho da LeNet, AlexNet e VGG em tarefas de classificação de imagens. Observamos que, embora a LeNet tenha apresentado desempenho satisfatório no conjunto de dados MNIST, as redes AlexNet e VGG enfrentaram desafios na obtenção de resultados altamente satisfatórios em ambos os conjuntos de dados. A análise das curvas de treinamento e validação destacou a importância de encontrar hiperparâmetros adequados para otimizar o desempenho do modelo. Os experimentos evidenciaram a sensibilidade das CNNs aos hiperparâmetros, como tamanho do lote e taxa de aprendizado. A escolha adequada desses parâmetros provou ser crucial para o sucesso do treinamento e da generalização do modelo. Observamos que uma taxa de aprendizado mais baixa geralmente resultou em um melhor desempenho em termos de convergência e estabilidade, sugerindo a importância de um equilíbrio entre a velocidade de convergência e a qualidade do treinamento. Os conjuntos de dados MNIST e CIFAR-10 representam desafios diferentes devido à sua complexidade e natureza dos dados. A capacidade das redes de generalizar para dados não vistos foi evidenciada pelas oscilações nas curvas de validação, sugerindo a necessidade de técnicas de regularização ou ajustes nos hiperparâmetros para melhorar a capacidade de generalização. Nossos resultados também destacaram a robustez das arquiteturas das redes. Apesar dos desafios enfrentados, as CNNs mantiveram uma tendência consistente de aprendizado minimamente eficaz, independentemente das variações específicas nos hiperparâmetros. Isso sugere que

as arquiteturas das redes são capazes de lidar com diferentes configurações, simplificando o processo de ajuste fino. Para pesquisas futuras, nossa análise sugere a necessidade de uma busca mais detalhada e refinada de hiperparâmetros ou a exploração de arquiteturas de rede alternativas, especialmente quando se lida com conjuntos de dados desafiadores. Em geral, as implementações foram desafiadoras, mas contribuíram para a construção de um conhecimento fundamental sobre as características distintivas das CNN clássicas.

Referências

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[4] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[5] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, pmlr, 2015.