

CM203 - Exame

1st Bruno Benjamin Bertucci
Engenharia Eletrônica
ITA

São José dos Campos, Brasil
bruno.bertucci@ga.ita.br

2nd Gabriel Barbosa Martinz
Engenharia de Computação
ITA

São José dos Campos, Brasil
gabriel.martinz@ga.ita.br

3rd Vinicius Soledade Matos Amorim
Engenharia Eletrônica
ITA

São José dos Campos, Brasil
vinicius.amorim@ga.ita.br

I. INTRODUÇÃO

Em visão computacional, classificação de imagens é uma das tarefas mais comuns, com uma literatura extensa de métodos para resolver o problema. Um dos métodos é o uso de Redes Neurais Convolucionais (CNN, na sigla em inglês) para extrair *features* relacionadas à classificação e finalmente conseguir classificar a imagem. Neste trabalho, utilizaremos a arquitetura da rede neural Inception v3 para encontrar a raça de gatos domésticos. Foram selecionadas 10 raças para fazer o treinamento da rede neural.

II. FUNDAMENTAÇÃO TEÓRICA

A. Redes Neurais Convolucionais

CNNs são redes neurais que utilizam, em alguma camada, a operação de convolução ao invés da multiplicação de matrizes. A convolução é uma operação que aplica um filtro (chamado *kernel*) em uma imagem e retorna um valor de ativação para cada pixel da imagem. A matriz de ativações resultante é chamada de *feature map*, que mostra onde estão as *features* da imagem e quão relevante ela é. CNNs podem aprender vários filtros em paralelo, podendo caracterizar várias *features* numa imagem.

Para a classificação de imagens, costumam-se utilizar imagens anotadas com a classe que ela pertence. Precisa-se então traduzir a filtragem das camadas convolucionais para as classes existentes. Para isso utilizam-se camadas totalmente conectadas para levar o *feature map* até uma saída com o número total de classes e a probabilidade de ser cada uma delas. Seleciona-se então a classe de maior probabilidade para ser a classe prevista.

III. METODOLOGIA

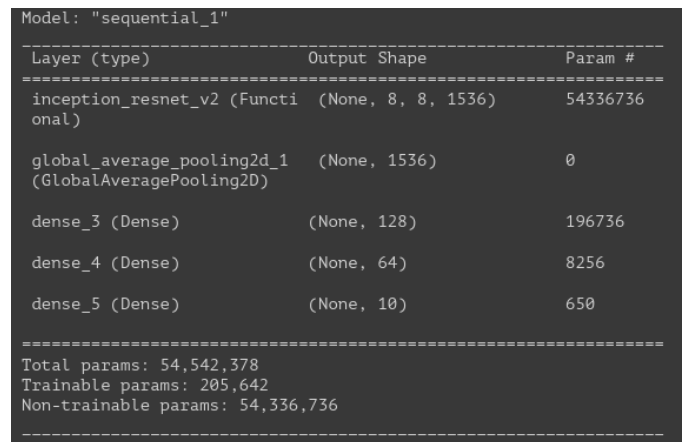
Foi utilizada a rede neural pré-treinada no banco de dados ImageNet Inception v3 [1]. A utilização de uma rede pré-treinada foi escolhida devido ao fato de redes convolucionais comumente terem muitos parâmetros para treinar, como é o caso dessa rede, que possui mais de 54 milhões de parâmetros. Foi então adicionado à saída da rede neural mais uma camada de *average pooling* e 3 camadas totalmente conectadas, duas com ativação ReLU e a última com ativação softmax, para poder colapsar a saída dela para 10 classes, que são as raças de gatos que selecionamos. Para a função de *loss* foi utilizada a entropia cruzada categórica, e a métrica utilizada foi a

acurácia. As 10 raças selecionadas para treinamento foram as 10 mais comuns no banco de dados.

O banco de dados de imagens utilizado foi retirado do Kaggle [2]. Na página do banco de dados, há o *notebook* em [3], que foi utilizado como comparativo.

IV. IMPLEMENTAÇÃO

O projeto foi implementado em um Jupyter Notebook no Google Colab, utilizando a linguagem Python com o *framework* TensorFlow. Foi necessário um esforço de balanceamento de classes para treinar o modelo final. Como a rede convolucional é pré-treinada, os parâmetros treináveis foram somente os das camadas totalmente conectadas. Mesmo assim, o número de parâmetros a serem treinados passou dos 200 mil, como mostrado na figura 1. O treinamento foi feito com 15 épocas e 100 passos por época.



Layer (type)	Output Shape	Param #
inception_resnet_v2 (Function)	(None, 8, 8, 1536)	54336736
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 1536)	0
dense_3 (Dense)	(None, 128)	196736
dense_4 (Dense)	(None, 64)	8256
dense_5 (Dense)	(None, 10)	650
Total params: 54,542,378		
Trainable params: 205,642		
Non-trainable params: 54,336,736		

Fig. 1. Sumário do modelo criado.

V. RESULTADOS E DISCUSSÕES

O modelo proposto obteve acurácia de 62.44% para os dados de treinamento e 61.79% para os dados de validação na última rodada do treinamento, como mostrado na figura 2.

Uma amostra das imagens e suas previsões está mostrada na figura 3.

Pode-se ver que houve uma acurácia significativa para classificar gatos em raças. Caminhos de melhoria existem ao poder ser possível treinar todos os parâmetros da rede.

ANEXO 1 - REPOSITÓRIO

O projeto está [neste repositório](#), junto do arquivo .h5 contendo o modelo treinado. O README contém instruções para a execução do projeto.

```
model.fit(x_train, validation_data=x_test, epochs=15, steps_per_epoch=100)
Epoch 1/15
100/100 [=====] - 203s 2s/step - loss: 1.5934 - accuracy: 0.4134 - val_loss: 1.4339 - val_accuracy: 0.4849
Epoch 2/15
100/100 [=====] - 188s 2s/step - loss: 1.2977 - accuracy: 0.5241 - val_loss: 1.2569 - val_accuracy: 0.5521
Epoch 3/15
100/100 [=====] - 189s 2s/step - loss: 1.2493 - accuracy: 0.5516 - val_loss: 1.2105 - val_accuracy: 0.5650
Epoch 4/15
100/100 [=====] - 190s 2s/step - loss: 1.2872 - accuracy: 0.5763 - val_loss: 1.2569 - val_accuracy: 0.5648
Epoch 5/15
100/100 [=====] - 186s 2s/step - loss: 1.1734 - accuracy: 0.5847 - val_loss: 1.2007 - val_accuracy: 0.5563
Epoch 6/15
100/100 [=====] - 189s 2s/step - loss: 1.1879 - accuracy: 0.5743 - val_loss: 1.1616 - val_accuracy: 0.5931
Epoch 7/15
100/100 [=====] - 186s 2s/step - loss: 1.1307 - accuracy: 0.5963 - val_loss: 1.1379 - val_accuracy: 0.5971
Epoch 8/15
100/100 [=====] - 185s 2s/step - loss: 1.0979 - accuracy: 0.6147 - val_loss: 1.1342 - val_accuracy: 0.5933
Epoch 9/15
100/100 [=====] - 188s 2s/step - loss: 1.1280 - accuracy: 0.5928 - val_loss: 1.1315 - val_accuracy: 0.5878
Epoch 10/15
100/100 [=====] - 186s 2s/step - loss: 1.1279 - accuracy: 0.6047 - val_loss: 1.1967 - val_accuracy: 0.5773
Epoch 11/15
100/100 [=====] - 185s 2s/step - loss: 1.1449 - accuracy: 0.5983 - val_loss: 1.0988 - val_accuracy: 0.6050
Epoch 12/15
100/100 [=====] - 187s 2s/step - loss: 1.1016 - accuracy: 0.6094 - val_loss: 1.1356 - val_accuracy: 0.6025
Epoch 13/15
100/100 [=====] - 185s 2s/step - loss: 1.1014 - accuracy: 0.6037 - val_loss: 1.1515 - val_accuracy: 0.5787
Epoch 14/15
100/100 [=====] - 189s 2s/step - loss: 1.0691 - accuracy: 0.6187 - val_loss: 1.1293 - val_accuracy: 0.5983
Epoch 15/15
100/100 [=====] - 185s 2s/step - loss: 1.0621 - accuracy: 0.6244 - val_loss: 1.1010 - val_accuracy: 0.6179
<keras.callbacks.History at 0x7fab328dac0>
```

Fig. 2. Épocas de treinamento do modelo



Fig. 3. Amostra de imagens com suas previsões e probabilidade de previsão.

Ao comparar com o *notebook* em [3], encontramos uma acurácia menor. Isso se deve às seguintes razões:

- foi feita uma seleção baseada na quantidade de observações de cada raça;
- o modelo utilizado foi diferente;
- não foi utilizada a mesma quantidade de passos de treinamento;
- possíveis erros na marcação das imagens.

VI. CONCLUSÃO

A estrutura de rede neural proposta conseguiu métricas relevantes para a classificação das raças dos gatos. Dada a quantidade de parâmetros da Inception v3, utilizar a rede pré-treinada foi o caminho menos custoso, mas caminhos de melhoria existem ao se treinar a rede inteira com todos os parâmetros.

REFERENCES

- [1] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," 2015. [Online]. Available: <https://arxiv.org/abs/1512.00567>
- [2] ma7555, "Cat breeds dataset," 2019. [Online]. Available: <https://www.kaggle.com/datasets/ma7555/cat-breeds-dataset>
- [3] F. Mehfooz, "Inceptionv3 with roc - auc 92%," 2021. [Online]. Available: <https://www.kaggle.com/code/fahadmehfooz/inceptionv3-with-roc-auc-92>