Lista 7 - Redes Neurais e CNN

Bruno Braga Guimarães Alves

Questão 1)

1- Sem a função de ativação em redes neurais, a rede neural seria uma simples combinação linear de seus inputs, independentemente do número de camadas, limitando severamente sua capacidade de modelar e aprender padrões complexos nos dados. As funções de ativação permitem que a rede aprenda representações de dados mais ricas e capture relações não-lineares entre os inputs e outputs.

2- A derivada da função de ativação, é essencial para o processo de treinamento da rede neural, principalmente durante a etapa de backpropagation. O backpropagation ajusta os pesos da rede para minimizar a função de perda, e isso é feito através de cálculos do gradiente da função de perda em relação a cada peso. A derivada da função de ativação é utilizada para propagar esses gradientes através das camadas da rede. Funções de ativação com derivadas bem comportadas facilitam o treinamento eficiente da rede.

3- Diferenças entre Funções de Ativação:
A função tangente hiperbólica (tanh) transforma os inputs em valores entre -1 e 1, sendo centrada em zero, o que pode acelerar o treinamento. Entretanto, pode sofrer com gradientes desvanecentes, dificultando a aprendizagem em redes profundas.

A função sigmoide converte os inputs em valores entre 0 e 1, sendo útil para modelar probabilidades, principalmente em saídas de classificadores binários. Assim como a tanh, também enfrenta o problema de gradientes desvanecentes.

A ReLU é a usada por sua simplicidade e eficácia em mitigar o problema de gradientes desvanecentes. Ela transforma os inputs negativos em zero e mantém os positivos inalterados, facilitando o treinamento de redes profundas. Contudo, pode causar "neurônios mortos", onde alguns neurônios nunca ativam e param de aprender.

A Leaky ReLU é uma variação da ReLU que permite um pequeno gradiente quando a entrada é negativa, evitando os "neurônios mortos". Isso é feito aplicando uma pequena inclinação aos valores negativos, mantendo a capacidade de aprendizagem desses neurônios.

A Softmax é usada principalmente na última camada de redes de classificação multi-classes. Ela transforma os logits (valores de entrada) em probabilidades, assegurando que a soma das saídas seja igual a 1. Embora seja útil para classificação, pode ser computacionalmente mais cara e suscetível a problemas de estabilidade numérica.

Questão 2)

1- Código alterado: https://github.com/Bruno0926/IA/blob/main/Novo%20c%C3%B3digo%20da%20CNN
Análise de desempenho: A acurácia aumentou e a perda diminuiu em comparação com a versão original, sugerindo que a nova estrutura da rede foi capaz de aprender mais características das imagens, resultando em melhor desempenho.

2-

Inicialização da Rede:

A rede neural convolucional foi inicializada utilizando a classe Sequential do Keras. Esta classe permite construir a rede camada por camada, adicionando cada uma sequencialmente.

Primeiras Camadas de Convolução e Pooling:

Foi adicionada a primeira camada de convolução com 64 filtros, cada um de tamanho 3x3, utilizando a função de ativação relu. Foi especificado o input_shape como (64, 64, 3) para imagens de entrada com 64x64 pixels e 3 canais de cor (RGB). A seguir, adicionei uma camada de pooling (MaxPooling2D) para reduzir a dimensionalidade do mapa de características, utilizando um pool size de 2x2.

Flattening e Conexão Completa:

Após as camadas de convolução e pooling, foi utilizada a camada Flatten para converter os mapas de características 2D em um vetor 1D. Em seguida, adicionei uma camada densa (totalmente conectada) com 256 unidades e função de ativação relu. A última camada é a camada de saída com uma unidade e função de ativação sigmoid, adequada para a classificação binária.

Compilação da Rede:

A rede neural foi compilada usando o otimizador adam, que é um método de otimização eficiente para funções objetivas estocásticas. A função de perda escolhida é binary_crossentropy, adequada para problemas de classificação binária. Foi utilizada a métrica accuracy para avaliar o desempenho do modelo durante o treinamento.

Pré-Processamento das Imagens:

Para treinar a rede, é essencial realizar o pré-processamento das imagens. Foi utilizada a classe ImageDataGenerator do Keras para aplicar transformações nas imagens de treino e de validação, como redimensionamento, zoom, e flip horizontal. O conjunto de treino é processado com transformações adicionais como shear e zoom, enquanto o conjunto de validação é apenas redimensionado.

Treinamento da Rede:

A rede foi treinada com o conjunto de treino e validamos com o conjunto de validação. Foram utilizados 8000 passos por época para o treinamento e 2000 passos para a validação. Realizamos cinco épocas de treinamento.

Resultados e Desempenho:

Comparando esses resultados com a estrutura original da rede, foi observado um aumento na acurácia e uma diminuição na perda. Essas melhorias indicam que a nova estrutura da rede foi capaz de aprender mais características das imagens.