Inteligência Artificial

Lista 07 – Redes Neurais e CNN

Bernard Paes Ferreira[1]
[1] Instituto de Ciências Exatas e Informática - Pontifícia Universidade Católica de Minas Gerais (PUC-MG) - Belo Horizonte - MG – Brasil

Questão 01: Faça um texto sobre funções de ativação usualmente empregadas em redes neurais, apresentando qual o papel da função de ativação e da derivada da função. Além disso, apresente as principais diferenças entre as funções: Tangente Hiperbólica, Sigmoide, RELU, softmax e Leaky Relu.

As **funções de ativação** em redes neurais desempenham um papel crucial na introdução de não-linearidades aos modelos, permitindo que aprendam e representem relações complexas nos dados. Sem essas funções, uma rede neural, independentemente de seu número de camadas, se comportaria como uma simples combinação linear dos inputs, limitando drasticamente sua capacidade de resolver problemas complexos. A função de ativação é, portanto, uma componente essencial na estrutura de uma rede neural, garantindo que ela possa capturar e modelar padrões não lineares nos dados de entrada.

A **derivada da função de ativação**, é fundamental para o processo de aprendizado das redes neurais, principalmente durante a retropropagação. A retropropagação é um algoritmo utilizado para ajustar os pesos das conexões na rede de forma a minimizar o erro na saída. Durante este processo, os pesos são atualizados com base nos gradientes calculados das funções de ativação. Funções de ativação com derivadas eficientes permitem que os gradientes sejam calculados e propagados de forma adequada, garantindo que o aprendizado ocorra de maneira estável e eficiente. Gradientes são valores que indicam a direção e a magnitude das mudanças necessárias nos pesos para reduzir o erro da rede. Derivadas que resultam em valores muito pequenos podem levar ao problema do "desvanecimento do gradiente", onde os gradientes se tornam tão pequenos que praticamente não ajustam os pesos, retardando o aprendizado. Por outro lado, derivadas muito grandes podem causar a “explosão do gradiente", onde os gradientes se tornam extremamente grandes, causando grandes ajustes nos pesos que podem desestabilizar a rede.

A **função tangente hiperbólica, ou tanh**, mapeia valores de entrada para um intervalo entre -1 e 1. Essa característica centraliza os dados em torno de zero, o que pode facilitar a convergência durante o treinamento. A centralização dos dados ajuda a estabilizar a rede, pois os valores de saída das camadas intermediárias são mais balanceados. No entanto, assim como a função sigmoide, a tanh sofre com o problema do desvanecimento do gradiente para valores extremos de entrada, onde suas derivadas se aproximam de zero. Quando os valores de entrada são muito grandes ou muito pequenos, a função tanh satura, tornando os gradientes muito pequenos e dificultando a atualização eficaz dos pesos.

A função **sigmoide**, por sua vez, mapeia a entrada para um intervalo entre 0 e 1, sendo amplamente utilizada em saídas de redes neurais onde se deseja um valor probabilístico. A saída da função sigmoide pode ser interpretada como uma probabilidade, o que é particularmente útil em tarefas de classificação binária. No entanto, devido à saturação da função para valores de entrada muito positivos ou negativos, os gradientes podem se tornar muito pequenos, dificultando o treinamento profundo. Assim como a tanh, a sigmoide sofre com o problema do desvanecimento do gradiente, especialmente em redes profundas onde os valores de entrada podem se tornar muito grandes ou muito pequenos após várias camadas.

A **ReLU (Rectified Linear Unit)** é uma função de ativação popular que mapeia a entrada para valores não-negativos, mantendo valores positivos inalterados e mapeando valores negativos para zero. Sua simplicidade e eficiência em evitar o problema do desvanecimento do gradiente fazem dela uma escolha comum para muitas arquiteturas de redes neurais. A ReLU permite que as redes neurais aprendam rapidamente, pois não satura para valores positivos. No entanto, pode sofrer do problema de "neurônios mortos", onde neurônios que entram na região zero da função não voltam a ativar. Neurônios mortos são aqueles que sempre produzem uma saída de zero e, portanto, deixam de contribuir para o aprendizado da rede.

A função **Softmax** é utilizada principalmente na camada de saída de redes neurais para problemas de classificação multiclasse. Ela mapeia um vetor de valores de entrada para uma distribuição de probabilidade, onde a soma de todas as probabilidades é igual a 1. Isso facilita a interpretação das saídas da rede como probabilidades associadas a diferentes classes. A Softmax transforma as pontuações de entrada em valores que podem ser interpretados diretamente como probabilidades, tornando-a ideal para tarefas onde a previsão precisa ser uma distribuição de probabilidade sobre várias classes mutuamente exclusivas.

A **Leaky ReLU** é uma variação da ReLU que introduz um pequeno gradiente para valores negativos de entrada, evitando o problema de neurônios mortos. Essa função mapeia valores positivos de forma idêntica à ReLU, mas para valores negativos, introduz uma pequena inclinação, permitindo que o aprendizado continue. A inclinação introduzida na Leaky ReLU impede que os neurônios fiquem completamente inativos, garantindo que eles possam eventualmente recuperar sua capacidade de aprendizado. Isso torna a Leaky ReLU uma escolha robusta para muitas aplicações onde a ReLU tradicional pode falhar.

Cada uma dessas funções de ativação apresenta características únicas que as tornam adequadas para diferentes tipos de problemas e arquiteturas de redes neurais. Assim como um aplicativo bem projetado, essas funções de ativação visam proporcionar uma "experiência de aprendizado" fluida e eficiente, adaptando-se às necessidades específicas dos modelos e facilitando a navegação pelo espaço de soluções possíveis. A escolha da função de ativação correta é essencial para garantir que a rede neural opere de forma otimizada, tal como uma interface intuitiva facilita a experiência do usuário, garantindo que as operações e interações sejam realizadas de maneira eficiente e eficaz.

Questão 02: Realizar alterações na estrutura da CNN, alterando o número de camadas de convolução, pooling, kernels etc, e escrever um relatório apontando as alterações realizadas, bem como, os resultados obtidos.

CNN: https://www.deeplearningbook.com.br/reconhecimento-de-imagens-com-redes-neurais-convolucionais-em-python-parte-4/

Base de Dados: https://www.kaggle.com/datasets/williamu32/dataset-bart-or-homer?resource=download

Github: https://github.com/BernardPaes/Lista07

Motivos das Escolhas e Alterações Realizadas

Para otimizar o desempenho de Redes Neurais Convolucionais (CNNs), a seleção e a variação dos hiperparâmetros são cruciais. As CNNs dependem fortemente da configuração dos hiperparâmetros, que determinam a capacidade do modelo de extrair características e aprender padrões complexos dos dados. Nesta análise, cinco diferentes configurações de hiperparâmetros foram testadas para observar seu impacto na performance da rede.

Hiperparâmetros Testados

1. **Modelo 1:**
    - Camadas de Convolução: 2
    - Filtros por Camada: 32
    - Tamanho do Kernel: (3, 3)
    - Unidades na Camada Densa: 128
    - Dropout: 0.5
2. **Modelo 2:**
    - Camadas de Convolução: 3
    - Filtros por Camada: 64
    - Tamanho do Kernel: (3, 3)
    - Unidades na Camada Densa: 256
    - Dropout: 0.5
3. **Modelo 3:**
    - Camadas de Convolução: 2
    - Filtros por Camada: 32
    - Tamanho do Kernel: (5, 5)
    - Unidades na Camada Densa: 128
    - Dropout: 0.3

4. **Modelo 4:**

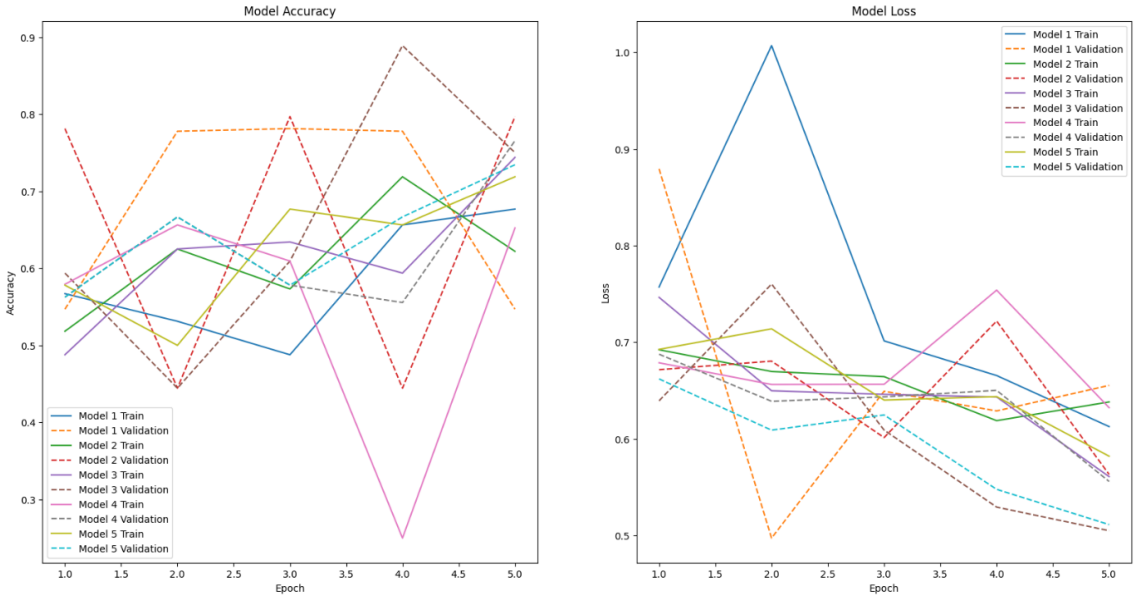
    - Camadas de Convolução: 4
    - Filtros por Camada: 64
    - Tamanho do Kernel: (3, 3)
    - Unidades na Camada Densa: 512
    - Dropout: 0.5

5. **Modelo 5:**

    - Camadas de Convolução: 3
    - Filtros por Camada: 32
    - Tamanho do Kernel: (3, 3)
    - Unidades na Camada Densa: 128
    - Dropout: 0.2

Resultados Obtidos

Os gráficos comparativos mostram a acurácia e a perda (loss) dos cinco modelos ao longo de 5 épocas de treinamento e validação.



Acurácia

A acurácia dos modelos variou significativamente ao longo das épocas:

- **Modelo 1:** Iniciou com uma acurácia baixa, mas melhorou com o tempo. A alta variação indica que o modelo ainda está ajustando seus parâmetros.
- **Modelo 2:** Demonstrou uma melhoria mais estável na acurácia, tanto no conjunto de treino quanto no de validação, sugerindo uma configuração mais robusta.
- **Modelo 3:** Apresentou uma acurácia instável, com variações menos consistentes comparadas ao Modelo 2.

- **Modelo 4:** A acurácia melhorou rapidamente, mas com flutuações, indicando um possível overfitting devido à alta capacidade do modelo.
- **Modelo 5:** Mostrou uma melhoria gradual na acurácia, mas com algumas flutuações, especialmente na validação.

Perda (Loss)

A perda dos modelos também apresentou variação considerável:

- **Modelo 1:** A perda inicial foi alta, reduzindo de forma variável ao longo das épocas.
- **Modelo 2:** Apresentou uma perda mais estável, diminuindo gradualmente, o que sugere um bom ajuste dos parâmetros.
- **Modelo 3:** A perda foi bastante variável, indicando dificuldades na aprendizagem das características dos dados.
- **Modelo 4:** A perda diminuiu rapidamente, mas com flutuações, novamente sugerindo overfitting.
- **Modelo 5:** Apresentou uma redução gradual na perda, com algumas flutuações.

Interpretação dos Resultados

1. **Número de Camadas de Convolução:** Modelos com mais camadas de convolução (Modelos 2, 4 e 5) tendem a apresentar um desempenho superior, indicando que camadas adicionais ajudam a capturar mais características dos dados.

2. **Número de Filtros e Tamanho do Kernel:** O aumento no número de filtros (Modelos 2 e 4) mostrou melhorias na acurácia e uma perda mais estável, sugerindo que mais filtros podem capturar melhor as características dos dados.

3. **Unidades na Camada Densa e Dropout:** A combinação de mais unidades na camada densa e uma taxa de dropout adequada ajuda a evitar overfitting. No entanto, uma taxa de dropout muito alta (Modelo 1) ou muito baixa (Modelo 5) pode prejudicar a performance.

4. **Flutuações:** As flutuações significativas nas métricas indicam que mais épocas de treinamento podem ser necessárias para observar padrões mais claros e estáveis. Isso também sugere que ajustes finos nos hiperparâmetros (como a taxa de aprendizado) poderiam melhorar a estabilidade.

Os resultados indicam que configurações mais complexas, como as dos Modelos 2 e 4, tendem a apresentar um desempenho superior, embora possam correr o risco de overfitting. A escolha dos hiperparâmetros é crucial e deve ser ajustada com base nos dados específicos e na tarefa em questão. Mais experimentos e ajustes finos podem ajudar a identificar a configuração ideal para maximizar a performance da CNN.