# Clasificación de imágenes con TensorFlow

...

Alejandro Solano - Málaga Python

MÁLAGA
PYTHON

TensorFlow


TensorFlow

gato

input

target

??

gato

TensorFlow

input

target

sin

\+

X

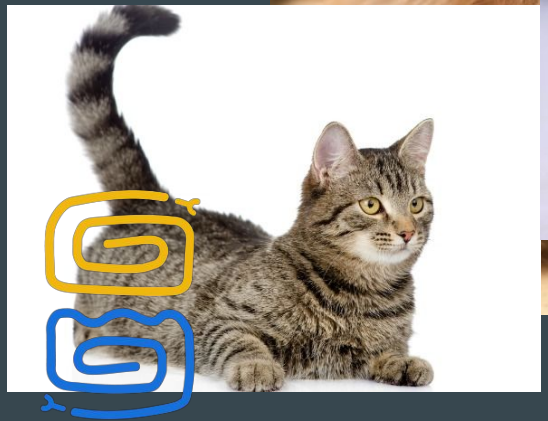
log

exp

gato

TensorFlow

input



target

sin

\+

x

log

exp

gato

TensorFlow

# Deep Learning

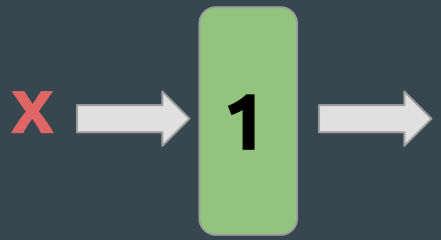
MODEL

**gato**

TensorFlow

# Red neuronal

$$x \cdot W_1 + b_1 = y$$

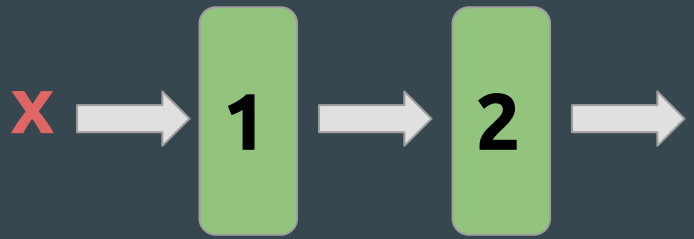



TensorFlow

# Red neuronal

$$x \cdot W_1 + b_1 = y$$

$$(x \cdot W_1 + b_1) \cdot W_2 + b_2 = y$$



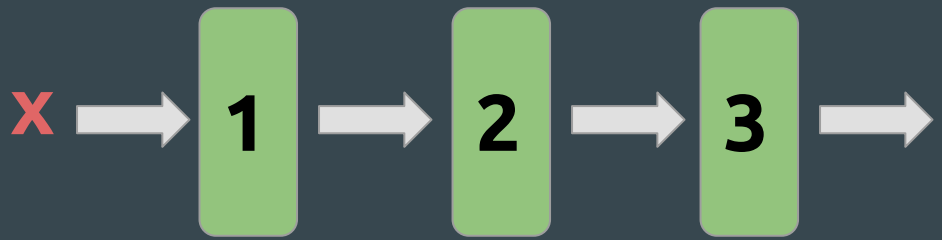

TensorFlow

# Red neuronal

$$x \cdot W_1 + b_1 = y$$

$$(x \cdot W_1 + b_1) \cdot W_2 + b_2 = y$$

$$((x \cdot W_1 + b_1) \cdot W_2 + b_2) \cdot W_3 + b_3 = y$$



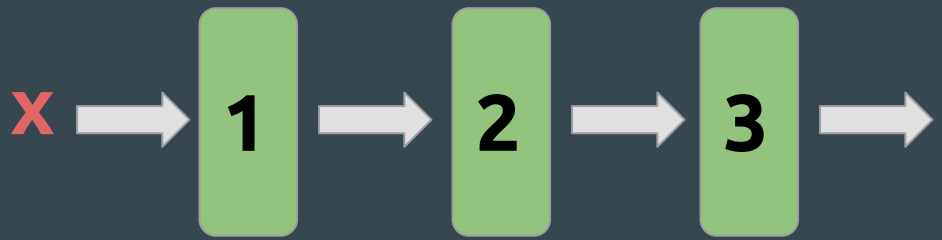

TensorFlow

# Red neuronal

$$x \cdot W_1 + b_1 = y$$

$$(x \cdot W_1 + b_1) \cdot W_2 + b_2 = y$$

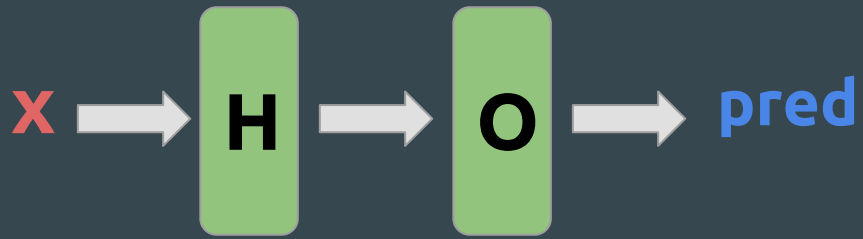
$$\tanh(\sigma(x \cdot W_1 + b_1) \cdot W_2 + b_2) \cdot W_3 + b_3 = y$$




TensorFlow

# Redes neuronales: intuición

Las primeras capas extraen información más básica, y las siguientes trabajan a partir de esa información.

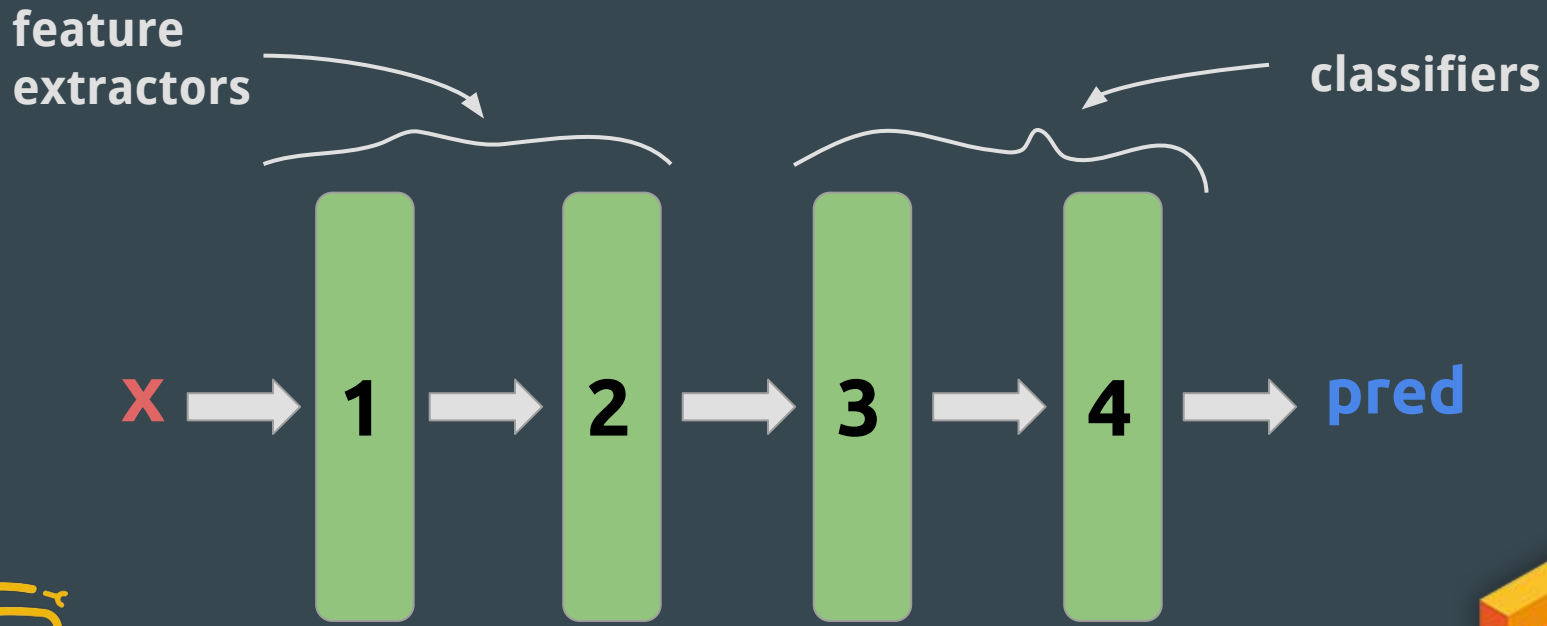



hace la suma

clasifica la suma

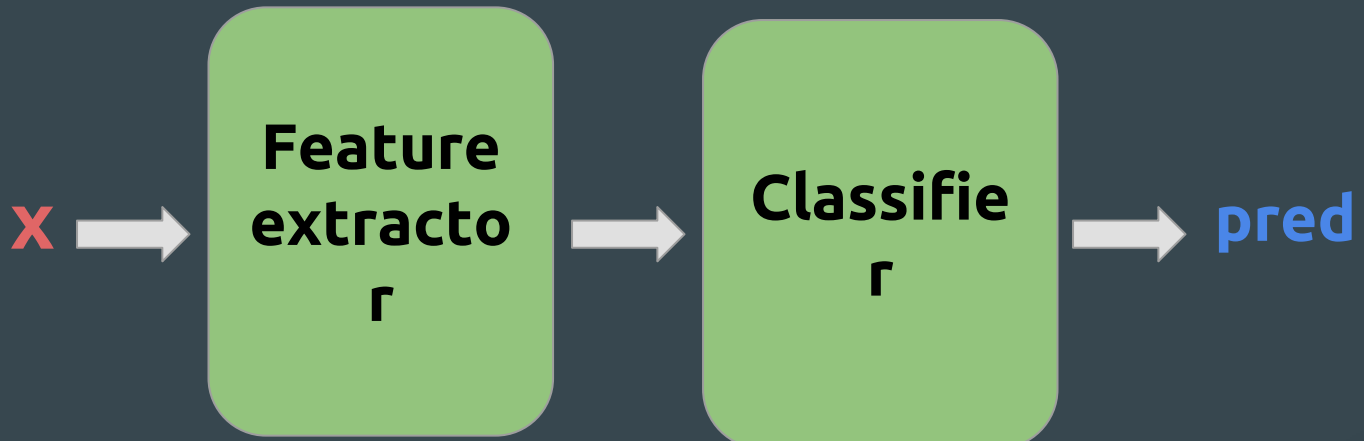
TensorFlow

# Redes neuronales: intuición




TensorFlow

# Redes neuronales: intuición




# Redes neuronales: intuición

Feature extractor

Classifier

[ 0.175,

0.825 ]

TensorFlow

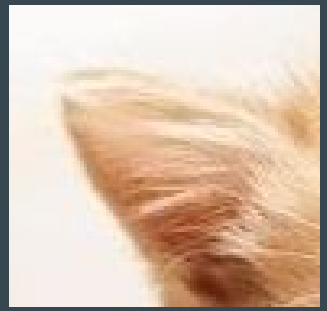
# Image Classification

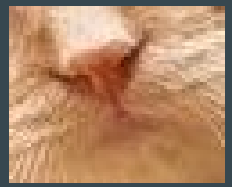
# Features: intuition

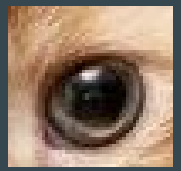
¿Qué hace que un gato sea un gato?

# Features: intuition

## ¿Qué hace que un gato sea un gato?

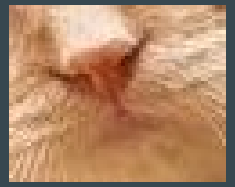


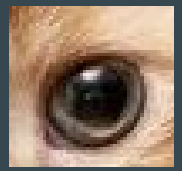


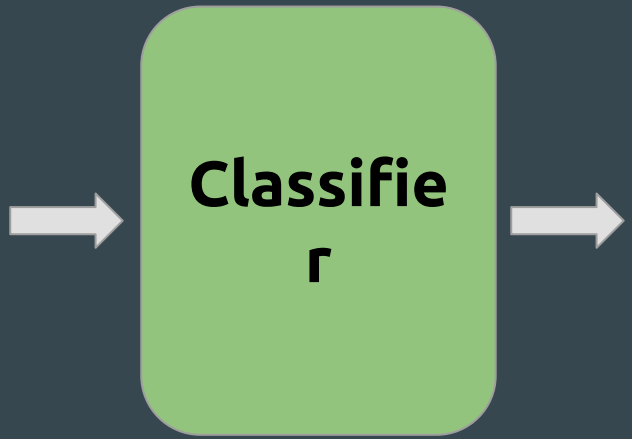


TensorFlow

# Features: intuition



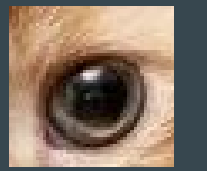





gato

TensorFlow

# Features: intuition

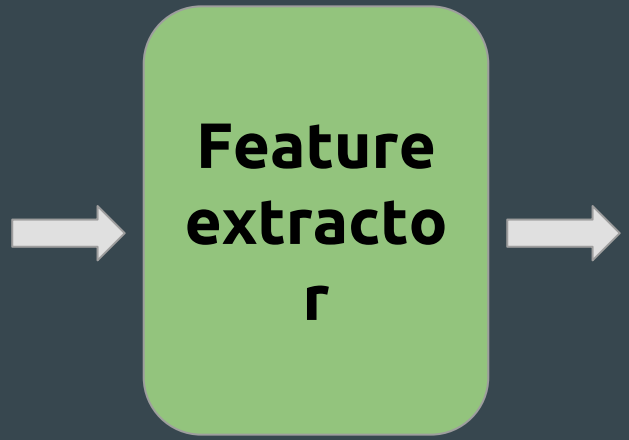


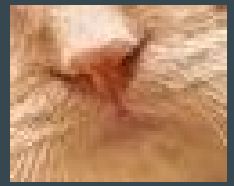
Classifie
r

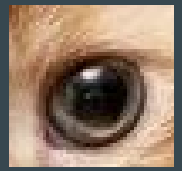
caballo

TensorFlow

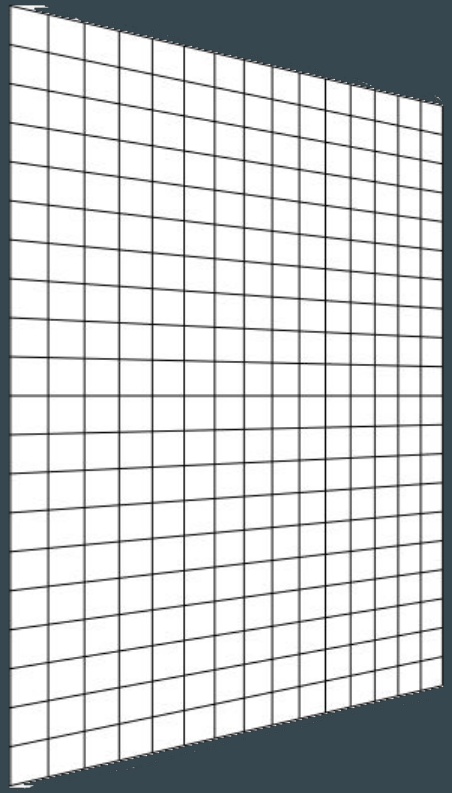
# Features: intuition





# Analyzing input data: Images

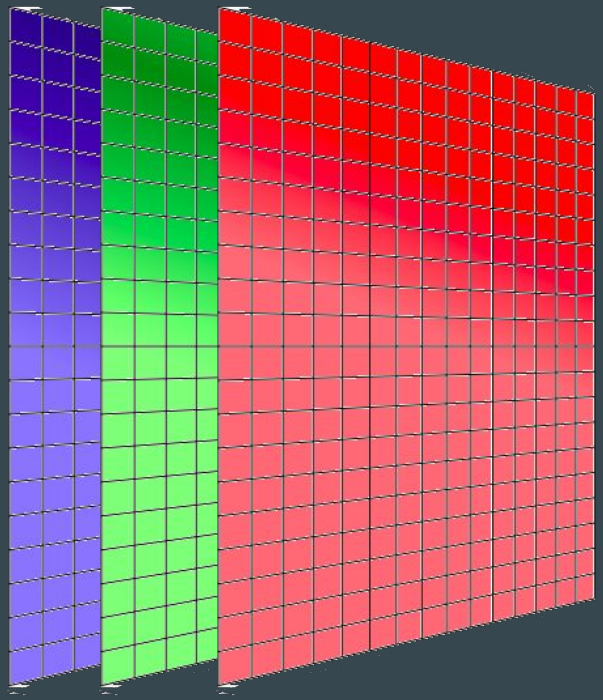


**Conjunto de píxeles**

**Cada píxel toma valores de 0 a 255**

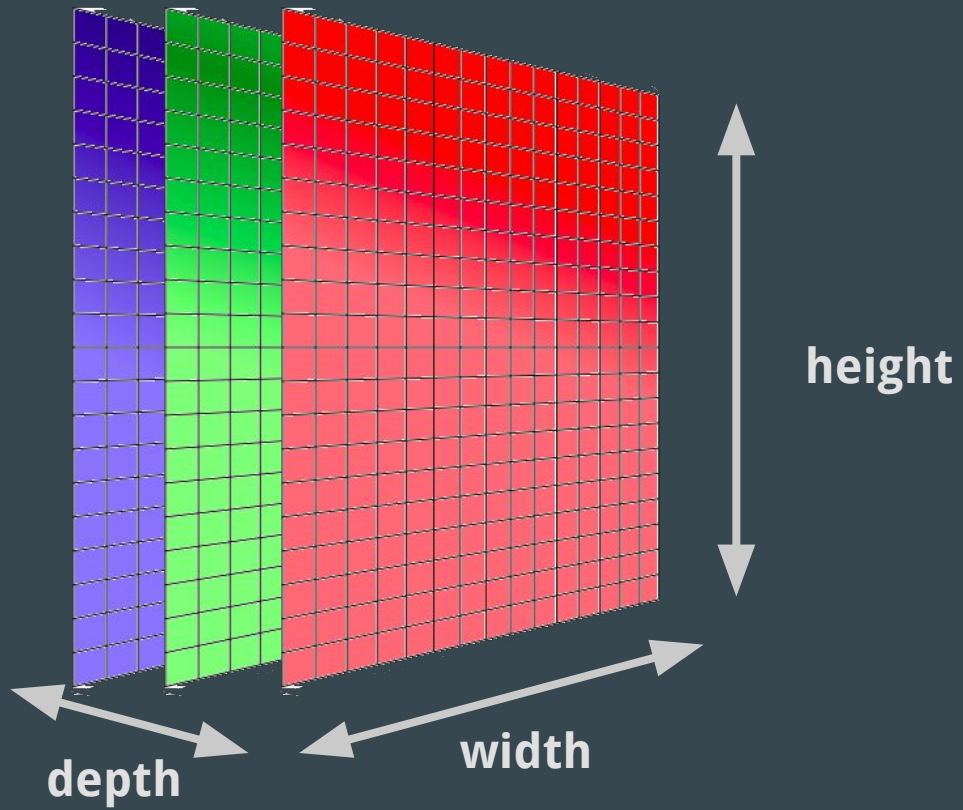
TensorFlow

# Analyzing input data: Images



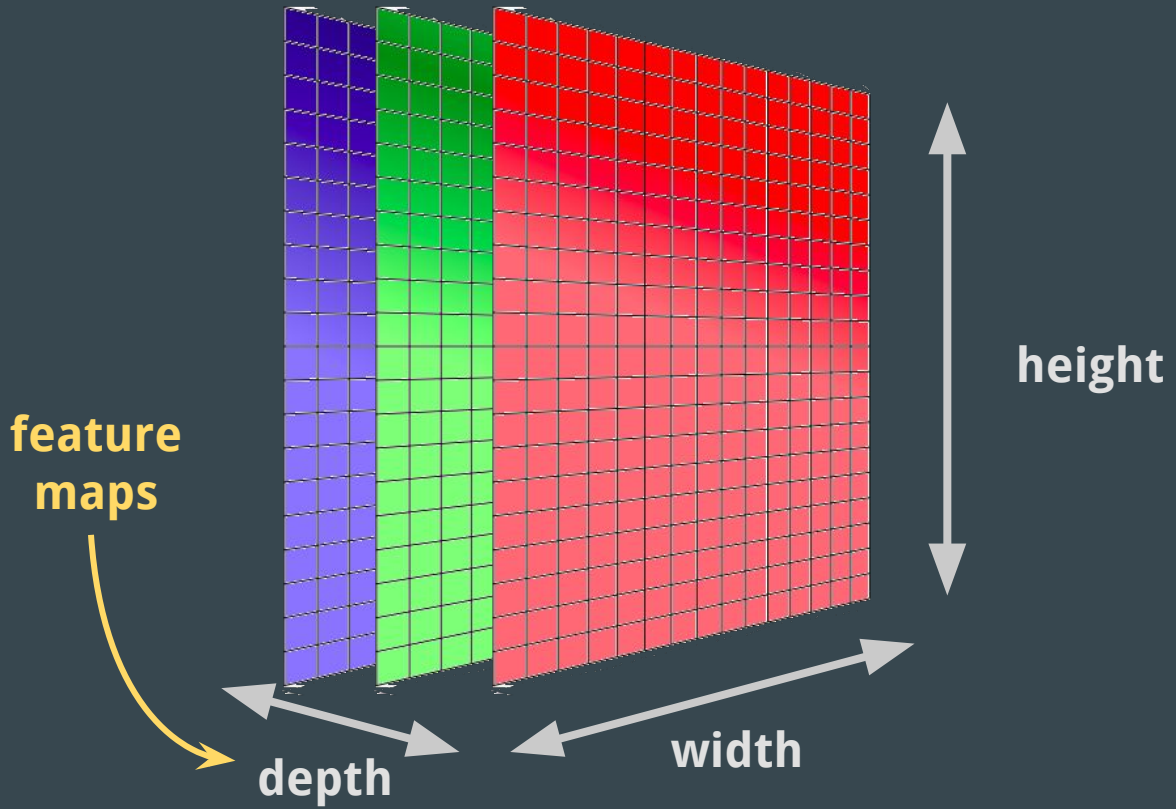
TensorFlow

# Analyzing input data: Images

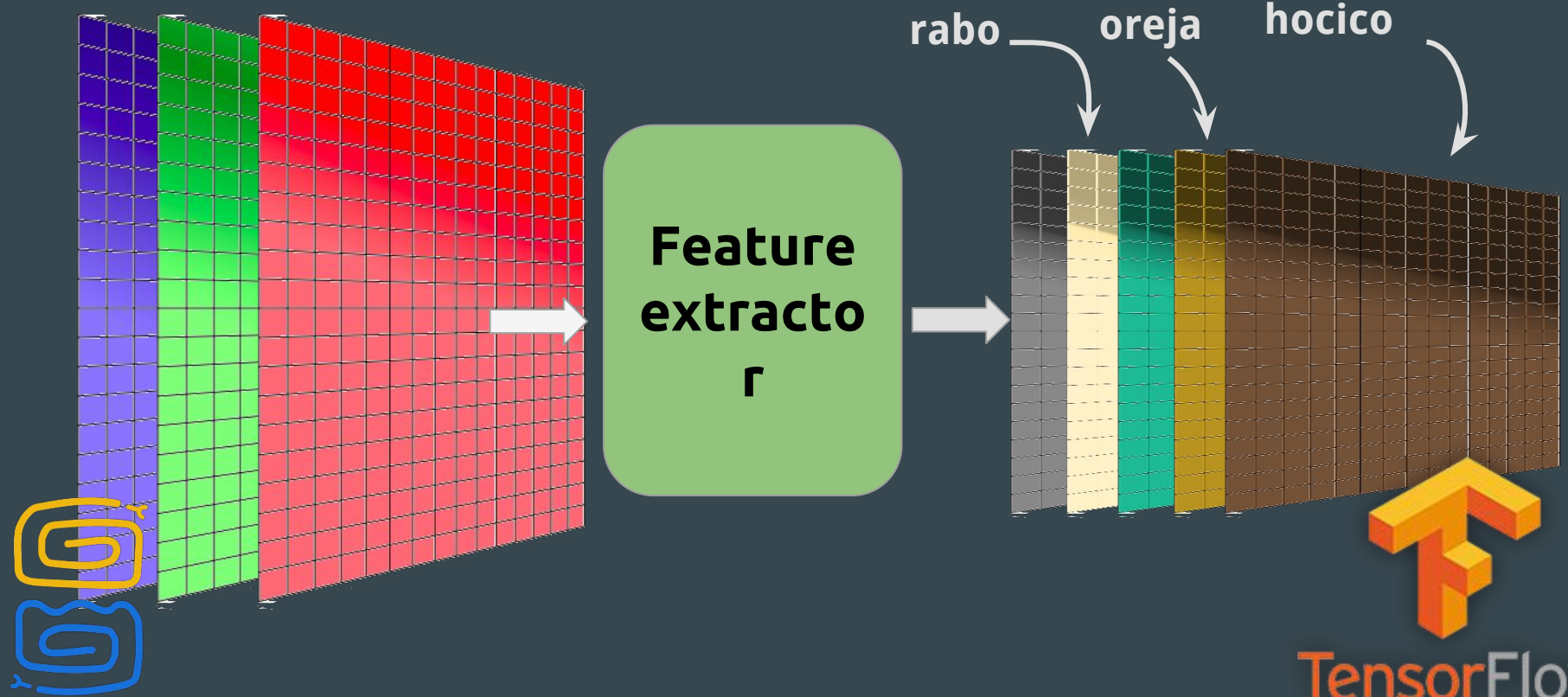



TensorFlow

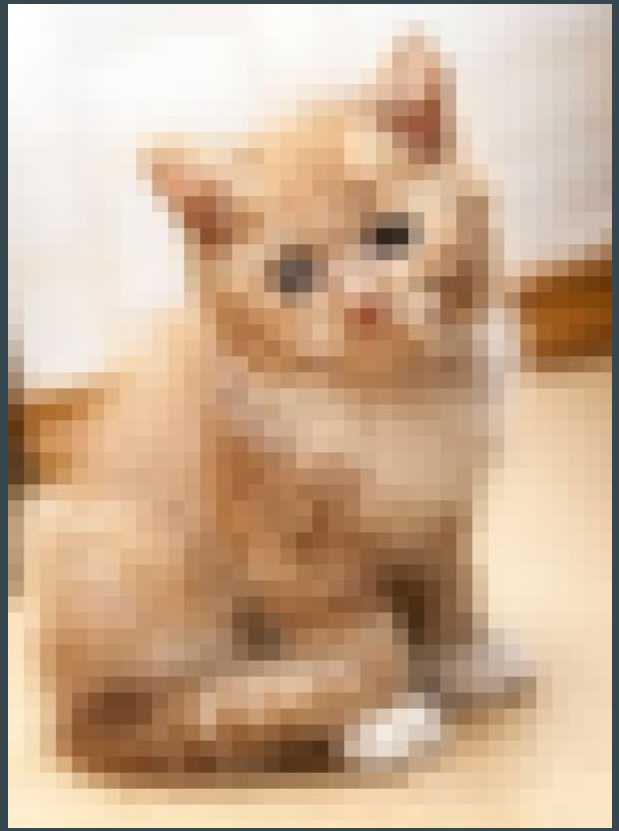
# Analyzing input data: Images



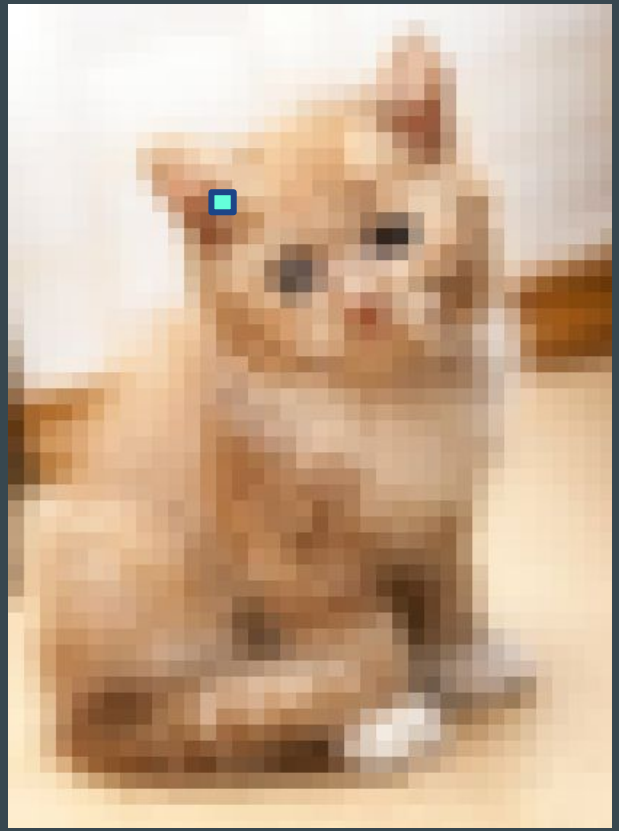

# Analyzing input data: Images



# Analyzing input data: Images

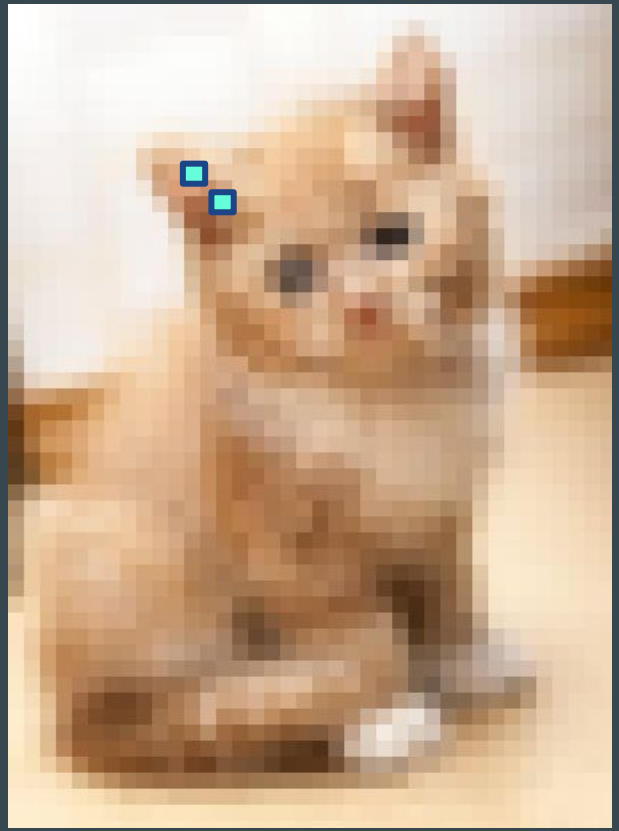


# Analyzing input data: Images

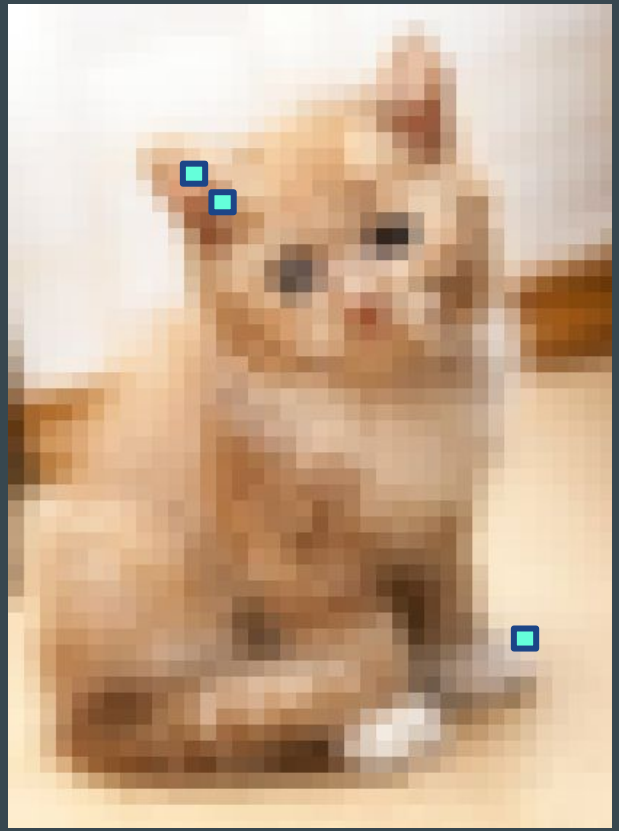


TensorFlow

# Analyzing input data: Images



# Analyzing input data: Images



# Analyzing input data: Images

# Analyzing input data: Images

# Analyzing input data: Images

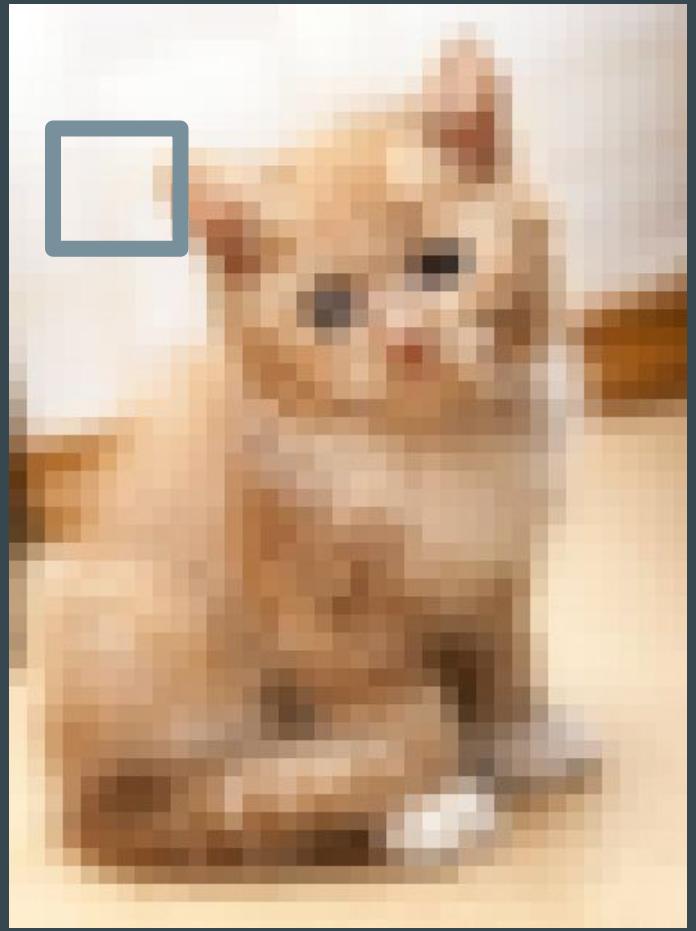




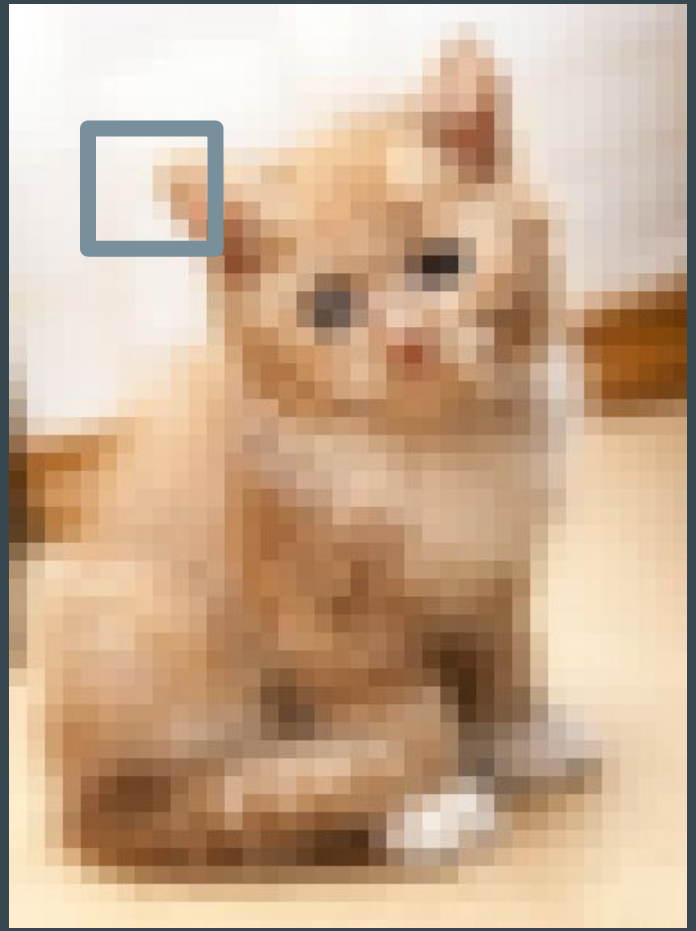
TensorFlow

# Analyzing input data: Images

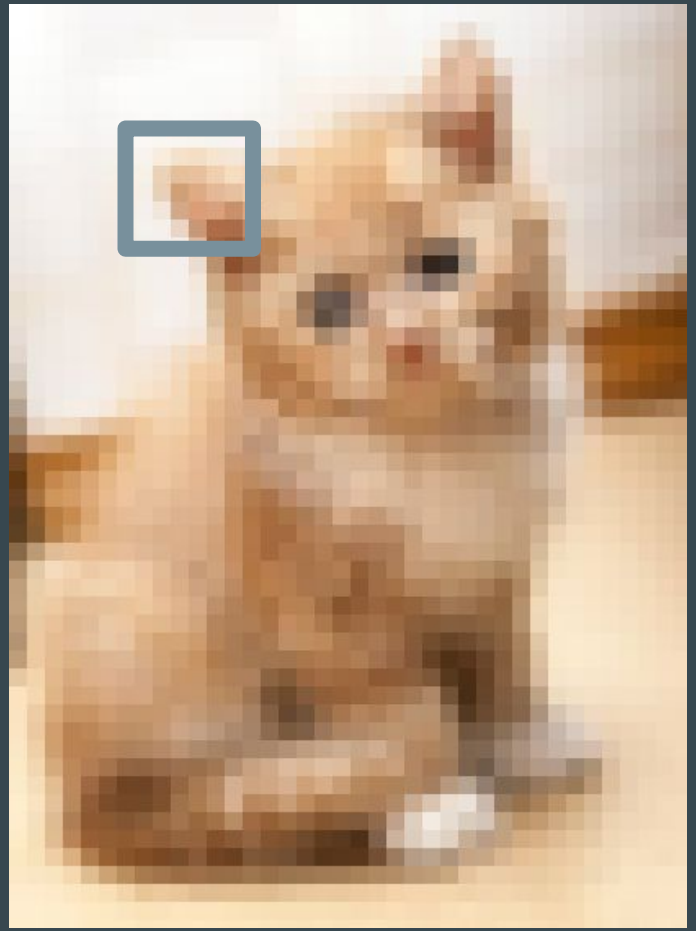
# Convolution



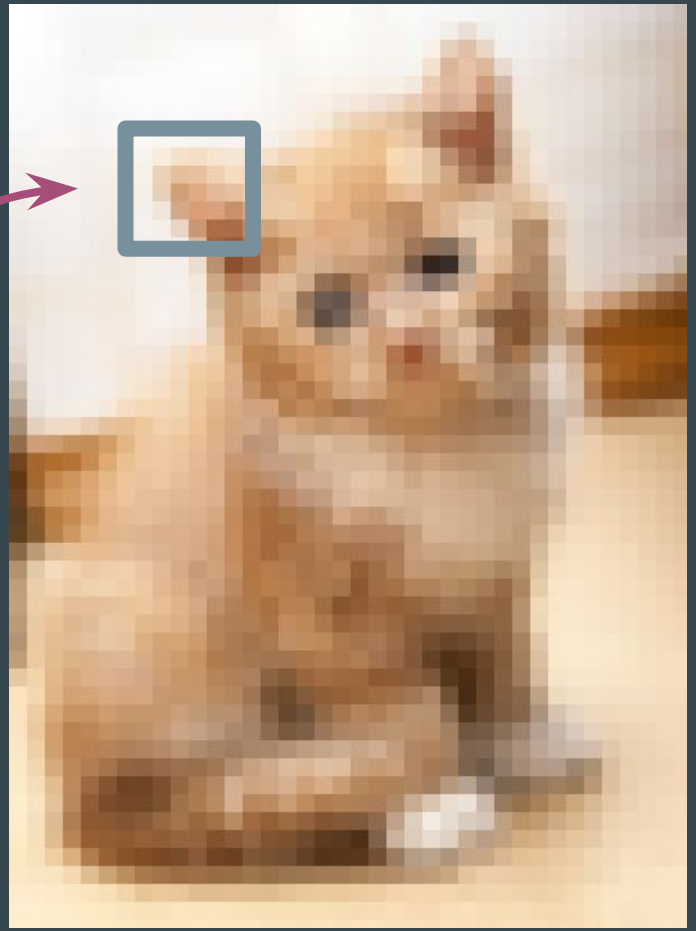
# Convolution



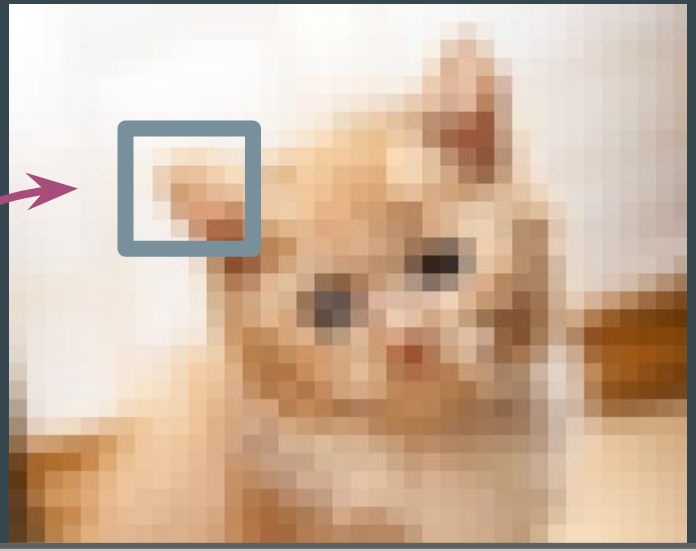
# Convolution



# Convolution



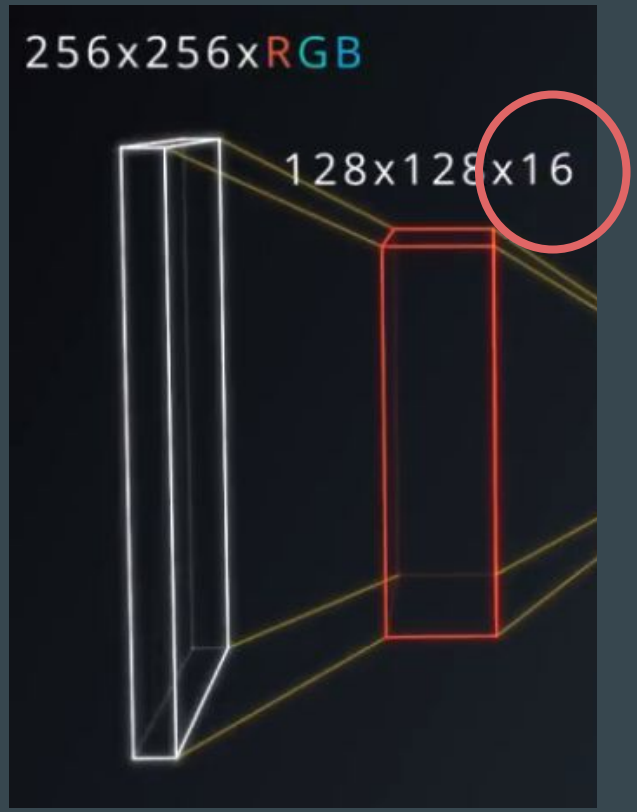

# Convolution



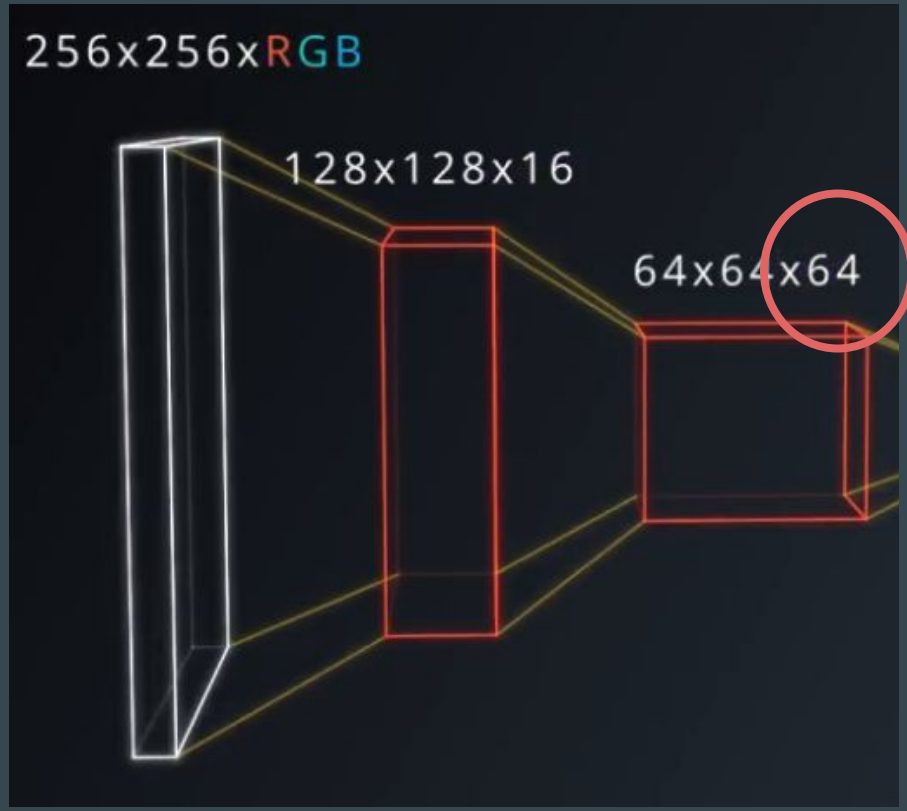

**each kernel learns to extract one feature**

TensorFlow

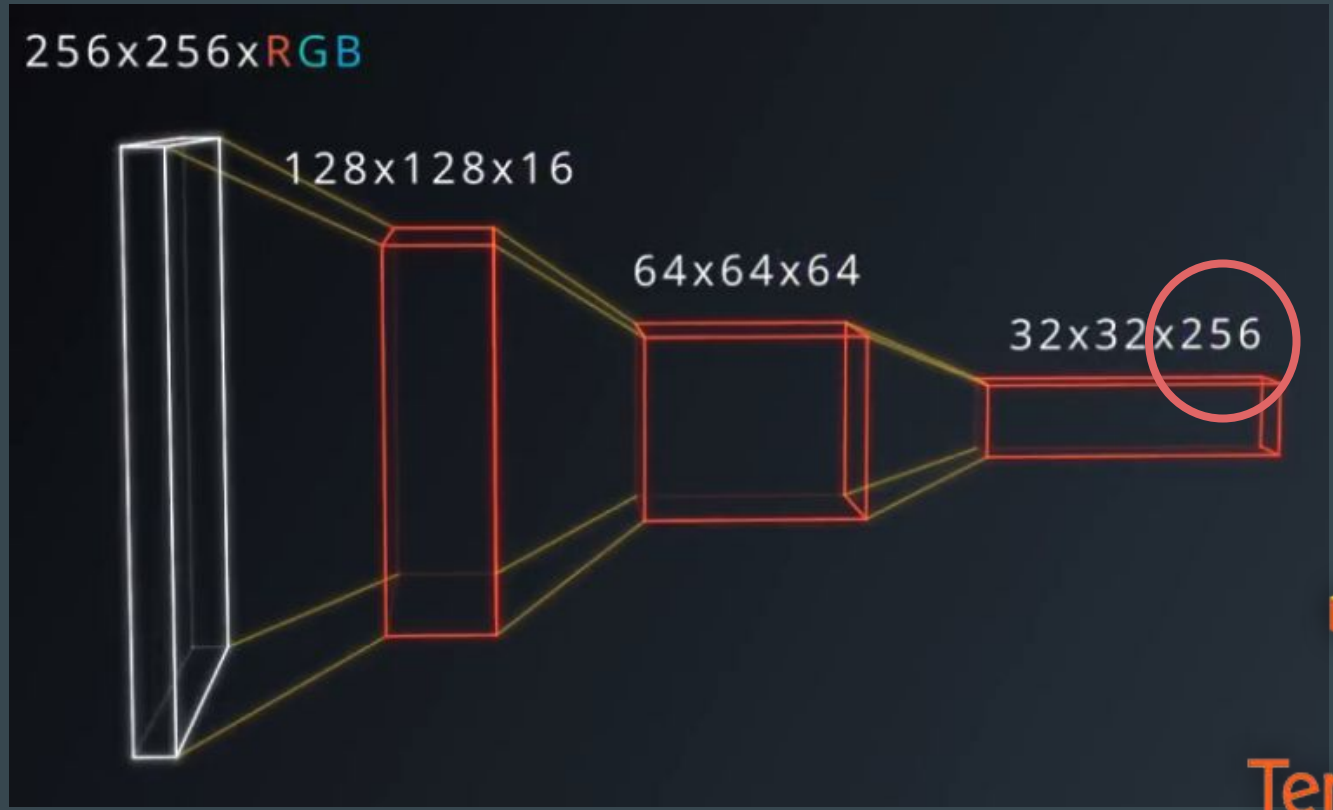
# Convolution




# Convolution

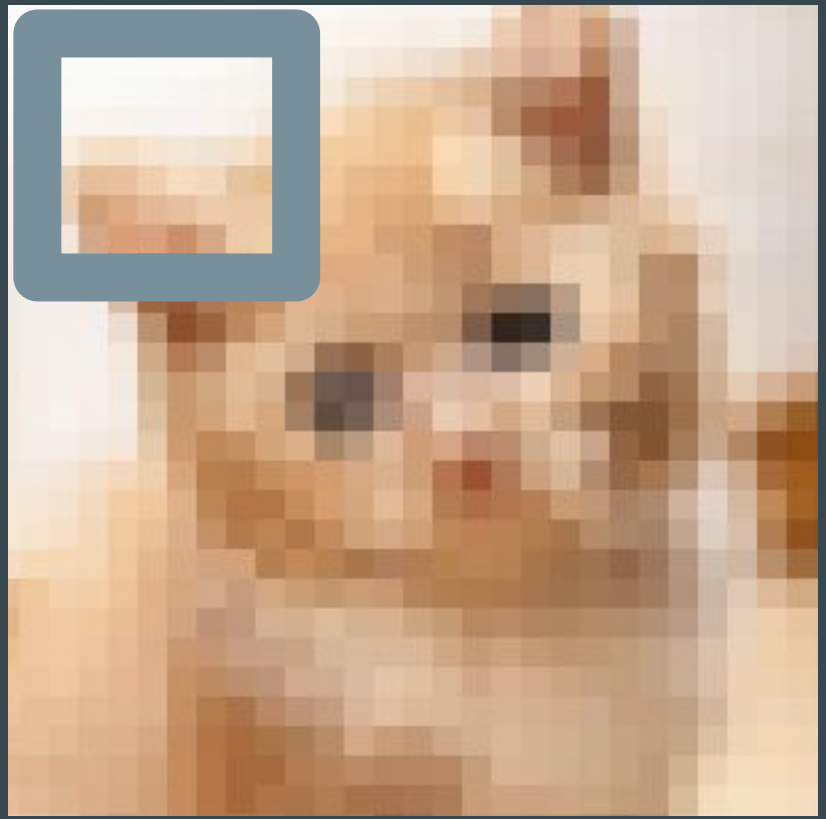



# Convolution



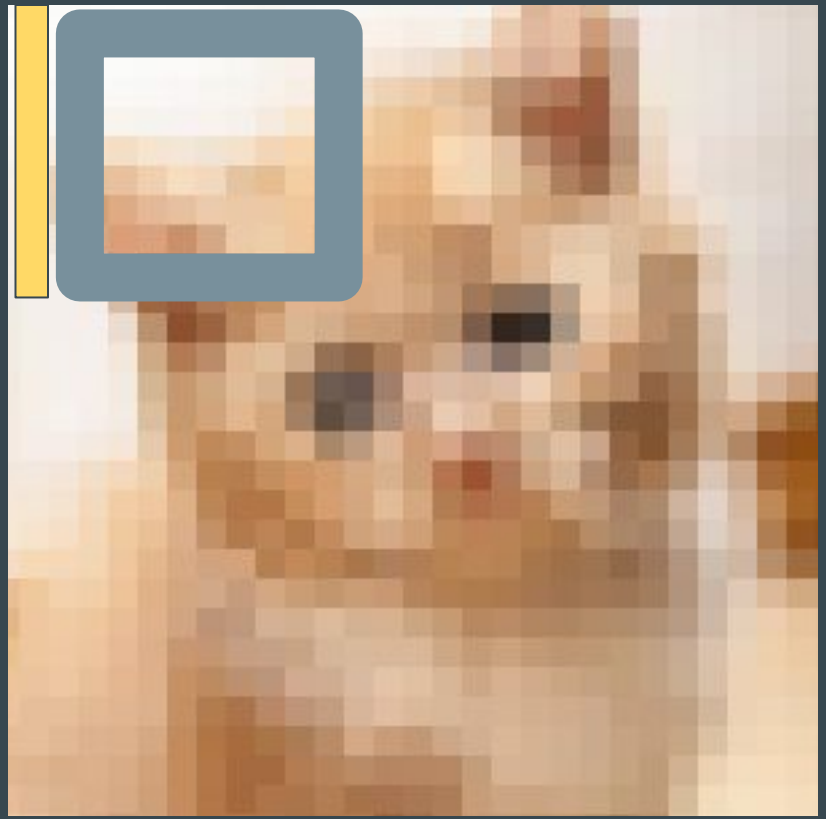

TensorFlow

# Strides and Padding



# Strides and Padding

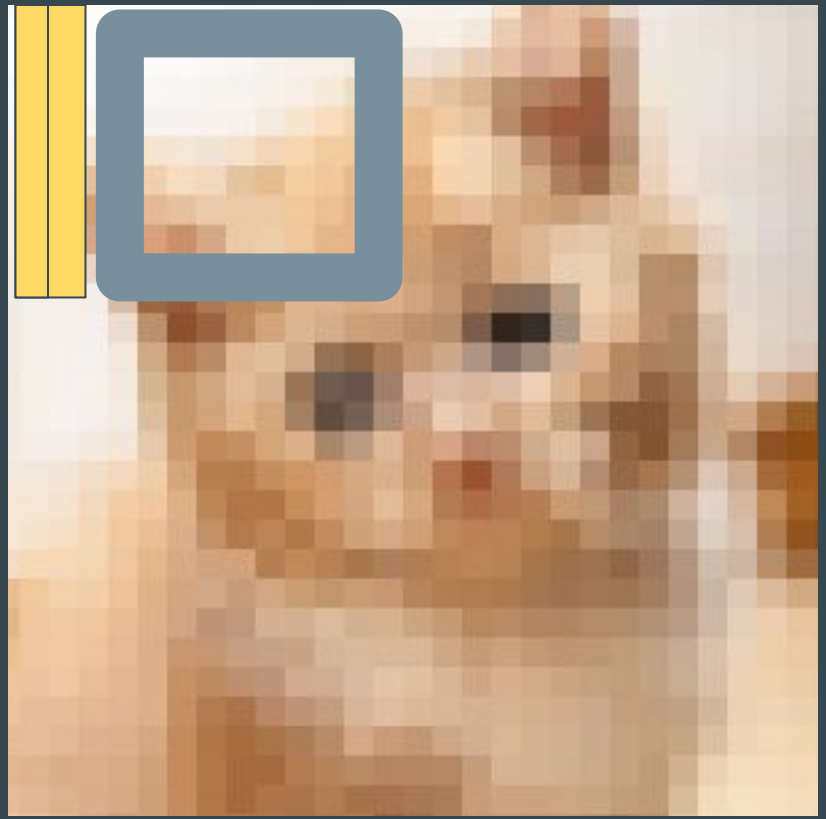
**stride 1**



TensorFlow

# Strides and Padding

**stride 1**

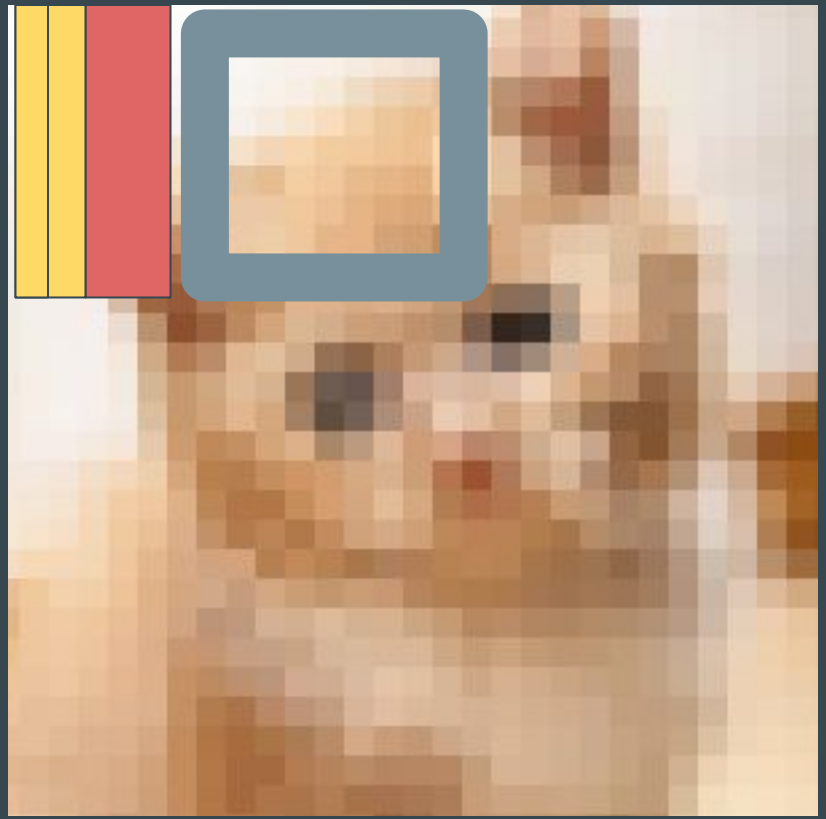


TensorFlow

# Strides and Padding

**stride 1**

**stride 2**



TensorFlow

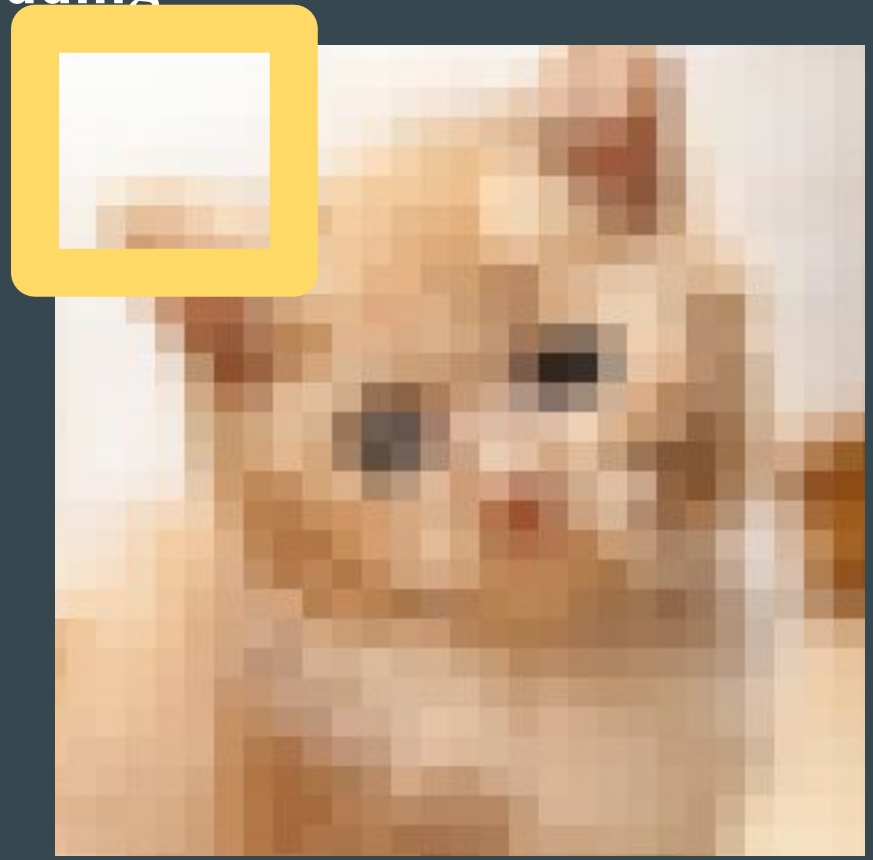
# Strides and Padding

**stride 1**

**stride 2**

TensorFlow

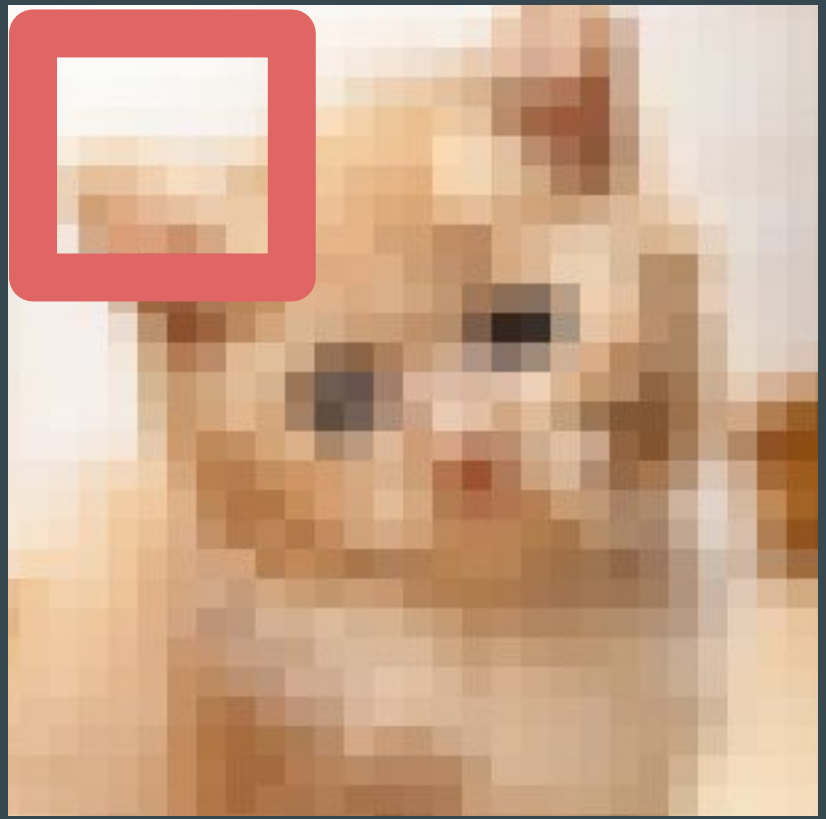
# Strides and Padding



**same padding**

TensorFlow

# Strides and Padding



same padding

valid padding

TensorFlow

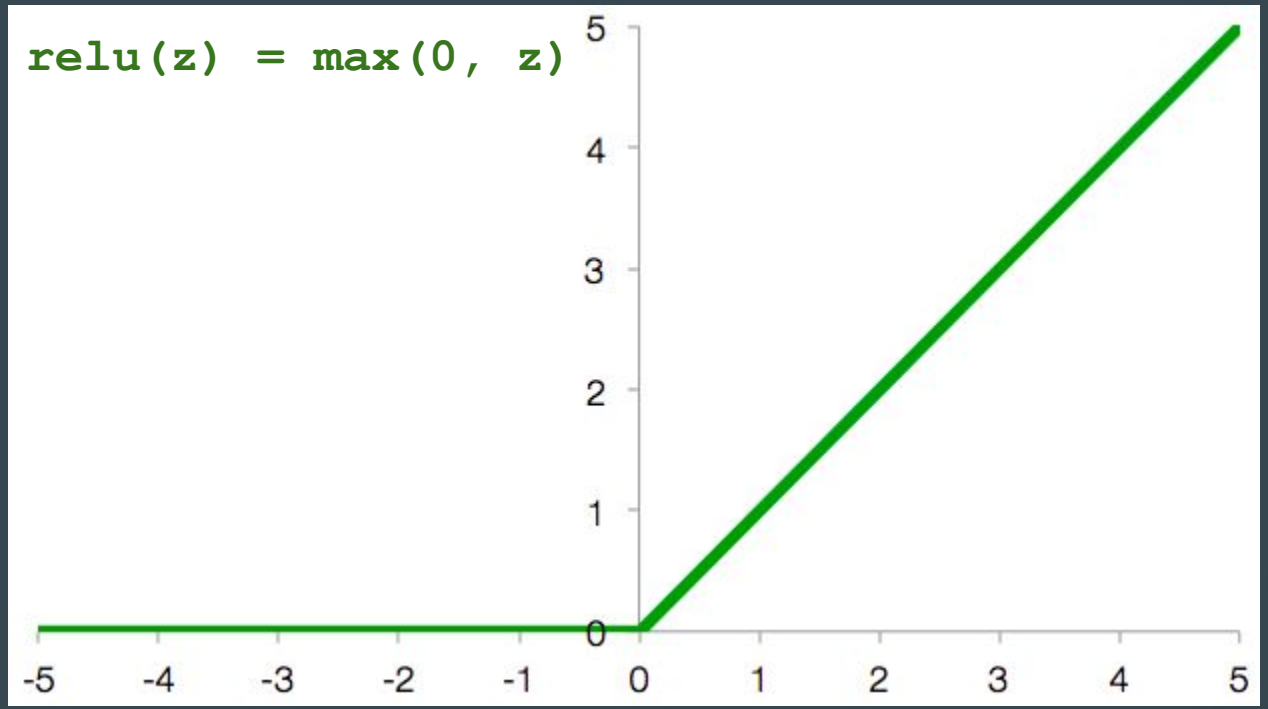
# Convolutional layer: code

```
# kernels
W = tf.Variable(tf.truncated_normal(
    [kernel_size, kernel_size, input_depth, num_kernels],
    stddev=0.05))

b = tf.Variable(tf.zeros([num_kernels]))
```

# Convolutional layer: code

```
# convolutional layer
conv = tf.nn.conv2d(x, W,
    strides=[1, strides, strides, 1],
    padding='SAME')

conv = tf.nn.bias_add(conv, b)
```

# Non-linear function: ReLu




TensorFlow

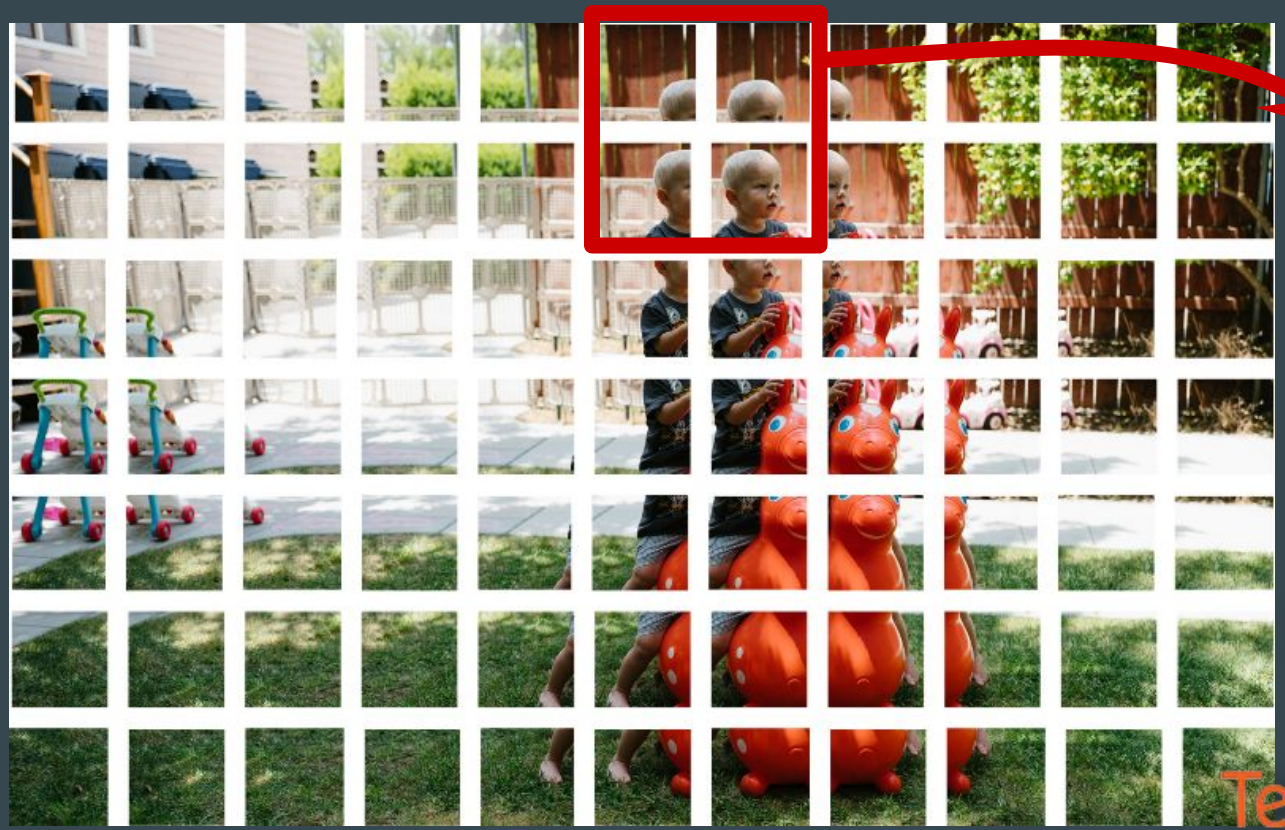
# Non-linear function: ReLu

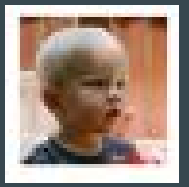
```
conv = tf.nn.relu(conv)
```

# Max-Pooling

# Max-Pooling

# Max-Pooling





# Max-Pooling

**Reduzco el 75% de los parámetros**

TensorFlow

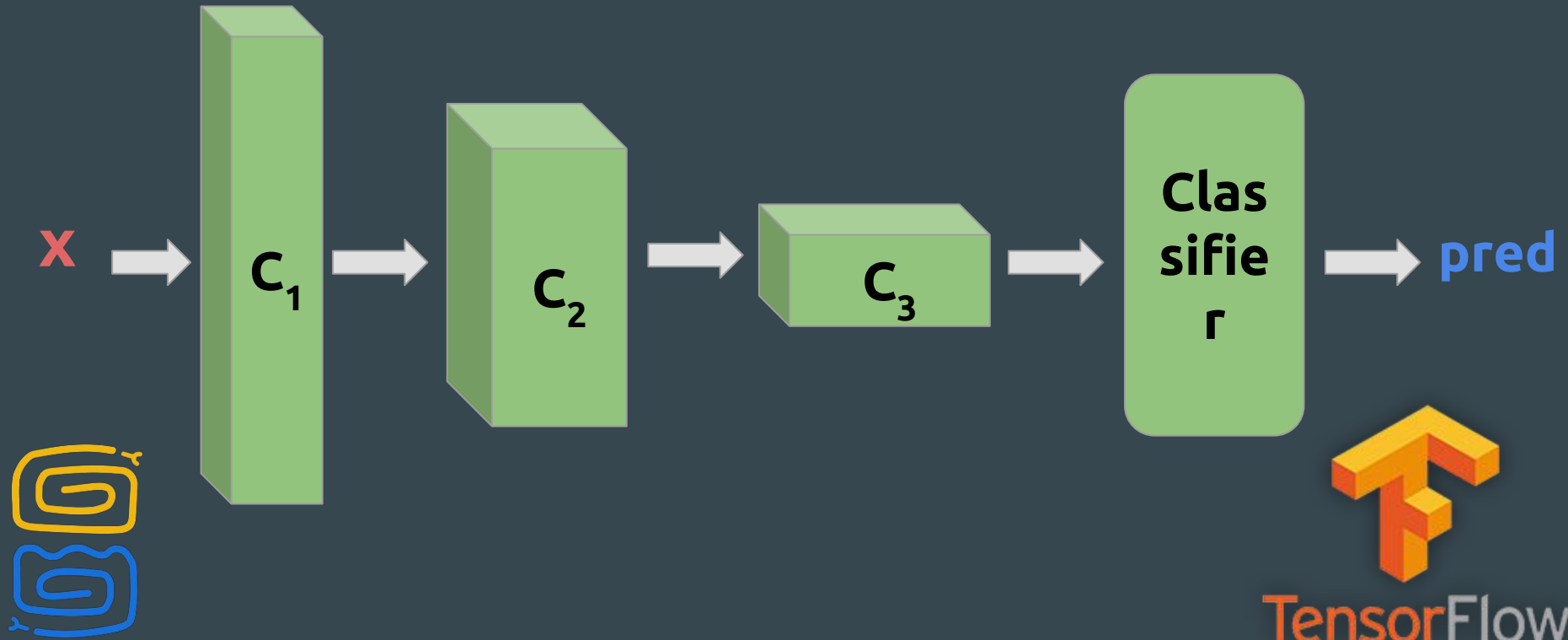
# Max-Pooling

```
x = tf.nn.max_pool(x,
        ksize=[1, k, k, 1],
        strides=[1, k, k, 1],
        padding='SAME')
```

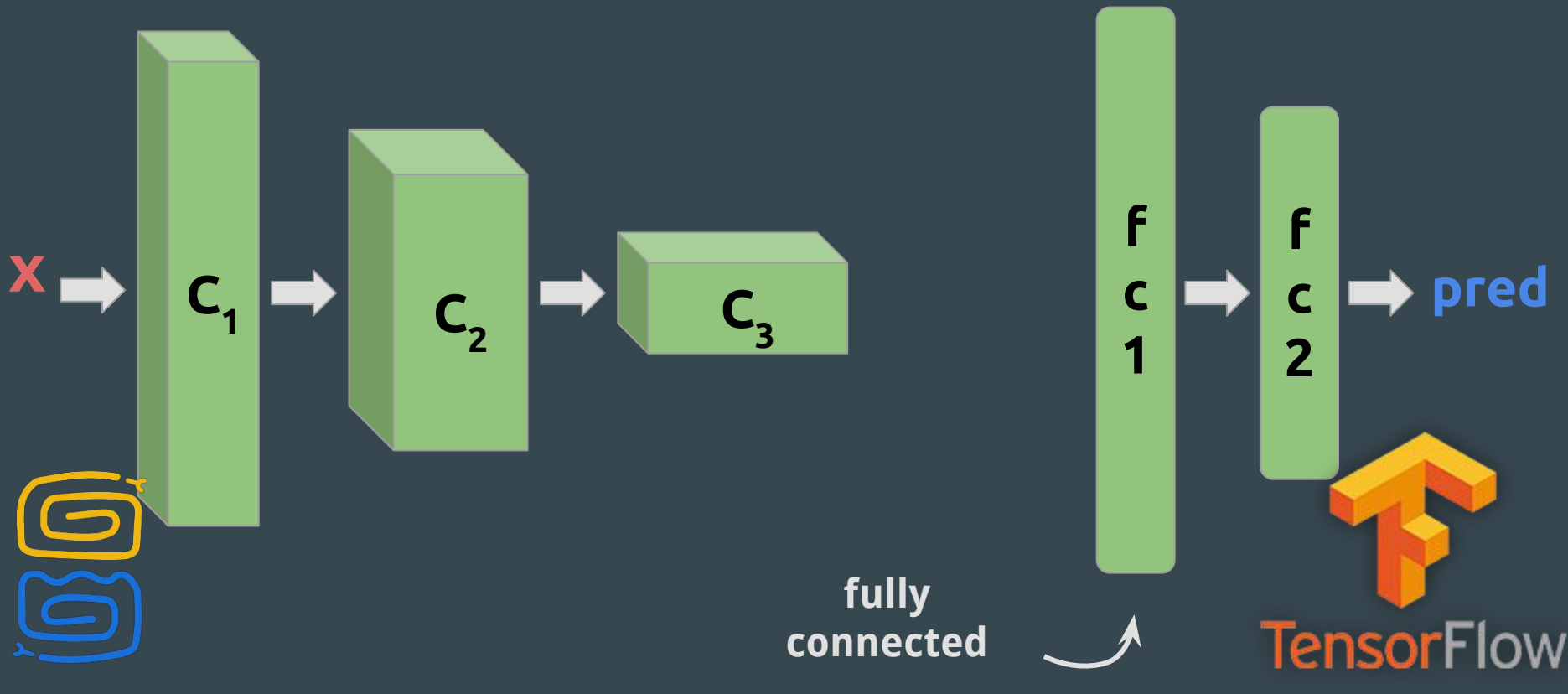
# Flatten Layer


TensorFlow

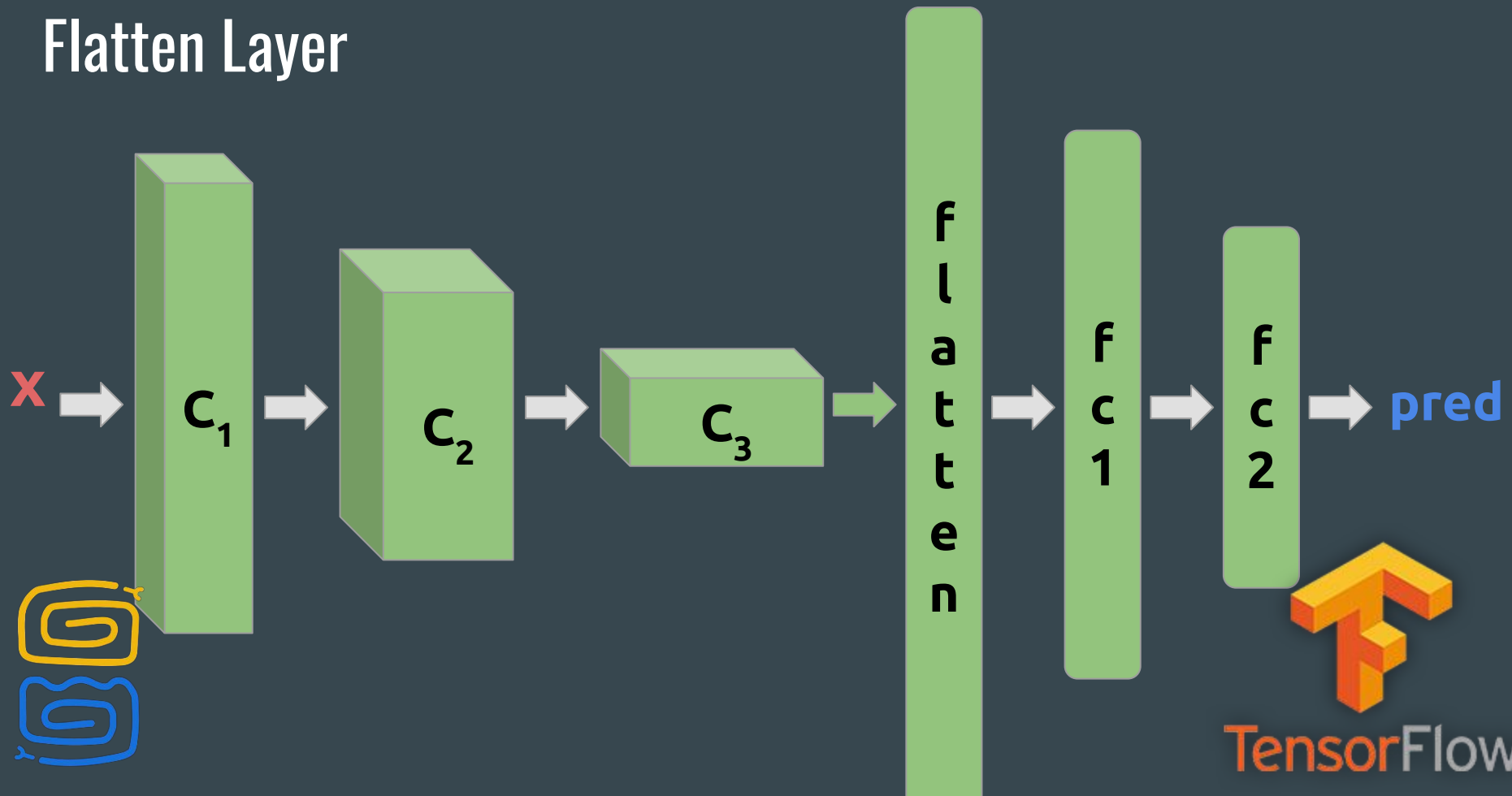
# Flatten Layer



# Flatten Layer



# Flatten Layer

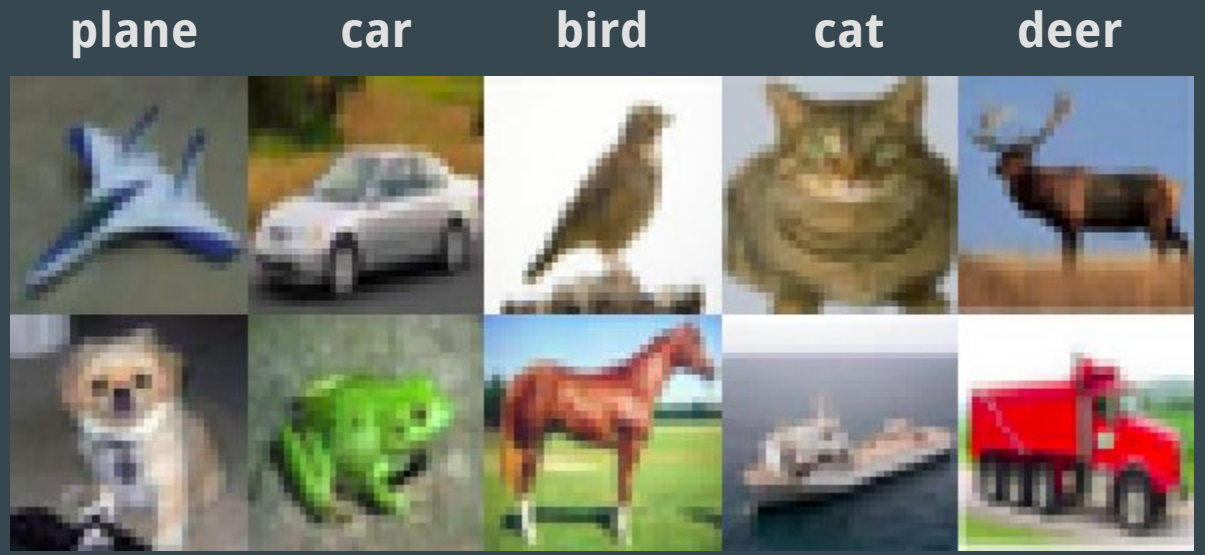


# Flatten Layer

```
x_shape = x.get_shape().as_list()

flatten_shape = x_shape[1] * x_shape[2] * x_shape[3]

flatten = tf.reshape(x, [-1, flatten_shape])
```

# CIFAR-10

# CIFAR-10




TensorFlow

# CIFAR-10

32x32x3

Feature extractor

Clas sifi er

0.1
0
0.05
0.1
0
0
0.1
0.6
0
0.05

# CIFAR-10

```
# PLACEHOLDERS
x = tf.placeholder(tf.float32, [None, 32, 32, 3])
y = tf.placeholder(tf.float32, [None, 10])
```

# Preprocessing

**input**

**standardize**

$$\frac{- \mu}{\sigma}$$

**target**

7

**one-hot encode**

`(0, 0, 0, 0, 0, 0, 0, 1, 0, 0)`

# Cost Function: cross-entropy

$$C = -\frac{1}{n} \sum_x [y \ln a + (1 - y) \ln(1 - a)] ,$$

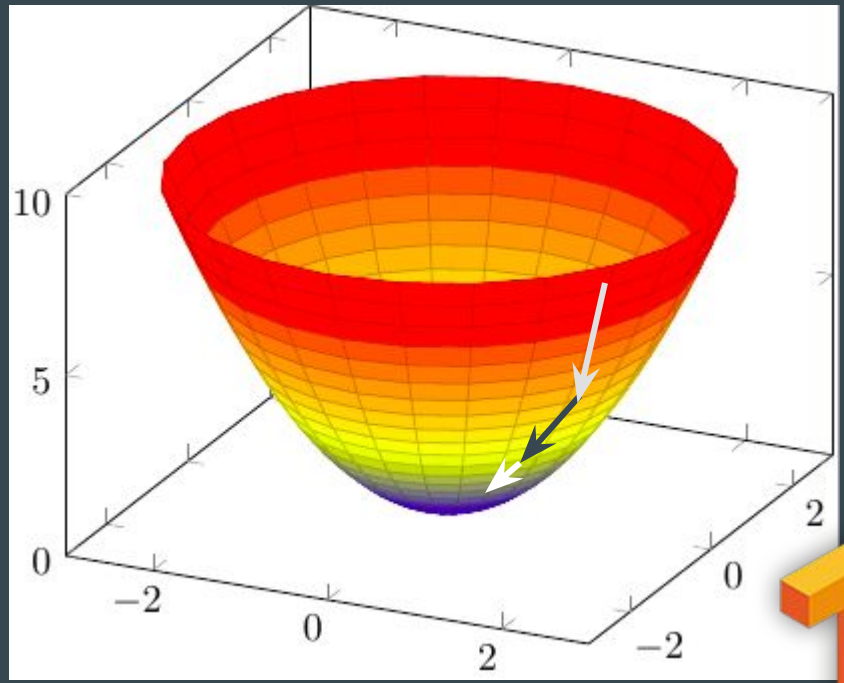
típica en clasificación

# Cost Function: cross-entropy

```
cost =
tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
logits=logits, labels=y))
```

# Adam Optimizer

ajusta el learning rate




TensorFlow

# Adam Optimizer

```
optimizer = tf.train.AdamOptimizer().minimize(cost)
```

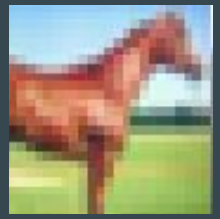
# Vamos al notebook

¿Es mejorable ese 70%?

# Data augmentation

original

flip



zoom

brighten

TensorFlow

# Enlarging the network


# Enlarging the network


TensorFlow

# Enlarging the network

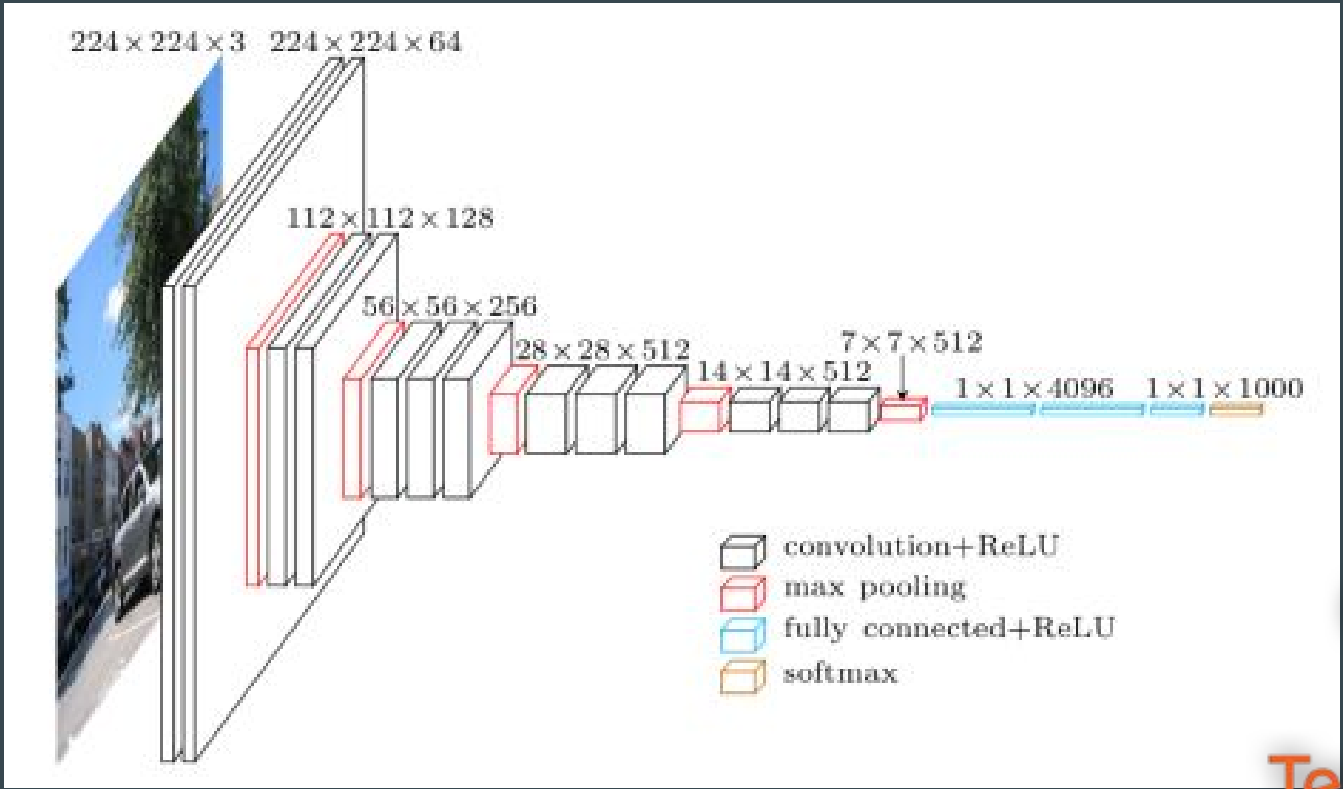

TensorFlow

# Transfer Learning

# ImageNet

TensorFlow

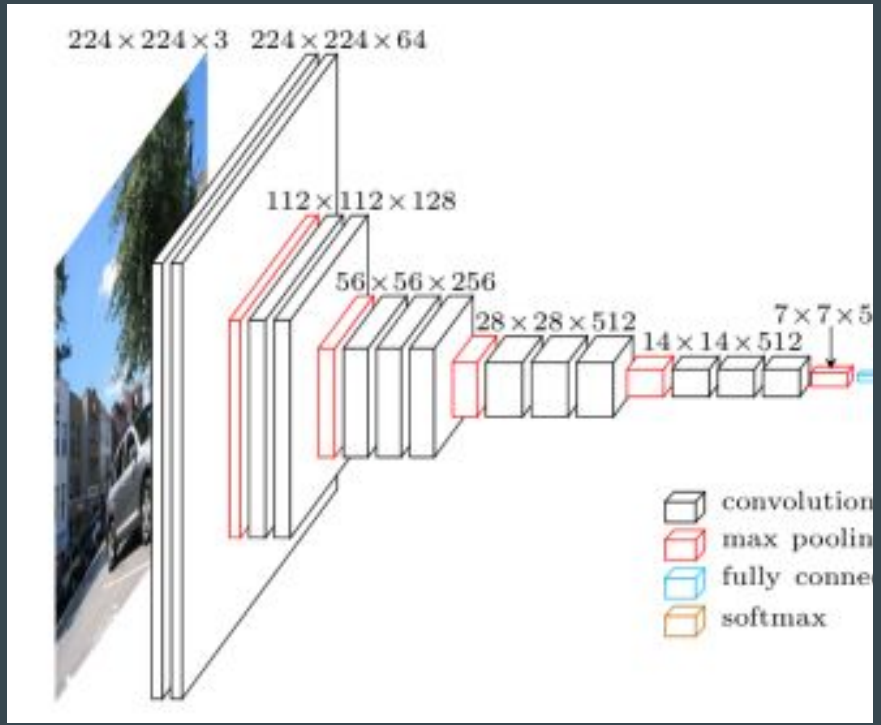
# VGG-16




TensorFlow

# transfer learning

x



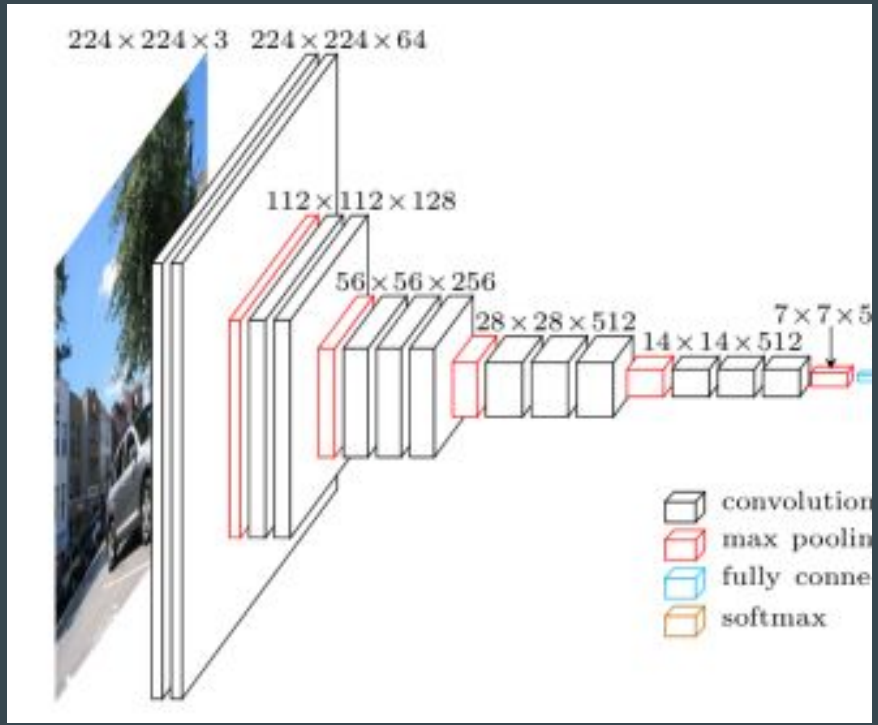

Clas
sifie
r

pred

TensorFlow

# transfer learning

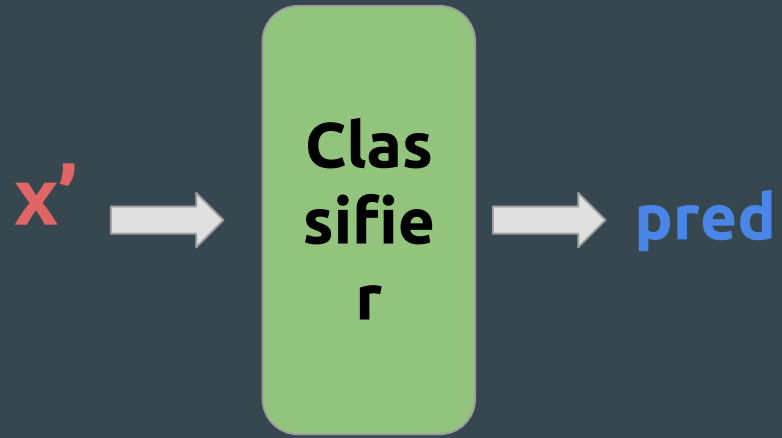
x




x'

TensorFlow

# transfer learning




# Vamos al notebook

Gracias!

alesolano/mastering_tensorflow