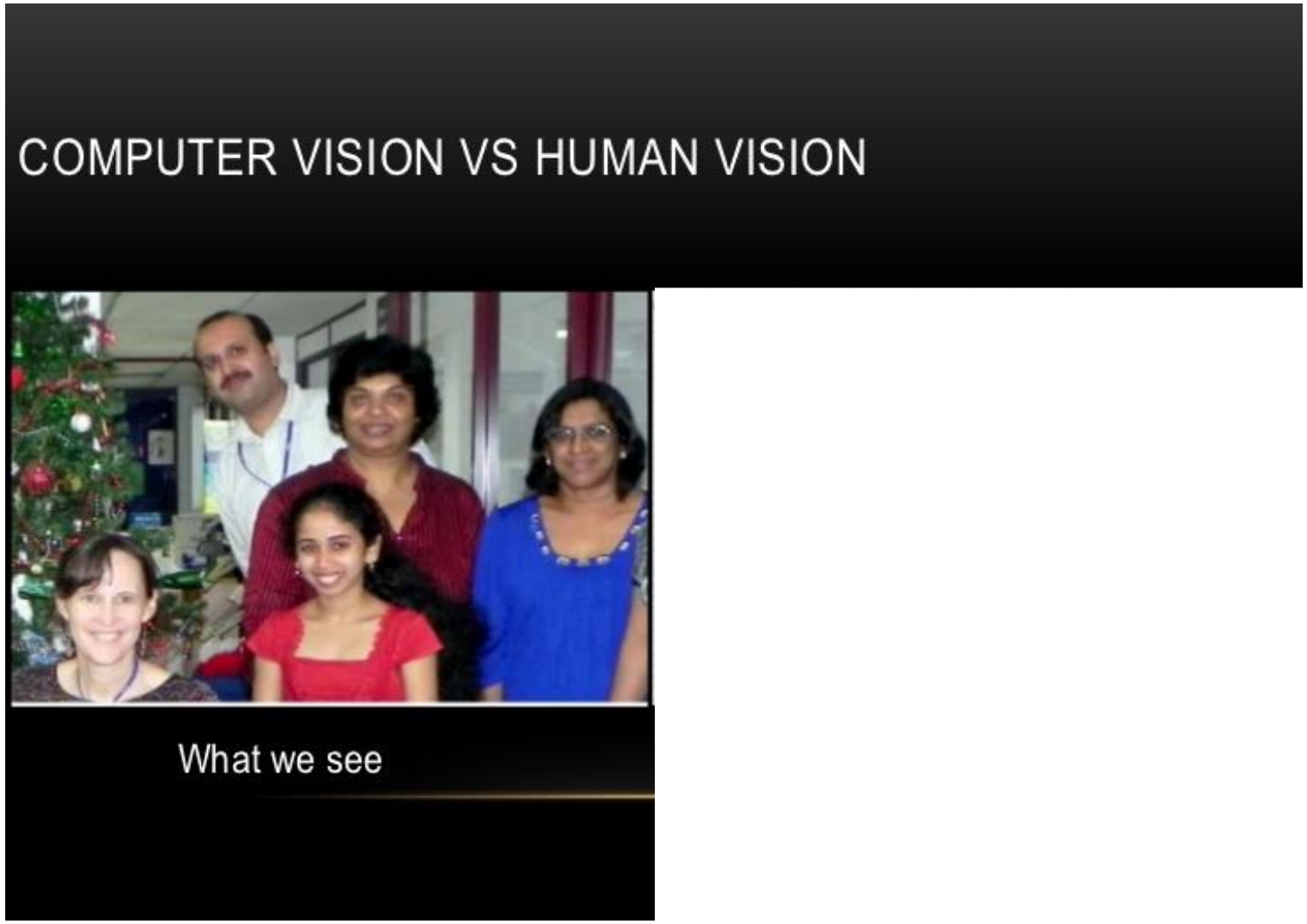
Computer Vision

What is computer vision

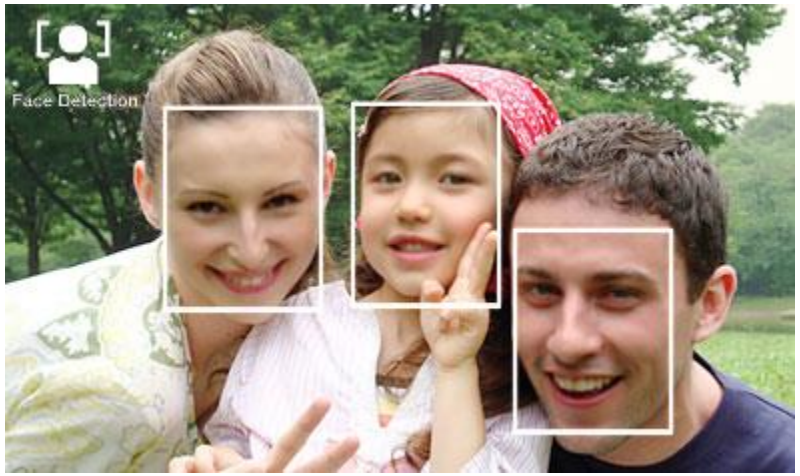
- Computer vision is a field that includes methods for acquiring, processing, analysing and understanding images
- Duplicate the abilities of human vision by electronically perceiving and understanding an image.
- Image data can take many forms, such as a video sequence, depth images, views from multiple cameras, medical scanner, satellite images etc.

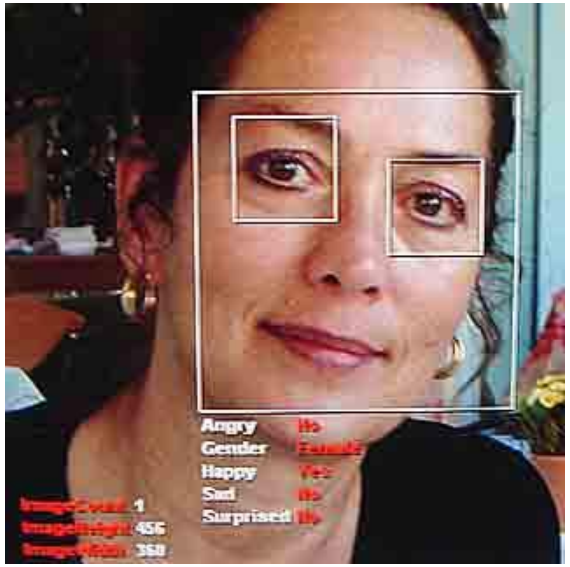
What a computer sees?



What a computer sees?

- Learn to recognize faces and expressions.





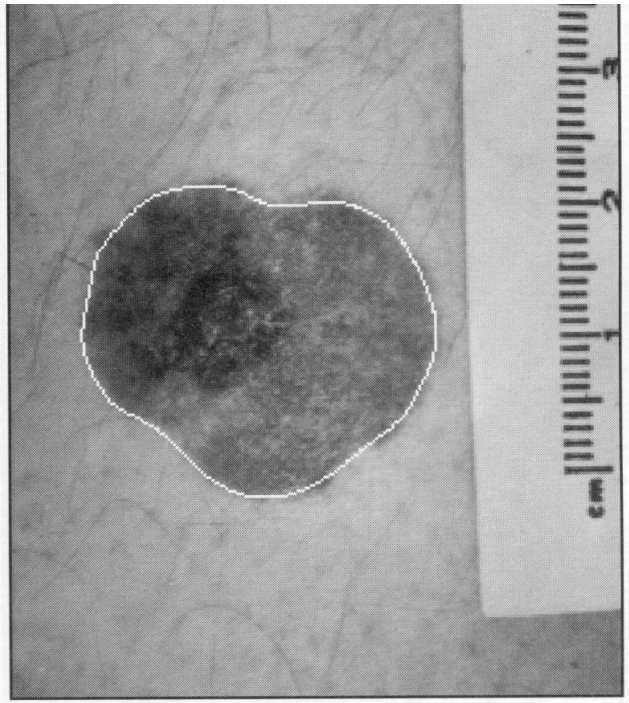
ALGORITHMS

Support Vector Machines

Adaboost

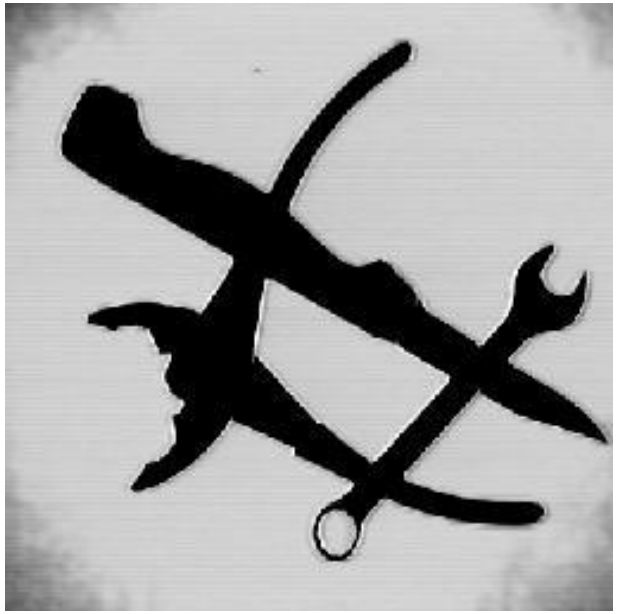
Medical Applications

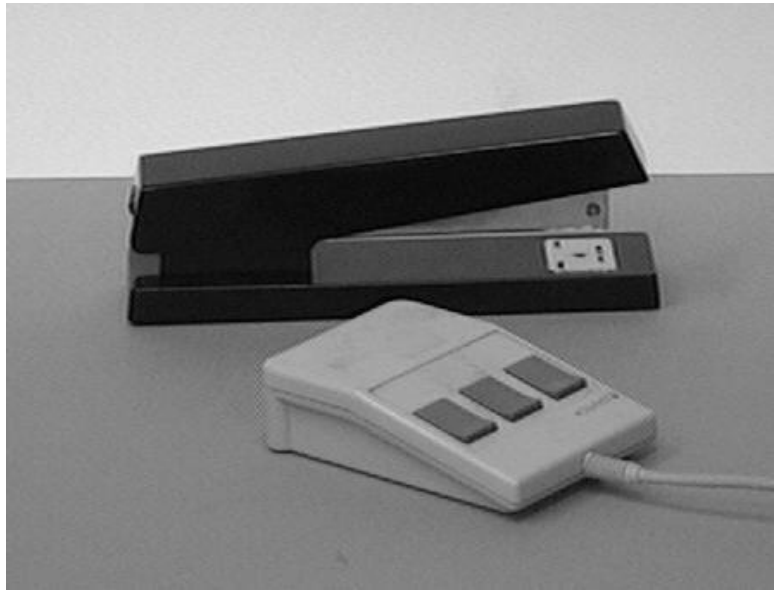
- skin cancer



breast cancer

Object Recognition

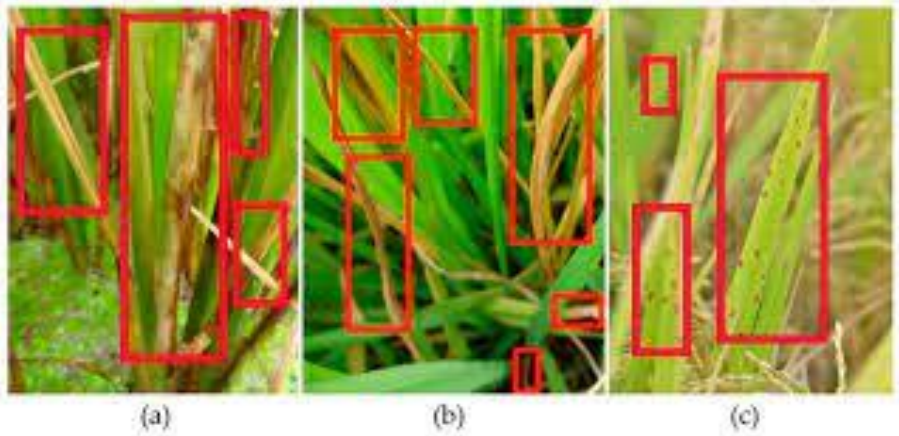




Object Detection

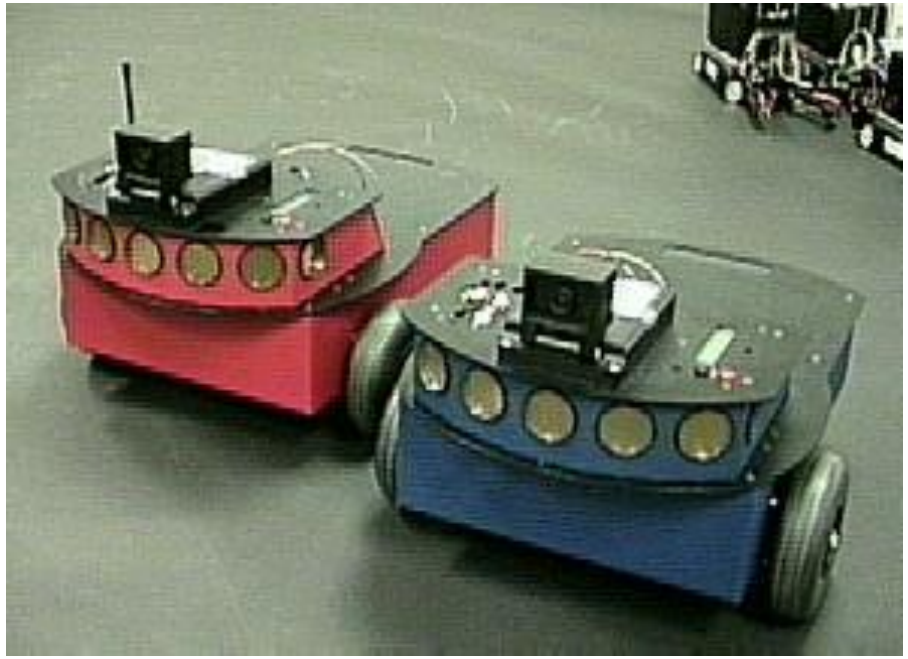
Agriculture

- Crop monitoring, flower detection
  plant disease detection
  Insect detection



Autonomous Vehicles

- Land, Underwater, Space



Interpretation of Aerial Photography

Interpretation of satellite Imagery

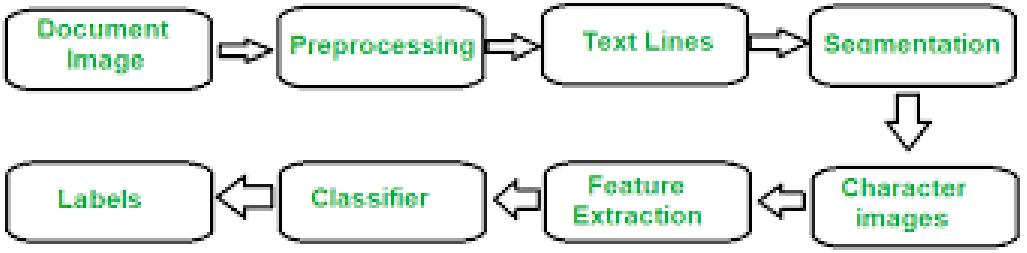
ALGORITHMS
Mask RCNN
Single shot detection
yolo

Mining

Computer vision and IOT

Character Recognition

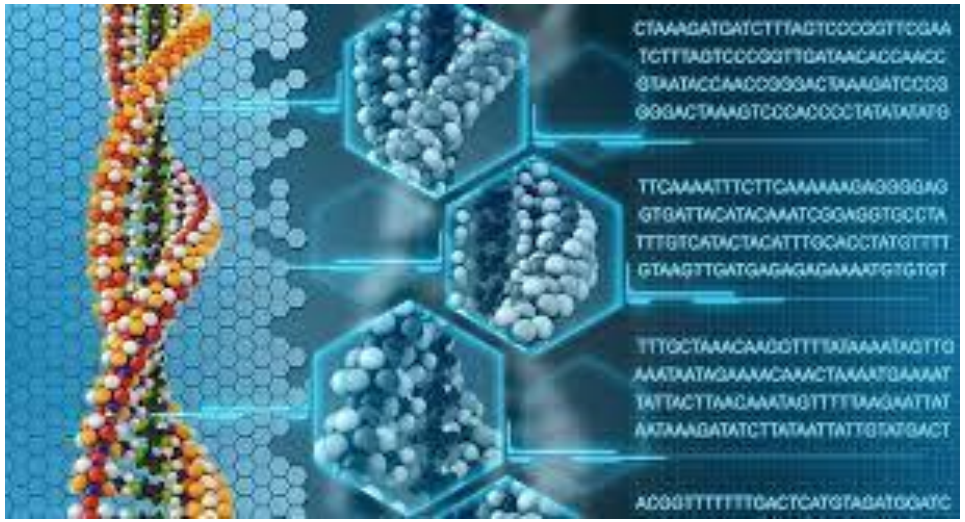
pytesseract



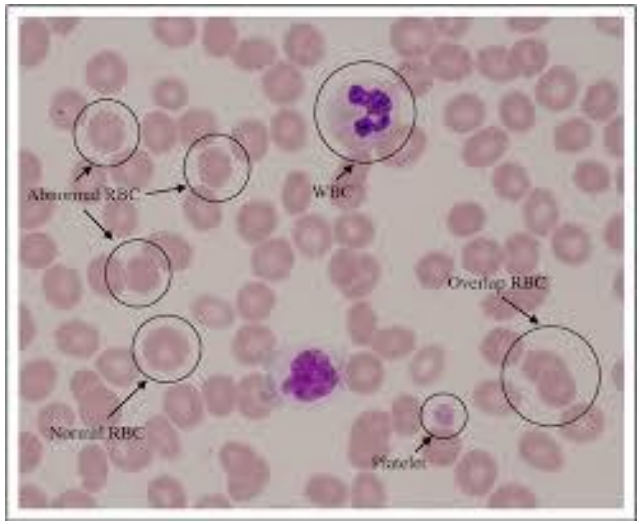
Flow Diagram of OCR

Many Applications: Invoice, id cards, Number plate detection etc

Bio Informatics

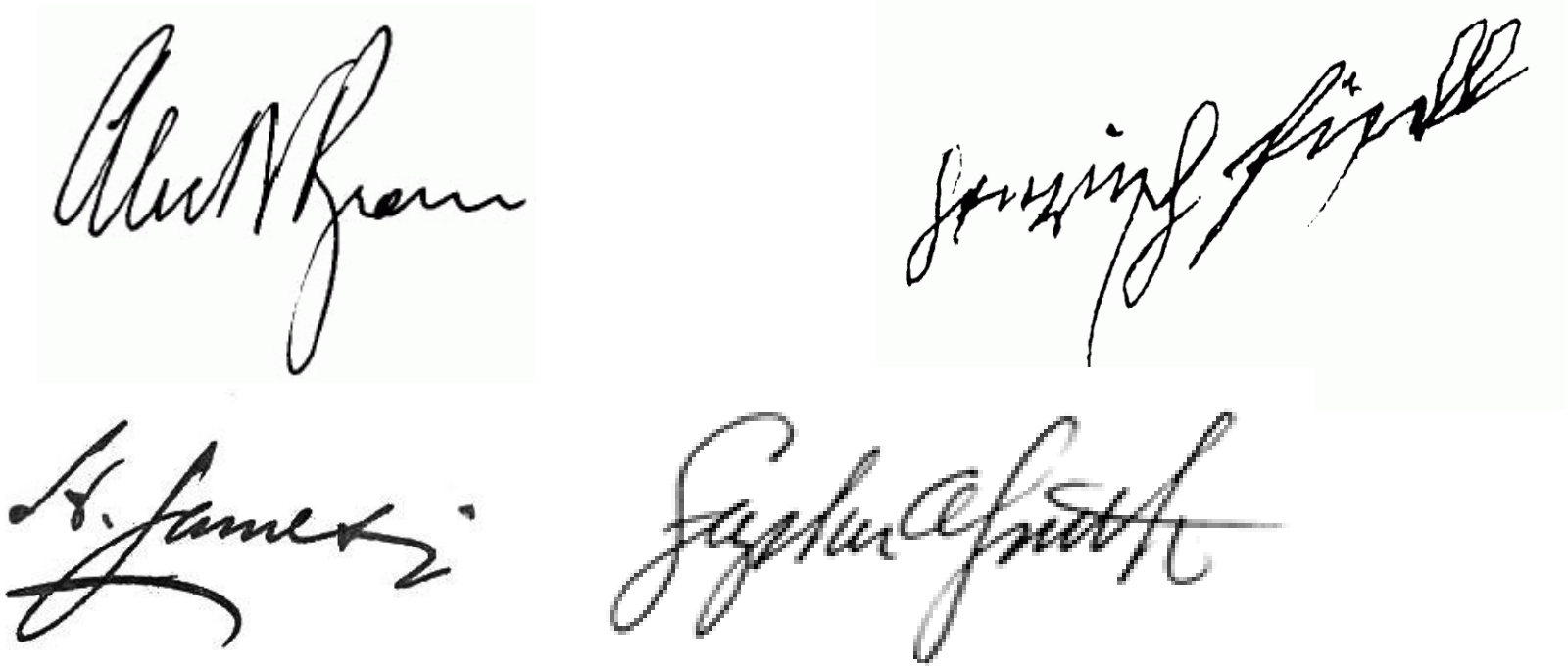


Cell classification

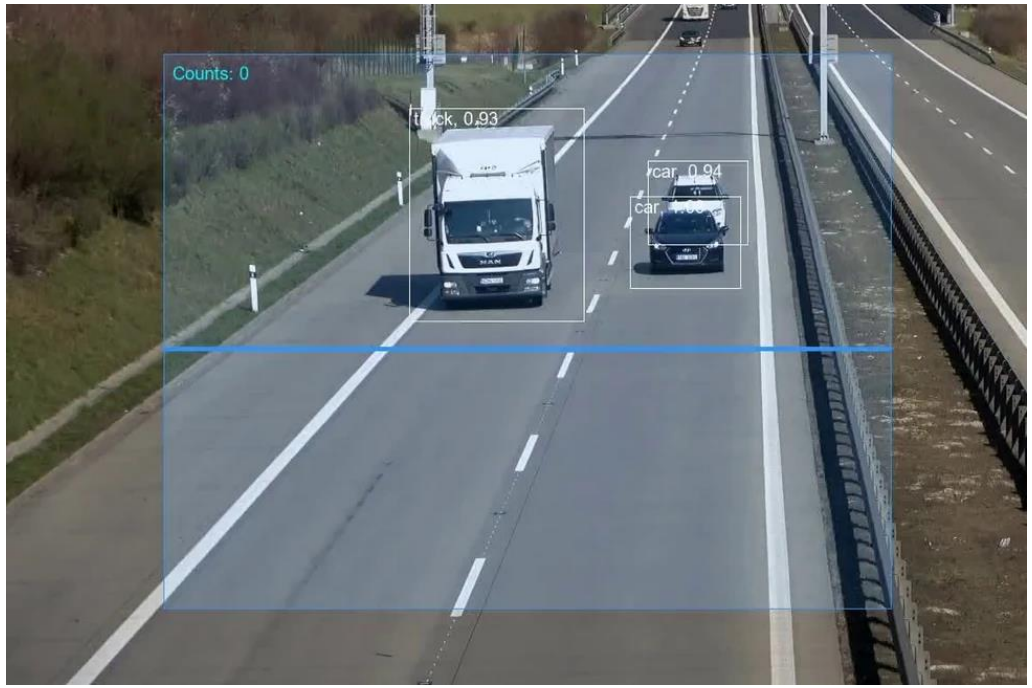


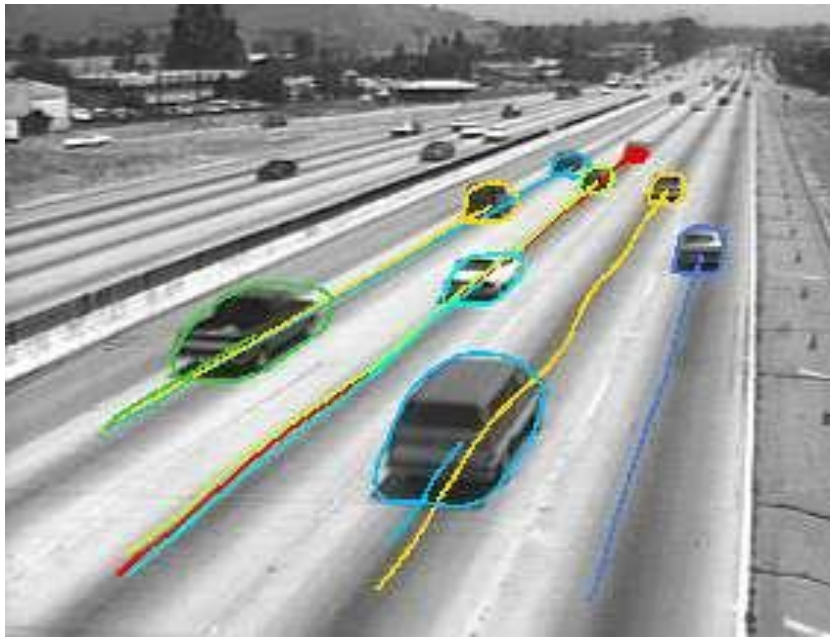
Signature recognition

- Recognize signatures by structural similarities which are difficult to quantify.
- does a signature belongs to a specific person, say Tony Blair, or not.



Traffic Monitoring





Customer tracking in Malls

Strategically placed counting devices throughout a retail store can gather data through machine learning processes about where customers spend their time, and for how long.

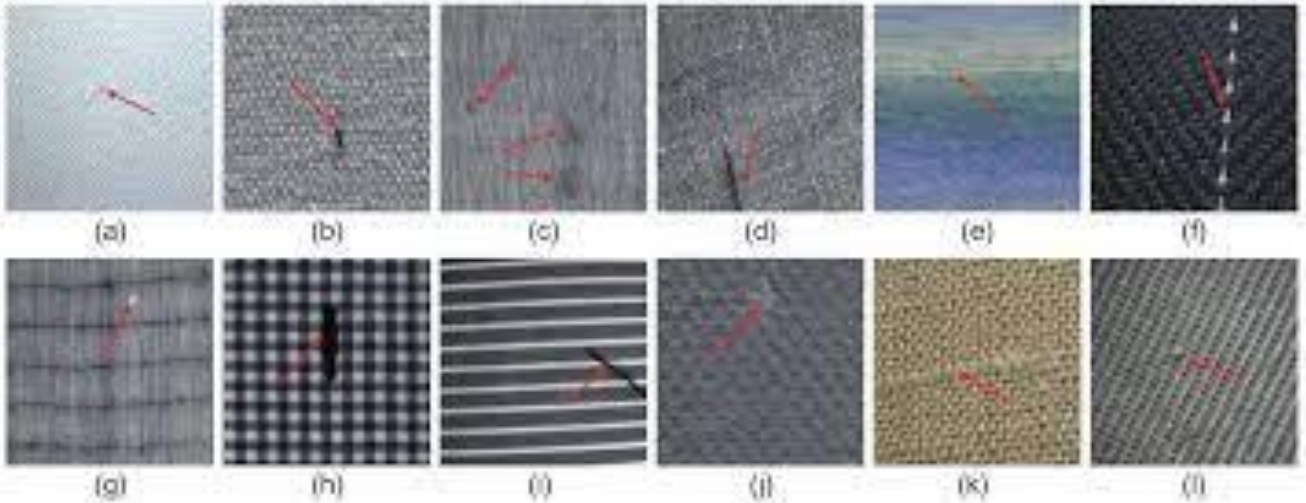
Customer analytics can improve retail stores' understanding of consumer interaction and improve store layout optimization.

Image transformation using GANS

Create faces that never exists

Art using GANS

Defect detection in Industry



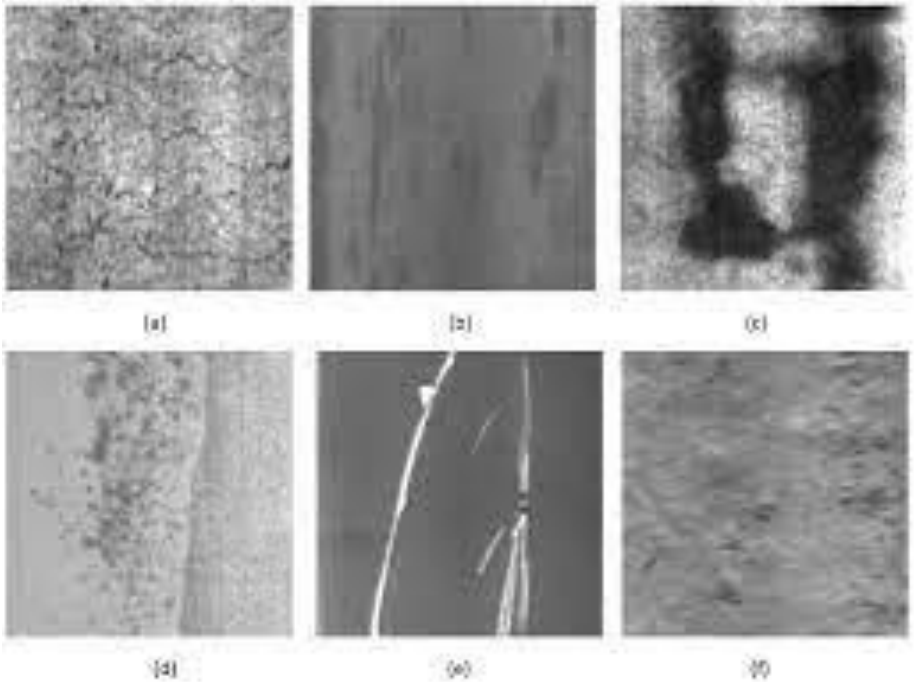


Image Annotation

Given an image, what are the words that describe the image?

Model	
Human Annotation	sky jet plane smoke
Automatic Annotation	plane jet smoke flight prop

Image Retrieval

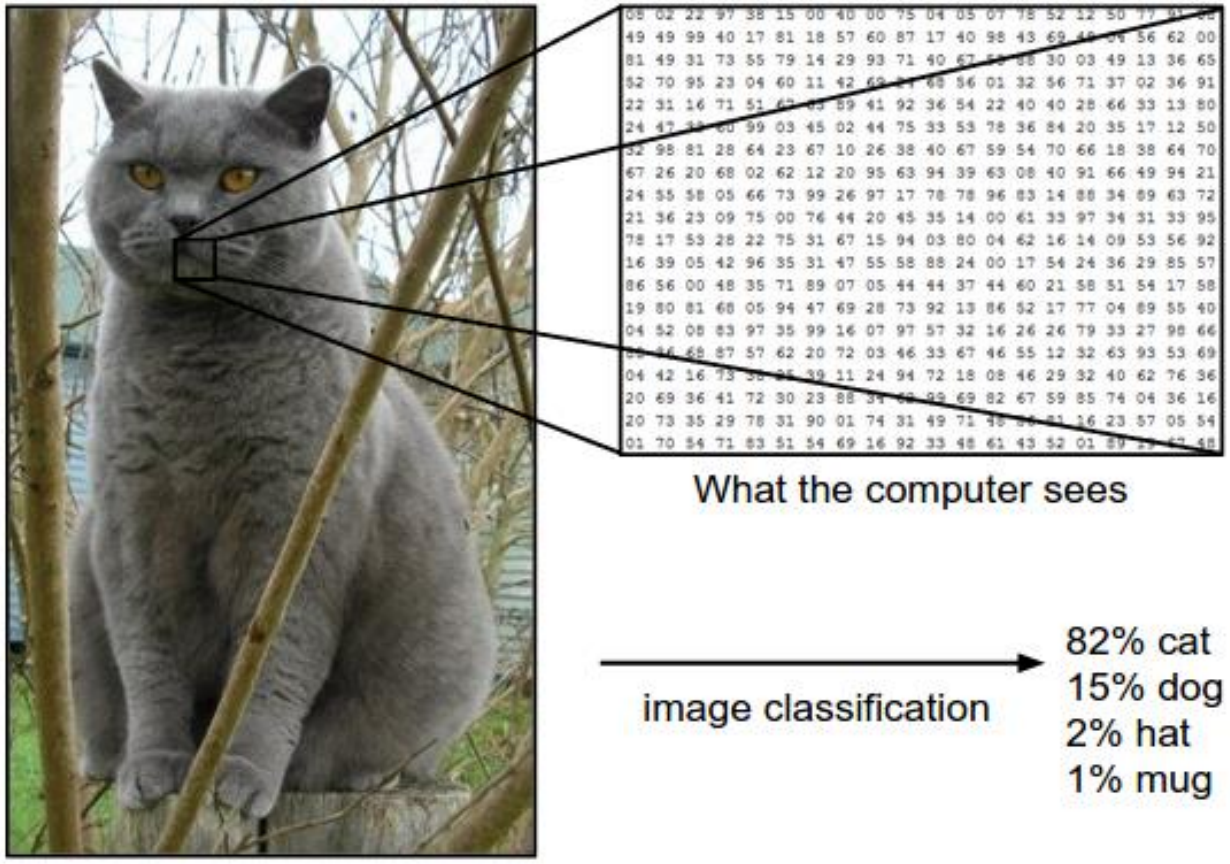
Given a database of images and a query string (e.g. words), what are the images that are described by the words?

Query String: “jet”

Image classification

Image classification

Image classification is where a computer can analyze an image and identify the ‘class’ the image falls under. (Or a probability of the image being part of a ‘class’.) A class is essentially a label, for instance, ‘car’, ‘animal’, ‘building’ and so on



For example, you input an image of a cat. Image classification is the process of the computer analyzing the image and telling you it’s a cat. (Or the probability that it’s a cat.)

Image classification

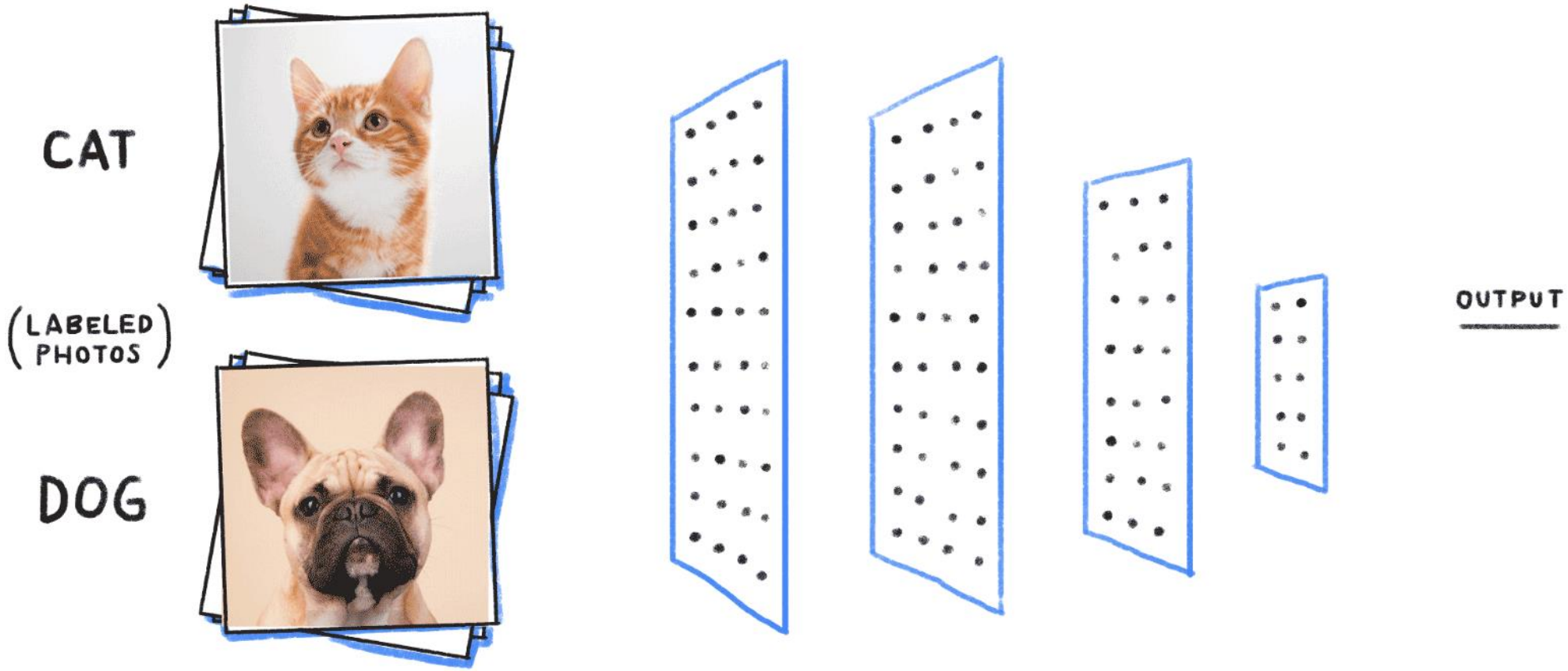
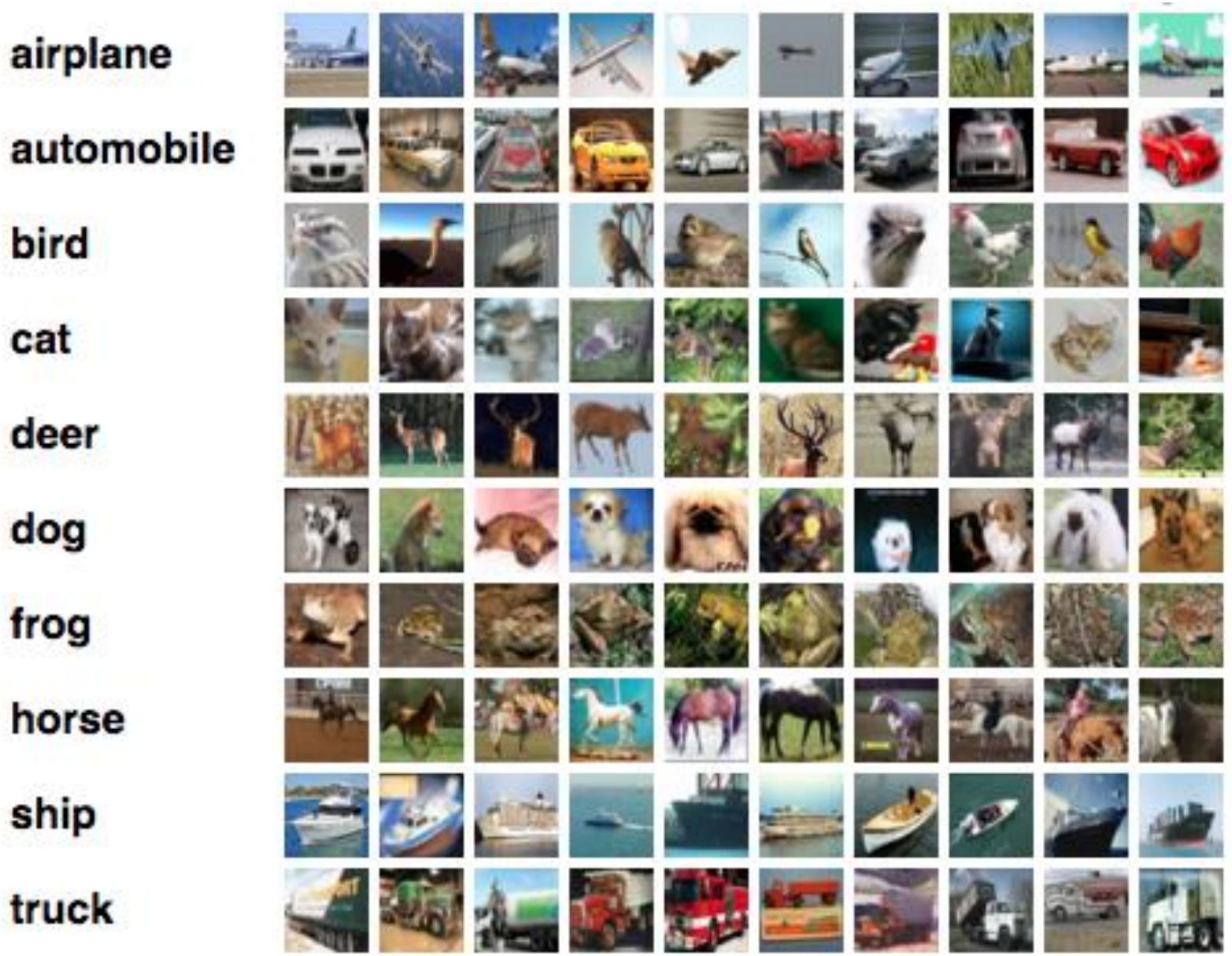


Image classification With Hand written digits

Image classification with CIFAR-10



Early Image classification

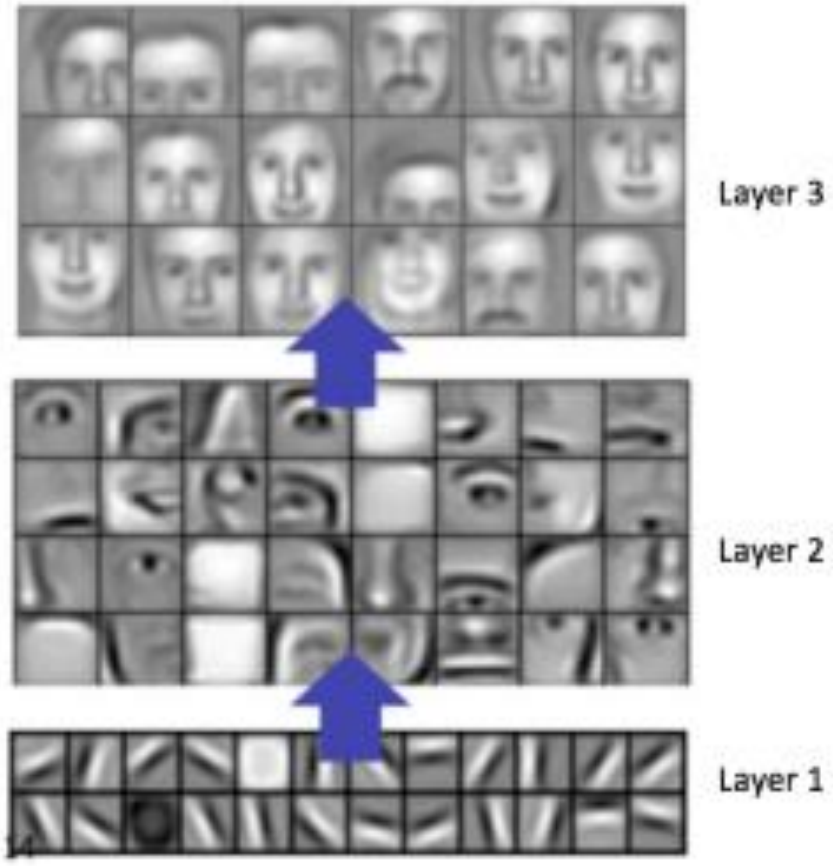
- Early image classification relied on raw pixel data.
- This meant that computers would break down images into individual pixels.
- The problem is that two pictures of the same thing can look very different.
- They can have different backgrounds, angles, poses, etcetera.
- This made it quite the challenge for computers to correctly ‘see’ and categorize images.

Adding Deep Learning for Image classification

- Deep learning involves the use of computer systems known as neural networks.
- In neural networks, the input filters through hidden layers of nodes. These nodes each process the input and communicate their results to the next layer of nodes. This repeats until it reaches an output layer, and the machine provides its answer.
- Image classification with deep learning most often involves convolutional neural networks, or CNNs. In CNNs, the nodes in the hidden layers don’t always share their output with every node in the next layer (known as convolutional layers).

Adding Deep Learning for Image classification

- Deep learning allows machines to identify and extract features from images. This means they can learn the features to look for in images by analysing lots of pictures. So, programmers don’t need to enter these filters by hand.



Higher level Feature

Mid level Feature

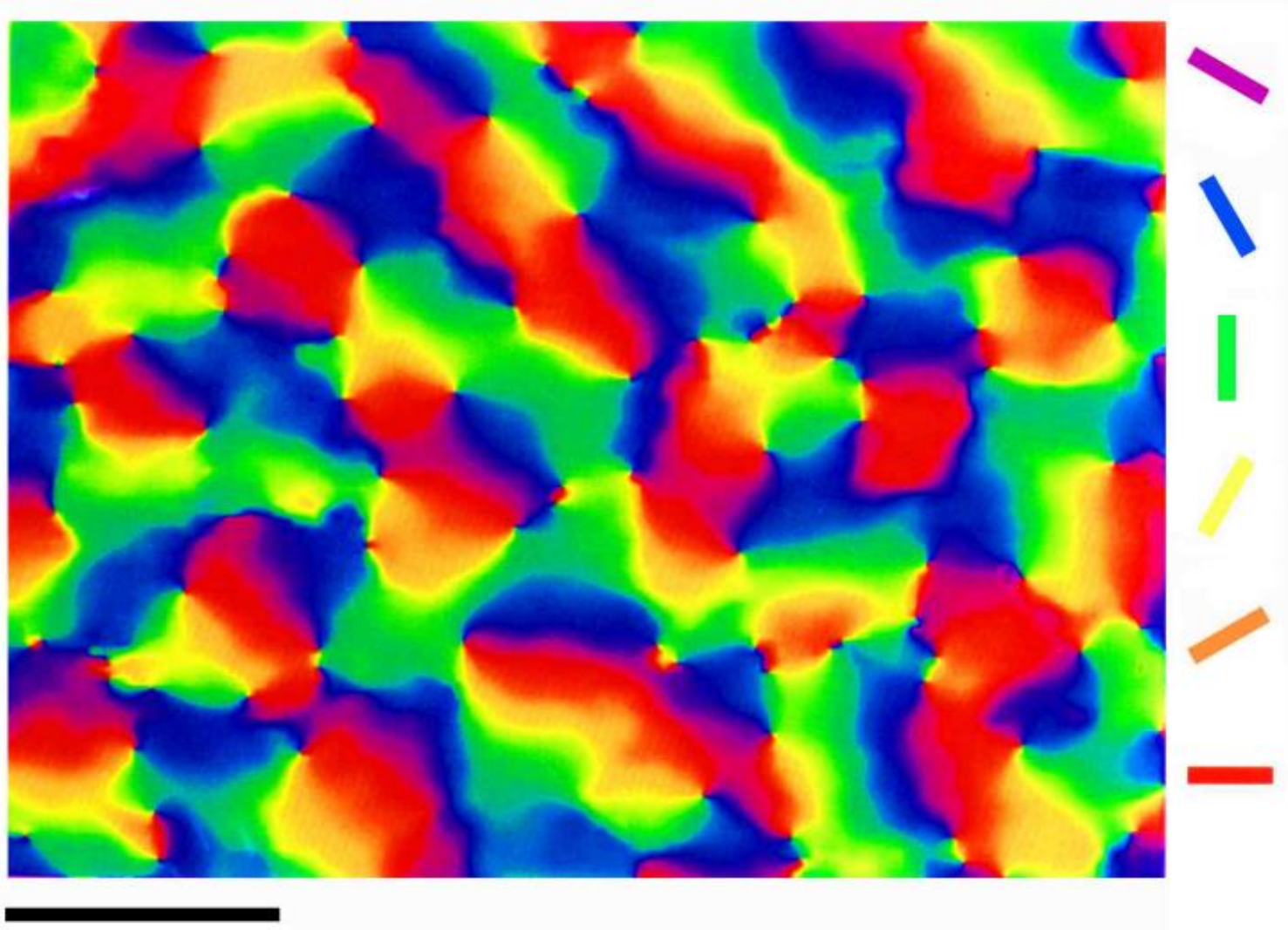
Low level Feature

Convolutional Neural Networks

Using correlations in images

- Natural images contain spatial correlations
- For example, pixels along a contour or edge
- How can we use these correlations?

Biological inspiration



What is a convolution?

```
array = np.array([0, 0, 0, 0, 0, 1, 1, 1, 1, 1])
kernel = np.array([-1, 1])
conv = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0])
conv[0] = (kernel * array[0:2]).sum()
conv[1] = (kernel * array[1:3]).sum()
conv[2] = (kernel * array[2:4]).sum()
...
for ii in range(8):
    conv[ii] = (kernel * array[ii:ii+2]).sum()
conv
```

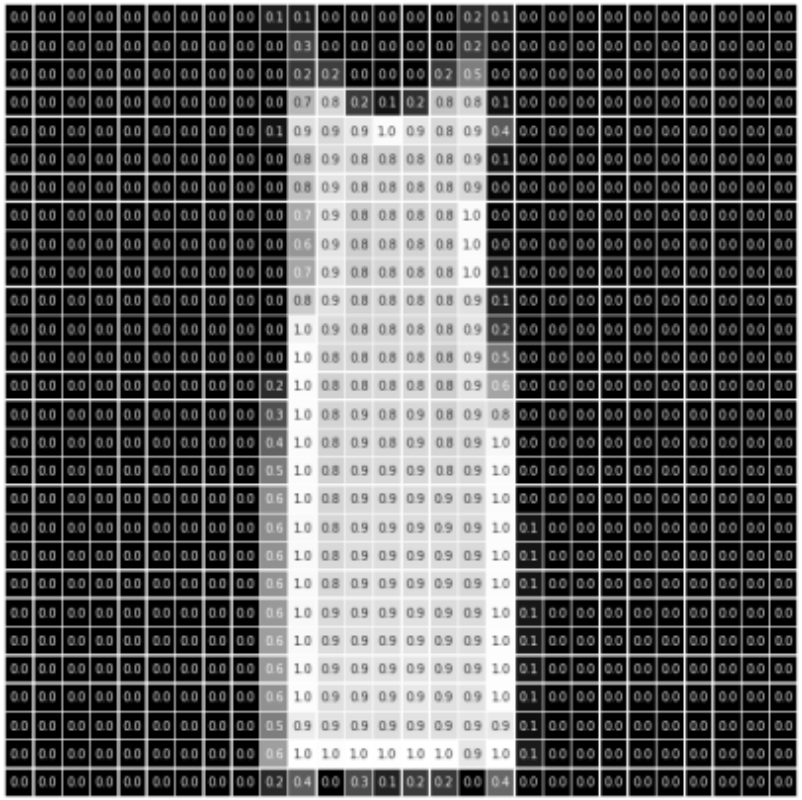
```
array([0, 0, 0, 0, 1, 0, 0, 0, 0])
```

Convolution in one dimension

```
array = np.array([0, 0, 1, 1, 0, 0, 1, 1, 0, 0])
kernel = np.array([-1, 1])
conv = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0])
for ii in range(8):
    conv[ii] = (kernel * array[ii:ii+2]).sum()
conv
```

```
array([ 0,  1,  0, -1,  0,  1,  0, -1,  0])
```

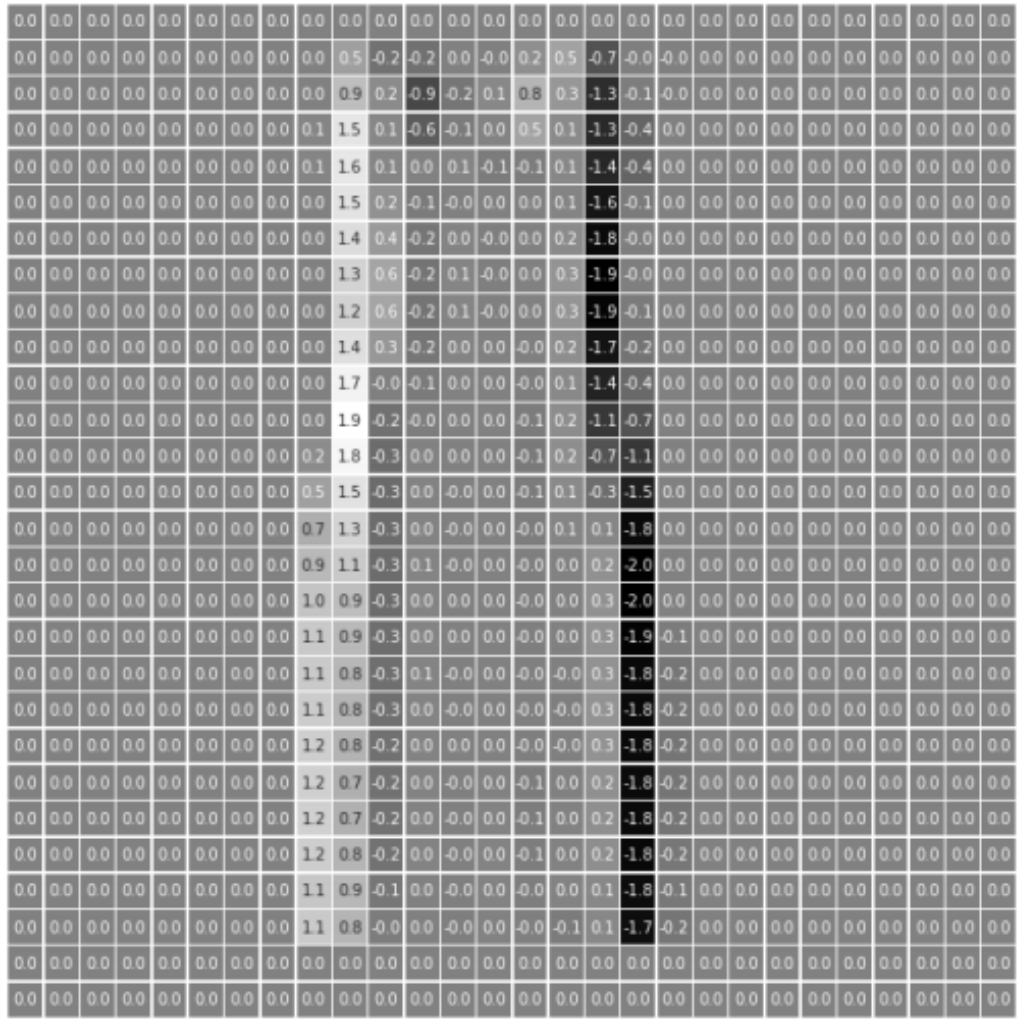
Image convolution



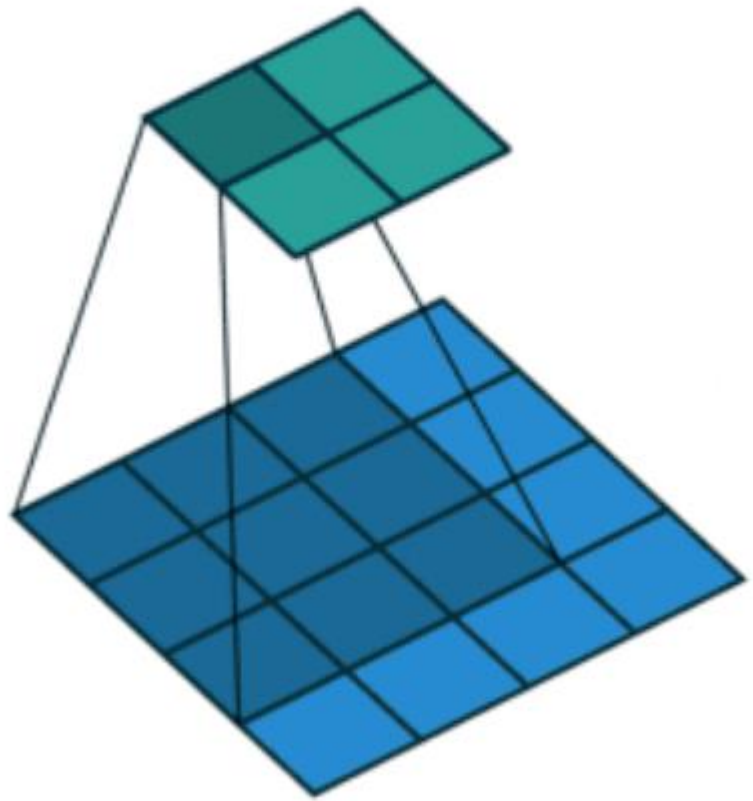
1.0

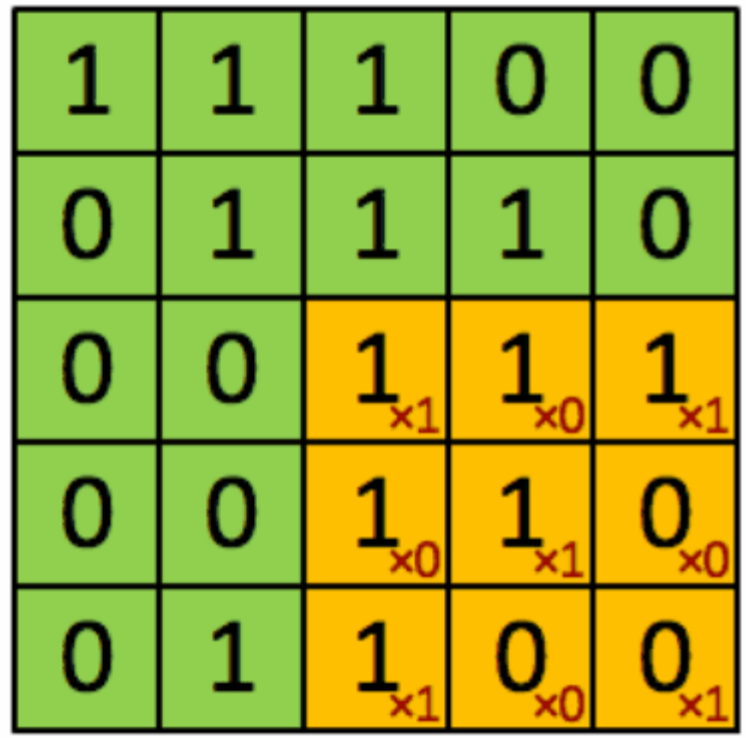
1.0

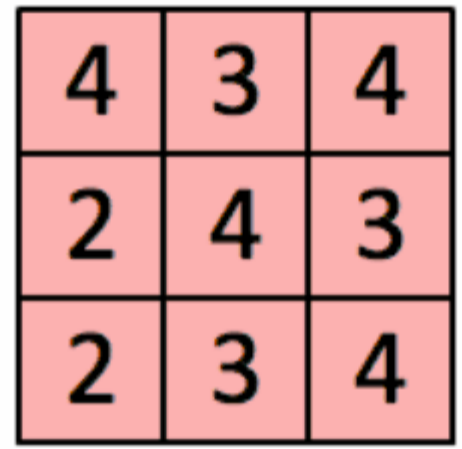
Image convolution



Convolution







Convoluting a 5×5×1 image with a 3×3×1 kernel to get a 3×3×1 convolved feature

Implementing convolutions in Keras

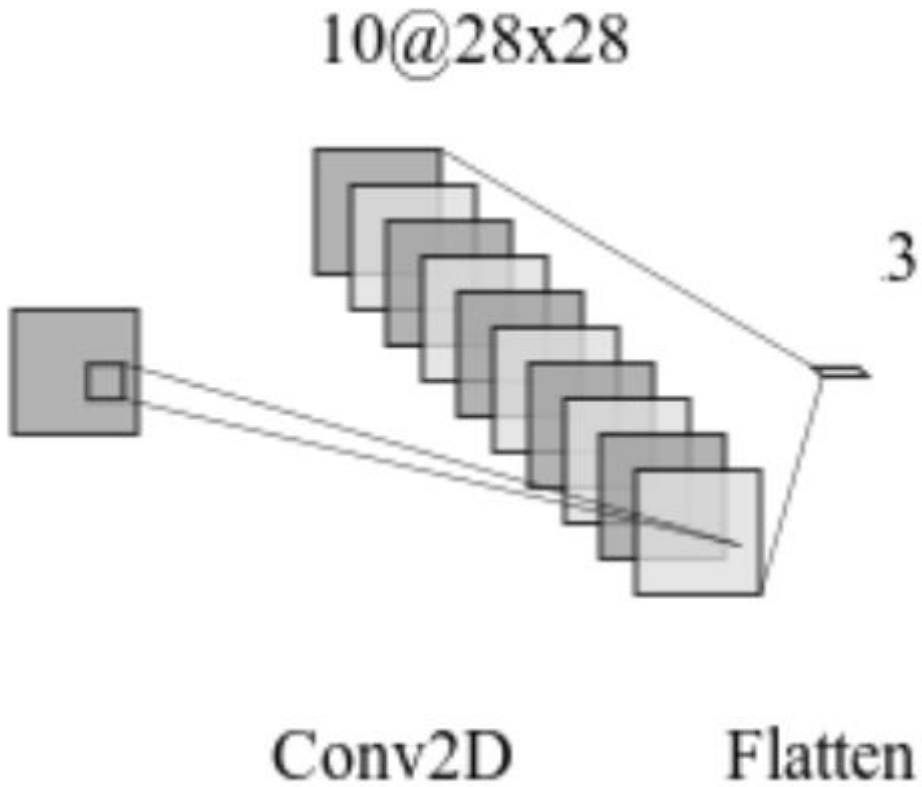
Keras Convolution layer

```
from keras.layers import Conv2D
Conv2D(10, kernel_size=3, activation='relu')
```

Integrating convolution layers into a network

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten
model = Sequential()
model.add(Conv2D(10, kernel_size=3, activation='relu',
          input_shape=(img_rows, img_cols, 1)))
model.add(Flatten())
model.add(Dense(3, activation='softmax'))
```

Our CNN



Fitting a CNN

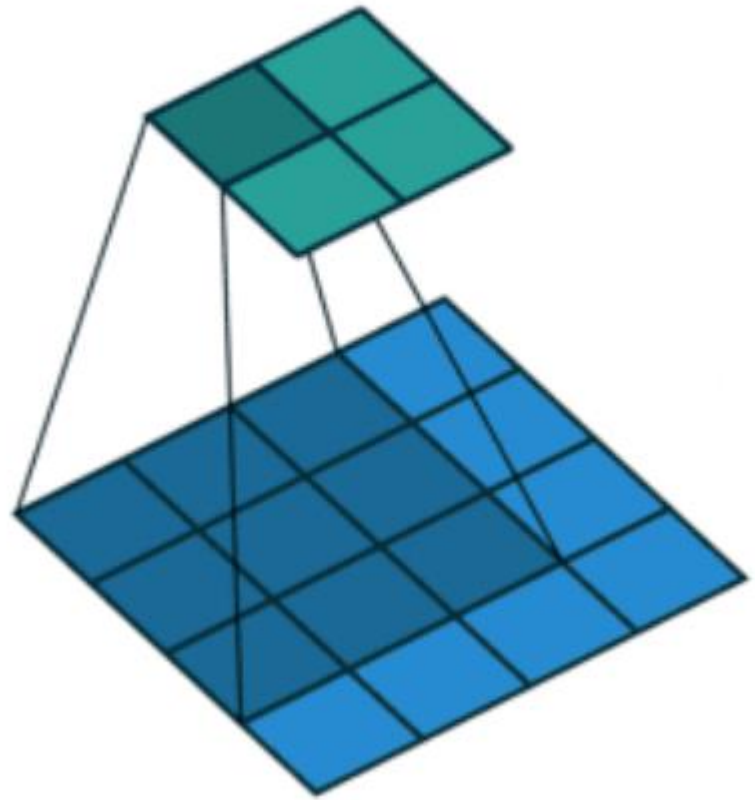
```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
train_data.shape
```

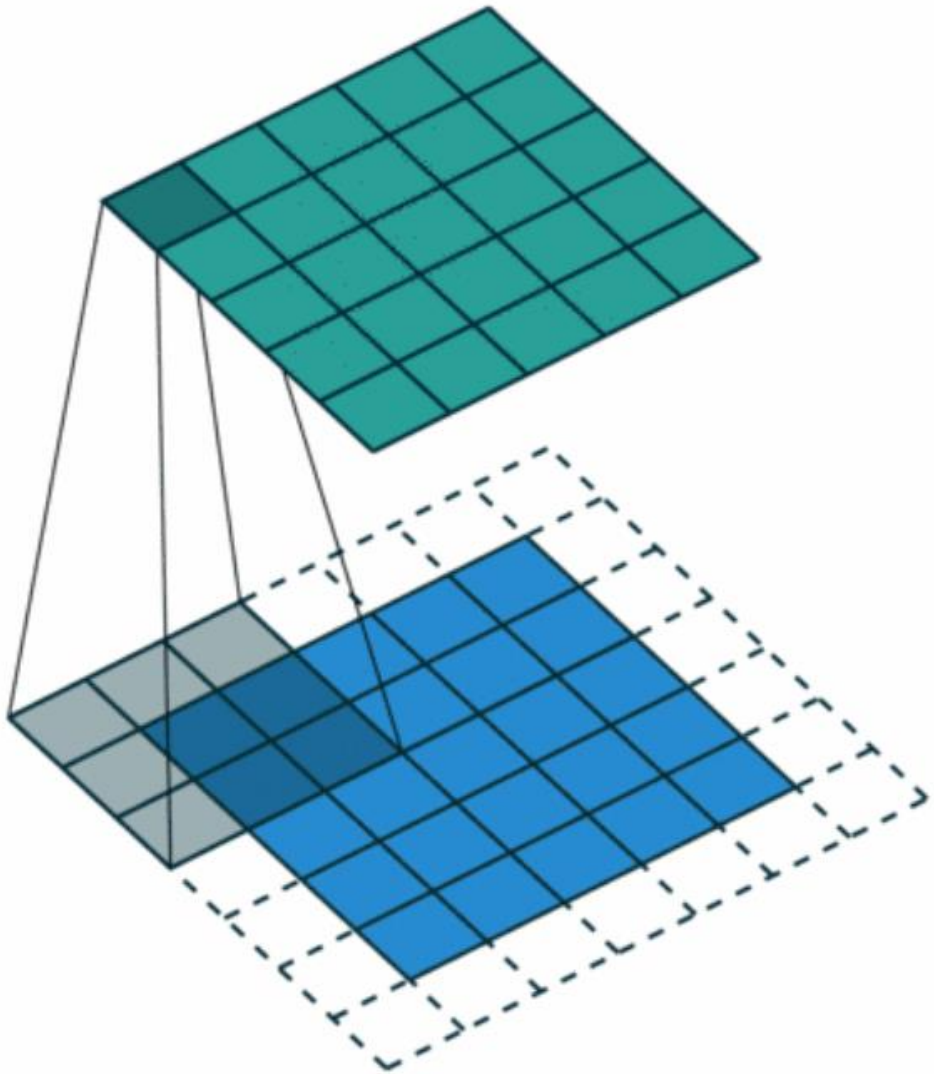
```
(50, 28, 28, 1)
```

```
model.fit(train_data, train_labels, validation_split=0.2,
          epochs=3)

model.evaluate(test_data, test_labels, epochs=3)
```

Tweaking your convolutions

Convolution





SAME padding: 5×5×1 image is padded with 0s to create a 6×6×1 image

Zero padding in Keras

```
model.add(Conv2D(10, kernel_size=3, activation='relu',
                 input_shape=(img_rows, img_cols, 1)),
                 padding='valid')
```

Zero padding in Keras

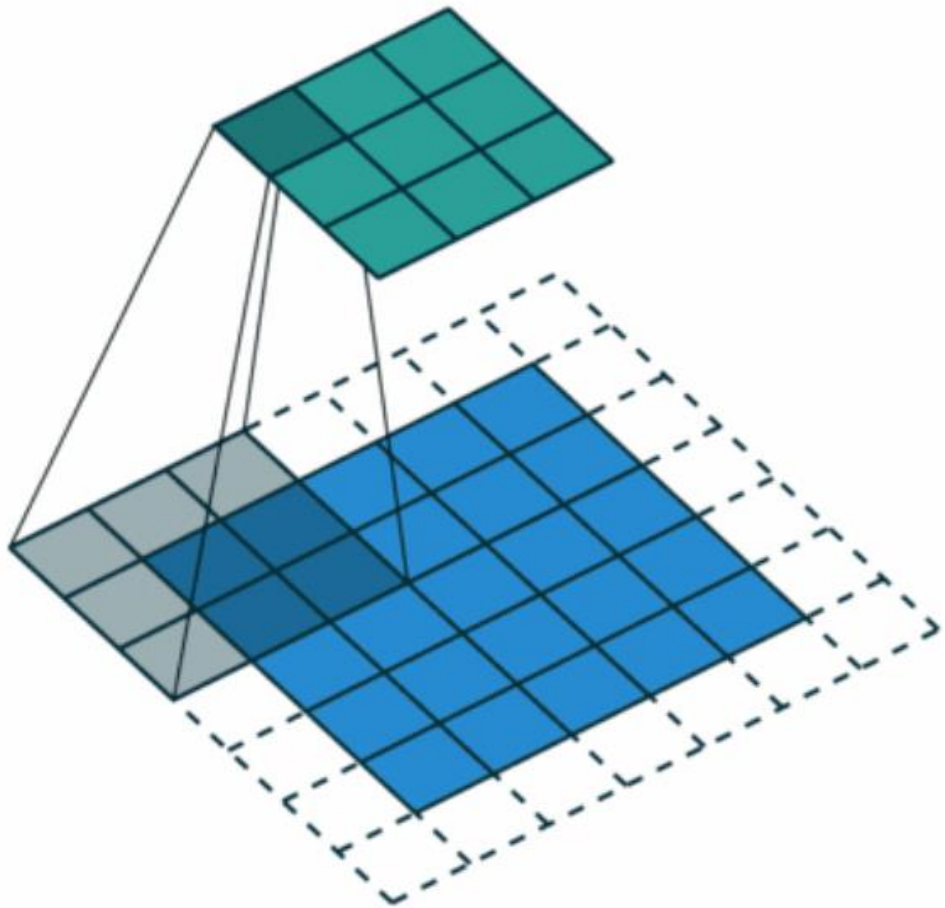
```
model.add(Conv2D(10, kernel_size=3, activation='relu',
                 input_shape=(img_rows, img_cols, 1)),
                 padding='same')
```

Strides in Keras

```
model.add(Conv2D(10, kernel_size=3, activation='relu',
                 input_shape=(img_rows, img_cols, 1)),
          strides=1)
```

Strides in Keras

```
model.add(Conv2D(10, kernel_size=3, activation='relu',
                 input_shape=(img_rows, img_cols, 1)),
          strides=2)
```



Convolution Operation with Stride Length = 2

Calculating the size of the output

$$O = ((I - K + 2P)/S) + 1$$

where

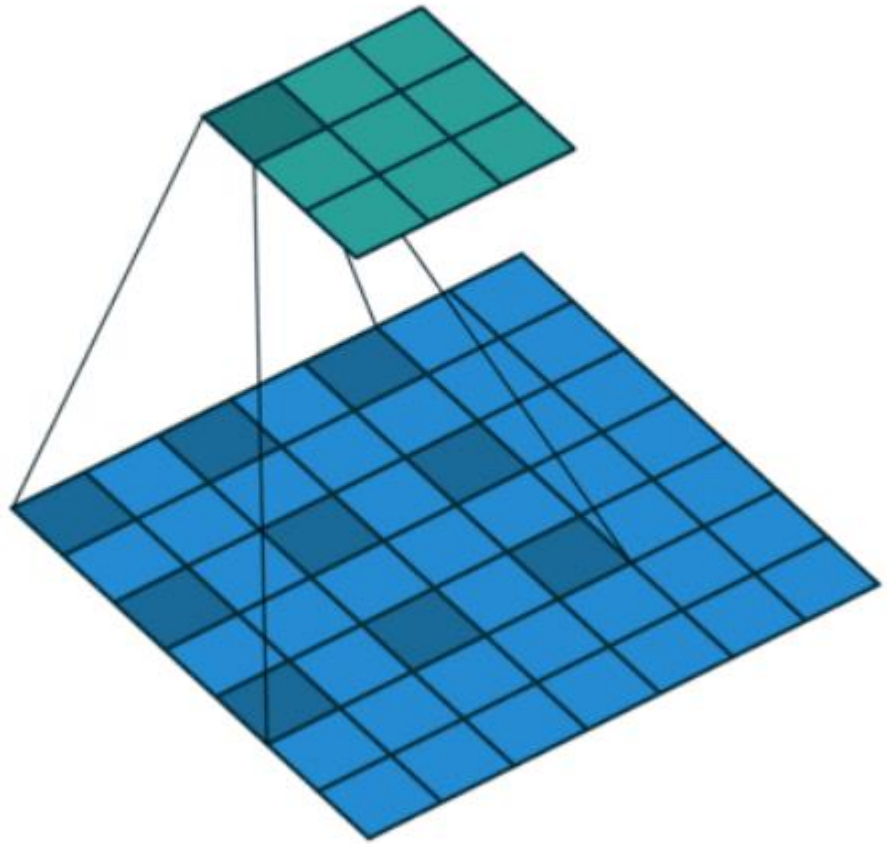
- I = size of the input
- K = size of the kernel
- P = size of the zero padding
- S = strides

Calculating the size of the output

$$28 = ((28 - 3 + 2)/1) + 1$$

$$10 = ((28 - 3 + 2)/3) + 1$$

Dilated convolutions

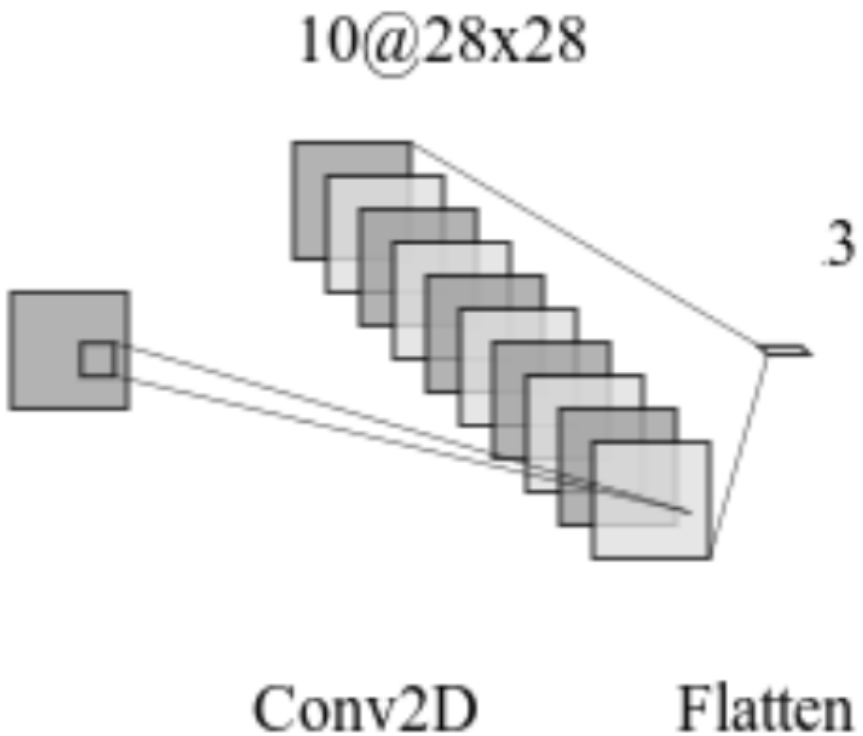


Dilation in Keras

```
model.add(Conv2D(10, kernel_size=3, activation='relu',
                 input_shape=(img_rows, img_cols, 1)),
          dilation_rate=2)
```

Going deeper

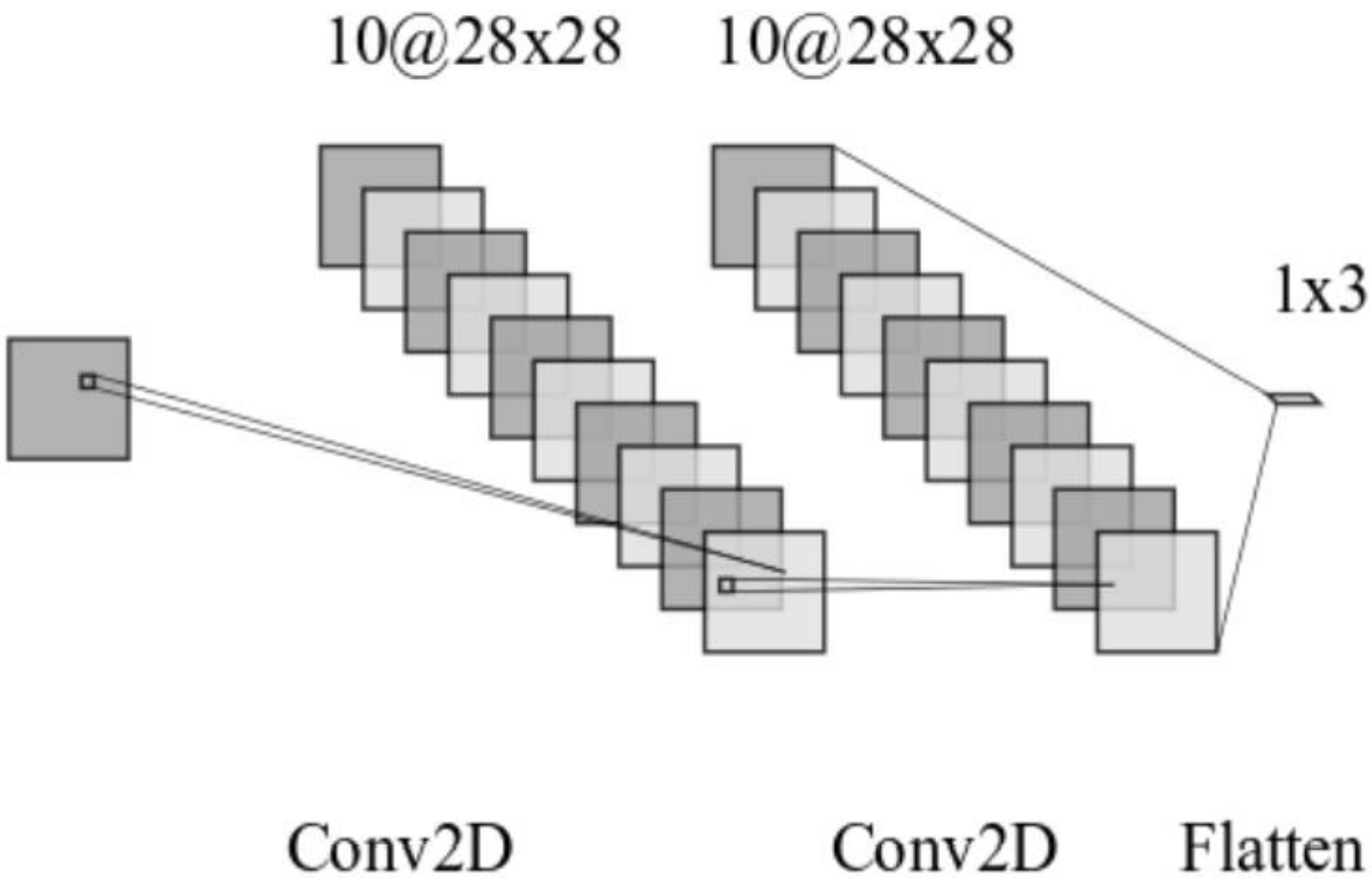
Network with one convolutional layer



Network with one convolutional layer: implementation

```
model = Sequential()
model.add(Conv2D(10, kernel_size=2, activation='relu',
                 input_shape=(img_rows, img_cols, 1)))
model.add(Flatten())
model.add(Dense(3, activation='softmax'))
```

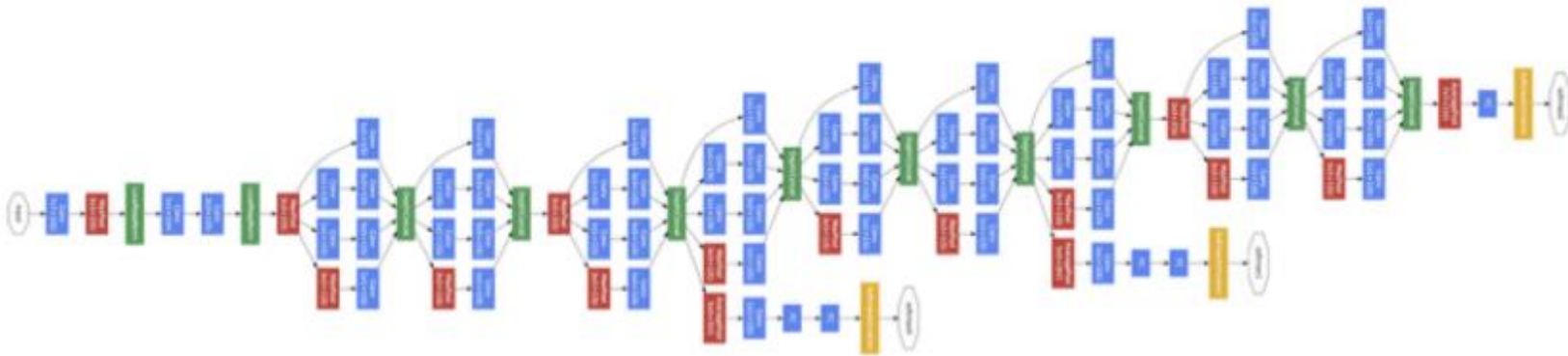
Building a deeper network



Building a deep network

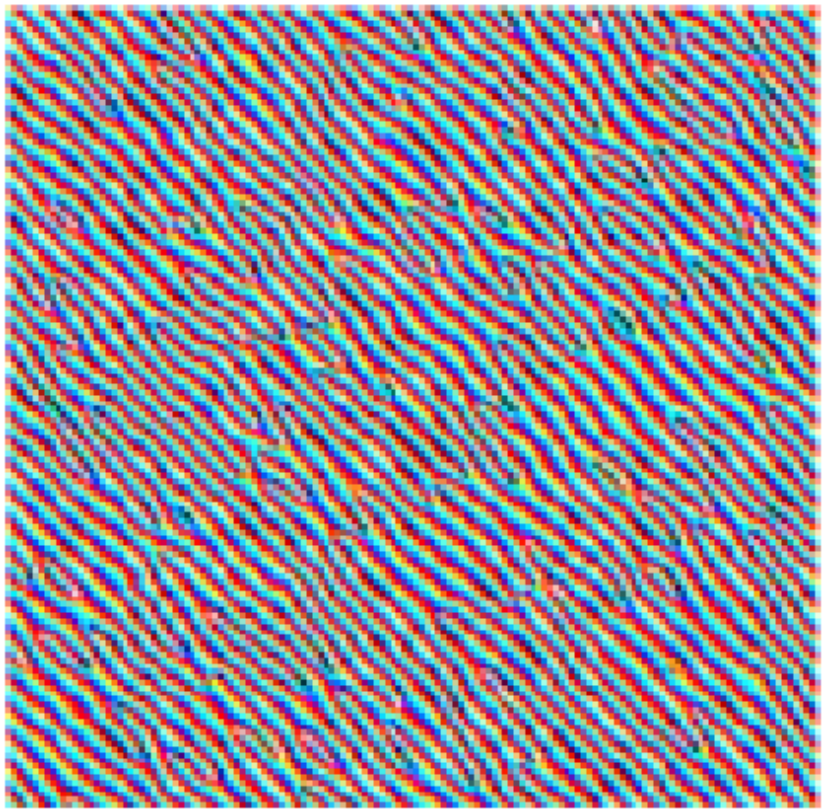
```
model = Sequential()
model.add(Conv2D(10, kernel_size=2, activation='relu',
                 input_shape=(img_rows, img_cols, 1),
                 padding='equal'))
# Second convolutional layer
model.add(Conv2D(10, kernel_size=2, activation='relu')
model.add(Flatten())
model.add(Dense(3, activation='softmax'))
```

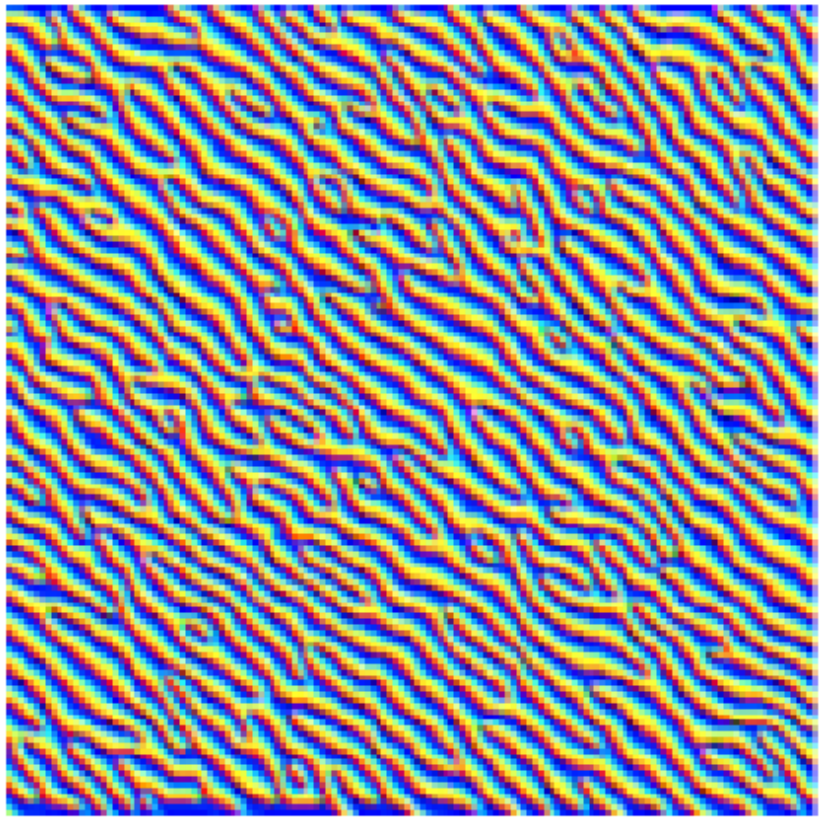
Why do we want deep networks?



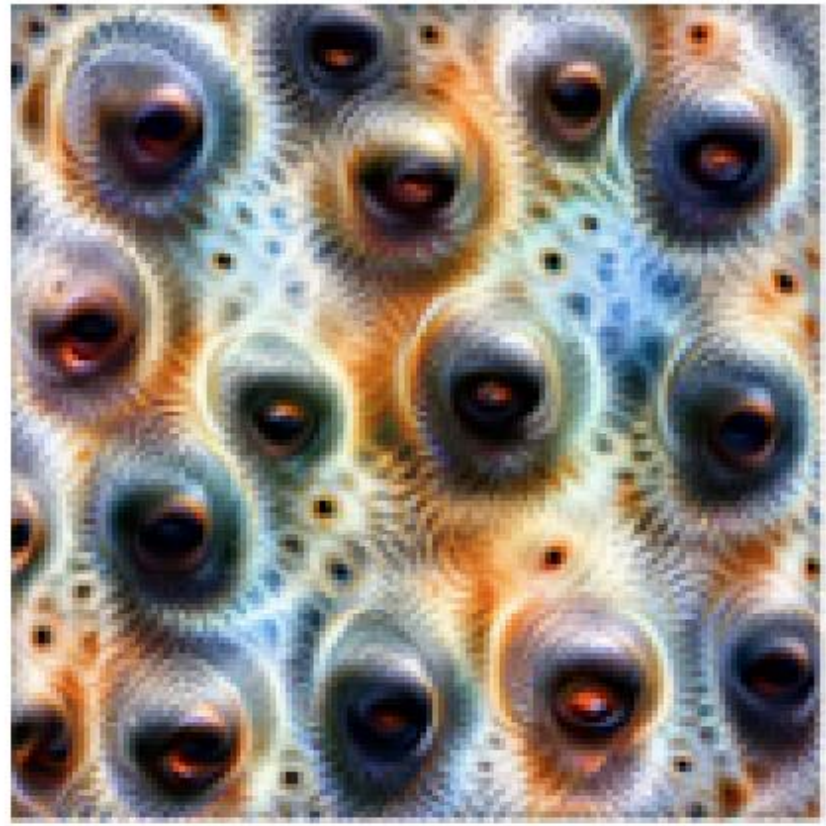
Convolution
Pooling
Softmax
Other

Features in early layers

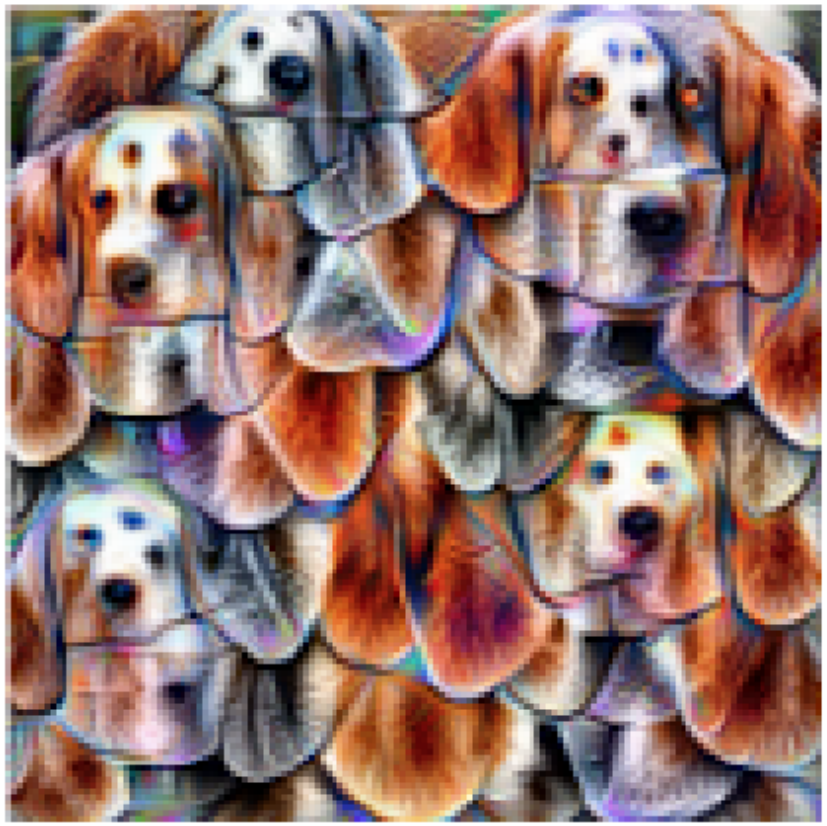


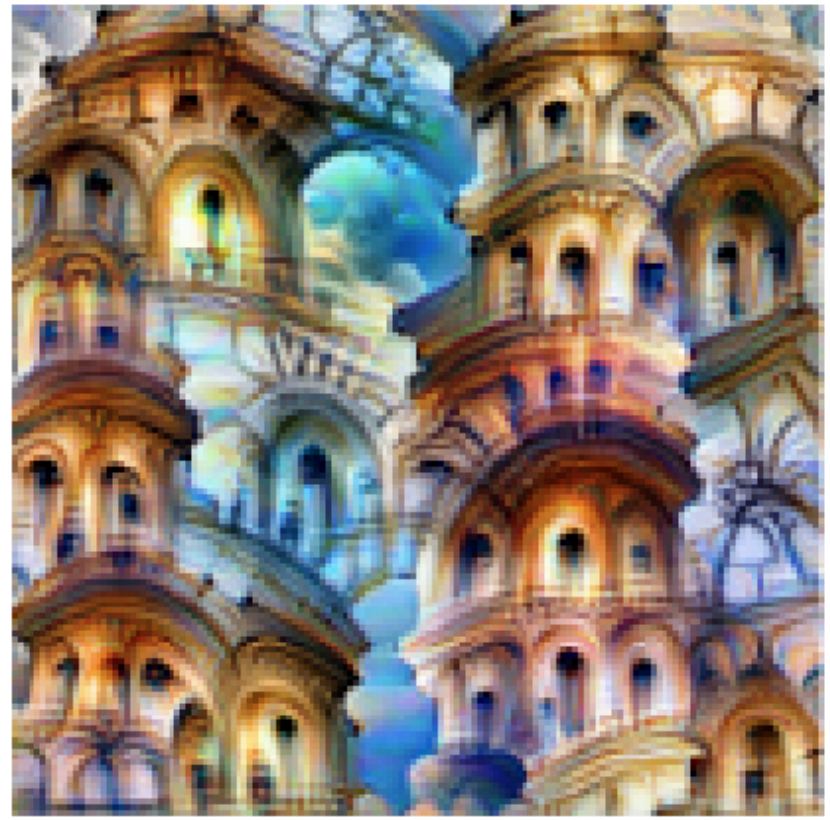


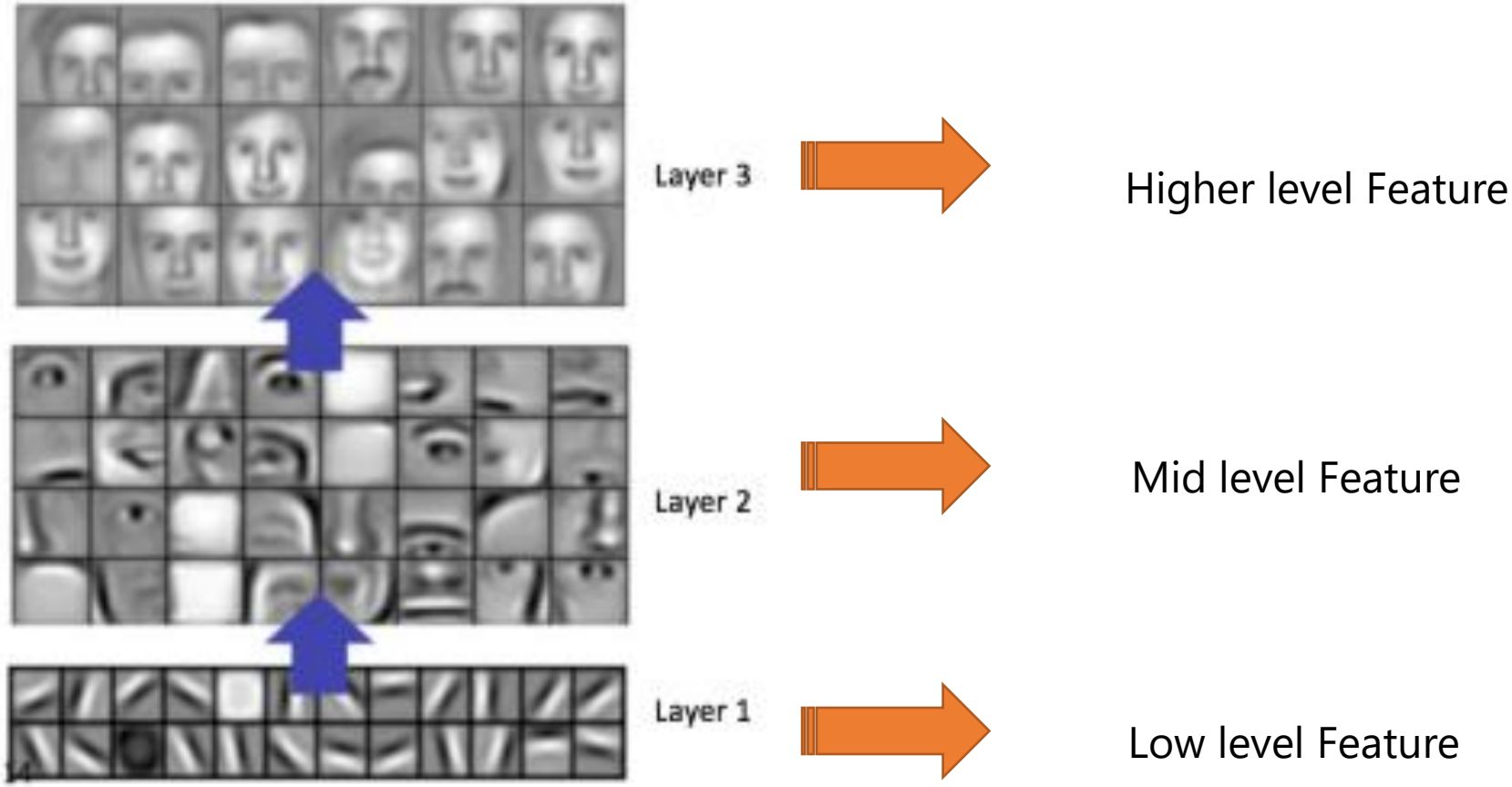
Features in intermediate layers



Features in late layers






Layer 3
Layer 2
Layer 1
Higher level Feature
Mid level Feature
Low level Feature

How deep?

- Depth comes at a computational cost
- May require more data

How many parameters?

Counting parameters

```
model = Sequential()

model.add(Dense(10, activation='relu',
                input_shape=(784,)))

model.add(Dense(10, activation='relu'))

model.add(Dense(3, activation='softmax'))
```

```
# Call the summary method
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 10)                7850
_________________________________________________________________
dense_2 (Dense)              (None, 10)                110
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 33
=================================================================
Total params: 7,993
Trainable params: 7,993
Non-trainable params: 0
_________________________________________________________________
```

Counting parameters

```
model.add(Dense(
        10, activation='relu',
        input_shape=(784,)))
```

$$parameters = 784 * 10 + 10$$

$$= 7850$$

```
model.add(Dense(
        10, activation='relu'))
```

$$parameters = 10 * 10 + 10$$

$$= 110$$

```
model.add(Dense(
        3, activation='softmax'))
```

$$parameters = 10 * 3 + 3$$

$$= 33$$

$$7850 + 110 + 33 = 7993$$

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 10)                7850
_________________________________________________________________
dense_2 (Dense)              (None, 10)                110
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 33
=================================================================
Total params: 7,993
Trainable params: 7,993
Non-trainable params: 0
_________________________________________________________________
```

The number of parameters in a CNN

```
model = Sequential()

model.add(Conv2D(10, kernel_size=3, activation='relu',
                 input_shape=(28, 28, 1), padding='same'))

model.add(Conv2D(10, kernel_size=3, activation='relu',
                 padding='same'))

model.add(Flatten())

model.add(Dense(3, activation='softmax'))
```

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 10)        100
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 10)        910
_________________________________________________________________
flatten_3 (Flatten)          (None, 7840)              0
_________________________________________________________________
dense_4 (Dense)              (None, 3)                 23523
=================================================================
Total params: 24,533
Trainable params: 24,533
Non-trainable params: 0
_________________________________________________________________
```

The number of parameters in a CNN

```
model.add(
 Conv2D(10, kernel_size=3,
        activation='relu',
        input_shape=(28, 28, 1),
        padding='same'))
model.add(
 Conv2D(10, kernel_size=3,
        activation='relu',
        padding='same'))


model.add(Flatten())
```

$$parameters = 9 * 10 + 10$$

$$= 100$$

.

$$parameters = 10 * 9 * 10 + 10$$

$$= 910$$

$$parameters = 0$$

```
model.add(Dense(
       3, activation='softmax'))
```

$$parameters = 7840 * 3 + 3$$

$$= 23523$$

$$100 + 910 + 0 + 23523 = 24533$$

Increasing the number of units in each layer

```
model = Sequential()

model.add(Dense(5, activation='relu',
                input_shape=(784,), padding='same'))

model.add(Dense(15, activation='relu', padding='same'))

model.add(Dense(3, activation='softmax'))
```

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 5)                 3925
_________________________________________________________________
dense_2 (Dense)              (None, 15)                90
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 48
=================================================================
Total params: 4,063
Trainable params: 4,063
Non-trainable params: 0
_________________________________________________________________
```

Increasing the number of units in each layer

```
model = Sequential()

model.add(Conv2D(5, kernel_size=3, activation='relu',
                 input_shape=(28, 28, 1),
                 padding="same"))

model.add(Conv2D(15, kernel_size=3, activation='relu',
                 padding="same"))

model.add(Flatten())

model.add(Dense(3, activation='softmax'))
```

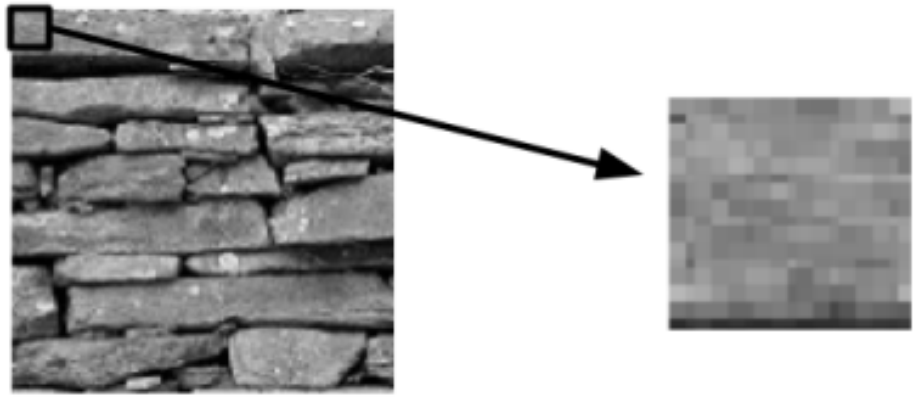
```
model.summary()
```

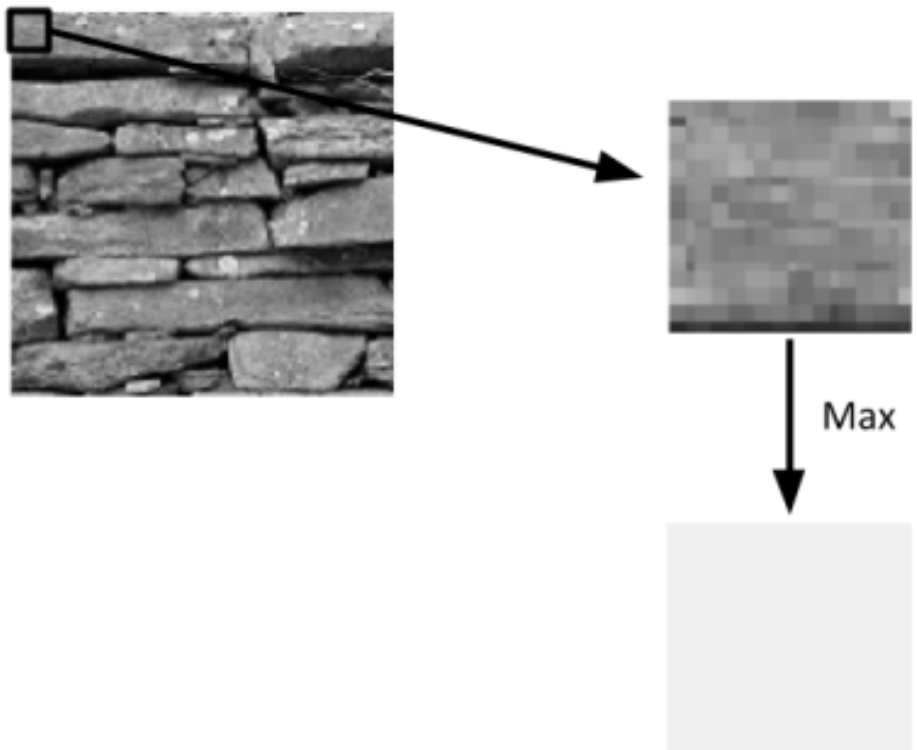
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 28, 28, 5)         50
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 28, 28, 15)        690
_________________________________________________________________
flatten_6 (Flatten)          (None, 11760)             0
_________________________________________________________________
dense_9 (Dense)              (None, 3)                 35283
=================================================================
Total params: 36,023
Trainable params: 36,023
Non-trainable params: 0
_________________________________________________________________
```

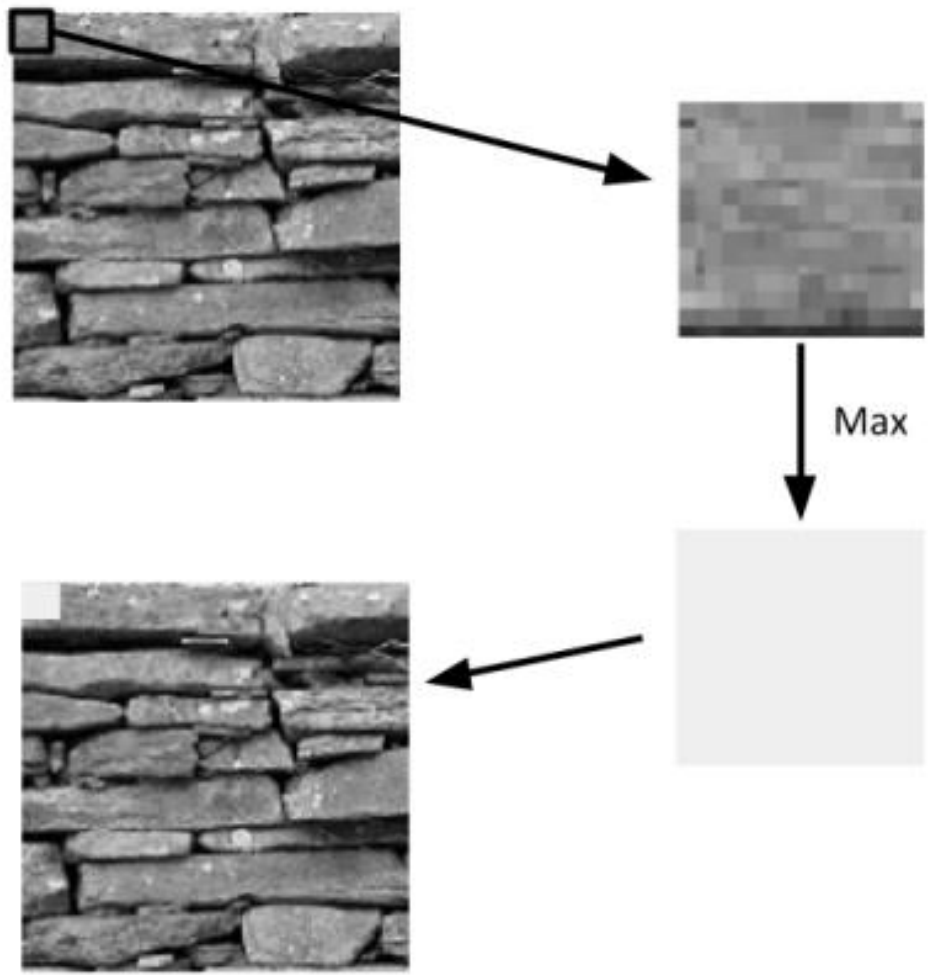
Reducing parameters with pooling

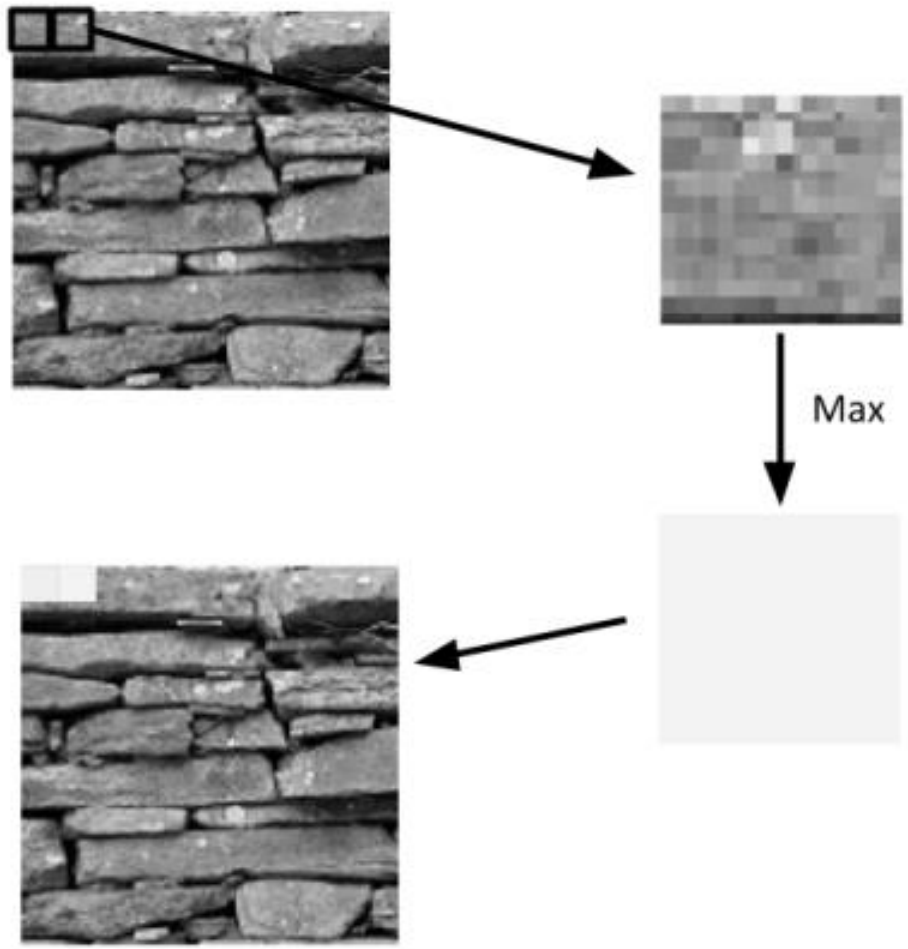
```
model.summary()
```

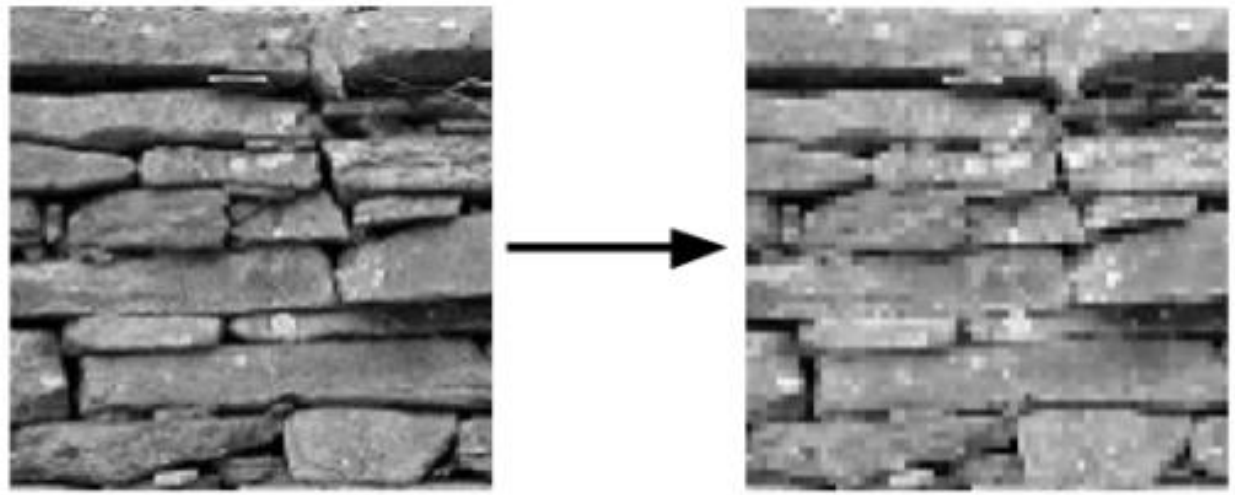
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 28, 28, 5)         50
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 28, 28, 15)        690
_________________________________________________________________
flatten_6 (Flatten)          (None, 11760)             0
_________________________________________________________________
dense_9 (Dense)              (None, 3)                 35283
=================================================================
Total params: 36,023
Trainable params: 36,023
Non-trainable params: 0
_________________________________________________________________
```



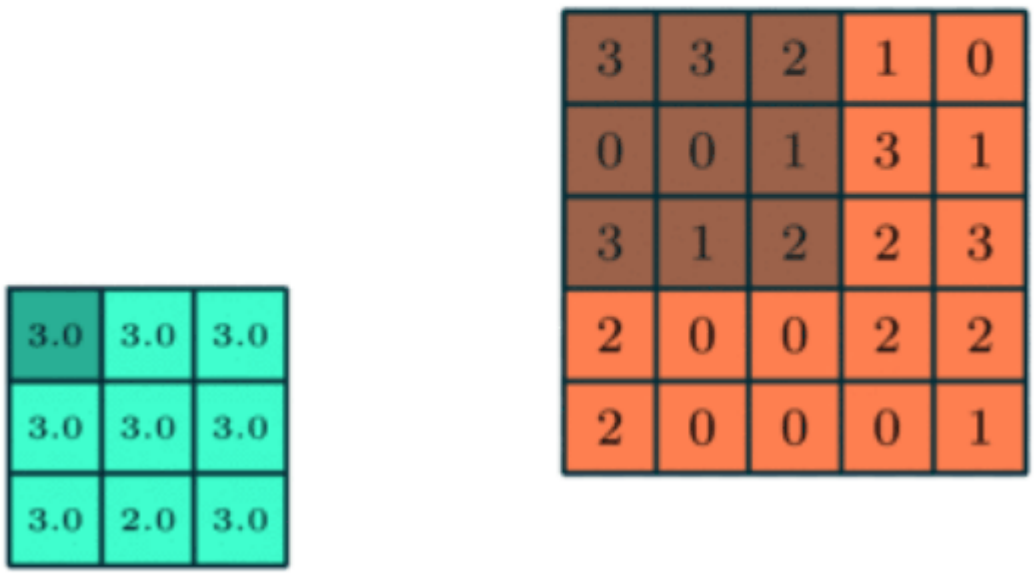

Max

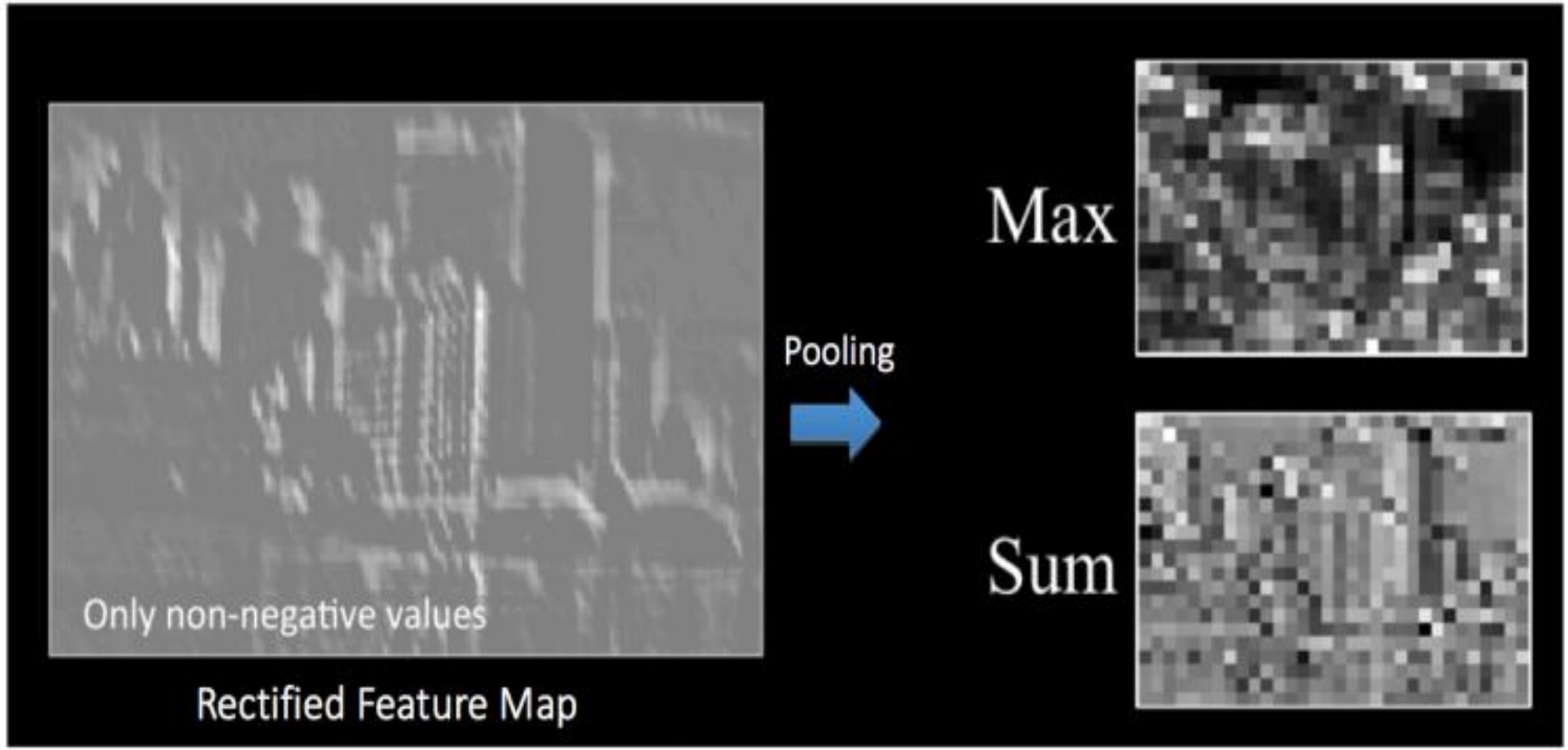

Max


Max



Pooling Layer



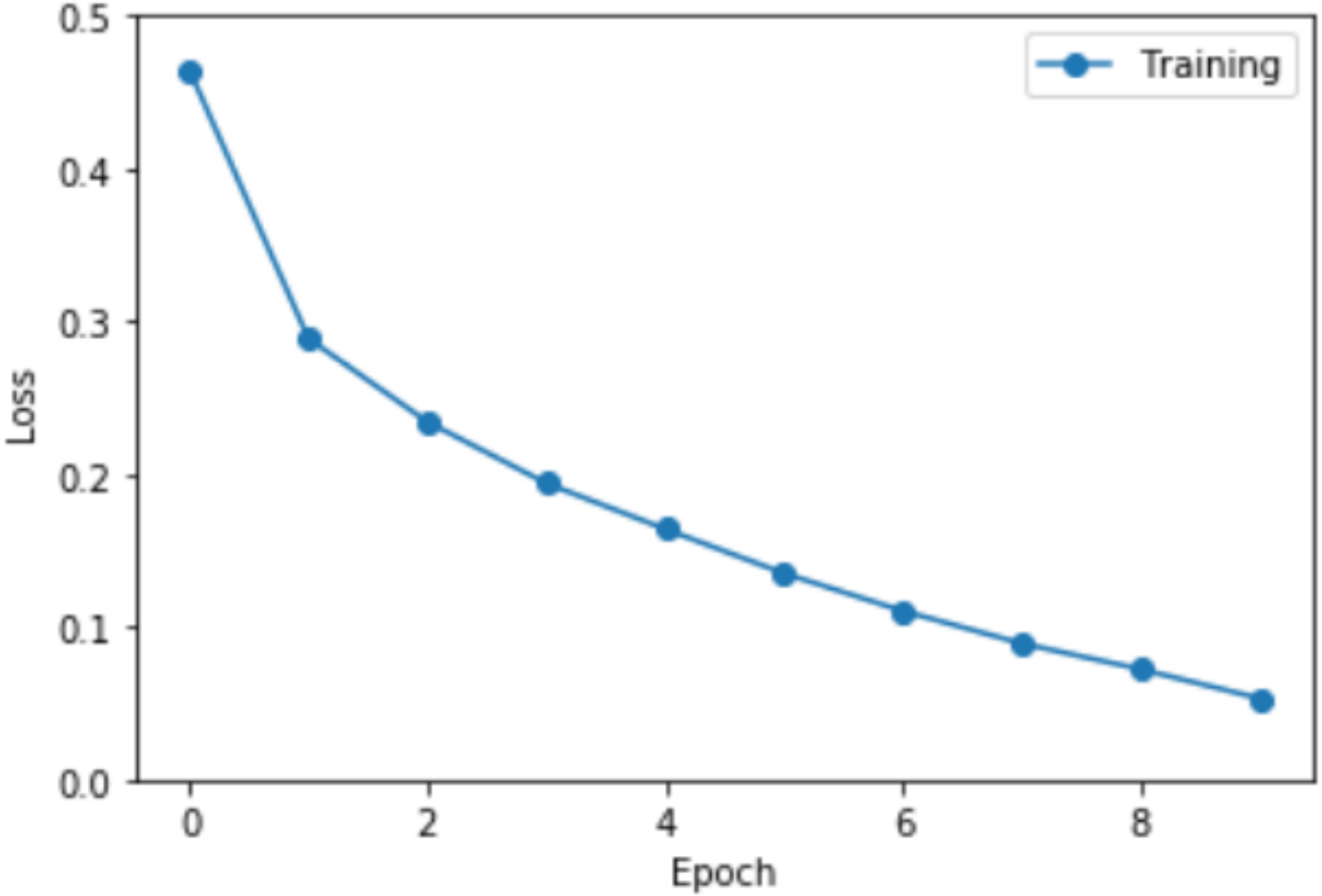

Only non-negative values
Rectified Feature Map
Pooling
Max
Sum

Max pooling in Keras

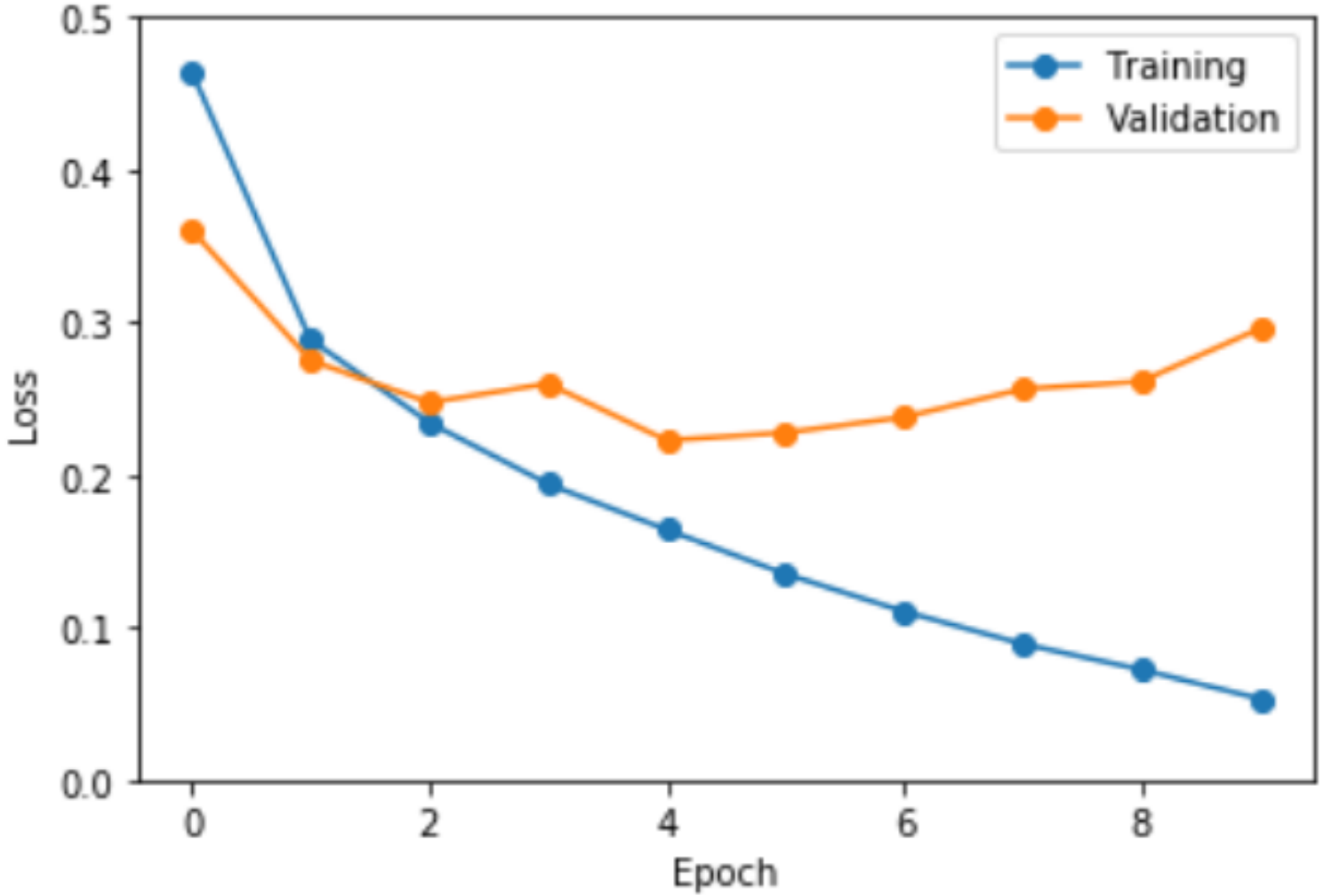
```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten, MaxPool2D
model = Sequential()
model.add(Conv2D(5, kernel_size=3, activation='relu',
                 input_shape=(img_rows, img_cols, 1)))
model.add(MaxPool2D(2))
model.add(Conv2D(15, kernel_size=3, activation='relu',
                 input_shape=(img_rows, img_cols, 1)))
model.add(MaxPool2D(2))
model.add(Flatten())
model.add(Dense(3, activation='softmax'))
```

Tracking learning

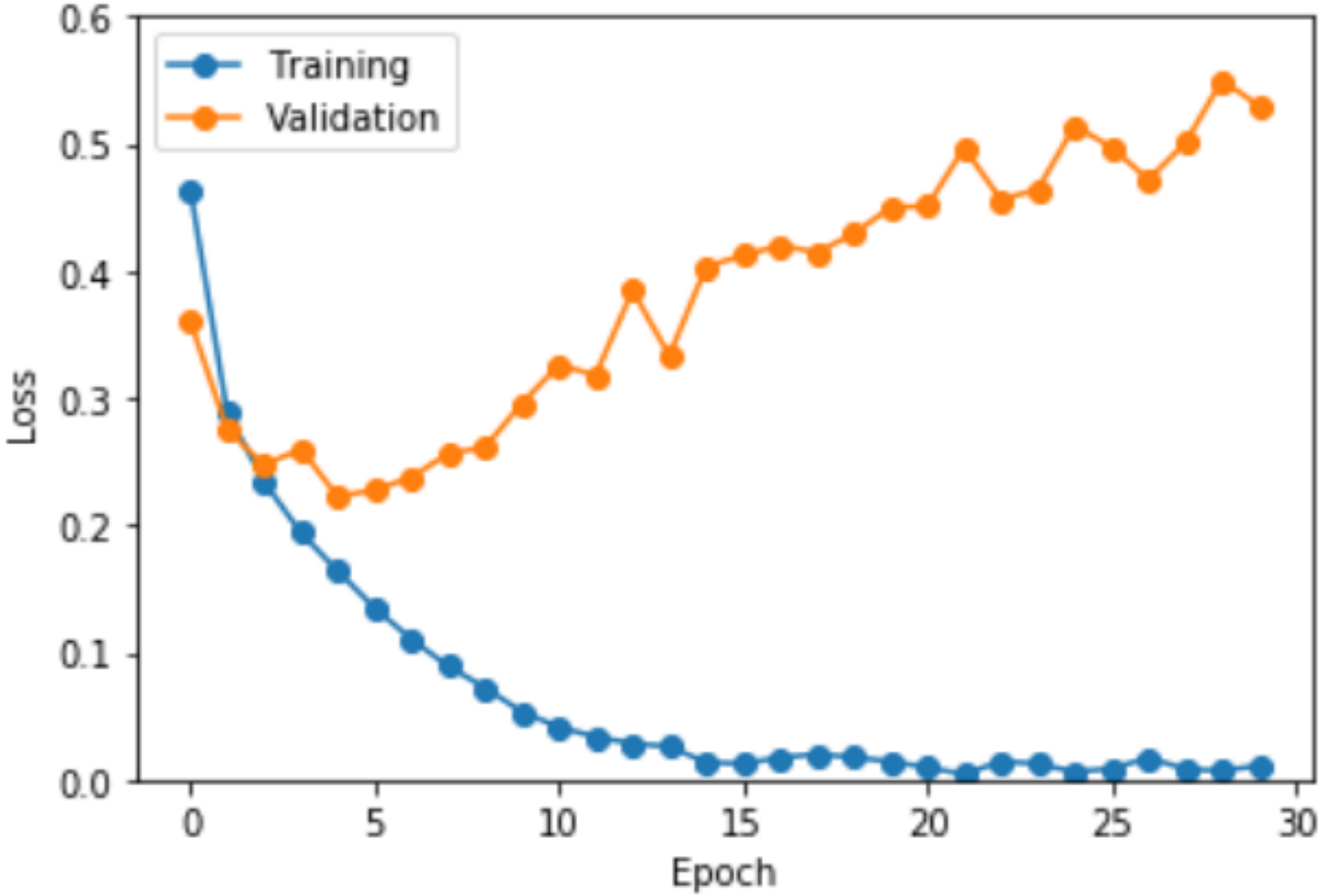
Learning curves: training



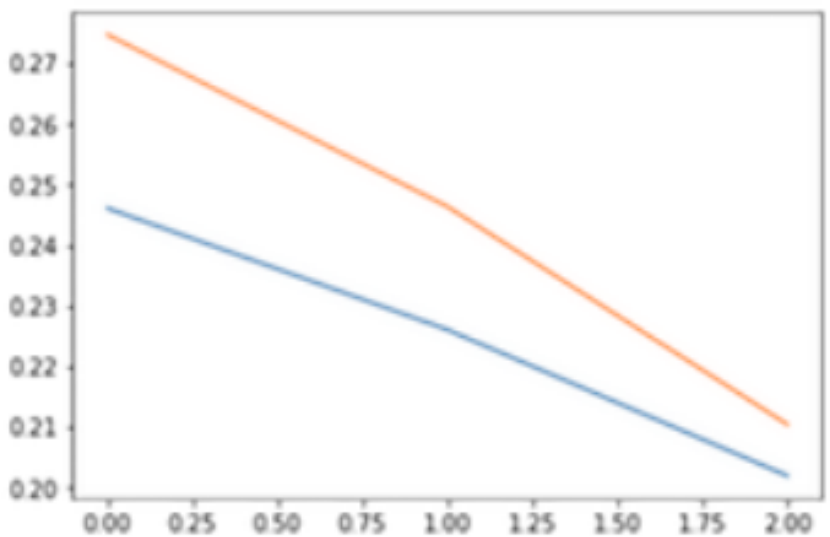
Learning curves: validation



Learning curves: overfitting



```
training = model.fit(train_data, train_labels,
                     epochs=3, validation_split=0.2)
import matplotlib.pyplot as plt
plt.plot(training.history['loss'])
plt.plot(training.history['val_loss'])
plt.show()
```



Storing the optimal parameters

```
from keras.callbacks import ModelCheckpoint

# This checkpoint object will store the model parameters
# in the file "weights.hdf5"
checkpoint = ModelCheckpoint('weights.hdf5', monitor='val_loss',
                             save_best_only=True)
# Store in a list to be used during training
callbacks_list = [checkpoint]
# Fit the model on a training set, using the checkpoint as a
#callback
model.fit(train_data, train_labels, validation_split=0.2,
          epochs=3, callbacks=callbacks_list)
```

Loading stored parameters

```
model.load_weights('weights.hdf5')
model.predict_classes(test_data)
array([2, 2, 1, 2, 0, 1, 0, 1, 2, 0])
```

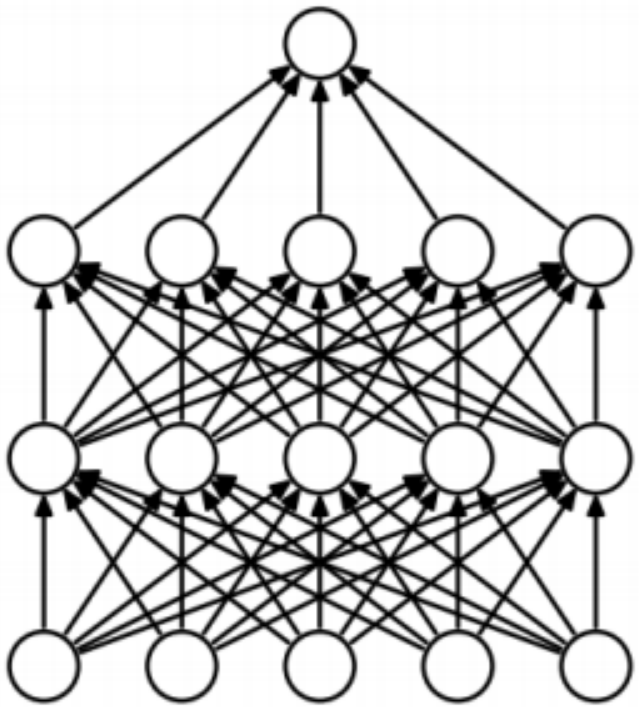
Neural network regularization

Dropout

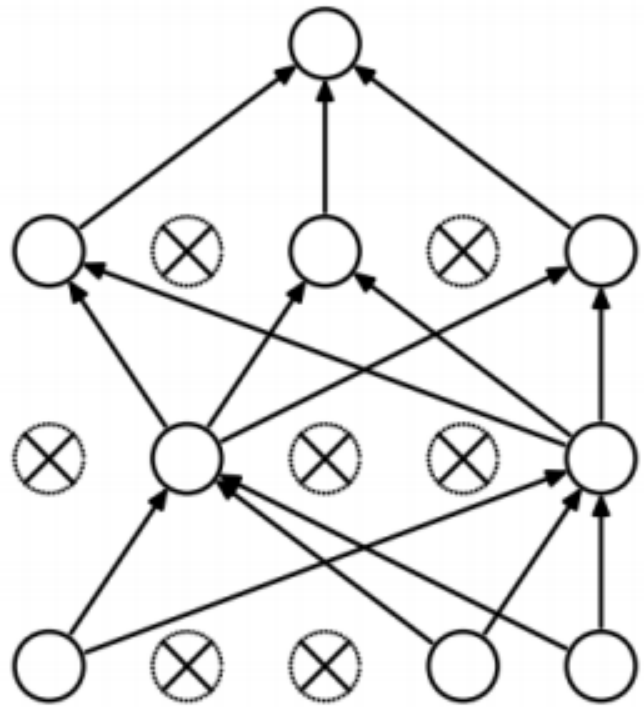
In each learning step:

- Select a subset of the units
- Ignore it in the forward pass
- And in the back-propagation of error

Dropout



(a) Standard Neural Net



(b) After applying dropout.

Dropout in Keras

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten, Dropout
model = Sequential()
model.add(Conv2D(5, kernel_size=3, activation='relu',
                 input_shape=(img_rows, img_cols, 1)))
model.add(Dropout(0.25))
model.add(Conv2D(15, kernel_size=3, activation='relu'))
model.add(Flatten())
model.add(Dense(3, activation='softmax'))
```

Batch normalization

- Rescale the outputs

Batch Normalization in Keras

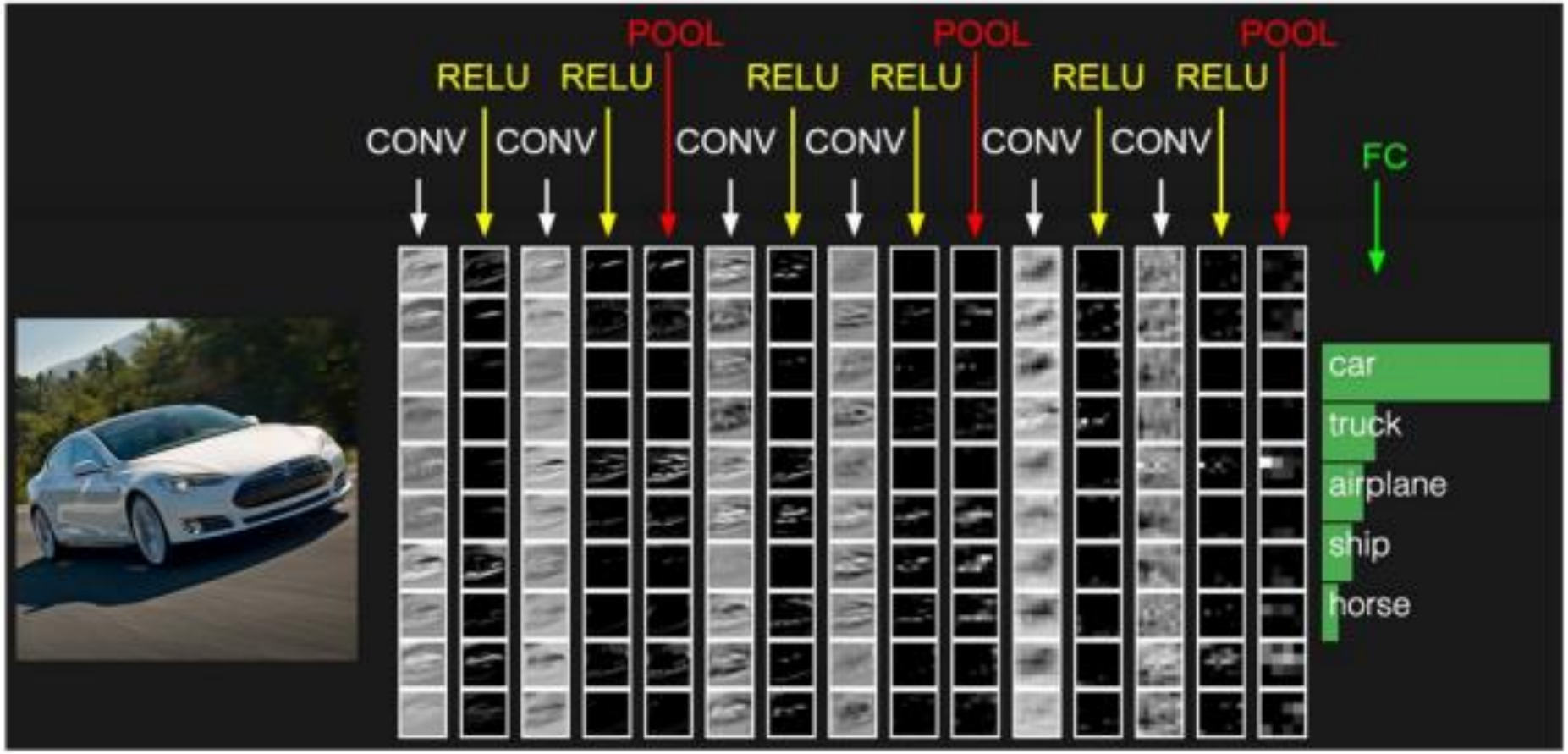
```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten, BatchNormalization

model = Sequential()
model.add(Conv2D(5, kernel_size=3, activation='relu',
                 input_shape=(img_rows, img_cols, 1)))
model.add(BatchNormalization())
model.add(Conv2D(15, kernel_size=3, activation='relu'))
model.add(Flatten())
model.add(Dense(3, activation='softmax'))
```

Be careful when using them together!

The disharmony between dropout and batch normalization

Interpreting the model

CNN Visualization



What did we learn?

- Image classification
- Convolutions
- Reducing the number of parameters
  - Tweaking your convolutions
  - Adding pooling layers
- Improving your network
  - Regularization
- Understanding your network
  - Monitoring learning
  - Interpreting the parameters