

APRENDIZADO DE MÁQUINA (2024)

Docente:

- Ricardo Cerri

Alunos:

- João Gabriel Sasseron Roberto Amorim (12542564)
- Bernardo Maia Coelho (12542481)

INTRODUÇÃO

Contexto: Nas áreas verdes da UFSCar, câmeras equipadas com sensores de movimento registram vídeos dos animais presentes.





2016/08/08
11:11:12

INTRODUÇÃO

Problema: A triagem e classificação são feitas manualmente, uma tarefa repetitiva e demorada.

Solução: Aplicação de algoritmos de Inteligência Artificial (IA) para classificar imagens, tornando o processo de triagem mais rápido e eficiente.

ABORDAGEM

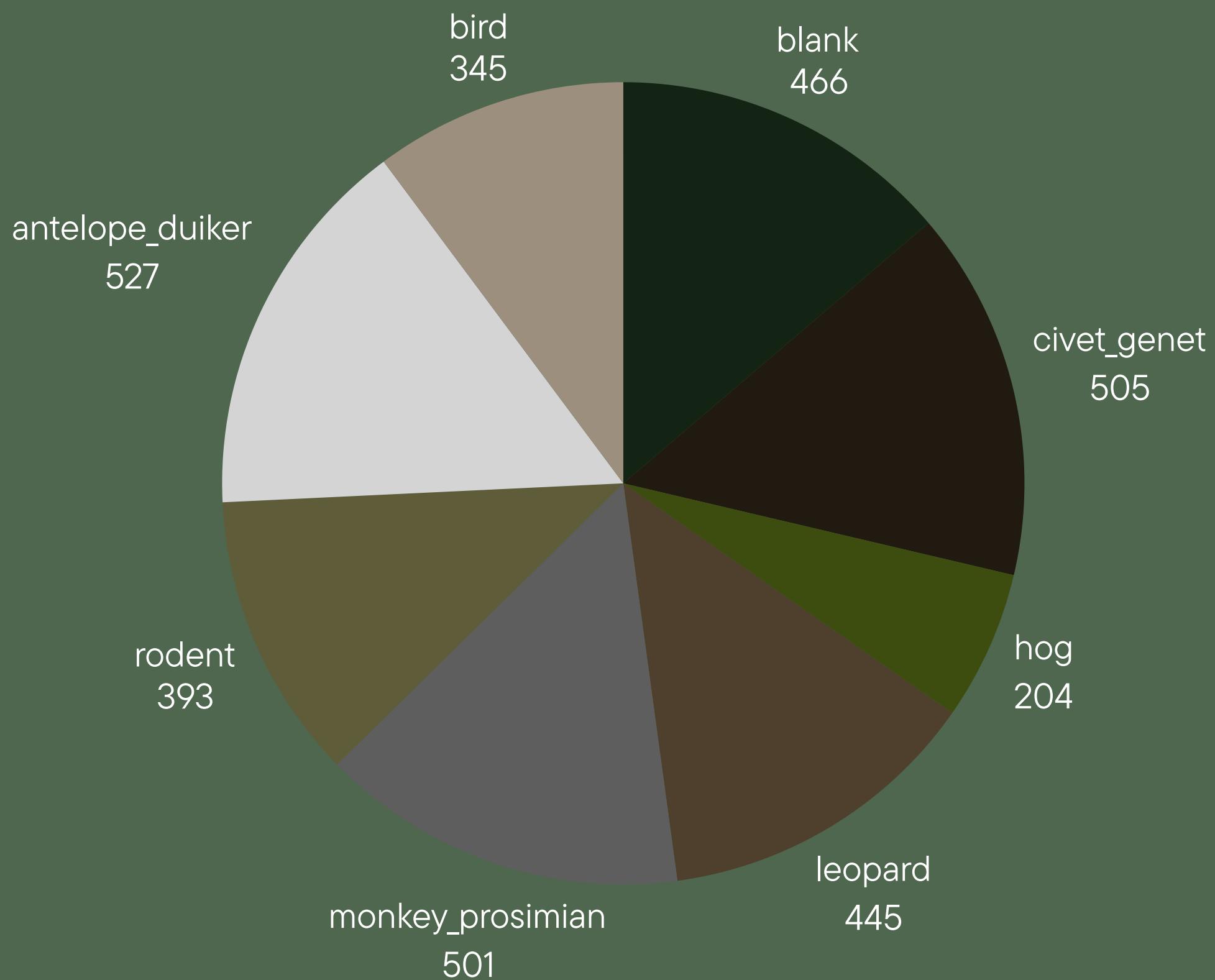
Devido à **baixa quantidade de dados** obtidos pelas câmeras de segurança, optamos por utilizar um **dataset do Kaggle** para estudar o tema e identificar o melhor algoritmo para nossa tarefa.

Desta maneira assim que possuímos um conjunto de dados suficiente grande já teremos uma solução para nossa tarefa.

ANÁLISE EXPLORATÓRIA

O número de classes é **balanceado**, exceto pela classe "hog", que tem poucos exemplos.

Porém, **mantivemos essa proporção** para garantir a representatividade e robustez do modelo, evitando distorções.



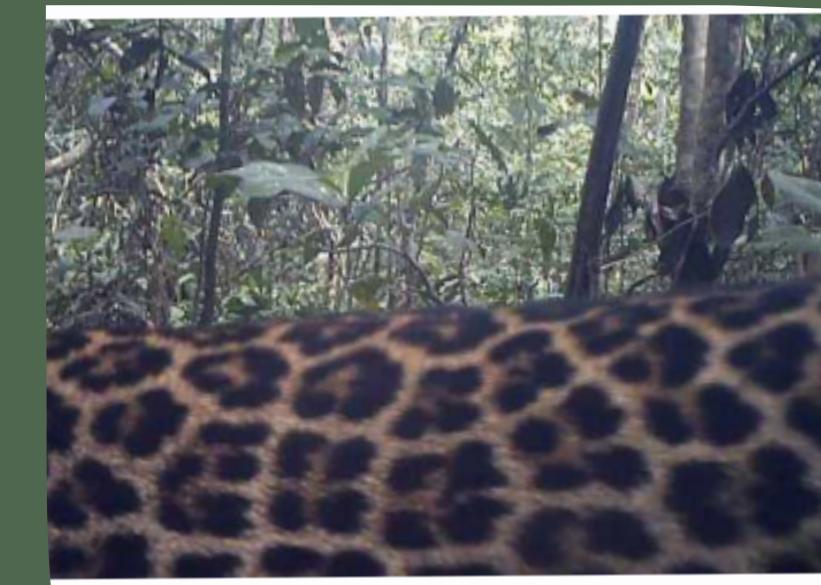
ANÁLISE EXPLORATÓRIA



civet_genet



hog



leopard



monkey_prosimian



blank



rodent



antelope_duiker



bird

PRÉ-PROCESSAMENTO

A etapa de pré-processamento se constituiu de **redimensionar** e **normalizar** os dados.

Para os algoritmos tradicionais, como Naïve Bayes e Árvore de Decisão, as imagens foram **redimensionadas para 32x32**, convertidas em **vetores unidimensionais** através do método 'flatten' e **normalizadas usando MinMaxScaler**.

Para as redes neurais, criamos um dataset usando PyTorch com **redimensionalização adequada para cada rede**, além da **normalização** utilizando $\text{mean}=(0.485, 0.456, 0.406)$ e $\text{std}=(0.229, 0.224, 0.225)$.

MODELOS

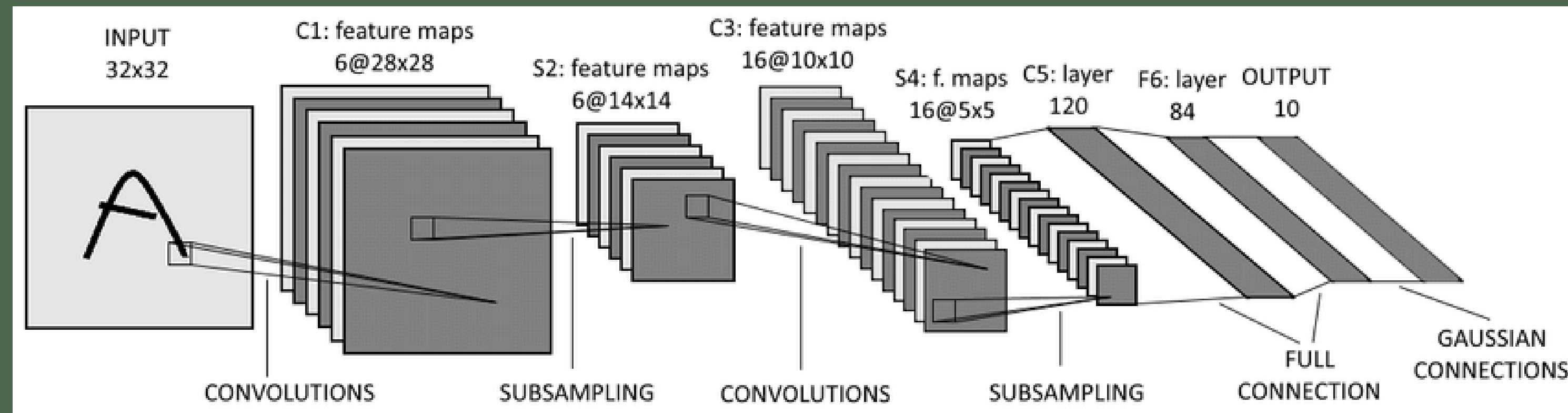
Modelos Clássicos Selecionados para a Classificação:

- **Naive Bayes**
- **Árvore de Decisão**
- **KNN**
- **Random Forest**

Observação: SVM e Ridge Classifier não foram implementados devido a limitações de hardware.

MODELOS

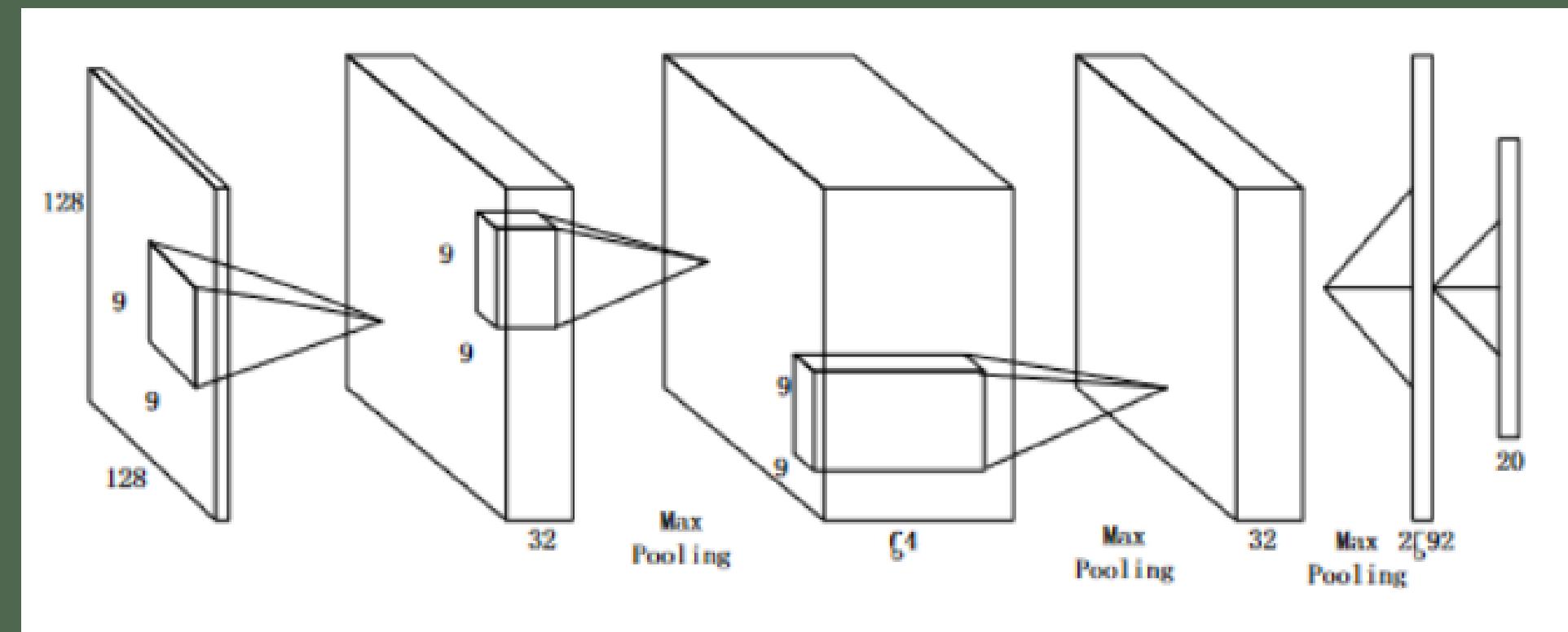
LeNet5



A LeNet-5, desenvolvida por Yann LeCun em 1998, é um marco no reconhecimento de dígitos em imagens. Com uma arquitetura simples de 3 camadas convolucionais, 2 camadas de pooling e 2 camadas totalmente conectadas

MODELOS

CNN-Artigo IEEE



O estudo descreve uma CNN composta por 3 camadas convolucionais e 3 camadas de max pooling, seguidas por uma camada totalmente conectada. A rede é projetada para operar em imagens de 128x128 pixels, utilizando convoluções e pooling para extrair características importantes antes da classificação final.

MODELOS

CNN-Custom

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 224, 224]	896
BatchNorm2d-2	[-1, 32, 224, 224]	64
MaxPool2d-3	[-1, 32, 56, 56]	0
Conv2d-4	[-1, 64, 56, 56]	18,496
BatchNorm2d-5	[-1, 64, 56, 56]	128
MaxPool2d-6	[-1, 64, 18, 18]	0
Dropout-7	[-1, 64, 18, 18]	0
Flatten-8	[-1, 20736]	0
Linear-9	[-1, 8]	165,896

Total params: 185,480
Trainable params: 185,480
Non-trainable params: 0

TREINAMENTO

Clássicos

Para treinar os modelos clássicos da arquitetura, utilizamos o método **GridSearchCV** junto com **K-Fold**, com a divisão do conjunto de treinamento em 5 partes para validação cruzada

Após determinar os **melhores parâmetros para cada arquitetura**, treinamos o modelo utilizando o conjunto completo de dados de treinamento com esses parâmetros otimizados.

O melhor desempenho foi obtido pela **Random Forest**, com os parâmetros:

- **criterion: entropy**,
- **max_depth: 20**,
- **min_samples_leaf: 1**,
- **min_samples_split: 2**,
- **n_estimators: 200**

TREINAMENTO

Redes Neurais

As redes neurais foram treinadas usando o **dataset inteiro para maximizar uso dos dados.**

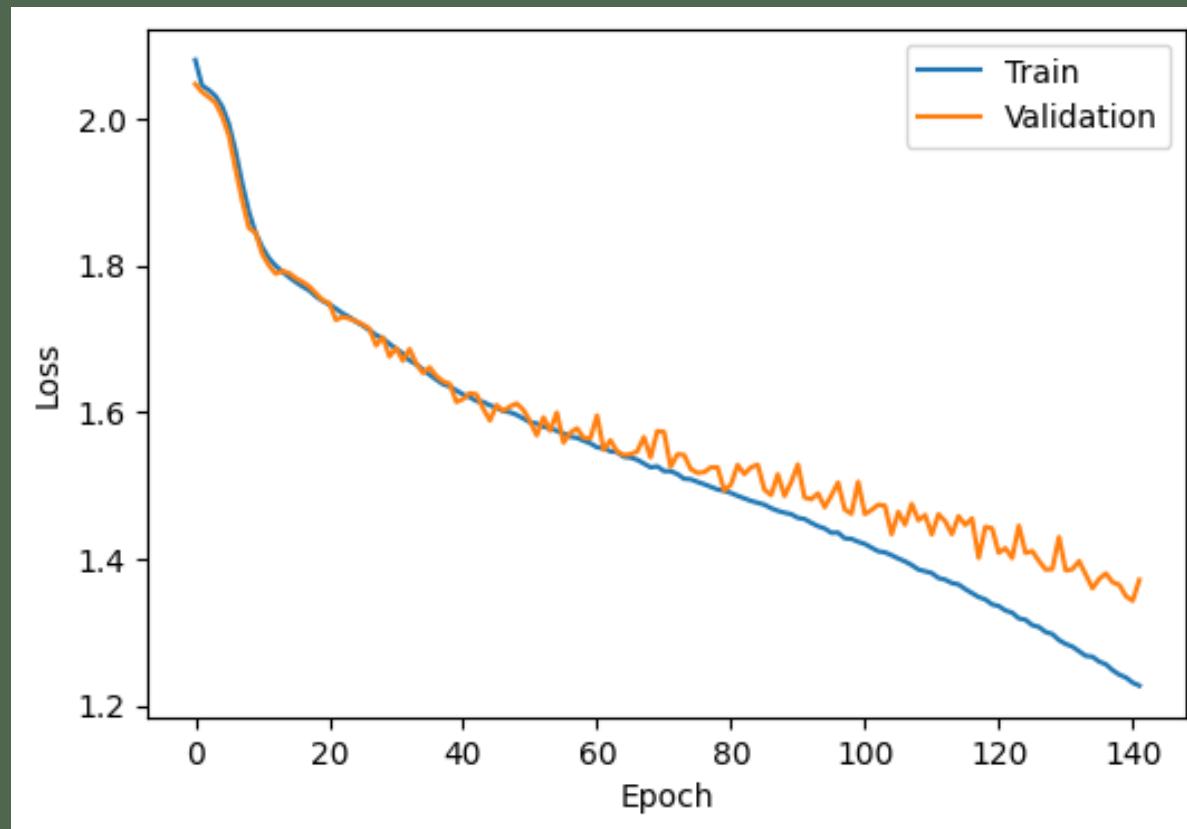
Além disso, foi implementado o **early stopping** com tolerância de 10 épocas e minidelta de 0.1 **para evitar overfitting** em modelos sem dropout como na LeNet5 e o modelo do artigo.

Loss: Cross Entropy Loss

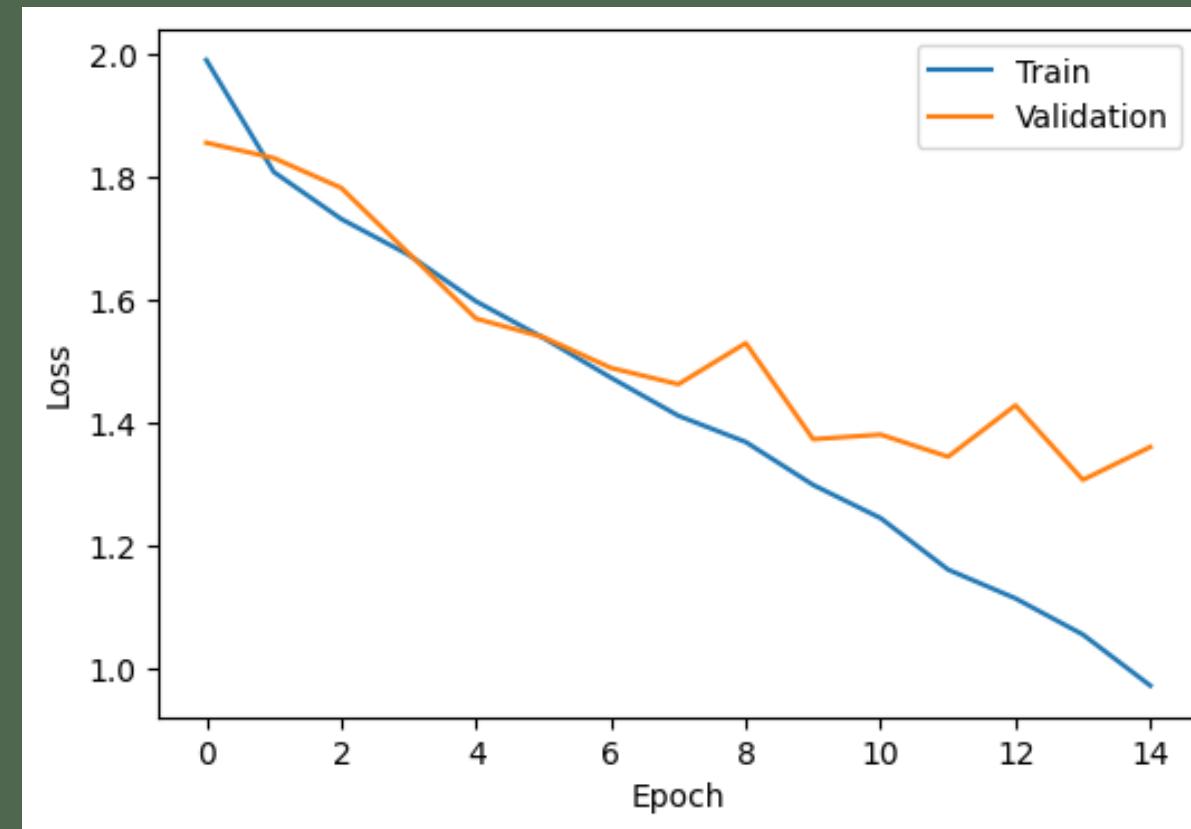
Learning ration: 0.01

TREINAMENTO

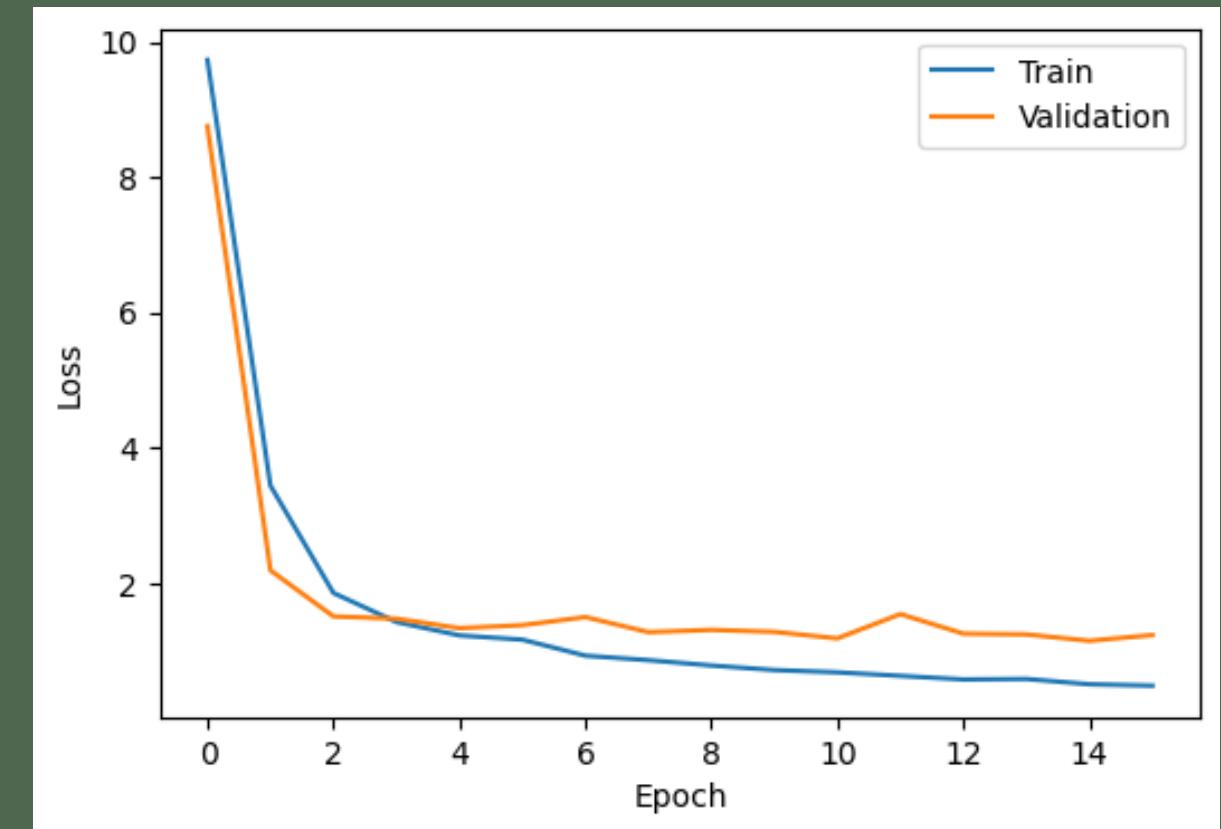
Redes Neurais



LeNet5



CNN-Artigo



CNN-Custom

AVALIAÇÃO

Para a avaliação iremos considerar as seguintes **métricas**:

- **Precision** (Macro e Micro),
- **Recall** (Macro e Micro),
- **F1-Score** (Macro e Mircro),
- **Acurácia**.

Além disso, optamos pela **F1-Score micro** como métrica principal devido à sua capacidade de fornecer uma medida global do desempenho do modelo em todas as classes, sem favorecer nenhuma em particular.

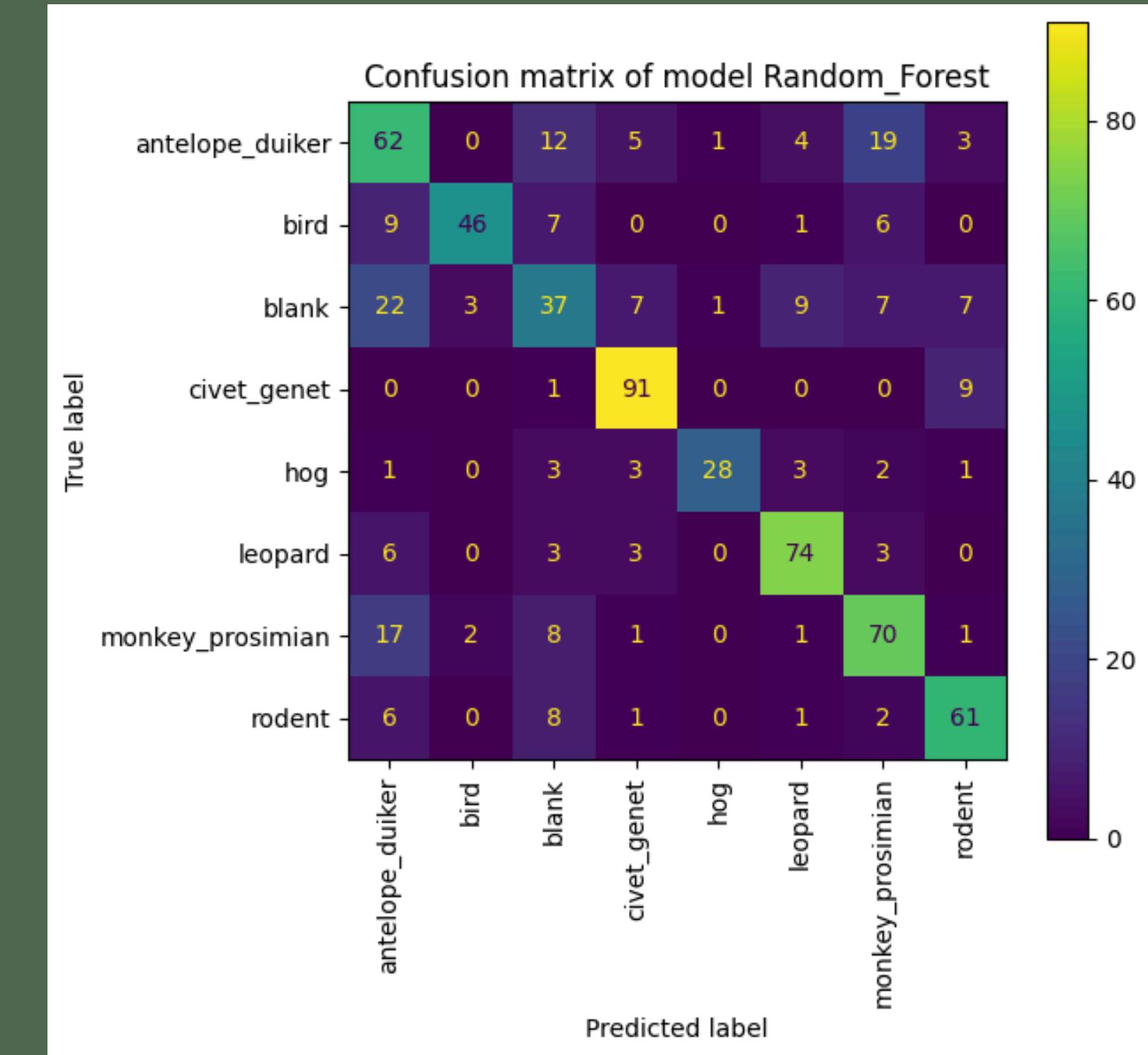
AVALIAÇÃO

MODELO	PRECISION MACRO	PRECISION MICRO	RECALL MACRO	RECALL MICRO	F1 MACRO	F1 MICRO	ACCURACY
LeNet5	0.515685	0.497050	0.473371	0.497050	0.467615	0.497050	0.497050
CNN-Custom	0.703963	0.675516	0.681868	0.675516	0.690685	0.675516	0.675516
CNN-Artigo	0.612119	0.525074	0.525095	0.525074	0.547754	0.525074	0.525074
KNN	0.686739	0.675516	0.690379	0.675516	0.687437	0.675516	0.675516
Decison Tree	0.534642	0.514749	0.516108	0.514749	0.521641	0.514749	0.514749
Random Forest	0.725340	0.697640	0.705136	0.697640	0.711559	0.697640	0.697640
Naive Bayes	0.237157	0.272861	0.257760	0.272861	0.228900	0.272861	0.272861

AVALIAÇÃO

Ao analisar as matrizes de confusão, observa-se que, apesar da classe “**hog**” ter um número menor de exemplos, seus resultados são consistentemente bons em todos os modelos avaliados, indicando que sua baixa incidência não prejudica significativamente os resultados.

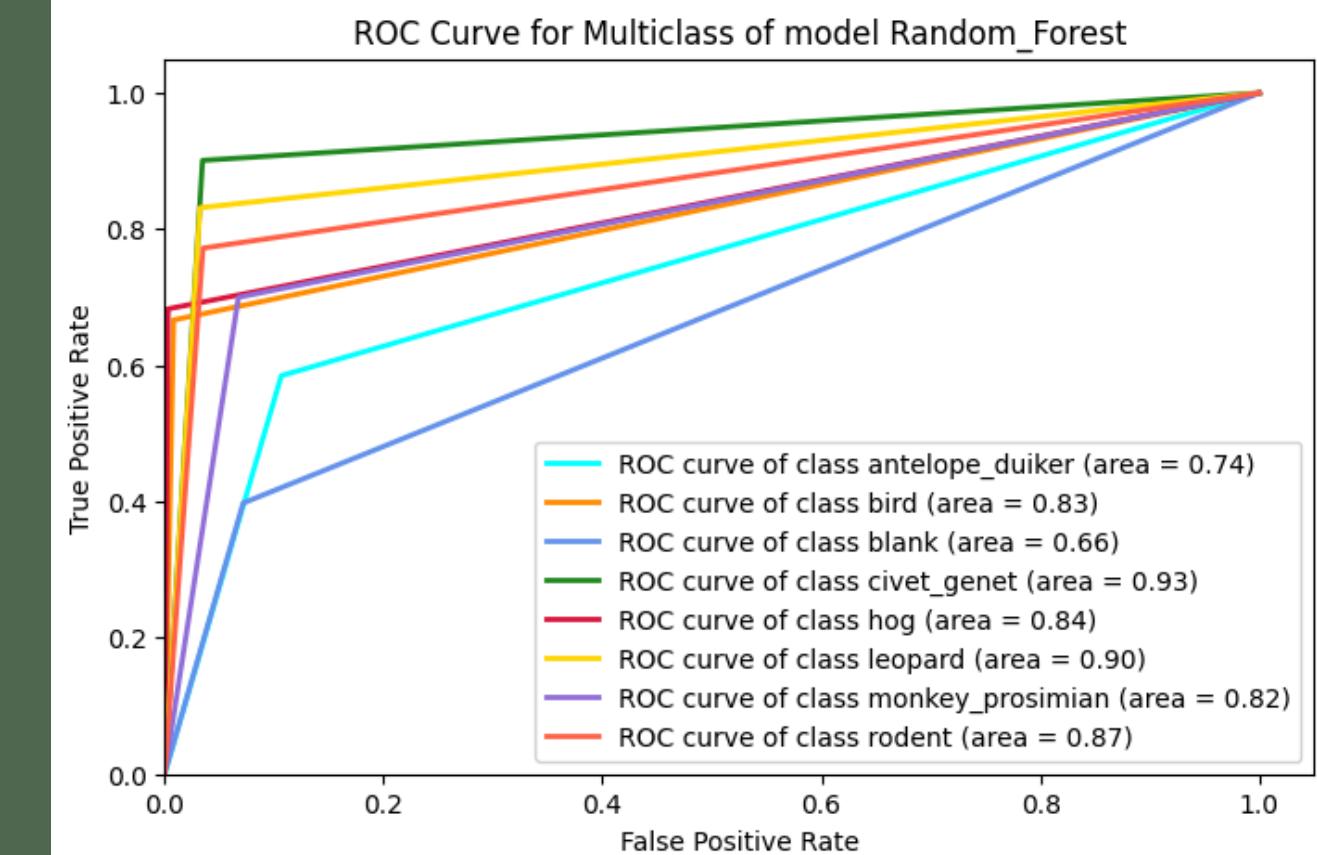
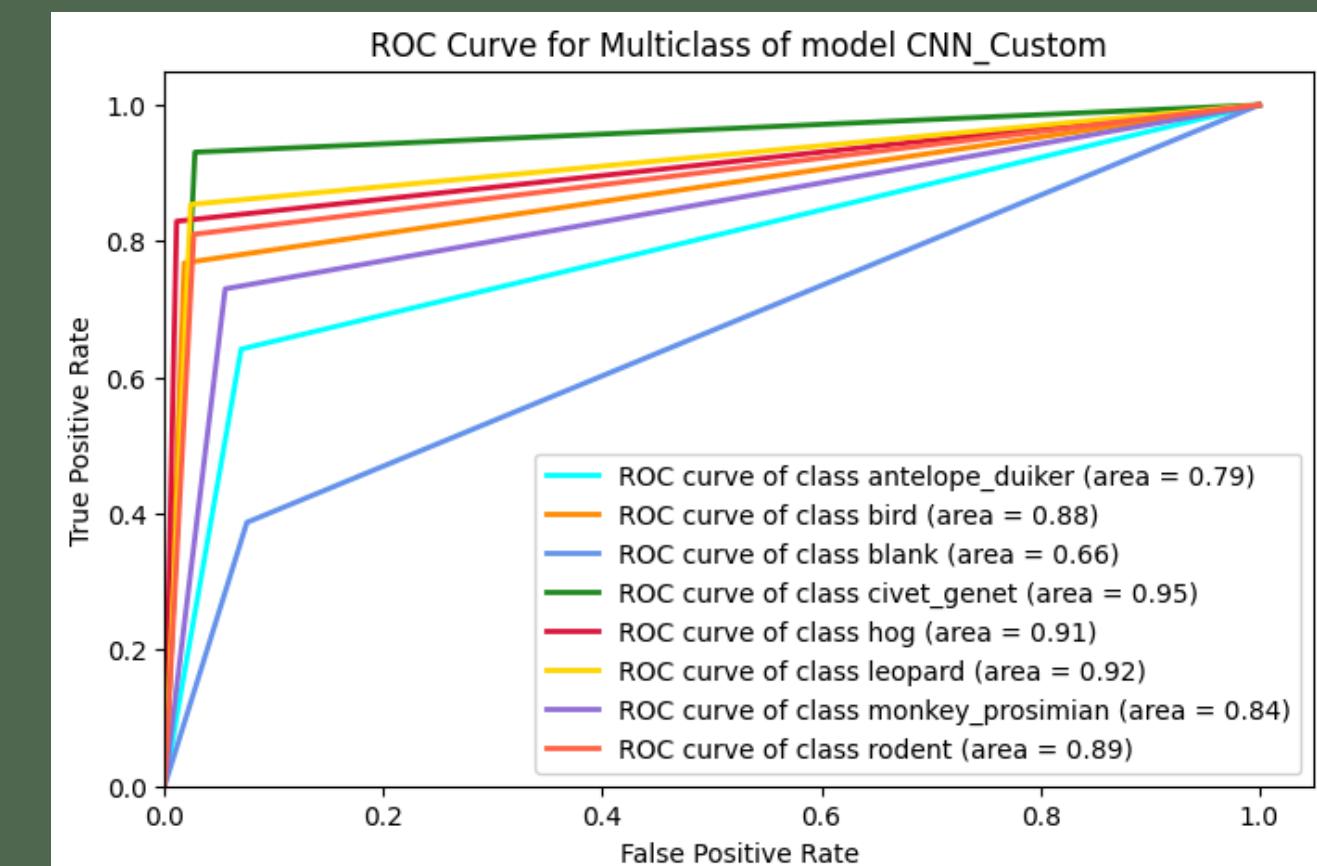
Em contraste, a classe “**blank**” apresenta os piores desempenhos, possivelmente devido à sua distribuição mais equilibrada entre as outras classes, o que pode dificultar a distinção precisa pelos modelos.



AVALIAÇÃO

Melhores desempenho por classe:

1. **antelope_duiker**: Random_Forest e CNN_Custom (0.74)
2. **bird**: KNN (0.85)
3. **blank**: CNN_Custom (0.67)
4. **civet_genet**: Random_Forest e CNN_Custom (0.93)
5. **hog**: KNN (0.89)
6. **leopard**: Random_Forest e CNN_Custom (0.90)
7. **monkey_prosimian**: Random_Forest (0.82)
8. **rodent**: KNN (0.88)



CONCLUSÃO

Podemos concluir que a **classificação de animais a partir de um conjunto de imagens é viável**, e isso nos incentiva a prosseguir com nosso projeto.

O modelo **Random Forest se destacou**, alcançando uma F1-Score Macro de 0.697640.

No entanto, como próximos passos, seria benéfico **expandir nosso conjunto de dados** e testar novos modelos para garantir a robustez do modelo.

FIM!