Deep Learning
Convolutional Neuronal Networks

Manuel Piñar Molina

Applications of CNNs

The CNNs have achieved improvement in many areas of AI

- **Voice interfaces** (Alexa, google nest…)
- **NLP (natural language processing)**
- **Computer Vision**
- **Reinforcement learning**
- **Robotics**

Applications of CNNs

Voice interfaces

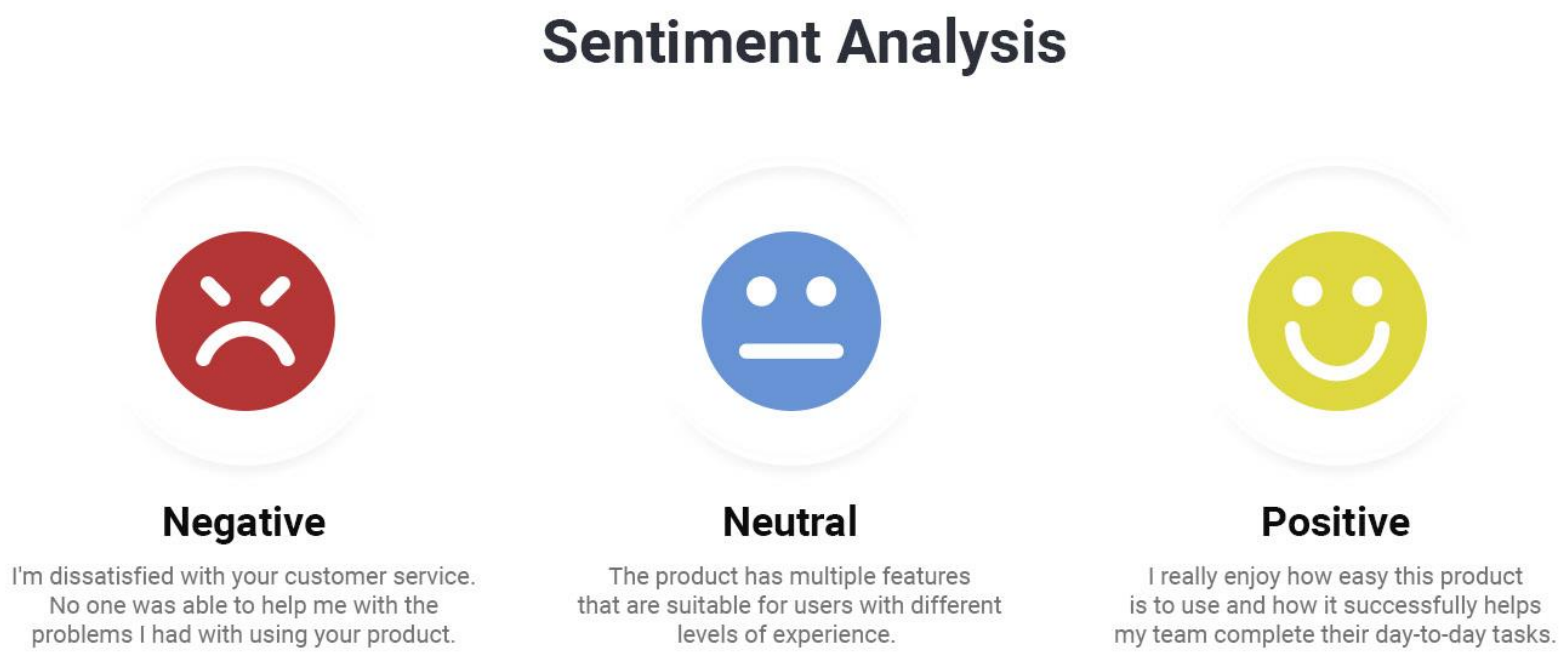
Applications of CNNs

NLP

a) Text classification and order
b) Sentiment analysis
c) Information extraction
d) Named-entity recognition
e) Speech recognition
f) Natural language understanding and generation (NLU)
g) Translate

Applications of CNNs

Information extraction

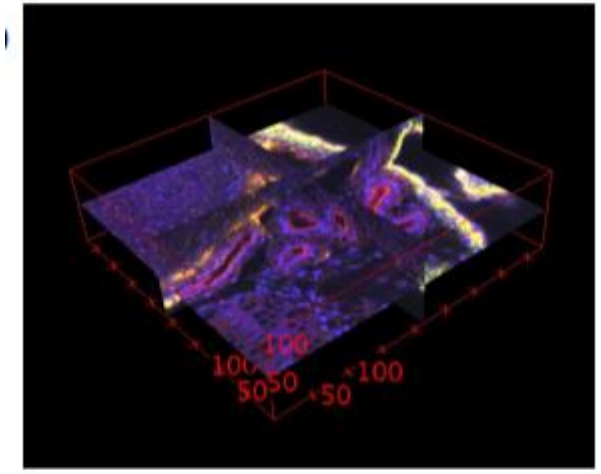


Applications of CNNs

COMPUTER VISION (CLASSIFICATION)

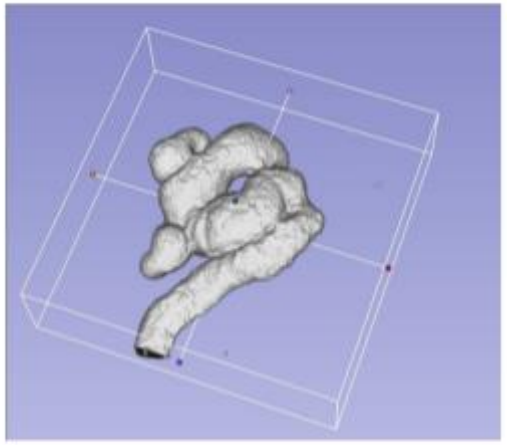
Applications of CNNs

COMPUTER VISION (SEGMENTATION)



raw image

apply trained 3D u-net



dense segmentation

Applications of CNNs

U-NET

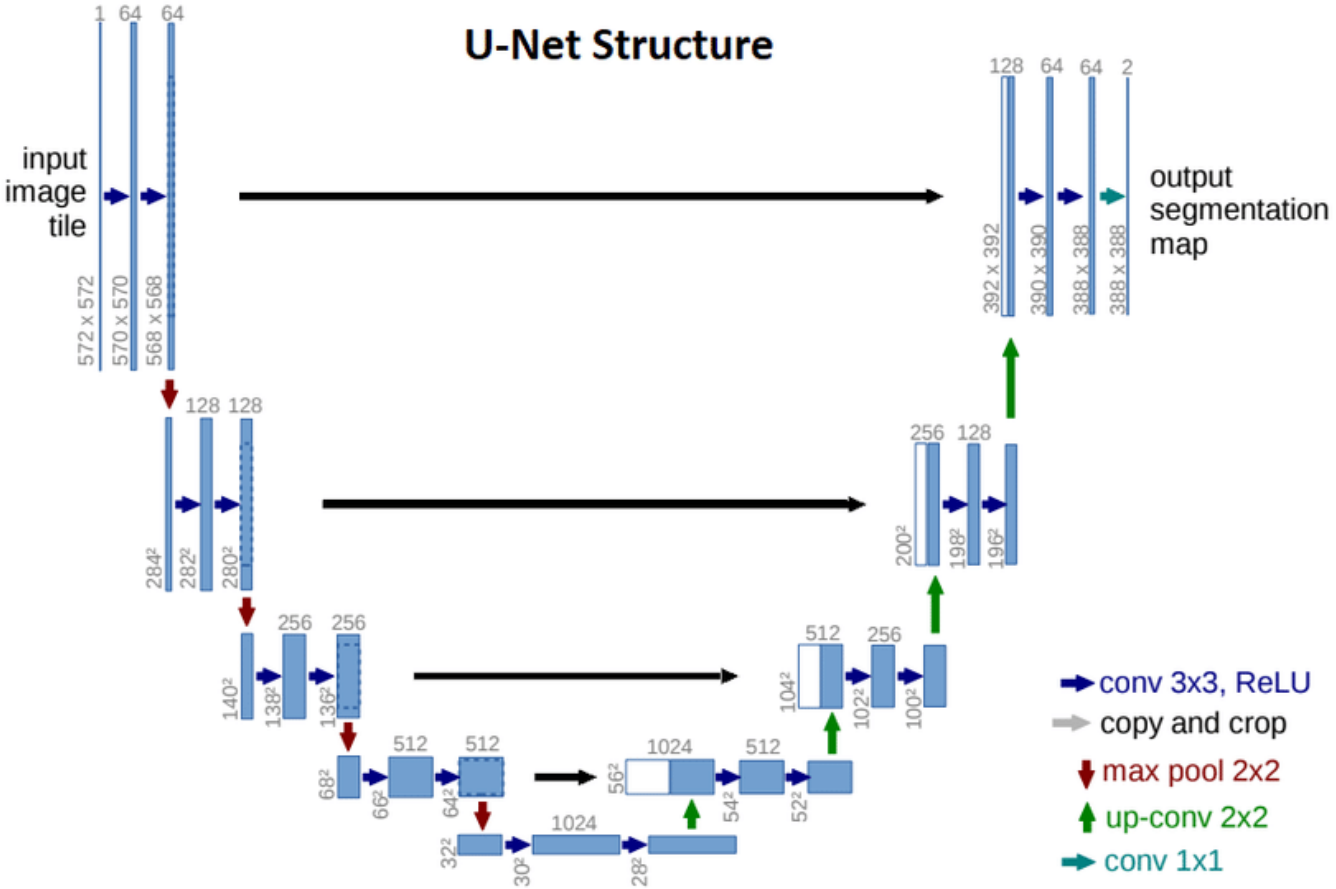


Image from Ronneberger et al. in 2015

Applications of CNNs

COMPUTER VISION (SEGMENTATION)

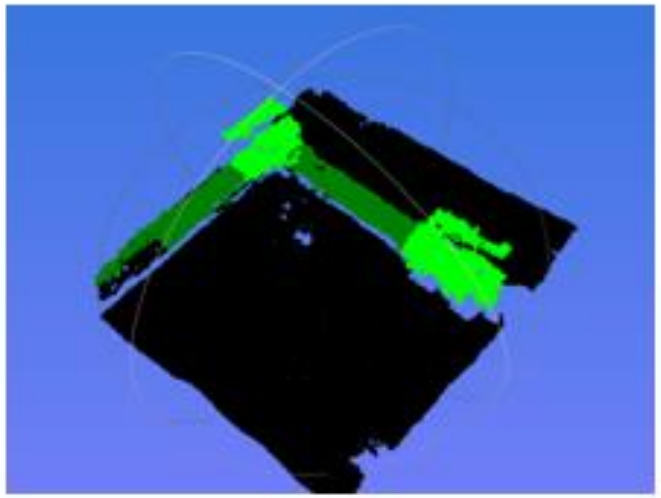


Fig 2 Segmentación semántica 3D

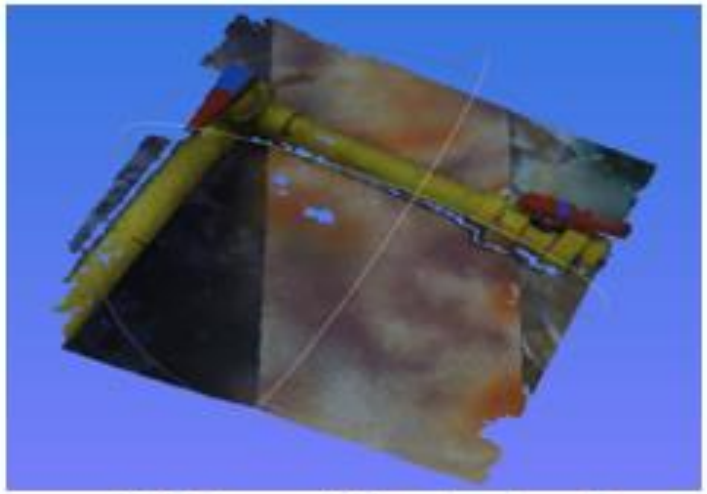


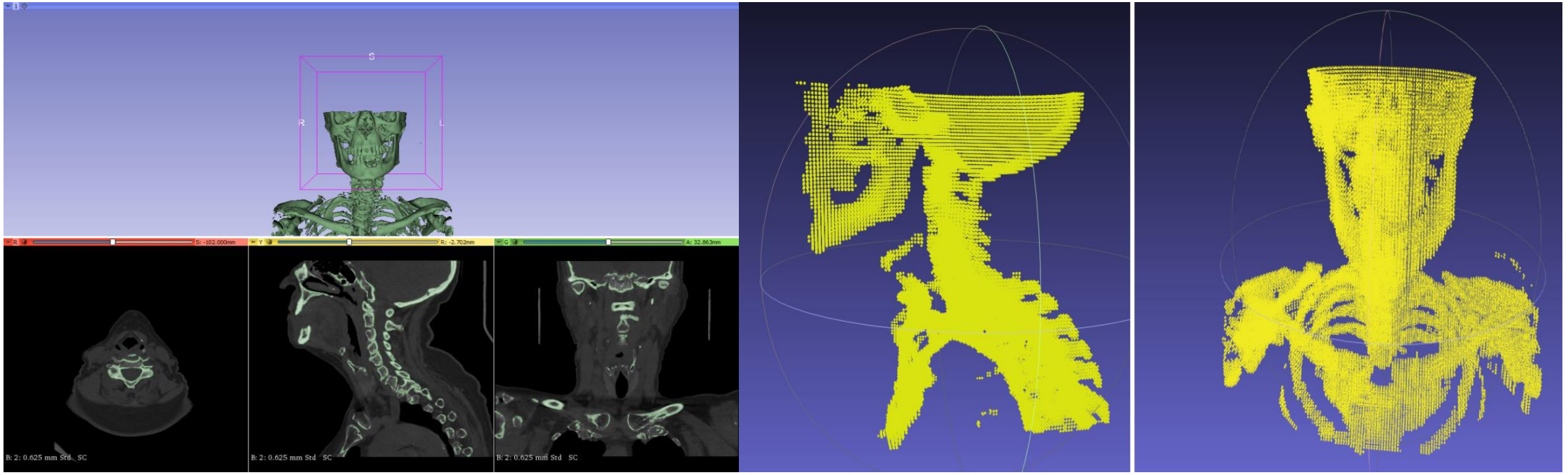
Fig 3 Imagen3D real submarina

Applications of CNNs

COMPUTER VISION (SEGMENTATION)

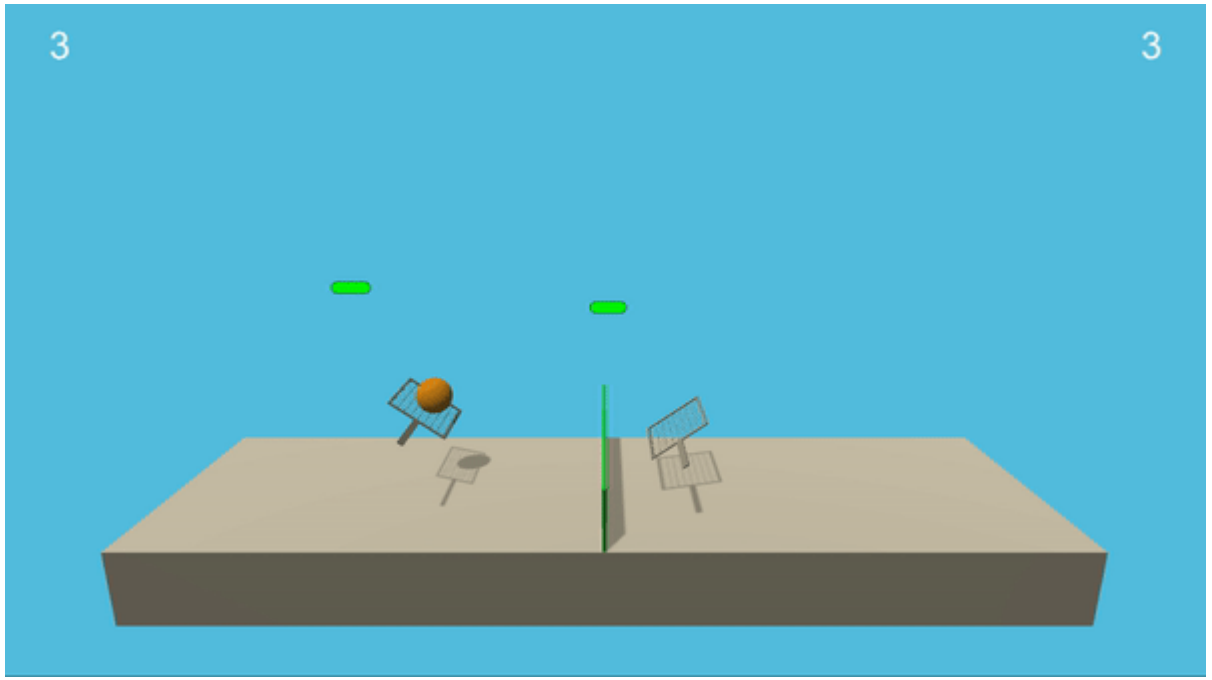
Applications of CNNs

COMPUTER VISION (SEGMENTATION)



Applications of CNNs

REINFORCEMENT LEARNING



Applications of CNNs

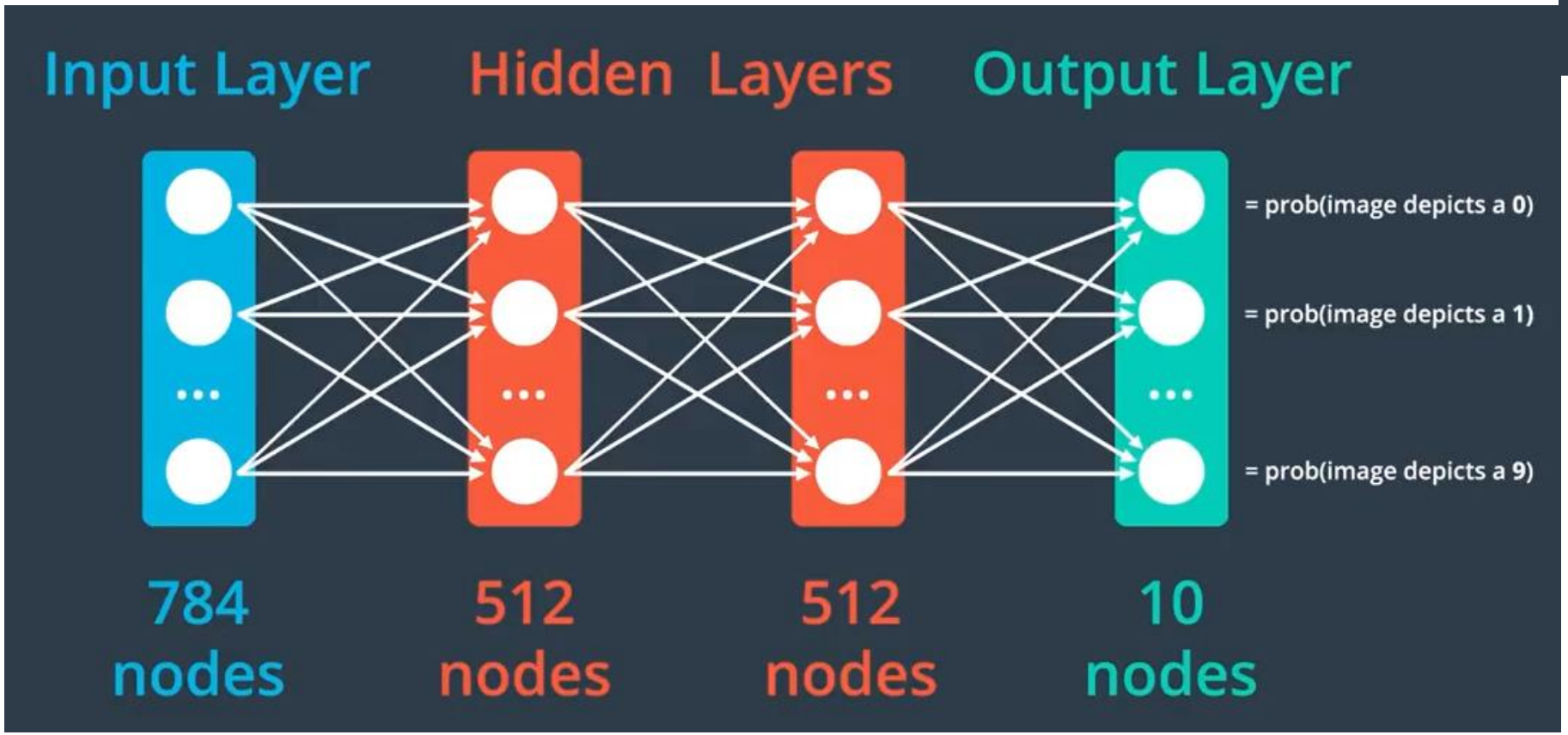
REINFORCEMENT LEARNING

Applications of CNNs

ROBOTICS

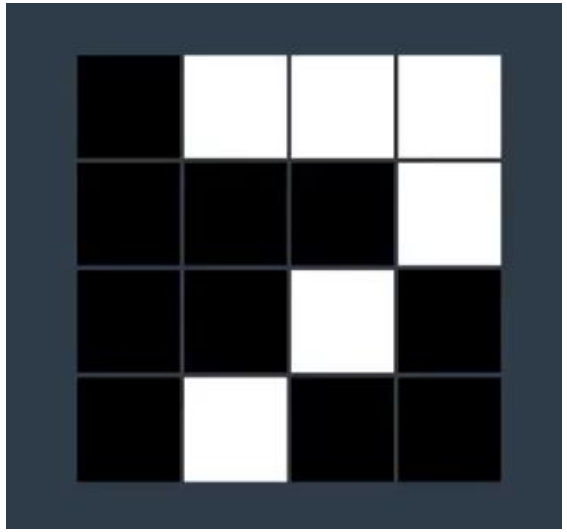
MLP vs CNNs

When do MLPs (not) work well?



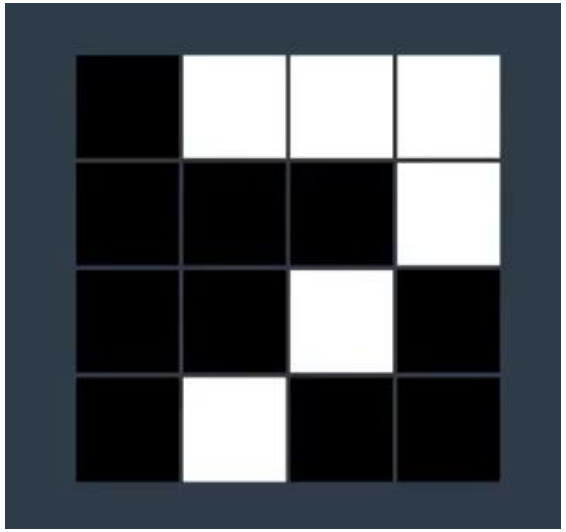
MLP vs CNNs

When do MLPs (not) work well?



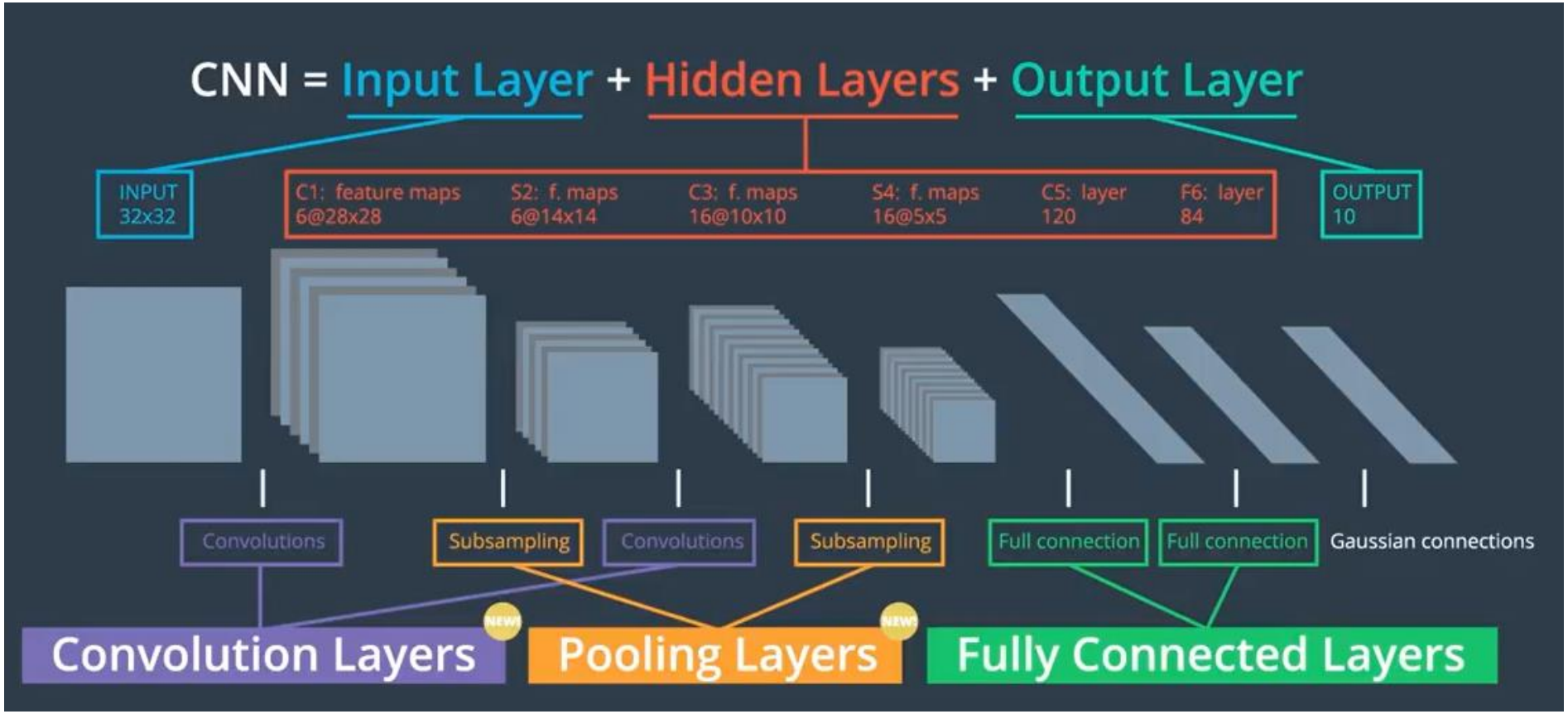
MLP vs CNNs

MLPs Input

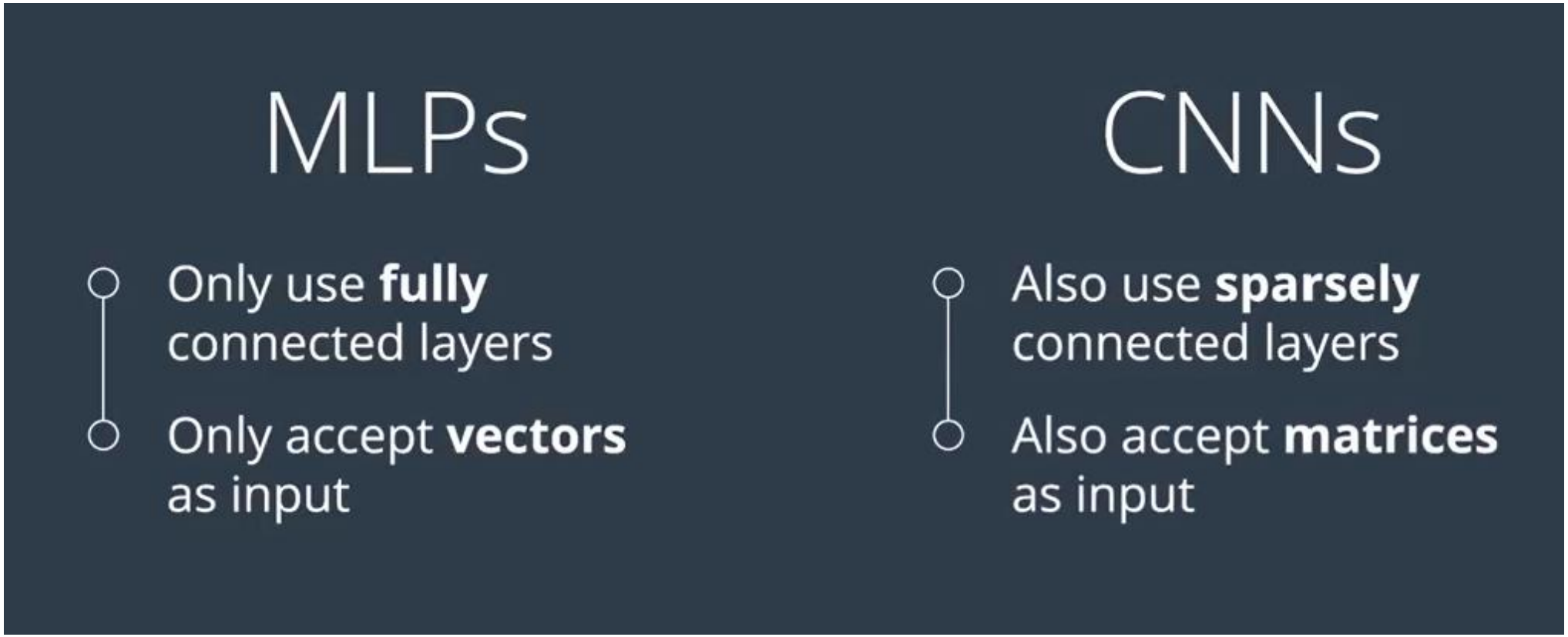


MLP vs CNNs

More differences

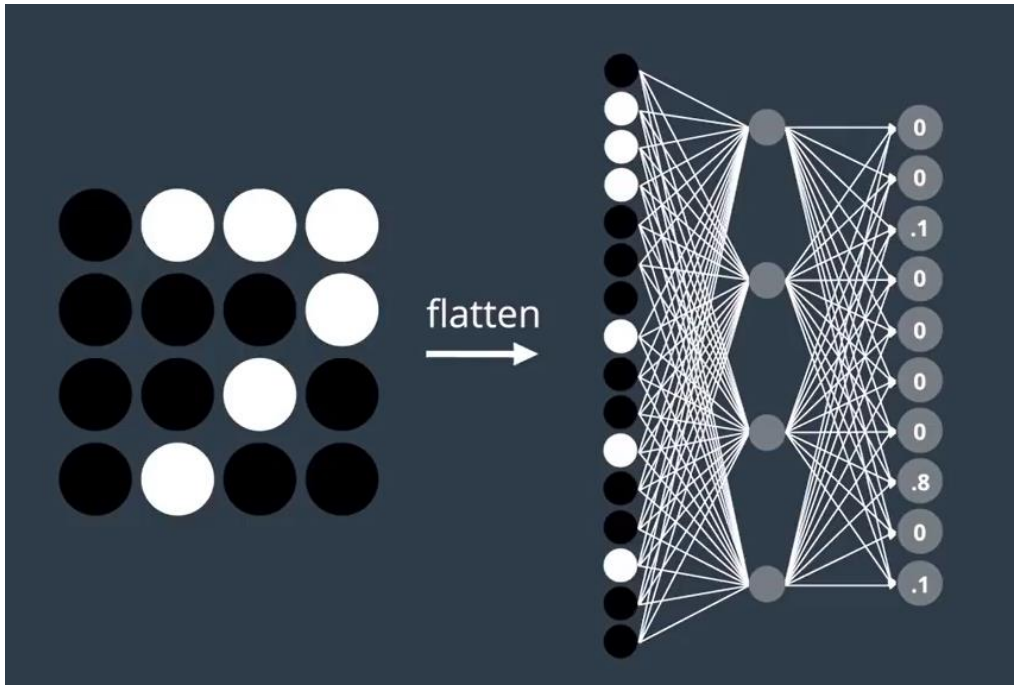


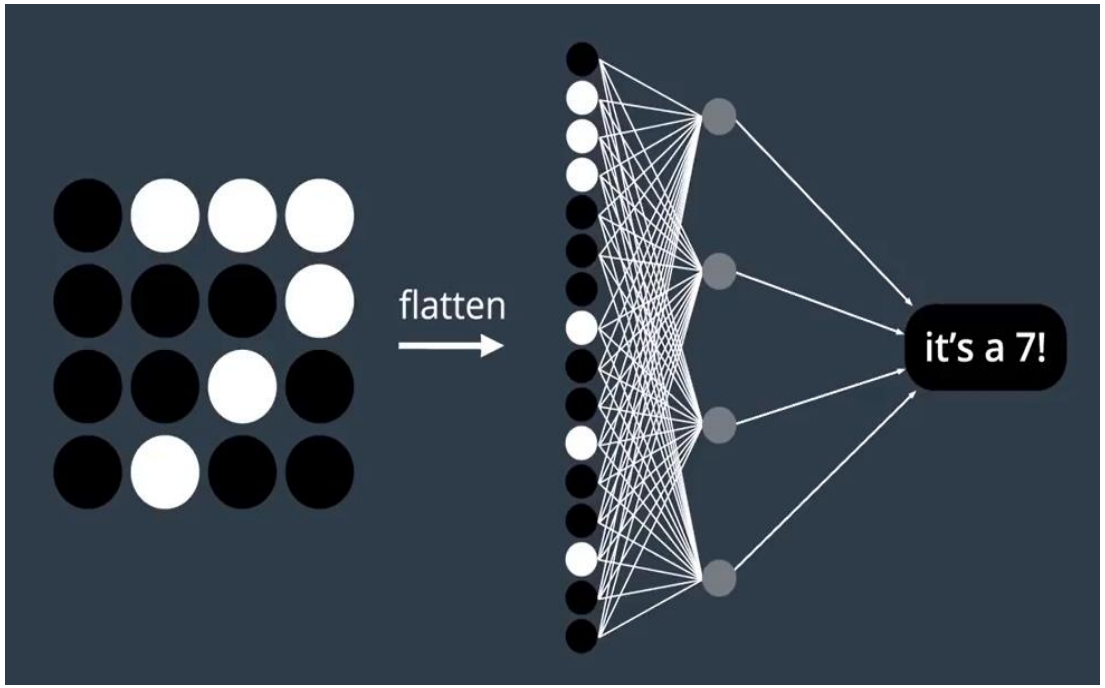
MLP vs CNNs


MLPs
Only use **fully** connected layers
Only accept **vectors** as input
CNNs
Also use **sparsely** connected layers
Also accept **matrices** as input

CNNs

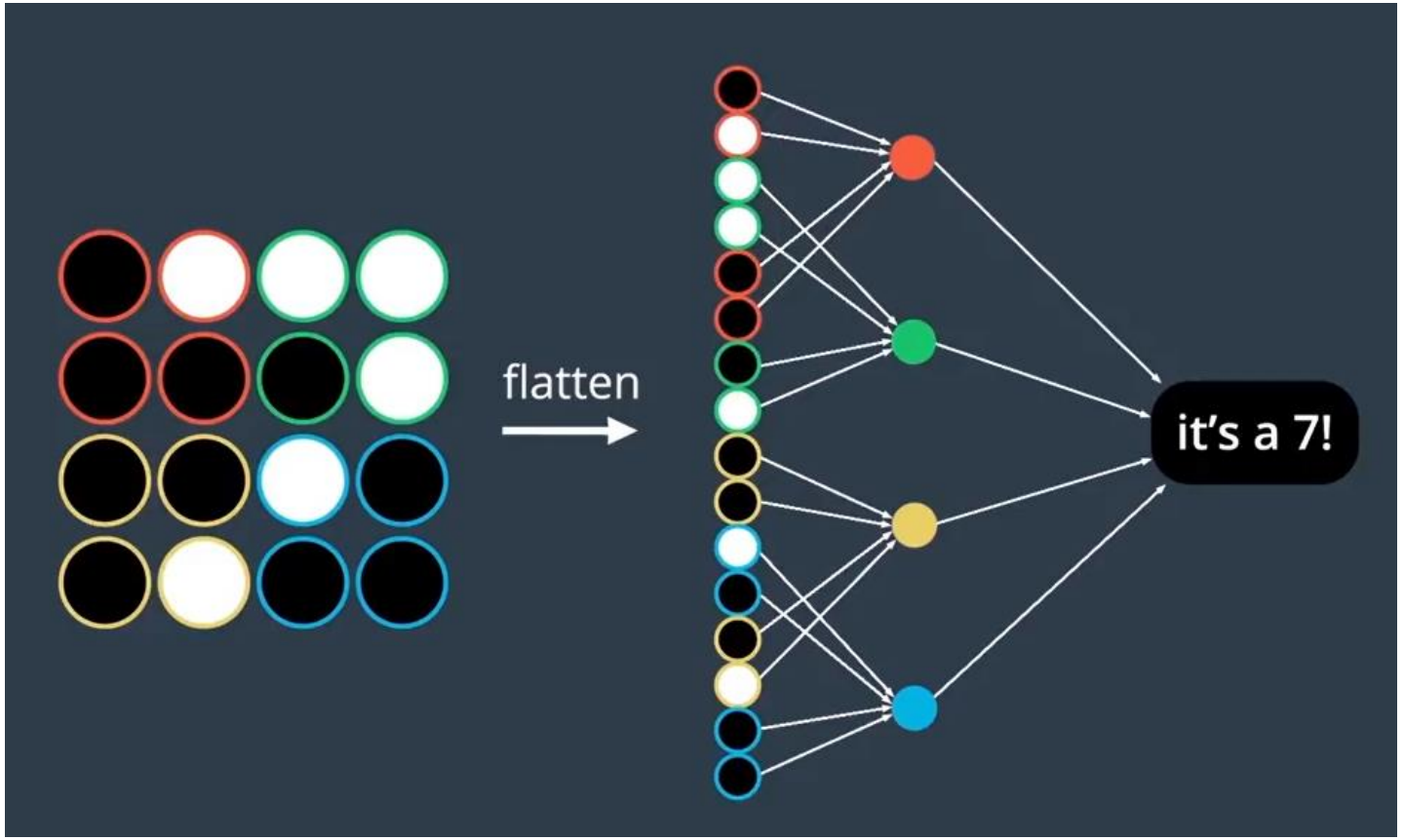
Local connectivity





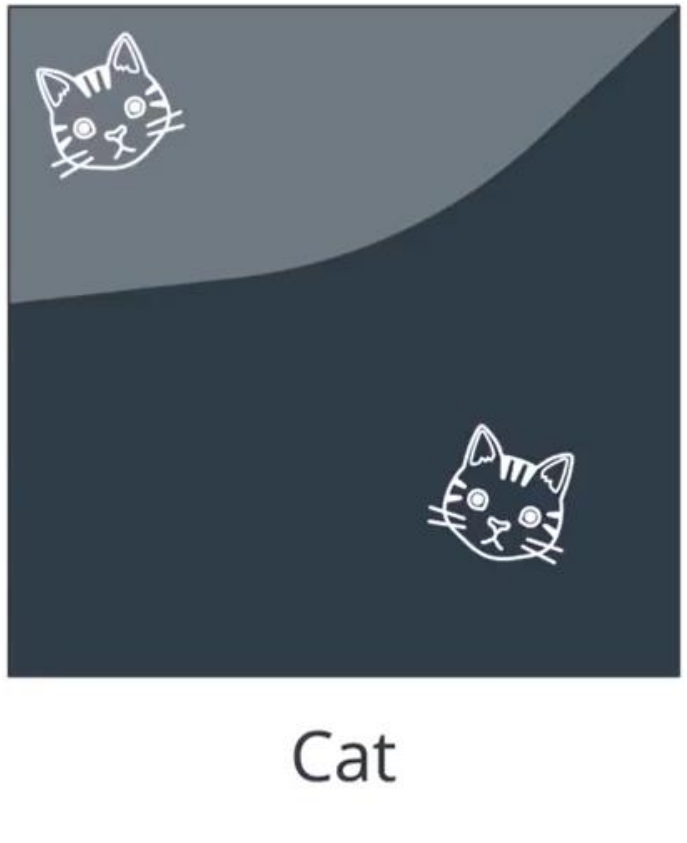
CNNs

Local connectivity



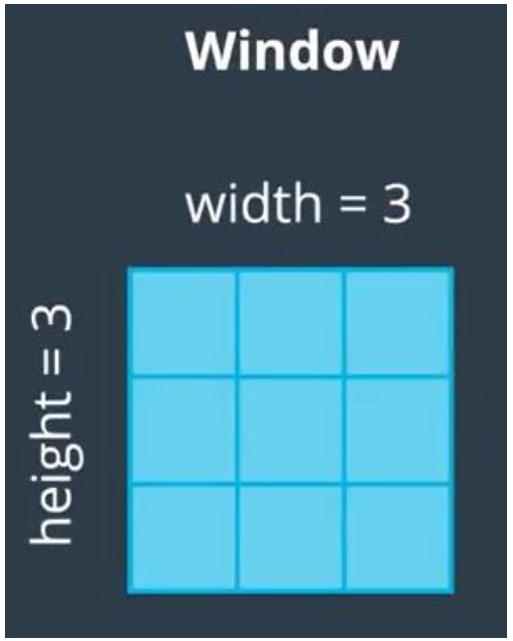
CNNs

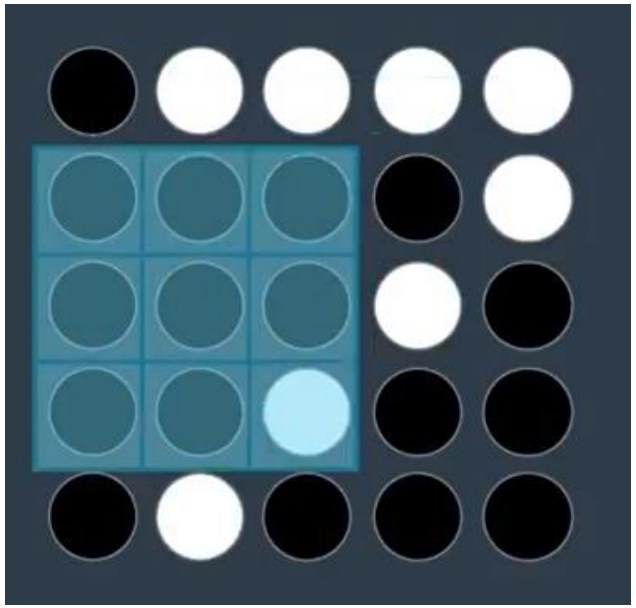
Local connectivity

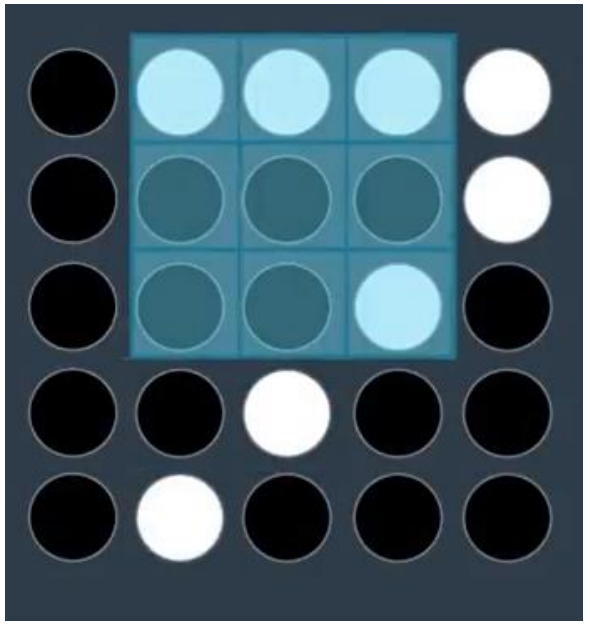


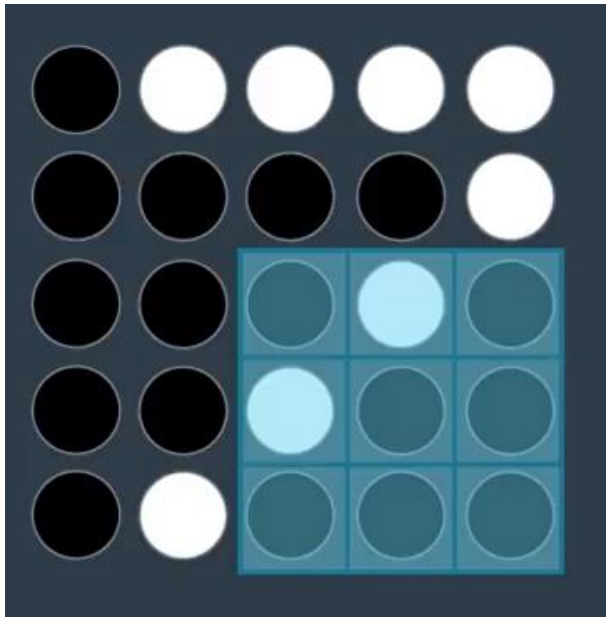
CNNs

Convolutional layer



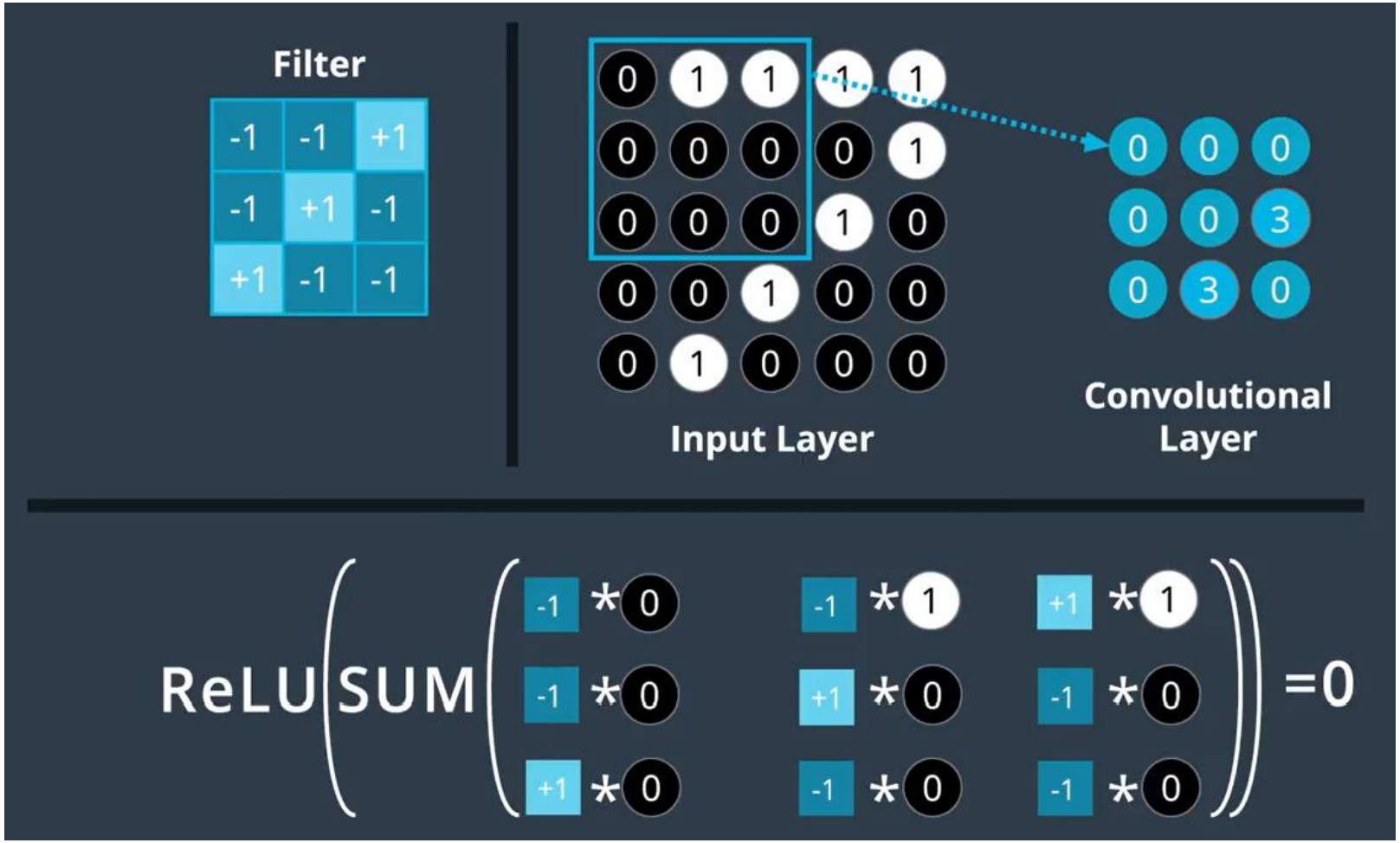






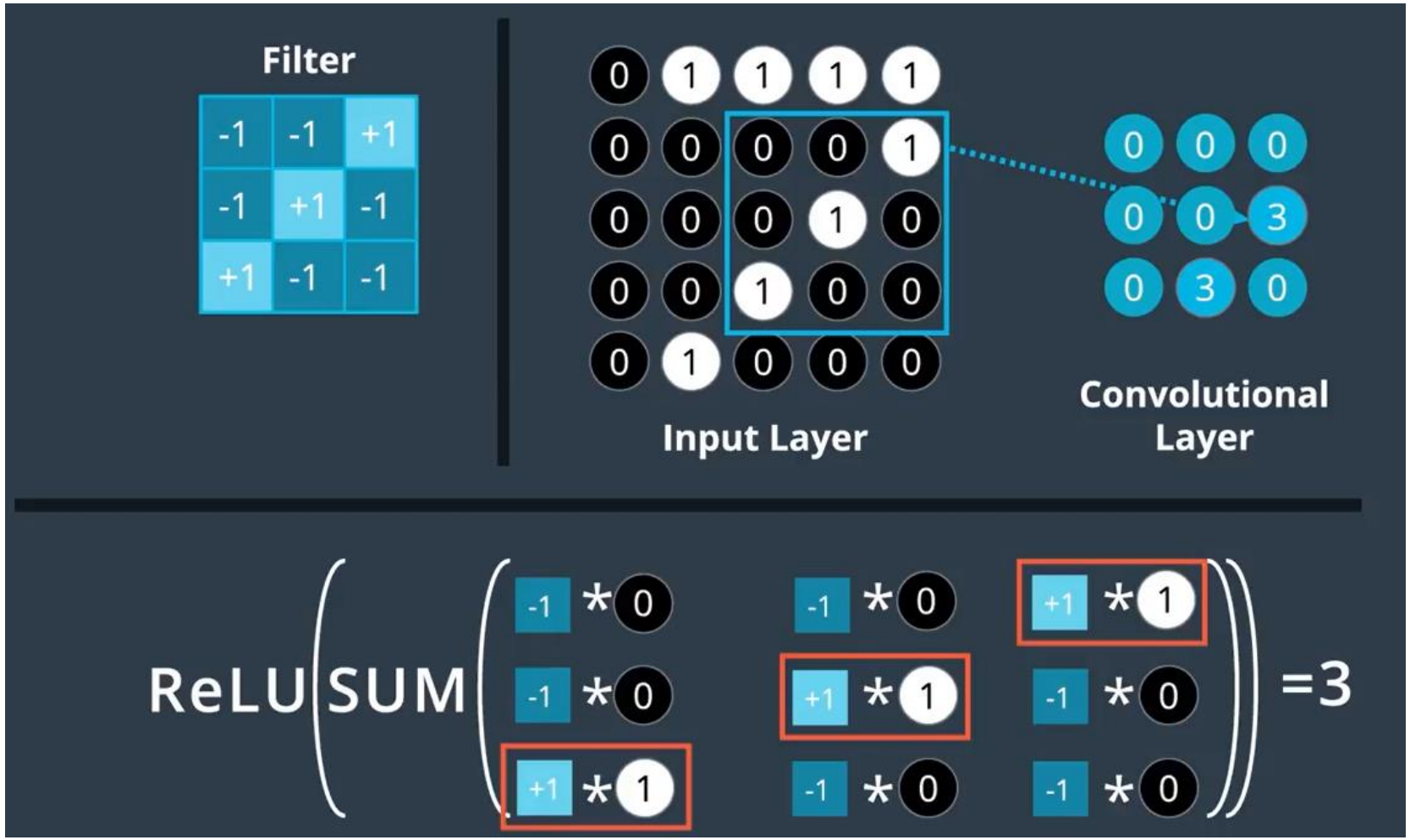
CNNs

Convolutional layer



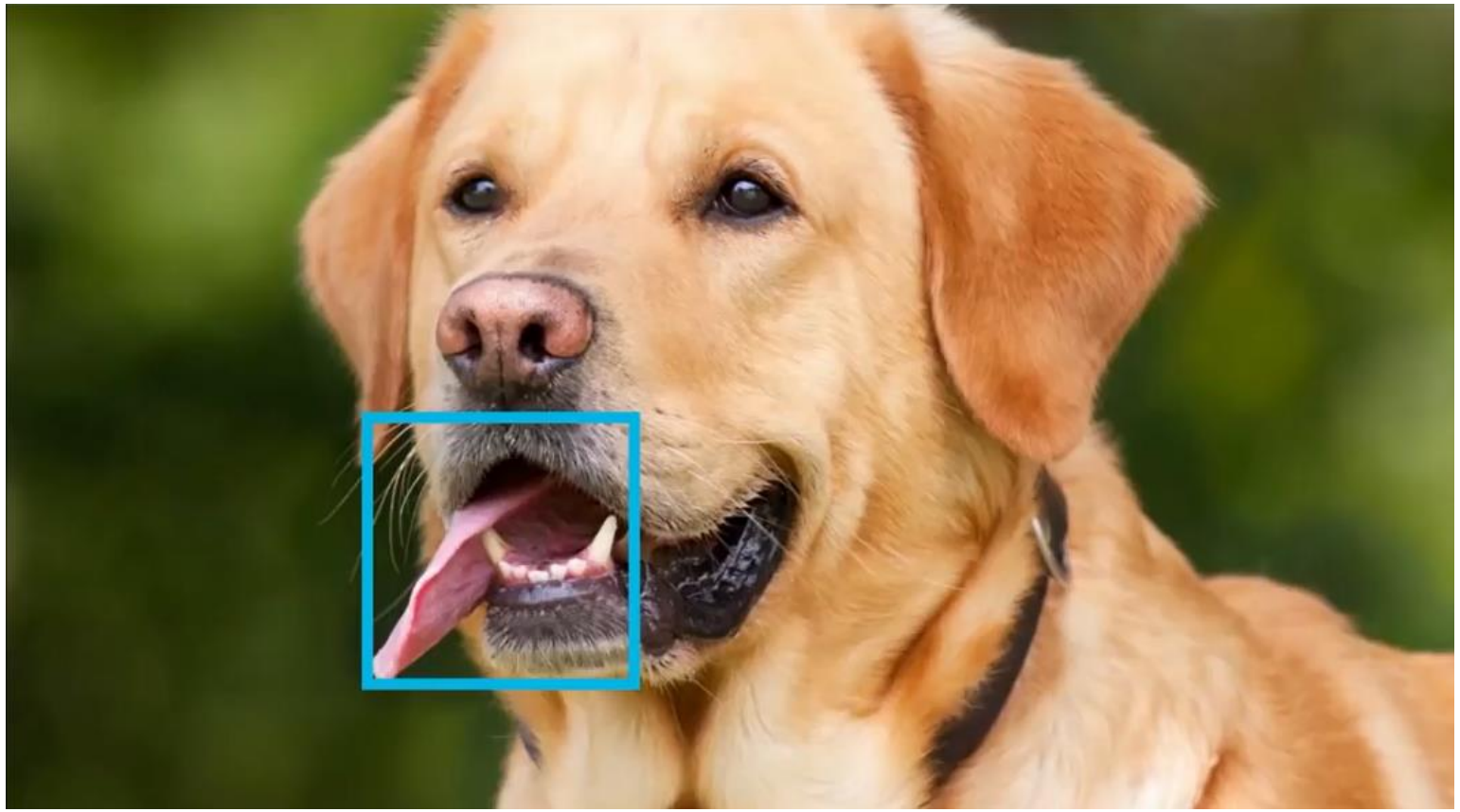
CNNs

Convolutional layer



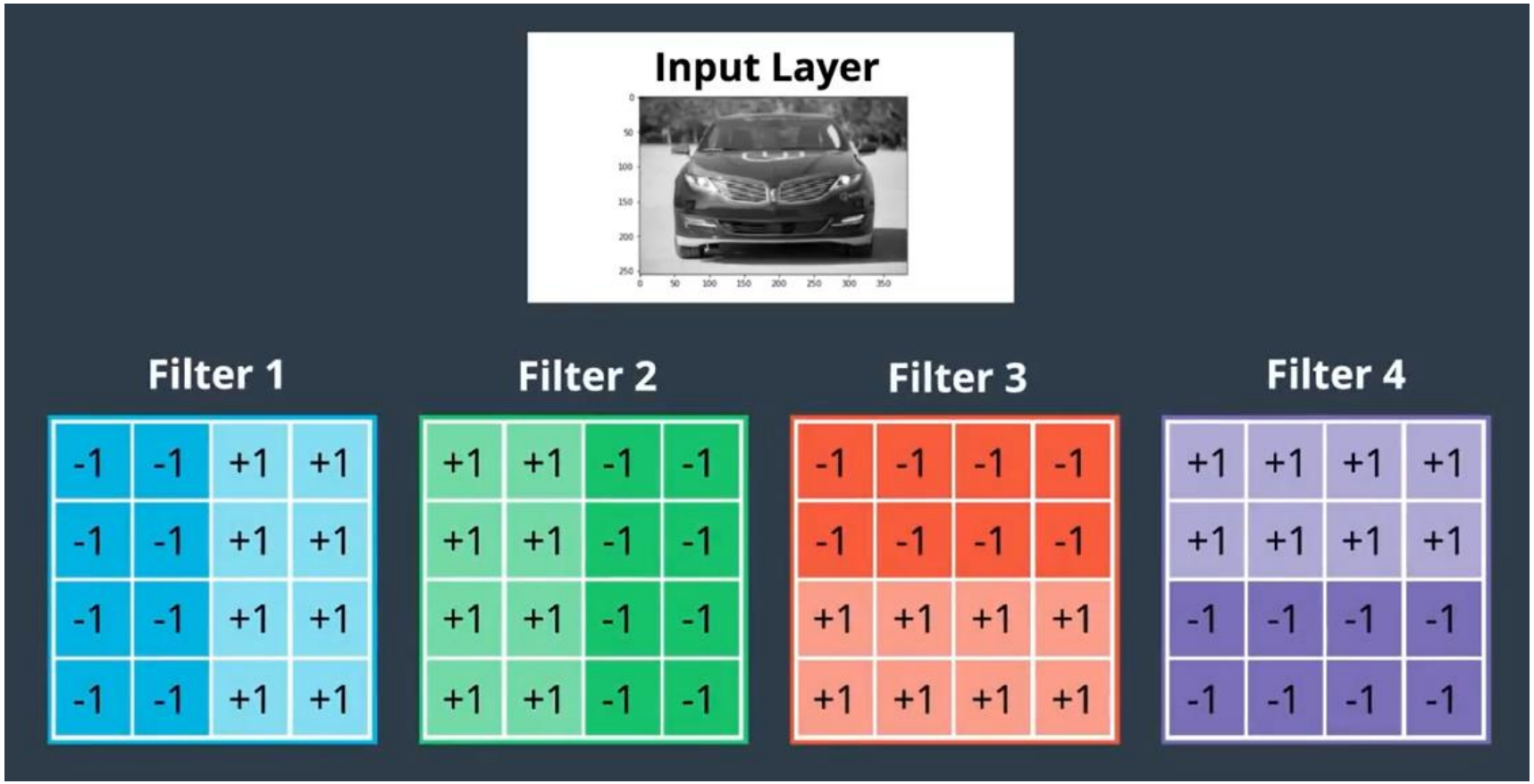
CNNs

Convolutional layer



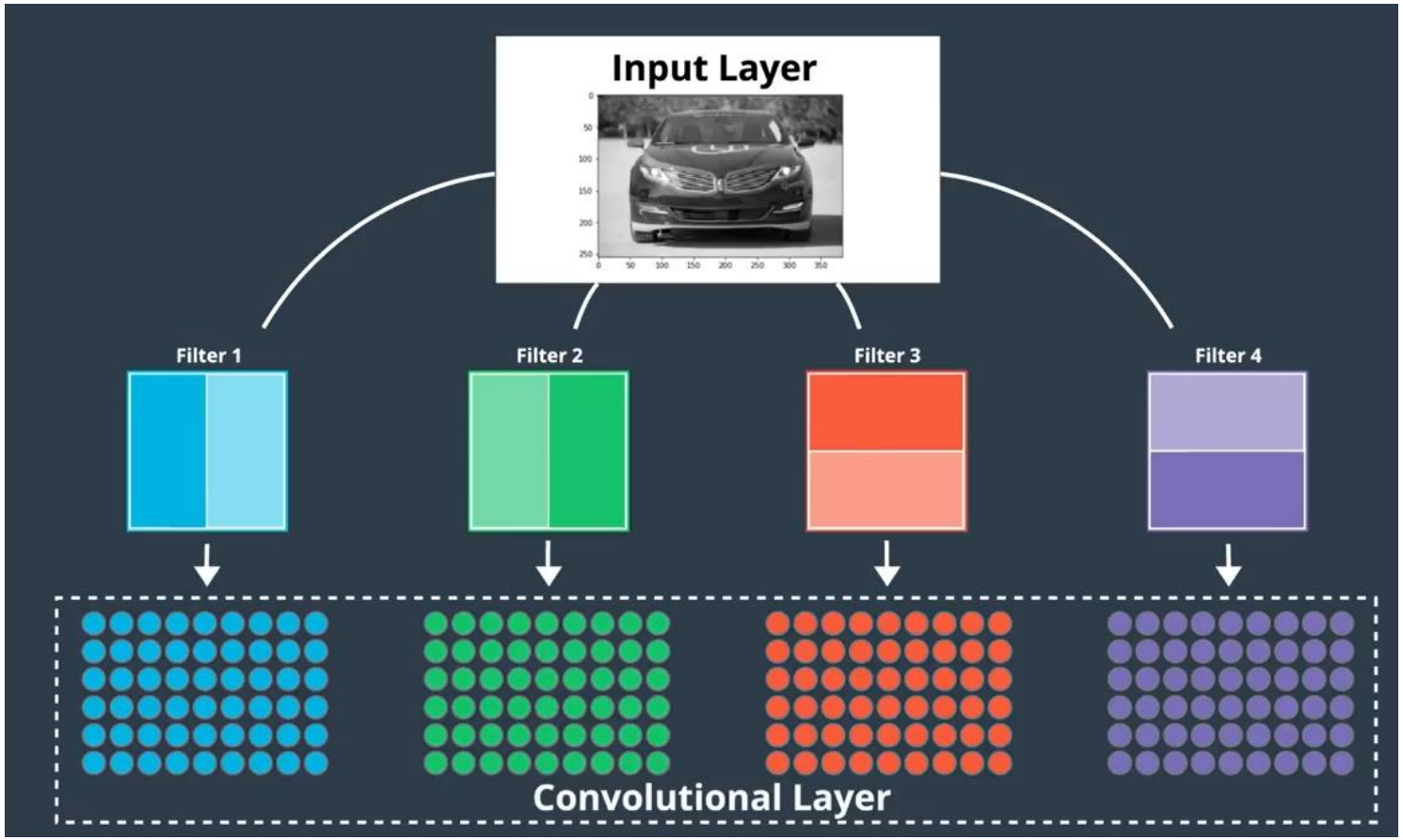
CNNs

Convolutional layer



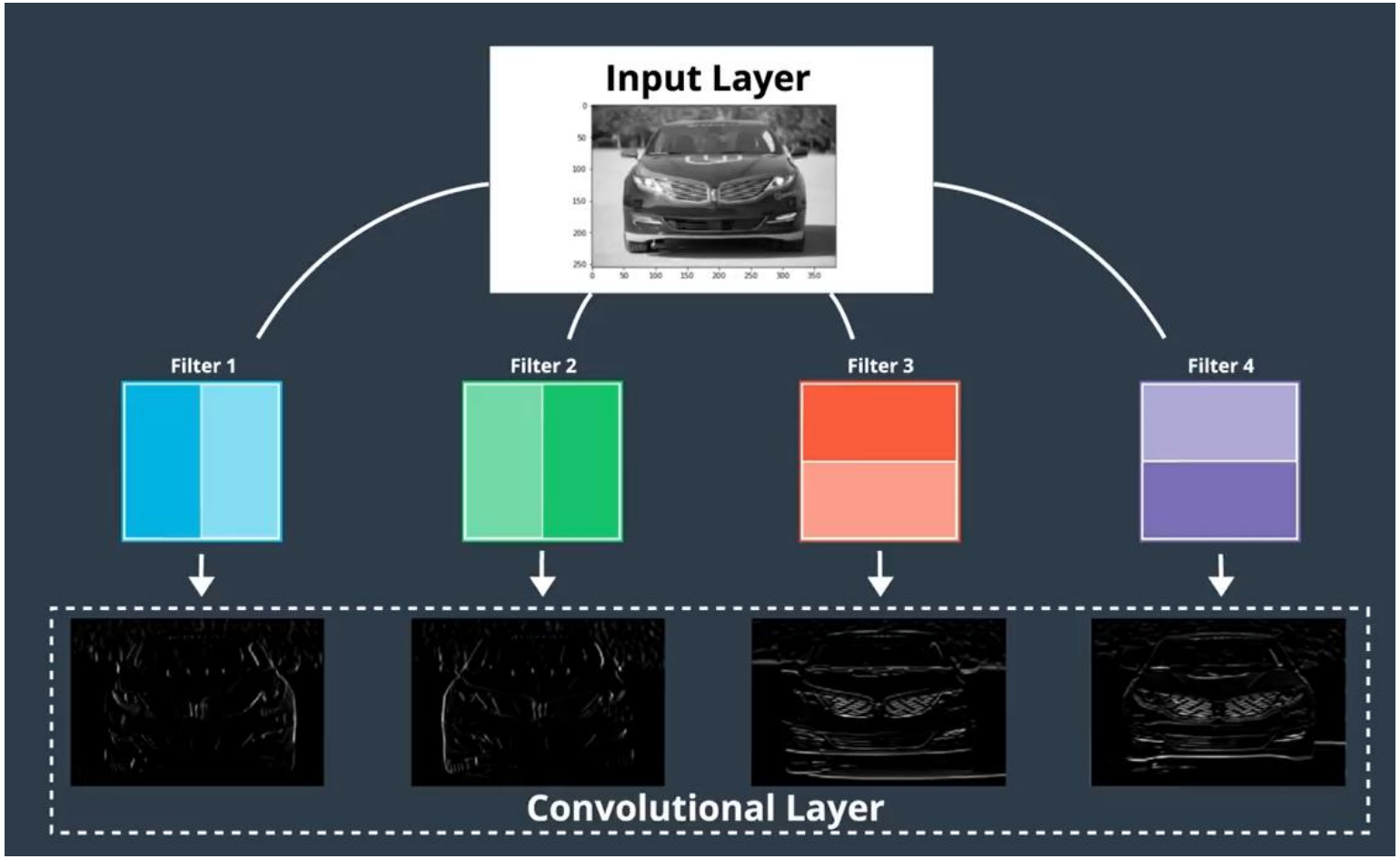
CNNs

Convolutional layer



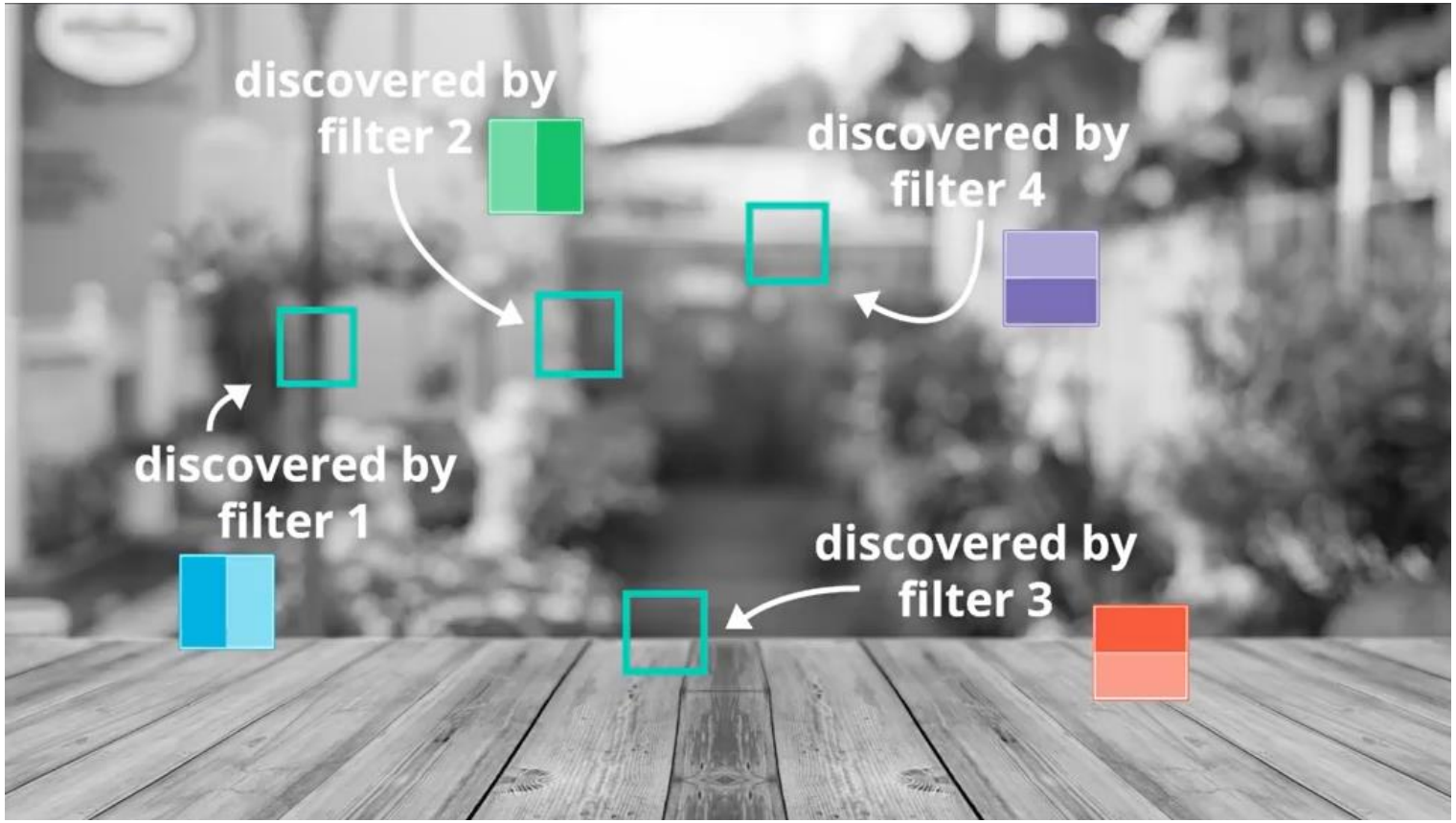
CNNs

Convolutional layer



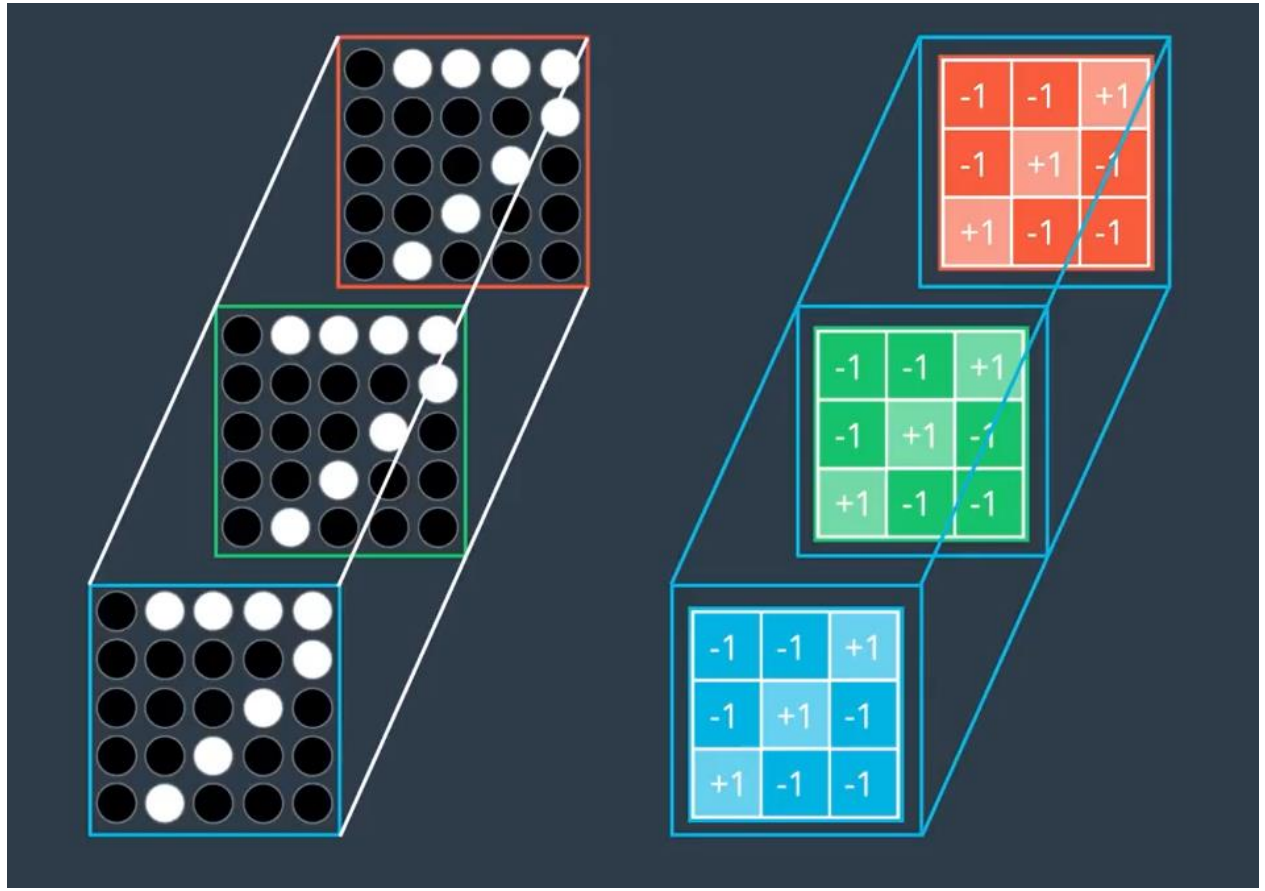
CNNs

Convolutional layer



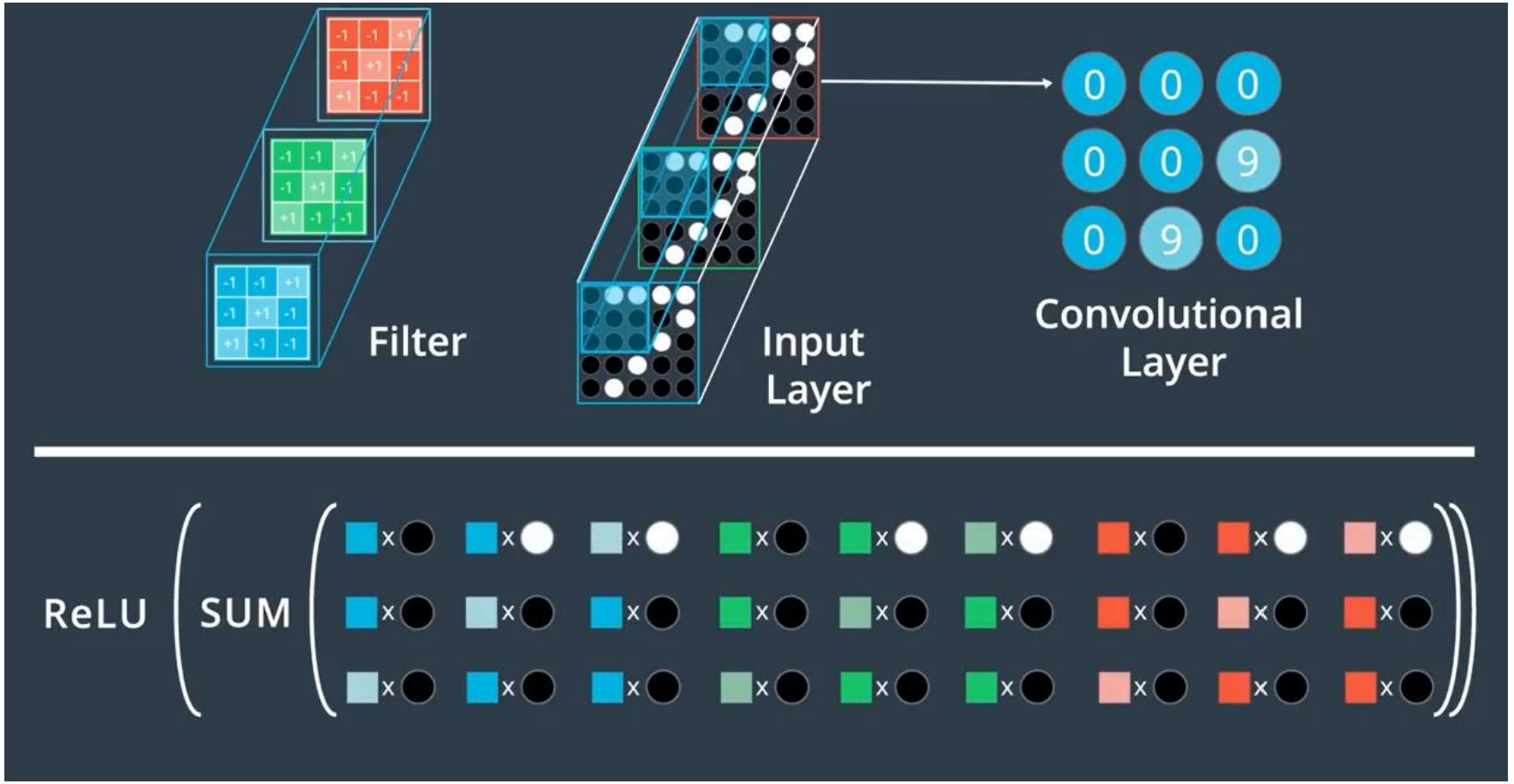
CNNs

Convolutional layer (color images)



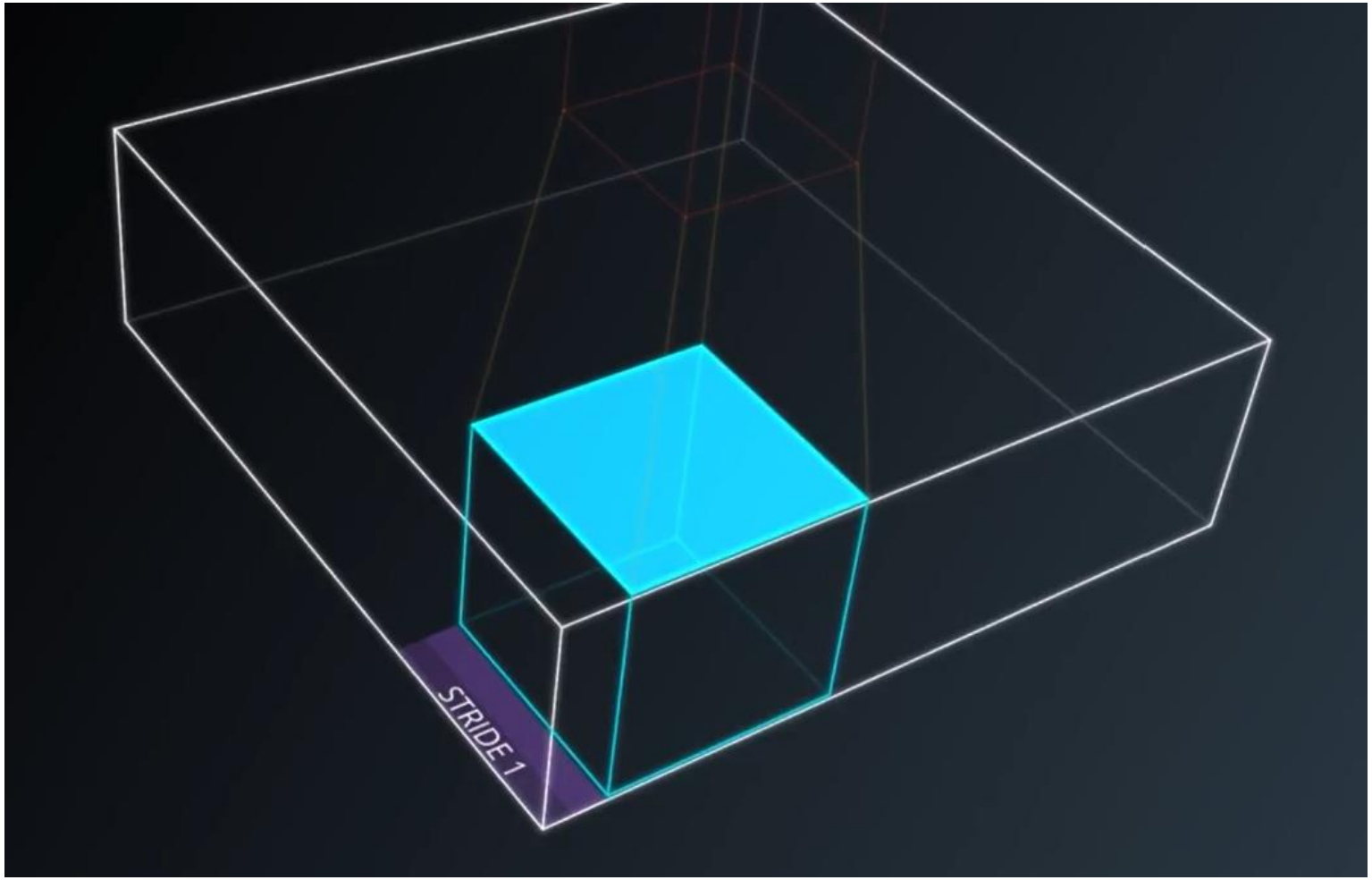
CNNs

Convolutional layer (color images)



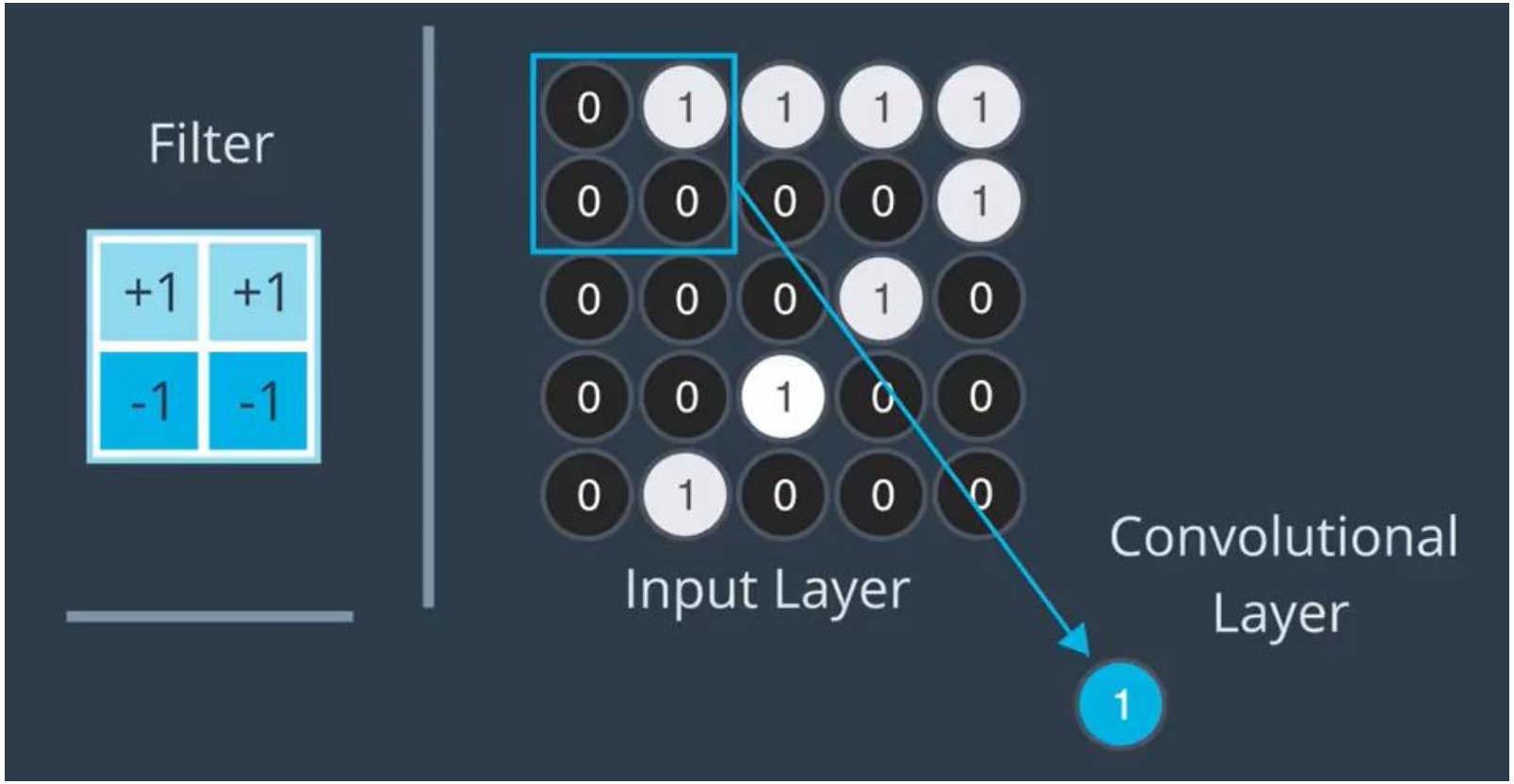
CNNs

Stride



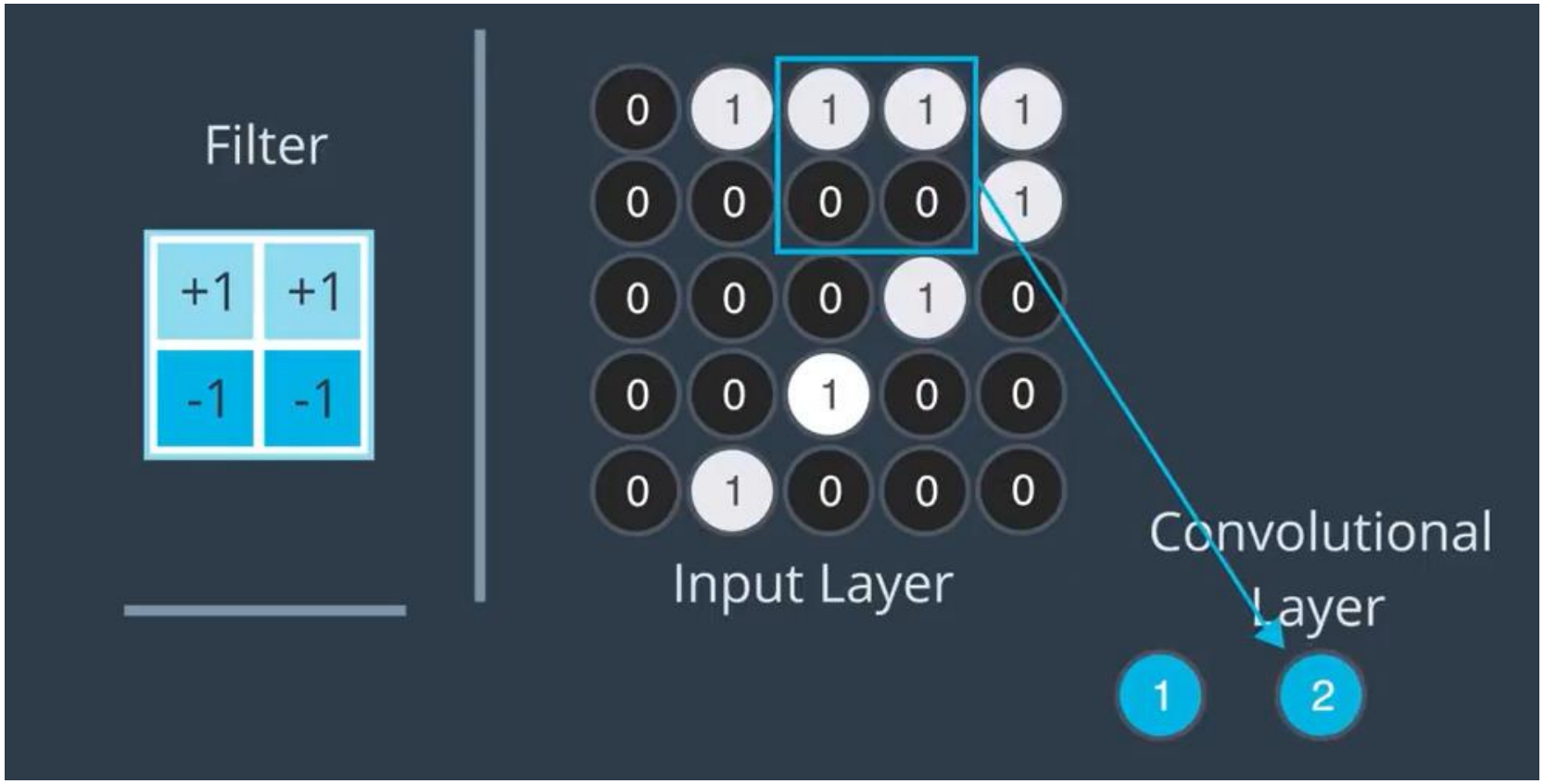
CNNs

Stride


Filter
+1
+1
-1
-1
0
1
1
1
1
0
0
0
0
1
0
0
0
1
0
0
0
1
0
0
0
1
0
0
0
Input Layer
Convolutional Layer
1

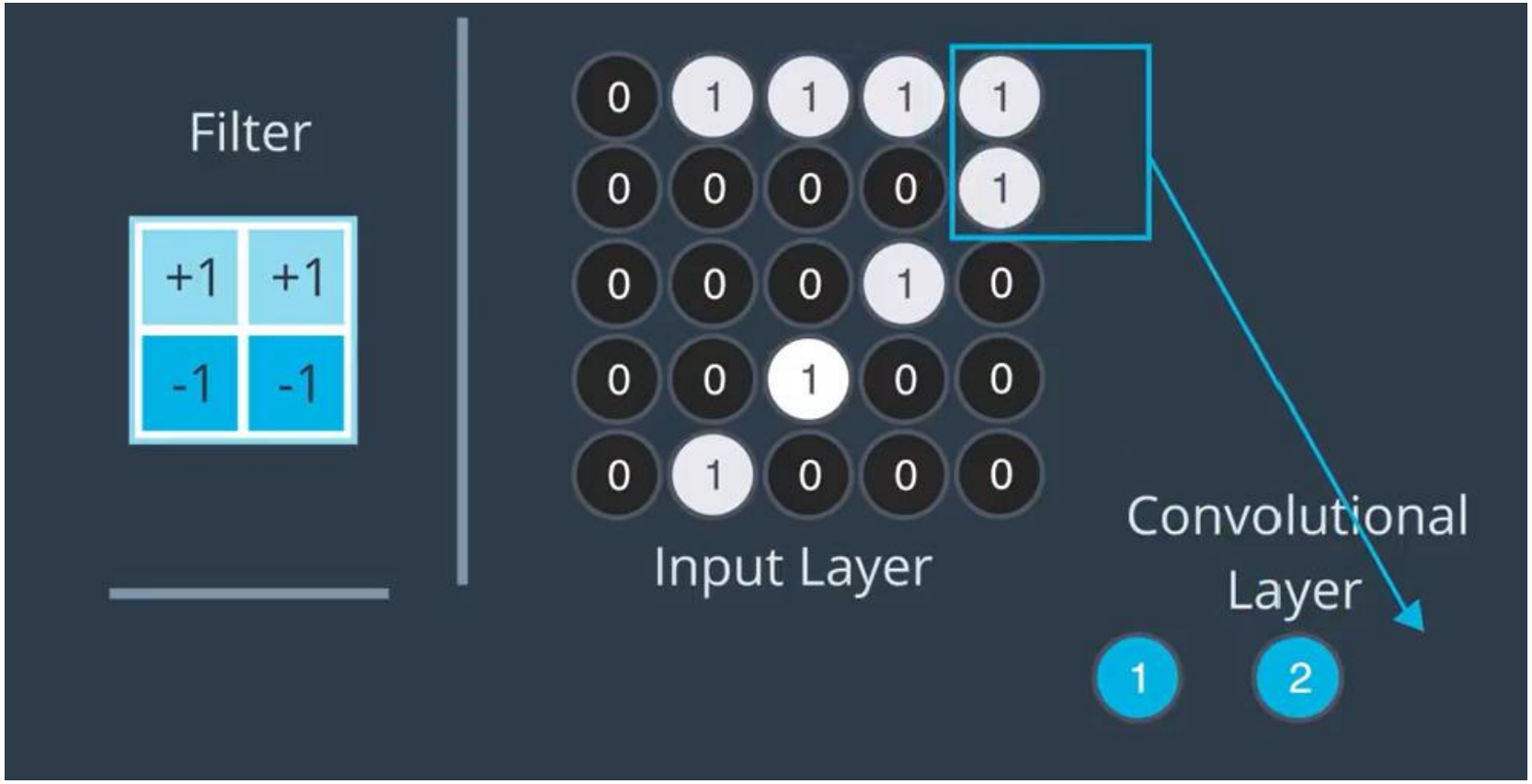
CNNs

Stride


Filter
+1
+1
-1
-1
0
1
1
1
1
0
0
0
0
1
0
0
0
1
0
0
0
1
0
0
0
1
0
0
Input Layer
Convolutional Layer
1
2

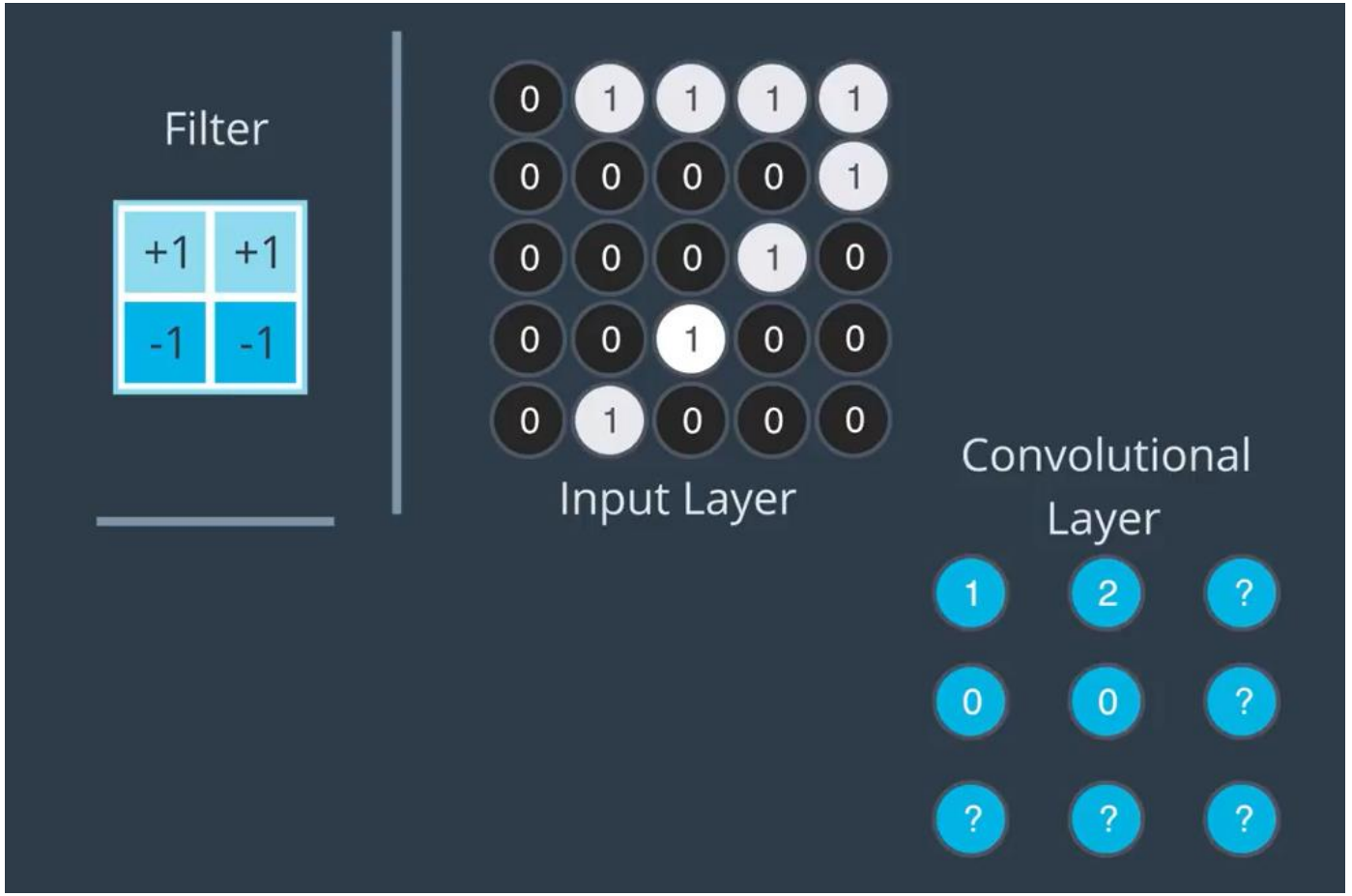
CNNs

Stride


Filter
+1
+1
-1
-1
0 1 1 1 1
0 0 0 0 1
0 0 0 1 0
0 0 1 0 0
0 1 0 0 0
Input Layer
Convolutional Layer
1
2

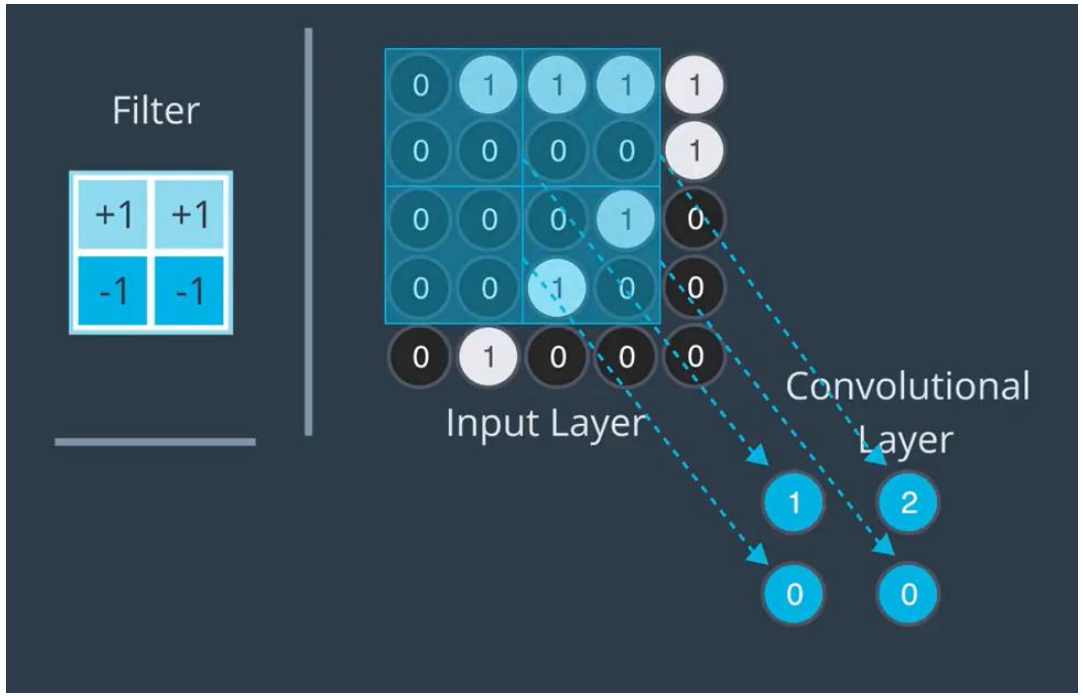
CNNs

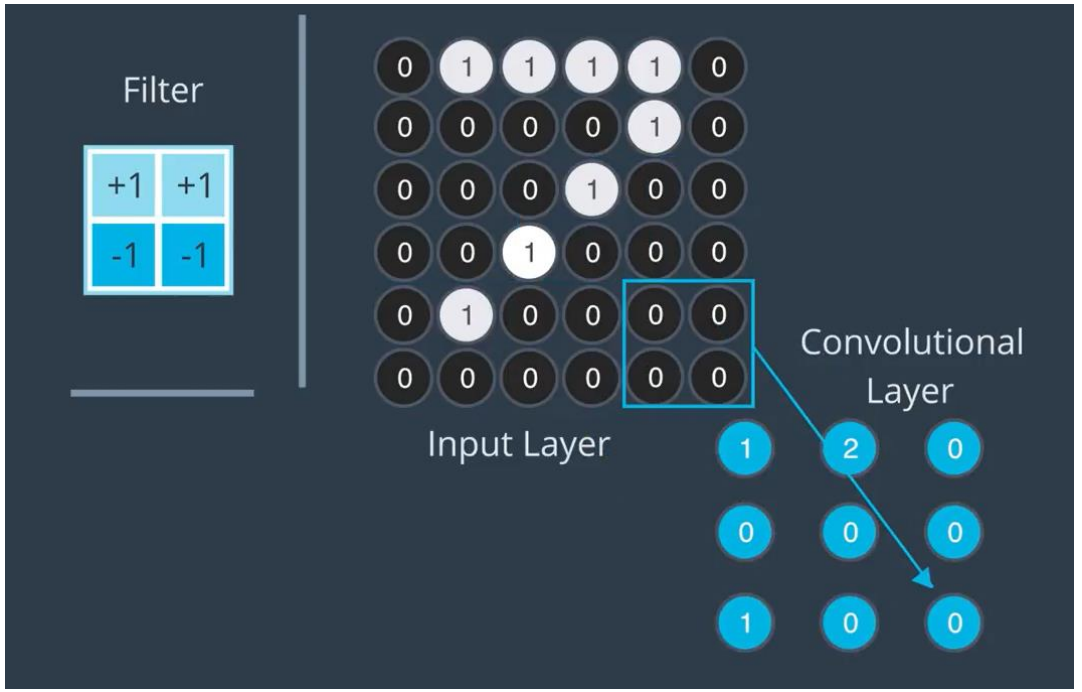
Stride


Filter
+1 +1
-1 -1
0 1 1 1 1
0 0 0 0 1
0 0 0 1 0
0 0 1 0 0
0 1 0 0 0
Input Layer
Convolutional Layer
1 2 ?
0 0 ?
? ? ?

CNNs

Padding



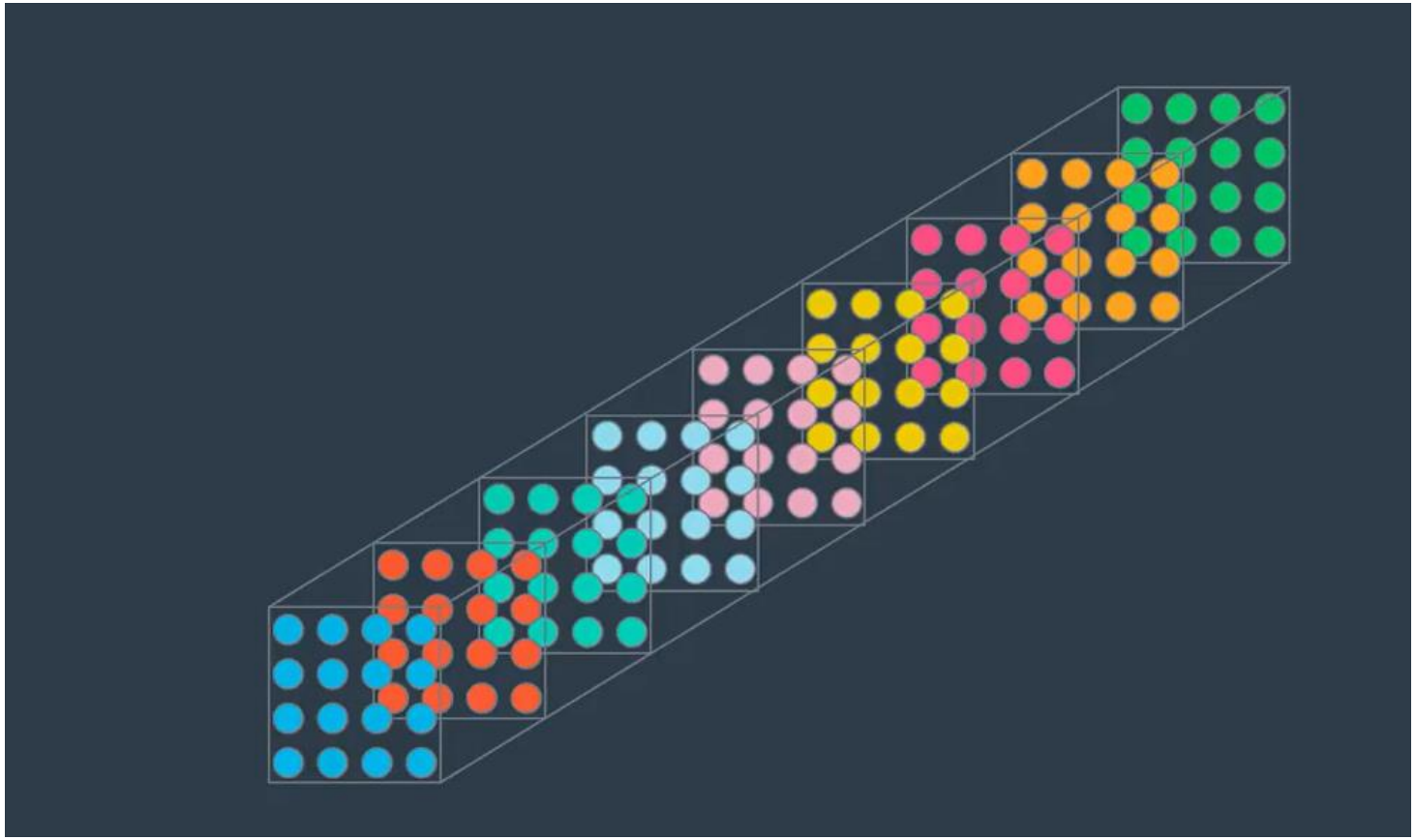


CNNs

Pooling layers

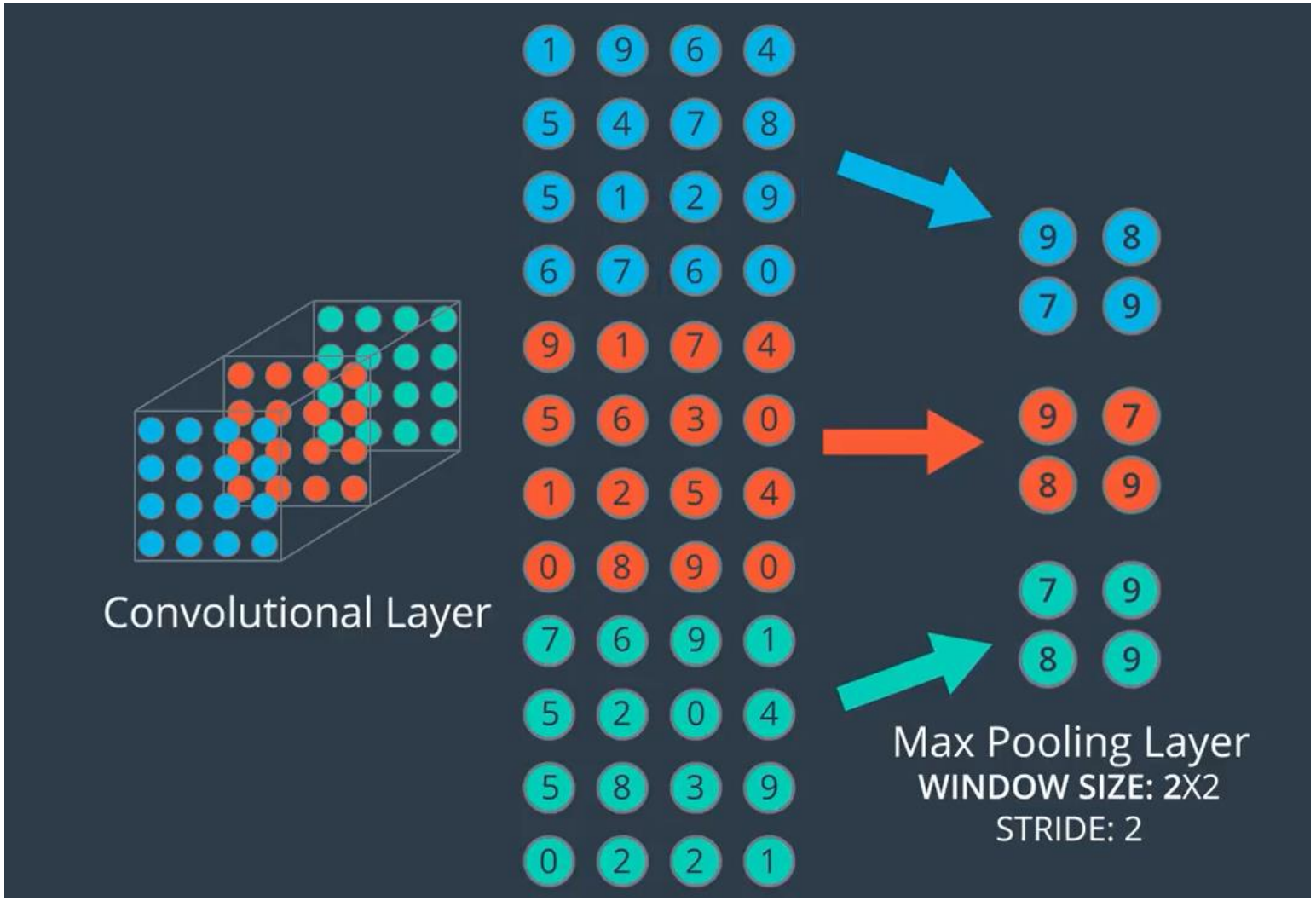
CNNs

Pooling layers



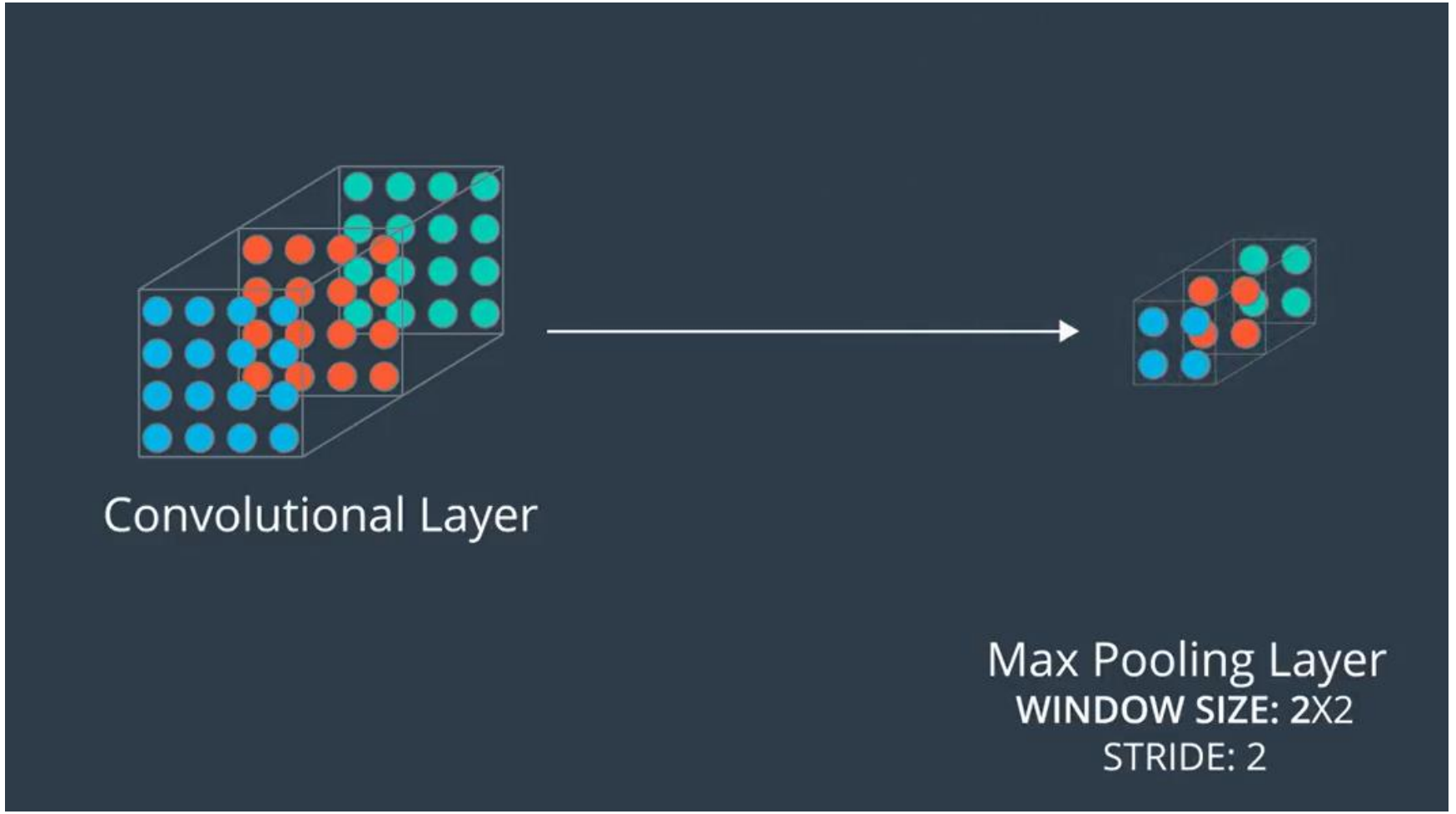
CNNs

Pooling layers



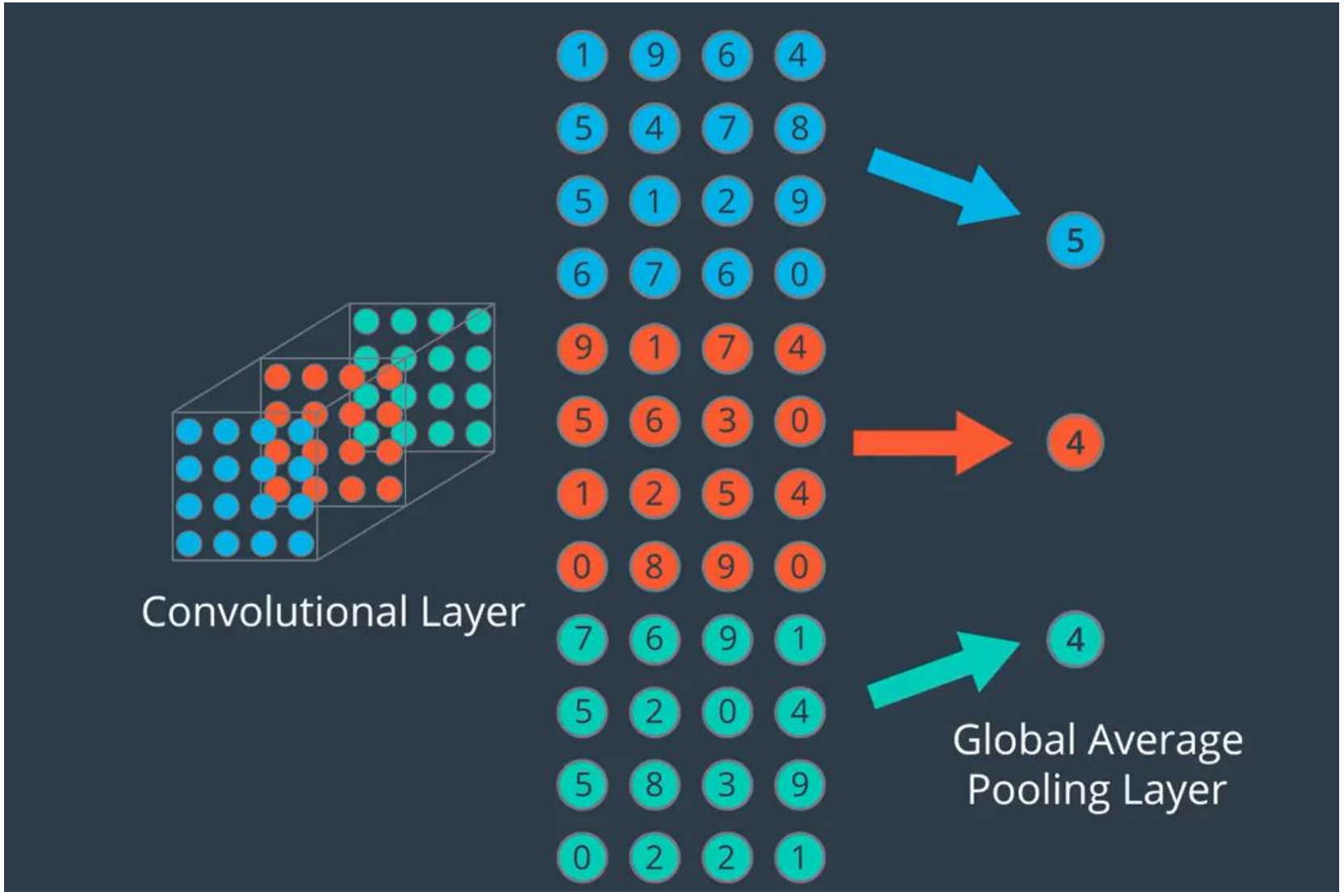
CNNs

Pooling layers



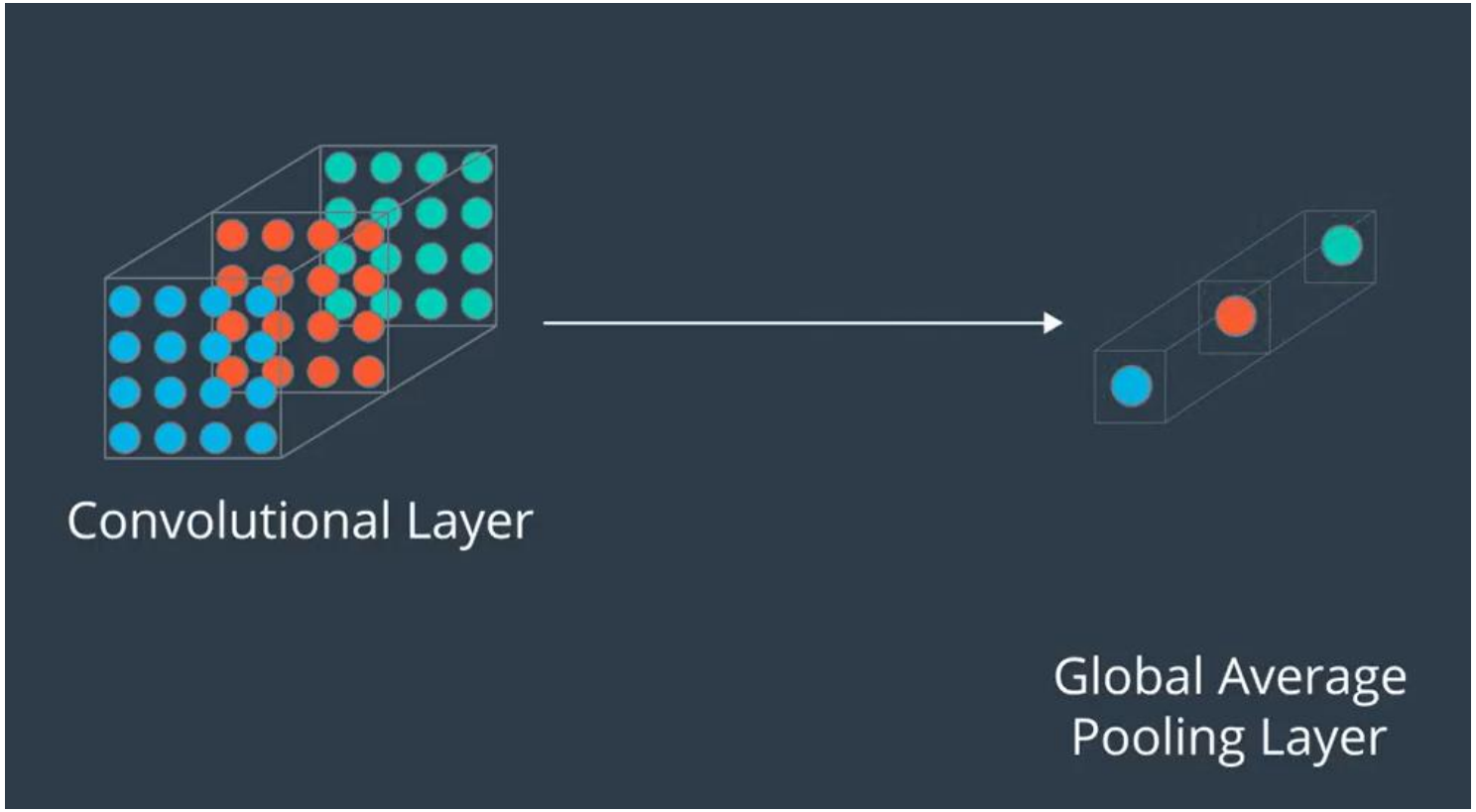
CNNs

Pooling layers



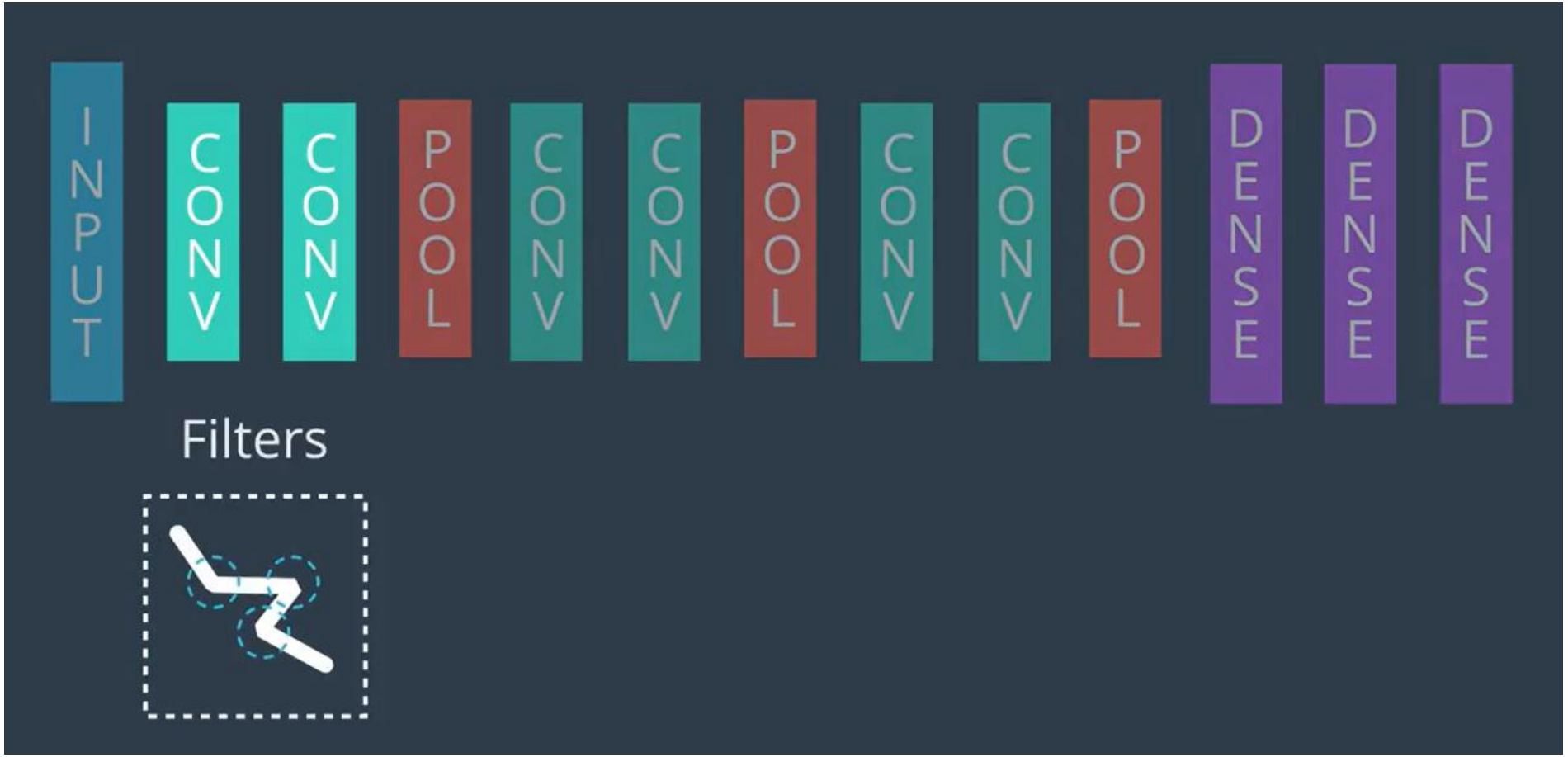
CNNs

Pooling layers



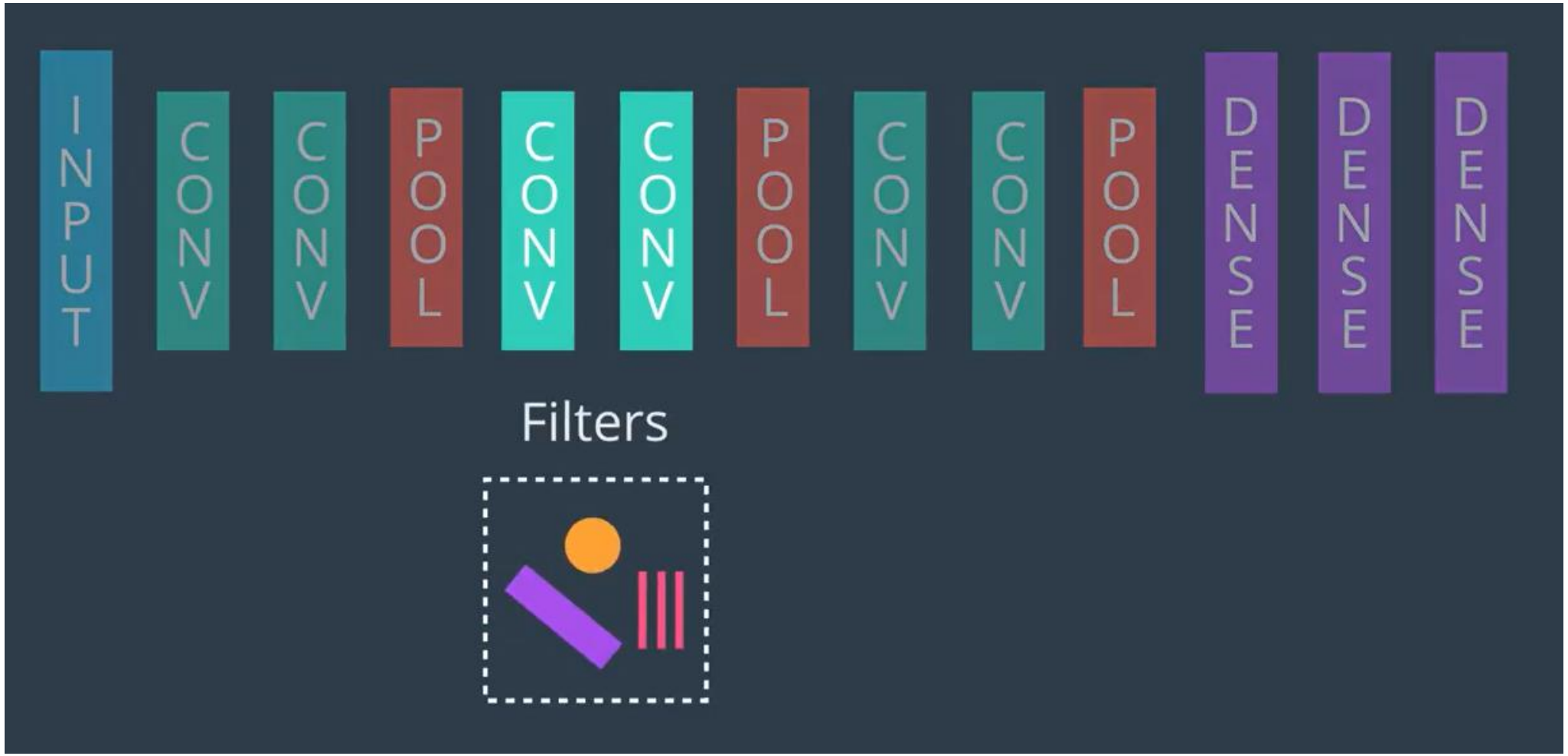
CNNs

Transfer learning



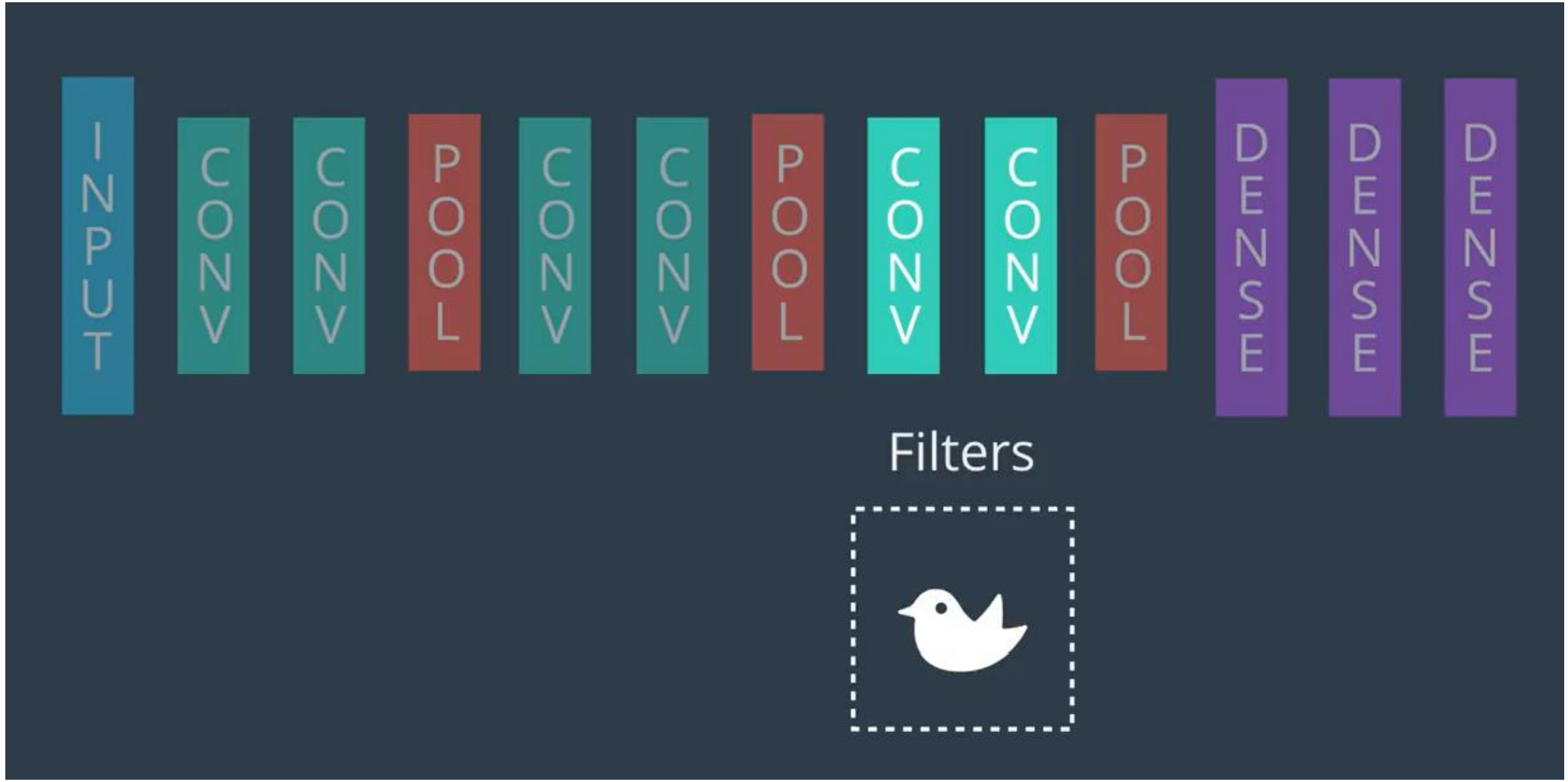
CNNs

Transfer learning



CNNs

Transfer learning



CNNs

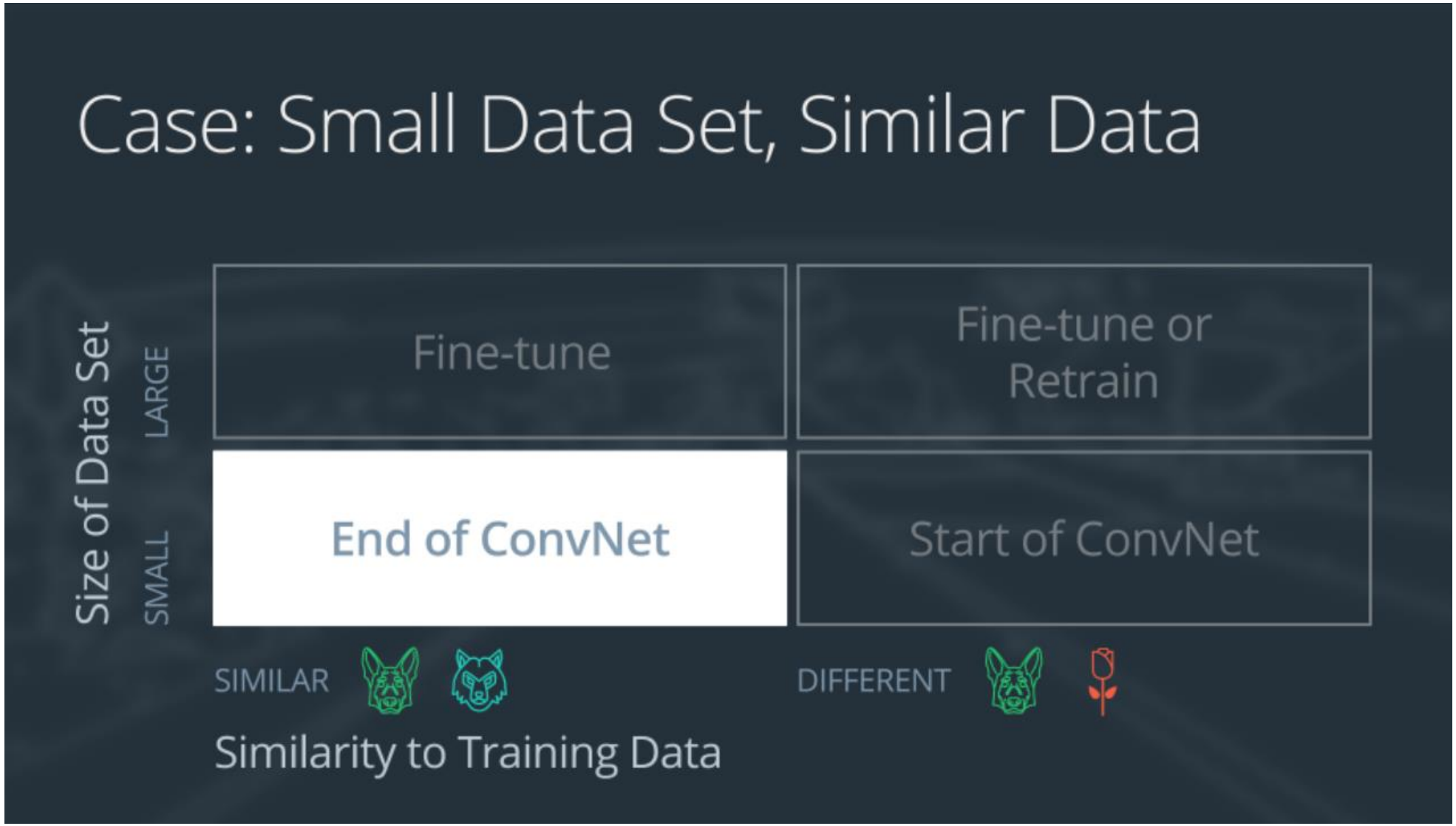
Transfer learning

CNNs

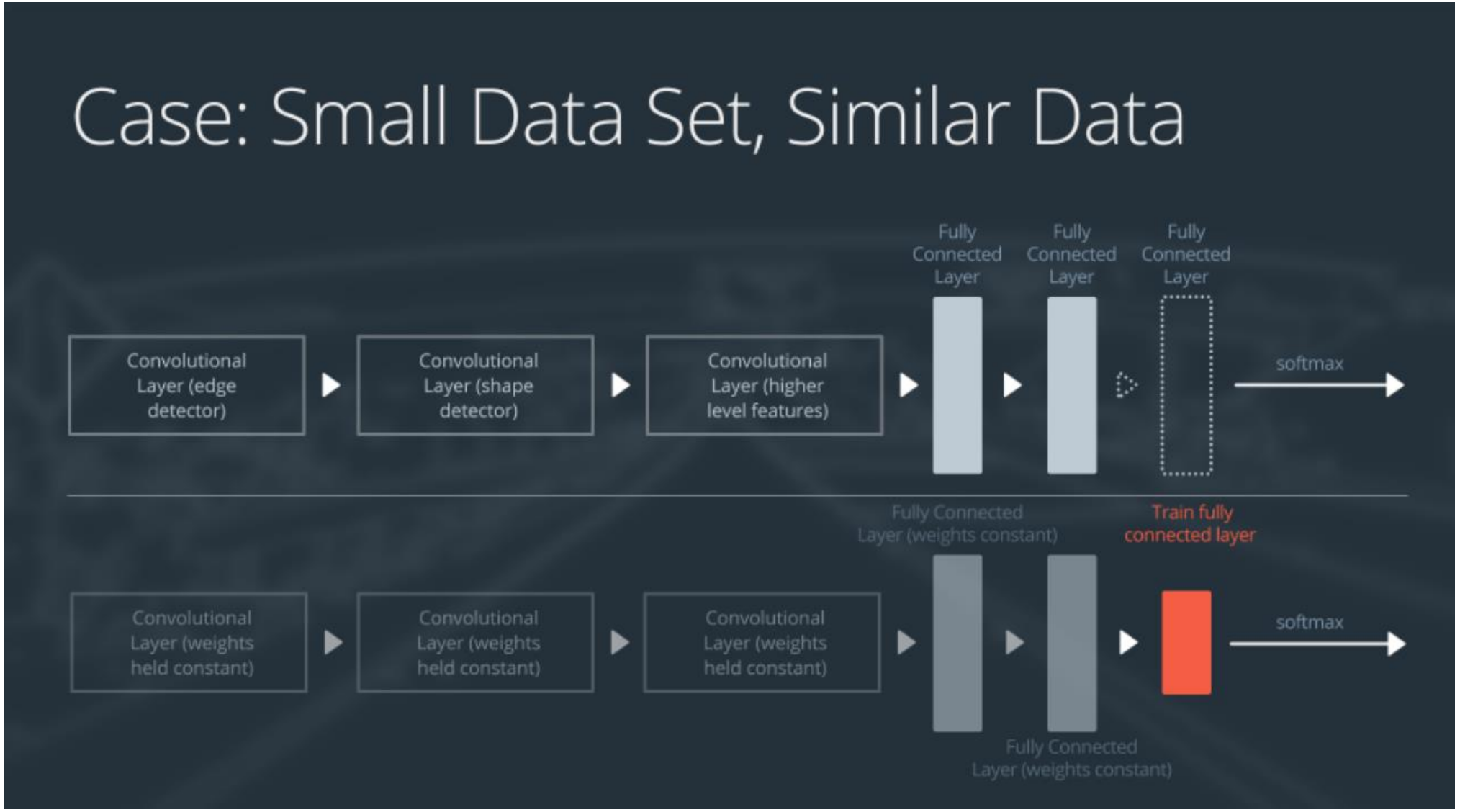
Guide to use Transfer learning

1. New data set is small, new data is similar to original training data
2. New data set is small, new data is different from original training data
3. New data set is large, new data is similar to original training data
4. New data set is large, new data is different from original training data

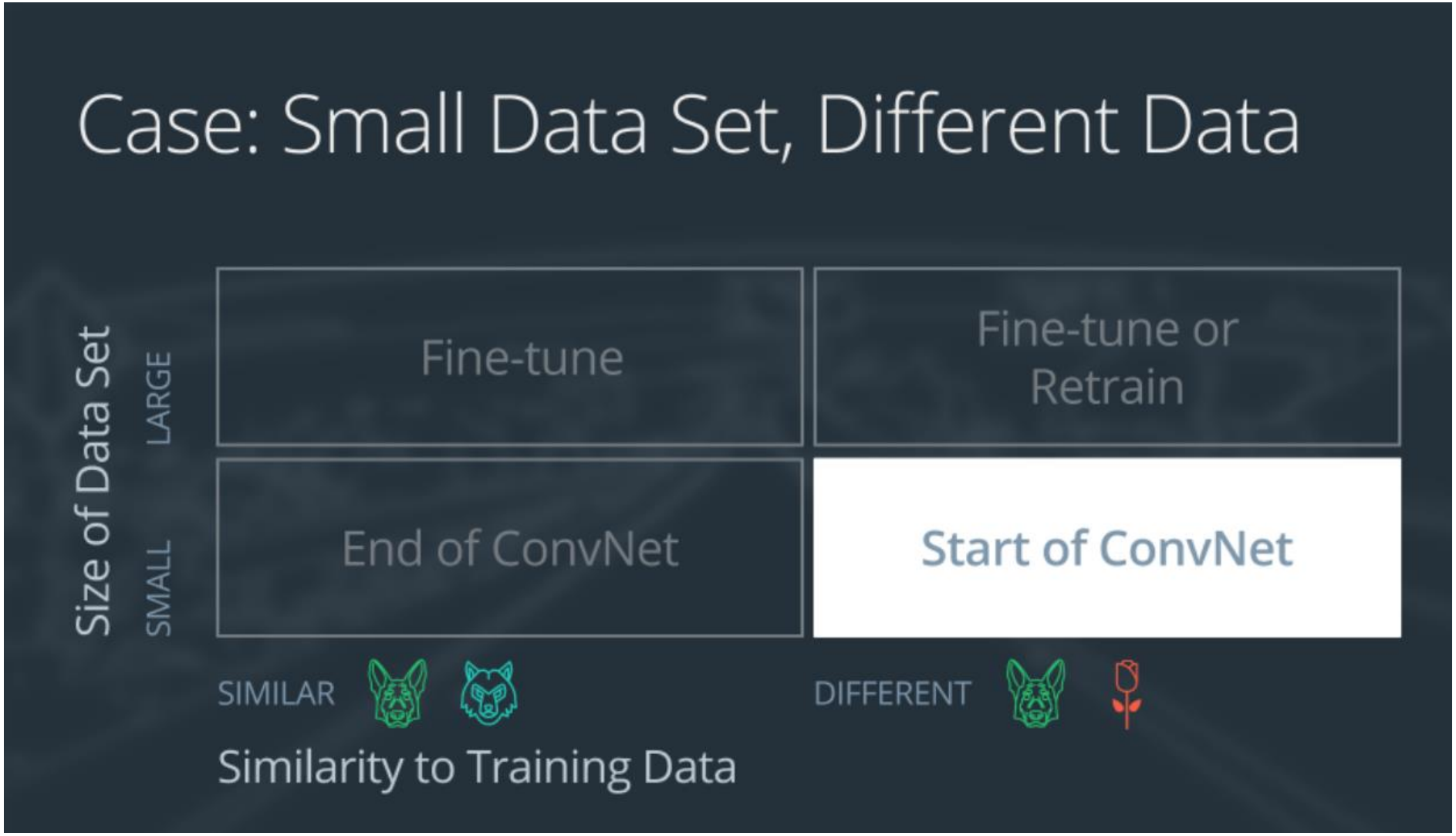
CNNs


Case: Small Data Set, Similar Data
Size of Data Set
LARGE
SMALL
Fine-tune
Fine-tune or Retrain
End of ConvNet
Start of ConvNet
SIMILAR
DIFFERENT
Similarity to Training Data

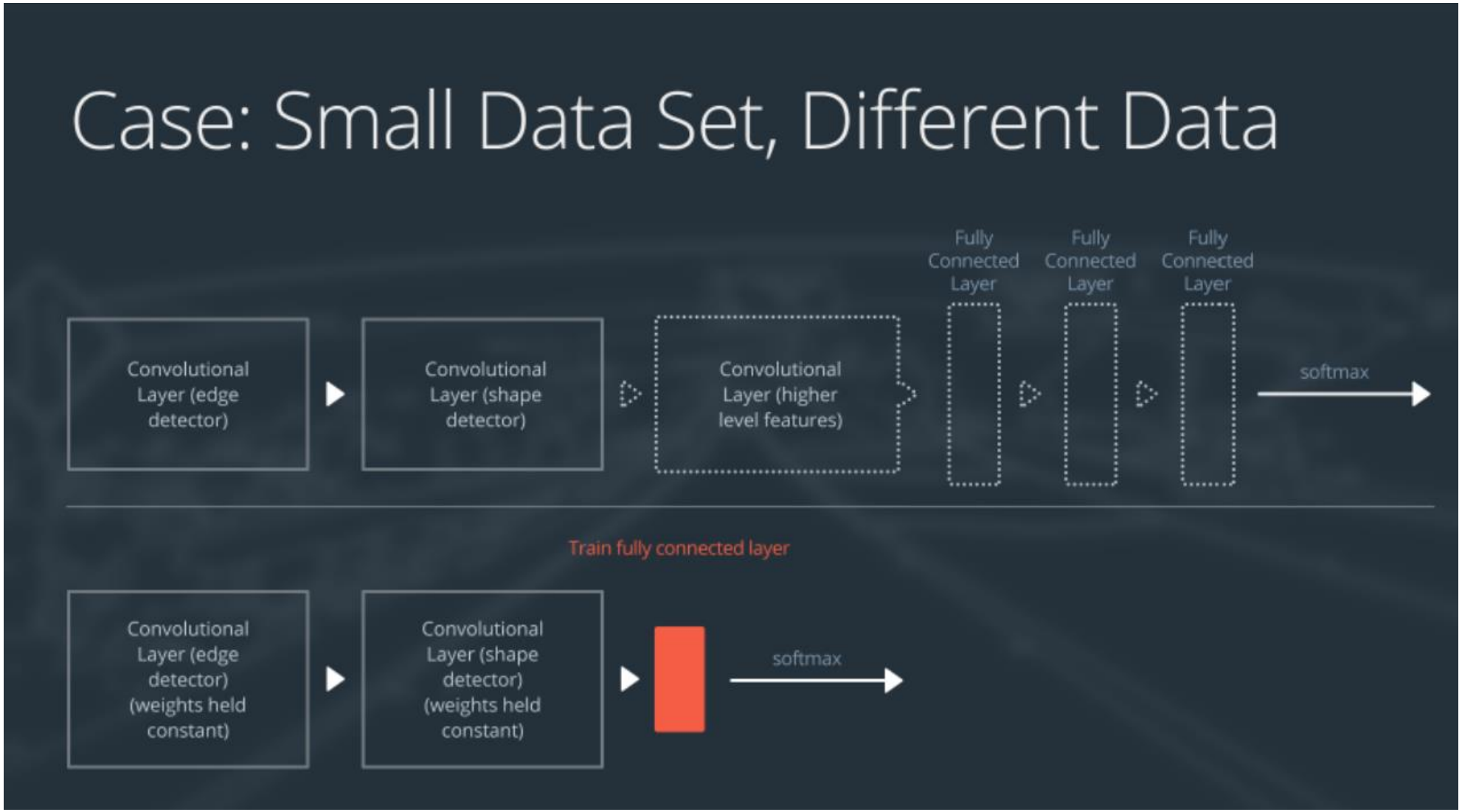
CNNs


Case: Small Data Set, Similar Data
Convolutional Layer (edge detector)
Convolutional Layer (shape detector)
Convolutional Layer (higher level features)
Fully Connected Layer
Fully Connected Layer
Fully Connected Layer
softmax
Convolutional Layer (weights held constant)
Convolutional Layer (weights held constant)
Convolutional Layer (weights held constant)
Fully Connected Layer (weights constant)
Train fully connected layer
Fully Connected Layer (weights constant)
softmax

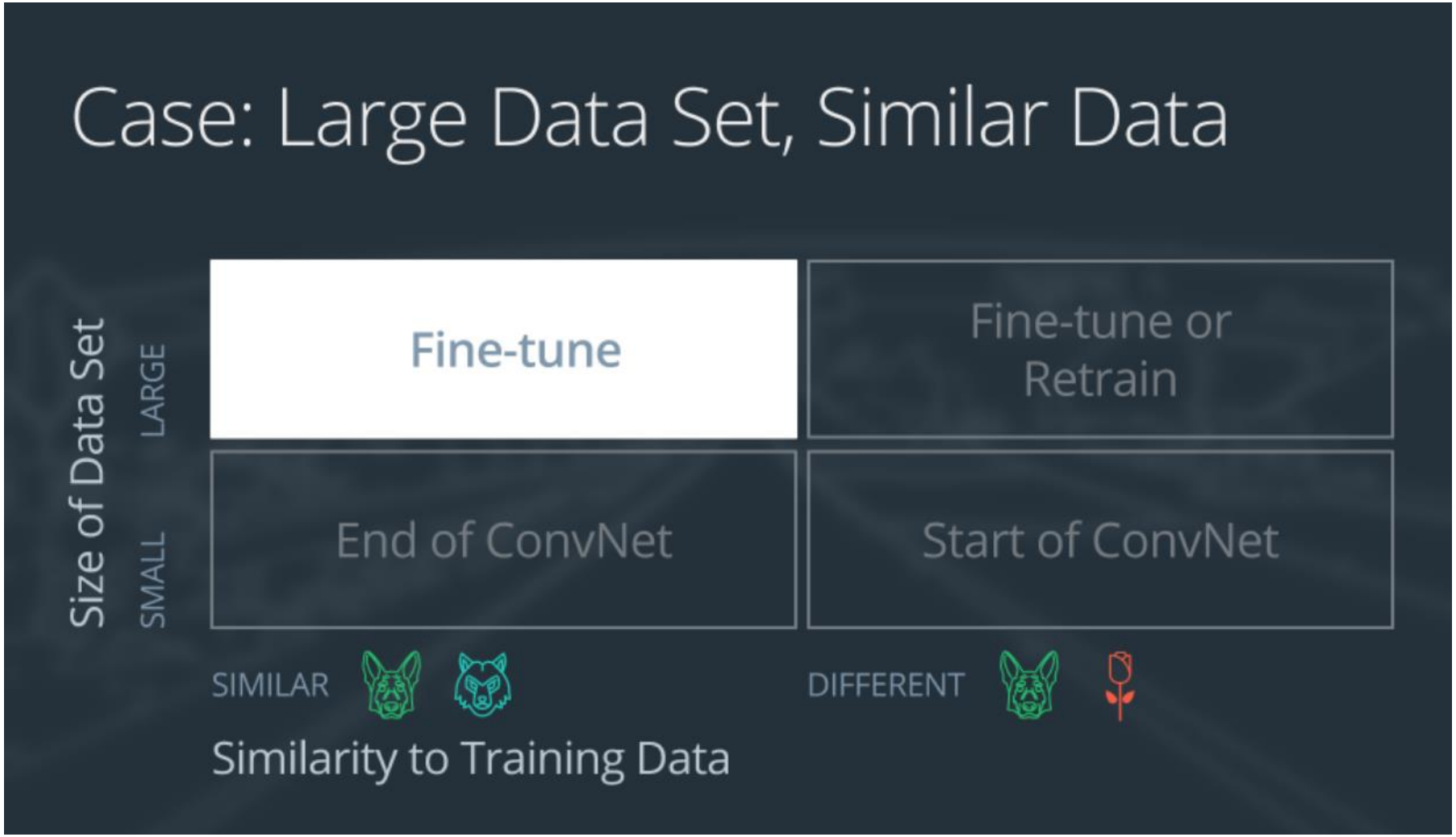
CNNs


Case: Small Data Set, Different Data
Size of Data Set
LARGE
SMALL
Fine-tune
Fine-tune or Retrain
End of ConvNet
Start of ConvNet
SIMILAR
DIFFERENT
Similarity to Training Data

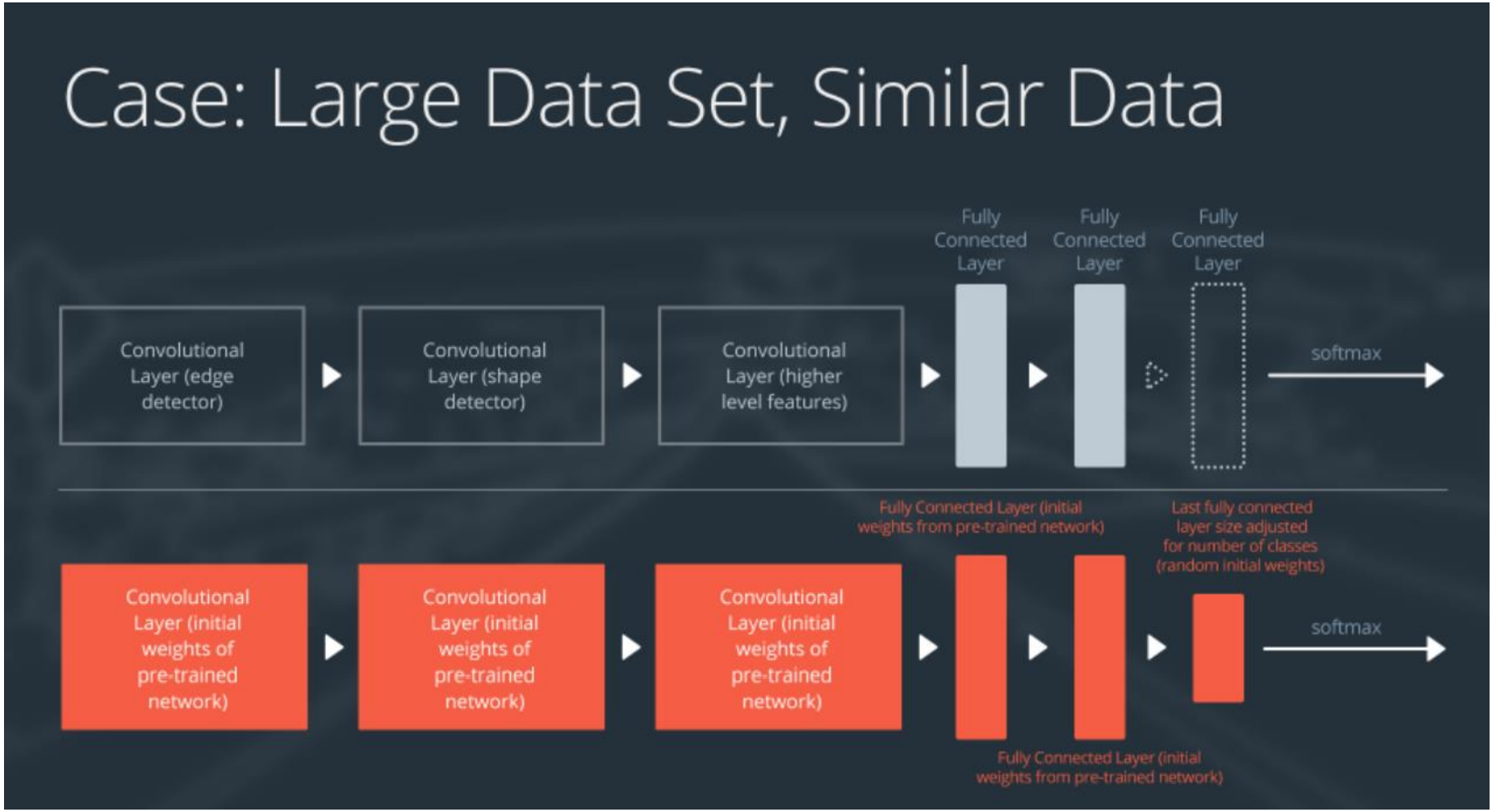
CNNs


Case: Small Data Set, Different Data
Convolutional Layer (edge detector)
Convolutional Layer (shape detector)
Convolutional Layer (higher level features)
Fully Connected Layer
Fully Connected Layer
Fully Connected Layer
softmax
Train fully connected layer
Convolutional Layer (edge detector) (weights held constant)
Convolutional Layer (shape detector) (weights held constant)
softmax

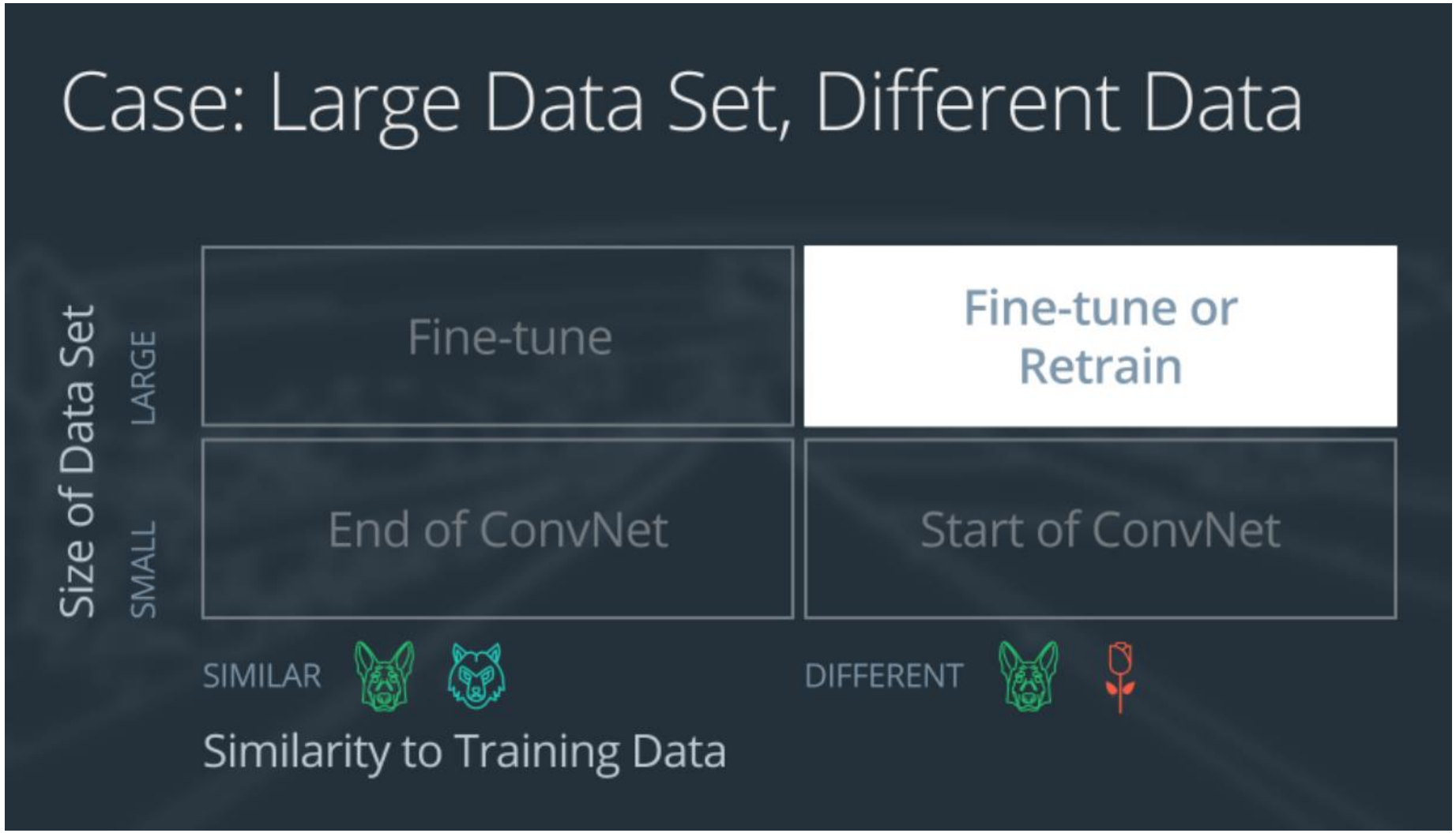
CNNs


Case: Large Data Set, Similar Data
Size of Data Set
LARGE
SMALL
Fine-tune
Fine-tune or Retrain
End of ConvNet
Start of ConvNet
SIMILAR
DIFFERENT
Similarity to Training Data

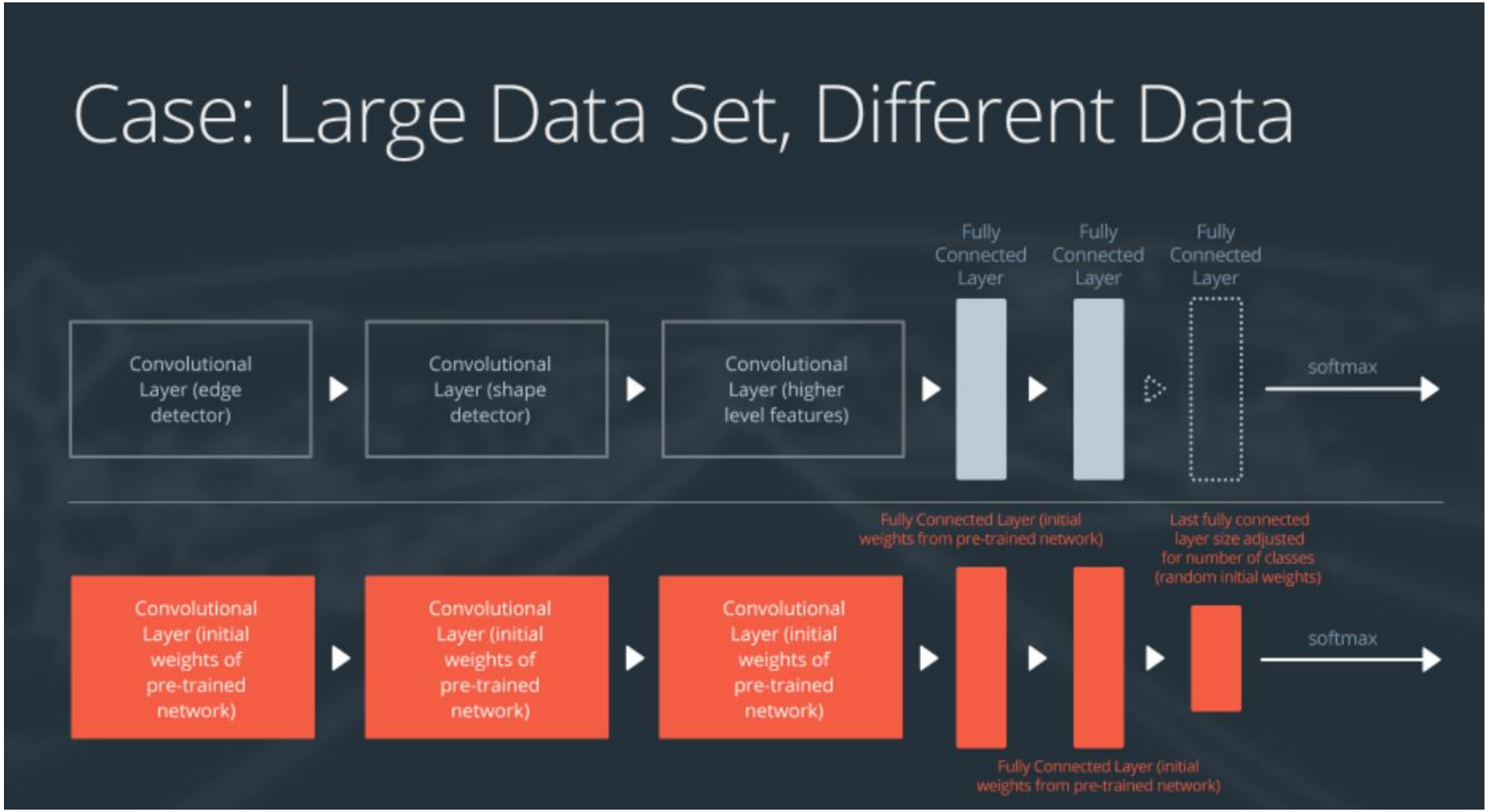
CNNs


Case: Large Data Set, Similar Data
Convolutional Layer (edge detector)
Convolutional Layer (shape detector)
Convolutional Layer (higher level features)
Fully Connected Layer
Fully Connected Layer
Fully Connected Layer
softmax
Convolutional Layer (initial weights of pre-trained network)
Convolutional Layer (initial weights of pre-trained network)
Convolutional Layer (initial weights of pre-trained network)
Fully Connected Layer (initial weights from pre-trained network)
Fully Connected Layer (initial weights from pre-trained network)
Last fully connected layer size adjusted for number of classes (random initial weights)
softmax

CNNs


Case: Large Data Set, Different Data
Size of Data Set
LARGE
SMALL
Fine-tune
Fine-tune or Retrain
End of ConvNet
Start of ConvNet
SIMILAR
DIFFERENT
Similarity to Training Data

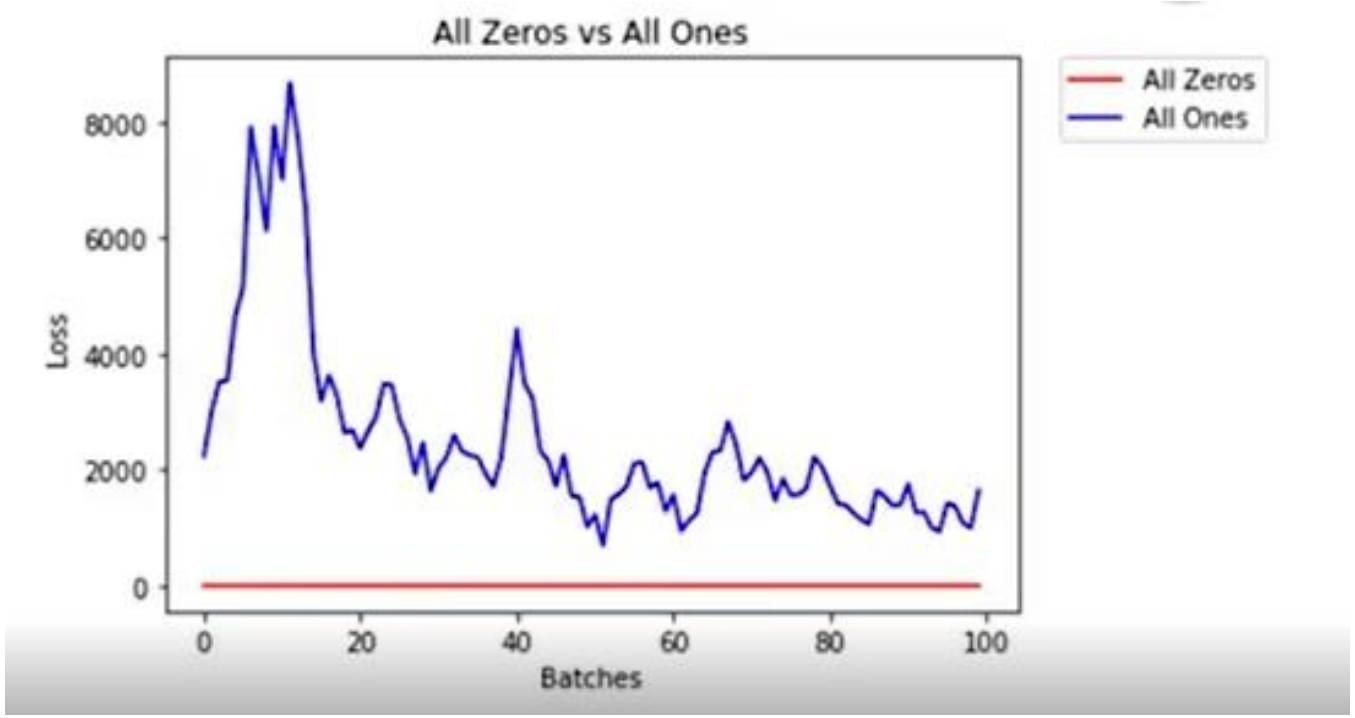
CNNs


Case: Large Data Set, Different Data
Convolutional Layer (edge detector)
Convolutional Layer (shape detector)
Convolutional Layer (higher level features)
Fully Connected Layer
Fully Connected Layer
Fully Connected Layer
softmax
Convolutional Layer (initial weights of pre-trained network)
Convolutional Layer (initial weights of pre-trained network)
Convolutional Layer (initial weights of pre-trained network)
Fully Connected Layer (initial weights from pre-trained network)
Fully Connected Layer (initial weights from pre-trained network)
Last fully connected layer size adjusted for number of classes (random initial weights)
softmax

CNNs

Weight initialization

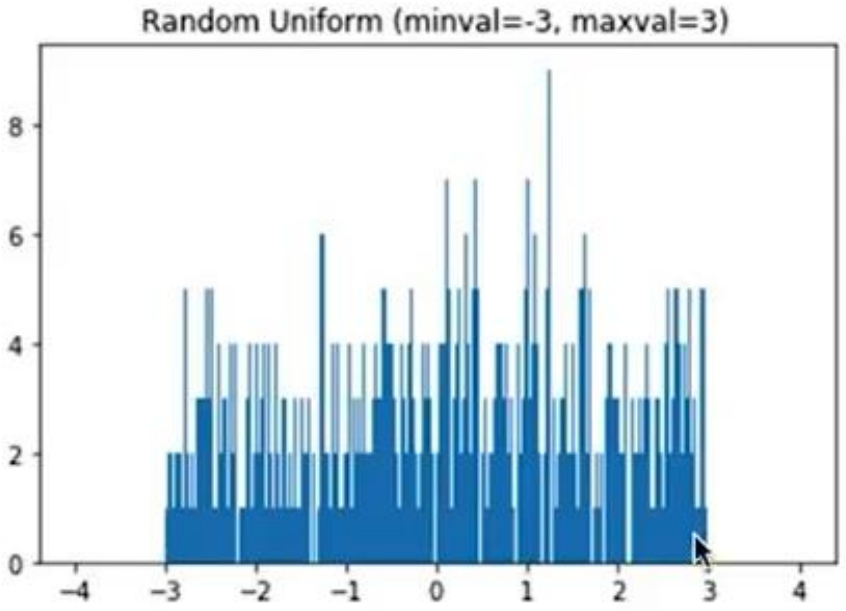
- All ones vs all zeros

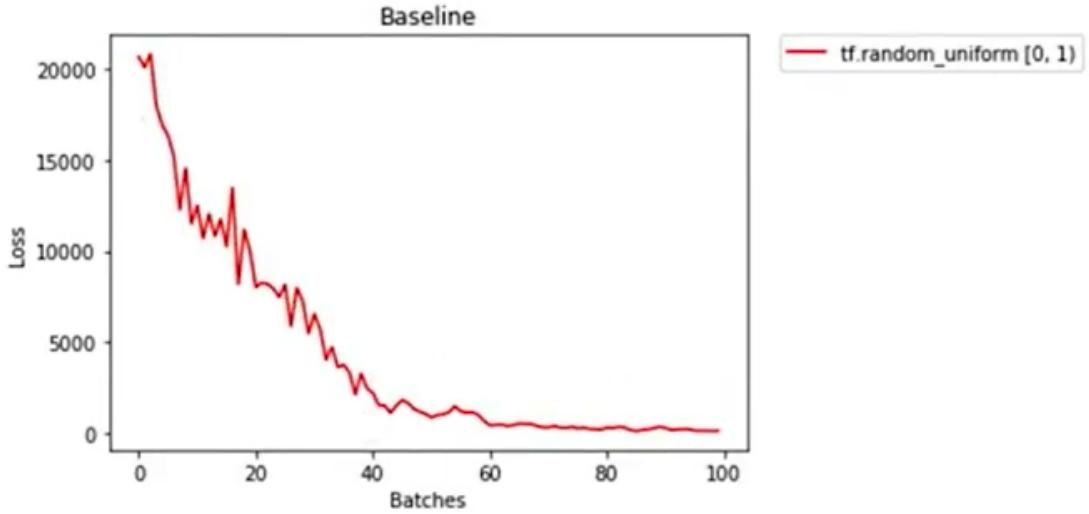


CNNs

Weight initialization

- Random distribution (Uniform)

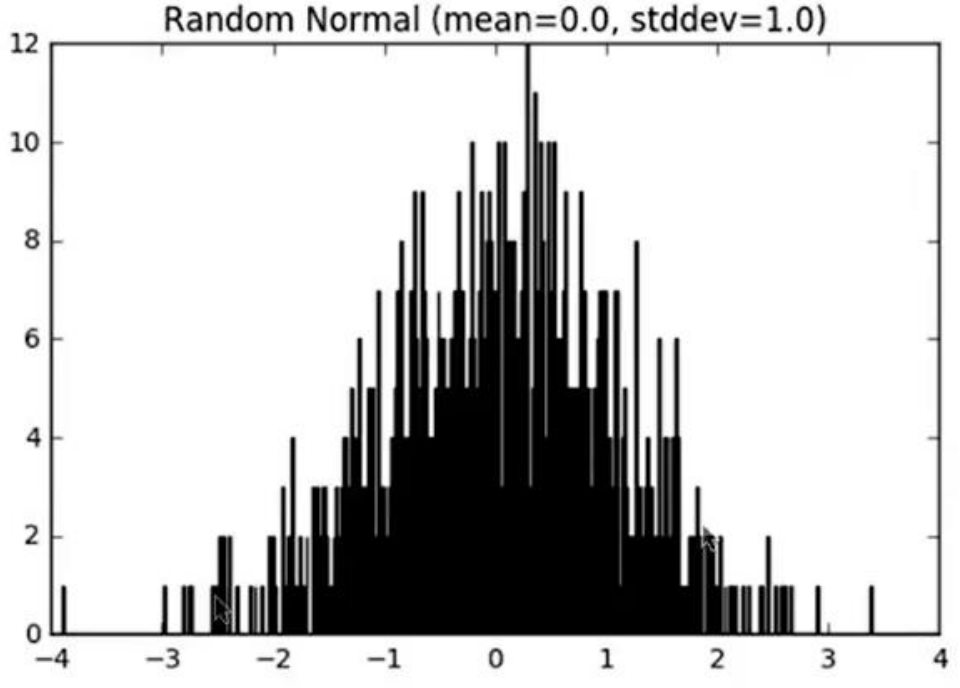


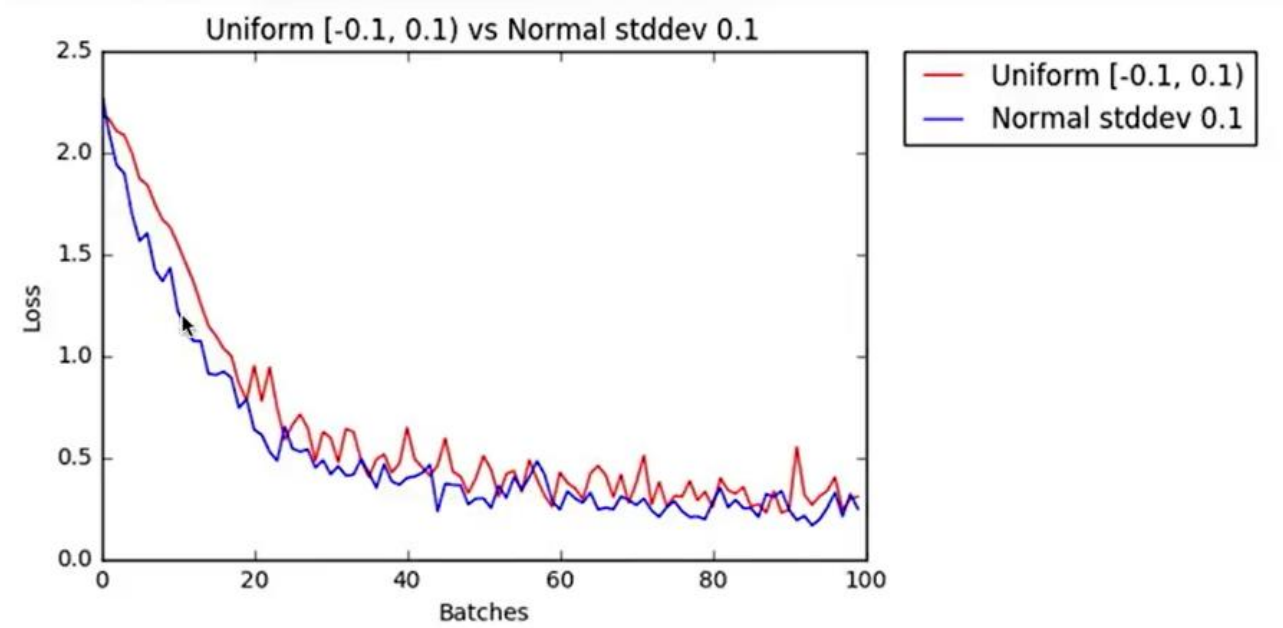


CNNs

Weight initialization

- Random distribution (normal distribution)



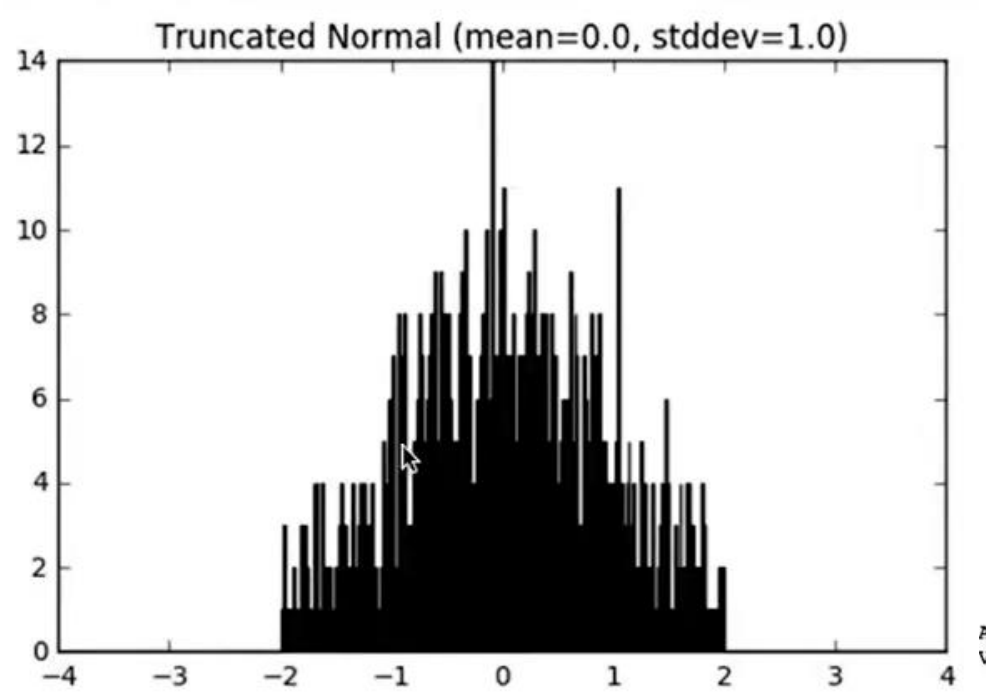


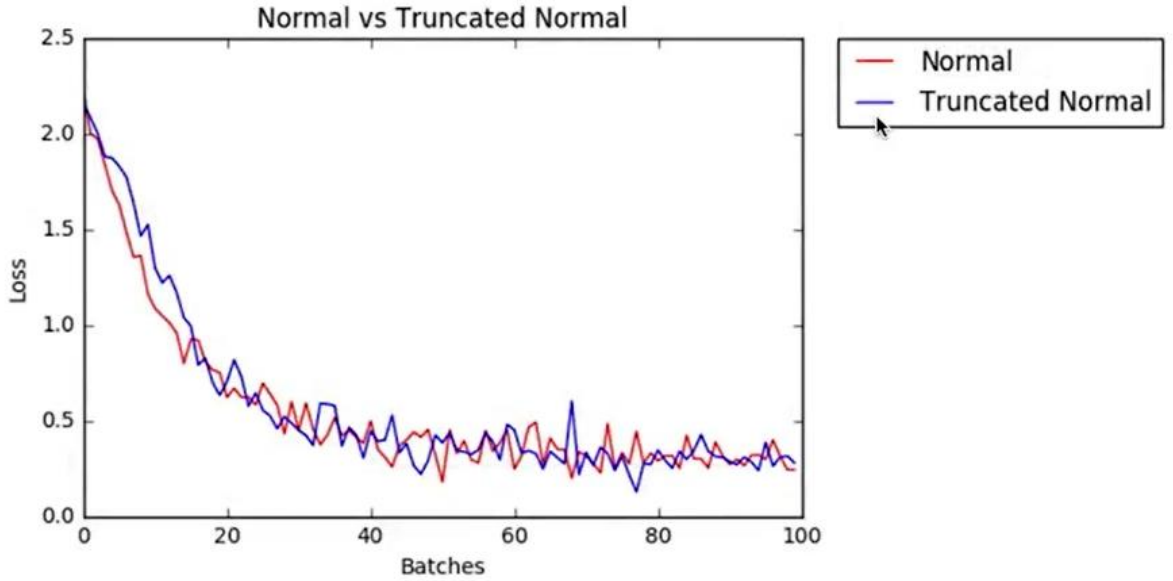

```
After 858 Batches (2 Epochs):
Validation Accuracy
   96.920% -- Uniform [-0.1, 0.1)
   97.200% -- Normal stddev 0.1
```

CNNs

Weight initialization

- Random distribution (normal distribution truncated)

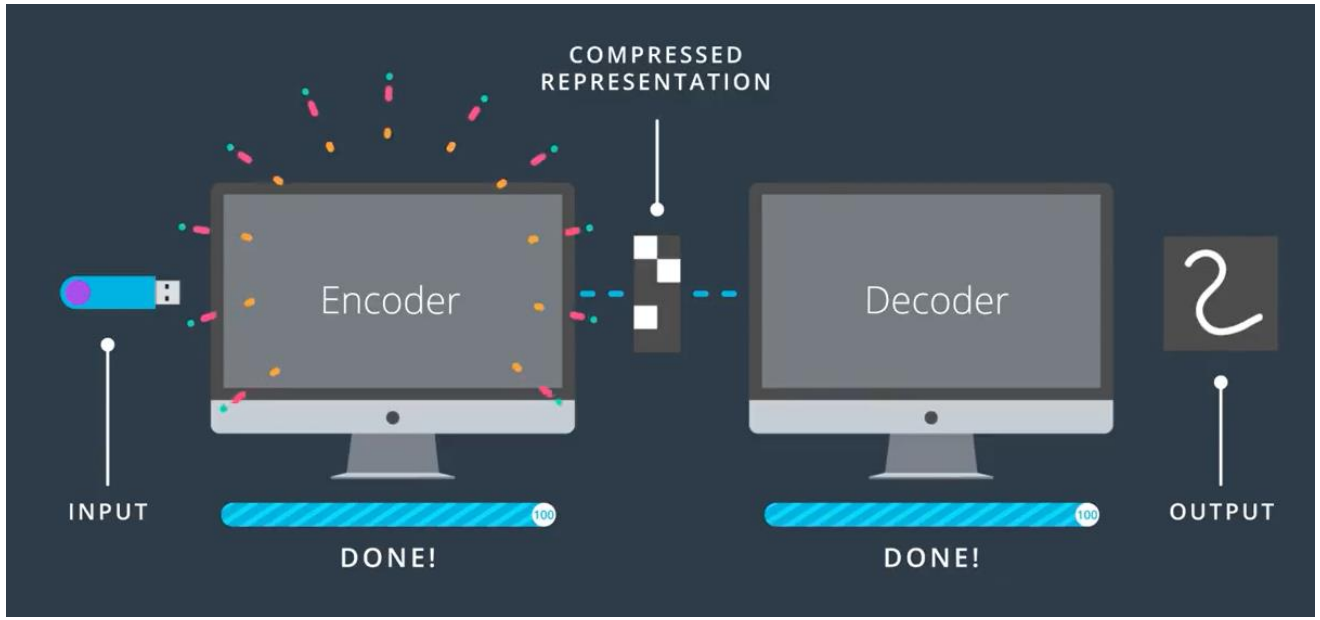


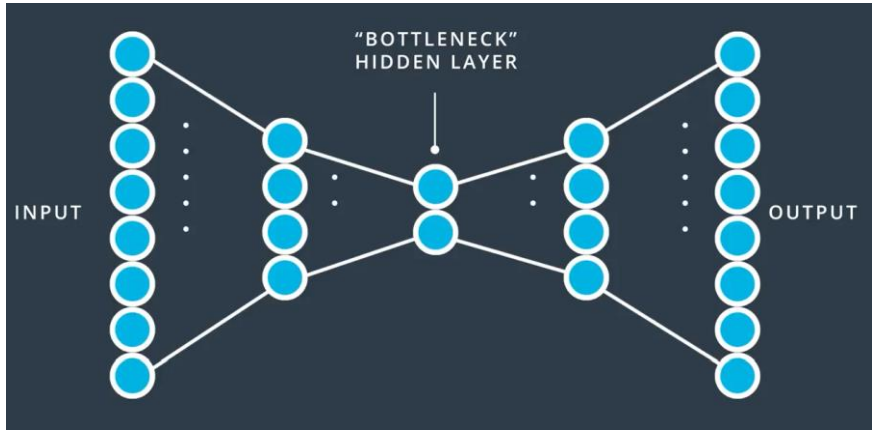



```
After 858 Batches (2 Epochs):
Validation Accuracy
   97.020% -- Normal
   97.480% -- Truncated Normal
```

CNNs

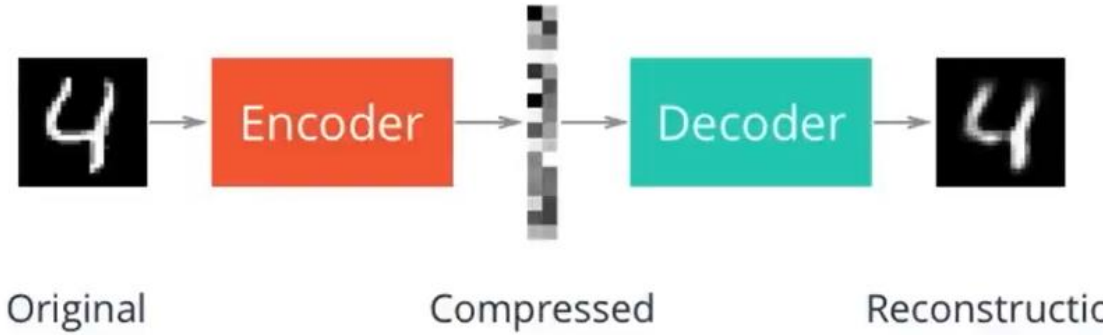
Autoencoders

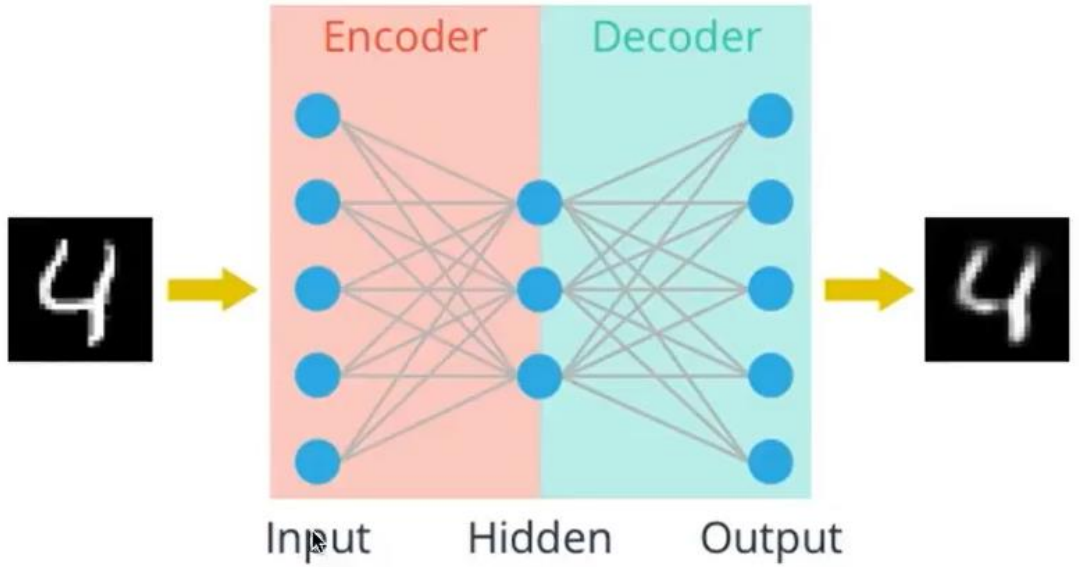




CNNs

Autoencoders





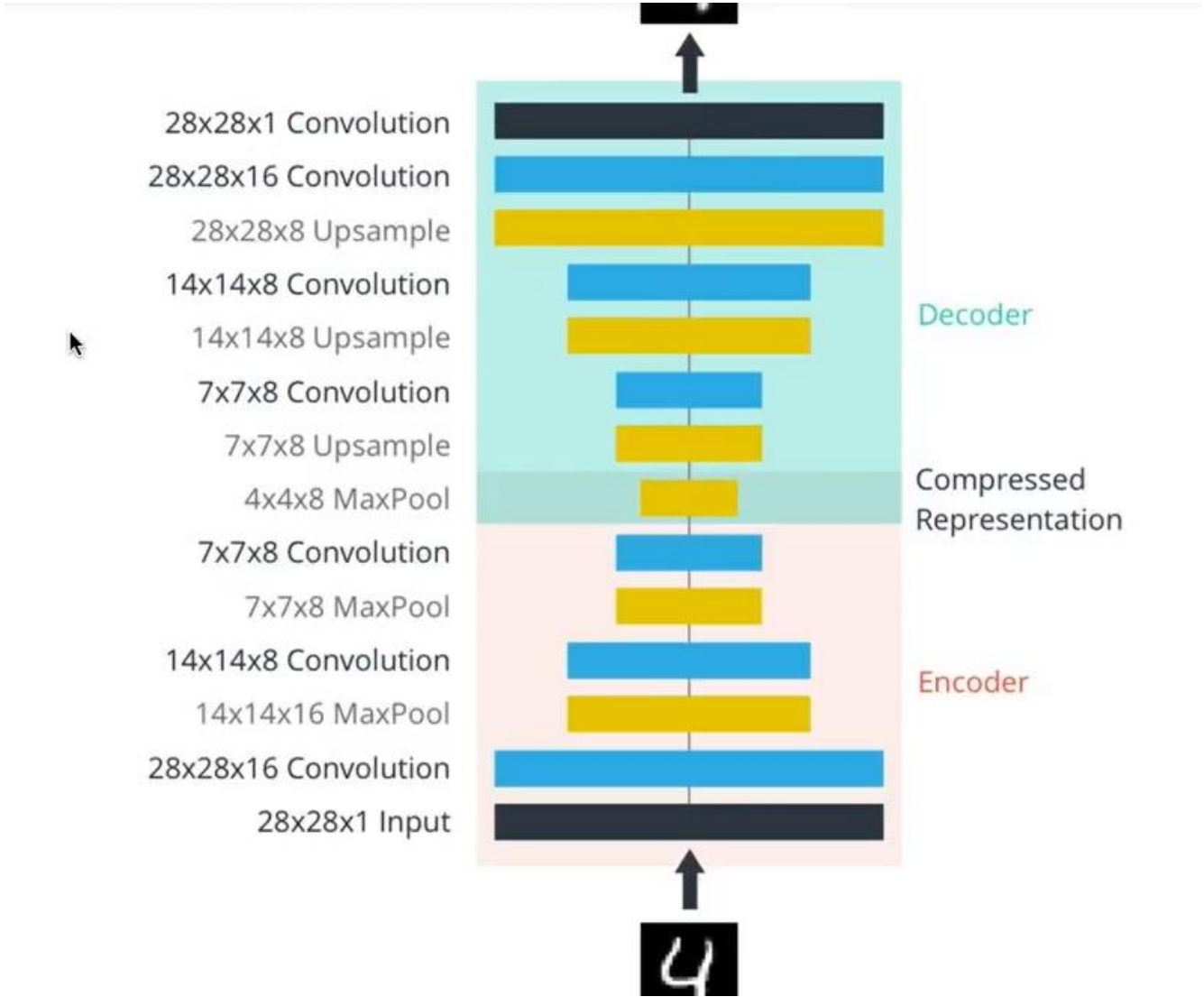
CNNs

Autoencoders are used in deep learning for several reasons:

1.Unsupervised learning: Autoencoders are a type of neural network that can learn from unlabeled data, making them well-suited for unsupervised learning tasks. This means that they can be trained on large datasets without requiring explicit labels for the data, which can be time-consuming and expensive to obtain.
2.Feature extraction: Autoencoders can be used to extract useful features from input data, such as images, text, or audio. These features can be used for downstream tasks such as classification, object detection, or segmentation.
3.Data compression: Autoencoders can be used for data compression, where the network learns a compressed representation of the input data that can be stored or transmitted more efficiently. This can be useful for applications where storage or bandwidth is limited.
4.Denoising: Autoencoders can be used for image denoising, where the network is trained on noisy images and learns to remove the noise while preserving the underlying structure of the image.
5.Generative modeling: Autoencoders can be used for generative modeling, where the network learns to generate new data samples that are similar to the training data. This can be useful for applications such as image synthesis, where the network can be used to generate realistic images of objects or scenes that do not exist in the real world.

CNNs

Convolutional Autoencoders



CNNs

END

Images from udacity.com