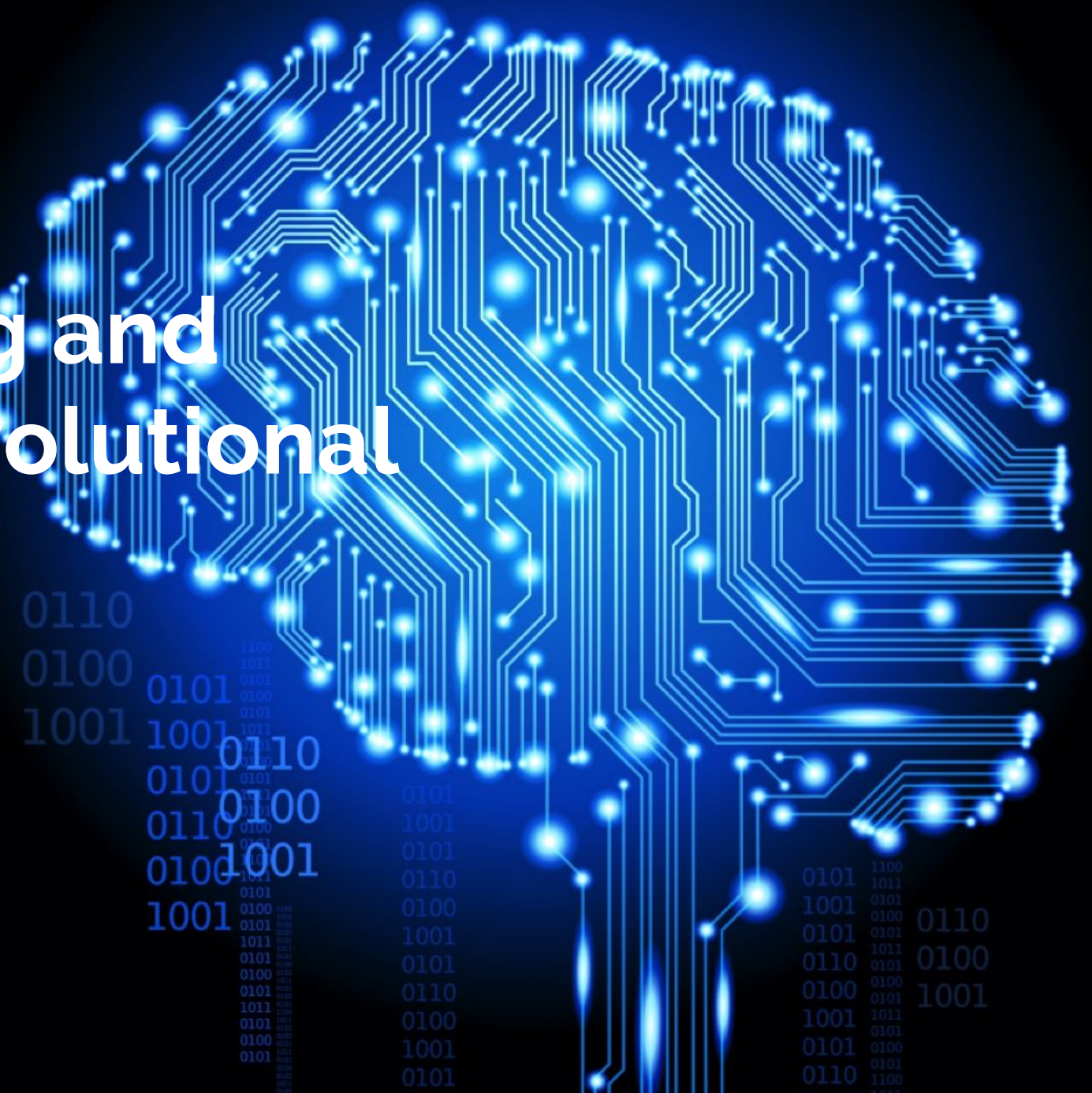
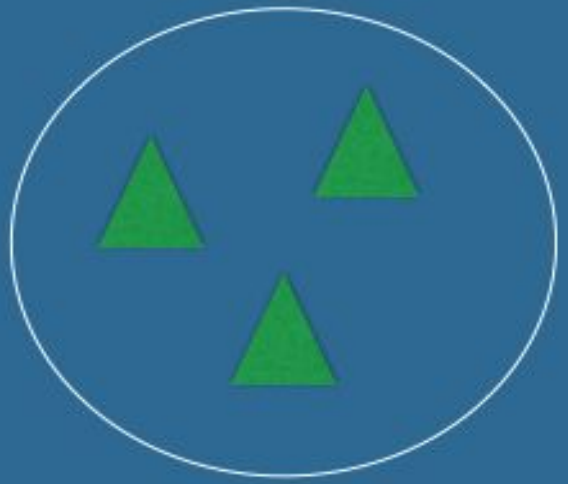
Transfer Learning and Fine-tuning Convolutional Neural Networks

ESADE - MIBA (FALL 2017)



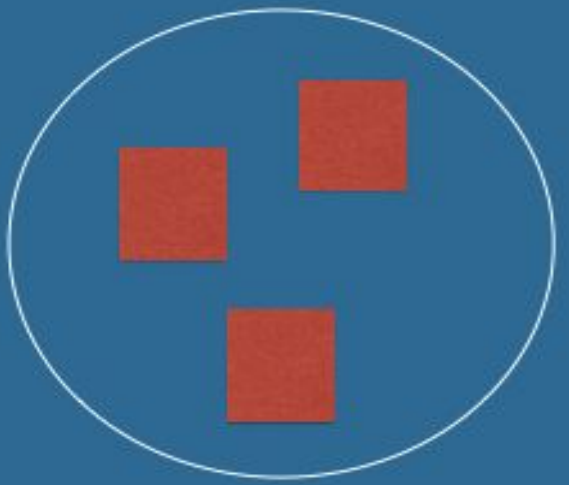
JORDI **TORRES** | FRANCESC **SASTRE**

Transfer learning



Source task / domain

Target task / domain



Storing knowledge gained solving one problem and applying it to a different but related problem.

Model

Model

Knowledge

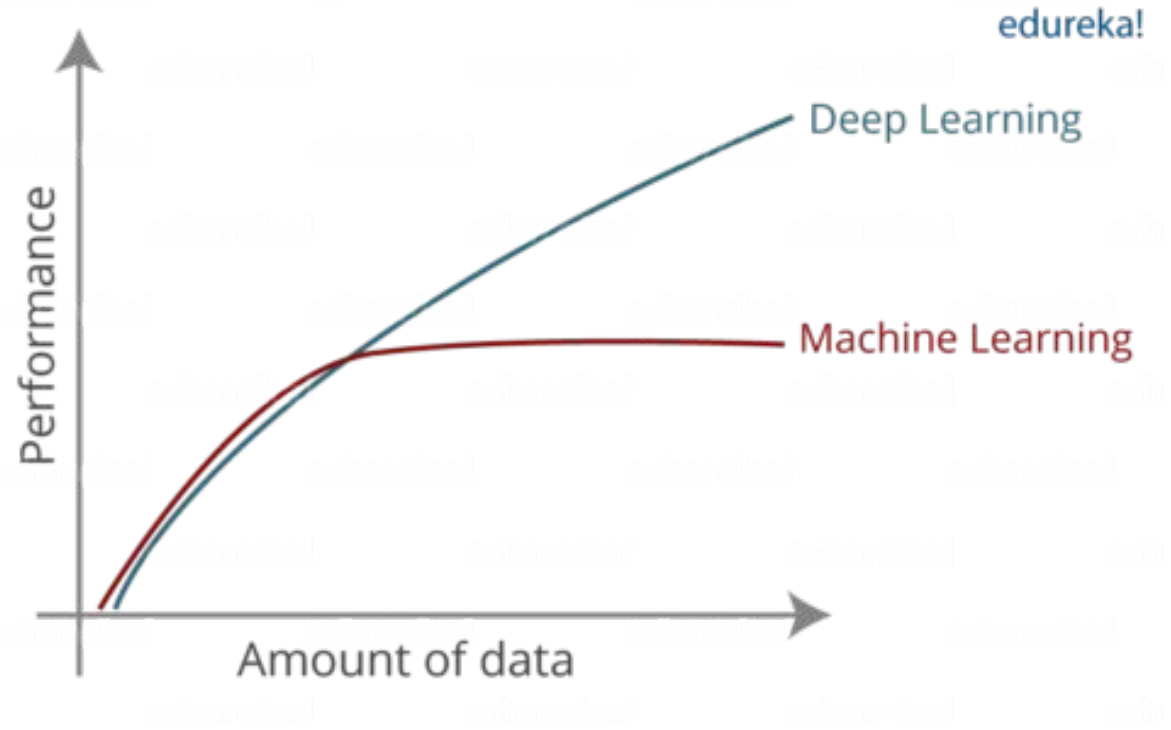
Source: Sebastian Ruder

Transfer Learning

- Use pretrained networks with other datasets
- Avoid random initialization
- Use convolutional layers features as inputs for other ML algorithms

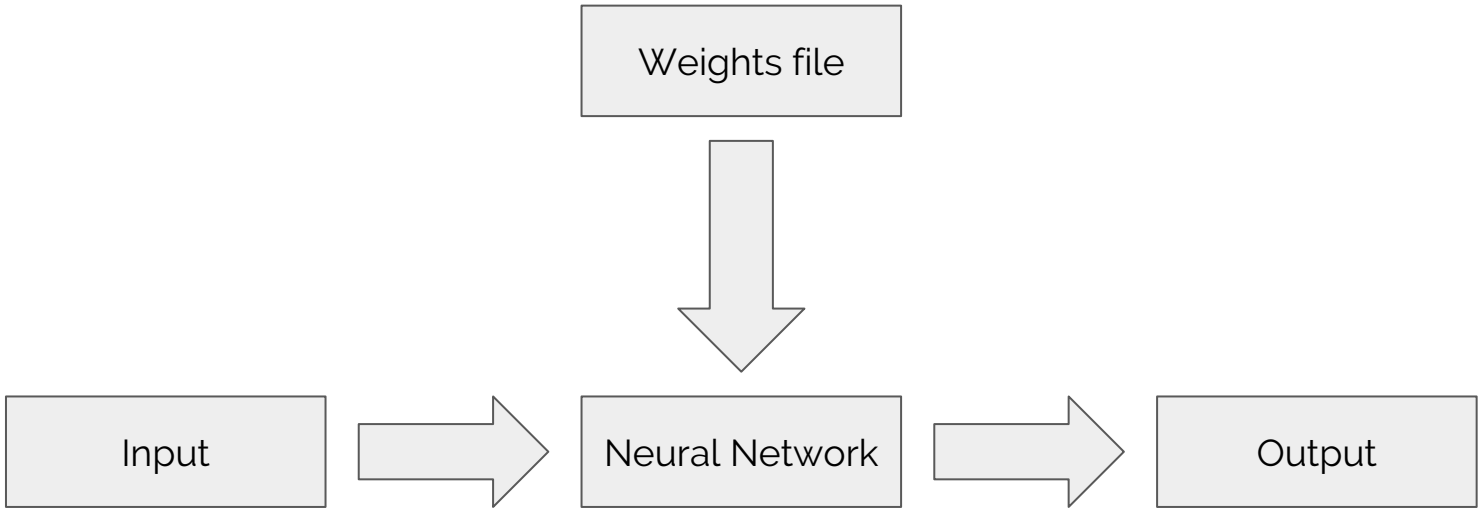
Transfer Learning

- We do not have enough data
- Big generic datasets
- Subsets are similar to our data



Source: edureka

Case 1: Pretrained Network



- Weights and biases initialized with trained values
- No training needed

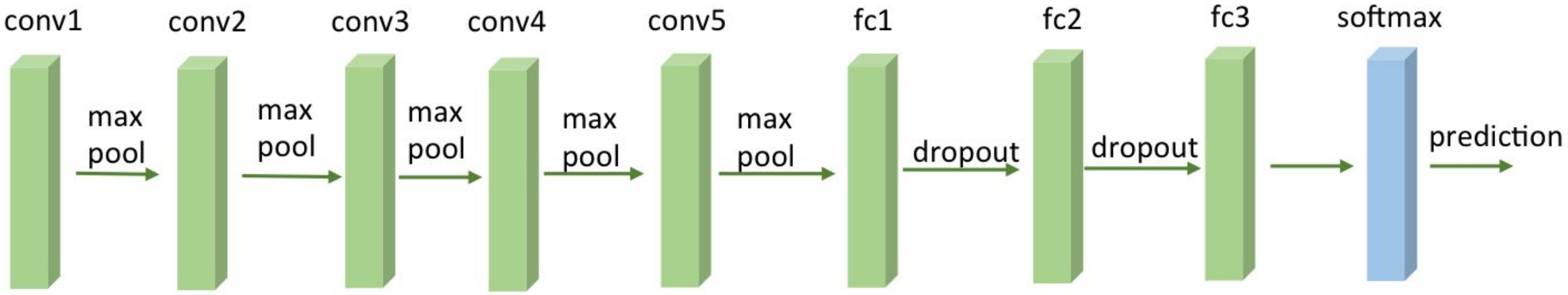
Case 1: Pretrained Network

In [ ]:
```
from keras.applications.inception_v3 import InceptionV3, preprocess_input
from keras.preprocessing import image
from keras.applications.imagenet_utils import decode_predictions
import numpy as np
import h5py
import matplotlib.pyplot as plt
import urllib.request
%matplotlib inline
```

In [ ]:
```
model = InceptionV3(weights='imagenet')
```

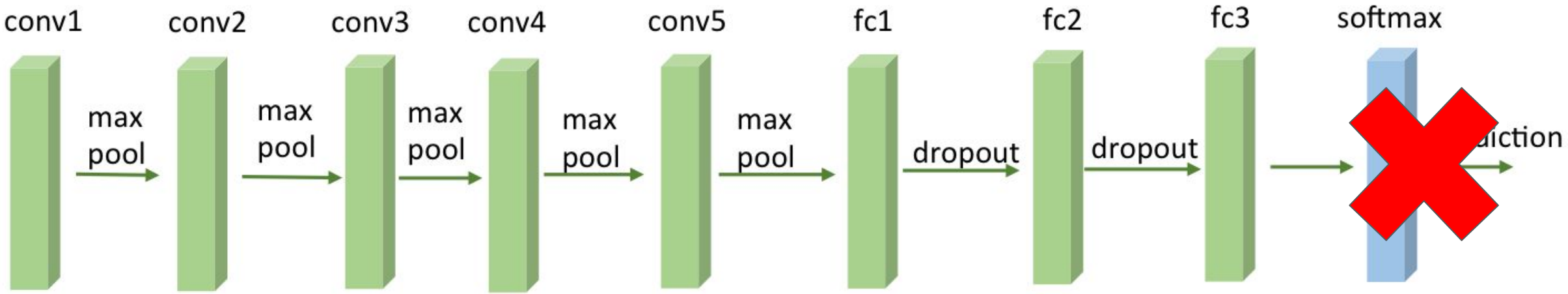
```
img = image.load_img(img_path, target_size=(299, 299))
plt.imshow(img)
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model.predict(x)
# decode the results into a list of tuples (class, description, probability)
# (one such list for each sample in the batch)
print('Predicted:', decode_predictions(preds, top=3)[0])
```

Case 2: Feature extractor

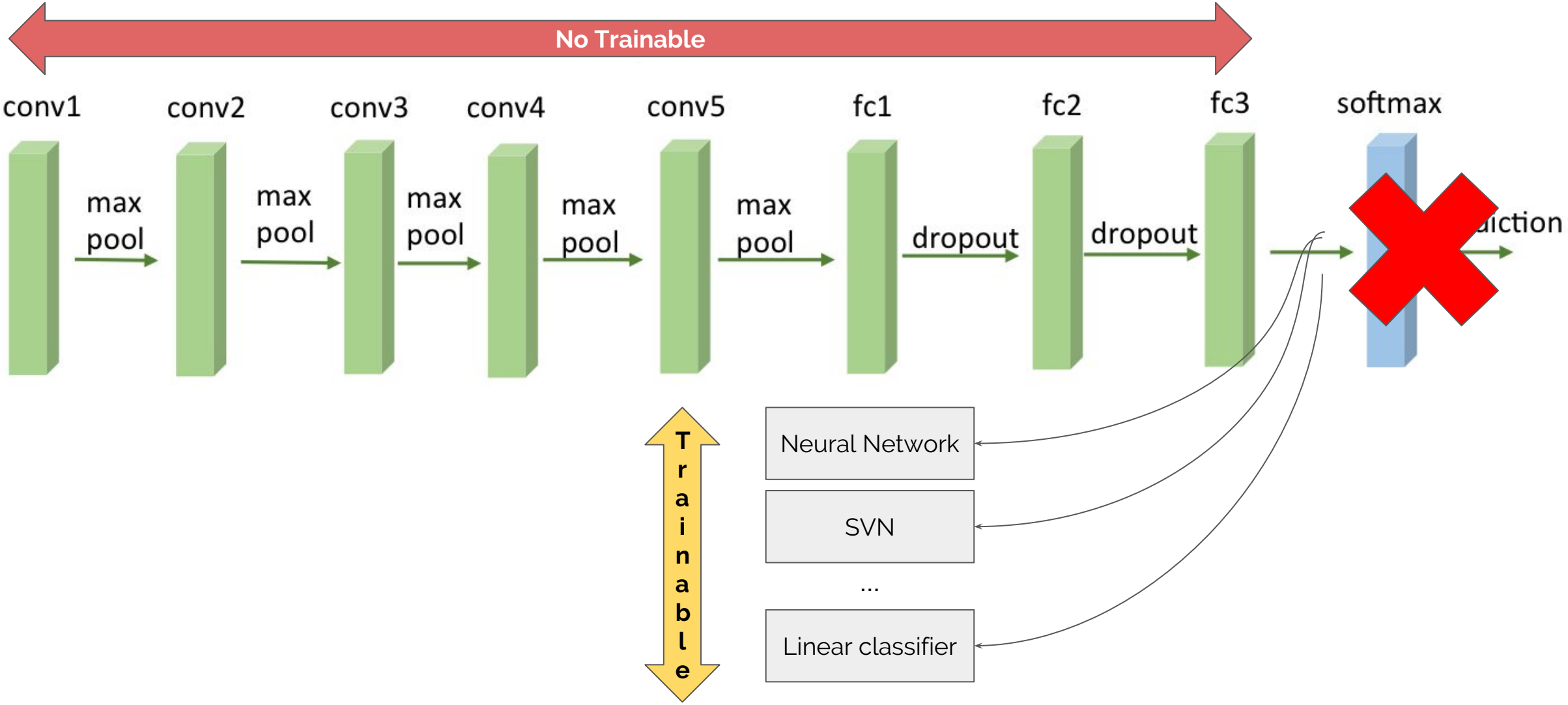


- Weights and biases initialized with trained values

Case 2: Feature extractor



Case 2: Feature extractor



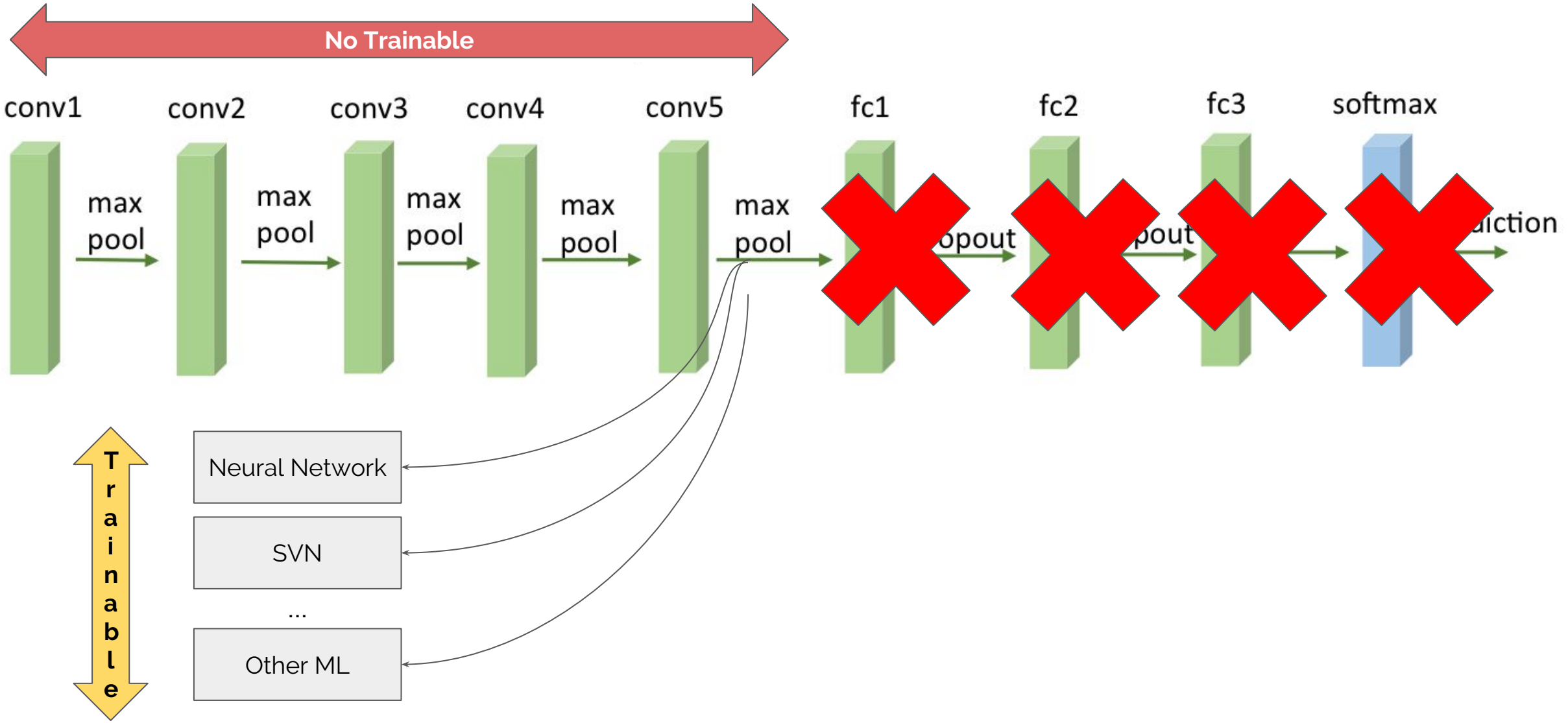
Case 2: Feature extractor

Extract features with VGG16

```
from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
import numpy as np

model = VGG16(weights='imagenet', include_top=False)

img_path = 'elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

features = model.predict(x)
```

Source: keras docs

Case 2: Feature extractor



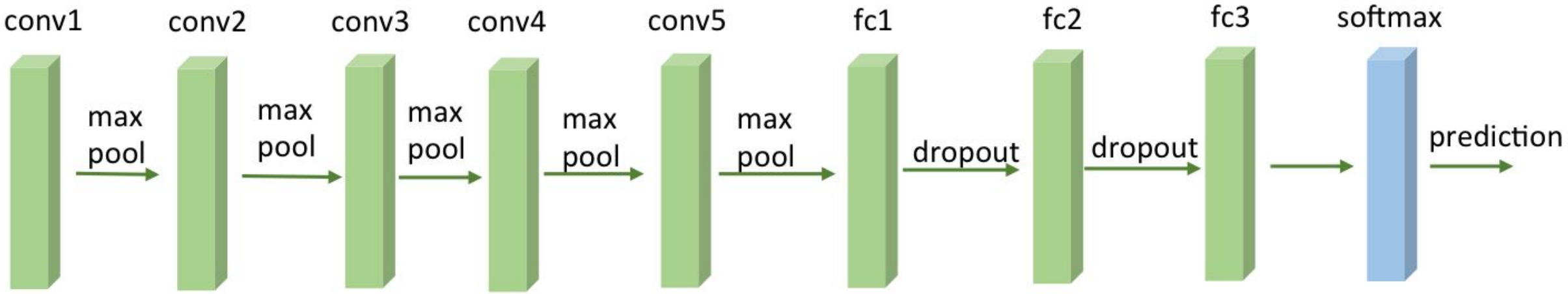
Case 2: Feature extractor

Extract features from an arbitrary intermediate layer with VGG19

```
from keras.applications.vgg19 import VGG19
from keras.preprocessing import image
from keras.applications.vgg19 import preprocess_input
from keras.models import Model
import numpy as np

base_model = VGG19(weights='imagenet')
model = Model(inputs=base_model.input, outputs=base_model.get_layer('block4_pool').output)

img_path = 'elephant.jpg'
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

block4_pool_features = model.predict(x)
```

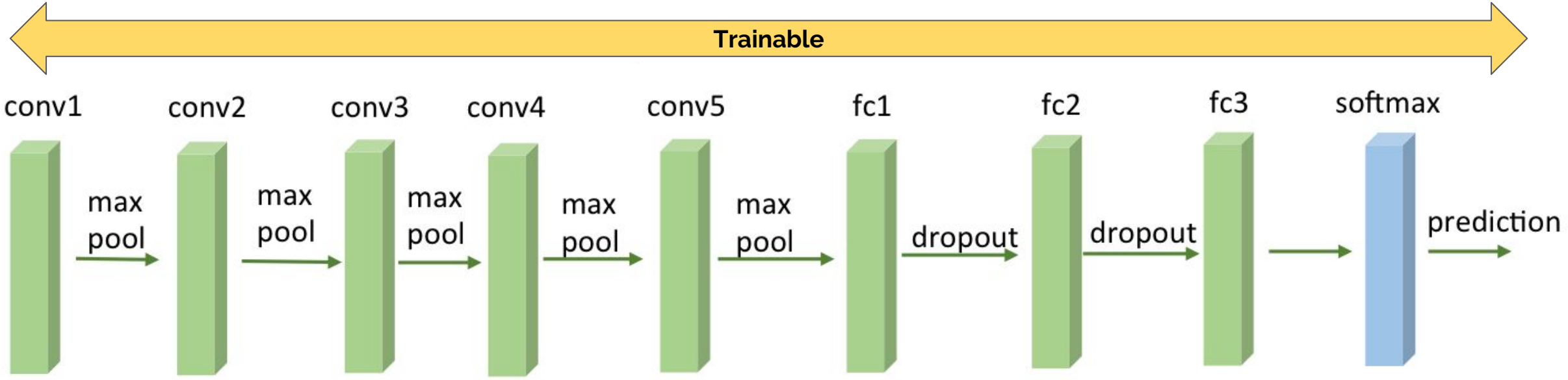
Source: keras docs

Case 3: Fine tuning



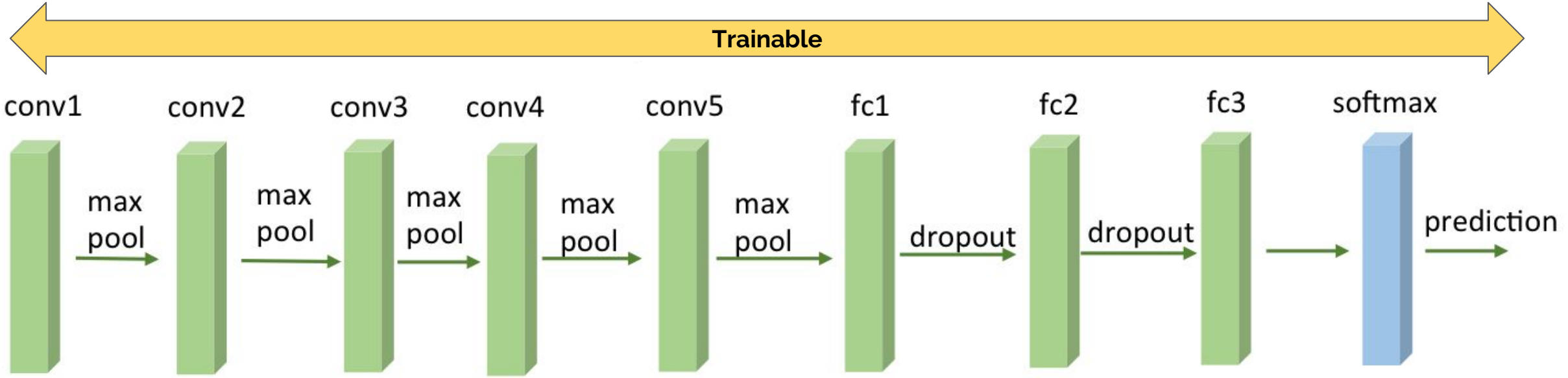
- Weights and biases initialized with trained values

Case 3a: Fine tuning



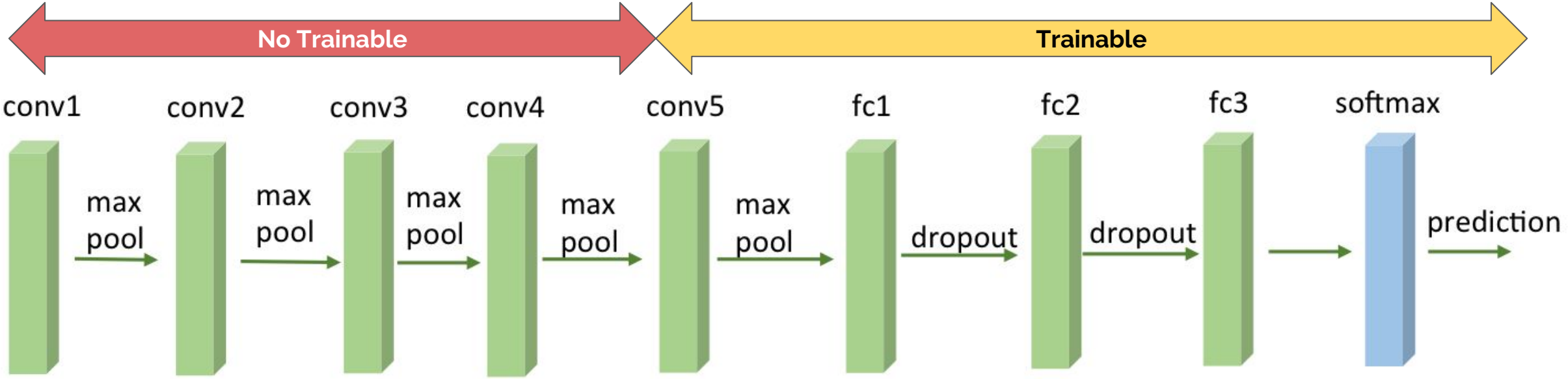
- Weights and biases initialized with trained values
- Train all the network with the new data

Case 3a: Fine tuning



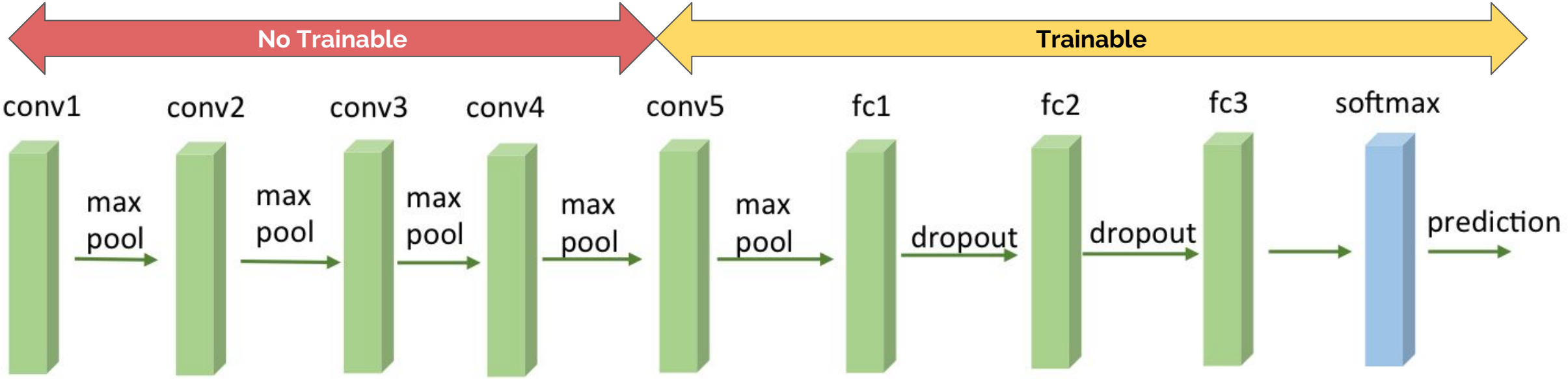
- Weights and biases initialized with trained values
- Train all the network with the new data
- Reduce the learning rate

Case 3b: Fine tuning



- Weights and biases initialized with trained values
- Train some parts of the network with the new data

Case 3b: Fine tuning



- Weights and biases initialized with trained values
- Train some parts of the network with the new data
- Reduce the learning rate

Case 3: Fine tuning

```
from keras import applications
from keras.preprocessing.image import ImageDataGenerator
from keras import optimizers
from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dense

# path to the model weights files.
weights_path = '../keras/examples/vgg16_weights.h5'
top_model_weights_path = 'fc_model.h5'
# dimensions of our images.
img_width, img_height = 150, 150

train_data_dir = 'cats_and_dogs_small/train'
validation_data_dir = 'cats_and_dogs_small/validation'
nb_train_samples = 2000
nb_validation_samples = 800
epochs = 50
batch_size = 16

# build the VGG16 network
model = applications.VGG16(weights='imagenet', include_top=False)
print('Model loaded.')
```

```
# build a classifier model to put on top of the convolutional model
top_model = Sequential()
top_model.add(Flatten(input_shape=model.output_shape[1:]))
top_model.add(Dense(256, activation='relu'))
top_model.add(Dropout(0.5))
top_model.add(Dense(1, activation='sigmoid'))

# note that it is necessary to start with a fully-trained
# classifier, including the top classifier,
# in order to successfully do fine-tuning
top_model.load_weights(top_model_weights_path)

# add the model on top of the convolutional base
model.add(top_model)

# set the first 25 layers (up to the last conv block)
# to non-trainable (weights will not be updated)
for layer in model.layers[:25]:
    layer.trainable = False
```

Source: keras github

Case 3: Fine tuning

```
from keras import applications
from keras.preprocessing.image import ImageDataGenerator
from keras import optimizers
from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dense

# path to the model weights files.
weights_path = '../keras/examples/vgg16_weights.h5'
top_model_weights_path = 'fc_model.h5'
# dimensions of our images.
img_width, img_height = 150, 150

train_data_dir = 'cats_and_dogs_small/train'
validation_data_dir = 'cats_and_dogs_small/validation'
nb_train_samples = 2000
nb_validation_samples = 800
epochs = 50
batch_size = 16

# build the VGG16 network
model = applications.VGG16(weights='imagenet', include_top=False)
print('Model loaded.')
```

```
# build a classifier model to put on top of the convolutional model
top_model = Sequential()
top_model.add(Flatten(input_shape=model.output_shape[1:]))
top_model.add(Dense(256, activation='relu'))
top_model.add(Dropout(0.5))
top_model.add(Dense(1, activation='sigmoid'))

# note that it is necessary to start with a fully-trained
# classifier, including the top classifier,
# in order to successfully do fine-tuning
top_model.load_weights(top_model_weights_path)

# add the model on top of the convolutional base
model.add(top_model)

# set the first 25 layers (up to the last conv block)
# to non-trainable (weights will not be updated)
for layer in model.layers[:25]:
    layer.trainable = False
```

Source: keras github

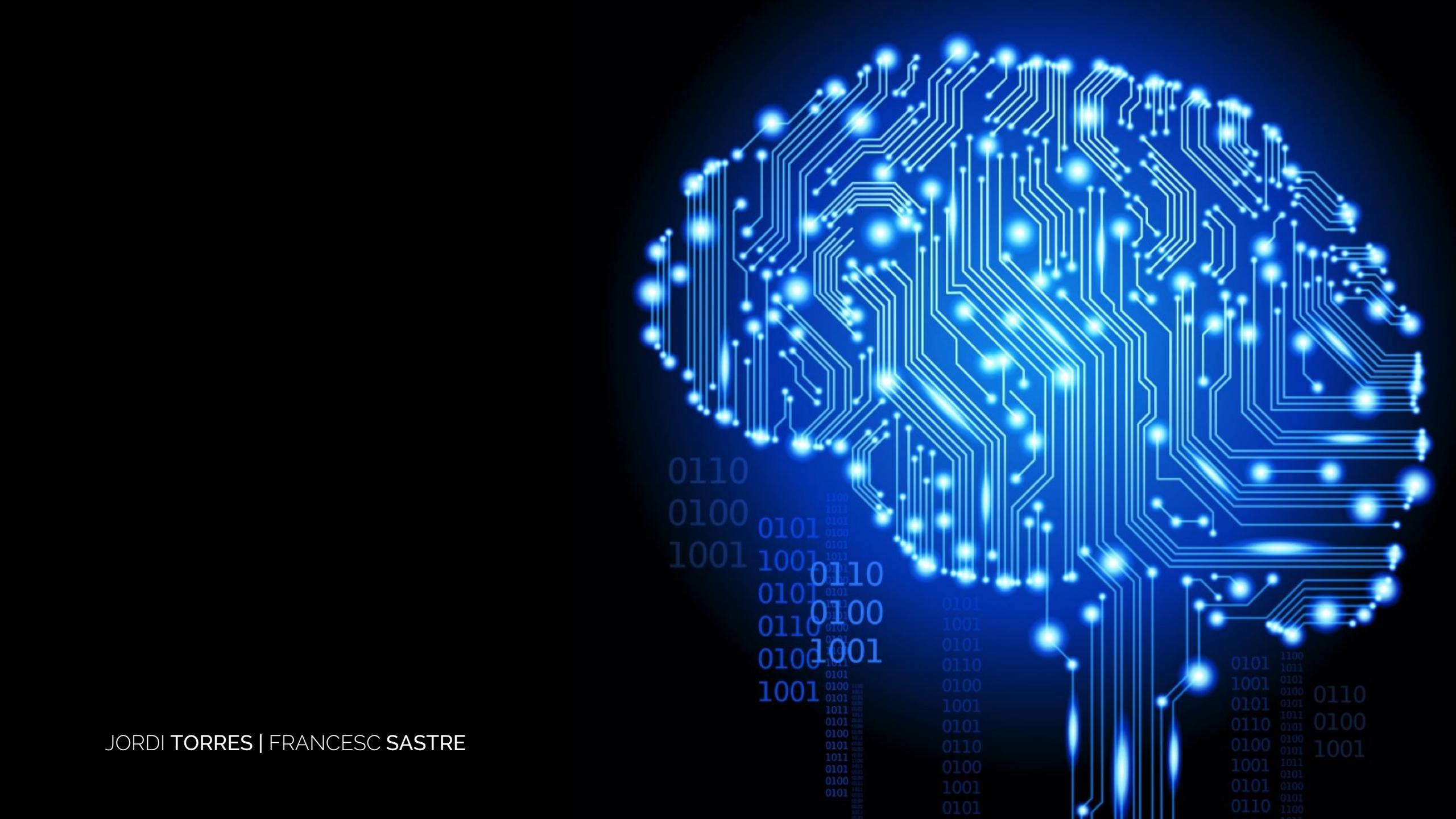
Transfer Learning

	Similar Data	**Different Data**
Small Data	Feature extractor (All layers) + Other classifier	Feature extractor (First layers) + Other classifier
Big Data	Fine tuning	From scratch - No transfer learning

Source: Stanford CS231n

Transfer Learning

- **Caffe**
  - Model Zoo - A platform for third party contributors to share pre-trained caffe models
- **Keras**
  - Keras Application - Implementation of popular state-of-the-art Convnet models like VGG16/19, googleNetNet, Inception V3, and ResNet
- **TensorFlow**
  - VGG16
  - Inception V3
  - ResNet
- **Torch**
  - LoadCaffe - Maintains a list of popular models like AlexNet and VGG .Weights ported from Caffe
- **MxNet**
  - MxNet Model Gallery - Maintains pre-trained Inception-BN (V2) and Inception V3.



JORDI **TORRES** | FRANCESC **SASTRE**