Advanced Machine Learning

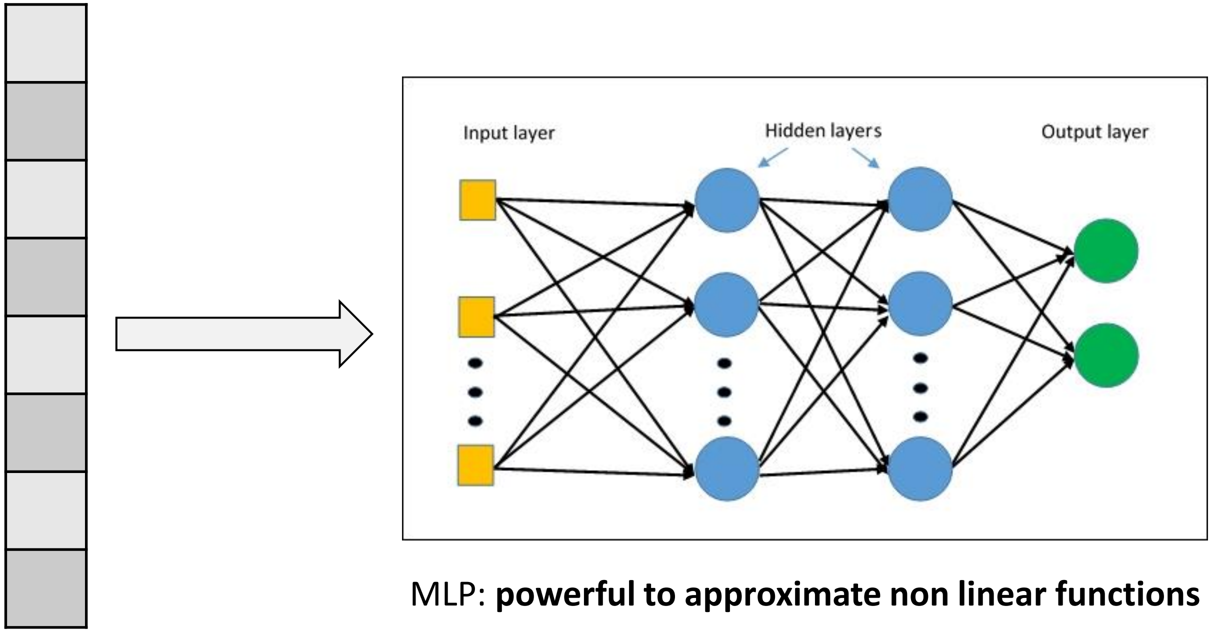
Bilel GUETARNI, PhD

bilel.guetarni@junia.com

JUNIA ISEN

PowerPoint slides from **Halim Benhabiles, PhD** and **Farès BOUGOURZI, PhD**

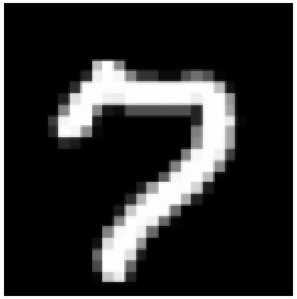
From Artificial Neural Network to Convolutional Neural Network (ANN to CNN)



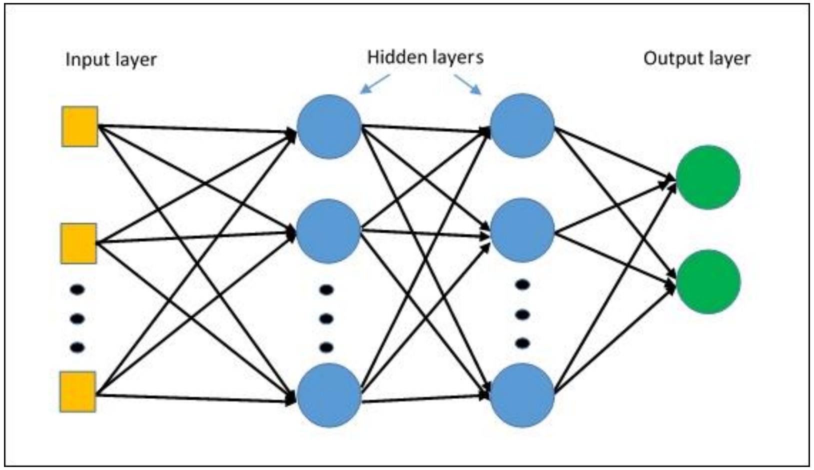
Attribute vector

MLP: **powerful to approximate non linear functions**

From Artificial Neural Network to Convolutional Neural Network (ANN to CNN)



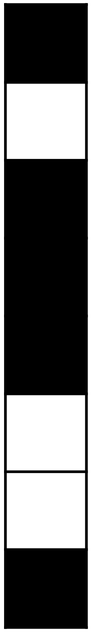
Conversion in one dimension vector

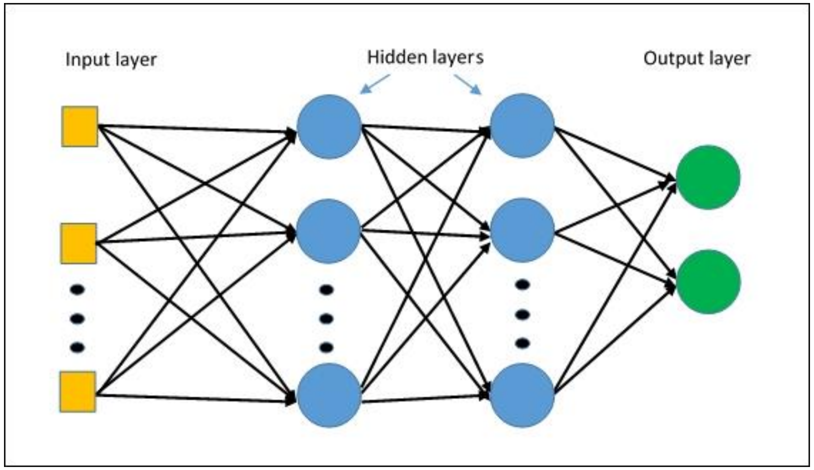


MLP: **powerful to approximate non linear functions**

From Artificial Neural Network to Convolutional Neural Network (ANN to CNN)

Conversion in one dimension vector

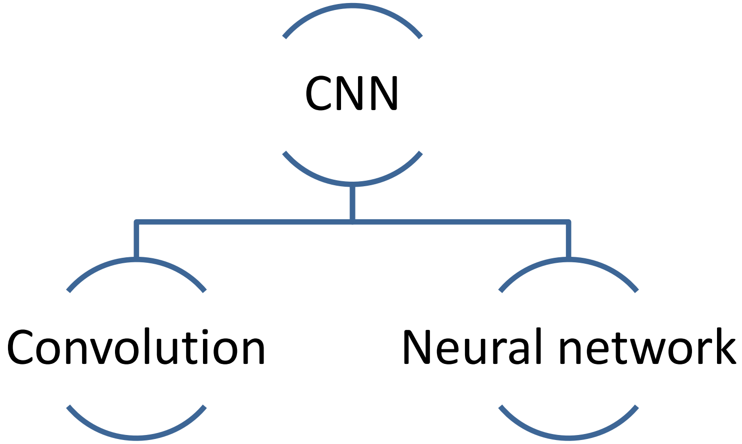




MLP: **powerful to approximate non linear functions**

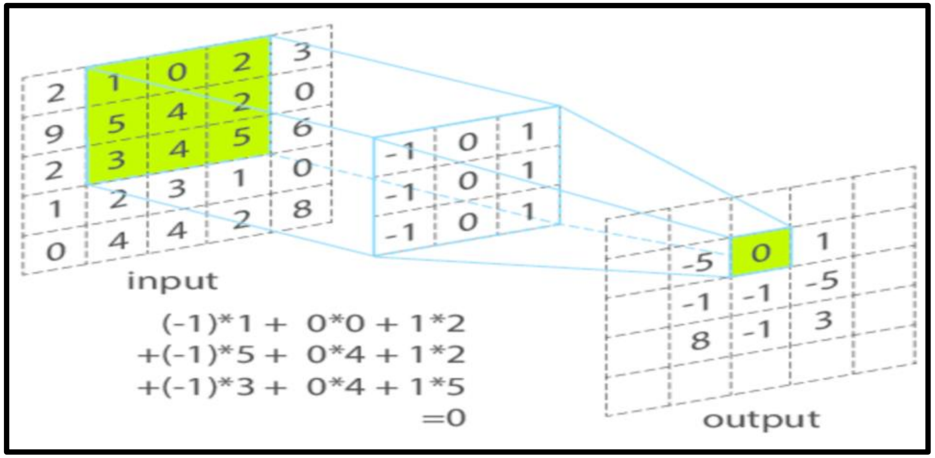
Spatial information loss (pixels arrangement)

From Artificial Neural Network to Convolutional Neural Network (a deep learning model)

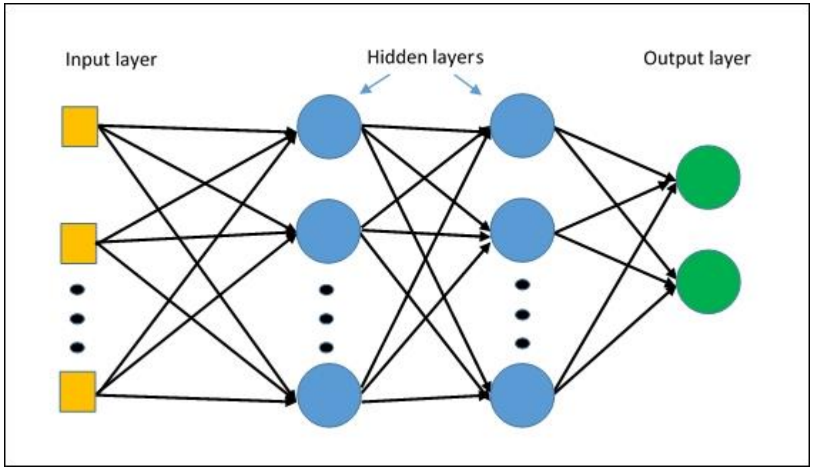

CNN

Convolution

Neural network



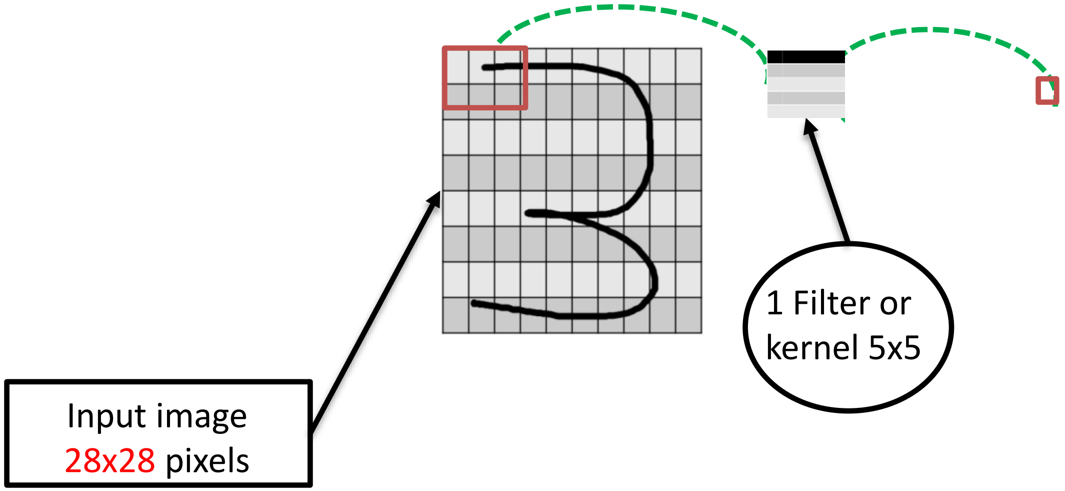
✚



https://perso.esiee.fr/~perretb/I5FM/TAI/convolution/index.html

From Artificial Neural Network to Convolutional Neural Network (a deep learning model)

- Convolution in neural network

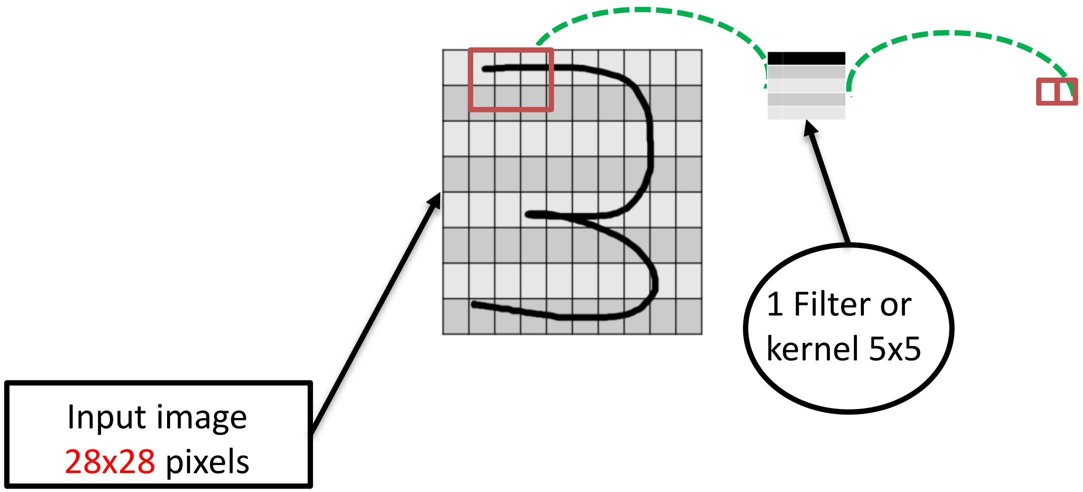


Input image
28x28 pixels

1 Filter or
kernel 5x5

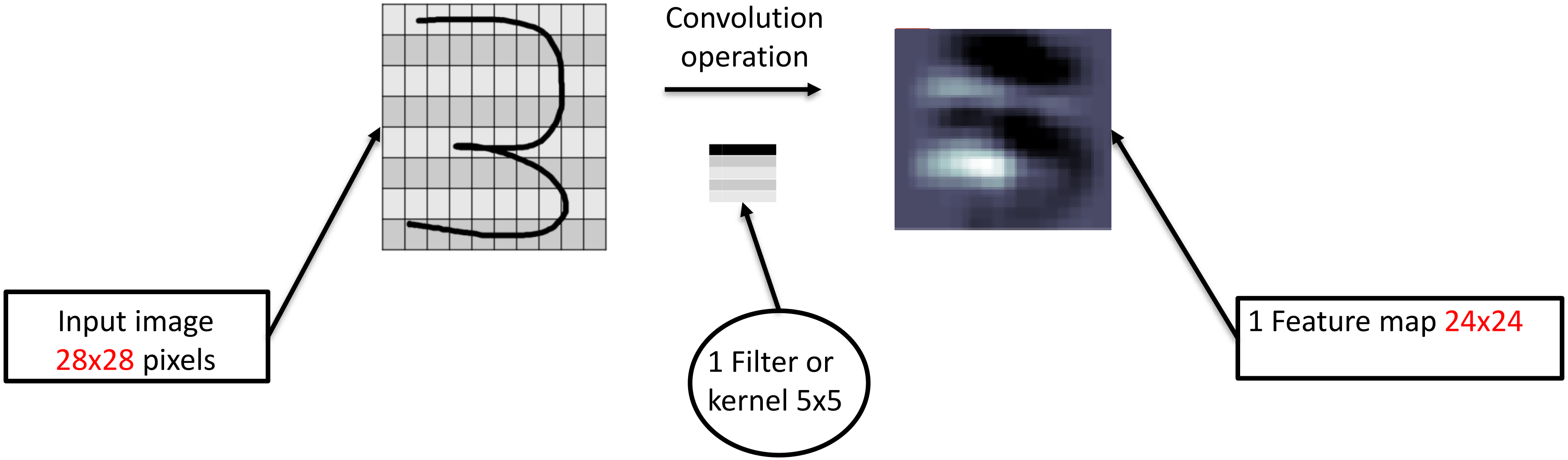
From Artificial Neural Network to Convolutional Neural Network (a deep learning model)

- Convolution in neural network



From Artificial Neural Network to Convolutional Neural Network (a deep learning model)

- Convolution in neural network



Input image 28x28 pixels

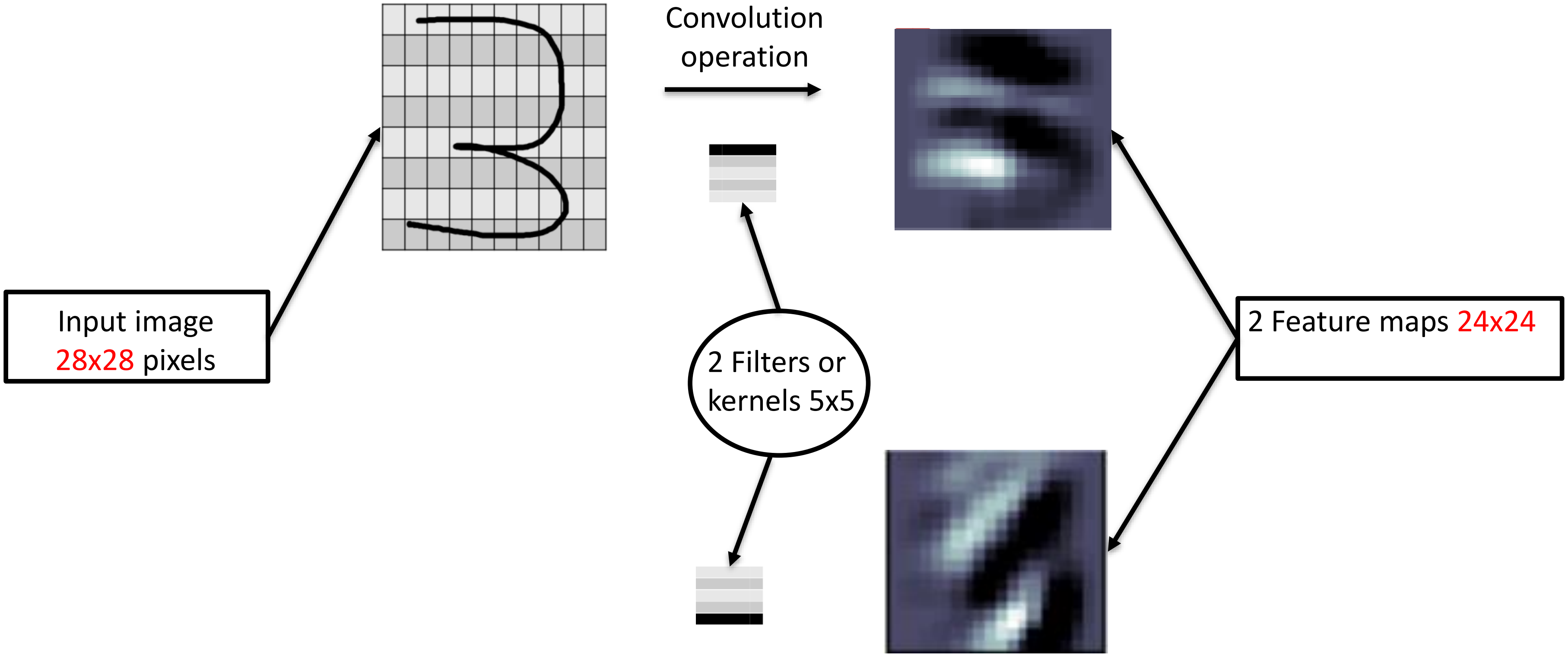
Convolution operation

1 Filter or kernel 5x5

1 Feature map 24x24

From Artificial Neural Network to Convolutional Neural Network (a deep learning model)

- Convolution in neural network



Convolution operation

Input image 28x28 pixels

2 Filters or kernels 5x5

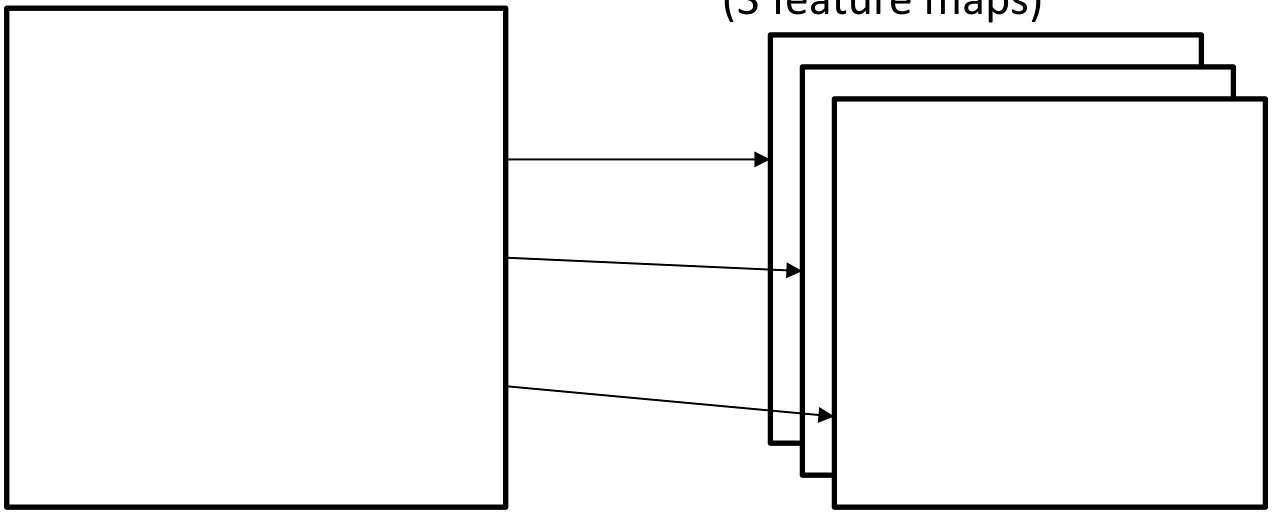
2 Feature maps 24x24

From Artificial Neural Network to Convolutional Neural Network (a deep learning model)

- CNN representation example

28 x 28 input image

First convolution layer: 3 x (24 x 24) neurons (3 feature maps)



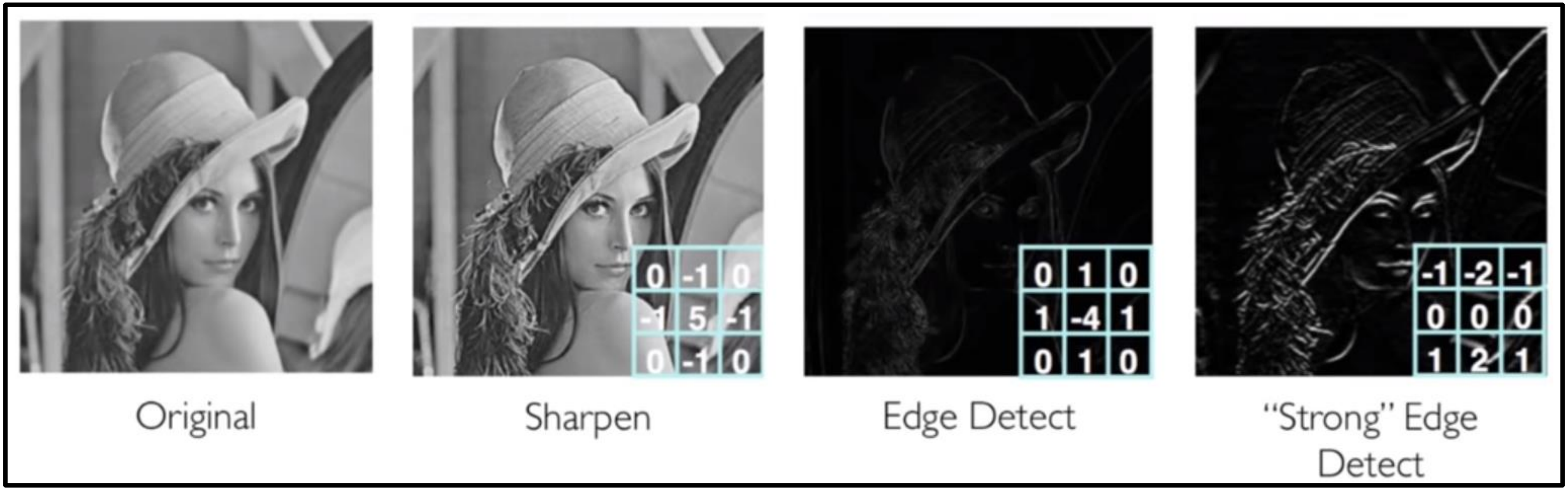
a) Width = [(image width – kernel width)+stride]

24 = 28 – 5 + 1

b) Same rule for height

From Artificial Neural Network to Convolutional Neural Network (a deep learning model)

- What feature maps contain



Sharpen:

0	-1	0
-1	5	-1
0	-1	0

Edge Detect:

0	1	0
1	-4	1
0	1	0

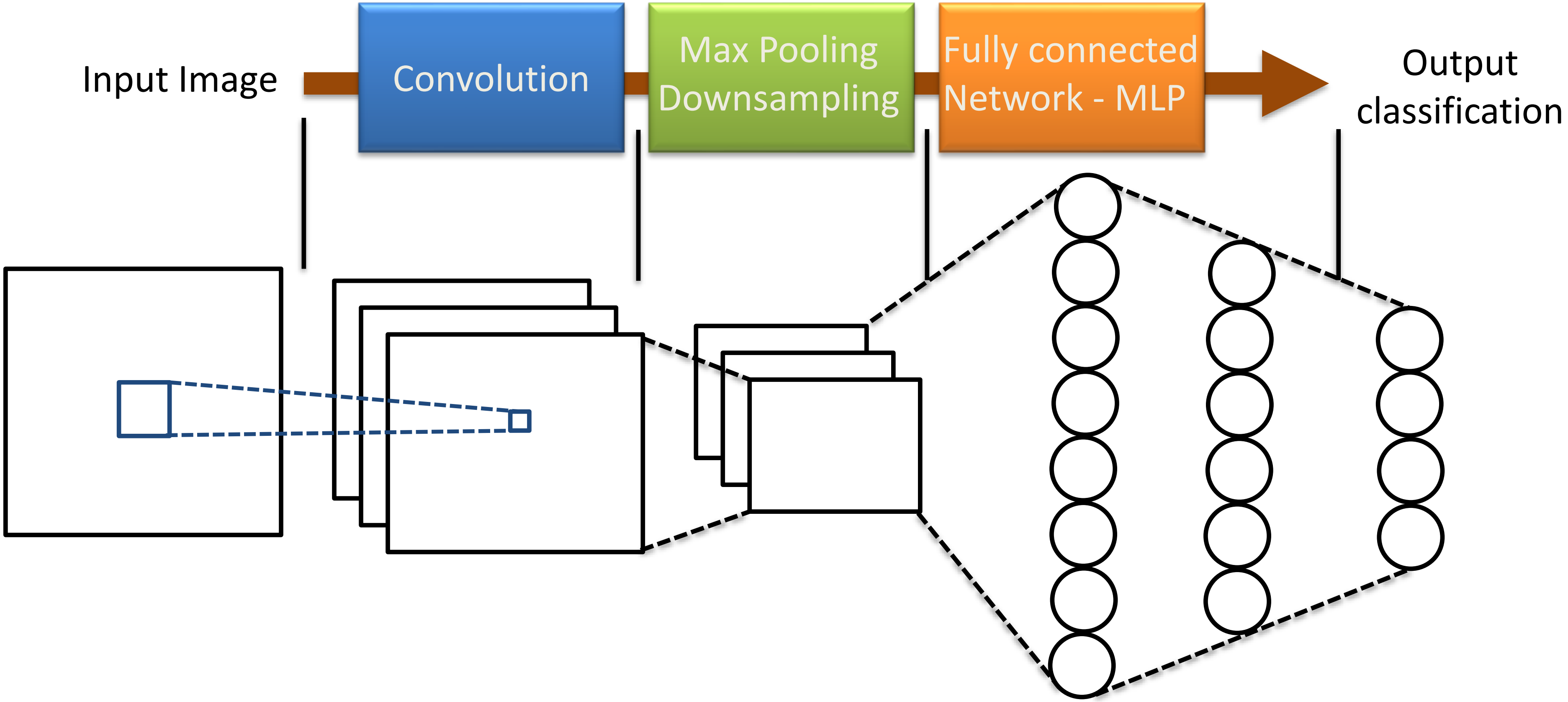
"Strong" Edge Detect:

-1	-2	-1
0	0	0
1	2	1

Input image and 3 associated feature maps
Extracted from https://www.youtube.com/watch?v=H-HVZJ7kGI0

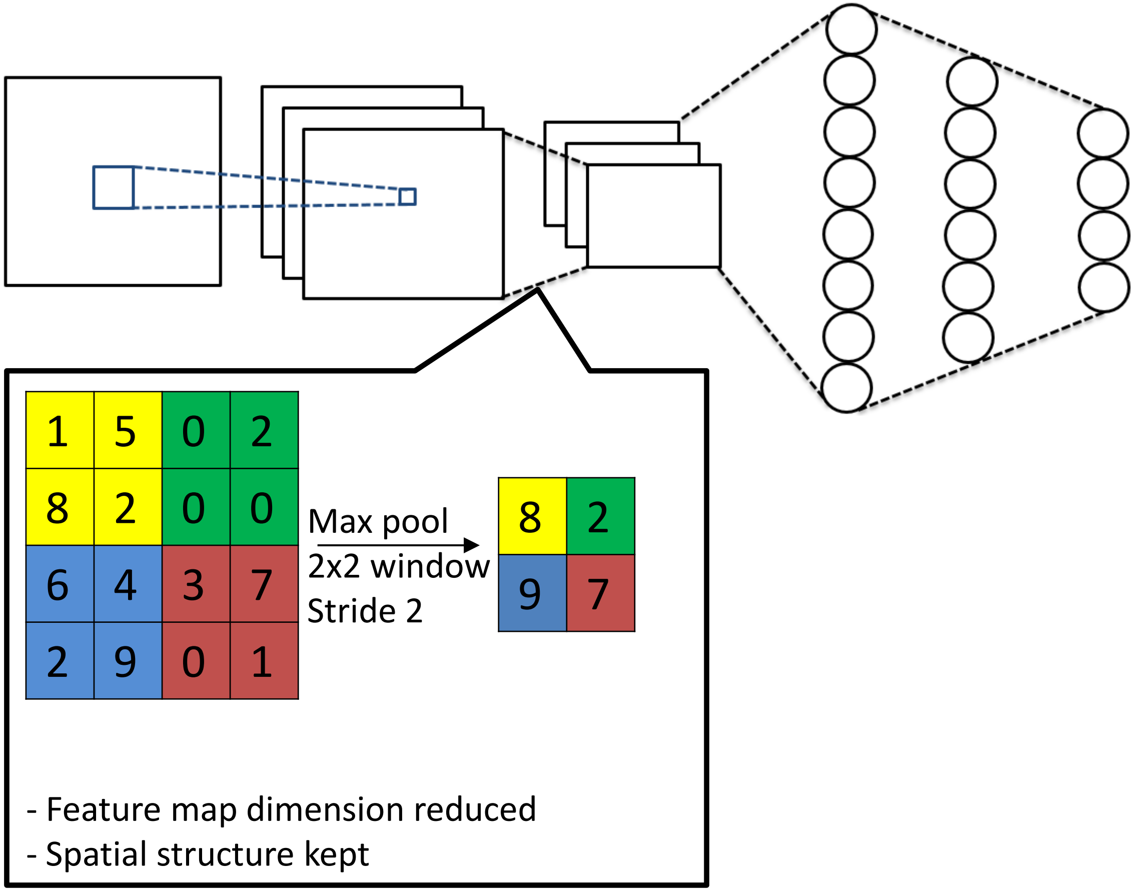
From Artificial Neural Network to Convolutional Neural Network (a deep learning model)

- How to classify with CNNs



From Artificial Neural Network to Convolutional Neural Network (a deep learning model)

- Maxpooling in CNNs



1	5	0	2
8	2	0	0
6	4	3	7
2	9	0	1

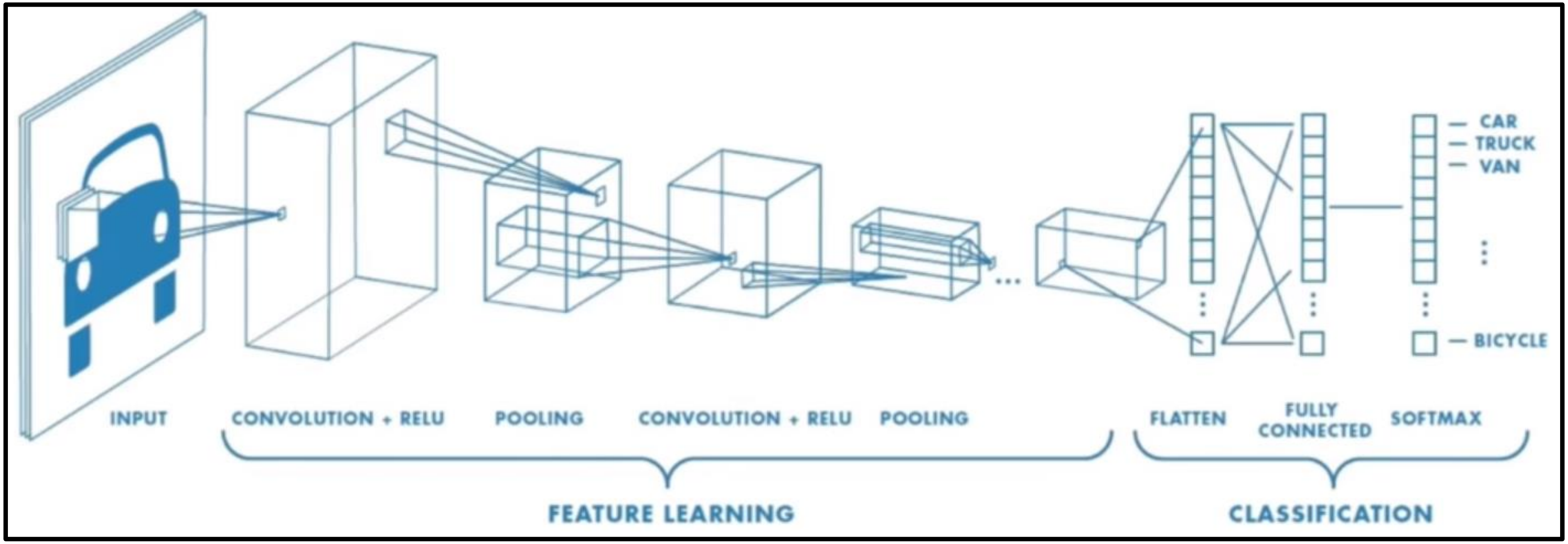
Max pool
2x2 window
Stride 2

8	2
9	7

- Feature map dimension reduced
- Spatial structure kept

From Artificial Neural Network to Convolutional Neural Network (a deep learning model)

- Training a CNN having multiple convolution layers



Architecture example extracted from
https://www.youtube.com/watch?v=H-HVZJ7kGI0

From Artificial Neural Network to Convolutional Neural Network (a deep learning model)

- Why do we need multiple convolution layers

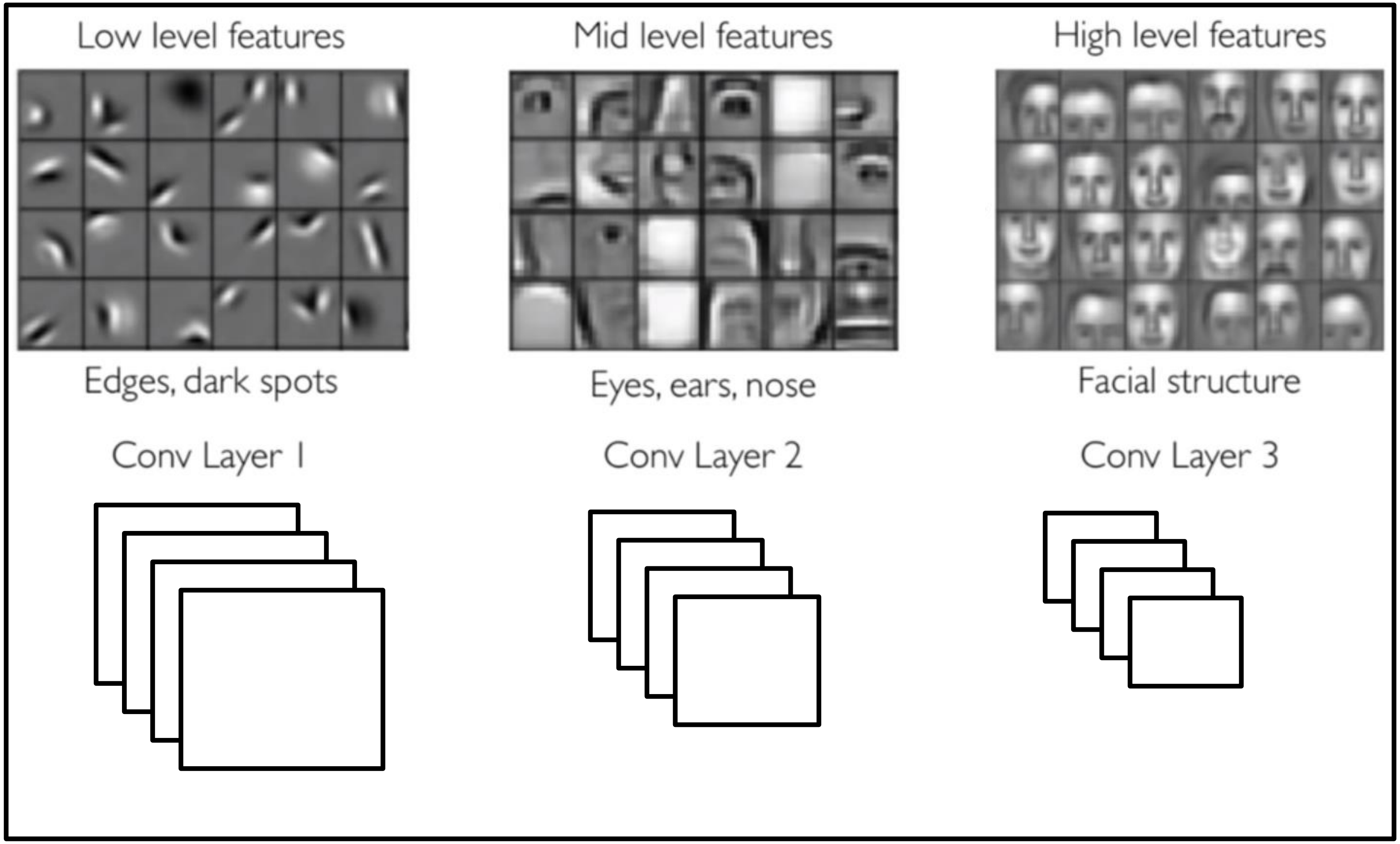
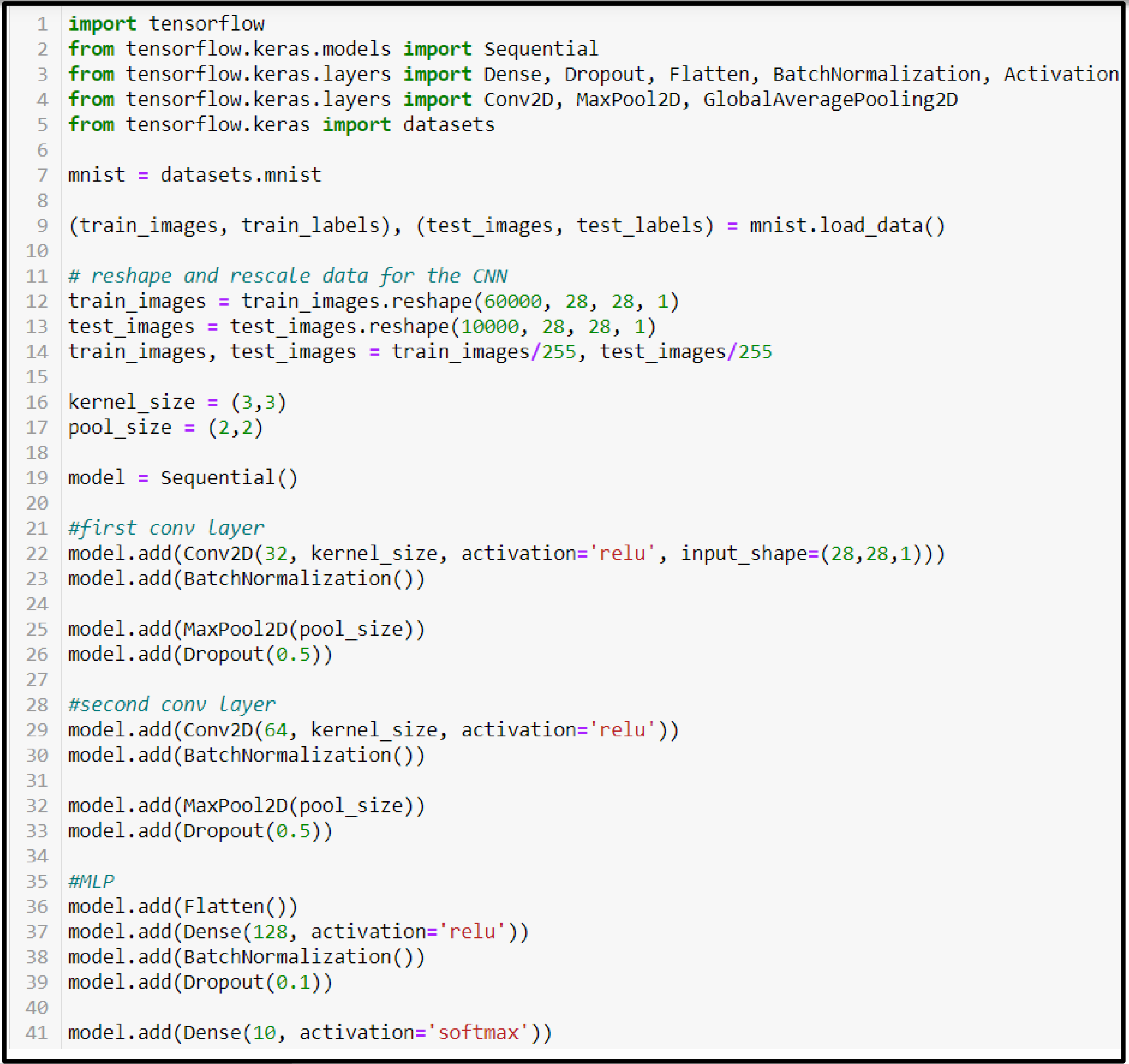


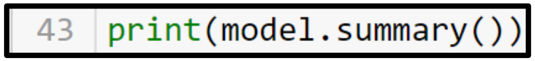
Figure extracted from https://www.youtube.com/watch?v=H-HVZJ7kGI0

Building a CNN using Keras – TensorFlow core

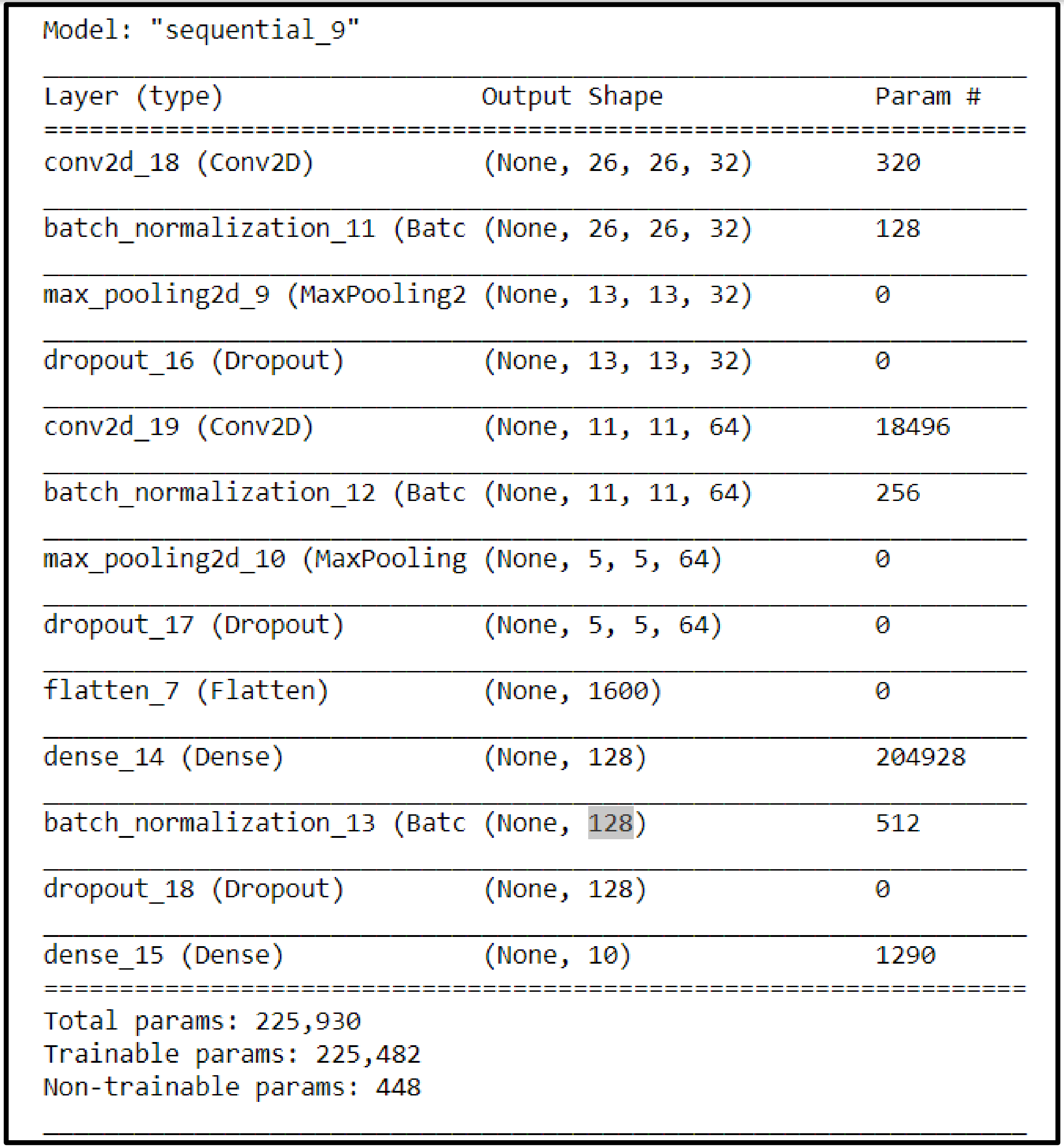


```python
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, BatchNormalization, Activation
from tensorflow.keras.layers import Conv2D, MaxPool2D, GlobalAveragePooling2D
from tensorflow.keras import datasets

mnist = datasets.mnist

(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

# reshape and rescale data for the CNN
train_images = train_images.reshape(60000, 28, 28, 1)
test_images = test_images.reshape(10000, 28, 28, 1)
train_images, test_images = train_images/255, test_images/255

kernel_size = (3,3)
pool_size = (2,2)

model = Sequential()

#first conv layer
model.add(Conv2D(32, kernel_size, activation='relu', input_shape=(28,28,1)))
model.add(BatchNormalization())

model.add(MaxPool2D(pool_size))
model.add(Dropout(0.5))

#second conv layer
model.add(Conv2D(64, kernel_size, activation='relu'))
model.add(BatchNormalization())

model.add(MaxPool2D(pool_size))
model.add(Dropout(0.5))

#MLP
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.1))

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

Analyzing CNN model parameters using Keras – TensorFlow core

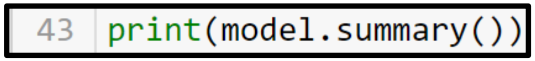


```
print(model.summary())
```



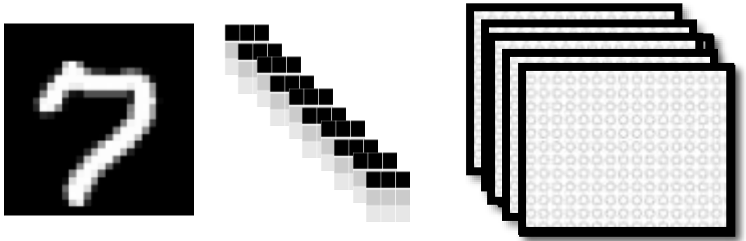
```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_18 (Conv2D)           (None, 26, 26, 32)        320       
_________________________________________________________________
batch_normalization_11 (Batc (None, 26, 26, 32)        128       
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 13, 13, 32)        0         
_________________________________________________________________
dropout_16 (Dropout)         (None, 13, 13, 32)        0         
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 11, 11, 64)        18496     
_________________________________________________________________
batch_normalization_12 (Batc (None, 11, 11, 64)        256       
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 5, 5, 64)          0         
_________________________________________________________________
dropout_17 (Dropout)         (None, 5, 5, 64)          0         
_________________________________________________________________
flatten_7 (Flatten)          (None, 1600)              0         
_________________________________________________________________
dense_14 (Dense)             (None, 128)               204928    
_________________________________________________________________
batch_normalization_13 (Batc (None, 128)               512       
_________________________________________________________________
dropout_18 (Dropout)         (None, 128)               0         
_________________________________________________________________
dense_15 (Dense)             (None, 10)                1290      
=================================================================
Total params: 225,930
Trainable params: 225,482
Non-trainable params: 448
_________________________________________________________________
```

Analyzing CNN model parameters using Keras – TensorFlow core



```python
print(model.summary())
```

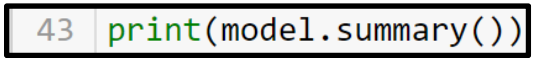
```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_18 (Conv2D)           (None, 26, 26, 32)        320
```



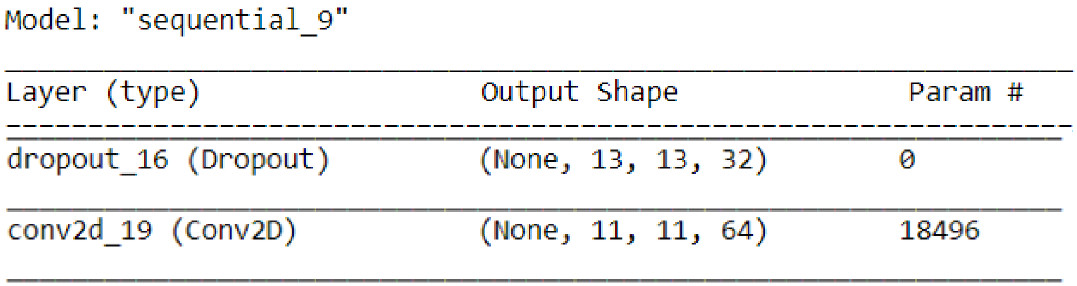
Total of weights:
[(3x3)+1]x32 = 320

```
_________________________________________________________________
```

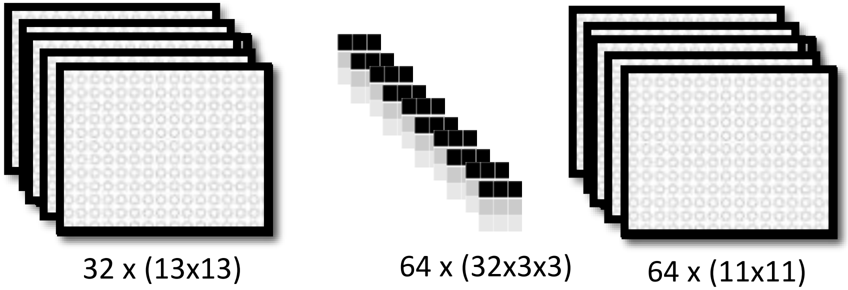
Analyzing CNN model parameters using Keras – TensorFlow core



```
print(model.summary())
```



```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dropout_16 (Dropout)         (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 11, 11, 64)        18496
_________________________________________________________________
```



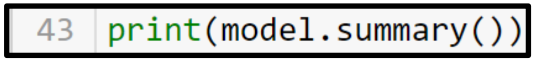
32 x (13x13)

64 x (32x3x3)

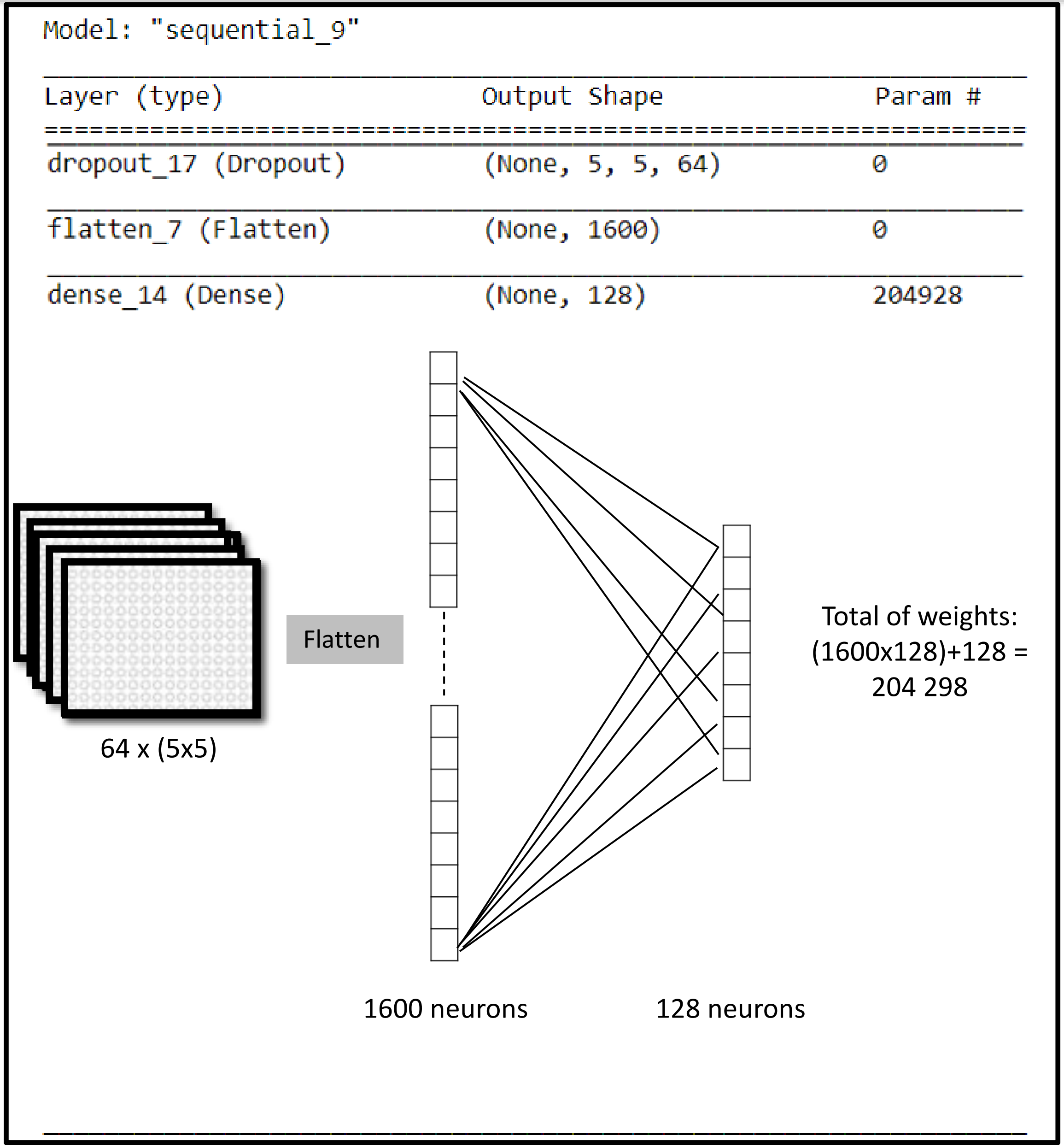
64 x (11x11)

Total of weights:
[(32x3x3)+1x64] =
18 496

Analyzing CNN model parameters using Keras – TensorFlow core



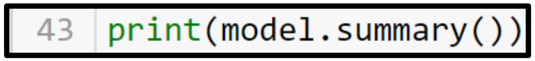
```
print(model.summary())
```



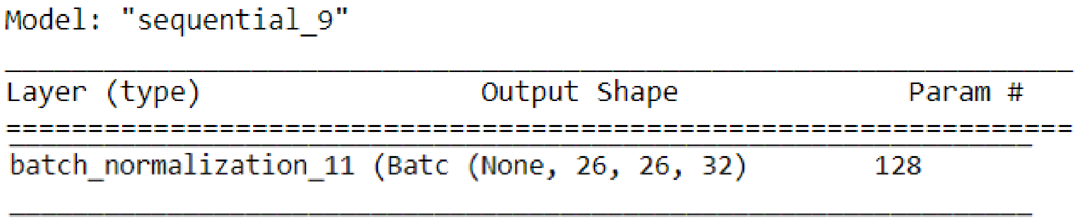
```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dropout_17 (Dropout)         (None, 5, 5, 64)          0
_________________________________________________________________
flatten_7 (Flatten)          (None, 1600)              0
_________________________________________________________________
dense_14 (Dense)             (None, 128)               204928
```

Total of weights:
(1600x128)+128 =
204 298

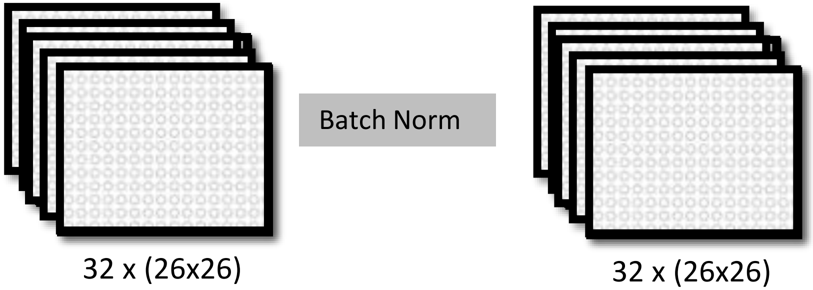
Analyzing CNN model parameters using Keras – TensorFlow core



```
print(model.summary())
```



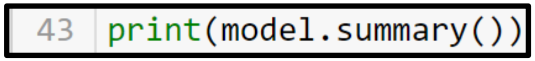
```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
batch_normalization_11 (Batc (None, 26, 26, 32)        128
_________________________________________________________________
```



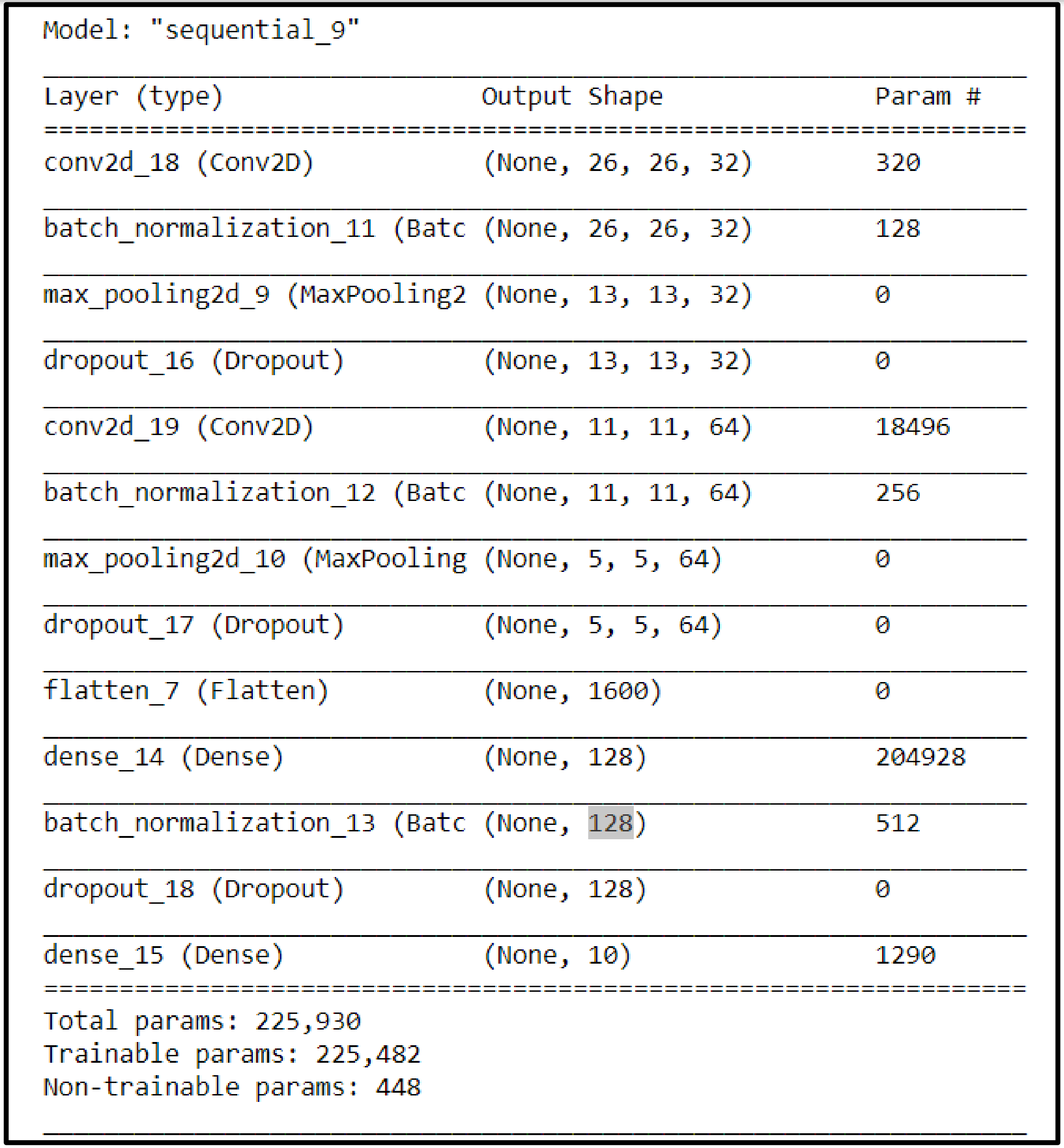
Total of params
32x4

1- mean
2- variance
3- scale (trainable)
4- shift (trainable)

Analyzing CNN model parameters using Keras – TensorFlow core



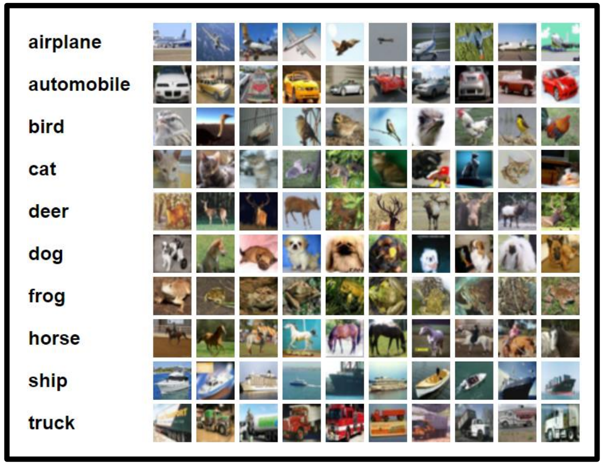
```python
print(model.summary())
```



```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_18 (Conv2D)           (None, 26, 26, 32)        320       
_________________________________________________________________
batch_normalization_11 (Batc (None, 26, 26, 32)        128       
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 13, 13, 32)        0         
_________________________________________________________________
dropout_16 (Dropout)         (None, 13, 13, 32)        0         
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 11, 11, 64)        18496     
_________________________________________________________________
batch_normalization_12 (Batc (None, 11, 11, 64)        256       
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 5, 5, 64)          0         
_________________________________________________________________
dropout_17 (Dropout)         (None, 5, 5, 64)          0         
_________________________________________________________________
flatten_7 (Flatten)          (None, 1600)              0         
_________________________________________________________________
dense_14 (Dense)             (None, 128)               204928    
_________________________________________________________________
batch_normalization_13 (Batc (None, 128)               512       
_________________________________________________________________
dropout_18 (Dropout)         (None, 128)               0         
_________________________________________________________________
dense_15 (Dense)             (None, 10)                1290      
=================================================================
Total params: 225,930
Trainable params: 225,482
Non-trainable params: 448
_________________________________________________________________
```

Lab session Keras – TensorFlow core

- Build your first CNN to solve the hand written digits problem
    - Use MNIST dataset
    - Give the number of trainable parameters of your model and explain how they have been calculated
- Train the same model on CIFAR10 dataset
    - Give the performance
- Adapt it to train on CIFAR100 dataset



- Give the best performance of your CNN by varying the different parameters seen in the course
    - For each best run on datasets show training/validation accuracy curves