Deep Learning with Keras

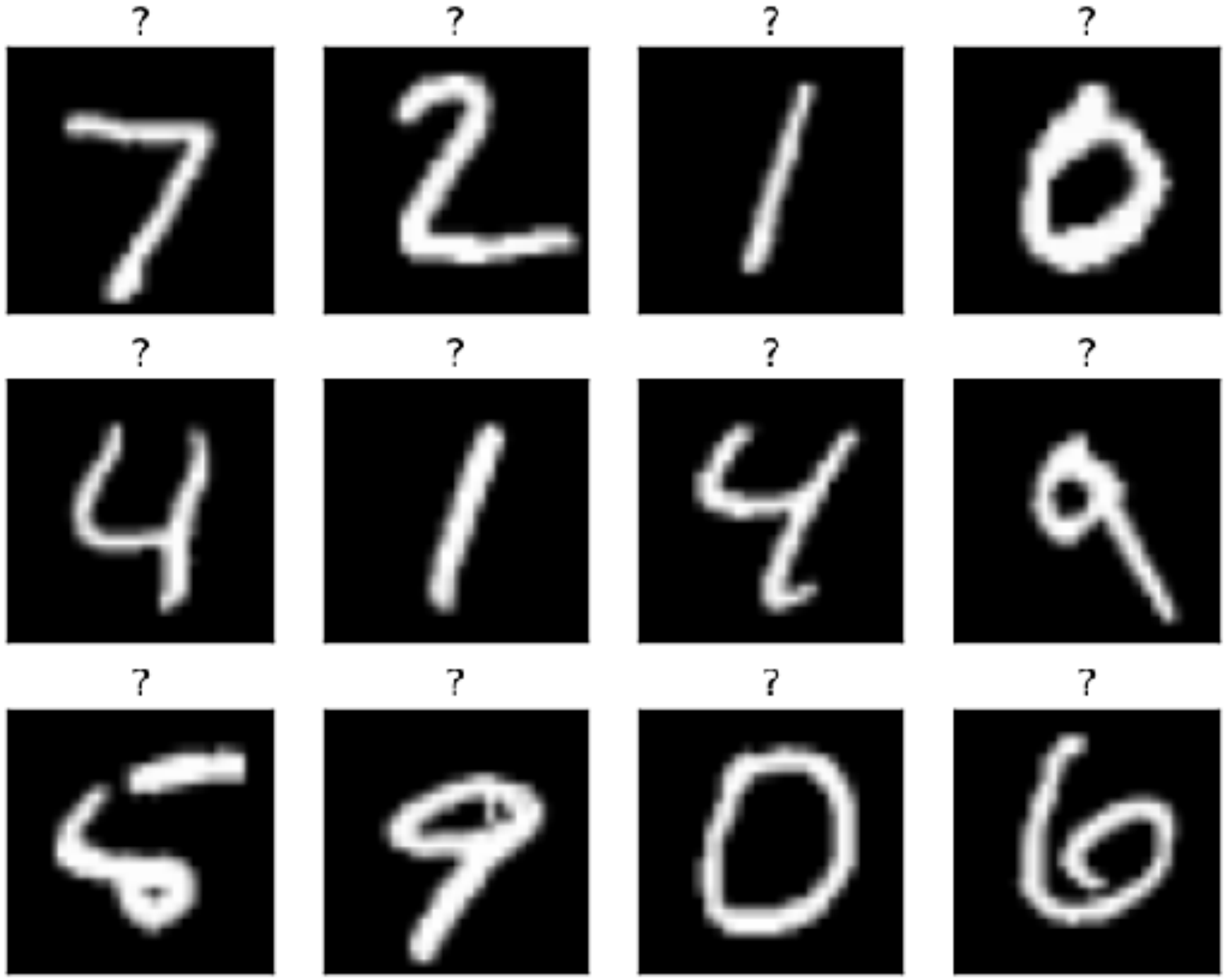
Agis Chartsias
IDCOM, School of Engineering, University of Edinburgh

Keras

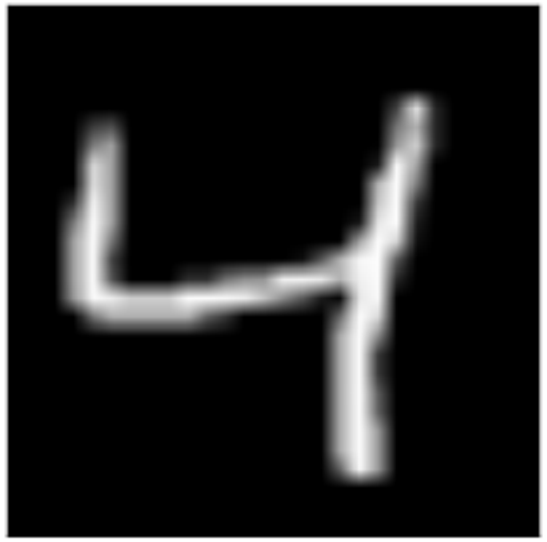
- Deep learning library in Python
- API for Theano, Tensorflow backend
- Easy to use, modular, extensible

Let’s start with an example

- 28 x 28 pixels
- 10 classes
- 60000 training, 10000 testing examples
- Find the regions in the 784 dim space that correspond to the digits



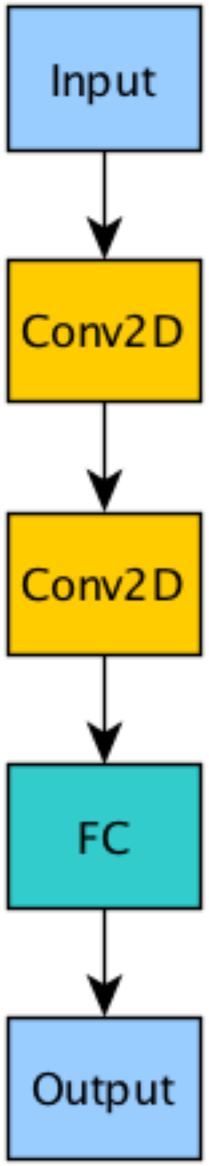
MNIST Example



```
[0, 0, 0,  0,  0,   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0,  0,  0,   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0,  0,  0,   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0,  0,  0,   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0,  0,  0,   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0,  0,  0,   0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 67, 232, 39, 0, 0, 0, 0, 0],
[0, 0, 0,  0,  62,  81, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,120, 180, 39, 0, 0, 0, 0, 0],
[0, 0, 0,  0,  126, 163, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 153, 210, 40, 0, 0, 0, 0, 0],
[0, 0, 0,  0,  220, 163, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 27, 254, 162, 0, 0, 0, 0, 0, 0],
[0, 0, 0,  0,  222, 163, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 183, 254, 125, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 46,  245, 163, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 198, 254, 56, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 120, 254, 163, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 23, 231, 254, 29, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 159, 254, 120, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 163, 254, 216, 16, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 159, 254, 67, 0, 0, 0, 0, 0, 0, 0, 0, 0, 14, 86, 178, 248, 254, 91, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 159, 254, 85, 0, 0, 0, 47, 49, 116, 144, 150, 241, 243, 234, 179, 241, 252, 40, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 150, 253, 237, 207, 207, 207, 253, 254, 250, 240, 198, 143, 91, 28, 5, 233, 250, 0, 0, 0, 0,0,0,0,0],
[0, 0, 0,  0, 119, 177, 177, 177, 177, 177, 98, 56, 0, 0, 0, 0, 0, 102, 254, 220, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0,  0,  0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 169, 254, 137, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0,  0,  0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 169, 254, 57, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0,  0,  0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 169, 254, 57, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0,  0,  0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 169, 255, 94, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0,  0,  0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 169, 254, 96, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0,  0,  0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 169, 254, 153, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0,  0,  0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 169, 255, 153, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0,  0,  0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 96, 254, 153, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0,  0,  0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0,  0,  0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0,  0,  0,  0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
```

MNIST Classifier

→ 4



Input

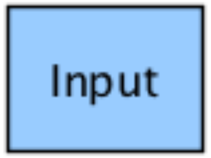
Conv2D

Conv2D

FC

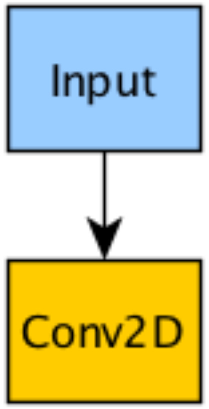
Output

MNIST Classifier

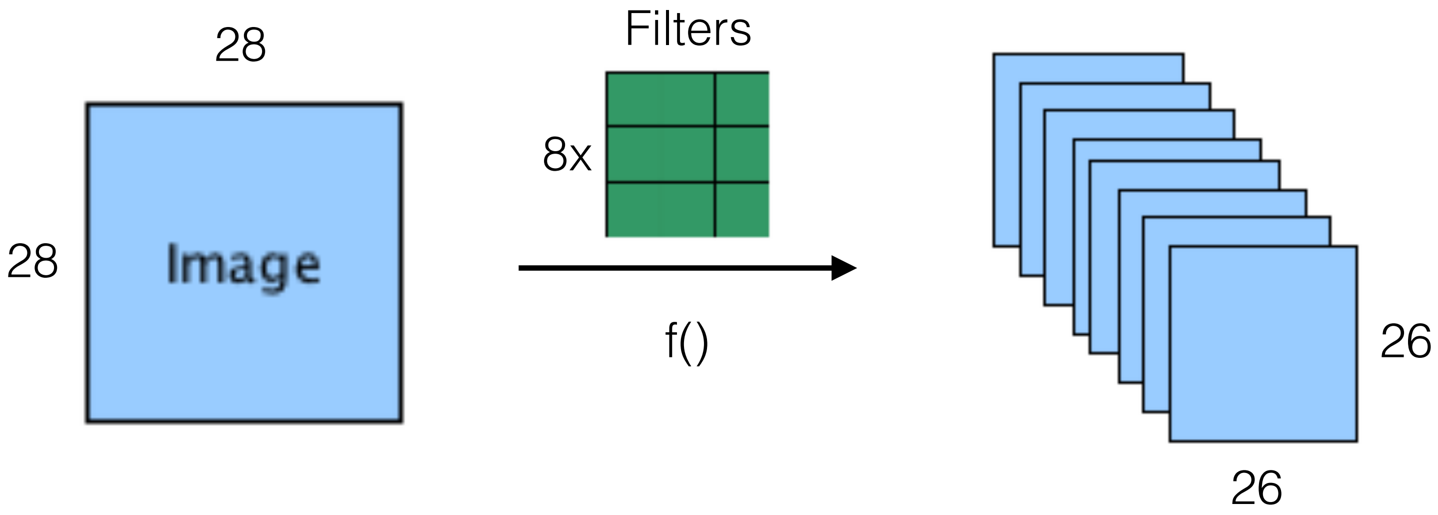


```
from keras.datasets import mnist
(x_train, y_train), (x_test, y_test) =
mnist.load_data()

inp = Input(shape=(1,28,28))
```

MNIST Classifier

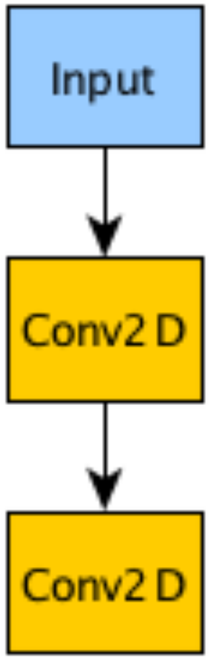


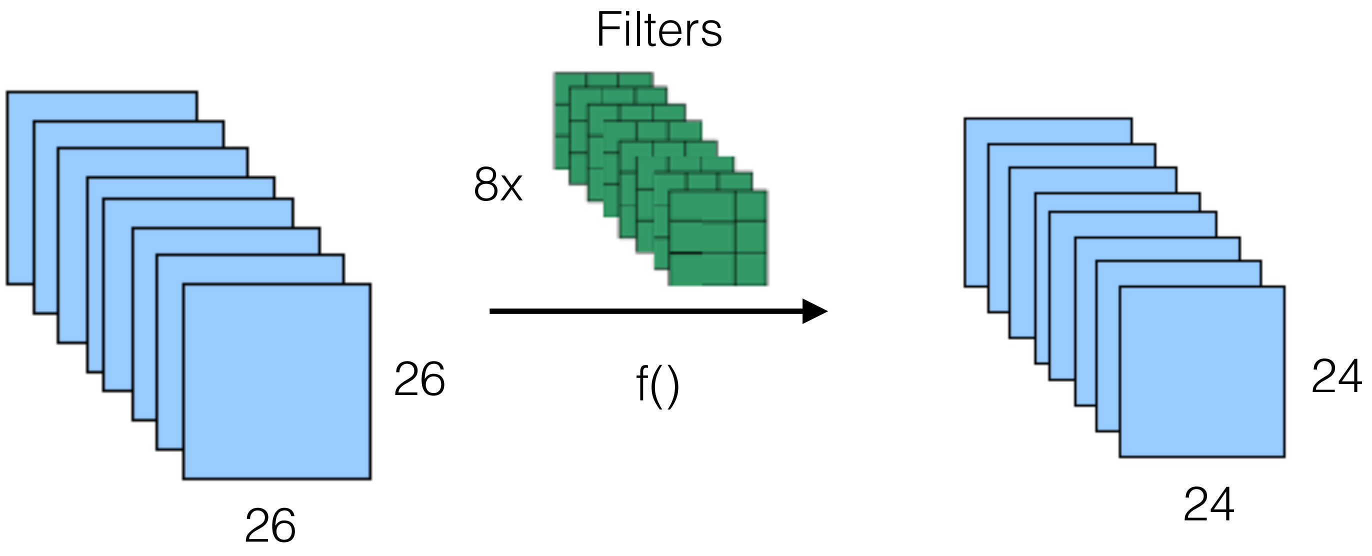
```
from keras.datasets import mnist
(x_train, y_train), (x_test, y_test) =
mnist.load_data()

inp = Input(shape=(1,28,28))
L1 = Conv2D(8, 3, activation='relu')(inp)
```



MNIST Classifier

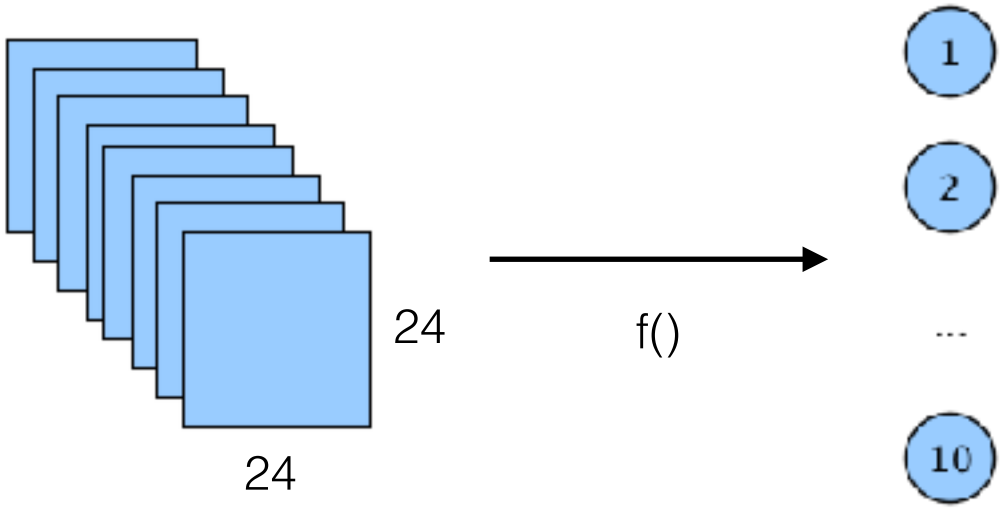
```
from keras.datasets import mnist
(x_train, y_train), (x_test, y_test) =
mnist.load_data()

inp = Input(shape=(1,28,28))
L1 = Conv2D(8, 3, activation='relu')(inp)
L2 = Conv2D(8, 3, activation='relu')(L1)
```

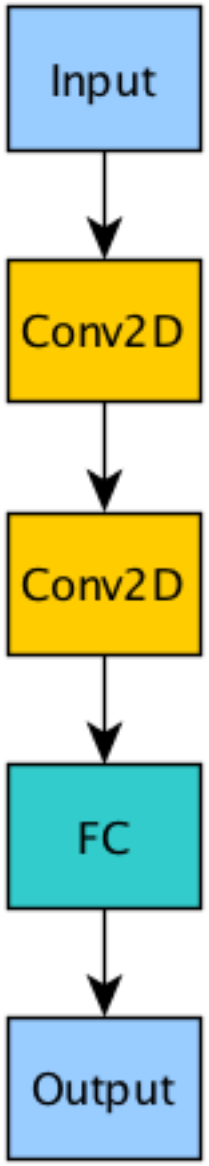




MNIST Classifier

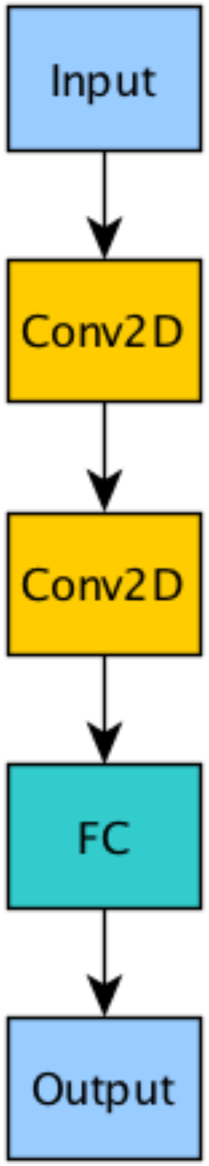
```
from keras.datasets import mnist
(x_train, y_train), (x_test, y_test) =
mnist.load_data()

inp = Input(shape=(1,28,28))
L1 = Conv2D(8, 3, activation='relu')(inp)
L2 = Conv2D(8, 3, activation='relu')(L1)
L3 = Flatten()(L2)
out = Dense(10, activation='softmax')(L3)
```





MNIST Classifier

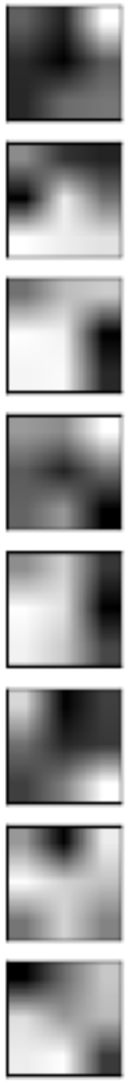
```
from keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()

inp = Input(shape=(1,28,28))
L1 = Conv2D(8, 3, activation='relu')(inp)
L2 = Conv2D(8, 3, activation='relu')(L1)
L3 = Flatten()(L2)
out = Dense(10, activation='softmax')(L3)

model = Model(inputs=inp, outputs=out)
model.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=[‘acc'])

h = model.fit(x_train, y_train, validation_split=0.1,
batch_size=32, epochs=5)
```

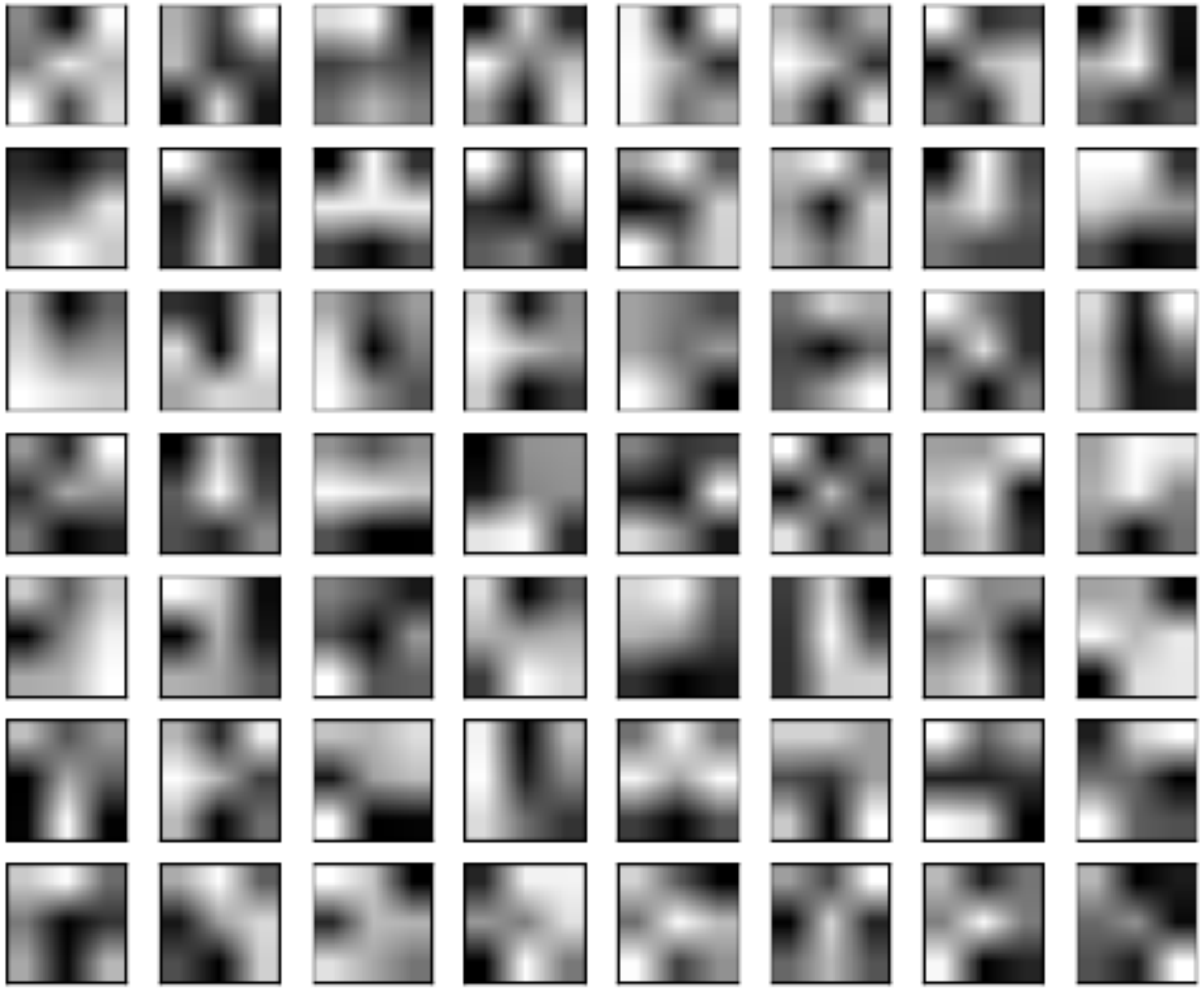


- What is epoch?
- Batch size?

MNIST Classifier

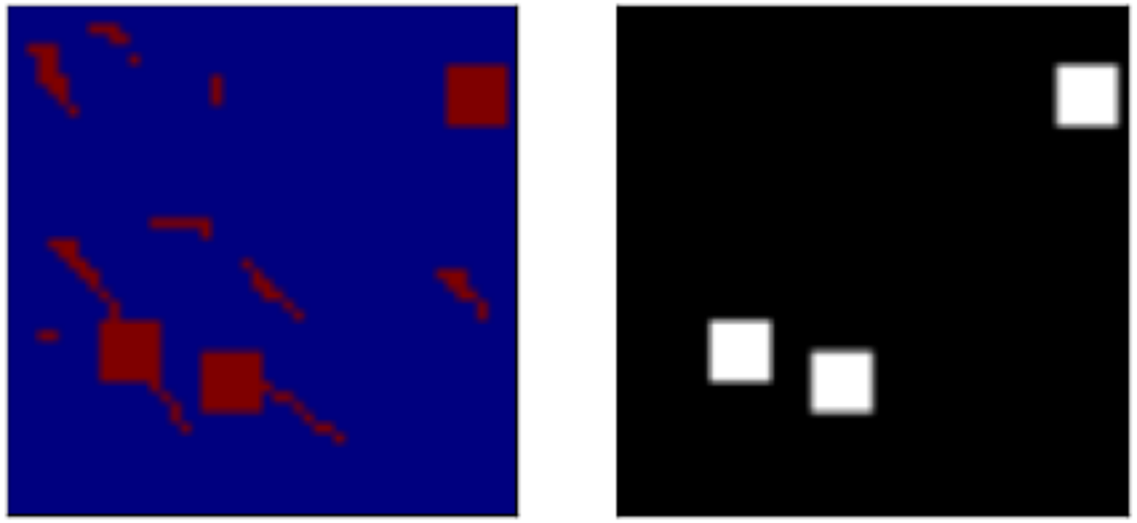
```
w1=model.layers[1].get_weights() # shape: (3,3,1,8)
w2=model.layers[2].get_weights() # shape: (3,3,8,8)
```





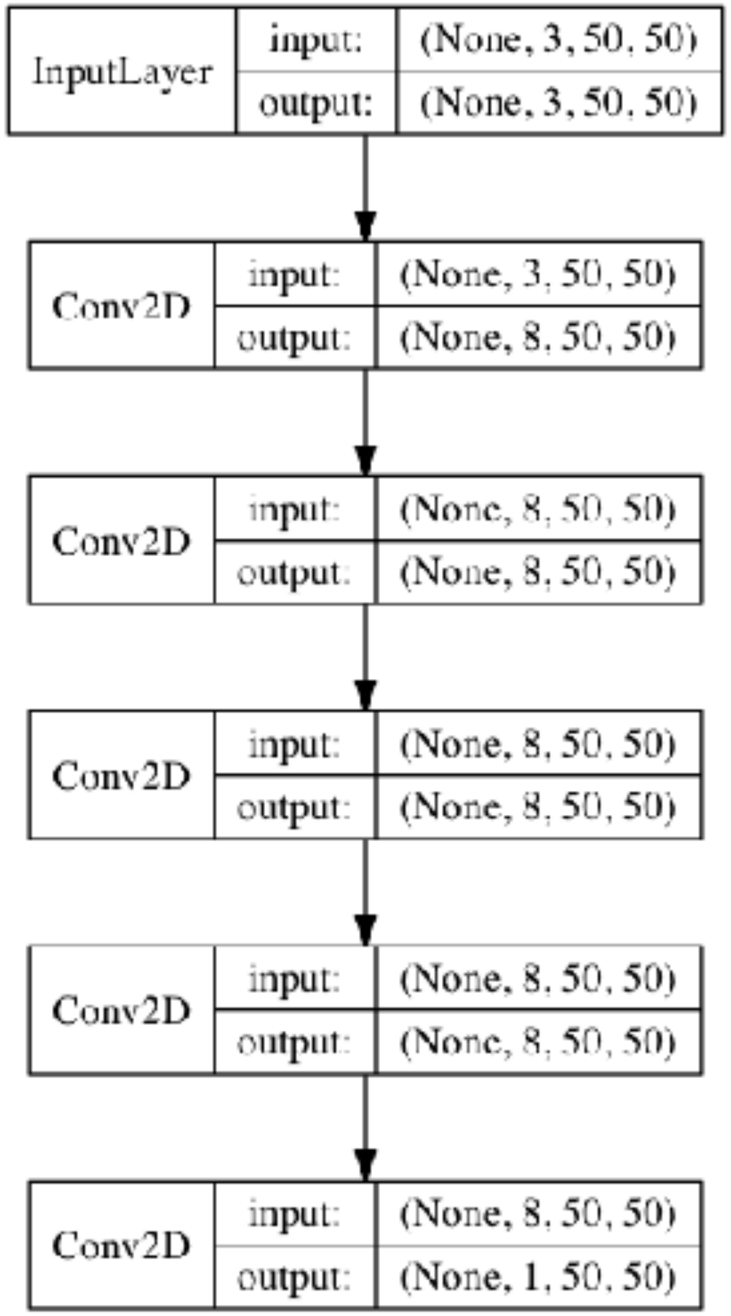
Segmentation

- Randomly generated dataset
- 2000 examples
- 50 x 50 pixels
- Segment rectangles



Segmentation

```
inp = Input(shape=(3,50,50))
L1 = Conv2D(8, 3, activation='relu',
           padding='same')(inp)
L2 = Conv2D(8, 3, activation='relu',
          padding='same')(L2)
L3 = Conv2D(8, 3, activation='relu',
          padding=‘same')(L2)
L4 = Conv2D(8, 3, activation='relu',
          padding='same')(L3)
out = Conv2D(1, 1, activation='sigmoid',
             padding='same')(L4)

model = Model(inputs=inp, outputs=out)
model.compile(loss=dsc_loss,
optimizer='adam',
               metrics=[‘acc’, dsc])

h = model.fit(x, y, validation_split=0.1,
batch_size=32, epochs=5)
```



InputLayer | input: (None, 3, 50, 50) | output: (None, 3, 50, 50)

Conv2D | input: (None, 3, 50, 50) | output: (None, 8, 50, 50)

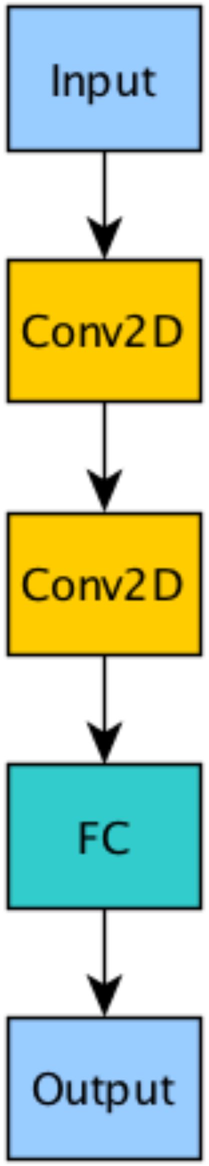
Conv2D | input: (None, 8, 50, 50) | output: (None, 8, 50, 50)

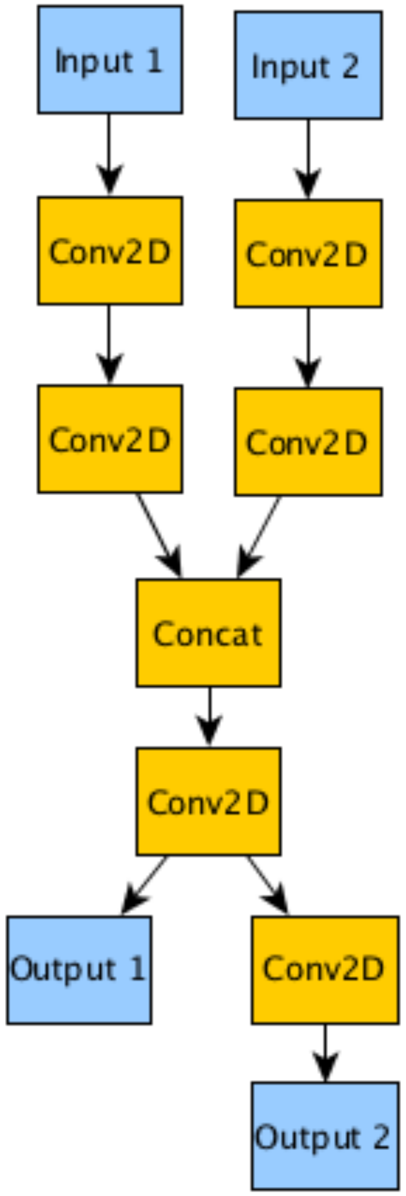
Conv2D | input: (None, 8, 50, 50) | output: (None, 8, 50, 50)

Conv2D | input: (None, 8, 50, 50) | output: (None, 8, 50, 50)

Conv2D | input: (None, 8, 50, 50) | output: (None, 1, 50, 50)

Architectures





Basic Layers

- Fully Connected

```
Dense(dims, activation)
```

- Convolutional

```
Conv1D(filters, kernel_size, strides, padding, activation)
Conv2D(filters, kernel_size, strides, padding, activation)
```

- Pooling and upsampling

```
MaxPooling2D(pool_size, strides, padding)
AveragePooling2D(pool_size, strides, padding)
UpSampling2D(size)
```

- Merge

```
Concatenate()
Add()
Multiply()
```

Other Layers

- Dropout
- BatchNormalization
- Activation: linear, relu, LeakyRelu, sigmoid, tanh, etc
- Noise
- SimpleRNN
- Lambda

Summary

- Create a model

```
model = Model(inputs=[inp1, inp2], outputs=[out1, out2])
```

- Configure learning

```
model.compile(loss={out1: 'mse', out2:'categorical_crossetropy'},
              loss_weights={'out1': 0.3, 'out2': 0.7},
              optimizer='sgd', metrics=['acc', my_metric_fnc])
```

- Train

```
model.fit(x, y, batch_size, epochs, validation_data)
```

- Test

→ `model.predict` → car

Keras Overview

- Neural Network is a keras Model
- Architecture is a graph of layers
- Configure Model: loss function, optimiser
- Train

Thank You!

https://github.com/agis85/keras_tutorial