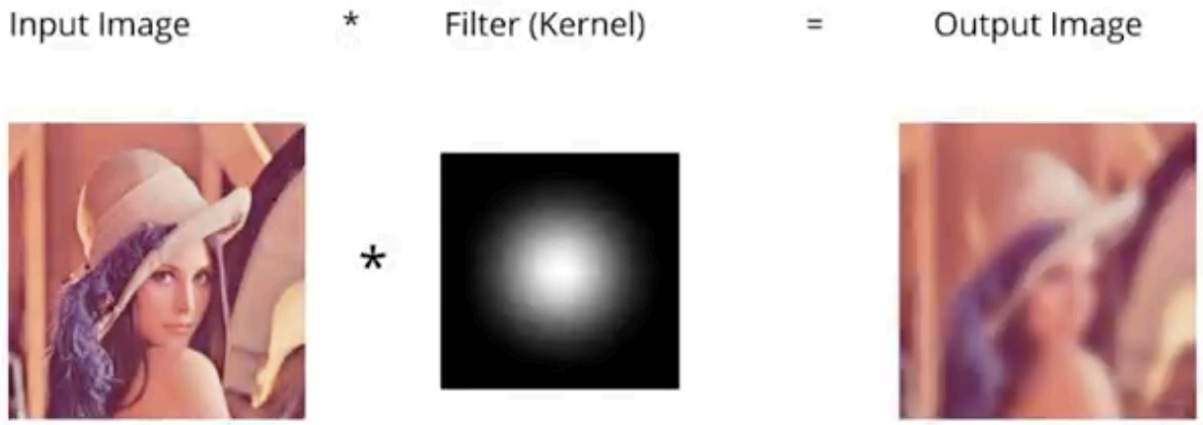
16 - Prática: Redes Neurais Convolucionais 2 (Deep Learning) (II)

Higor Miller Grassi

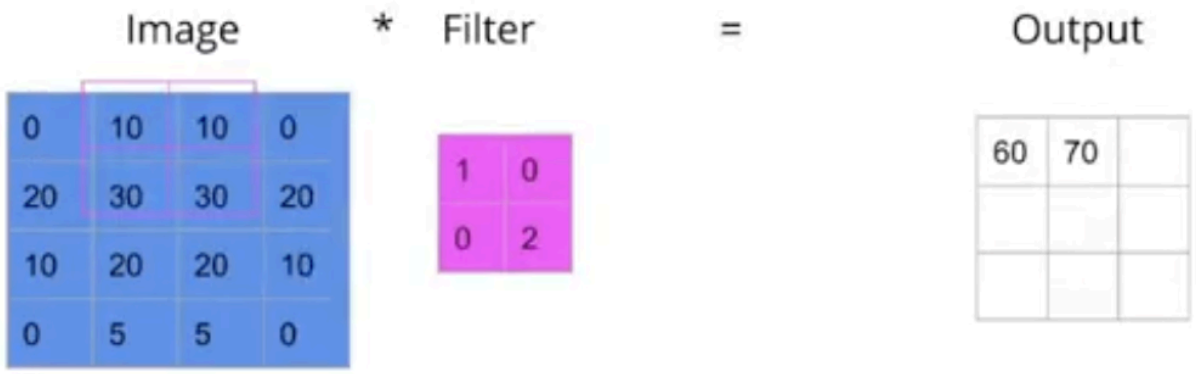
Descrição da Atividade: nas aulas foram falados de Redes Neurais Convolucionais (CNNs), explicando a aplicação de filtros para extrair características de imagens, diferentes modos de convolução (valid, same, full), e técnicas como pooling para reduzir a dimensionalidade. Também discutiu funções de perda como MSE e Binary Cross-Entropy, otimização com Adam e Momentum, e o processo de backpropagation para ajustar pesos durante o treinamento.

Convolution é um simplesmente um modificador de imagem, realizando somas e multiplicações, como mostrado na imagem a seguir, temos a imagem de entrada juntamente com a convolução do filtro,realizando operações matemáticas para executar a transformação da imagem, tendo várias técnicas diferentes para necessidades distintas , resultando na imagem de saída.

Como citado anteriormente, o que diferencia uma rede neural convolucional uma da outra é o filtro usado na aplicação, por exemplo, em um filtro gaussiano a imagem é desfocada, já em um filtro de detecção de bordas, obtém apenas a borda da imagem de entrada



O cálculo realizado no filtro depende muito na ocasião, mas dado um certo filtro, as operações matemáticas seguem a mesma lógica, dada a imagem a seguir, o cálculo é feito avançando os quadrados da imagem da esquerda para a direita, de cima para baixo, assim conseguindo o output inteiramente completo, sempre ele sendo N(image) - k(filtro) + 1.



Modos de convolução:

-Valid Mode: Não adiciona zeros ao redor das bordas, respeitando o limite da imagem de entrada, tendo uma camada de saída menor.

-Same Mode: São adicionados zeros ao redor da imagem de entrada, esta técnica chama-se padding, isso nao reduz a camada de saída

-Full Mode: O kernel e aplicado em todas as posições, isto incluindo depois do limite da imagem de entrada, resultando em uma camada de saída maior

São utilizadas um filtro para cada característica buscada, como por exemplo em um ser humano, para encontrar os olhos, boca e nariz, cada um terá o seu filtro, que melhor que procurar pelo número dentro do pixel, o que pode acarretar em falhas, eles procuram por semelhança, sendo mais preciso. Já com múltiplas camadas, mesmo havendo vários filtros, a imagem de saída e a mesma. Enquanto as redes 2D lidam com dados bidimensionais, as redes 3D processam dados tridimensionais, utilizando filtros tridimensionais para captar as relações entre as três dimensões, o que amplia significativamente a capacidade de análise e modelagem em diferentes tipos de dados e problemas.

Para implementar uma rede neural convolucional (CNN) utilizando o API funcional, podemos aproveitar bibliotecas como TensorFlow ou Keras, que oferecem suporte tanto para programação imperativa quanto para uma abordagem funcional. A programação funcional envolve o uso de funções puras e operações sem estado, e pode ser aplicada no design e treinamento da rede, como por exemplo o conv2D onde o mesmo aplica uma operação de convolução 2D sobre os dados de entrada, que são tipicamente imagens, para extrair características espaciais.

Na imagem a seguir contém a estrutura básica para a criação de uma rede neural, tendo o modelo sequencial com as camadas de entrada, camada oculta e a camada de saída com as suas células de ativação, depois temos o model.fit

responsavel por rodar a rede neural e por fim o model.predict responsável por fazer previsões do modelo treinado

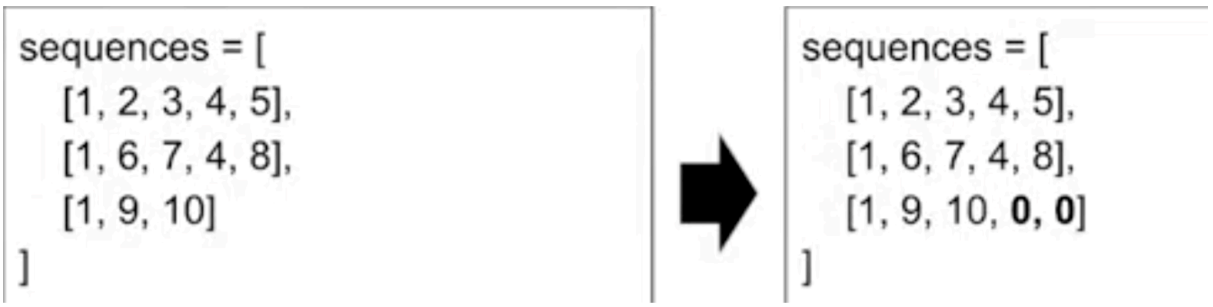
```
model = Sequential([
  Input(shape=(D,)),
  Dense(128, activation='relu'),
  Dense(K, activation='softmax'),
])

...
model.fit(...)
model.predict(...)
```

Normalização ajusta os dados para que fiquem entre 0 e 1, dividindo pelos valores máximo e mínimo, e é útil quando as variáveis têm escalas diferentes. **Padronização** transforma os dados para que tenham média 0 e desvio padrão 1, subtraindo a média e dividindo pelo desvio padrão. Ambas melhoram a performance de modelos de aprendizado de máquina, ajudando na convergência e evitando que variáveis com grandes valores dominem o modelo.

Palavras não são contínuas, diferentemente de números. Caso tenhamos uma sequência de t palavras e cada uma delas se tornar um vetor de tamanho v, obtivemos uma matriz t x v, podendo passar para a rede neural recorrente normalmente.

O one-hot seria um problema para tal ambiente pois ao ser perguntado da distância entre duas palavras sempre será a raiz quadrada do mesmo número, assim não tendo uma estrutura geométrica útil e ineficaz, fazendo as palavras não terem relações diferentes com palavras distintas.

Dito isto, temos um Embeddings, uma sequência de palavras se torna uma sequência de números inteiros após ser divindades em tokens, como as frases tem que ter o mesmo tamanho, o que “falta” para preencher usa-se o número 0, depois é mapeado para vetores para facilitar a rede neural.



- Pooling: Diminuir ainda mais a dimensionalidade dos dados e para que uma rede neural consiga fazer a detecção em ambientes diferentes, possibilitando a rede neural se adaptar e possíveis mudanças.

- **Mean Squared Error (MSE)** é frequentemente utilizada para avaliar funções de regressão, realizando o cálculo da média do quadrado da diferença entre os valores reais e os valores previstos pela função de aproximação, com o uso do quadrado tem dois objetivos principais, entre eles, garantir que os valores sejam sempre positivos, evitando o cancelamento na soma, e penalizar de forma mais significativa os erros maiores.

- Augmentação de Dados: Serve para aumentar a diversidade do seu conjunto de dados de treinamento, o que ajuda a evitar o overfitting e melhora a generalização do modelo. No contexto de redes neurais convolucionais para tarefas de imagem, a argumentação permite aplicar transformações aleatórias nas imagens durante o treinamento, criando diferentes variações das imagens originais.

- A **Binary Cross-Entropy** é uma função de perda aplicada em tarefas de classificação binária, se baseando na equação de probabilidade de Bernoulli, obtida por meio de uma função de ativação sigmoid.

- Categorical Cross-Entropy é baseada na distribuição categórica e é usada para calcular o erro quando há várias saídas possíveis. Essa função é comum em redes neurais de classificação com múltiplas saídas, com o cálculo tradicional exigindo o uso da codificação One-Hot-Encoding, que pode ser ineficiente, dito isso, há bibliotecas que implementam a Sparse One-Hot Encoding, que economiza espaço e reduz a quantidade de operações, obtida através do Softmax.

- Descida do Gradiente (Atualização de Pesos): Calcula-se a derivada parcial para mover os pesos na direção do gradiente, minimizando o erro. A fórmula do sigmoid é usada para ajustar os pesos de acordo com a derivada.

- Cálculo do Delta (Gradiente): O delta da camada de saída é obtido multiplicando o erro pela derivada da função de ativação. O delta da camada oculta é calculado com base no delta de saída, para cada registro. tendo também o ajuste dos pesos(BackPropagation)**,** com o cálculo feito de forma recursiva, ajustando os pesos da camada de saída para a entrada, considerando a quantidade de épocas definidas para o treinamento.

- **Momentum:** foi introduzido para otimizar o processo de treinamento, adicionando um efeito de "aceleração" que facilita a superação de áreas com gradientes muito pequenos, ajudando a rede a convergir mais rapidamente. Já a taxa de aprendizado adaptativa permite ajustar a taxa de aprendizado ao longo do tempo, o que significa que, se um peso estiver se atualizando muito rapidamente, a taxa de aprendizado será reduzida para evitar que a rede passe da solução ideal, prevenindo assim o fenômeno conhecido como overshooting.

-Adam: utilizado para otimização, combinando as vantagens de dois métodos de otimização, o Gradiente Estocástico (SGD) e o método de momentos, como o momentum e o RMSProp.Com o objetivo principal do Adam é ajustar os parâmetros do modelo de forma eficiente durante o treinamento, minimizando a função de perda, envolvendo o cálculo de dois momentos. O primeiro momento é a média móvel dos gradientes, que é similar ao que o momentum faz no SGD, ajudando o algoritmo a manter a direção de otimização, e o segundo momento, é a média móvel dos quadrados dos gradientes, o que ajuda a regularizar a taxa de aprendizado, adaptando-a de acordo com a magnitude do gradiente em diferentes direções.

Conclusão: a atividade proporcionou uma compreensão detalhada dos principais conceitos e técnicas envolvidos em Redes Neurais Convolucionais (CNNs), que são fundamentais para o processamento de imagens e outros dados estruturados. A utilização de filtros, modos de convolução, pooling, e a aplicação de funções de perda e otimização como Adam e Momentum são essenciais para melhorar a eficiência e a precisão do treinamento de modelos.