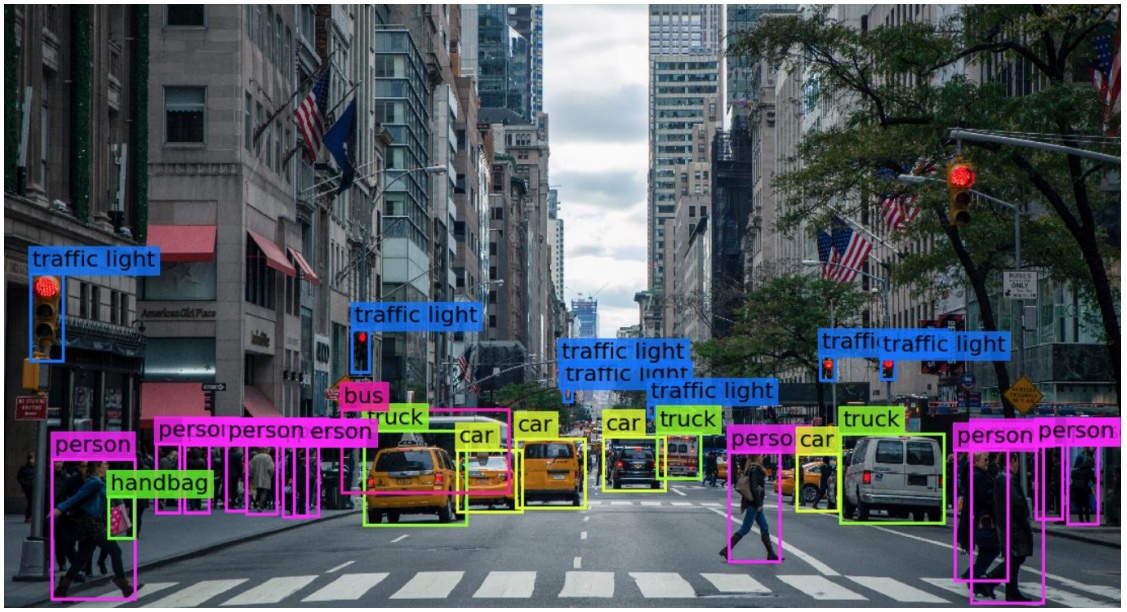


Computer vision in the new era of Artificial Intelligence and Deep Learning

Visión por computador en la nueva era de la Inteligencia Artificial y el Deep Learning

Rubén Usamentiaga*, Alberto Fernández°
*** University of Oviedo**
° TSK

Gijón (Spain)
5 – 16 April 2021

https://github.com/albertofernandezvillan/computer-vision-and-deep-learning-course

TensorFlow and Keras

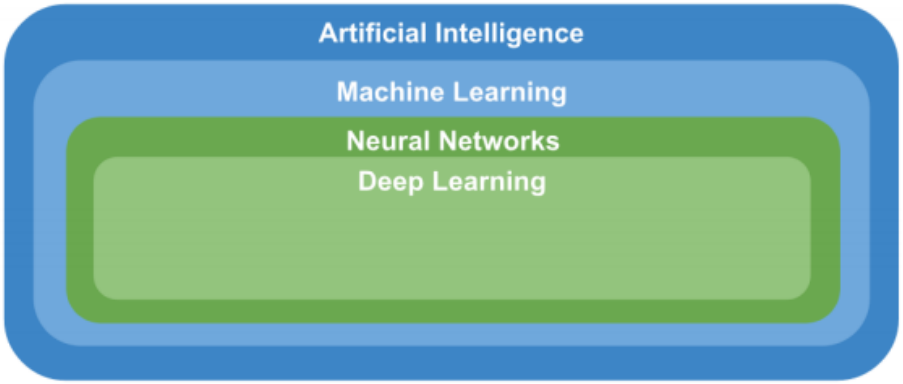
Introduction to TensorFlow and Keras

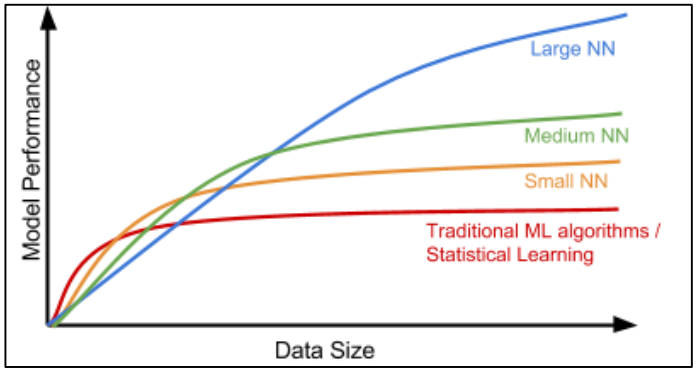
- keras_applications_prediction.ipynb
- set_up_kaggle_api_in_colab.ipynb
- keras_applications_feature_extraction_for_classification.ipynb
- keras_applications_feature_extraction_for_clustering.ipynb
- keras_imagedatagenerator_and_dataset_augmentation.ipynb
- keras_applications_transfer_learning_dogs.ipynb

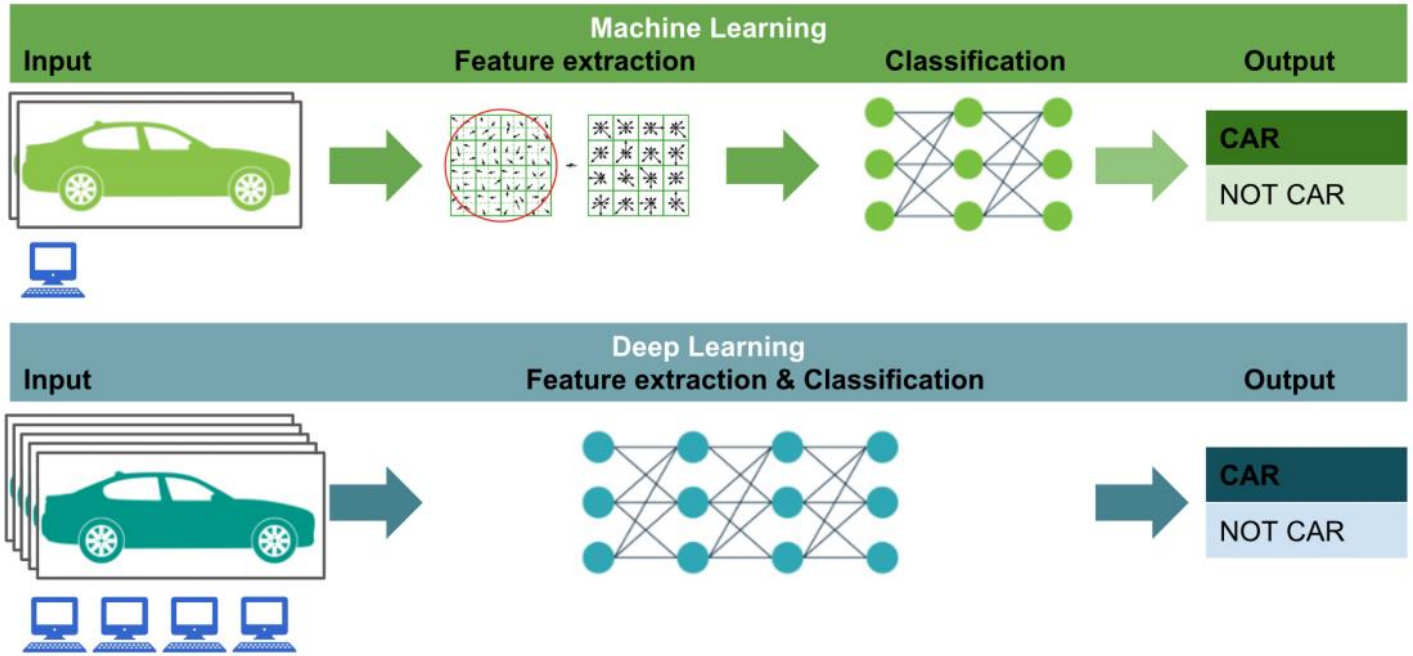
- keras_applications_prediction.ipynb
- set_up_kaggle_api_in_colab.ipynb
- keras_applications_feature_extraction_for_classification.ipynb
- keras_applications_feature_extraction_for_clustering.ipynb
- keras_imagedatagenerator_and_dataset_augmentation.ipynb
- keras_applications_transfer_learning_dogs.ipynb

https://github.com/albertofernandezvillan/computer-vision-and-deep-learning-course

Machine learning vs deep learning

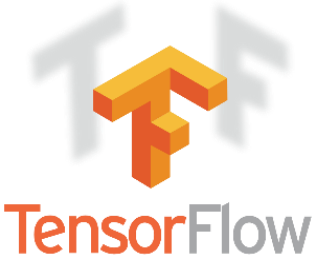






Deep learning frameworks

Deep learning frameworks introduced in this course



Introduction to Keras

Models API	Different ways to create Keras models
Layers API	API for creating layers and also using many built-in layers
Callbacks API	Performing "actions" at various stages of training (e.g. At start or end of an epoch)
Data preprocessing	Transfor raw data on disk to a Dataset. Also other functions to work with images (e.g. load an image into PIL format)
Optimizers	Different available optimizers (e.g. SGD, RMSprop, Adam)
Metrics	Many available metrics to judge the permormance of your model
Losses	Many available losses, which compute the quantity that a model should seek to minimize during training
Built-in datasets	Few toy datasets (already-vectorized, in Numpy format) that can be used for debugging a model or creating simple code examples. See also TensorFlow Datasets
Keras Applications	Deep learning models that are made available alongside pre-trained weights. These models can be used for prediction, feature extraction, and fine-tuning
Utilities	Many utilities for: 1) model plotting, 2) serialization, 3) Python, …

Introduction to Keras Applications

Keras Applications: Keras Applications are deep learning models that are made available alongside pre-trained weights. These models can be used for 1) prediction, 2) feature extraction, and 3) fine-tuning.

Example for prediction

```
from tensorflow.keras.applications.resnet50 import ResNet50
```

```
model = ResNet50(weights='imagenet')
```

Instantiates the ResNet50 architecture with the weights pre-trained on ImageNet (note that include_top is True by default)

Introduction to Keras Applications

Example for prediction

```
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input, decode_predictions
```

```
model = ResNet50(weights='imagenet')
```

Instantiates the ResNet50 architecture with the weights pre-trained on ImageNet (note that include_top is True by default

```
img_loaded = image.load_img(IMG_NAME, target_size=(224, 224))
```

Load the image (PIL format)

```
x = image.img_to_array(img_loaded)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)
```

Converts a PIL Image instance to a NumPy array

Shape: (1, 224, 224, 3)

The images are converted from RGB to BGR, then each color channel is zero-centered

```
preds = model.predict(x)
```

Get the predictions

```
decoded_preds = decode_predictions(preds, top=3)[0]
```

decode the results into a list of tuples (class, description, probability)

Introduction to Keras Applications

Example for prediction

```
from tensorflow.keras.applications.resnet50 i
from tensorflow.keras.preprocessing import im
from tensorflow.keras.applications.resnet50 i
```

tions

```
model = ResNet50(weights='imagenet')
```

Inst
tra

nts pre-
y default

```
img_loaded = image.load_img(IMG_NAME, target_
```

rmat)

```
x = image.img_to_array(img_loaded)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)
```

y array

GR, then


```
preds = model.predict(x)
```

Get the predictions

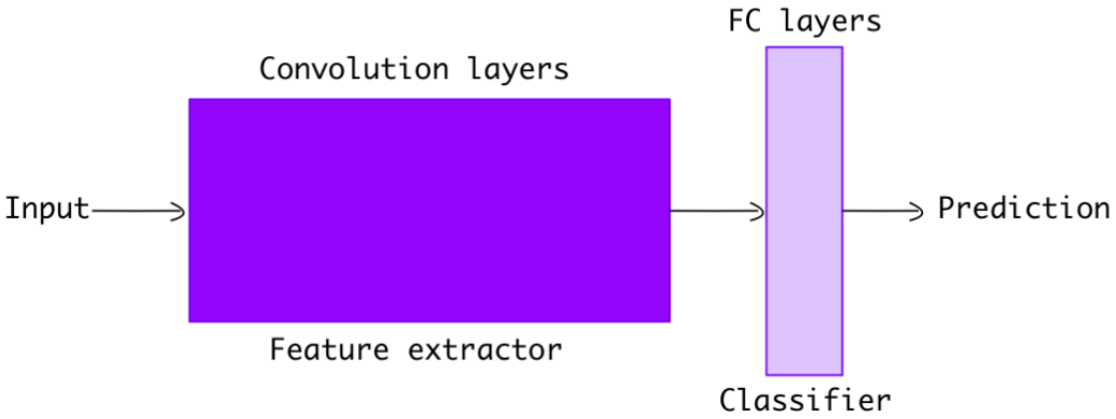
```
decoded_preds = decode_predictions(preds, top=3)[0]
```

decode the results into a list of tuples
(class, description, probability)

Introduction to Keras Applications

Example for feature extraction in a classification problem

- Convolution layers extract features from the image
- Fully connected layers classify the image using extracted features.



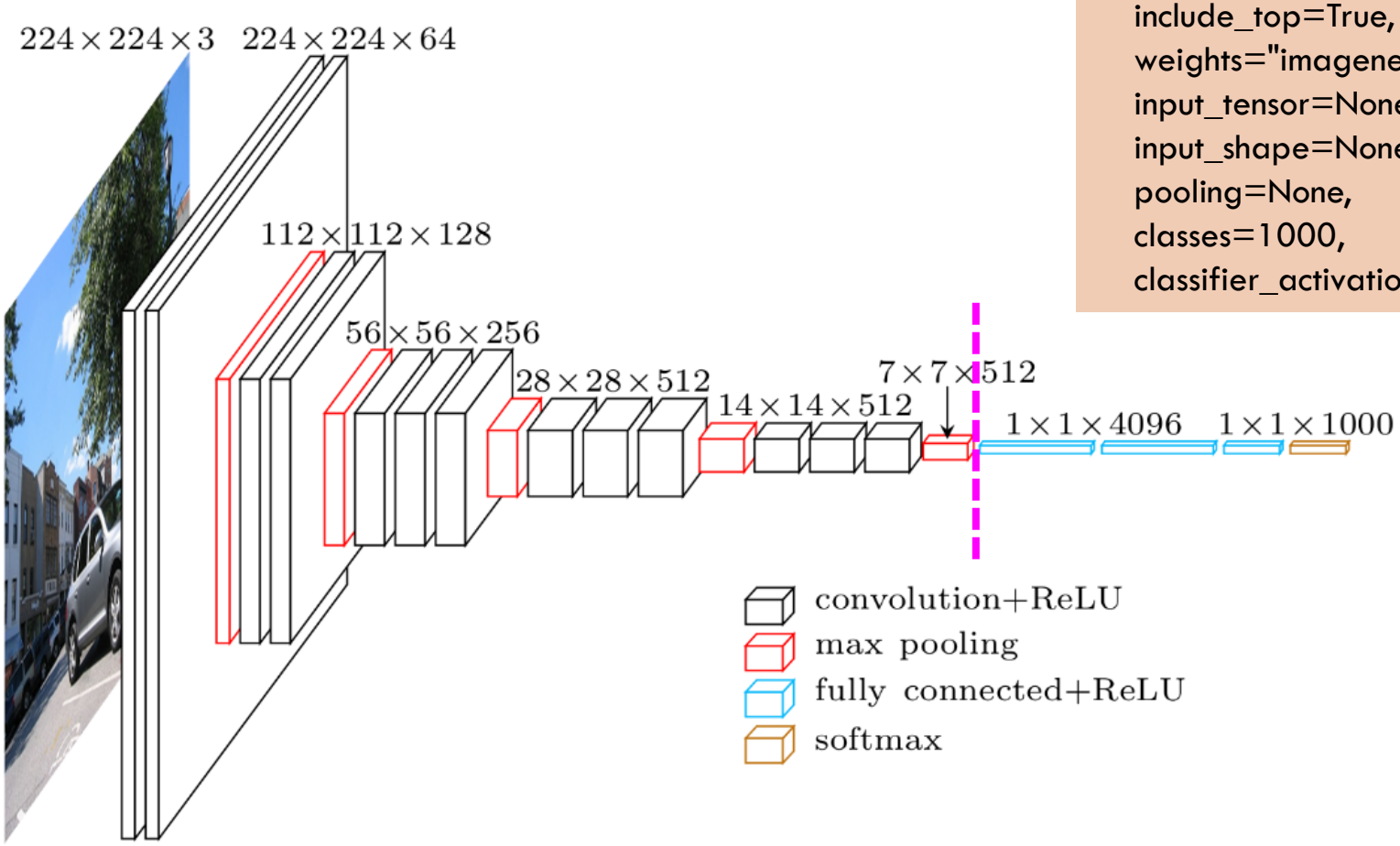
`include_top` parameter when instanciating the model indicates if including (`True`) or not including (`False`) the fully connected layers at the top of the network

Introduction to Keras Applications

Example for feature extraction in a classification problem

VGG16

```
tf.keras.applications.VGG16(
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    input_shape=None,
    pooling=None,
    classes=1000,
    classifier_activation="softmax")
```



Introduction to Keras Applications

Example for feature extraction in a classification problem

VGG16

We will use the first one. See the notebook for more information

```
model = VGG16(weights="imagenet", include_top=False, input_shape=(224, 224, 3))
•  Output from block5_pool (MaxPooling2D): (None, 7, 7, 512)
```

```
model_2 = VGG16(weights="imagenet", include_top=False, input_shape=(112, 112, 3))
•  Output from block5_pool (MaxPooling2D)   (None, 3, 3, 512)
```

```
model_pooling = VGG16(weights="imagenet", include_top=False, input_shape=(224, 224, 3),
pooling='max')
•  Output from global_max_pooling2d (GlobalMaxPooling2D) (None, 512)
```

```
model_2_pooling = VGG16(weights="imagenet", include_top=False, input_shape=(112, 112, 3),
pooling='max')
•  Output from global_max_pooling2d_1 (GlobalMaxPooling2D) (None, 512)
```

```
model_flatten = tf.keras.models.Sequential()
model_flatten.add(VGG16(weights="imagenet", include_top=False),input_shape=(224, 224, 3))
model_flatten.add(tf.keras.layers.Flatten())
•  Output from flatten: (None, 25088)
```

Introduction to Keras Applications

Example for feature extraction in a classification problem

VGG16

```
model = VGG16(weights="imagenet", include_top=False, input_shape=(224, 224, 3))
```

- Output from block5_pool (MaxPooling2D): **(None, 7, 7, 512)**

```
import numpy as np

def get_features(path_image):
  img = load_img(path_image, target_size=(224, 224))
  img = img_to_array(img)
  img = np.expand_dims(img, axis=0)
  img = preprocess_input(img)

  vgg16_feats= model.predict(img)
  vgg16_feats_flat = vgg16_feats.flatten()

  return vgg16_feats_flat
```

For each image, this function returns a flatten array with a shape of (25088,)

Introduction to Keras Applications

Example for feature extraction in a classification problem

N_DOGS = 100

N_CATS = 100

```
vgg16_feature_list = []
labels = []

for each image:
 features = get_features(path_image)
 vgg16_feature_list.append(features)
 labels.append(class_name)
```

```
vgg16_feature_list_np = np.array(vgg16_feature_list) #(200,25088)
labels_np = np.array(labels) #(200,)
```

```
from sklearn.linear_model import LogisticRegression

log_reg_model = LogisticRegression()
log_reg_model.fit(vgg16_feature_list_np, labels_np)
```

```
label =
log_reg_model.predict(get_features(path_image).reshape(1,-1))[0]
```

Introduction to Keras Applications

Example for feature extraction in a clustering problem

N_DOGS = 100

N_CATS = 100

```
vgg16_feature_list = []
labels = []

for each image:
 features = get_features(path_image)
 vgg16_feature_list.append(features)
 labels.append(class_name)
```

```
vgg16_feature_list_np = np.array(vgg16_feature_list) #(200,25088)
labels_np = np.array(labels) #(200,)
```

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2, random_state=0)
Kmeans. fit(vgg16_feature_list_np)
```

```
label =
log_reg_model.predict(get_features(path_image).reshape(1,-1))[0]
```

Introduction to Keras Applications

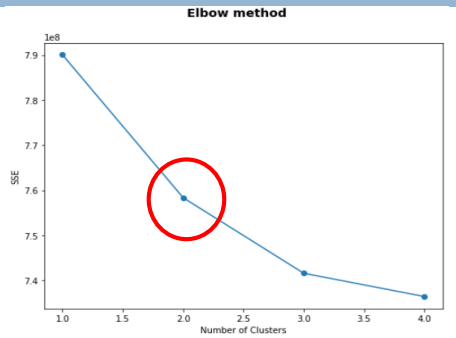
Example for feature extraction in a clustering problem

N_DOGS = 100

N_CATS = 100

```
vgg16_feature_list = []
labels = []

for each image:
 features = get_features(path_image)
 vgg16_feature_list.append(features)
 labels.append(class_name)
```



elbow method (see notebook)

```
vgg16_feature_list_np = np.array(vgg16_feature_list) #(200,25088)
labels_np = np.array(labels) #(200,)
```

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=2, random_state=0)
Kmeans. fit(vgg16_feature_list_np)
```

```
label =
log_reg_model.predict(get_features(path_image).reshape(1,-1))[0]
```

Data augmentation

https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

```
import tensorflow as tf

data_generator =
tf.keras.preprocessing.image.ImageDataGenerator(rotation_range=30)
image_iterator = data_generator.flow(images)
show_augs(image_iterator)
```

Data augmentation

https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

```
import tensorflow as tf

data_generator = tf.keras.preprocessing.image.ImageDataGenerator(zoom_range=0.25)
image_iterator = data_generator.flow(images)
show_augs(image_iterator)
```

Data augmentation

https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

```
import tensorflow as tf

data_generator = tf.keras.preprocessing.image.ImageDataGenerator(width_shift_range=0.3)
image_iterator = data_generator.flow(images)
show_augs(image_iterator)
```

Data augmentation

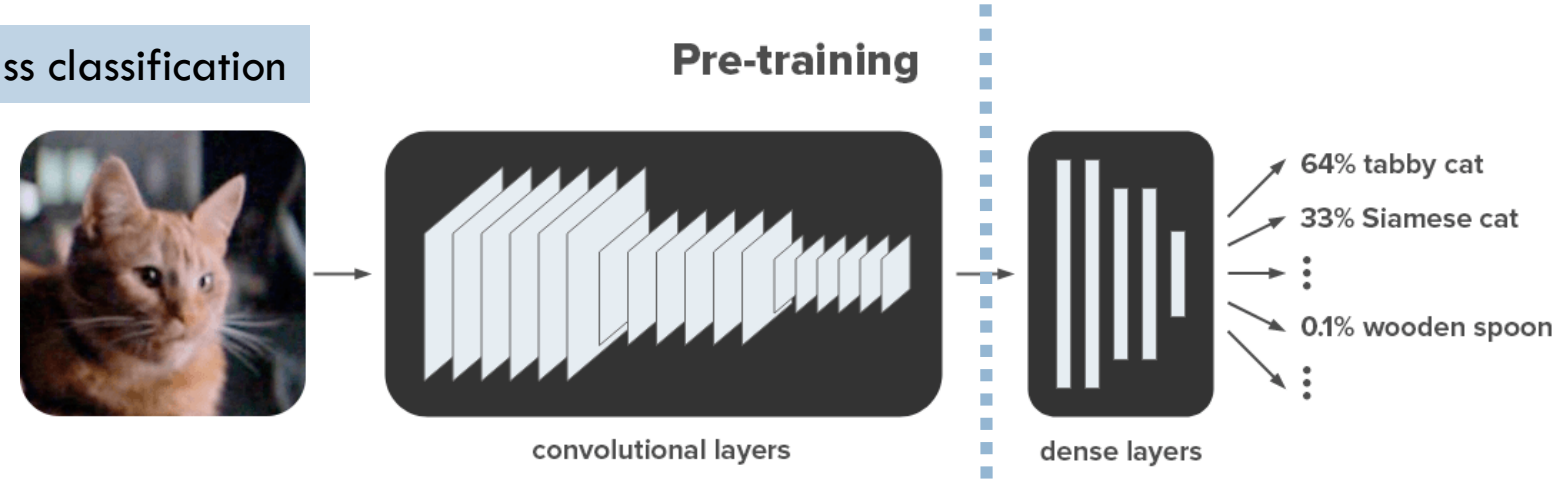
```
data_generator = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1. / 255,
    zoom_range=0.2,
    rotation_range = 5,
    horizontal_flip=True)
```

Introduction to Keras Applications

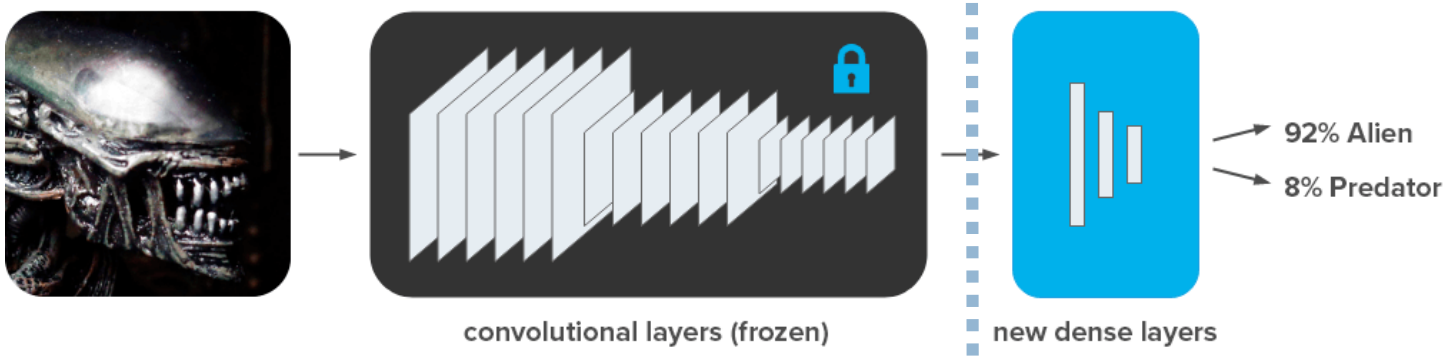
Example for transfer learning

Image taken from here

Multiclass classification



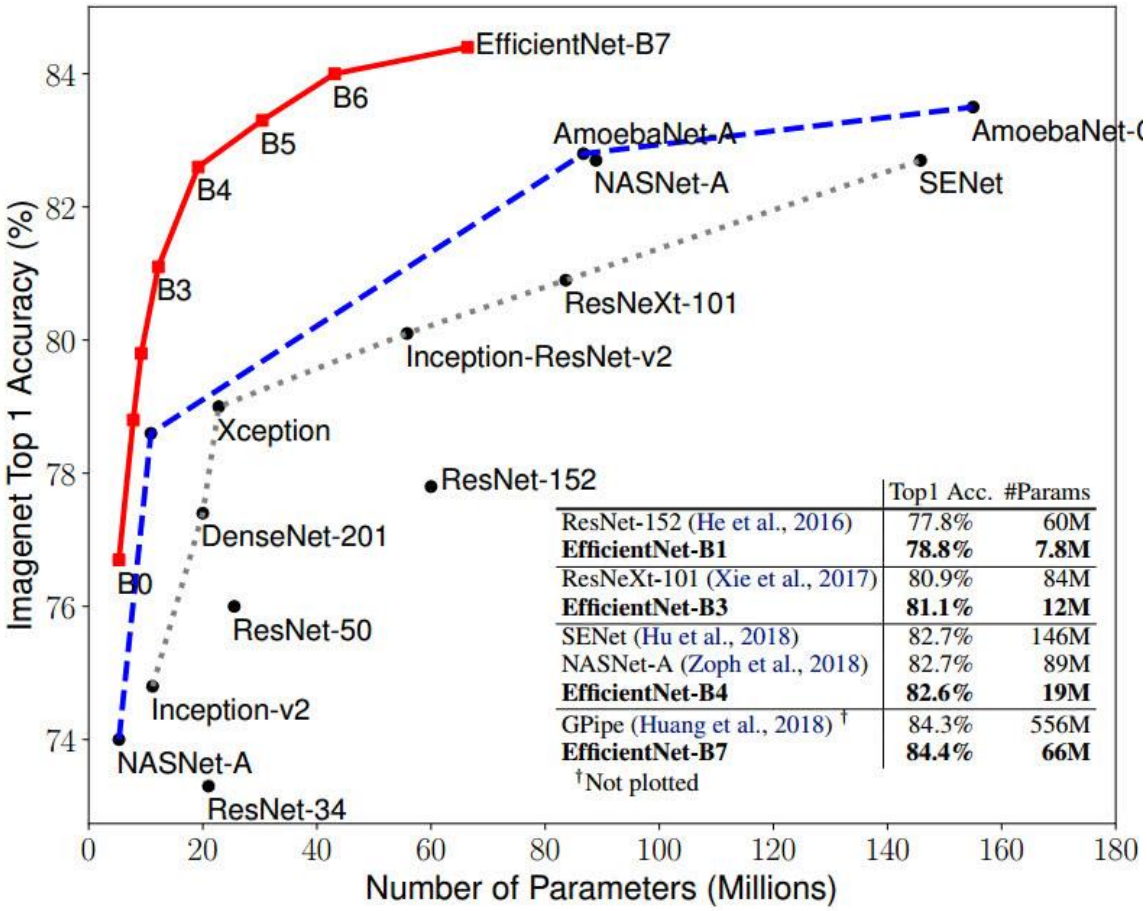
Binary classification



Introduction to Keras Applications

Example for transfer learning

See the notebook for more information



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**78.8%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.1%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.6%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted

Base model	resolution
EfficientNetB0	224
EfficientNetB1	240
EfficientNetB2	260
EfficientNetB3	300
EfficientNetB4	380
EfficientNetB5	456
EfficientNetB6	528
EfficientNetB7	600

Introduction to Keras Applications

```
from tensorflow.keras.applications import EfficientNetB0

base_model_eff = EfficientNetB0(include_top=False,
input_shape=(INPUT_SHAPE[0], INPUT_SHAPE[1], 3),
weights="imagenet")
```

output is a 4D tensor

(None, 7, 7, 1280)

```
base_model_eff = EfficientNetB0(include_top=False, input_shape=(INPUT_SHAPE[0],
INPUT_SHAPE[1], 3), weights="imagenet")
model = models.Sequential()
model.add(base_model_eff)
model.add(layers.GlobalMaxPooling2D())
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
efficientnetb0 (Functional)  (None, 7, 7, 1280)        4049571
_________________________________________________________________
global_max_pooling2d (Global (None, 1280)              0
=================================================================
Total params: 4,049,571
Trainable params: 4,007,548
Non-trainable params: 42,023
```

output is a 2D tensor

(None, 1280)

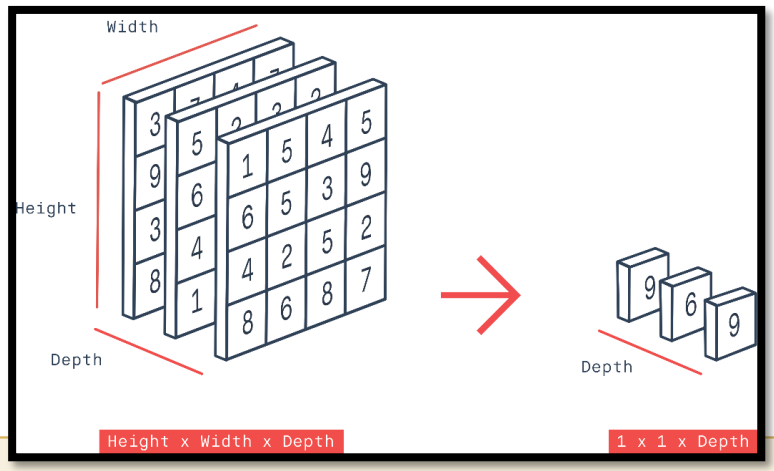
GlobalMaxPooling2D() results in a much smaller number of features compared to the Flatten()layer

tf.keras.layers.GlobalMaxPool2D()

```
#(batch_size, rows, cols, channels)
input_shape = (1, 3, 3, 3)

x = tf.random.normal(input_shape)
y = tf.keras.layers.GlobalMaxPool2D()(x)

#(batch_size, channels)
print(y.shape) # (1, 3)
```



See the notebook for more information

TensorFlow and Keras

Introduction to TensorFlow and Keras

https://github.com/albertofernandezvillan/computer-vision-and-deep-learning-course