UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

PAULO RICARDO DANTAS

MODELO PARA CLASSIFICAÇÃO DE IMAGENS DE RAPOSAS, GATOS E CACHORROS

Natal/RN

2025

SUMÁRIO

1 INTRODUÇÃO 3
2 METODOLOGIA 4
2.1 PRÉ-PROCESSAMENTO 4
2.1.1 Divisão do conjunto de dados 4
2.1.2 Auto-orientação 4
2.1.3 Redimensionamento 4
2.1.4 Aumento dos dados 4
2.2 TREINAMENTO 5
3 RESULTADOS 6
4 CONCLUSÕES 9
REFERÊNCIAS 10

1 INTRODUÇÃO

Com o avanço das tecnologias, cada vez mais a informática está presente no nosso cotidiano e muitas vezes nem imaginamos o tanto. Uma área que está cada vez mais presente em nossas vidas, é a área da Inteligência Artificial que tem aplicações em mais diversas áreas, desde verificar se uma fruta está madura ou não analisando uma foto dela, até processar e criar textos.

Como foi mencionado, a Inteligência Artificial possui uma gama muito ampla de aplicações e uma delas é o Aprendizado de máquina. “Aprendizado de Máquina é uma área de IA cujo objetivo é o desenvolvimento de técnicas computacionais sobre o aprendizado bem como a construção de sistemas capazes de adquirir conhecimento de forma automática.” (MONARD; BARANAUSKAS, 2003).

A motivação para a realização desse trabalho está relacionada a necessidade de obtenção de experiências relacionadas a área de machine learning, visto que essa é uma área que possibilita muitas aplicações práticas, que podem ser aplicadas no cotidiano.

O principal objetivo desse trabalho é treinar um modelo que vai ser capaz de classificar uma imagem dentro das classes: cachorro, gato ou raposa. Também será analisado o desempenho do modelo, tendo a acurácia como principal parâmetro a ser analisado.

O conjunto de dados que foi utilizado como base para o treinamento do modelo foi o *Animal Image Dataset - Cats, Dogs, and Foxes* (KAGGLE, 2025). O dataset possui 300 imagens no total, sendo 100 imagens de cada uma das categorias.

2 METODOLOGIA

Durante esse capítulo, vai ser abordado as etapas de pré-processamento que foram realizadas no conjunto de dados original e algumas etapas de treinamento, como a arquitetura usada para treinamento do modelo.

2.1 Pré-processamento

Para facilitar as etapas de pré-processamento, foi utilizado a plataforma *Roboflow.* Nessa plataforma, existem muitas etapas de pré-processamento que já estão implementadas, então para economizar tempo, foi adotado essa estratégia.

2.1.1 Divisão do conjunto de dados

O conjunto de dados foi dividido da seguinte forma: 70% para treinamento, 20% para validação e 10% para teste. Essa divisão é importante, para fazer com que o modelo mostre uma acurácia média mais semelhante a real e não aprenda apenas as imagens usadas no treinamento.

2.1.2 Auto-orientação

Nessa etapa, o algoritmo do *roboflow* identifica as orientações das imagens e ajusta-as caso estejam incorretas, como por exemplo de cabeça para baixo.

2.1.3 Redimensionamento

Para fins de redução de tamanho de imagens e padronização nos testes, todas as imagens foram redimensionadas para um tamanho de 512 x 512 pixels.

2.1.4 Aumento dos dados

Nessa etapa, foram aplicados filtros de saturação entre -25% e 25%, rotação de -15 e 15 graus e uma multiplicação de 3x nas imagens. Desse modo, é inserido um certo ruído nas imagens com o intuito de deixar o modelo mais generalista.

Por fim, o conjunto de dados ficou com 739 imagens totais, sendo 648 para treinamento, 61 para validação e 30 para teste.

2.2 Treinamento

A IDE usada para treinar o modelo foi a do Google Colab, rodando o Python 3 na GPU T4. Para realizar o treinamento iremos utilizar a ferramenta *YOLO11* que é uma ferramenta da biblioteca *ultralytics*. "YOLO11, a versão mais recente do aclamado modelo de deteção de objetos e segmentação de imagens em tempo real" (ULTRALYTICS, 2025). A versão específica que iremos utilizar para o treinamento do nosso modelo é a versão *yolo11x-cls.pt* que é a melhor versão para tarefa de classificação de imagens da versão 11 do *YOLO*.

Os parâmetros que foram especificados para o treinamento do modelo foram: épocas, *batch size*, e *image size.* A quantidade de épocas foi fixada em 15, que seria a quantidade de iterações de treinamento do modelo. O *batch size* utilizado foi 4, que é a quantidade de imagens que vão ser processadas por vez. Por fim, o *image size* foi definido como 512, durante o pré-processamento, todas as imagens já foram redimensionadas para esse tamanho, porém o parâmetro foi deixado por precaução. O tempo gasto para terminar o treinamento foi de aproximadamente 8 minutos.

Figura 1 – Trecho do código do treinamento.

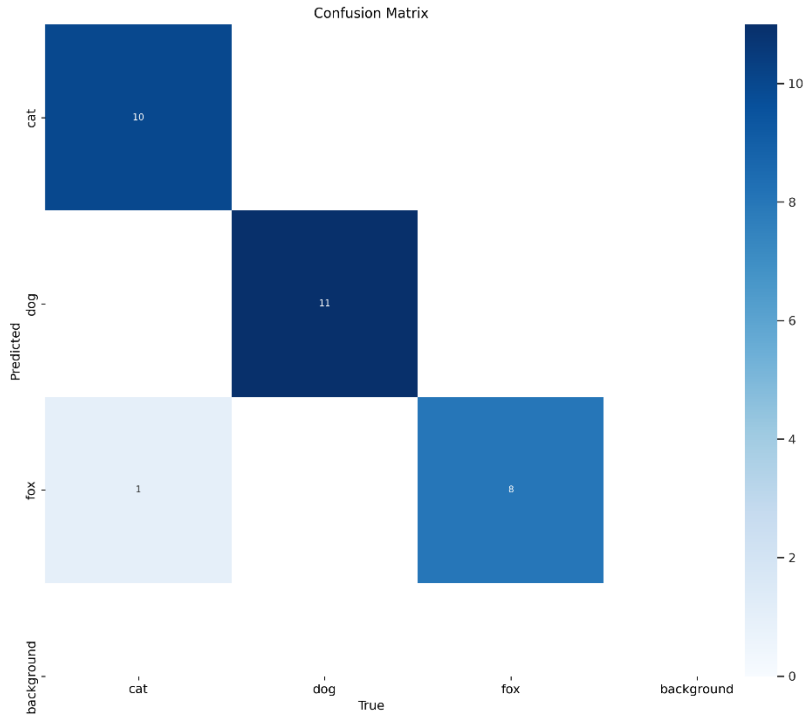
```
# Carregando o modelo
model = YOLO('yolo11x-cls.pt')

# Treinando o modelo
model.train(data='/content/dataset', epochs=15, batch=4, imgsz=512)
```

Fonte: Autor

3 RESULTADOS

Aqui nesta seção iremos abordar as métricas dadas pela própria interface de treinamento do *YOLO*. A acurácia do modelo após treiná-lo com 15 épocas foi de **96,7%.** Essa foi uma excelente acurácia, mas vale lembrar que no contexto de classificação de imagens, a acurácia não diz tudo sobre o modelo, então logo a seguir serão analisados algumas outras métricas.

Figura 2 – Matriz de confusão do modelo treinado.

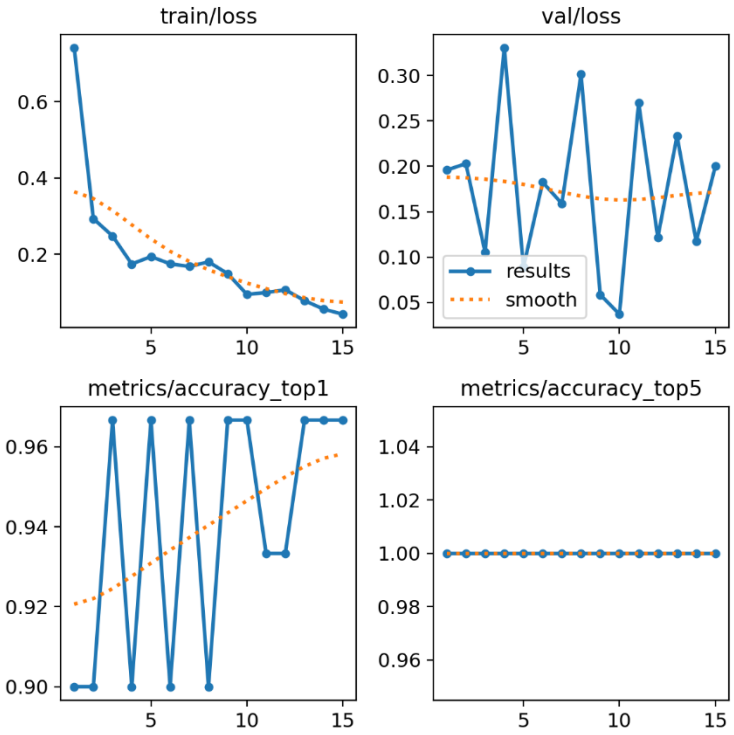


Fonte: Autor

A figura 2 mostra a matriz de confusão do conjunto de teste do modelo. Para gerar essa matriz, o modelo faz previsões para as imagens do conjunto de teste, que são imagens que não foram apresentadas para o modelo durante o treinamento. Como o modelo "nunca viu" essas imagens, é uma boa maneira de testar como o modelo se comporta no quesito de generalização.

Na matriz de confusão apresentada foi possível observar que o modelo errou apenas em uma das 30 imagens de teste, na qual ele previu que a imagem fosse de uma raposa, mas era de um gato.

Figura 3 – Métricas geradas pela interface do YOLO.



Fonte: Autor

Já na figura 3 é possível observar algumas outras métrica que são geradas pela própria interface que foi utilizada para o treinamento. Em todos os gráficos é mostrado como a métrica em questão se comportou conforme as épocas foram passando.

No primeiro gráfico, vemos como o *loss* do treinamento foi diminuindo conforme as épocas foram passando, esse é um bom indicador e mostra que o modelo está aprendendo conforme o modelo é iterado. Já no segundo gráfico foi o que levantou uma certa preocupação, pois mostra como o *loss* se comportou durante a validação e nele houve uma variação em praticamente todas as épocas. Essa variação pode indicar problemas como *overfitting* ou instabilidade do modelo.

Ainda para fins de análises de resultados do modelo, foram baixadas da internet 6 imagens, 2 de cada categoria, para analisar como o modelo se comportaria com imagens fora do padrão do conjunto de treinamento. O resultado desse teste poderá ser visto na figura 4.

Figura 4 – Teste do modelo com imagens baixadas da internet.

```
image 1/1 /content/test_images/cat_test1.jpg: 512x512 cat 1.00, fox 0.00, dog 0.00, 36.4ms
Speed: 24.8ms preprocess, 36.4ms inference, 0.1ms postprocess per image at shape (1, 3, 512, 512)
Resultados para a imagem test_images/cat_test1.jpg:
Classe: cat, Confiança: 1.00
--------------------------------------------------

image 1/1 /content/test_images/cat_test2.jpg: 512x512 cat 1.00, dog 0.00, fox 0.00, 35.0ms
Speed: 18.4ms preprocess, 35.0ms inference, 0.2ms postprocess per image at shape (1, 3, 512, 512)
Resultados para a imagem test_images/cat_test2.jpg:
Classe: cat, Confiança: 1.00
--------------------------------------------------

image 1/1 /content/test_images/fox_test1.jpg: 512x512 fox 1.00, dog 0.00, cat 0.00, 34.2ms
Speed: 34.0ms preprocess, 34.2ms inference, 0.1ms postprocess per image at shape (1, 3, 512, 512)
Resultados para a imagem test_images/fox_test1.jpg:
Classe: fox, Confiança: 1.00
--------------------------------------------------

image 1/1 /content/test_images/fox_test2.jpg: 512x512 fox 0.59, cat 0.35, dog 0.06, 34.1ms
Speed: 17.5ms preprocess, 34.1ms inference, 0.1ms postprocess per image at shape (1, 3, 512, 512)
Resultados para a imagem test_images/fox_test2.jpg:
Classe: fox, Confiança: 0.59
--------------------------------------------------

image 1/1 /content/test_images/dog_test1.jpg: 512x512 dog 1.00, fox 0.00, cat 0.00, 29.4ms
Speed: 24.2ms preprocess, 29.4ms inference, 0.1ms postprocess per image at shape (1, 3, 512, 512)
Resultados para a imagem test_images/dog_test1.jpg:
Classe: dog, Confiança: 1.00
--------------------------------------------------

image 1/1 /content/test_images/dog_test2.jpg: 512x512 dog 0.99, cat 0.01, fox 0.00, 29.4ms
Speed: 27.4ms preprocess, 29.4ms inference, 0.1ms postprocess per image at shape (1, 3, 512, 512)
Resultados para a imagem test_images/dog_test2.jpg:
Classe: dog, Confiança: 0.99
```

Fonte: Autor

O modelo conseguiu acertar as classes de todas as 6 imagens baixadas da internet, ou seja, mostrou uma acurácia de 100% para esse teste.

4 CONCLUSÕES

Com base nos resultados obtidos, é possível concluir que o modelo de classificação apresentou um desempenho satisfatório no conjunto de treinamento, evidenciado pela redução contínua no *loss* do treinamento. Porém, como o *loss* ainda estava caindo conforme as épocas passavam, uma possível melhoria seria um aumento na quantidade de épocas de treinamento.

Além disso durante os testes extras que foram realizados na parte dos resultados, o modelo classificou a imagem *fox_test2.jpg* como raposa, o que é a classe certa, mas com o grau de confiança bem abaixo da média dos outros testes. Essa imagem é de uma raposa que é encontrada na Caatinga brasileira, isso indica que as imagens que foram usadas para treinar possam ter um padrão específico.

Outra sugestão para melhorar o modelo seria aumentar cada vez o conjunto de treinamento, a fim de cada vez mais o modelo se tornar o

REFERÊNCIAS

Google Colab. Colab, http://colab.google/. Acessado 13 de janeiro de 2025.

KAGGLE. *Animal Image Dataset - Cats, Dogs, and Foxes*. https://www.kaggle.com/datasets/snmahsa/animal-image-dataset-cats-dogs-and-foxes. Acessado 13 de janeiro de 2025.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. In: **ENGEL, P. M. B.** (Org.). Sistemas inteligentes: fundamentos e aplicações. 1. ed. Barueri-SP: Manole, 2003. p. 89–114.

Projeto_U2_DL.ipynb.https://colab.research.google.com/drive/12nsWz0JxdNYWWfdOFAPYnrcImNSYiqV7. Acessado 13 de janeiro de 2025.

Python. Python.Org, 19 de dezembro de 2024, https://www.python.org/.

Roboflow: Computer Vision Tools for Developers and Enterprises. https://roboflow.com/. Acessado 13 de janeiro de 2025.

ULTRALYTICS. *YOLO*. https://docs.ultralytics.com/pt/. Acessado 13 de janeiro de 2025.