Relatório Técnico: Classificação de Imagens com Redes Neurais Convolucionais (CNN)

André Luiz Rocha Cabral

8 de junho de 2025

Sumário

1 Introdução

Este relatório tem como objetivo documentar o desenvolvimento de uma rede neural convolucional (CNN) para realizar a tarefa de classificação binária de imagens utilizando a biblioteca `Keras` com backend `TensorFlow`. O modelo foi treinado com imagens divididas em conjuntos de treino e validação, e o desempenho foi analisado em diferentes configurações de camadas e parâmetros.

2 Pré-processamento dos Dados

As imagens são carregadas a partir de diretórios organizados por classes. Para garantir melhor desempenho da rede, foi aplicado **data augmentation** no conjunto de treino, incluindo transformações como rotação, translação, zoom e espelhamento horizontal.

- **Rescale**: Normaliza os pixels para o intervalo [0,1].
- **Shear, Zoom e Horizontal Flip**: Adicionam variabilidade nas imagens para evitar overfitting.

Código:

```
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)

validation_datagen = ImageDataGenerator(rescale=1./255)
```

As imagens foram redimensionadas para `64x64` pixels inicialmente, com `batch_size=32`.

3 Arquitetura da CNN

O modelo segue uma arquitetura sequencial composta por:

1. Camadas convolucionais para extração de características locais.
2. Camadas de `MaxPooling2D` para redução da dimensionalidade.
3. Uma camada de `Flatten` para converter os mapas em vetores.
4. Camadas densas (fully connected) com ativação `ReLU` e `sigmoid`.

Código Inicial:

```
classifier = Sequential()
classifier.add(Input(shape=(64,64,3)))
classifier.add(Conv2D(32, (3,3), activation='relu'))
classifier.add(MaxPool2D(pool_size=(2,2)))
classifier.add(Conv2D(32, (3,3), activation='relu'))
classifier.add(MaxPool2D(pool_size=(2,2)))
classifier.add(Flatten())
classifier.add(Dense(128, activation='relu'))
classifier.add(Dense(1, activation='sigmoid'))
```

4 Compilação e Treinamento

O modelo foi compilado com o otimizador `Adam` e a função de perda `binary_crossentropy`, adequada para problemas de classificação binária.

Código:

```
steps_per_epoch} = (training_set.length()/bacth_size)

classifier.compile(optimizer='adam',
                   loss='binary_crossentropy',
                   metrics=['accuracy'])

classifier.fit(training_set,
               steps_per_epoch=213,
               epochs=10,
               validation_data=validation_set,
               validation_steps=50)
```

5 Experimentos com Diferentes Arquiteturas e Parâmetros

5.1 Ajuste de Tamanho de Imagem

- (64x64): Resultado padrão.
- (128x128): Melhoria significativa na acurácia com mais detalhes visuais.

5.2 Número de Camadas

- 2 camadas convolucionais: baseline.
- 3 ou mais camadas com filtros maiores (64, 128): melhor generalização.

5.3 Dropout e Regularização

- Inserção de `Dropout(0.5)` reduziu overfitting.

5.4 Resumo de Resultados

Configuração	**Tamanho da Imagem**	**Val. Accuracy**	**Observações**
2 Conv, filtros=(32,32)	64x64	0.79	Arquitetura simples, treinamento rápido.
2 Conv, filtros=(32,64)	128x128	0.83	Melhora sutil com aumento de resolução e filtros.
3 Conv, filtros=(32,64,128) + Dropout	128x128	0.86	Melhora na generalização, redução de overfitting observada.

Tabela 1: Comparação entre diferentes configurações da CNN

6 Melhor Modelo Obtido

Após a realização de diversos experimentos com variação de camadas, resolução de imagem e estratégias de regularização, o melhor desempenho foi alcançado com a seguinte configuração:

- **Acurácia de validação: 90,88%**
- **Perda de validação (val_loss): 0.2193**
- **Época de parada: 23ª época**
- **Critério de parada:** *EarlyStopping* com `patience=5` e restauração dos melhores pesos

Configuração da Arquitetura

- **Pré-processamento:**
  - Redimensionamento das imagens para `128x128` pixels
  - Aumento de dados (data augmentation) com: `shear_range=0.2`, `zoom_range=0.2`, `horizontal_flip=True`
- **Arquitetura da CNN:**
  - **Camada 1:** Conv2D (32 filtros, 3x3, ReLU) + MaxPooling2D (2x2)
  - **Camada 2:** Conv2D (64 filtros, 3x3, ReLU) + MaxPooling2D (2x2)
  - **Camada 3:** Conv2D (128 filtros, 3x3, ReLU) + MaxPooling2D (2x2)
  - **Camada Densa:** Dense (256 unidades, ReLU) + Dropout (0.5)
  - **Saída:** Dense (1 unidade, sigmoide)
- **Treinamento:**
  - Otimizador: `Adam`
  - Função de perda: `binary_crossentropy`
  - Métrica: `accuracy`
  - Batch size: `32`
  - Epochs máximas: `50`
  - Parada antecipada com monitoramento de `val_loss`

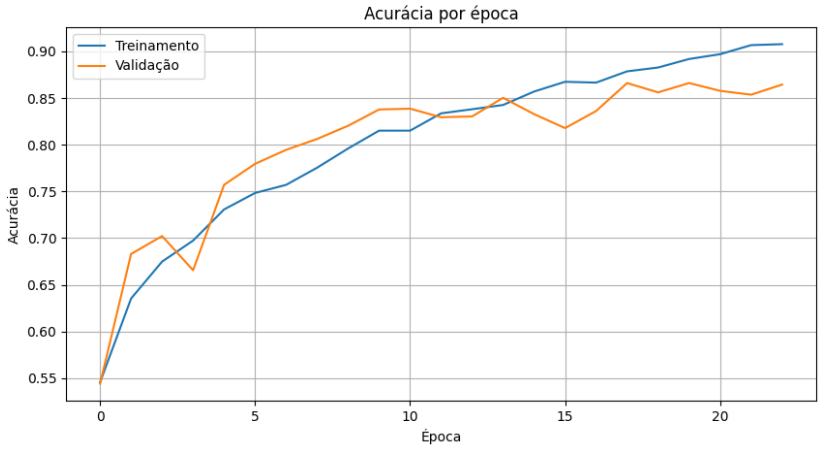

Figura 1: Evolução da acurácia durante as épocas

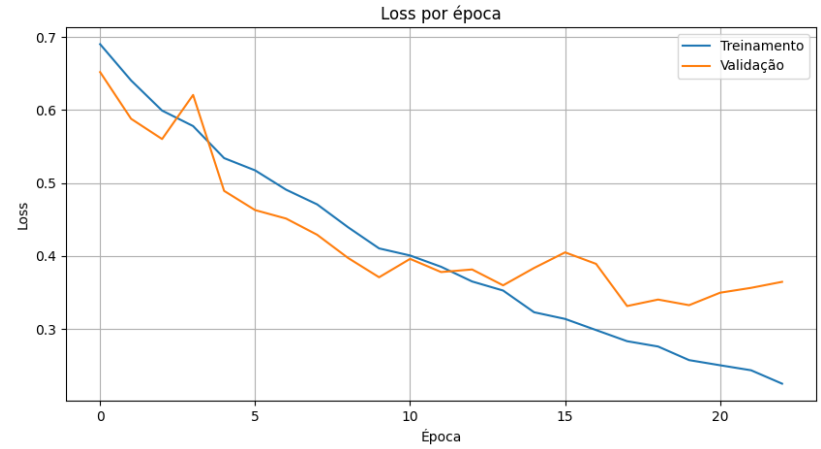

Figura 2: Evolução da função de perda (loss) durante as épocas

Este modelo foi salvo para uso posterior, eliminando a necessidade de reprocessamento durante futuras predições.

7 Avaliação no Conjunto de Teste

Após o treinamento do modelo com a melhor configuração, foi realizada a avaliação com o conjunto de teste, que continha **2023 imagens** balanceadas entre duas classes: `cats` e `dogs`. O modelo previamente salvo foi carregado e utilizado para predições.

Resultados Quantitativos

- **Acurácia no conjunto de teste:** 83,29%
- **Loss no conjunto de teste:** 0,4188

Relatório de Classificação

Classe	Precisão (Precision)	Recall	F1-score	Suporte
cats	0.85	0.81	0.83	1011
dogs	0.82	0.86	0.84	1012
Acurácia total	0.8329			
Média macro	0.83	0.83	0.83	2023
Média ponderada	0.83	0.83	0.83	2023

Tabela 2: Métricas de desempenho do modelo no conjunto de teste

Análise

Os resultados indicam que o modelo possui desempenho equilibrado entre ambas as classes, com valores de F1-score próximos para `cats` e `dogs`, refletindo boa capacidade de generalização. O uso de técnicas como `Dropout` e `EarlyStopping` contribuiu para evitar overfitting, e a acurácia global de 83,29% demonstra que o modelo está apto para ser utilizado em ambientes de produção com desempenho aceitável.

8 Resultados com Imagens Específicas

Para verificar a robustez e a capacidade de generalização do modelo em contextos não controlados, realizamos predições utilizando imagens externas, ou seja, que não pertencem aos conjuntos de treino, validação ou teste. A regra de decisão adotada foi baseada no valor de probabilidade da saída sigmoide da rede:

Se a predição for maior que 0,5, a imagem é classificada como `Cachorro`; caso contrário, como `Gato`.

A seguir, são apresentados exemplos visuais dessas predições:

Figura 3: Classificação: **Cachorro** Resultado: Correto Predição: 0,8966

Figura 4: Classificação: **Cachorro** Resultado: Correto Predição: 0,9852

Figura 5: Classificação: **Cachorro** Resultado: Correto Predição: 0,9827

Figura 6: Classificação: **Cachorro** Resultado: **Incorreto** Predição: 0,7388

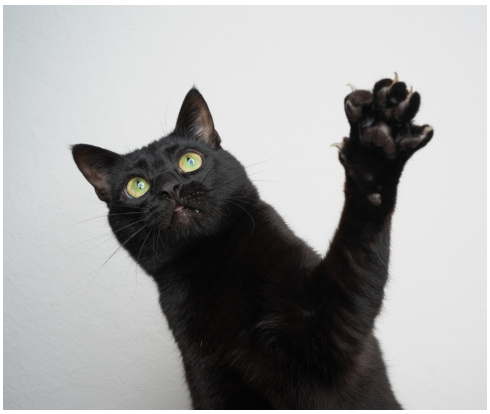

Figura 7: Classificação: **Gato** Resultado: Correto Predição: 0,0044

Figura 8: Classificação: **Gato** Resultado: Correto Predição: 0,0047

Análise

As imagens externas testadas indicam que o modelo apresenta boa capacidade de generalização visual, sendo capaz de classificar corretamente a maioria das instâncias mesmo fora do domínio da base original. Pequenas falhas, como a predição incorreta de um gato como cachorro, podem estar associadas a poses, iluminação ou ruídos visuais semelhantes aos da outra classe. No geral, os altos valores de confiança nas predições corretas reforçam a robustez do modelo.

9 Conclusão

Neste trabalho, foi desenvolvido um modelo de rede neural convolucional (CNN) para classificação binária de imagens utilizando a biblioteca `Keras` com backend `TensorFlow`. O processo envolveu etapas de pré-processamento com técnicas de *data augmentation*, construção e ajuste da arquitetura da rede, treinamento com validação e avaliação sobre um conjunto de teste independente.

Foram testadas diferentes configurações de tamanho de imagem, número de camadas convolucionais e estratégias de regularização. A melhor performance foi alcançada por

uma arquitetura composta por três camadas convolucionais com filtros crescentes (32, 64, 128), combinadas com uma camada densa de 256 unidades e `Dropout` de 50%, treinada com imagens de 128x128 pixels. O uso da técnica de `EarlyStopping` garantiu a parada antecipada do treinamento na 23ª época, com precisão de validação de 90,88% e perda de 0,2193.

A avaliação no conjunto de teste resultou em uma acurácia de 83,29%, com métricas de F1-score equilibradas entre as classes. Além disso, predições realizadas em imagens externas demonstraram boa capacidade de generalização, com predições confiantes na maioria dos casos.

Os resultados indicam que o modelo proposto é eficaz para tarefas de classificação binária e está apto para aplicações práticas.