# Comparação de Diversos Modelos CNN Aplicados à Classificação de Imagens

1st Alison Augusto Miranda Pereira
*Programa de Pós-Graduação em Ciência da Computação*
*ICT-UNIFESP*
São José dos Campos, Brasil
alisonamp.eco@gmail.com
https://github.com/aaperei/data-science/blob/main/benchmark_deep_neural_networks.ipynb

***Resumo*—O presente trabalho tem o objetivo de comparar diversos modelos de Redes Neurais Convolucionais na tarefa de classificação de imagens. Utilizou-se da técnica de transferência de aprendizado a fim de se reutilizar modelos pré-treinados, fazendo apenas pequenos ajustes na camada de saída do modelo para a adaptação para o conjunto de dados em questão.**

***Palavras-chave*—Redes Neurais Convolucionais, classificação de imagens, transferência de aprendizado, modelos pré-treinados**

## I. Introdução

A tarefa de classificação de imagens por meio de aprendizado de máquina tem ampla utilização prática nos dias atuais. Na resolução desse problema ganham destaque as redes neurais convolucionais ou simplesmente CNN (*Convolutional Neural Networks*) [4]. Tal problema pode ser definido da seguinte forma: *dada uma imagem e uma lista de possíveis classes, qual a classe dessa imagem ou probabilidade dessa imagem pertencer a cada uma dessas classes?*

No processo de escolha de qual modelo CNN utilizar, estão entre os critérios aplicados: acurácia apresentada pelo modelo escolhido para os dados de validação, tamanho/complexidade do modelo e tempo gasto nas etapas de treinamento e classificação [1]. Num cenário ideal, sempre se busca pelo modelo com maior acurácia e menor tempo. Entretanto, encontrar um modelo que cumpra com excelência os dois critérios nem sempre é fácil, visto que um modelo com maior acurácia será possivelmente mais complexo e executará um maior número de operações. Isto, consequentemente, dará a esse mesmo modelo um tempo elevado nas etapas de treinamento e predição.

Diante do cenário apresentado, entra em campo um conceito chamado transferência de aprendizado (do inglês *transfer learning*). No contexto de redes neurais artificiais, transferência de aprendizado é o processo de se reaproveitar um modelo já treinado para um problema específico e aplicá-lo na resolução de um problema similar. Apenas alguns ajustes na camada de saída desse modelo são necessários, de forma que esse mesmo modelo esteja apto a resolver o novo problema.

Neste trabalho foram escolhidos alguns modelos CNN pré-treinados, realizou-se a transferência de aprendizado e utilizou-se tais modelos na tarefa de classificação da base de imagens *cats vs dogs*. Feito isso, foi apresentada uma comparação dos resultados encontrados.

## II. Metodologia

Nesta seção, é apresentada a metodologia utilizada para a análise e comparação dos diferentes modelos CNN.

### A. Critérios

Os três critérios considerados foram **acurácia** durante a etapa de validação, **tamanho/complexidade** do modelo e **tempo** de processamento nas etapas de classificação e predição. Como citado anteriormente, o objetivo é encontrar o modelo com maior acurácia e menor tempo de processamento.

Utilizou-se de modelos pré-treinados através do conjunto de dados *ImageNet*. A seguir, é apresentada uma comparação entre diferente modelos treinados para a base de dados *ImageNet* [1]:

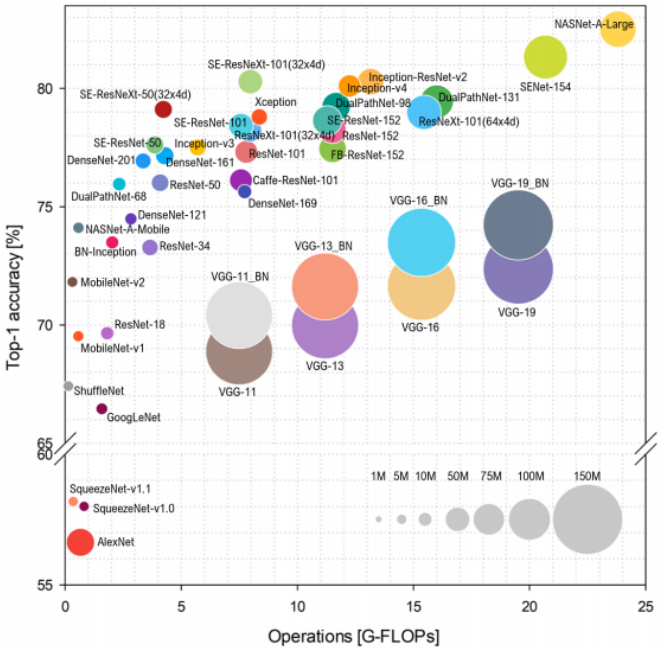


Figura 1. Acurácia vs. Tamanho e Número de Operações na tarefa de classificação de imagens (*ImageNet*) [1].

### B. Processo

Todo o experimento foi desenvolvido utilizando a linguagem de programação *Python* com o auxílio do pacote *tensorflow.keras*. Tal pacote já possui uma série de modelos pré-treinados a partir da base de dados *ImageNet*. De forma macro, o método aplicado pode ser descrito por meio dos seguintes passos:

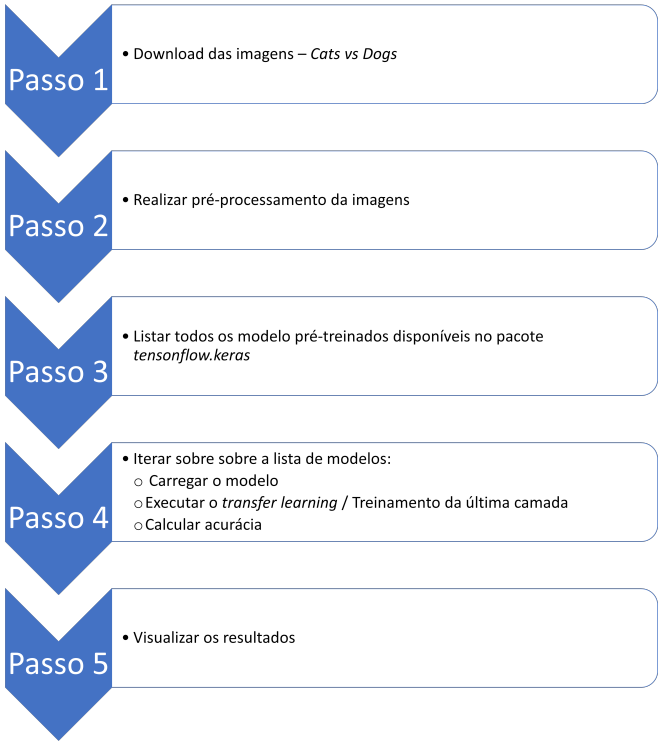


Figura 2. Fluxo do Método Aplicado no Experimento.

Na etapa de pré-processamento é feito o redimensionamento das imagens, pois alguns modelos exigem imagens no tamanho (224, 224, 3), enquanto outros no tamanho (331, 331, 3). Também foi aplicado *one-hot-encoding* às possíveis classes, de forma que seja possível utilizar *categorical crossentropy*.

A lista completa de todos os modelos disponíveis pode ser encontrada no site oficial https://www.tensorflow.org/versions/r2.7/api_docs/python/tf/keras/applications. Durante a execução desse experimento, *tensorflow* possuía 28 modelos disponíveis na versão 2.7:

— DenseNet121 — DenseNet169 — DenseNet201 — EfficientNetB0 EfficientNetB1 — EfficientNetB2 — EfficientNetB3 — EfficientNetB4 — EfficientNetB5 — EfficientNetB6 — EfficientNetB7 — InceptionResNetV2 — InceptionV3 — MobileNet — MobileNetV2 — MobileNetV3Large — MobileNetV3Small — NASNetLarge — NASNetMobile — ResNet101 — ResNet101V2 — ResNet152 — ResNet152V2 — ResNet50 — ResNet50V2 — VGG16 — VGG19 — Xception —

### C. Configurações

Nesta subseção foram listadas todas as configurações e detalhes relevantes do experimento, de forma que o mesmo seja reproduzível:

- Principais pacotes *Python* utilizados: *tensorflow* e *pandas*
- Base de dados utilizada: *Cats vs Dogs*. Baixado diretamente via tensorflow https://www.tensorflow.org/datasets/catalog/cats_vs_dogs
- Divisão treinamento/validação: 70% treinamento e 30% validação
- *Batch size*: 32 (podendo ser alterado de acordo com seus recursos disponíveis de processamento)
- Quantidade de épocas de treinamento: 3
- Função de perda: *categorical crossentropy*
- Métrica: acurácia
- Camada de saída: camada densa com função de ativação *softmax*, pois o objetivo é encontrar uma distribuição de probabilidade de cada uma das classes.
- Ambiente de execução: plataforma *Google Colaboratory PRO*, utilizando a GPU de *back-end* do *Google Compute Engine* em *Python 3* (NVIDIA Tesla P100), 27.3GB memória RAM e 170GB de disco.

## III. Resultados

Como mencionado anteriormente, foram treinados os 28 modelos de redes CNN pré-treinados do *tensorflow* e validados os resultados. Na tabela I são apresentados a acurácia para fase de validação, tamanho do modelo e tempo gasto na etapa de treinamento, ordenados de forma crescente pelo tamanho do modelo.

Tabela I
Resumo dos resultados obtidos

| Modelo | Tamanho | Acurácia | Treinamento |
|---|---|---|---|
| MobileNetV3Small | 1529968 | 0.629317 | 58.74655 |
| MobileNetV2 | 2257984 | 0.984095 | 62.38765 |
| MobileNet | 3228864 | 0.987964 | 54.21614 |
| EfficientNetB0 | 4049571 | 0.50437 | 92.57281 |
| MobileNetV3Large | 4226432 | 0.610403 | 65.70144 |
| NASNetMobile | 4269716 | 0.989253 | 132.2419 |
| EfficientNetB1 | 6575239 | 0.503797 | 128.1187 |
| DenseNet121 | 7037504 | 0.986818 | 135.2614 |
| EfficientNetB2 | 7768569 | 0.518699 | 132.6468 |
| EfficientNetB3 | 10783535 | 0.560539 | 164.7093 |
| DenseNet169 | 12642880 | 0.98997 | 164.0186 |
| VGG16 | 14714688 | 0.910446 | 139.2497 |
| EfficientNetB4 | 17673823 | 0.496203 | 216.4068 |
| DenseNet201 | 18321984 | 0.99083 | 207.4677 |
| VGG19 | 20024384 | 0.894971 | 159.0307 |
| Xception | 20861480 | 0.987964 | 167.872 |
| InceptionV3 | 21802784 | 0.98911 | 103.4869 |
| ResNet50V2 | 23564800 | 0.985385 | 126.2637 |
| ResNet50 | 23587712 | 0.653532 | 121.7114 |
| EfficientNetB5 | 28513527 | 0.523714 | 310.976 |
| EfficientNetB6 | 40960143 | 0.587047 | 388.8864 |
| ResNet101V2 | 42626560 | 0.988394 | 203.5168 |
| ResNet101 | 42658176 | 0.676171 | 202.4899 |
| InceptionResNetV2 | 54336736 | 0.992979 | 247.2473 |
| ResNet152V2 | 58331648 | 0.987821 | 343.2831 |
| ResNet152 | 58370944 | 0.672589 | 288.5863 |
| EfficientNetB7 | 64097687 | 0.496203 | 552.7204 |
| NASNetLarge | 84916818 | 0.994555 | 1270.618 |

Além disso, as figura 3 e 4 ilustram de forma gráfica e mais visual os resultado tabulados:

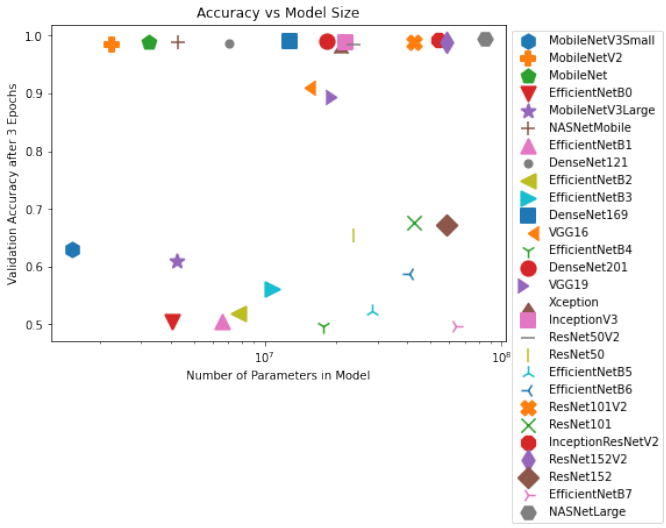


Figura 3. Acurácia na Etapa de Validação vs. Tamanho do Modelo

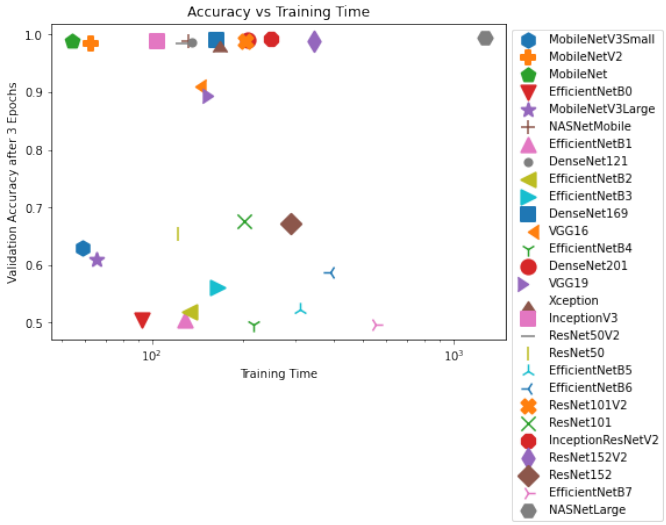


Figura 4. Acurácia na Etapa de Validação vs. Tempo de Treinamento

Analisando os resultados, nota-se que, conforme mostrado por [1], o modelo ***NASNetLarge*** é o maior (maior número de parâmetros) e apresentou maior tempo de treinamento e maior acurácia nos dados de validação. Trata-se de uma rede neural do tipo *Neural Search Space* [2], na qual foi possível atingir uma acurácia de 99,45% na classificação da base de imagens *cats vs dogs*. Por outro lado, a família de modelos ***EfficientNet*** foi a que apresentou menor acurácia (variando entre 49% e 58%).

Já os modelos da família ***MobileNet*** são simplificados, pois, como o nome sugere, foram modelos criados para dispositivos móveis e aplicações embarcadas [3]. Sendo assim, tais modelos são os menores (menor quantidade de parâmetros) e com baixo tempo de treinamento. Mesmo assim, alguns desses modelos possuem excelente acurácia. Alguns exemplos são os modelos ***MobileNet*** e ***MobileNetV2*** com 98% de acurácia apresentada para os dados de validação.

As diferentes acurácias apresentadas possuem desvio padrão de 0.212358354, o que é um número bastante elevado considerando que a acurácia é um número no intervalo [0, 1]. Isso demonstra uma alta variação na performance dos modelos, ocasionada pelas diferentes arquiteturas e quantidades de parâmetros. Comparando as figures 3 e 4, fica evidente que há uma forte relação entre tamanho do modelo e tempo de processamento. Também é possível concluir a partir dessas imagens, que nem sempre um modelo maior implica em uma acurácia maior. A partir dos resultos, é possível observar modelos de tamanhos variados com acurácia muito próximo de 100%.

## IV. Considerações Finais

Neste trabalho, foi apresentada uma comparação entre diferentes modelos CNN aplicados na classificação da base de imagens *cats vs dogs*. Tal experimento pode ser facilmente replicado para outras tarefas semelhantes, de forma que torna possível uma decisão baseada em dados concretos sobre qual é o melhor modelo para um determinado problema em questão. Além disso, a técnica de transferência de aprendizado possibilitou experimentar diversos modelos com um baixo tempo de processamento (todo o experimento foi processado em cerca de 104 minutos com os recursos descritos anteriormente ), pois apenas a última camada de cada modelo foi de fato treinada.

## Referências


[1] S. Bianco, R. Cadene, L. Celona , e P. Napoletano, "Benchmark Analysis of Representative Deep Neural Network Architectures," arXiv:1810.00736v2 [cs.CV] 19 Oct 2018.

[2] B. Zoph, V Vasudevan, J. Shlens, e Q. V. Le, "Learning Transferable Architectures for Scalable Image Recognition," arXiv:1707.07012v4 [cs.CV] 11 Apr 2018.

[3] A. G. Howard, M. Zhu, B. Chen, e D. Kalenichenko, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv:1707.07012v4 [cs.CV] 11 Apr 2018.

[4] F. Sultana, A. Sufian, e P. Dutta, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv:1905.03288v1 [cs.CV] 8 May 2019.