Lista 12 - CNN

Aluna: Caroline Freitas Alvernaz

Código utilizado para o desenvolvimento da lista: CNN.ipynb - Colab

A implementação tem como objetivo treinar uma rede neural convolucional (CNN) para classificar imagens entre gatos e cachorros. Ele realiza o upload e organização dos dados, aplica pré-processamento com redimensionamento, normalização e aumento de dados, constrói uma CNN simples, treina o modelo com 70% dos dados, validada com 15% e testa com os 15% restantes, além de exibir métricas de desempenho como acurácia, precisão, recall e F1-score.

Explicação do código

```
uploaded = files.upload()
for fname in uploaded.keys():
    with zipfile.ZipFile(fname, 'r') as zip_ref:
        zip_ref.extractall('/content')
```

Esse trecho realiza o upload manual de um arquivo .zip contendo as imagens e em seguida extrai todo o conteúdo no diretório do Colab.

```
for root, dirs, files in os.walk('/content'):
    if 'cats' in dirs and 'dogs' in dirs:
        orig_dir = root
        break
```

Busca automaticamente o diretório onde estão as pastas com imagens das duas classes, evitando inserir diretamente no código valores fixos.

```
for classe in ['cats', 'dogs']:
    arquivos = sorted(os.listdir(os.path.join(orig_dir, classe)))[:5000]
    ...
```

Seleciona as 5.000 primeiras imagens de cada classe e copia para um novo diretório. Isso reduz o volume total de dados, tornando o treinamento mais rápido. Essa escolha foi feita, uma vez que o dataset inicial estava muito grande e demandando um

tempo de execução muito longo, porém os resultados obtidos ainda assim foram satisfatórios.

Após, as imagens são redistribuídas em três subpastas, de modo proporcional, garantindo integridade do aprendizado e avaliação.

```
datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,
    width_shift_range=0.2,
    ...
)
```

Essa configuração aplica normalização e transformações visuais (data augmentation) ao conjunto de treino. Para validação e teste, usa-se apenas rescale=1./255.

```
train_generator = datagen.flow_from_directory(..., subset='training')
val_generator = datagen.flow_from_directory(..., subset='validation')
test_generator = test_datagen.flow_from_directory(..., shuffle=False)
```

Permitem que as imagens sejam carregadas em tempo real em batches, reduzindo uso de memória.

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

A rede utiliza três camadas convolucionais com pooling, uma camada densa intermediária e uma camada final com saída binária (sigmoid).

```
optimizer = tf.keras.optimizers.Adam(learning_rate=1e-4)
model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])
history = model.fit(...)
```

Define a função de perda, o otimizador e executa o processo de treinamento por 10 épocas.

```
plt.plot(epochs, acc, label='Acurácia Treino')
plt.plot(epochs, val_acc, label='Acurácia Validação')
```

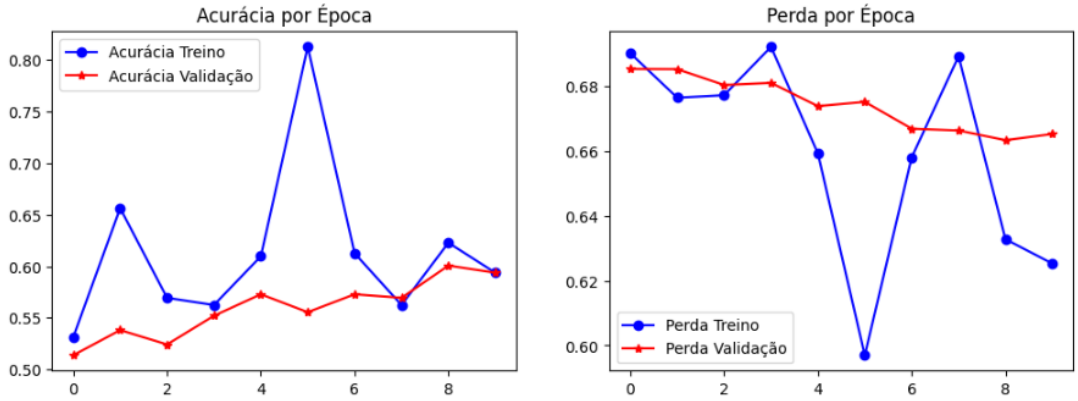
Gera gráficos com as curvas de acurácia e perda, permitindo avaliar o comportamento do modelo durante o treinamento.

```python
from sklearn.metrics import classification_report, confusion_matrix
preds = model.predict(test_generator)
pred_labels = (preds > 0.5).astype(int)
print(classification_report(test_generator.classes, pred_labels))
```

Compara os rótulos reais com as previsões e calcula precisão, revocação e F1-score, além da matriz de confusão para inspecionar o desempenho.

A rede neural convolucional utilizada neste código é composta por três camadas convolucionais, todas utilizando kernel 3x3 e função de ativação ReLU. Após cada convolução, há uma camada de pooling do tipo MaxPooling2D, totalizando três operações de redução espacial. Em seguida, a saída tridimensional é "achatada" por uma camada Flatten, que conecta os dados a uma camada densa com 128 neurônios e ativação ReLU. Por fim, a saída é processada por uma camada densa com um único neurônio e ativação sigmoide, apropriada para tarefas de classificação binária.

Saídas



Os gráficos mostram como o desempenho do modelo mudou durante as épocas de treinamento. No gráfico da esquerda, vemos que a acurácia (porcentagem de acertos) no conjunto de treino variou bastante, com um pico muito alto na sexta época. No entanto, a acurácia no conjunto de validação se manteve mais estável e sempre abaixo da de treino, o que pode indicar que o modelo começou a se ajustar demais aos dados de treino, sem melhorar sua capacidade de acertar em dados novos.

Já no gráfico da direita, que mostra a perda (erro do modelo), vemos que a perda de treino diminuiu bastante em alguns momentos, enquanto a perda de validação ficou quase constante. Isso reforça a ideia de que o modelo está aprendendo os dados de treino,

mas não está generalizando tão bem para novos exemplos. Melhorias podem ser feitas com mais dados, ajustes na rede ou técnicas para evitar o sobreajuste.

```
Relatório de Classificação:
              precision    recall  f1-score   support

        Gato       0.72      0.35      0.47       151
    Cachorro       0.57      0.86      0.69       151

    accuracy                           0.61       302
   macro avg       0.64      0.61      0.58       302
weighted avg       0.64      0.61      0.58       302

Matriz de Confusão:
[[ 53  98]
 [ 21 130]]
```

O relatório de classificação e a matriz de confusão apresentados mostram o desempenho do modelo nos dados de teste. A acurácia geral foi de 61%, o que indica que o modelo acertou aproximadamente 6 em cada 10 imagens. Para a classe "Gato", o modelo obteve uma precisão de 72%, ou seja, quando previu “gato”, acertou na maioria das vezes; porém, o recall foi baixo (35%), indicando que ele deixou de reconhecer corretamente muitos gatos.

Já para a classe "Cachorro", o recall foi alto (86%), mostrando que a maioria dos cachorros foi corretamente identificada, embora com uma precisão menor (57%), o que sugere que ele também confundiu algumas imagens com essa classe. A matriz de confusão mostra que o modelo classificou corretamente 130 cachorros e 53 gatos, mas errou 98 gatos (classificados como cachorros) e 21 cachorros (classificados como gatos). Isso indica que o modelo está tendendo a prever mais "cachorros" do que realmente deveria, mostrando um desequilíbrio.

Testes

Ainda, é possível fazer o upload de novas imagens e verificar se a imagem se trata de um gato ou de um cachorro.

```
uploaded = files.upload()
img_path = list(uploaded.keys())[0]
```

Permite ao usuário selecionar uma imagem do seu computador. A imagem é carregada no ambiente do Colab e o caminho do arquivo é armazenado na variável img_path.

```
img = image.load_img(img_path, target_size=(150, 150))
plt.imshow(img)
plt.axis('off')
plt.title('Imagem enviada')
plt.show()
```

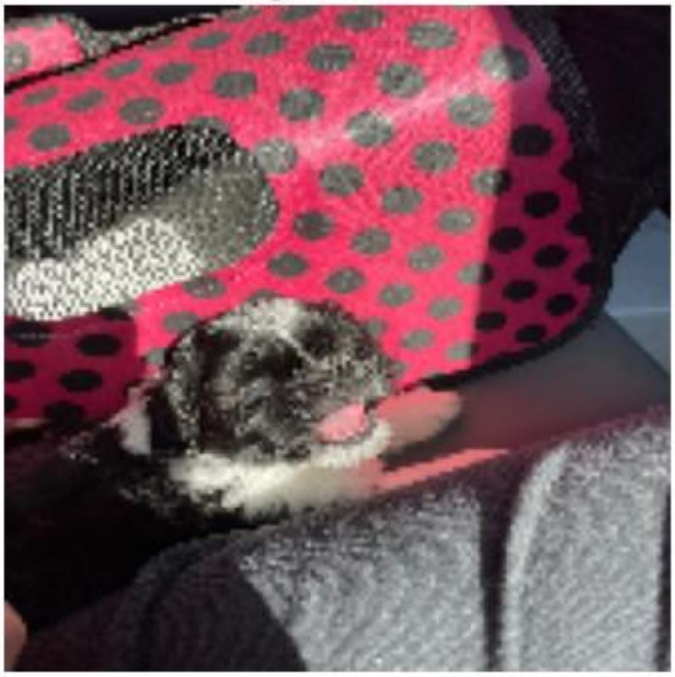
Carrega a imagem enviada, redimensionando para 150x150 pixels (tamanho exigido pela CNN), e exibe visualmente no notebook sem eixos, com um título indicando que foi a imagem submetida pelo usuário.

```
img_array = image.img_to_array(img) / 255.0
img_array = np.expand_dims(img_array, axis=0)  # (1, 150, 150, 3)
```

Transforma a imagem em um array numérico e normaliza os valores de pixel para a faixa de 0 a 1. A função expand_dims adiciona uma dimensão para representar o “lote” (batch) com apenas uma imagem, conforme exigido pela entrada do modelo. Após, passa a imagem pré-processada para o modelo treinado, que retorna uma probabilidade entre 0 e 1. Esse valor representa a confiança de que a imagem pertence à classe “Cachorro”. Define a classe prevista com base em um limiar de 0.5. Se a probabilidade for maior ou igual a 0.5, o modelo interpreta como “Cachorro”; caso contrário, como “Gato”. A variável confianca guarda a certeza do modelo em relação à classe atribuída.

A seguir, são apresentados alguns testes realizados.

Imagem enviada



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 115ms/step
Classe prevista: Cachorro
Confiança: 62.36%
```

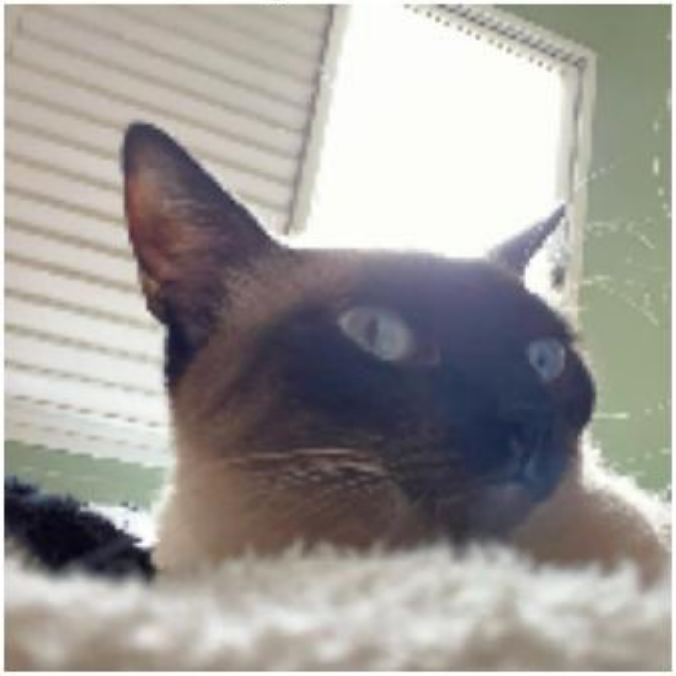
Imagem enviada

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step
Classe prevista: Cachorro
Confiança: 54.85%
```

Imagem enviada

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 52ms/step
Classe prevista: Gato
Confiança: 55.05%
```

Imagem enviada



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 70ms/step
Classe prevista: Gato
Confiança: 61.65%
```

Em conclusão, o modelo desenvolvido apresenta um desempenho inicial funcional para a tarefa de classificação de imagens entre gatos e cachorros, mas ainda

limitado em sua capacidade de generalização. A confiança das predições, frequentemente situada entre 50% e 60%, indica que a rede está incerta em grande parte das classificações, o que pode estar relacionado a um treinamento superficial. Embora o modelo consiga distinguir padrões básicos, os resultados sugerem que há espaço para aprimoramento, seja por meio de redes pré-treinadas, ajuste de hiperparâmetros ou aumento da qualidade e diversidade dos dados de entrada. A adoção dessas estratégias pode elevar significativamente a confiança e a acurácia das predições, tornando o sistema mais confiável para uso prático.