Relatório LAMIA 20
Prática: Visão Computacional (III)

Kaique Medeiros Lima

1 Introdução

O card 20, *Prática: Visão Computacional (III)* se dá ao uso de PyTorch, que funciona por meio do uso de tensores, os quais foram usados para a prática de visão computacional. A prática envolveu a criação de uma rede neural convolucional (CNN) para classificação de imagens, utilizando o dataset CIFAR-10. A prática também envolveu o uso de técnicas como *data augmentation* e otimização de hiperparâmetros para melhorar o desempenho do modelo.

2 Descrição da atividade

2.1 O que é Deep Learning

Deep learning, ou aprendizagem profunda, é uma técnica de inteligência artificial (IA) que permite que computadores aprendam a partir de dados não estruturados e não rotulados.

O deep learning é um subconjunto do machine learning, e a principal diferença entre os dois é o tipo de dados processados e os métodos de aprendizagem.

O deep learning funciona através de redes neurais artificiais, que são compostas por várias camadas de nós interconectados. Cada camada baseia-se na anterior para otimizar e refinar a categorização ou previsão.

O deep learning pode ser usado para realizar tarefas como:

- Reconhecimento de voz
- Identificação de imagens
- Realização de previsões
- Detecção de fraudes
- Carros autônomos
- Tradução automática

O deep learning é usado em vários setores e está integrado ao nosso dia a dia, por exemplo, em assistentes digitais, controles remotos de TV por voz e sistemas de segurança.

2.2 O que é PyTorch

O PyTorch é uma biblioteca de aprendizado de máquina de código aberto para Python, projetada para ser fácil de usar e permitir implementações rápidas. Ele oferece computação dinâmica, o que facilita o desenvolvimento de modelos complexos e a experimentação com novas arquiteturas. O PyTorch é amplamente utilizado para construir e treinar modelos de aprendizado de máquina de forma eficiente.

2.3 O que é Tensor

A estrutura de dados fundamental usada para criar redes neurais é o tensor, que pode ser criado com a função `torch.tensor()`. Os tensores são usados para representar as previsões de saída do modelo. Eles são os blocos de construção do machine learning, permitindo representar dados e cálculos de forma eficiente em redes neurais.

2.4 Vantagens de Usar um Tensor

Os tensores são otimizados para cálculos de deep learning, oferecendo alta performance em GPUs para acelerar o processamento. Além disso, são utilizados diretamente no PyTorch, facilitando a computação devido à API intuitiva, que permite manipulação eficiente de dados em paralelo.

2.5 O que é uma Rede Neural

Uma rede neural é um tipo de módulo de machine learning inspirado no funcionamento do cérebro humano, projetado para resolver problemas complexos, como reconhecimento de voz em assistentes virtuais e tradução automática de fala em texto.

2.5.1 Como Funciona

As redes neurais consistem em camadas de neurônios interconectados que realizam cálculos matemáticos. Os resultados dessas camadas são passados para os próximos neurônios, que produzem a saída com base nos pesos e bias, ajustados durante o treino para minimizar a loss. Assim, a rede neural aprende a generalizar dados e a fazer previsões com base em padrões.

2.6 Treinando uma Rede Neural

O primeiro passo para treinar uma rede neural é definir o objetivo, como classificação de imagens, previsão de resultados ou geração de novos dados. Em seguida, é necessário preparar o conjunto de dados, que deve ser vasto e representativo. Também é importante pré-processar os dados, normalizando e escalando conforme necessário. Após isso, é preciso desenvolver as camadas corretas para garantir que o treinamento atinja o resultado esperado.

2.7 Arquitetura de uma Rede Neural

A arquitetura de uma rede neural refere-se ao layout das camadas e neurônios dentro da rede. Um exemplo comum é a CNN (Convolutional Neural Network), que é composta

por camadas de convolução, camadas de pooling e uma camada totalmente conectada. Essa arquitetura é amplamente utilizada em visão computacional.

2.8 Função de Ativação e Função de Perda

2.8.1 Função de Ativação

Responsável por introduzir não-linearidade na rede, permitindo que ela aprenda padrões complexos. Ela é aplicada na entrada de um neurônio para determinar sua saída.

2.8.2 Função de Perda

A função de perda mede o desempenho da rede neural, comparando a saída prevista com o valor correto. Quanto menor o valor da perda, melhor a performance da rede.

2.9 Otimizadores

Os otimizadores são algoritmos que ajustam os pesos e bias das redes neurais durante o treinamento. Eles minimizam a função de perda ao calcular os gradientes dos parâmetros da rede e atualizá-los de forma a melhorar o desempenho.

2.10 Dataset e DataLoader

2.10.1 Dataset

É a coleção de dados usada para treinar a rede neural. Pode incluir imagens, texto ou valores numéricos.

2.10.2 DataLoader

Utilitário do PyTorch que facilita o carregamento do dataset, dividindo-o em batches menores para otimizar o treinamento.

2.11 Classificação de Imagens

Imagens são fontes ricas de informações, e redes neurais podem ser usadas para extrair features e padrões delas. Na classificação de imagens, a rede neural pega a imagem como entrada e produz uma distribuição de probabilidade sobre diversas classes como saída, a mais provável será a escolhida. O tipo de rede neural mais usado são as CNNs, por causa das camadas de convolução que aplicam filtros para extrair as features da imagem, produzindo um *feature map*.

As camadas de pooling encurtam os mapas, pegando os valores médios, reduzindo a dimensão enquanto preservando informações.

2.12 Hiperparâmetros

São os parâmetros setados antes do treinamento, como a taxa de aprendizado, número de camadas e o número de neurônios em cada camada. Não podem ser aprendidos durante o treinamento.

A taxa de aprendizado é muito importante, pois determina o quão rápido os pesos do modelo serão atualizados para as batches de dados. Se a taxa for muito baixa, o modelo irá demorar muito, se for muito alta, pode levar a má performance.

A quantidade de neurônios também é importante, pois caso seja uma contagem baixa, o modelo talvez não consiga aprender tudo o possível dos dados, mas se for muito alta, o modelo pode produzir overfitting.

2.13 Convolutional Neural Networks (CNNs)

As redes neurais convolucionais aprendem a reconhecer padrões em imagens por meio de verificação de pequenas partes da imagem de cada vez, sendo mais eficientes e corretas do que as redes neurais normais. Identificam os padrões da imagem pelas bordas e texturas aplicando filtros/kernels na imagem, criando mapas de características, como padrões específicos na imagem. Empilhando várias camadas convolucionais numa rede neural, o modelo pode aprender características mais complexas.

2.14 Data Augmentation

É o processo de gerar novo dado de treinamento transformando os dados já existentes em outros, aumentando a eficiência de aprendizado do modelo, como flipando ou espelhando uma mesma imagem para aumentar os dados de treinamento.

2.15 Auto Encoders

Tipo de rede neural que consiste em duas partes, o encoder e o decoder. O encoder reduz a dimensão da imagem de entrada, chamado de latent space. Já o decoder usa o latent space e o reconstrói para o normal. Autoencoders são usados para compressão, remoção de ruído e geração de dados. Capturam as features importantes descartando o ruído e informações desnecessárias. São usados para representar dados complexos em dimensões menores.

2.16 Variational Autoencoders

São um tipo de autoencoders que incorporam a interferência Bayesiana no processo de aprendizado. O encoder gera um vetor de média e outro de variância, os quais são usados para espaços latentes amostrais. Seu treinamento envolve minimizar o erro de reconstrução, mas maximizando o limite inferior da logaritimização dos dados, o qual é baseado na divergência Kullback-Leibler.

3 Conclusão

O deep learning tem transformado áreas como visão computacional e reconhecimento de voz, permitindo que computadores aprendam de maneira eficiente com grandes volumes de dados. Ferramentas como o PyTorch tornam o desenvolvimento de modelos mais acessível, enquanto técnicas como data augmentation e o uso de tensores otimizam o treinamento.