Relatório da Classificação de Imagens com Redes Neurais Convolucionais - Lista 12

Samuel Horta de Faria
801528
https://github.com/SamuelHortadeFaria/IA_Lista12

Junho de 2025

Sumário

1 Introdução

A classificação de imagens é um dos problemas fundamentais no campo da visão computacional e da inteligência artificial. O desafio consiste em treinar um modelo computacional para categorizar uma imagem com base em seu conteúdo visual. O problema "Dogs vs. Cats", popularizado pela plataforma Kaggle, serve como um excelente caso de estudo para a aplicação de técnicas de aprendizado profundo (*Deep Learning*).

O objetivo deste trabalho foi construir e treinar uma Rede Neural Convolucional (CNN), um tipo de arquitetura de rede neural especialmente projetada para processar dados com uma topologia de grade, como as imagens. As CNNs são o estado da arte para muitas tarefas de visão computacional, pois são capazes de aprender hierarquias de características espaciais de forma automática.

Este relatório documenta a metodologia empregada, dividida nas seguintes etapas:

1. **Preparação dos Dados:** Organização dos arquivos de imagem e extração de rótulos.
2. **Pré-processamento:** Divisão dos dados, normalização e aplicação de *data augmentation*.
3. **Construção do Modelo:** Definição da arquitetura da CNN, camada por camada.
4. **Treinamento e Avaliação:** Compilação do modelo, treinamento e análise das métricas de desempenho.
5. **Predição:** Utilização do modelo treinado para classificar novas imagens.

2 Metodologia

A solução foi desenvolvida em um ambiente Jupyter Notebook, utilizando Python como linguagem de programação e o framework TensorFlow com sua API de alto nível, Keras.

2.1 Preparação dos Dados

O conjunto de dados consiste em uma pasta de treino com 25.000 imagens, nomeadas como `cat.i.jpg` ou `dog.i.jpg`, e uma pasta de teste com 12.500 imagens numeradas.

O primeiro passo foi estruturar os caminhos dos arquivos e seus respectivos rótulos. Isso foi feito lendo os nomes dos arquivos do diretório de treino e extraindo a classe ("cat"ou "dog") de cada um. Essas informações foram organizadas em um DataFrame da biblioteca `pandas`, facilitando a manipulação e a divisão posterior dos dados. Os rótulos textuais foram convertidos para um formato numérico binário: **0 para gato** e **1 para cachorro**.

2.2 Pré-processamento e Geração de Dados

Antes de alimentar a rede neural, as imagens precisam ser pré-processadas. Para avaliar o modelo de forma robusta e evitar o *overfitting* (sobreajuste), o conjunto de dados de treino foi dividido em dois subconjuntos:

- **Conjunto de Treinamento (90%):** Usado para ajustar os pesos do modelo.
- **Conjunto de Validação (10%):** Usado para avaliar o desempenho do modelo em dados não vistos durante o treinamento a cada época.

A ferramenta `ImageDataGenerator` do Keras foi fundamental nesta etapa. Ela foi configurada para realizar duas tarefas principais:

1. Normalização dos Pixels: Os valores dos pixels de cada imagem, que variam de 0 a 255, foram reescalados para o intervalo [0, 1] dividindo-os por 255. Isso ajuda a estabilizar e acelerar o processo de treinamento.

2. Data Augmentation: Para o conjunto de treino, foram aplicadas transformações aleatórias em tempo real nas imagens (rotações, zooms, cisalhamentos, inversões horizontais). Essa técnica aumenta artificialmente a diversidade do conjunto de dados, forçando o modelo a aprender características mais gerais e robustas, o que melhora sua capacidade de generalização e combate o *overfitting*.

2.3 Arquitetura do Modelo

Foi projetada uma arquitetura de CNN sequencial, composta por blocos de convolução e pooling, seguidos por camadas densas para classificação. A estrutura é a seguinte:

- **Bloco Convolucional 1:** Uma camada `Conv2D` com 32 filtros (kernel 3x3) e ativação ReLU, seguida por uma camada `MaxPooling2D` (pool 2x2).
- **Bloco Convolucional 2:** Uma camada `Conv2D` com 64 filtros e ativação ReLU, seguida por `MaxPooling2D`.
- **Bloco Convolucional 3:** Uma camada `Conv2D` com 128 filtros e ativação ReLU, seguida por `MaxPooling2D`.
- **Camada de Achatamento (Flatten):** Transforma os mapas de características 2D resultantes em um vetor 1D.
- **Camadas de Classificação:** Uma camada `Dense` com 512 neurônios e ativação ReLU, uma camada `Dropout` com taxa de 0.5 para regularização, e a camada de saída `Dense` com 1 neurônio e ativação `sigmoid`, ideal para classificação binária.

O resumo da arquitetura é apresentado abaixo:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 32)      896

 max_pooling2d (MaxPooling2  (None, 63, 63, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 61, 61, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 30, 30, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 28, 28, 128)       73856

 max_pooling2d_2 (MaxPoolin  (None, 14, 14, 128)       0
 g2D)

 flatten (Flatten)           (None, 25088)             0

 dense (Dense)               (None, 512)               12845568

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 1)                 513

=================================================================
Total params: 12,939,329
Trainable params: 12,939,329
Non-trainable params: 0
_________________________________________________________________
```

Listing 1: Resumo da Arquitetura da CNN

3 Treinamento e Compilação

Antes do treinamento, o modelo foi compilado com os seguintes parâmetros:

- **Otimizador:** `adam`, um algoritmo de otimização eficiente e amplamente utilizado que adapta a taxa de aprendizado.
- **Função de Perda:** `binary_crossentropy`, a escolha padrão para problemas de classificação binária.
- **Métricas:** `accuracy`, para monitorar a acurácia do modelo durante o treinamento e a validação.

O modelo foi treinado por 15 épocas, utilizando os geradores de dados criados na etapa de pré-processamento.

4 Resultados e Análise

O desempenho do modelo foi monitorado ao longo das épocas através da acurácia e da perda (loss) nos conjuntos de treinamento e validação. A acurácia de treina-

mento e validação aumentam de forma consistente, enquanto as perdas diminuem. A proximidade entre as curvas de treino e validação indica que as técnicas de regularização (Dropout e Data Augmentation) foram eficazes em mitigar o *overfitting*, permitindo que o modelo generalize bem para dados não vistos.

5 Predição e Geração de Submissão

Após o treinamento, o modelo foi utilizado para prever as classes das 12.500 imagens do conjunto de teste. Um gerador de dados foi criado para este conjunto, aplicando apenas a normalização de pixels. As predições geradas pela camada `sigmoid` (valores de probabilidade) foram convertidas para classes binárias (0 ou 1) usando um limiar de 0.5.

Finalmente, os resultados foram formatados em um arquivo `submission.csv` com duas colunas, `id` e `label`, conforme especificado pelas regras da competição.

6 Conclusão

O projeto demonstrou com sucesso a aplicação de uma Rede Neural Convolucional para um problema prático de visão computacional. O pipeline implementado, abrangendo desde o tratamento dos dados até a avaliação do modelo, resultou em um classificador robusto e com boa capacidade de generalização.

Como trabalhos futuros, o modelo poderia ser aprimorado através de:

- **Transfer Learning:** Utilizar uma arquitetura de CNN pré-treinada em um conjunto de dados maior (como ImageNet) e ajustá-la para este problema específico.
- **Ajuste de Hiperparâmetros:** Otimizar parâmetros como a taxa de aprendizado, o tamanho dos lotes e a arquitetura da rede.
- **Aumento do Treinamento:** Treinar o modelo por um número maior de épocas para permitir uma convergência mais fina dos pesos.