Card18_Prática_Visão_Computacional

Davi Bezerra Barros

Fundamentos de Deep Learning

Deep learning: É um subcampo de aprendizado de máquina que se inspira na forma em que o cérebro humano é estruturado para realizar suas operações. Redes neurais de deep learning utilizam muitas camadas de neurônios conectados para aprender padrões em grandes conjuntos de dados.

PyTorch: Uma bilioteca de python desenvolvida pela Meta AI, criada especificamente para trabalhar com deep learning. PyTorch utiliza *tensores* como sua estrutura de dados fundamental, com suporte a GPUs para aceleração de treinamento.

Tensor: Estrutura de dados multidimensional utilizada em várias áreas do conhecimento, o que a torna ideal para trabalhar com deep learning, que lida com dados multidimensionais complexos.

Operações com tensores

Funções do PyTorch para realizar operações com tensores:

- **torch.matmul(input, other):** Realiza a multiplicação de matrizes (ou tensores com dimensões compatíveis).
- **torch.sum(input):** Calcula a soma de todos os elementos de um tensor.
- **torch.reshape(input, shape):** Altera a forma de um tensor sem mudar o número de elementos.
- **torch.squeeze(input):** Remove dimensões de tamanho 1 de um tensor.
- **torch.unsqueeze(input, dim):** Insere uma dimensão de tamanho 1 em um tensor na posição especificada.
- **None**: é usado como um marcador para inferir a dimensão em operações de reshape.
- **torch.cat()**: Concatena tensores ao longo de uma dimensão específica.
- **permute()**: Permuta as dimensões de um tensor.

Em visão computacional, o formato dos tensores é geralmente:

- **Tensor 1D:** Número total de elementos
- **Tensor 2D:** Linhas x colunas
- **Tensor 3D:** N° de canais x linhas x colunas
- **Tensor 4D: Batch size x n° de canais x linhas x colunas

Vantagens de usar tensores:
Tensores permitem a representação de dados multidimensionais como imagens, textos e dados numéricos. Essa representação permite operações como o cálculo de gradientes e backpropagation de forma mais eficiente, além de permitir a paralelização desses cálculos para serem computados em GPUs, o que acelera significativamente o processo de aprendizado de uma rede neural.

Construindo uma rede neural

Fluxo de treinamento de uma rede neural
O processo de criação e treinamento de uma rede neural geralmente segue as seguintes etapas:

1. Definir o objetivo: Classificar imagens, predizer um valor numérico ou gerar novos dados.
2. Montar o dataset: Dados que serão utilizados para treinar a rede neural a reconhecer os padrões necessários para a tarefa. Devem ser vastos o suficiente para abranger todas as possibilidades com que o modelo terá de lidar.
3. Pré processamento: Operações feitas nos dados para os padronizar e torná-los utilizáveis pelo modelo, e divisão dos dados de treinamento e validação.
4. Desenvolver a arquitetura da rede neural: Selecionar as camadas e funções de ativação apropriados para mapear oas dados. Pode ser necessário experimentar com arquiteturas diferentes.
5. Treinamento da rede neural: Alimentar o modelo com os dados de treinamento, visando minimizar as funções de perda. Algoritmos de otimização são utilizados para ajustar os pesos da rede.
6. Avaliação da performance: Testar o modelo com dados desconhecidos para testar suas habilidades de generalização. Métricas como *Accuracy*, *Precision,* e *recall* são comummente utilizados para esta avaliação.
7. Ajustes: Se o modelo não performar bem, é necessário avaliar e reajustar parâmetros da rede e refazer o treinamento.

Arquitetura da rede neural
A arquitetura de uma rede neural se refere à organização dos nós, unidades individuais que recebem os dados de entrada, os processam e passam adiante para outros nós, até a última camada da rede neural. São organizados em camadas onde cada uma processa os dados de uma forma específica, e sua organização varia de acordo com o objetivo da rede.

Exemplos de arquiteturas:

- **Rede Neural Convolucional:** Blocos de camadas convolucionais intercaladas com camadas de pooling, seguidos de camadas densamente conectadas.
- **Rede Neural recorrente:** Usada para processar dados sequenciais, como texto ou fala. Contém um loop que permite que a informação seja passada adiante na sequência.

Classificação de imagens:
Em uma rede neural a imagem é tomada como input, e o modelo produz uma distribuição de probabilidade de uma série de classes como saída. A classe mais provável é então escohida como rótulo para a imagem.

Hiperparâmetros: Parâmetros de uma rede neural que são definidos antes do treinamento, como taxa de aprendizado, número de camadas e número de neurônios em cada camada. Caso o treinamento não apresente resultados satisfatórios, os hiperparâmetros podem ser ajustados para alcançar os resultados desejados após um novo treinamento.

Normalização de dados
Normalizar imagens significa ajustar o valor de seus pixels para um intervalo padrão, geralmente entr 0 e 1 ou -1 e 1 . Este processo ajuda a remover inconsistências e viéses do modelo, melhorar a convergência ao estabilizar o processo de otimizaçao

CNNs

Ao processar uma imagem, as redes neurais totalmente conectadas receberiam todos os pixels da imagem em sua camada inicial como valores de entrada, tornando o treinamento extremamente custoso computacionalmente devido ao grande número de parâmetros associados. Já as redes

neurais convolucionais(CNNs) utilizam camadas que aprendem a reconhecer padrões em pequenas partes da imagem, gerando os "mapas de atributos" que são processados pelas redes totalmente conectadas, tornando o processo mais rápido e preciso.

Data Augmentation

É o processo de geração de novos dados a partir das imagens já existentes no dataset, aumentando a diversidade das imagens e melhorando a preicisão do modelo. Isso é feito aplicando rotações, espelhamentos, cortes e esticando as imagens do dataset.

Auto Encoders Networks

Geralmente utilizados para compressão de imagens, remoção de ruídos e geraçao de dados, os autoencoders são compostos por duas partes; o Encoder, que recebe uma imagem como input e a mapeia para um espaço dimensional inferior chamado de espaço latente, e o decoder, que reconstrói a imagem original a partir desta representação espacial. Eles podem aprender a capturar as características de uma imagem enquanto descartam o ruído.

Vantagens na utilização de autoencoders:

- Podem ser treinados sem supervisão;
- Representar dados com alta dimensionalidade;
- Reduzem o espaço de armazenamento utilizado pelo dataset ao comprimir as imagens;

Variational Autoencoders

São um tipo de autoencoders inferencia bayesiana no processo de aprendizagem. Diferentemente dos autoencoders comuns, nos VAEs a representação latente é modelada como uma distribuição de probabilidade, o que permite gerar novos dados semelhantes aos dados de treinamento, permitindo a criação de amostras mais diversas.

Projetos práticos

Esta seção apresenta três aplicações práticas de modelos de rede neural utilizando PyTorch.

Deep Fake: Cria imagens falsas utilizando deep learning e autoencoders.

Neural Style Transfer: Aplica as características de uma imagem em outra imagem qualquer.

Image Colorization: Coloriza imagens preto e branco de forma autônoma