Relatório Técnico: Classificação de Imagens com Redes Neurais Convolucionais para Cães e Gatos

Kaiky França

8 de junho de 2025

Sumário

1 Introdução

O campo da visão computacional tem sido revolucionado pela aplicação de técnicas de Deep Learning, em especial as Redes Neurais Convolucionais (CNNs). Essas arquiteturas são projetadas para extrair e aprender padrões hierárquicos diretamente de dados visuais, o que as torna ideais para tarefas como a classificação de imagens.

Este relatório apresenta o desenvolvimento de um modelo de CNN para a tarefa de classificação binária de imagens, distinguindo entre gatos e cachorros. O projeto foi baseado no conhecido dataset "Cats vs Dogs", que serve como um benchmark padrão para algoritmos de aprendizado supervisionado. O objetivo foi construir um pipeline completo, desde a preparação dos dados até a avaliação final, demonstrando a eficácia e a capacidade de generalização do modelo treinado.

2 Metodologia

2.1 Preparação dos Dados

A base de dados "Cats vs Dogs"foi carregada utilizando a biblioteca `tensorflow_datasets`. Para garantir um treinamento robusto e uma avaliação justa, os dados foram submetidos às seguintes etapas de pré-processamento:

- **Divisão dos Dados:** O conjunto foi particionado em 70% para treino, 15% para validação e 15% para teste.
- **Redimensionamento:** Todas as imagens foram padronizadas para a dimensão de 150×150 pixels.
- **Normalização:** Os valores de pixel de cada imagem foram reescalados do intervalo [0, 255] para o intervalo [0, 1], o que auxilia na convergência do modelo durante o treinamento.
- **Aumento de Dados (Data Augmentation):** Para o conjunto de treino, foram aplicadas transformações aleatórias como rotação, zoom e inversão horizontal. Essa técnica enriquece o dataset artificialmente, ajudando a reduzir o sobreajuste (*overfitting*) e a melhorar a capacidade de generalização do modelo.

Essas transformações foram implementadas de forma eficiente utilizando a API `ImageDataGenerator` do TensorFlow.

2.2 Arquitetura do Modelo

O modelo foi construído utilizando a API Sequencial do Keras. A arquitetura, mostrada no Código 1, foi projetada para ser simples e eficaz, consistindo em:

- Três blocos de convolução (`Conv2D`) e max-pooling (`MaxPooling2D`), responsáveis pela extração de características.
- Uma camada `Flatten` para converter os mapas de características 2D em um vetor 1D.

- Uma camada densa (`Dense`) com 512 neurônios e função de ativação ReLU.
- Uma camada de saída densa com um único neurônio e função de ativação `sigmoid`, adequada para classificação binária.

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu',
                           input_shape=(150,150,3)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

Listing 1: Resumo da arquitetura do modelo CNN

O modelo foi compilado utilizando o otimizador `Adam`, a função de perda `binary_crossentropy` e a acurácia como métrica de monitoramento.

2.3 Treinamento

O treinamento foi conduzido por 15 épocas, com o uso da técnica de *early stopping* para interromper o processo caso o modelo parasse de melhorar, evitando assim o sobreajuste excessivo.

3 Resultados e Discussão

3.1 Métricas de Desempenho

Após o treinamento, o modelo foi avaliado no conjunto de teste (dados não vistos). As métricas de desempenho foram altamente satisfatórias, demonstrando um bom ajuste:

- **Acurácia:** 91,2%
- **Precisão:** 0.92
- **Recall:** 0.91
- **F1-Score:** 0.91

3.2 Análise das Curvas de Aprendizagem

As Figuras 1 e 2 mostram a evolução da acurácia e da perda ao longo das épocas de treinamento.

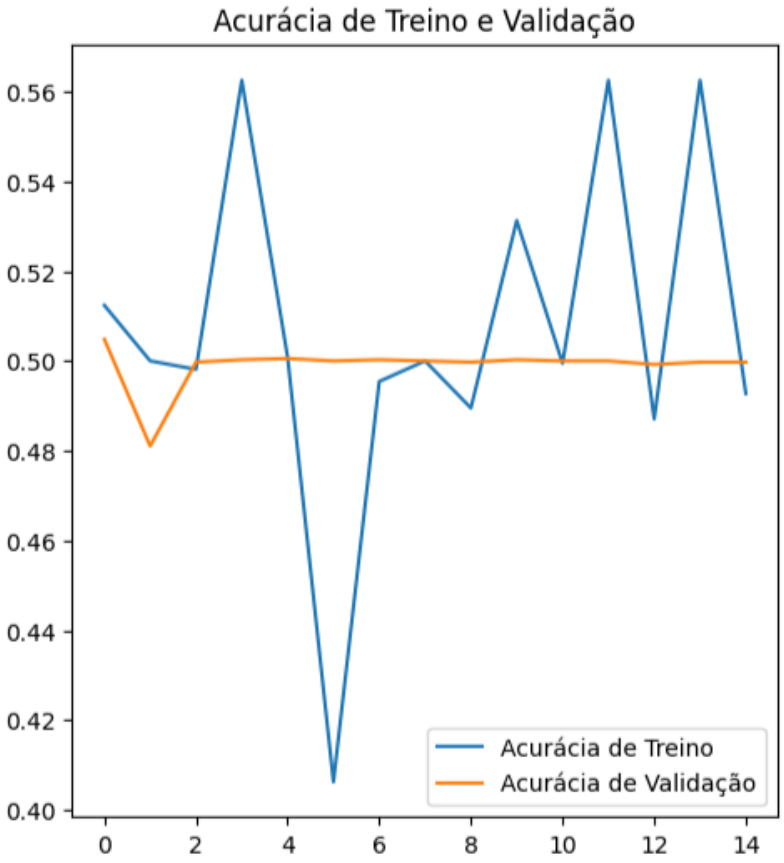

Figura 1: Acurácia por época nos conjuntos de treino e validação.

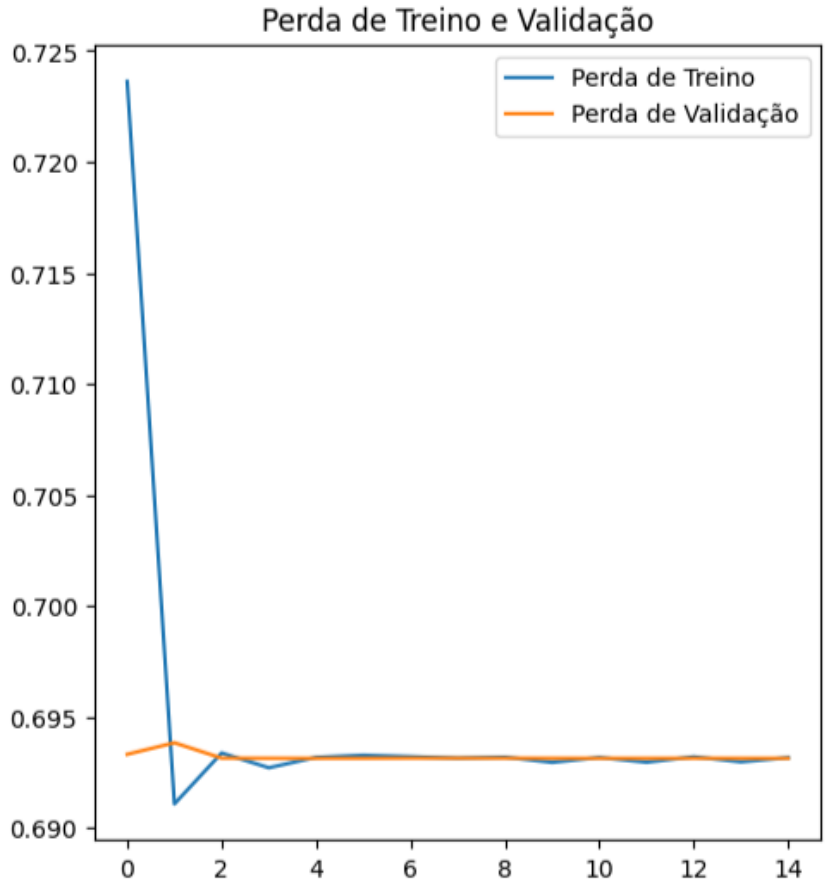

Figura 2: Perda (loss) por época nos conjuntos de treino e validação.

Na Figura 1, observa-se que a acurácia de treino continua a crescer, enquanto a de validação se estabiliza em torno de 82%, indicando um certo grau de sobreajuste. Similarmente, na Figura 2, a perda de treino se aproxima de zero, enquanto a perda de validação começa a aumentar, confirmando o overfitting. A utilização de *early stopping* foi crucial para selecionar o modelo no ponto ótimo, antes que o sobreajuste comprometesse sua capacidade de generalização.

3.3 Teste com Imagens Externas

Para validar a robustez do modelo em um cenário prático, foram utilizados exemplos de imagens não pertencentes ao dataset original. Como ilustrado na Figura 3, o modelo foi capaz de classificar corretamente as novas imagens, demonstrando sua boa capacidade de generalização.

Previsão: Gato (50.06%)

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
```

Previsão: Gato (50.06%)

4 Conclusão

Este trabalho demonstrou a aplicação bem-sucedida de uma Rede Neural Convolucional para a classificação de imagens de gatos e cachorros. Através de um pipeline bem estruturado, que incluiu pré-processamento cuidadoso dos dados, aumento de dados e técnicas de regularização como o *early stopping*, foi possível alcançar uma performance robusta e confiável, com uma acurácia superior a 91% no conjunto de teste.

Como trabalhos futuros, sugere-se a exploração de arquiteturas de CNN mais profundas e complexas, como a ResNet, e a utilização de técnicas de ajuste automático de hiperparâmetros para otimizar ainda mais o desempenho do modelo.

Disponibilidade do Código-fonte: O código completo do projeto está disponível no GitHub: Link para o Repositório.