iscte
TECNOLOGIAS
E ARQUITETURA

APVC

Modelos pré-treinados e Transferência de conhecimento

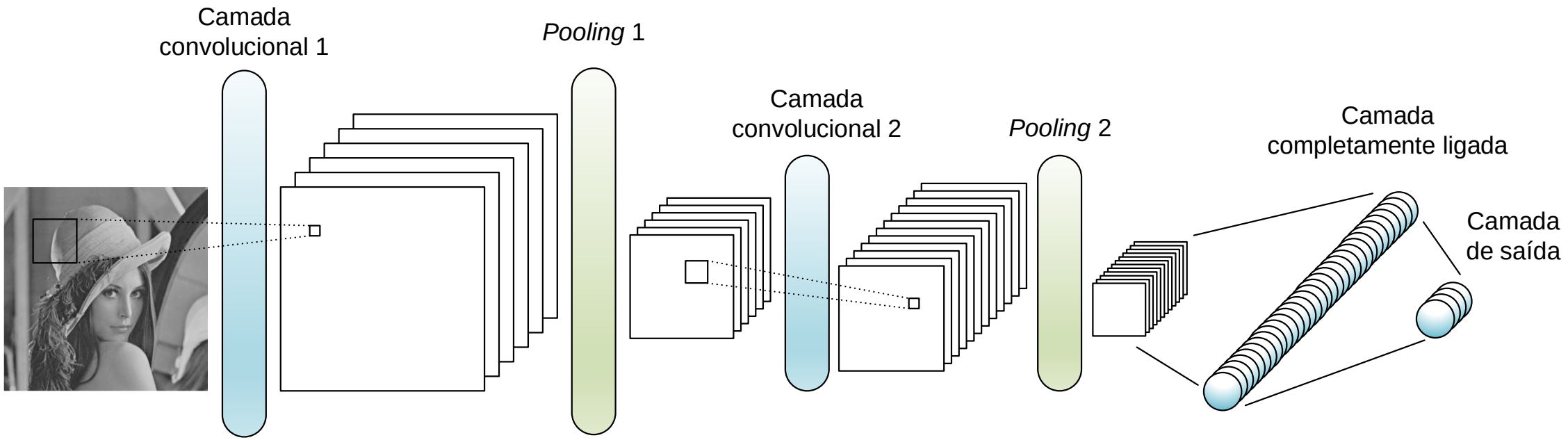
Sumário

- Modelos pré-treinados de redes neuronais convolucionais
  - Redes da família VGG
  - Redes MobileNet
  - Redes da família ResNet
- Transferência de conhecimento
  - Conceito
  - Exemplos

iscte
TECNOLOGIAS
E ARQUITETURA

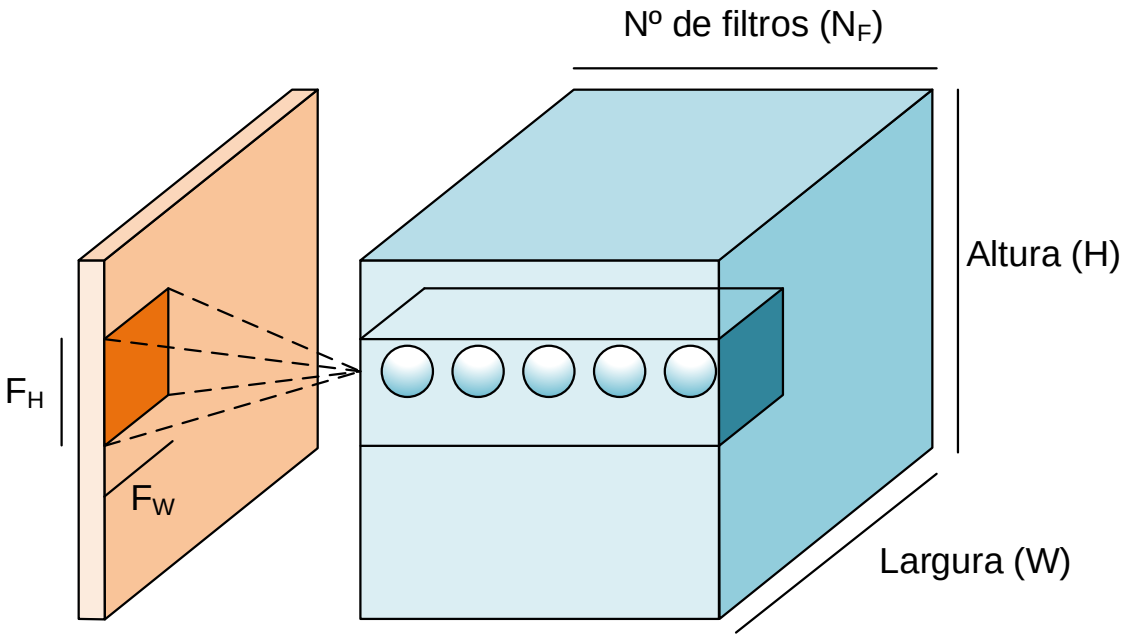
Representação de uma CNN

Relembrando a aula passada…



- **Camadas convolucionais**
- **Camadas de *pooling***
- **Camadas *densas* ou *completamente ligadas***

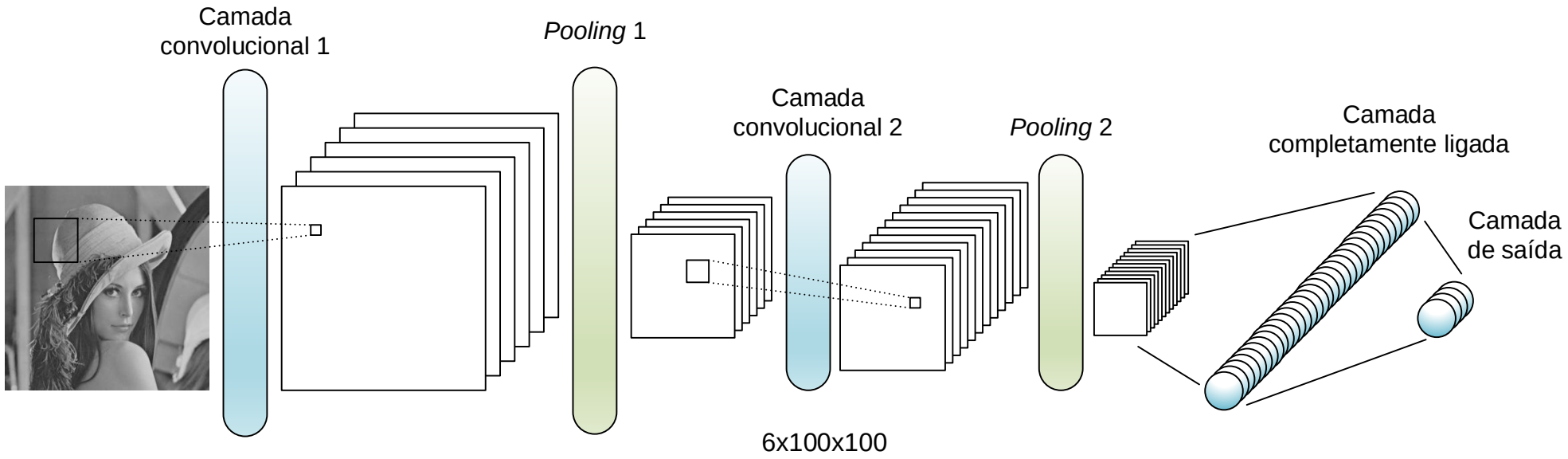
Representação das camadas de uma CNN



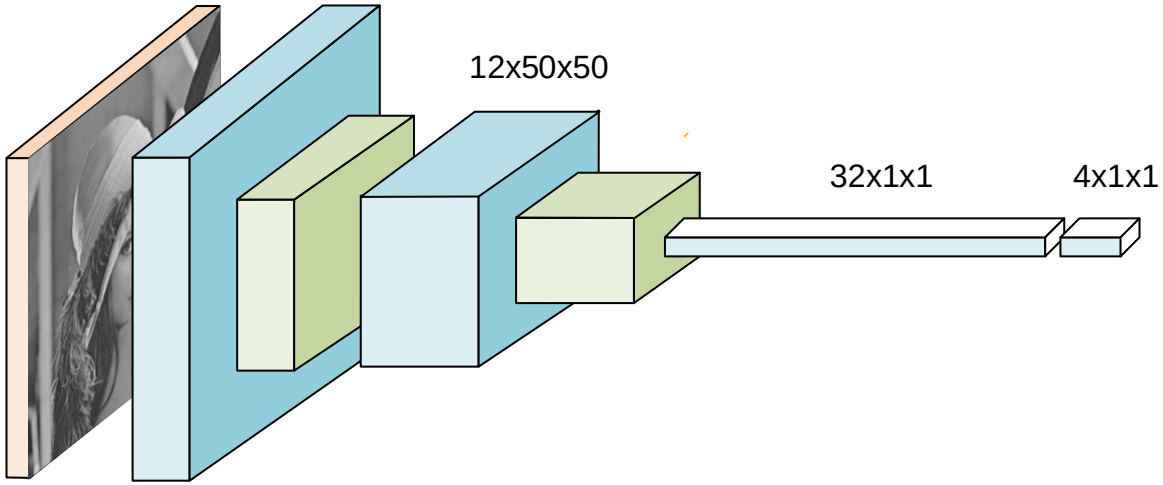
- **H e W** – dimensões horizontal e vertical de cada *feature map*
- **N_F** – número de filtros
- **F_H** e **F_W** – dimensões dos filtros
- **C** – número de matrizes (canais) que entram na camada

- Número total de neurónios: **N_F** x **H** x **W**
- Número total de parâmetros a estimar: (**F_H** x **F_W** x **C + 1)** x **N_F**

Representação “em volume” de uma CNN



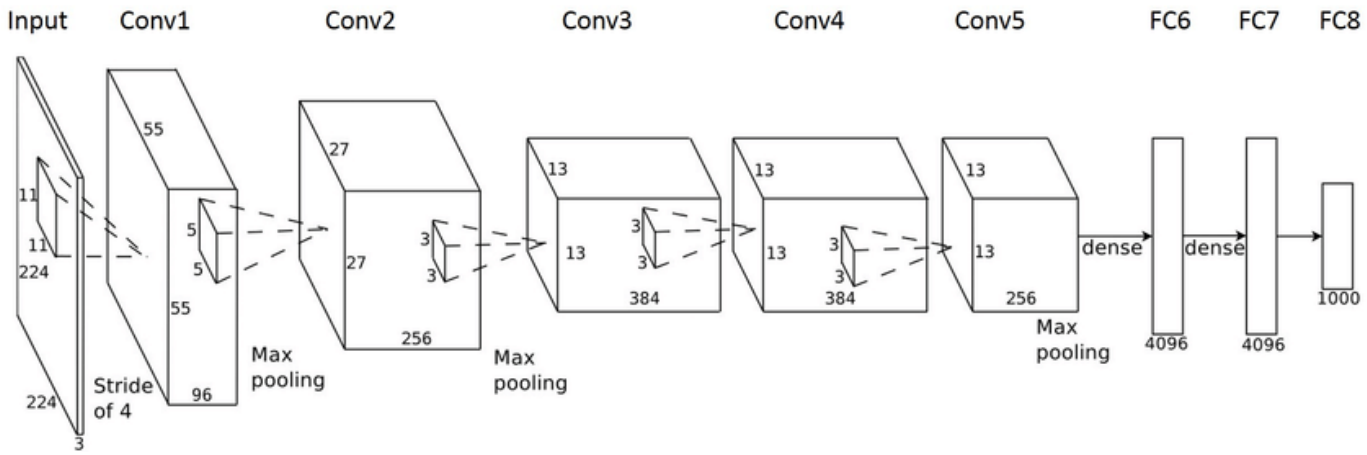
Supondo que aceita
imagens de 100x100…



iscte
TECNOLOGIAS
E ARQUITETURA

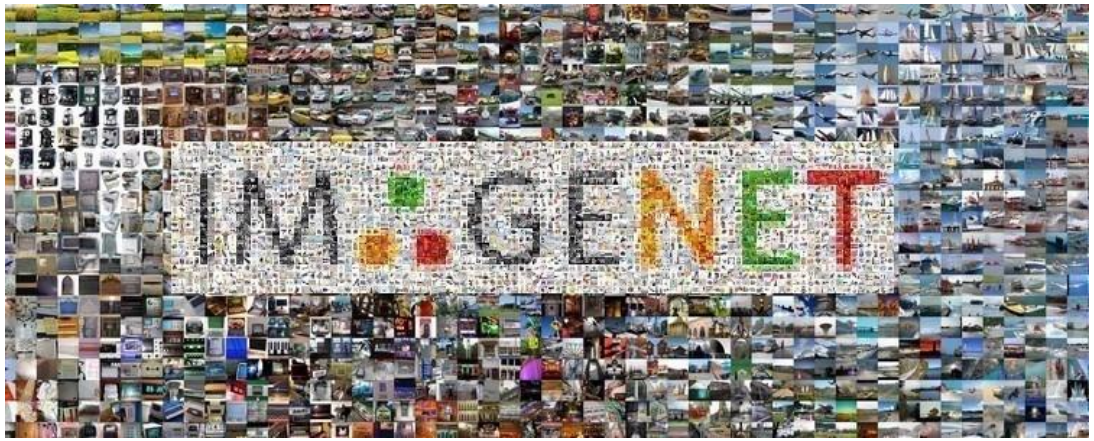
Modelos pré-treinados

AlexNet: a rede que colocou as CNNs na ribalta



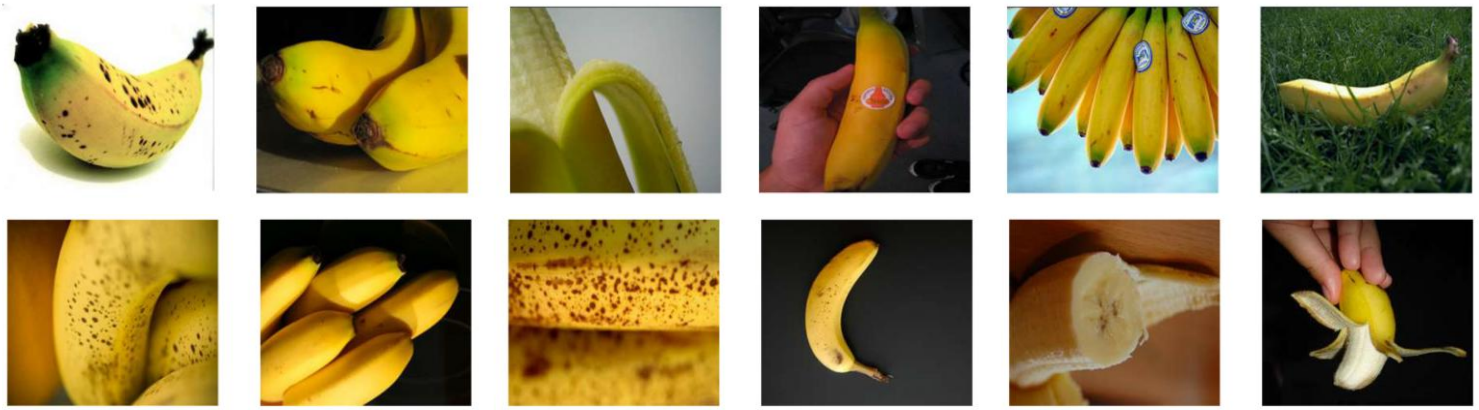
- AlexNet (2012)
  - Embora a primeira CNN – **Neocognitron** – tenha sido proposta em 1980 por Kunihiko Fukushima, a **AlexNet**, de Alex Krizhevsky, foi a que trouxe mais notoriedade às CNNs
  - Foi a primeira CNN a vencer a competição de classificação do ImageNet
  - Introdução de muitos conceitos usados hoje em dia
    - Função de ativação ReLU
    - *Data augmentation* e *dropout* no treino da rede para evitar *overfitting*
  - Composta por mais de 60 milhões de parâmetros a otimizar no treino da rede

O conjunto ImageNet



- Composto por mais de 14 milhões de imagens, distribuídas por 21841 classes

Exemplo: classe *banana,*
composta por 1409 imagens



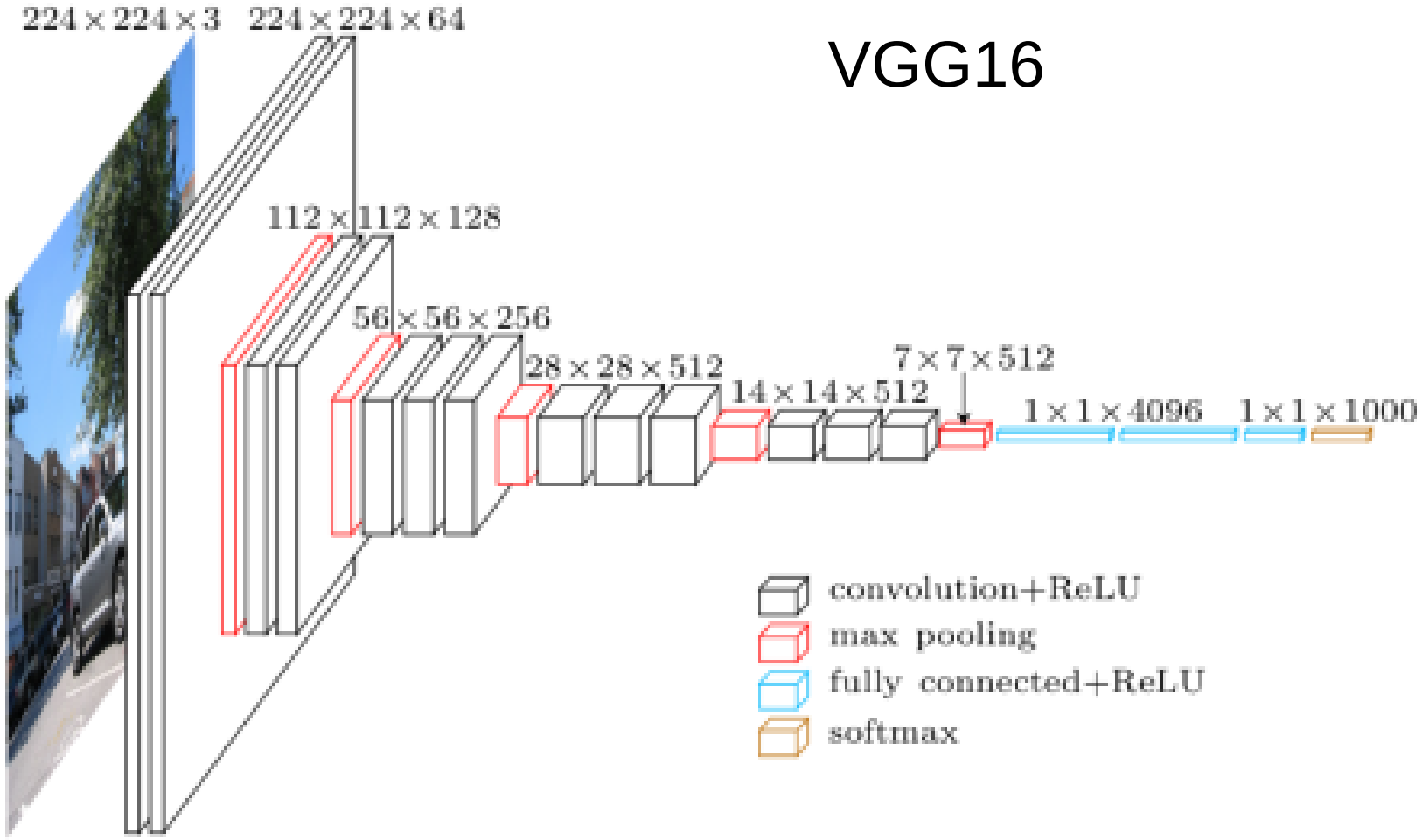
Modelos pré-treinados

- É possível utilizar redes CNN mais complexas que foram treinadas para resolver problemas de classificação similares
  - Diretamente, se as classes do nosso problema estiveram incluídas no treino original dessas redes
  - Usando *transferência de conhecimento*, se tivermos classes diferentes das usadas no treino dessas redes
- Existem várias famílias de redes pré-treinadas
  - VGG
  - ResNet
  - MobileNet
  - …

A família VGG

- Venceu competição de classificação do ImageNet em 2014
- Várias versões, cuja diferença principal é o número de camadas
  - VGG-11
  - VGG-13
  - VGG-16 (disponível no tensorflow)
  - VGG-19 (disponível no tensorflow)
- A principal diferença face à AlexNet reside nas dimensões dos filtros
  - em geral 3x3 na VGG…
  - … e que na realidade acabam por produzir efeitos semelhantes a filtros maiores, dado que a arquitetura utiliza camadas convolucionais consecutivas sem *pooling* entre elas

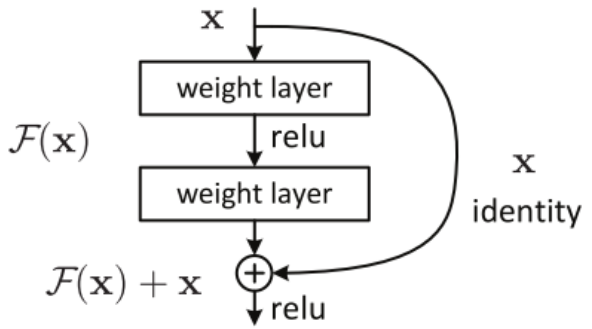
A família VGG



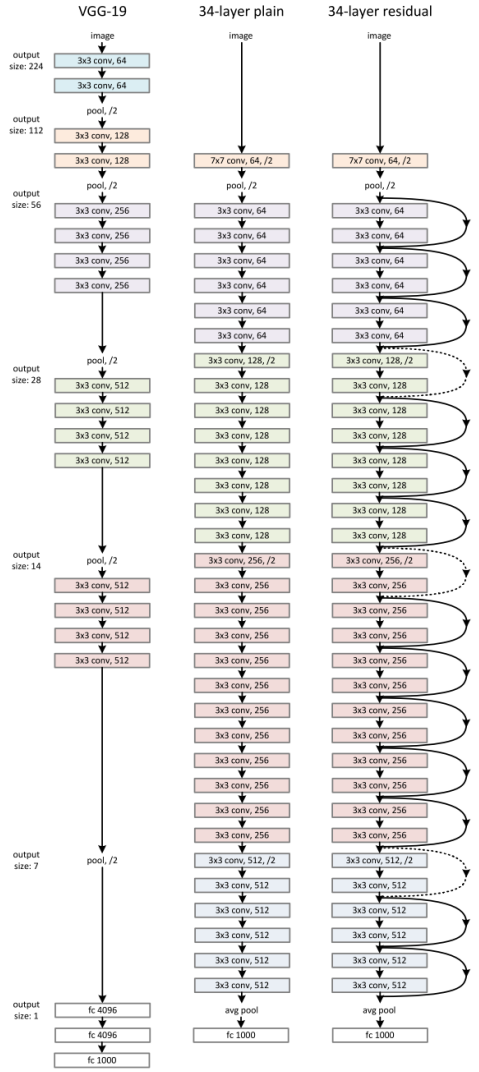
VGG16

A família ResNet

- Primeiras versões em 2015
  - ResNet34
  - ResNet50 (disponível no tensorflow)
  - ResNet101 (disponível no tensorflow)
  - ResNet152 (disponível no tensorflow)

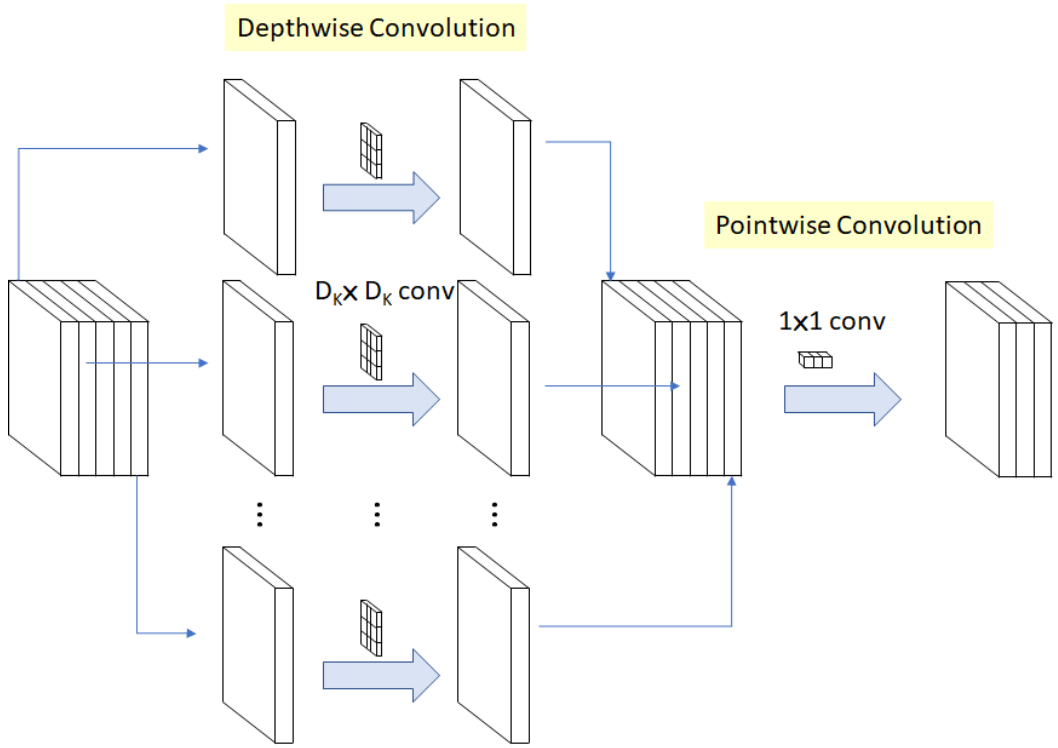


- Introduziram o conceito de *skip-connection*
  - O input de um módulo de camadas convolucionais tem um caminho em paralelo que lhe permitir “saltar” esse módulo
  - A vantagem é essencialmente no treino com back propagation, pois evita-se o problema do “vanishing gradient” em redes com uma profundidade muito grande



A família MobileNet

- Primeira versão em 2017
  - MobileNet
  - MobileNetV2
  - MobileNetV3
- Dimensionada tendo em vista a aplicação a telemóveis
- Introduziram o conceito de *convolução separável*



  - Reduz o número operações aritméticas relacionadas com a convolução à custa de uma pequena degradação no desempenho
  - De resto, conceitos similares ao que foi visto para nas famílias anteriores

Outras famílias dignas de registo

- Inception (2014), Inception-ResNet (2016) e Xception (2016)
  - Conceito de “módulos” que contemplam vários tipos de camadas convolucionais em paralelo
- EfficientNet (2019)
  - Uma arquitetura mais recente cujas dimensões (resoluções, profundidade, filtros) são facilmente escaladas para se adaptar ao contexto dos problemas
  - Pode ser configurada para ser mais leve, de modo que é tb uma alternativa à MobileNet

Utilização de redes pré-treinadas no Tensorflow

```
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras import utils
from tensorflow.keras.applications.vgg16 import preprocess_input, decode_predictions
import numpy as np

model = VGG16(weights='imagenet', classes=1000)

img_path = 'sampleImages/shark.jpg'
img = utils.load_img(img_path, target_size=(224, 224))
x = utils.img_to_array(img)
x = np.expand_dims(x)
x = preprocess_input(x)

preds = model.predict(x)

decoded_preds = decode_predictions(preds, top=5)[0]

print('Predicted: ')
for p in decoded_preds:
    print(p[1], " : ", p[2])
```

Importar o modelo. Dependendo do modelo que é, podem haver diferentes formas de parametrizar

Tipicamente os modelos pré-treinados incluem métodos para pre-processamento das imagens, que convém utilizar (alteram as dimensões, realizam normalizações, etc.)

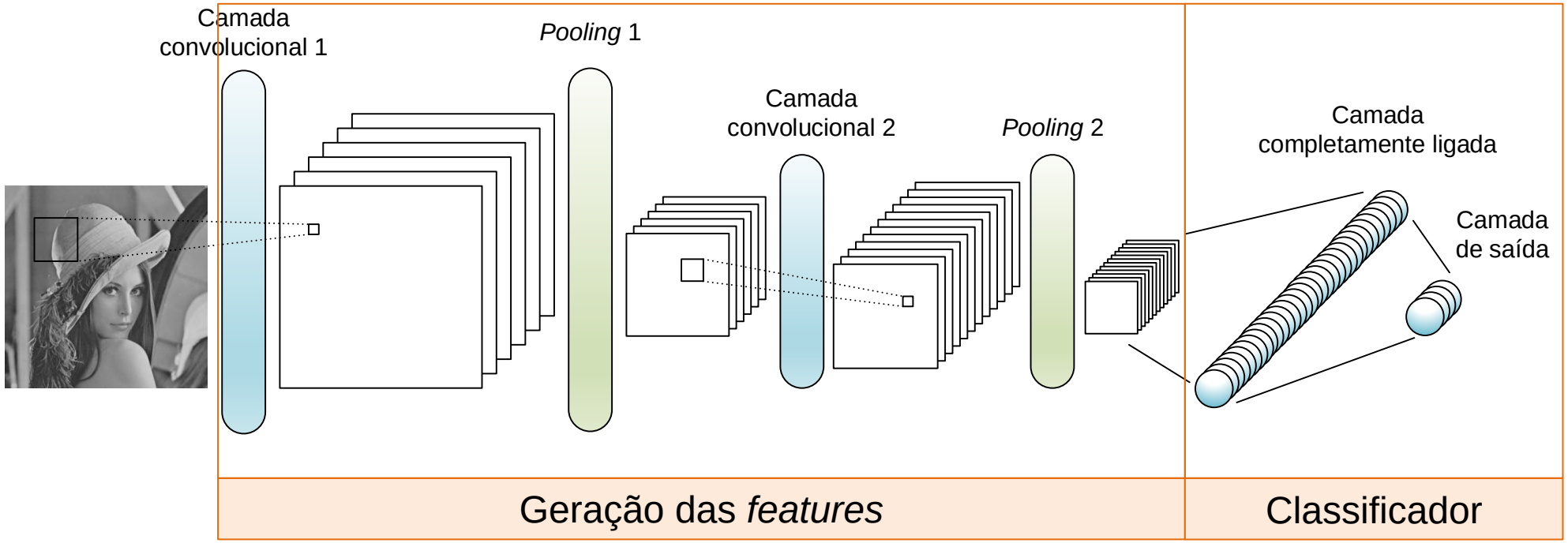
Outro método incluído no modelo – este serve para dar as predições de forma a que não tenhamos que realizar processamento para as obter

Mais info em: https://www.tensorflow.org/api_docs/python/tf/keras/applications

iscte
TECNOLOGIAS
E ARQUITETURA

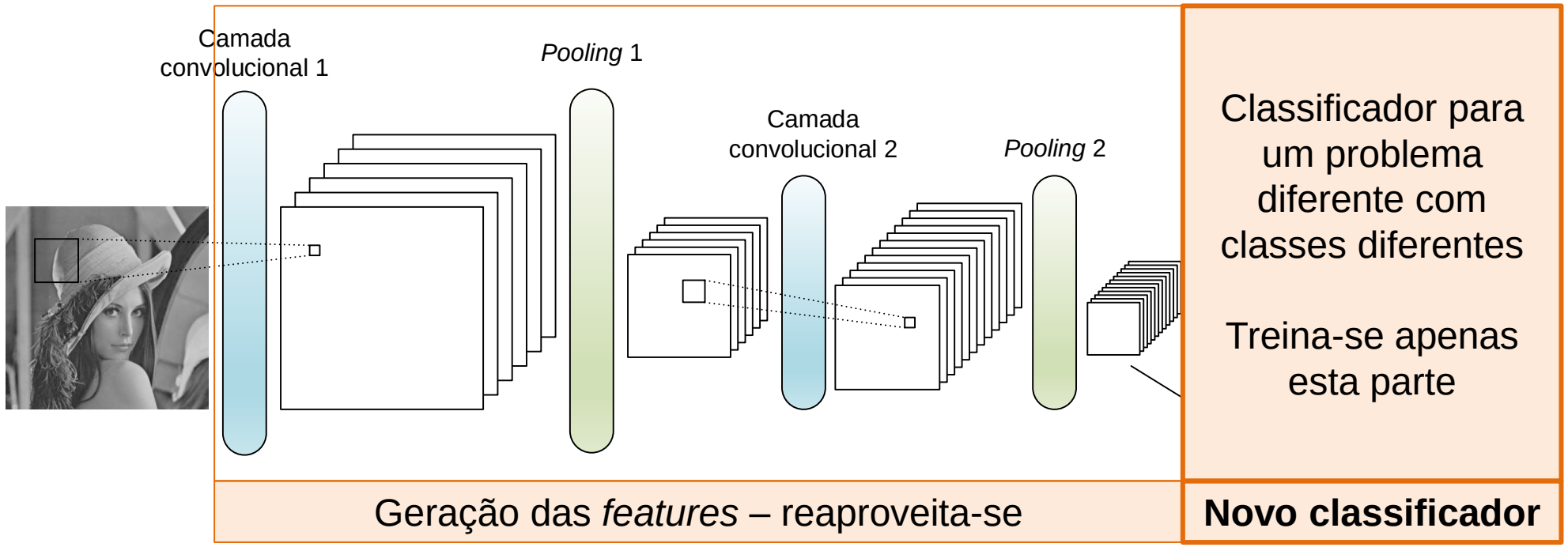
Transferência de conhecimento

Transferência de conhecimento



- A CNN pode ser vista como a junção de duas partes
  - Camadas convolucionais e de *pooling* – responsáveis por gerar as *features*
  - As camadas completamente ligadas e de saída – que na prática implementam um classificador idêntico a uma rede neuronal clássica

Transferência de conhecimento



- Este processo designa-se por **transferência de conhecimento (*transfer learning*)**
- É habitual usar desta forma arquiteturas mais complexas, *e.g.*, VGG16, MobileNet, etc. – que foram previamente treinadas

Transferência de conhecimento no Tensorflow

```
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.applications.mobilenet import MobileNet

mobileNetModel = MobileNet(input_shape=(img_height, img_width, 3), include_top=False)
mobileNetModel.summary()
mobileNetModel.trainable = False

model = tf.keras.models.Sequential([
    layers.Rescaling(2./255, offset=-1, input_shape=(img_height, img_width, 3)),
    mobileNetModel,
    layers.Flatten(),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(5, activation="softmax")
])

model.compile(…)
…
```

Importar o modelo. `include_top=False` significa que não se pretendem incluir as camadas densas originais do modelo

Importante! Assinalar que não se pretende treinar os pesos do modelo importado

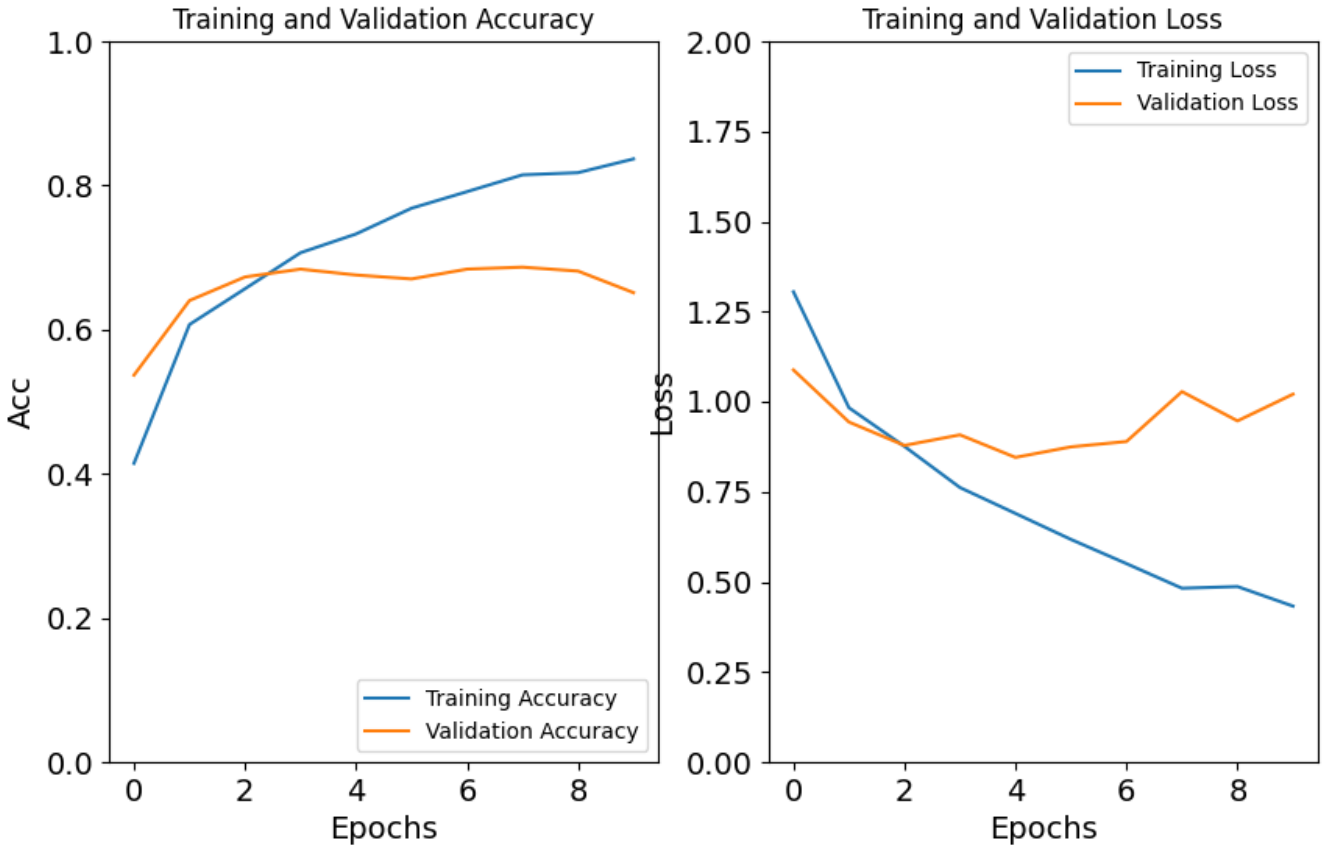
Definir o modelo

Normalizar os inputs para o que é esperado à entrada da MobileNet: valores entre -1 e 1 (documentação da MobileNet)

Juntar o modelo pré-treinado às restantes camadas da rede

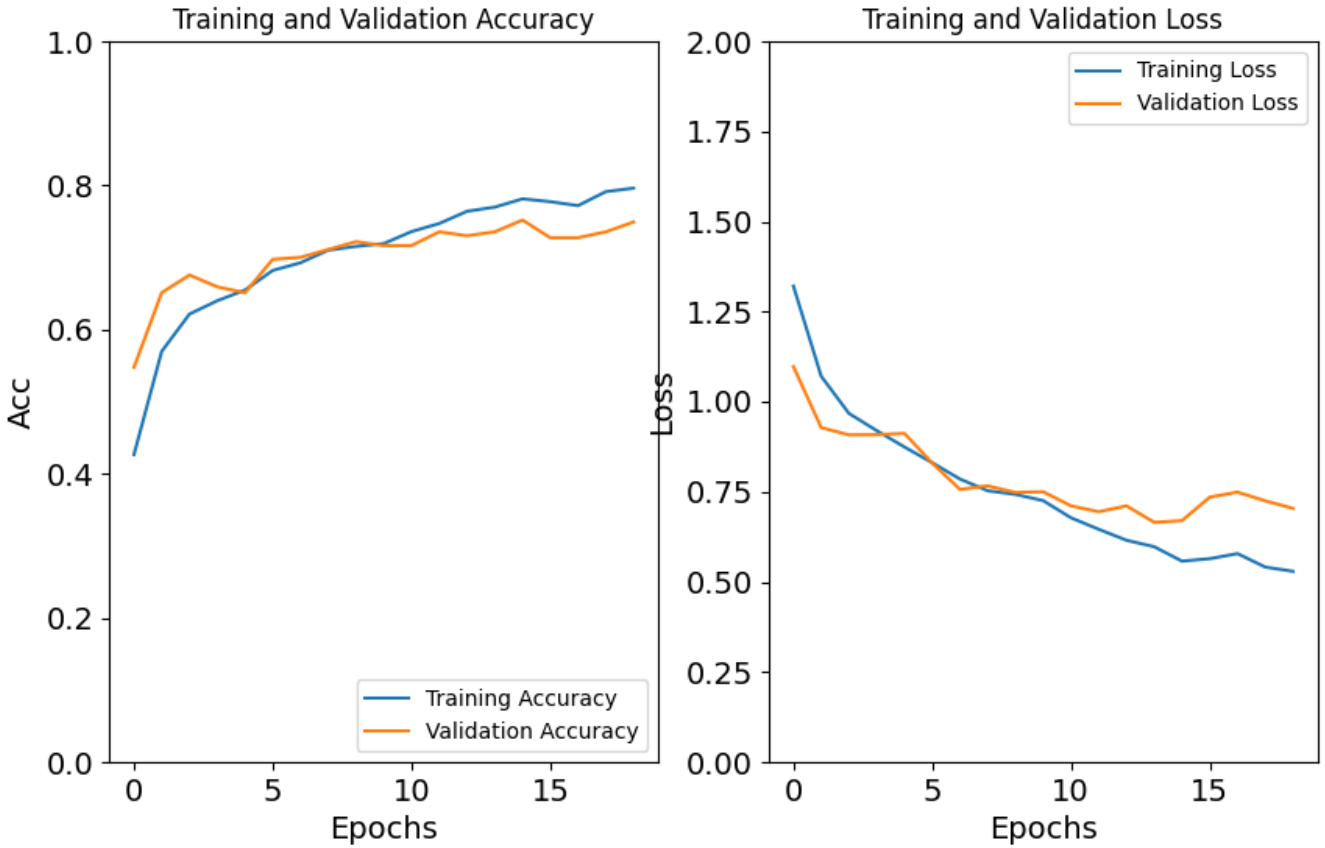
Resultados no dataset Flower_Photos

CNN *custom* (Acc_{val} = 68.7%)



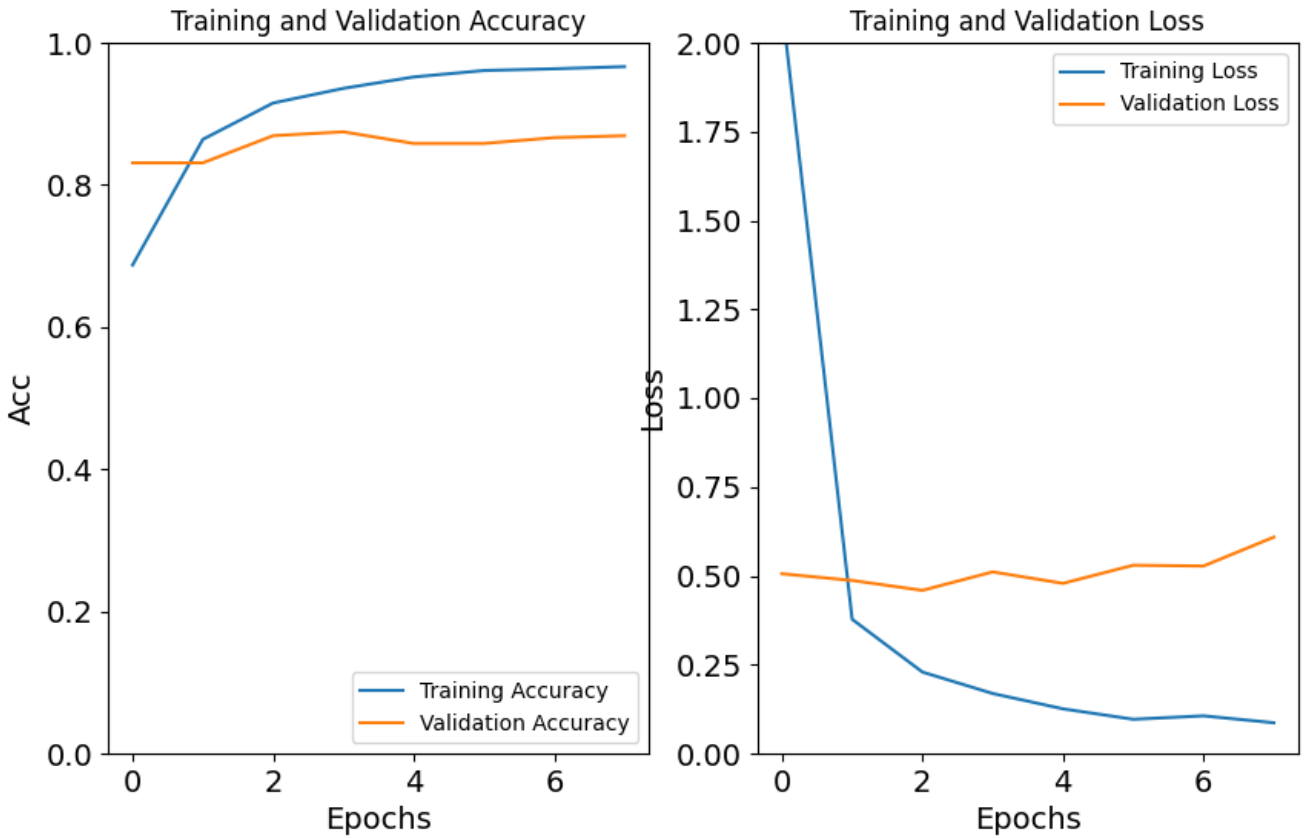
Resultados no dataset Flower_Photos

CNN *custom* + *dropout* + *data augmentation* (Acc_{val} = 75.2%)



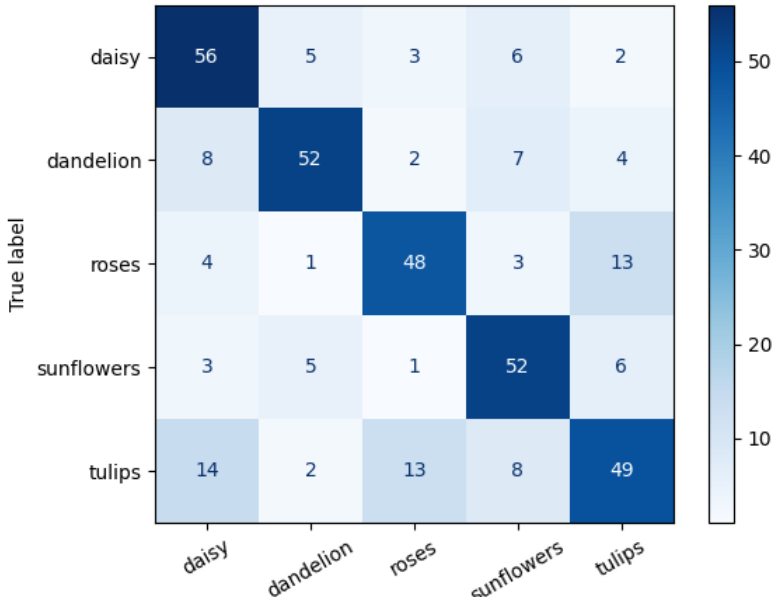
Resultados no dataset Flower_Photos

CNN com transferência de conhecimento – MobileNet (Acc_{val} = 87.5%)



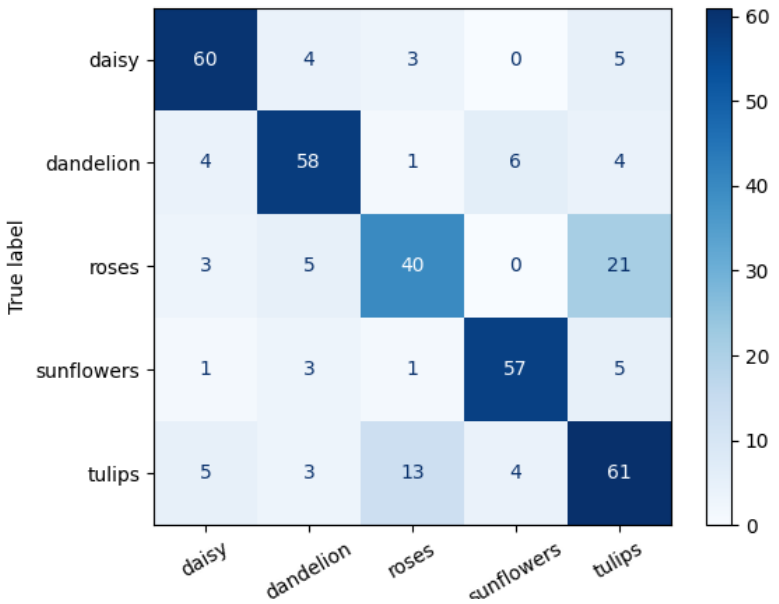
Resultados no dataset Flower_Photos

CNN *custom*



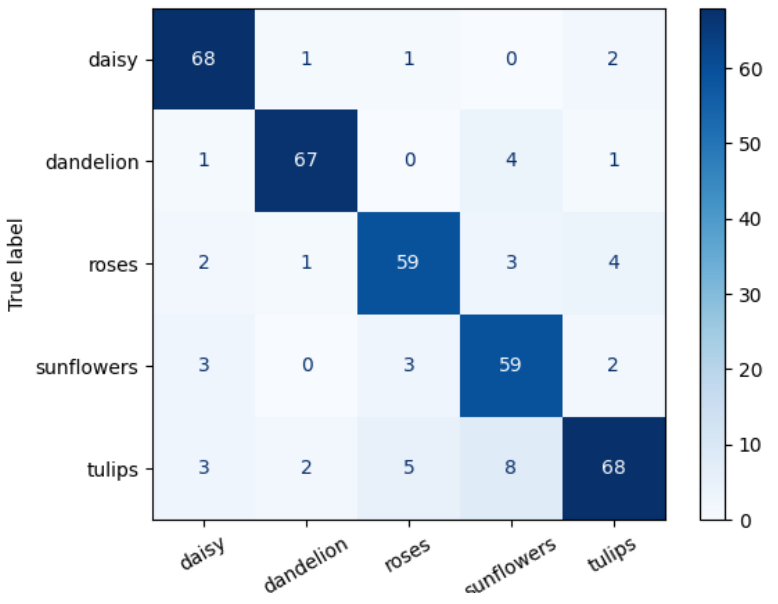
Acc_{val} = 68.7%

CNN *custom + dropout + data augmentation*



Acc_{val} = 75.2%

CNN com *transfer learning* (`MobileNet`)



Acc_{val} = 87.5%

Recursos

- **VGG**
  Karen Simonyan & Andrew Zisserman, ***Very Deep Convolutional Networks for Large-Scale Image Recognition***, Computer Vision and Pattern Recognition, 2015,
  https://arxiv.org/abs/1409.1556

- **ResNet**
  Kaiming He et. al., ***Deep Residual Learning for Image Recognition*****,** Computer Vision and Pattern Recognition, 2015,
  https://arxiv.org/abs/1512.03385

- **MobileNet**
  Andrew G. Howard et al., ***MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*****,** Computer Vision and Pattern Recognition, 2017,
  https://arxiv.org/abs/1704.04861

- **…**