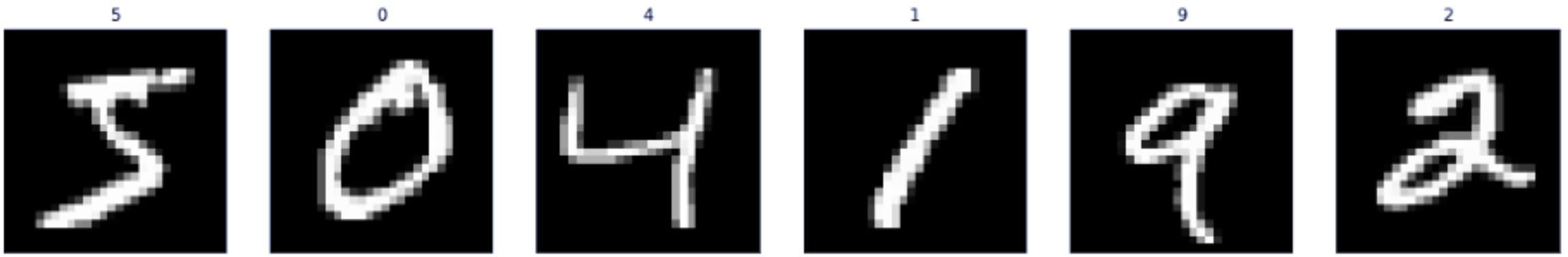
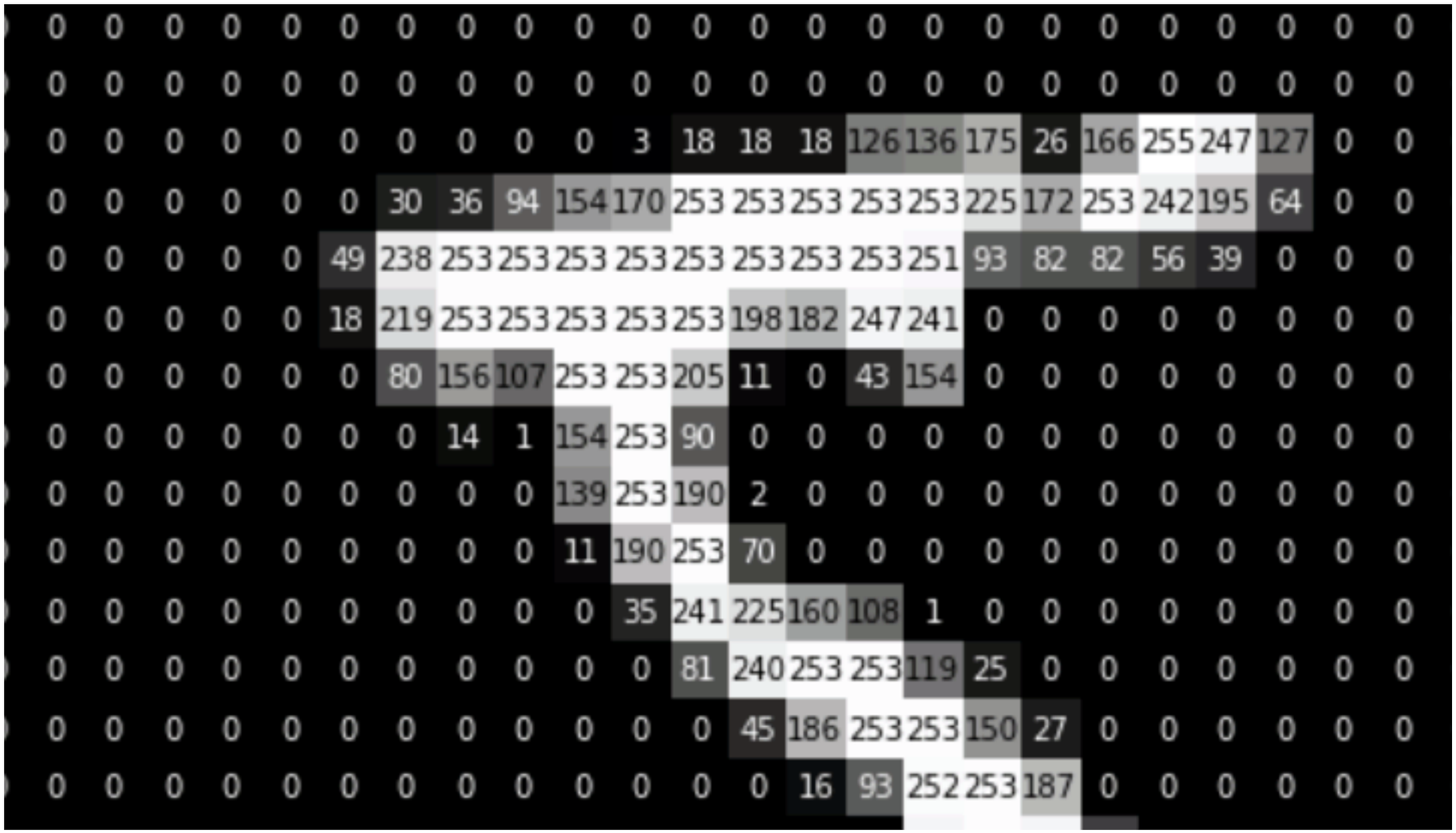
Convolution Neural Networks

MNIST

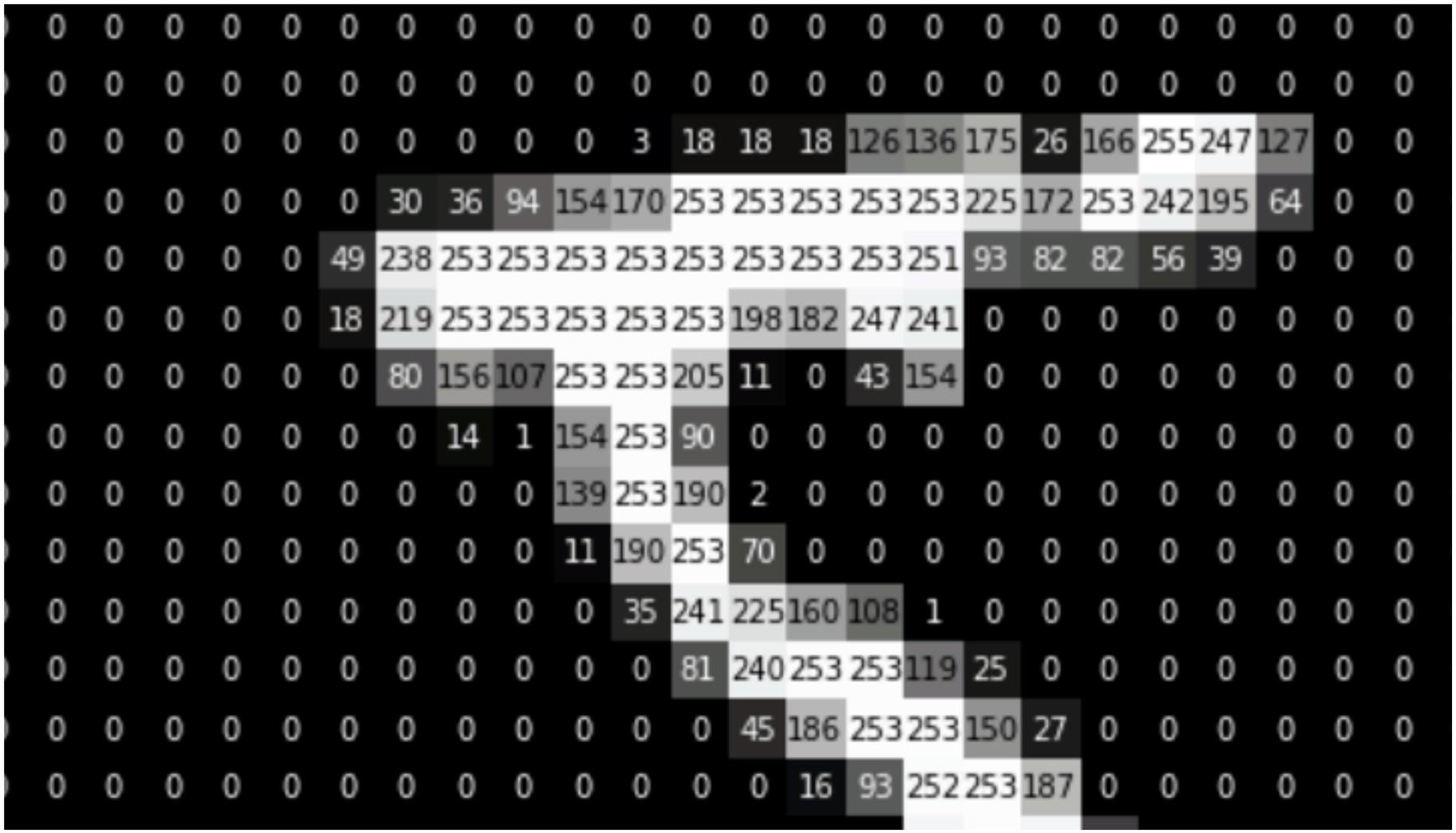
- Most Famous dataset in the ML world
- 70000 images of hand written digits
- Each image is 28x28 pixels black and white



MNIST : Details



MNIST : Details



MNIST : Labels

- The labels are numbers
- We need to convert them to one-hot encoding
- Can be done with to_categorical function in keras

```
# one-hot encode the labels
y_train = np_utils.to_categorical(y_train, 10)
y_test = np_utils.to_categorical(y_test, 10)
```

MNIST : Architecture

- We will use a 512x512x10 architecture
- Use Dropout to regularize

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               401920
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               262656
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                5130
=================================================================
Total params: 669,706.0
Trainable params: 669,706.0
Non-trainable params: 0.0
_________________________________________________________________
```

MNIST : Compilation

- Loss as categorical_crossentropy
- Optimizer as adam

- Accuracy after 10 epochs

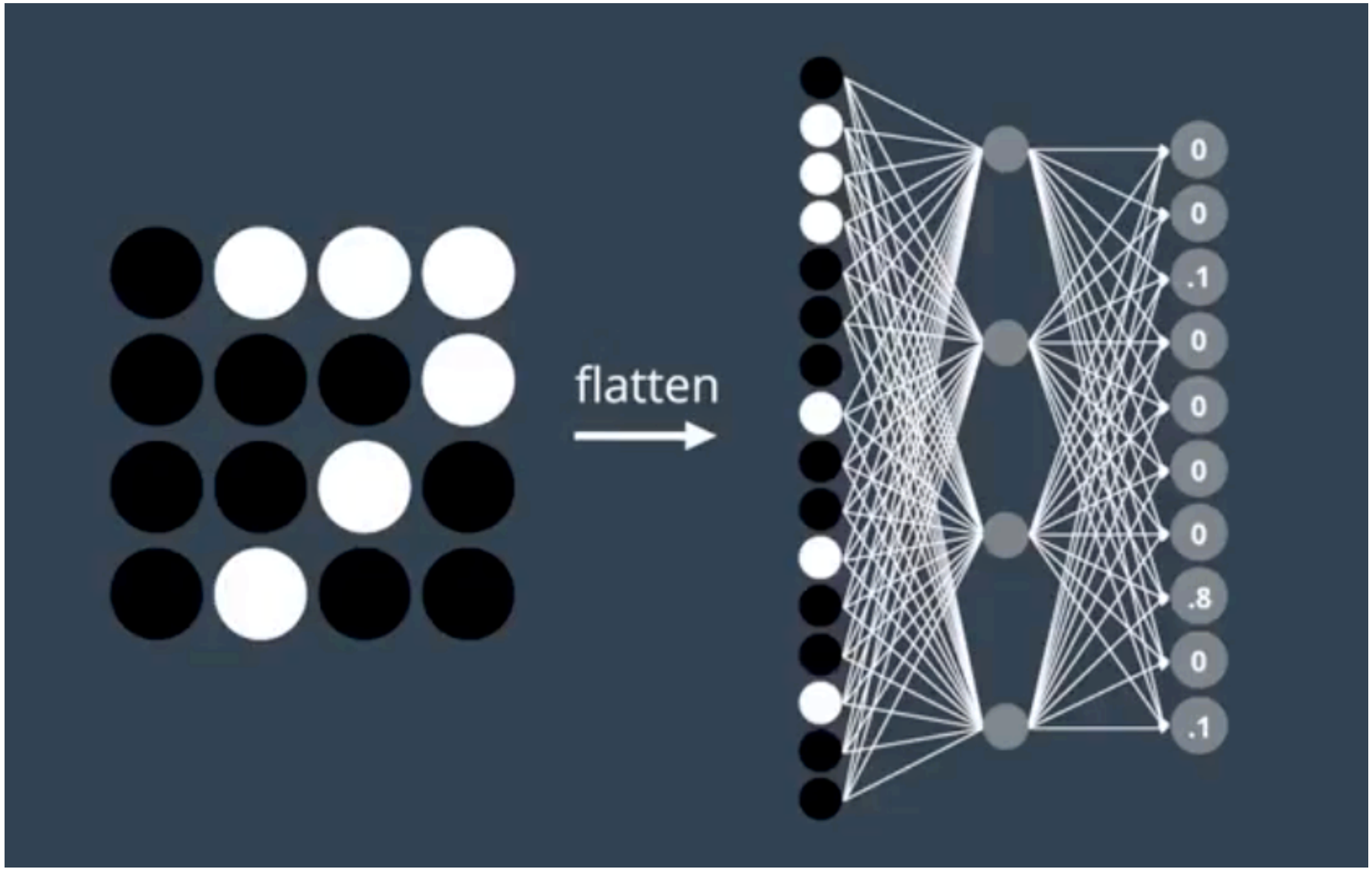
```
4
5 # print test accuracy
6 print('Test accuracy: %.4f%%' % accuracy)
```

```
Test accuracy: 98.2100%
```

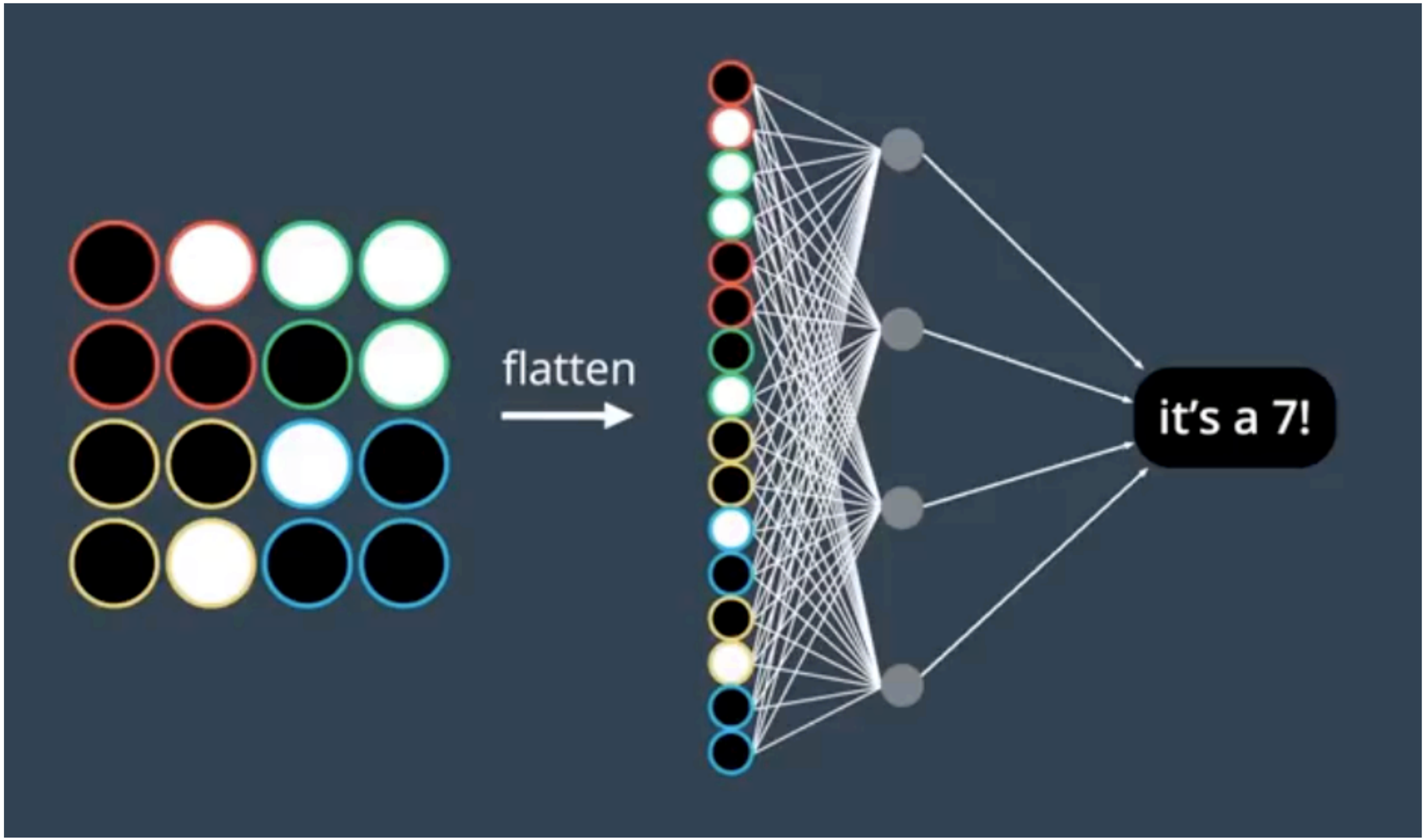
MNIST vs Others

- Dataset is simple and highly preprocess
- The digits are already in centre of the image
- Problem with previous Architecture
  - We convert the image to a vector
  - The spatial information is lost

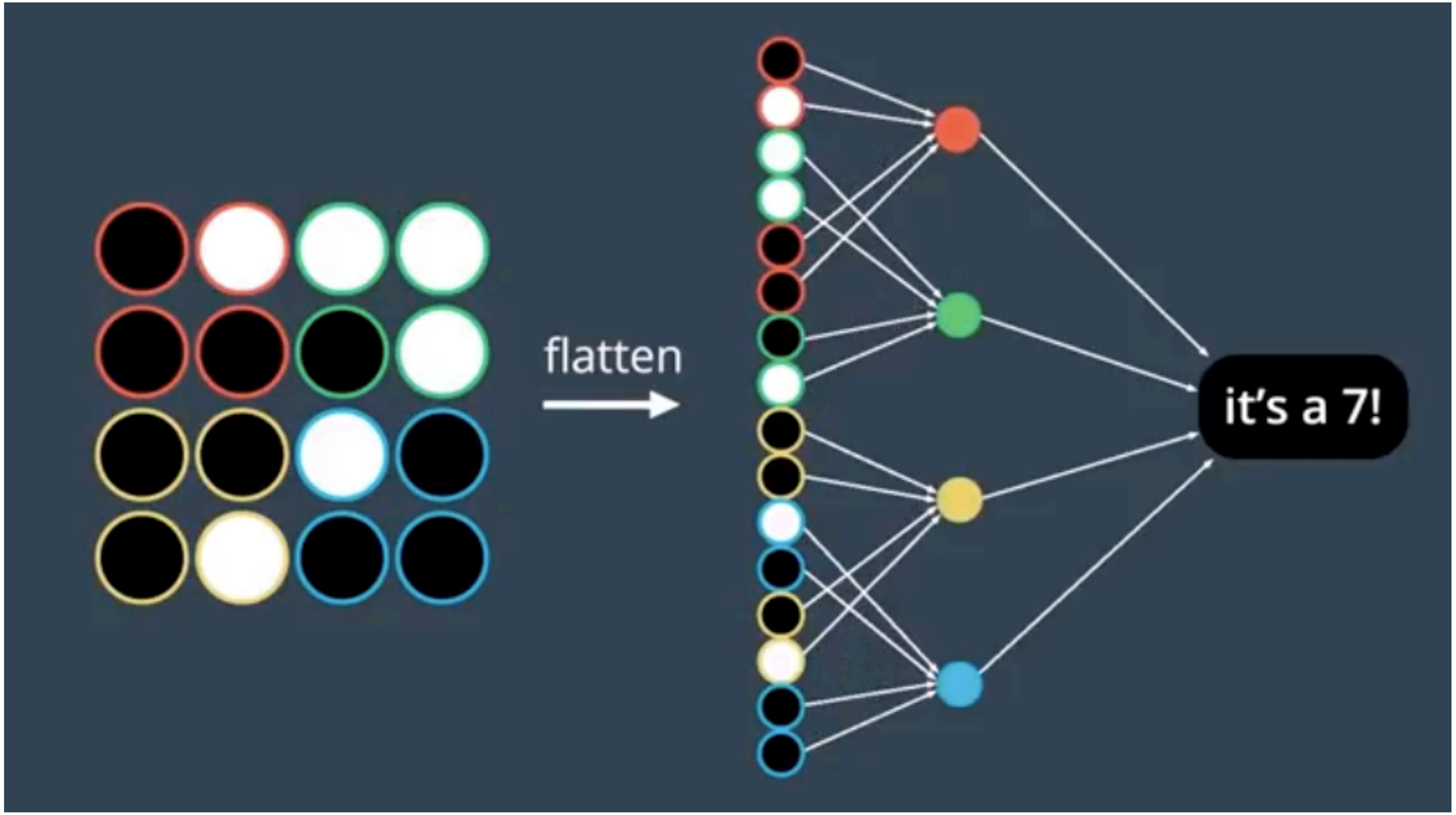
Fully connected Layer



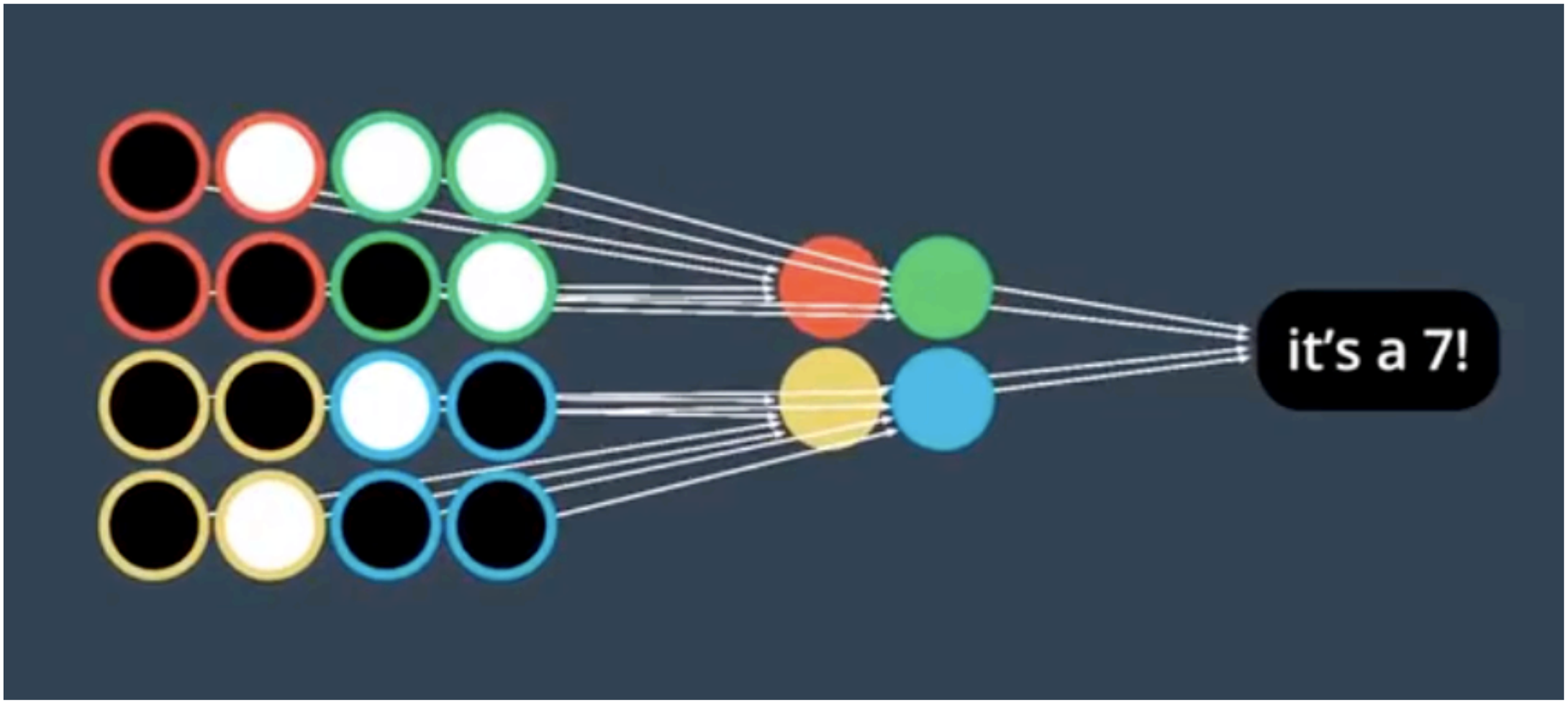
Still Fully connected



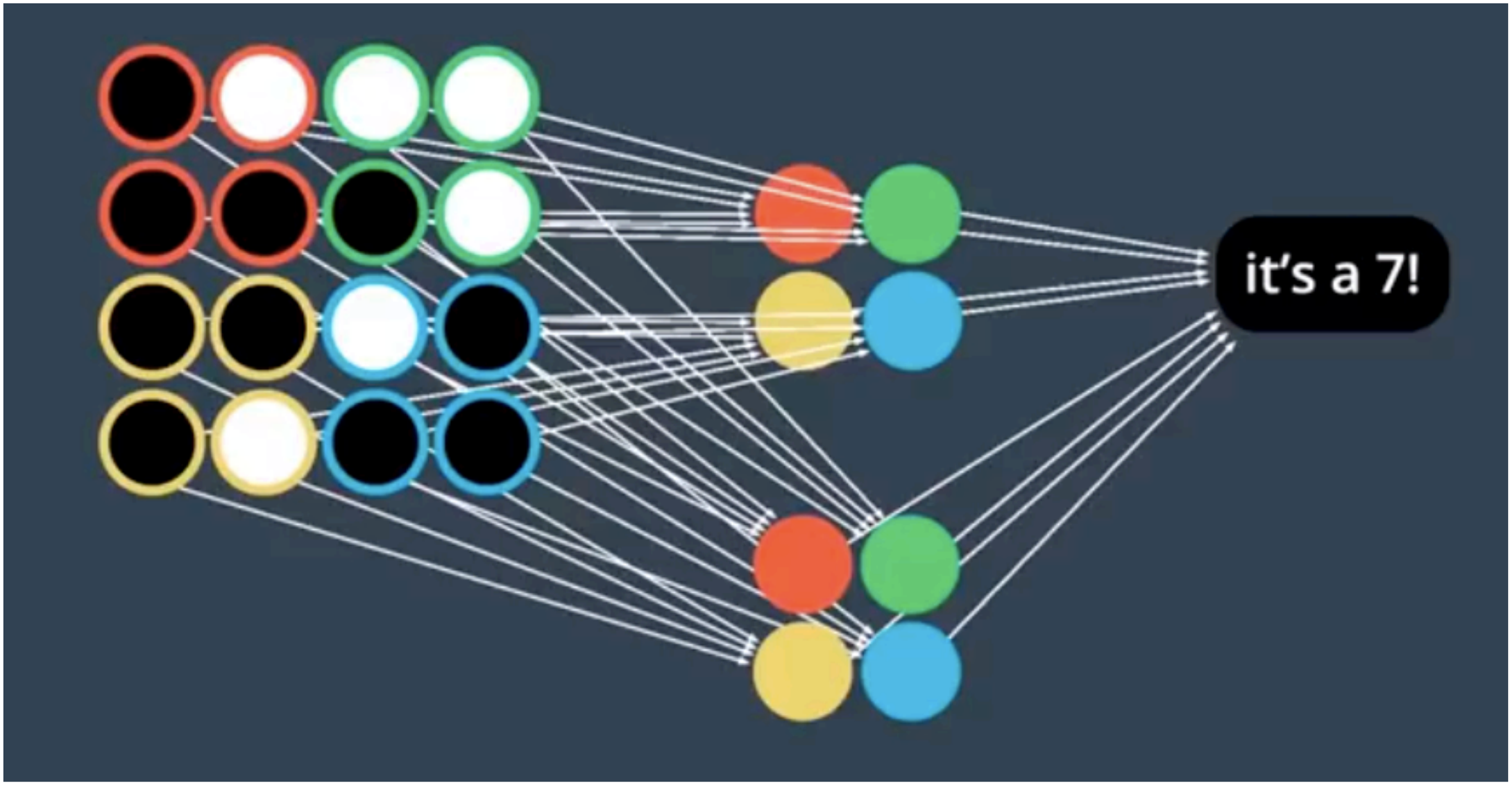
Local Connectivity



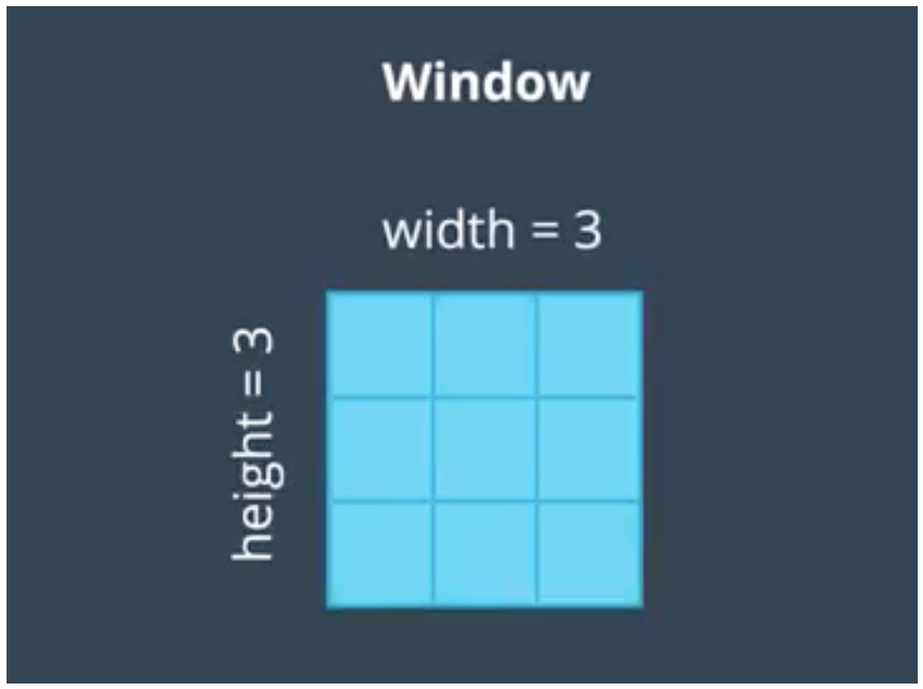
Local Connectivity Part 2



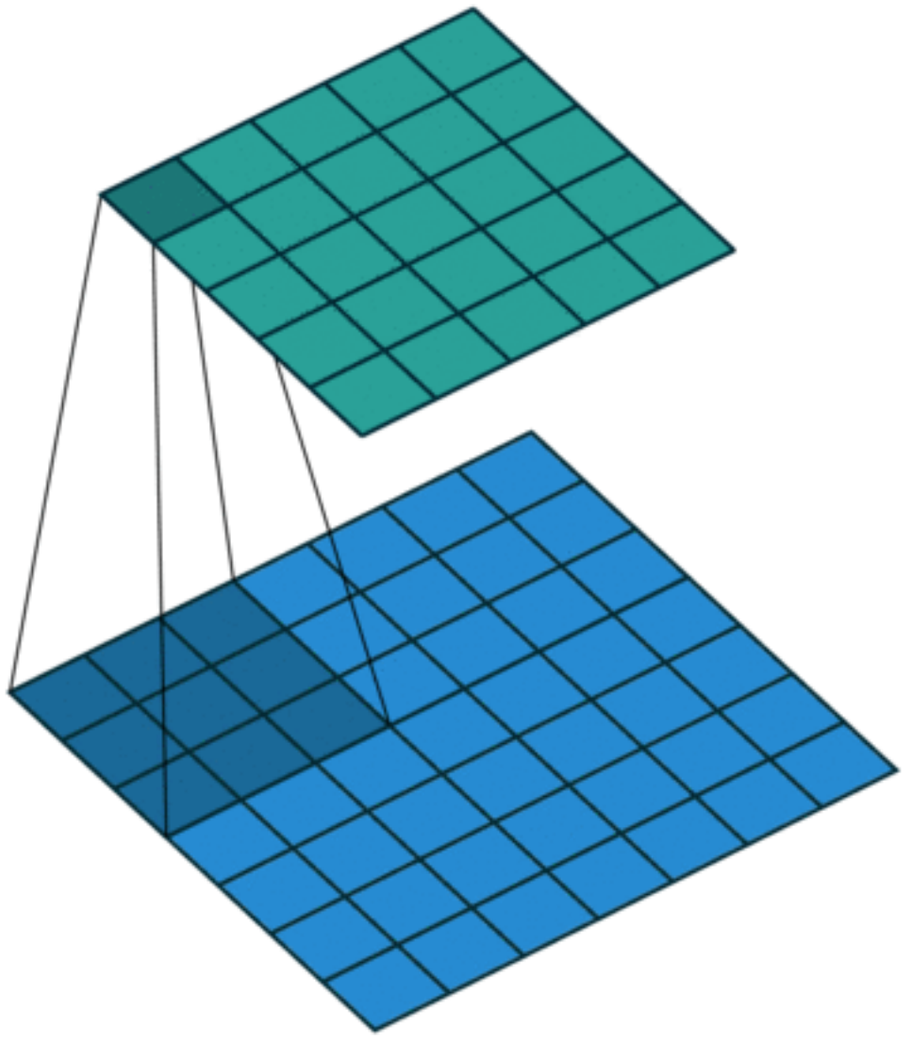
Local Connectivity Part 3



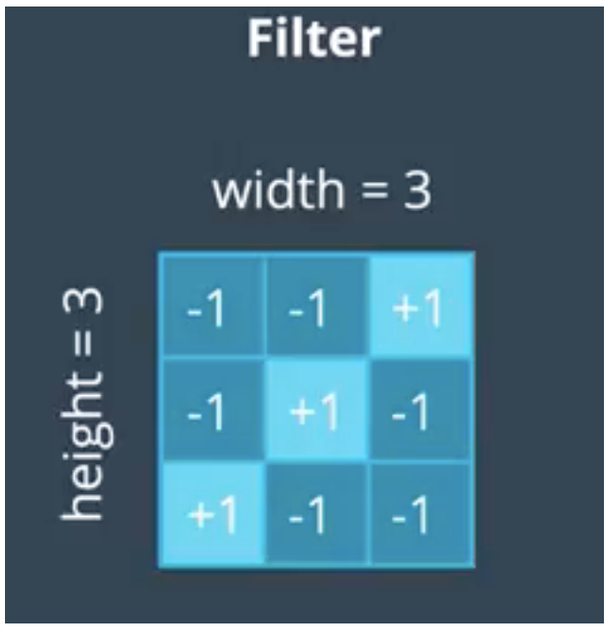
Convolution Filter



Convolution Filter

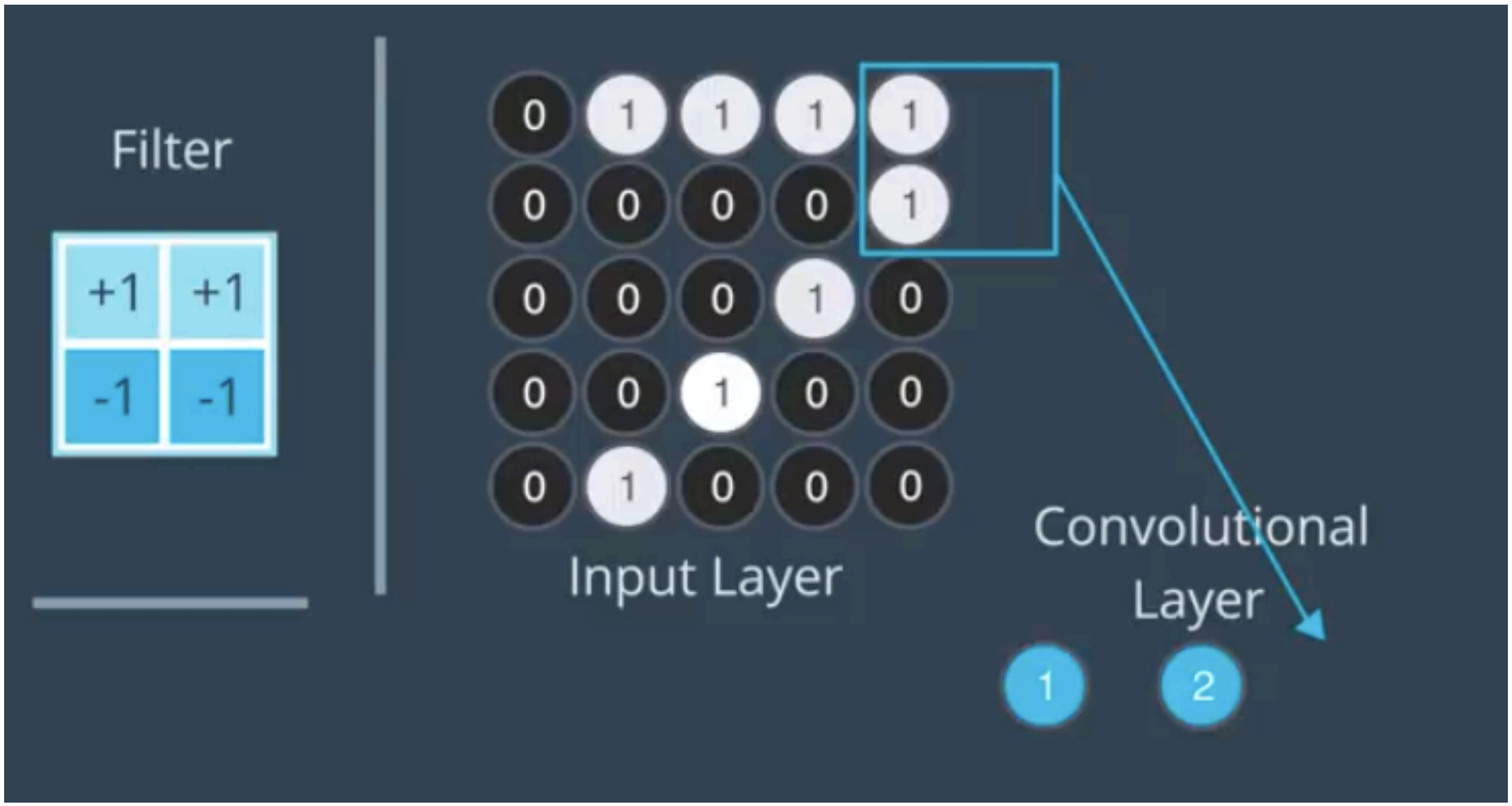


Convolution Filter



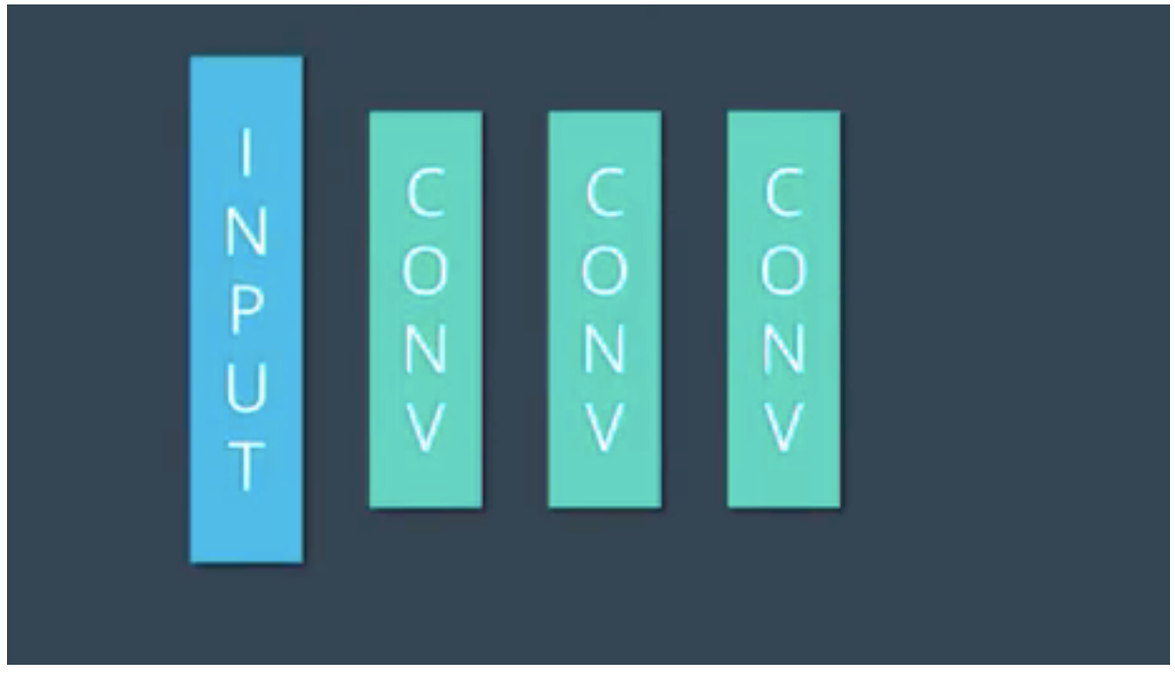
What does it identify ?

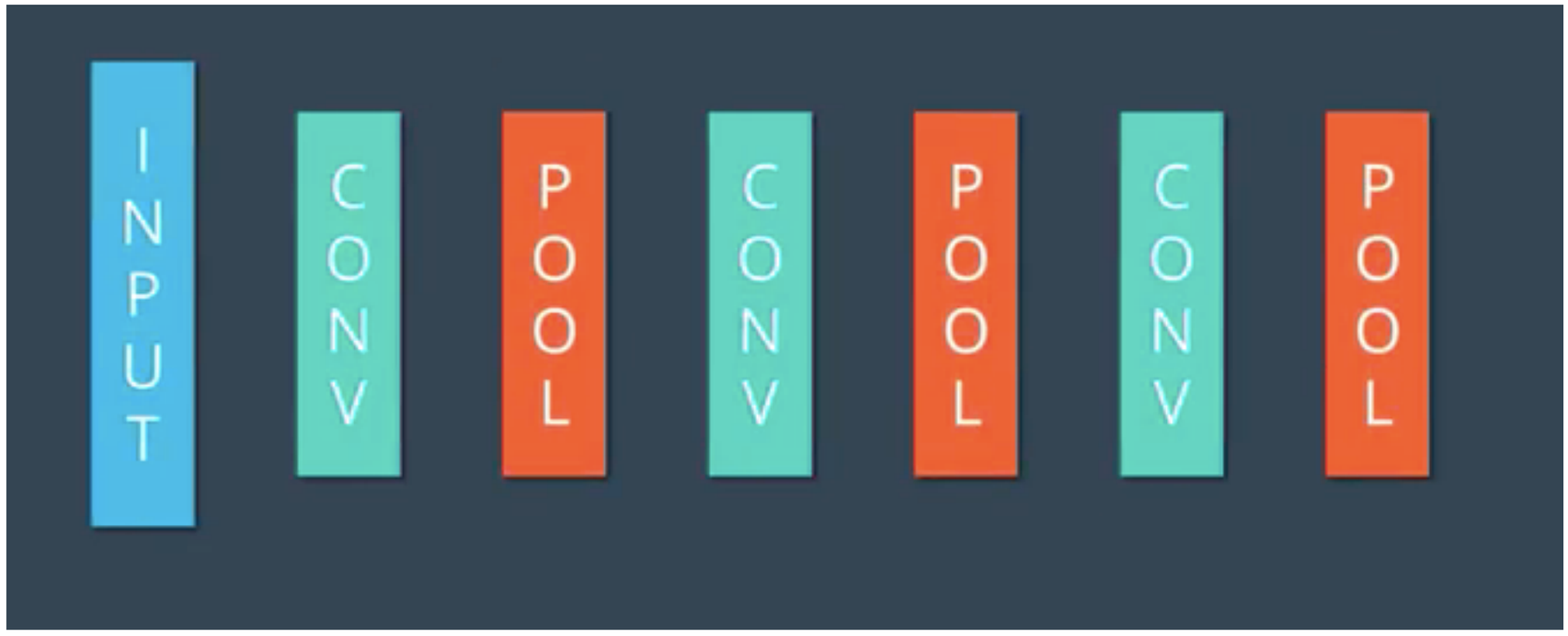
Strides and Padding



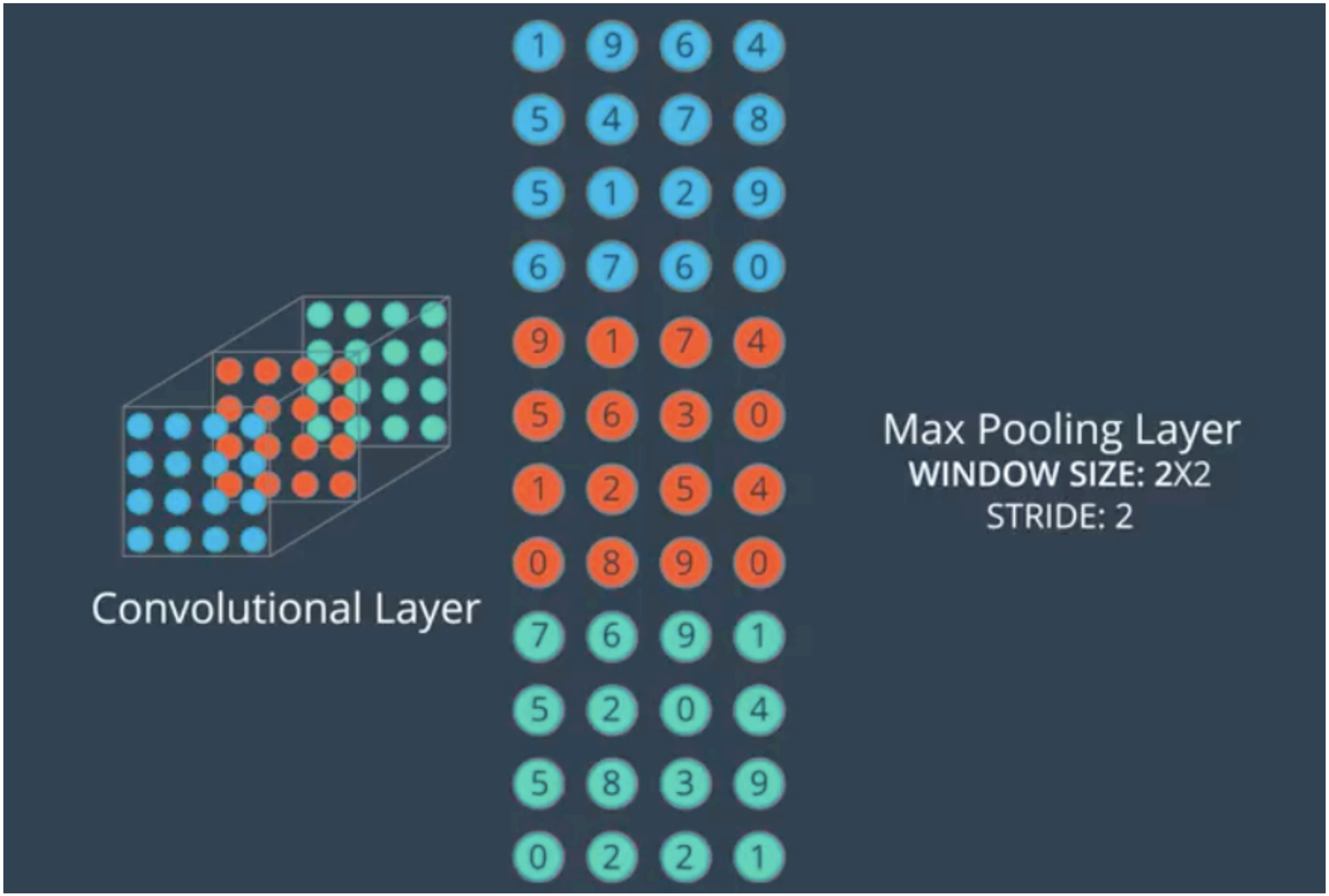
stride = 1 move by one pixel

Pooling Layers

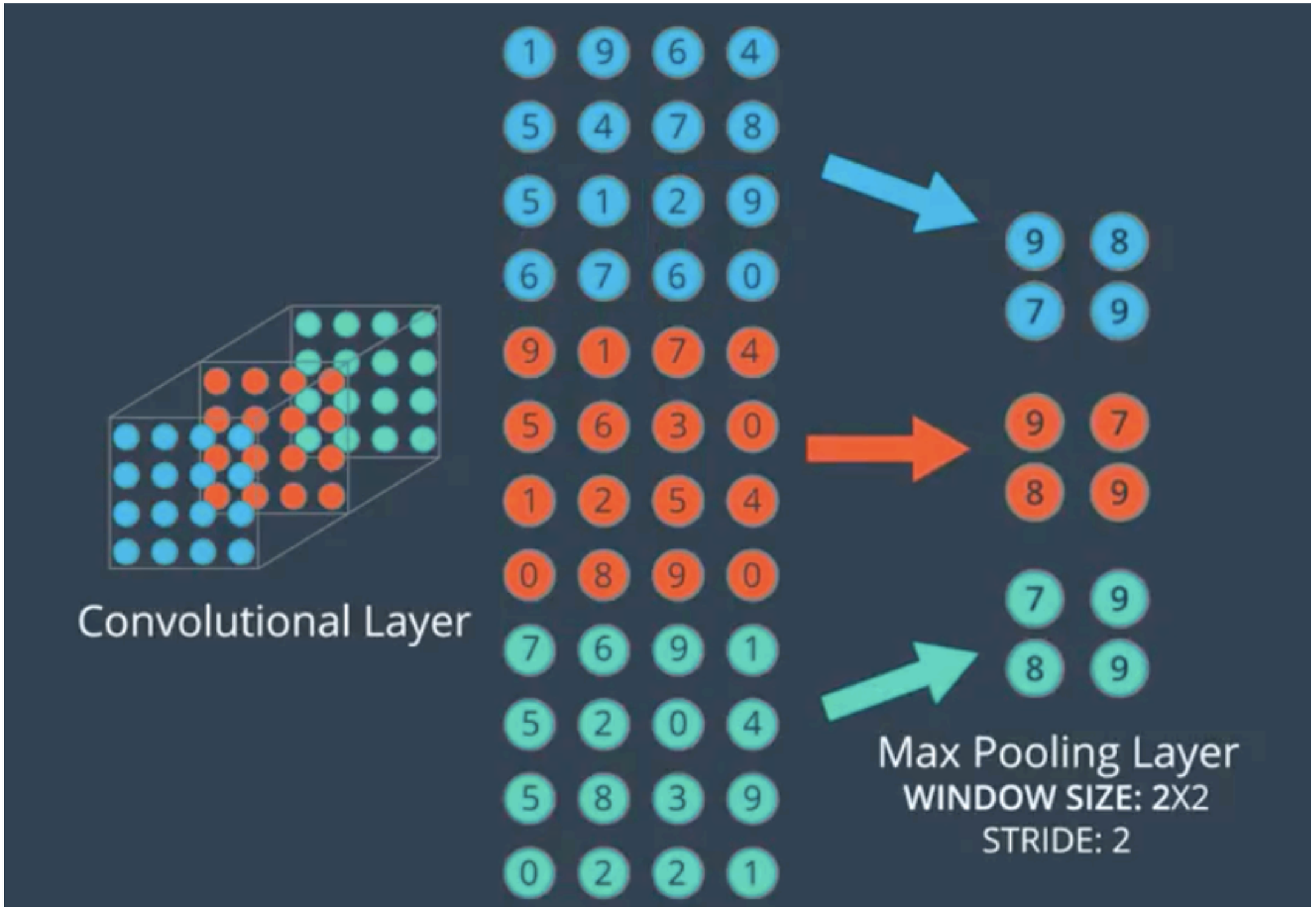




Pooling Layers : Max Pooling



Pooling Layers : Max Pooling



Pooling Layers : Global Average Pooling

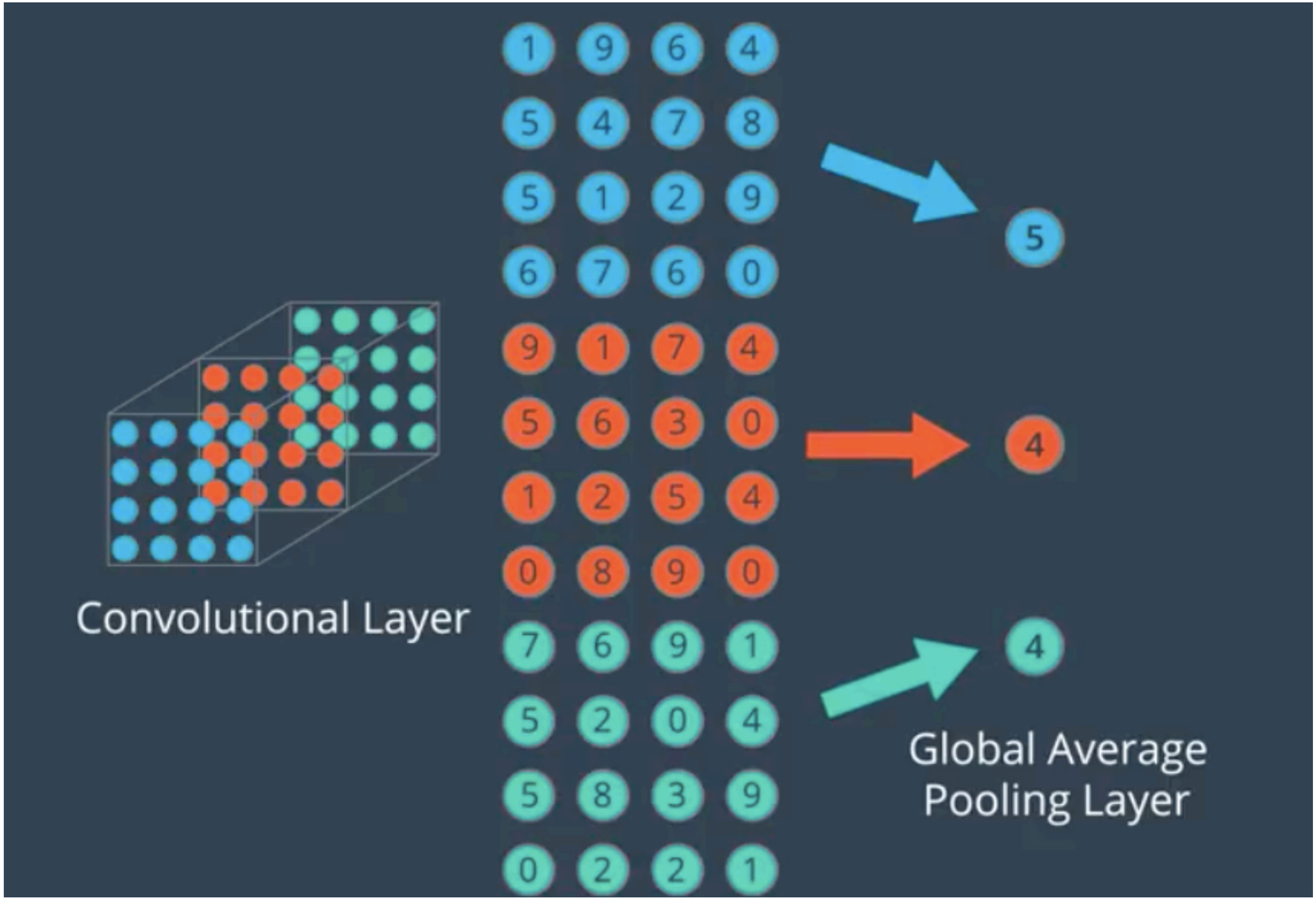


Image Classification : CNN

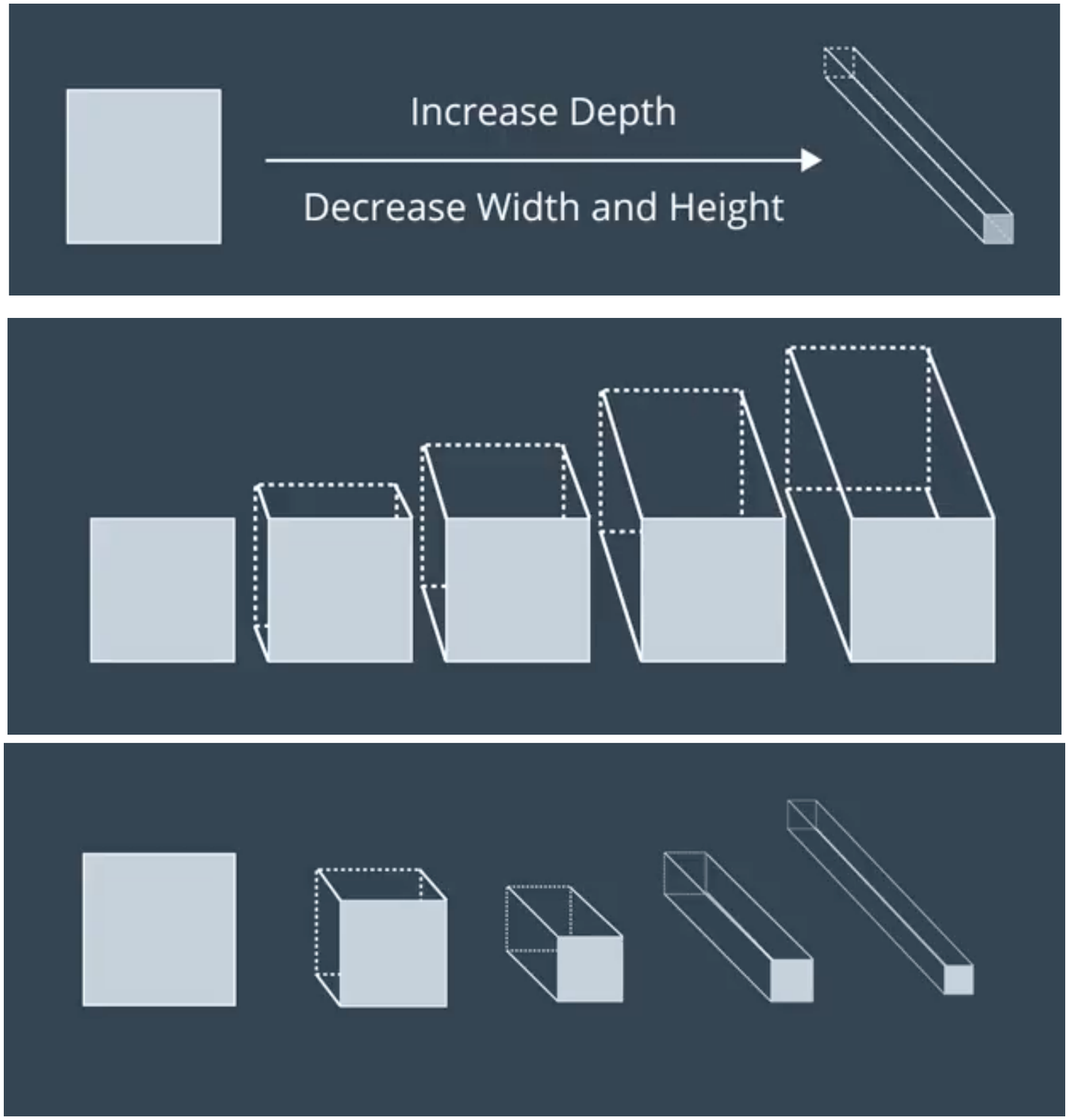


Image Classification : CNN

Image Classification : CNN

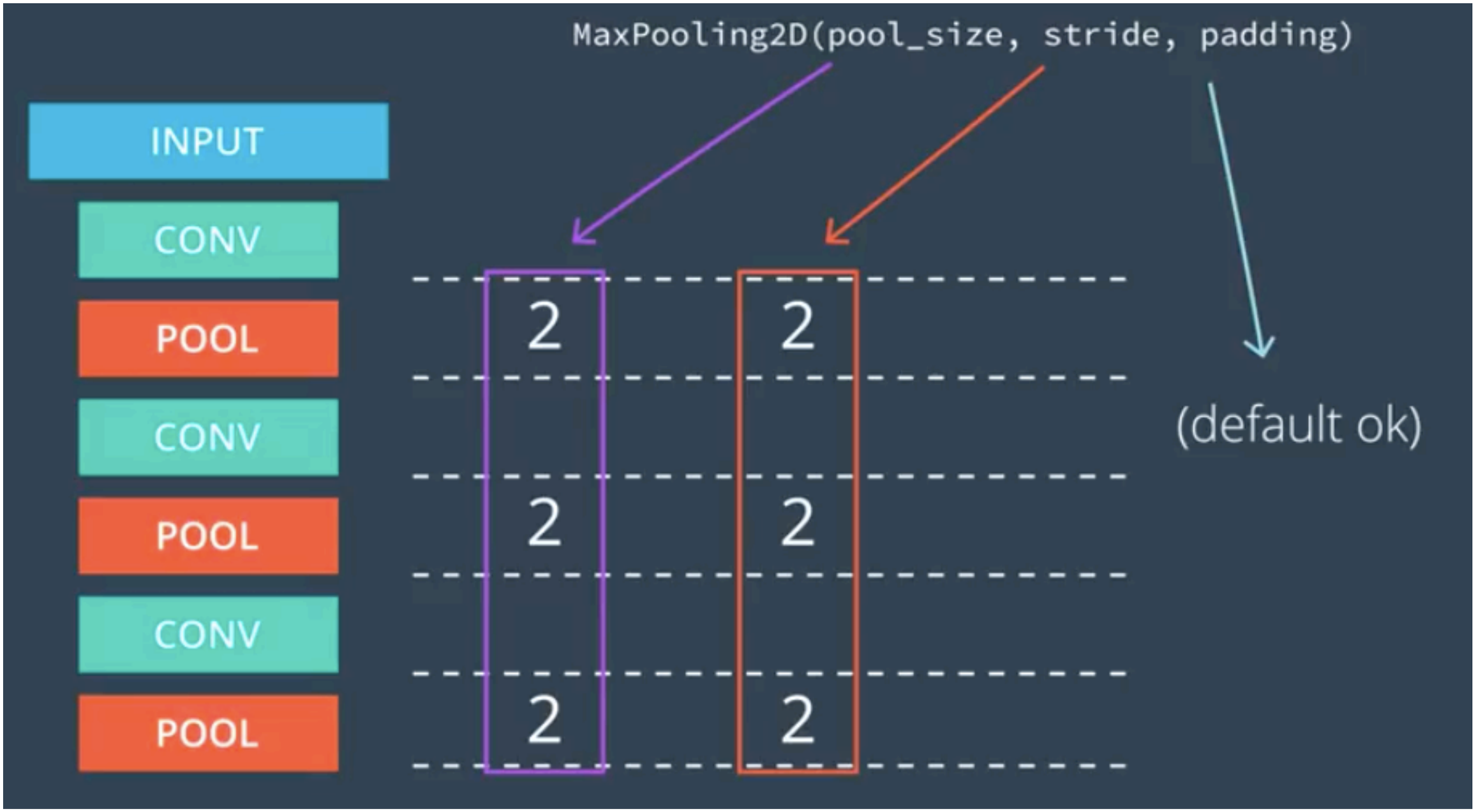


Image Classification : CNN

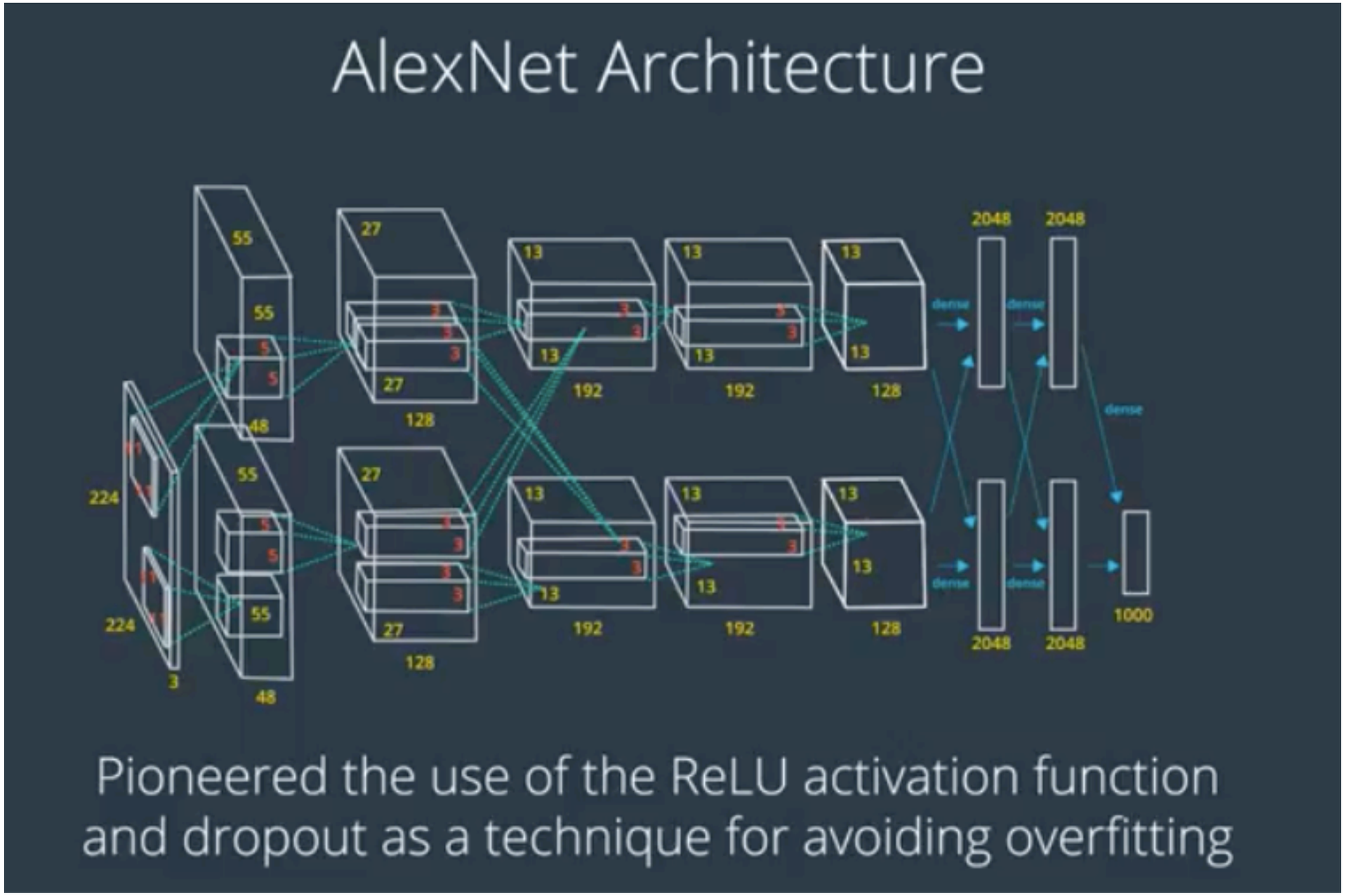
Image Classification : CNN

```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

model = Sequential()
model.add(Conv2D(filters=16, kernel_size=2, padding='same', activation='relu',
                         input_shape=(32, 32, 3)))
model.add(MaxPooling2D(pool_size=2))
model.add(Conv2D(filters=32, kernel_size=2, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Conv2D(filters=64, kernel_size=2, padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(500, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(10, activation='softmax'))

model.summary()
```

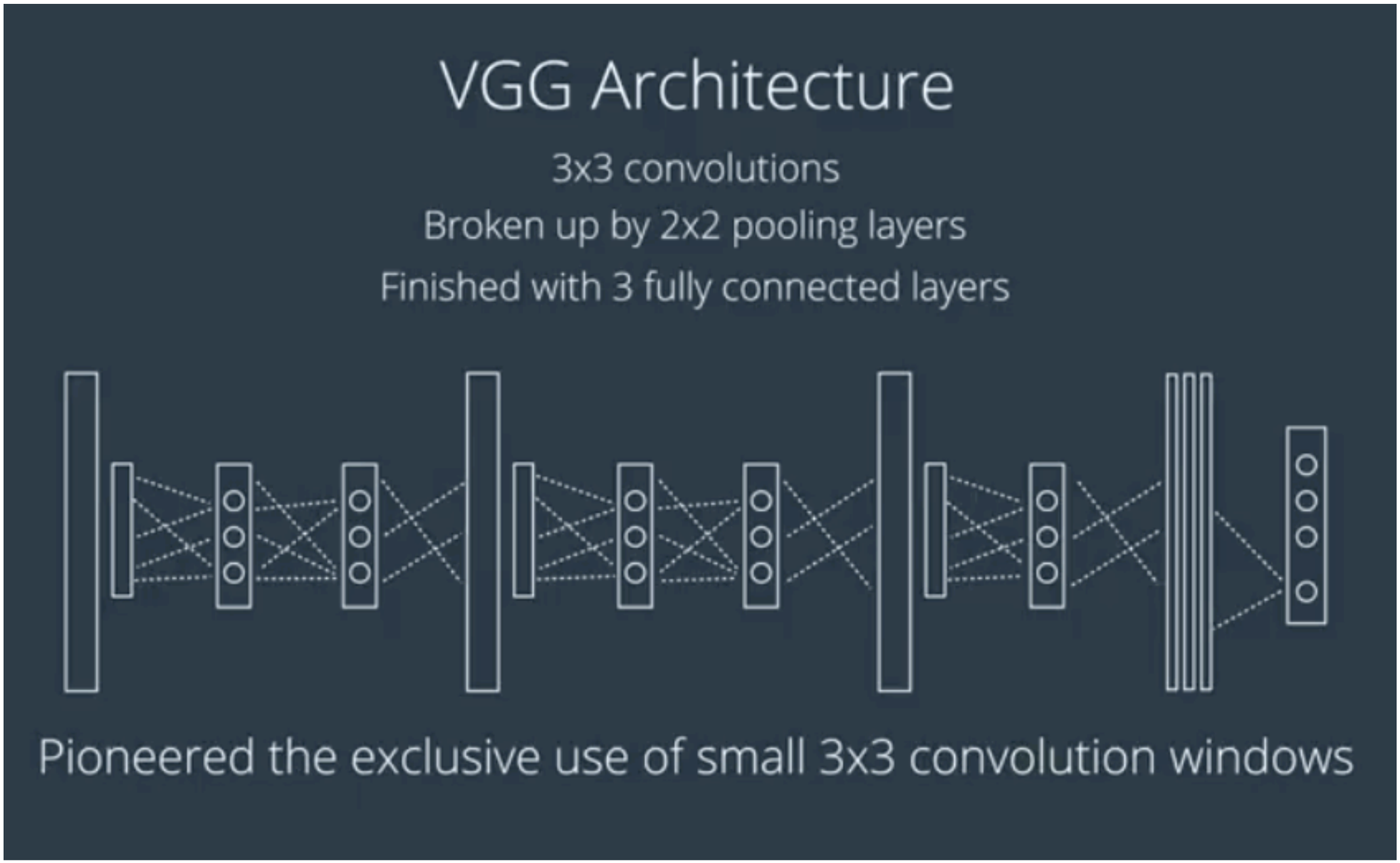
```
score = model.evaluate(x_test, y_test, verbose=0)
print('\n', 'Test accuracy:', score[1])
```

```
Test accuracy: 0.68
```

Important Architectures : AlexNet



Important Architectures : VGG



Important Architectures : Resnet