Deep Convolutional Neural Networks

Elizabeth Heon

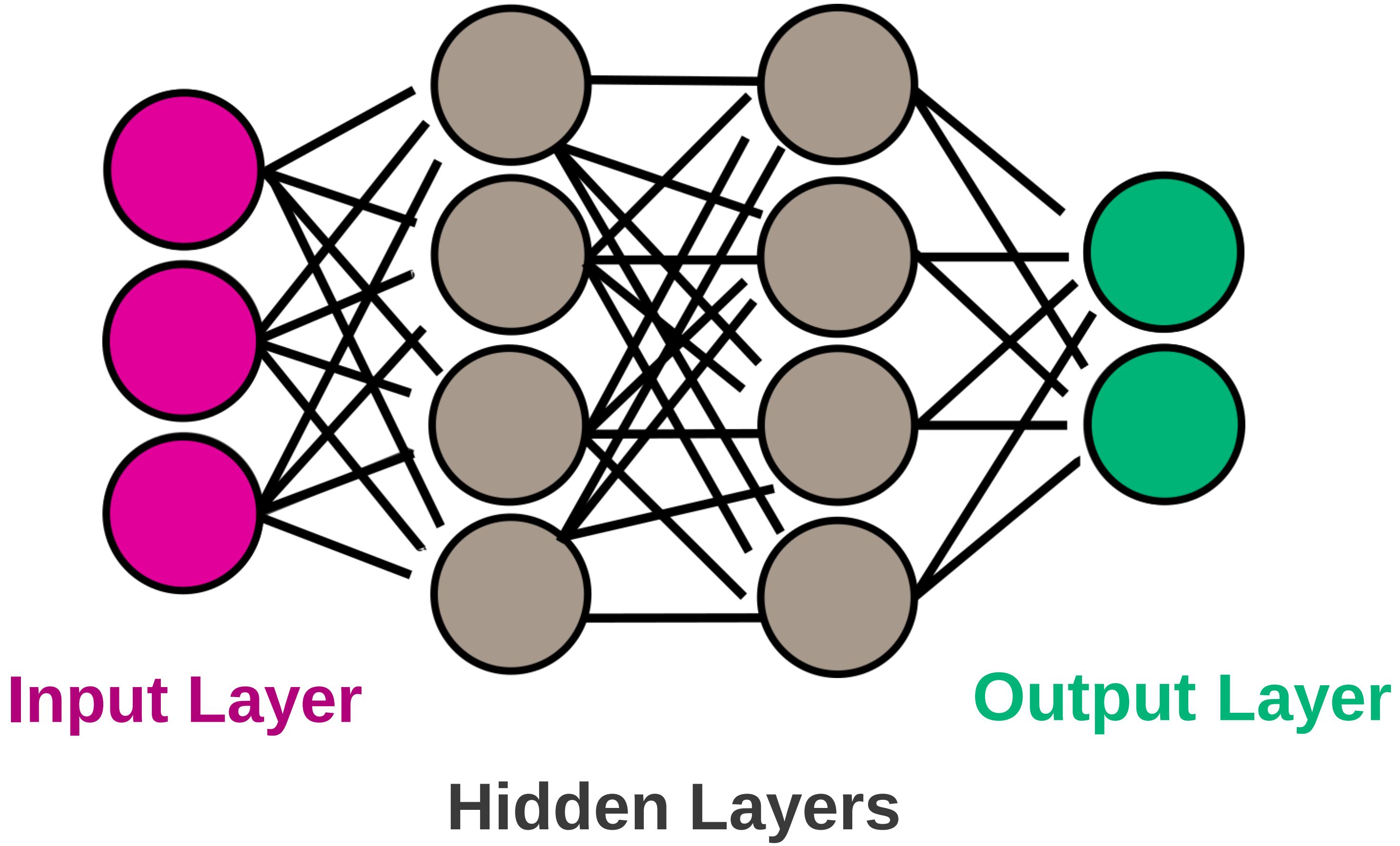
5/22/2025

RECAP

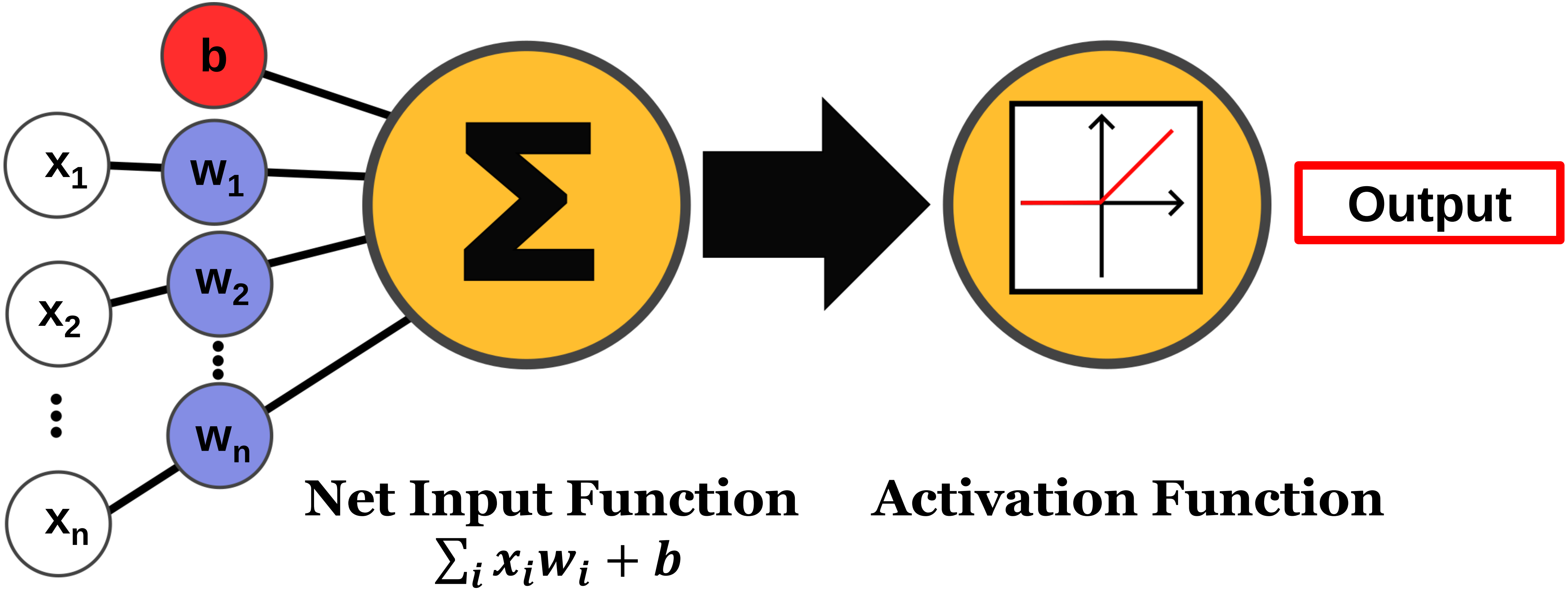
Basic Neural Networks

What is a Neural Network

A neural network is a machine learning model formed of interconnected layers of nodes, called neurons, which are designed to mimic the function of biological neurons



How Does an Artificial Neuron Work?

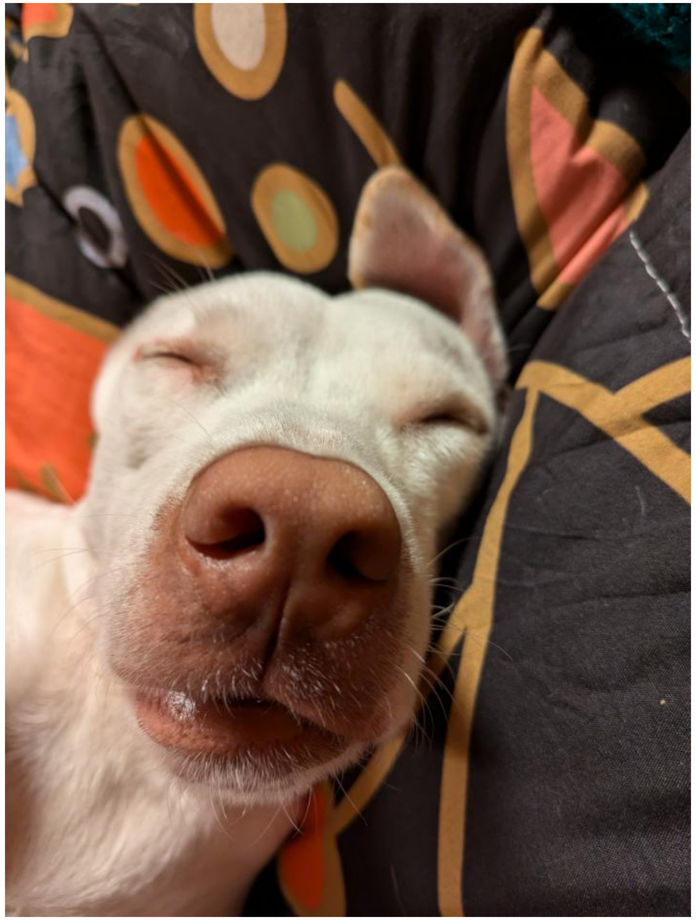


A neuron takes in a vector of inputs [x1 … xn] and forms the net input by multiplying with a vector of weights. The activation function is then applied to the net input, giving the function output

DOG-CAT

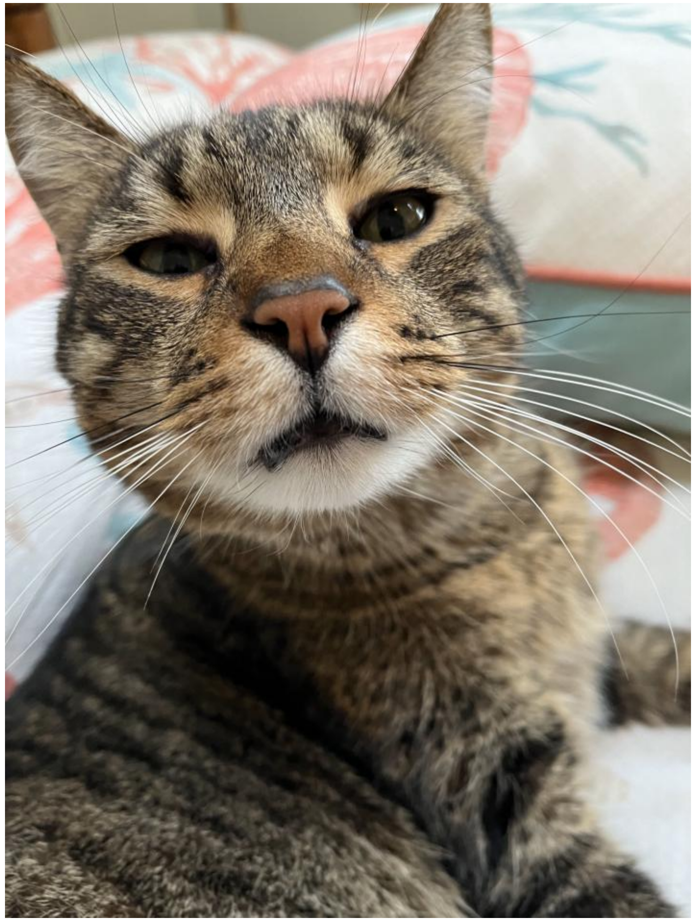
Making an Image Classifier

Image Classifier



DOG

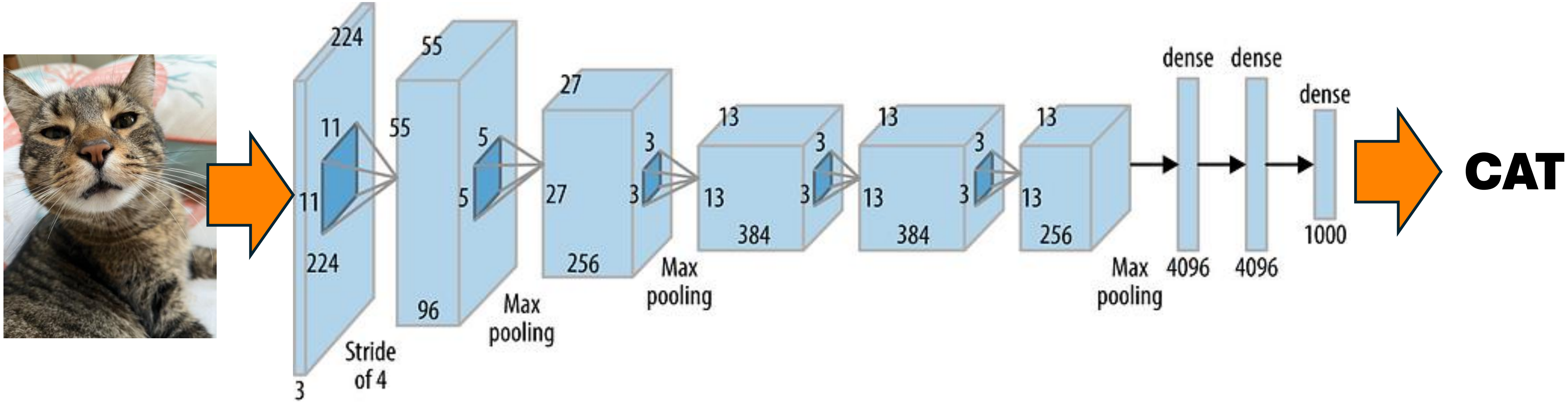
VS.



CAT

Convolutional Network

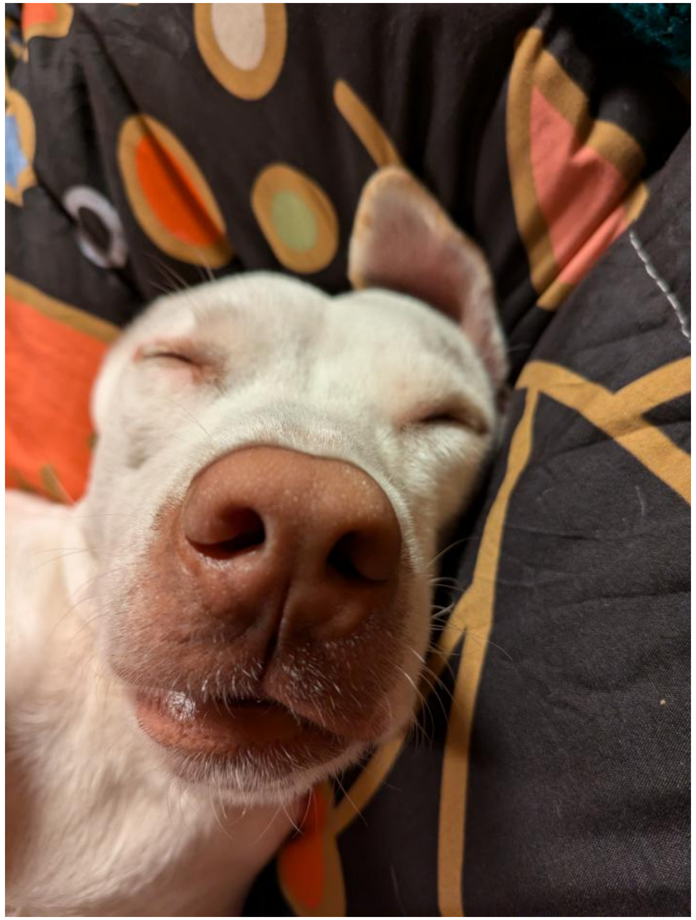
Structure of AlexNet, a DCNN for image classification



https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96

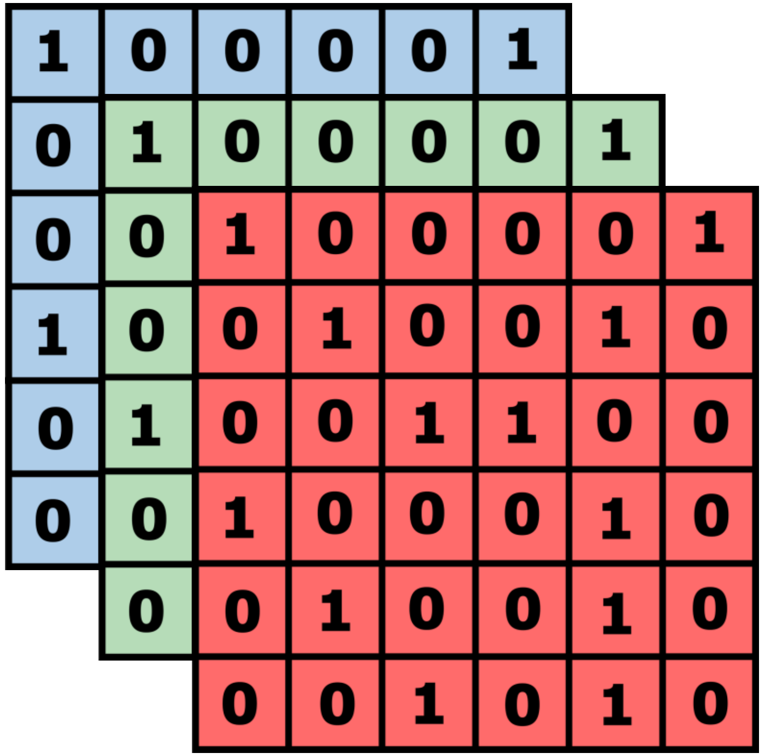
IMAGES AS DATA

Representing an Image Numerically



3742 * 4624 Pixels

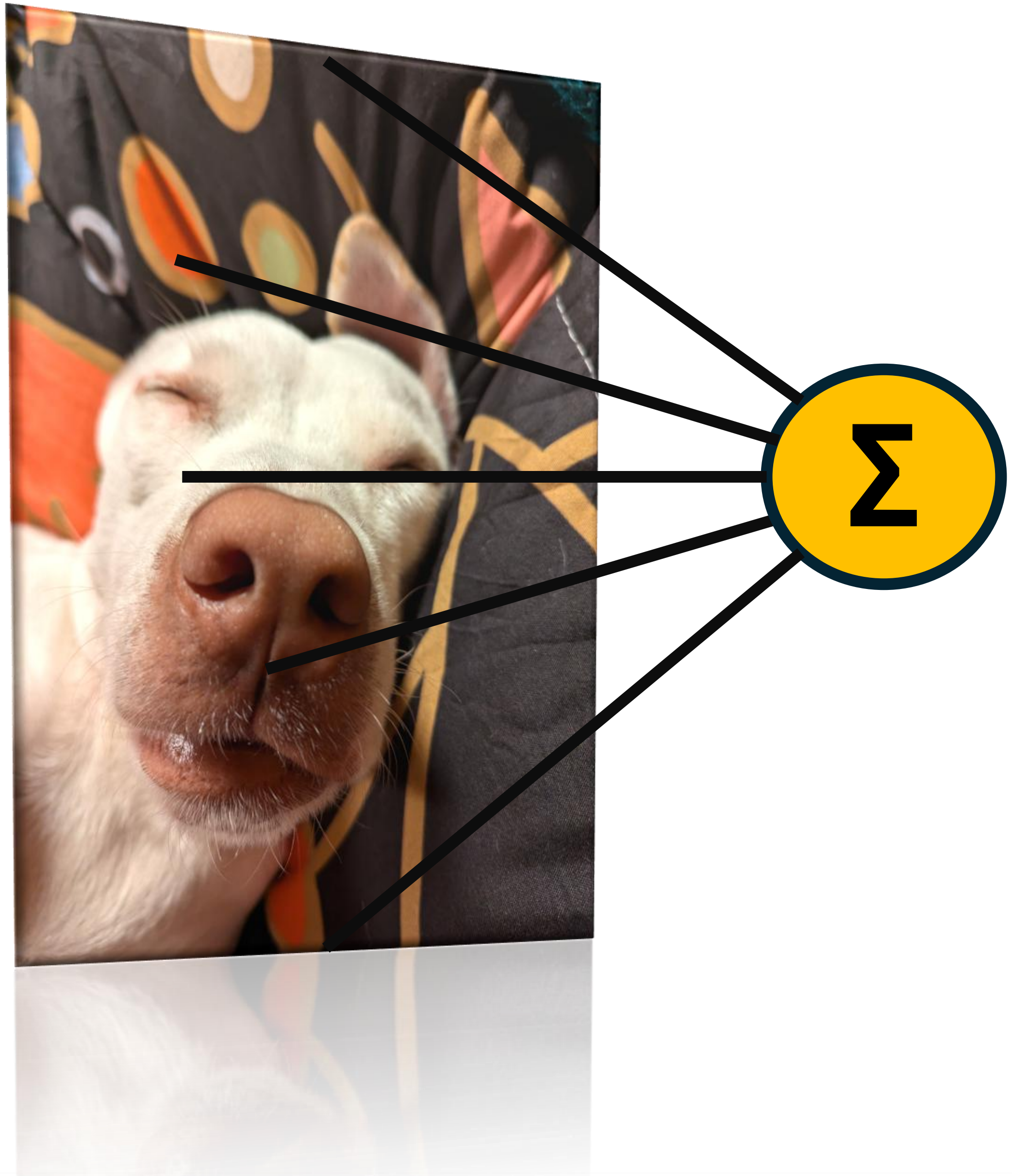
- An image is represented numerically as an array of values. RGB images contain 3 channels for red, green and blue



$$3743 * 4624 * 3 = 51{,}909{,}092$$

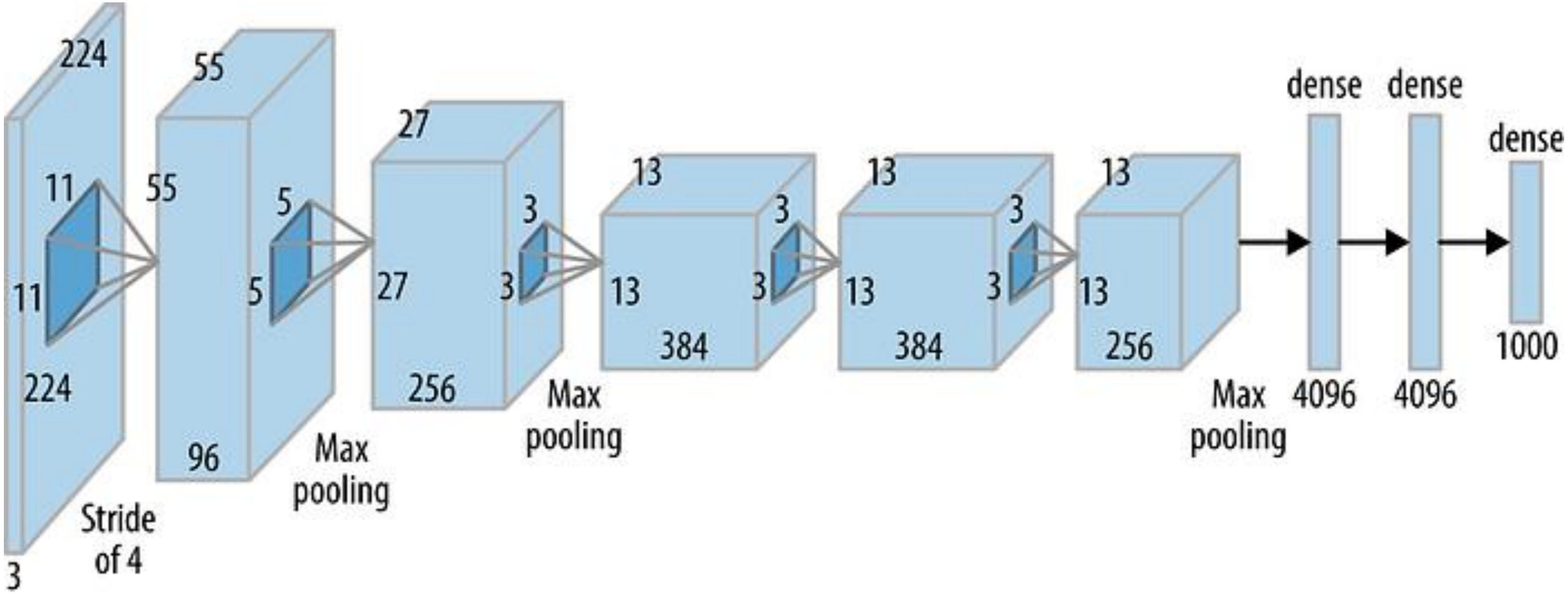
This picture is really big!

Image as Input to Neural Network



- If a fully connected network is used to process this image and every neuron has a weight for every pixel value, then every neuron needs:
  - 51,909,024 weights for RGB
  - 17,303,008 weights for single channel grayscale
- Even for more reasonably sized input images (say 128 by 128) it is highly inefficient to process data this large with a standard fully connected network. One of the advantages of convolutional neural networks is that is greatly reduces the number of parameters that must be learned
- Down sampling (making data smaller) is also important for processing image data

DCNN Structure

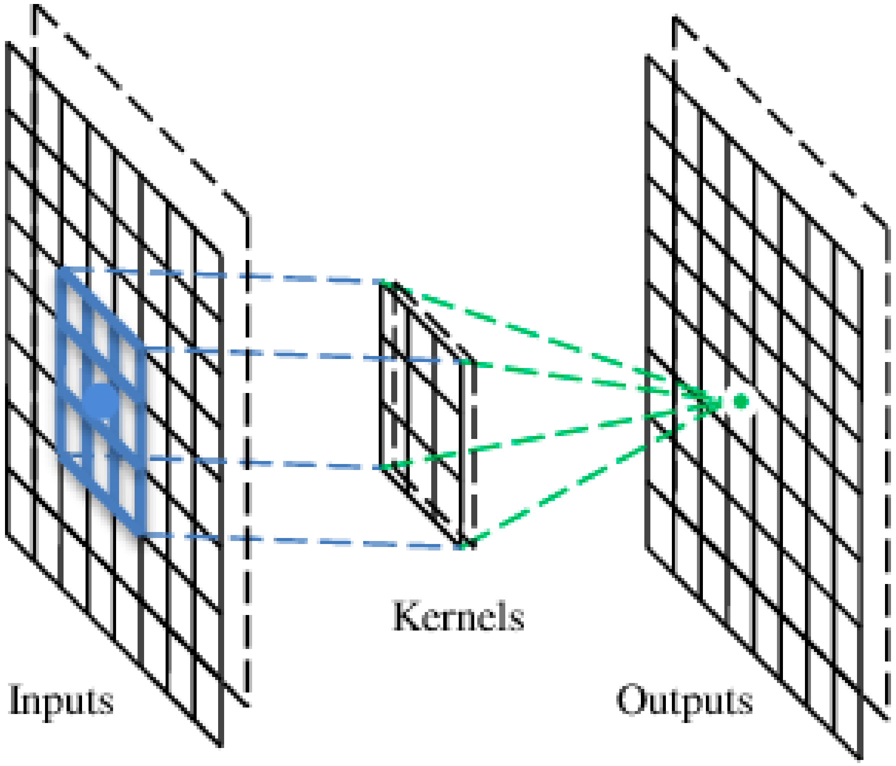


- Convolutional Neural Networks contain 3 types of layers:
  - Convolutional layers
  - Pooling layers
  - Fully Connected/Dense layers

CONVOLUTION

Convolutional Layers

- What is a convolutional layer?
  - A convolutional layer has a number of **filters (AKA Kernels)**
  - Filters detect specific patterns in input data. Filters can be used for feature extraction from images



- With kernels, each neuron only “listens” to a small portion of the input images
- This means the neurons don’t need as many weights

How do Filters Work - Verbal

- Convolution operates on two images (remember, an image is stored as a matrix of pixel values) - one input is the image, and the second acts as a filter
- Kernels, which are smaller than the total size image, are slid across the image, analyzing one region at a time
    - The image matrix is multiplied by the filter matrix to form an output, the filter matrix is then advanced across the image
    - The **stride** determines how many pixels the filter region moves by – bigger stride = less overlap
- Different kernels accomplish different tasks:
    - Sharpen
    - Blur
    - Edge detection

Convolution - Visual

The network must learn these kernels

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

$6 \times 6\ Image$

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

⋮ ⋮

Each filter detects a small pattern (3 x 3).

Convolution - Visual

Filter Advances

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

From Input Image **Kernel**

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix} = 3$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix} = -1$$

The filter is advanced over the image, at each location the dot product is taken, resulting in a scalar value. A new matrix is built from the resulting scalars

Convolution - Visual

1	-1	-1
-1	1	-1
-1	-1	1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

The filter identifies specific patterns in the input image
Repeat this process for all learned filters to make a features map

But what about color?

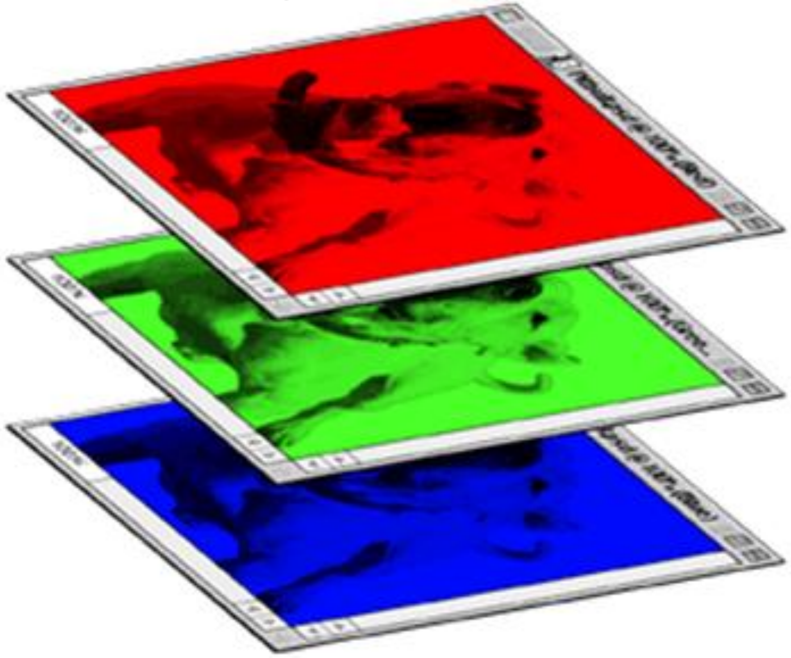
Filter 1

1	-1	-1
-1	1	-1
-1	-1	1

Filter 2

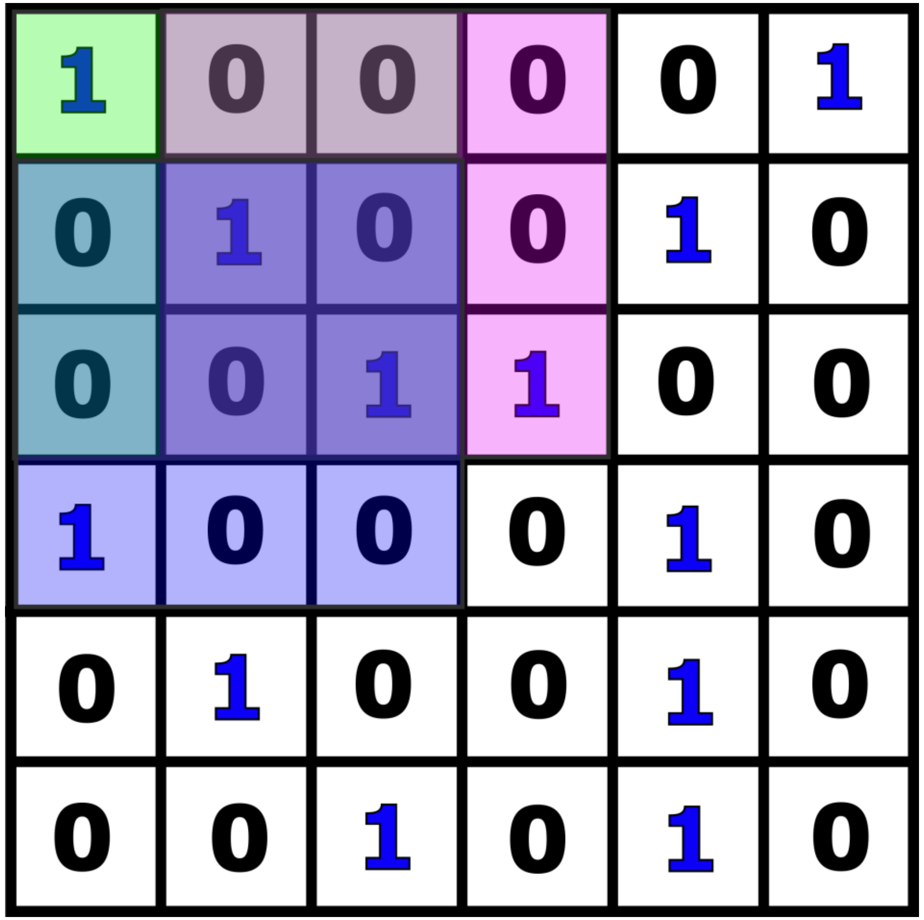
-1	1	-1
-1	1	-1
-1	1	-1

Color image

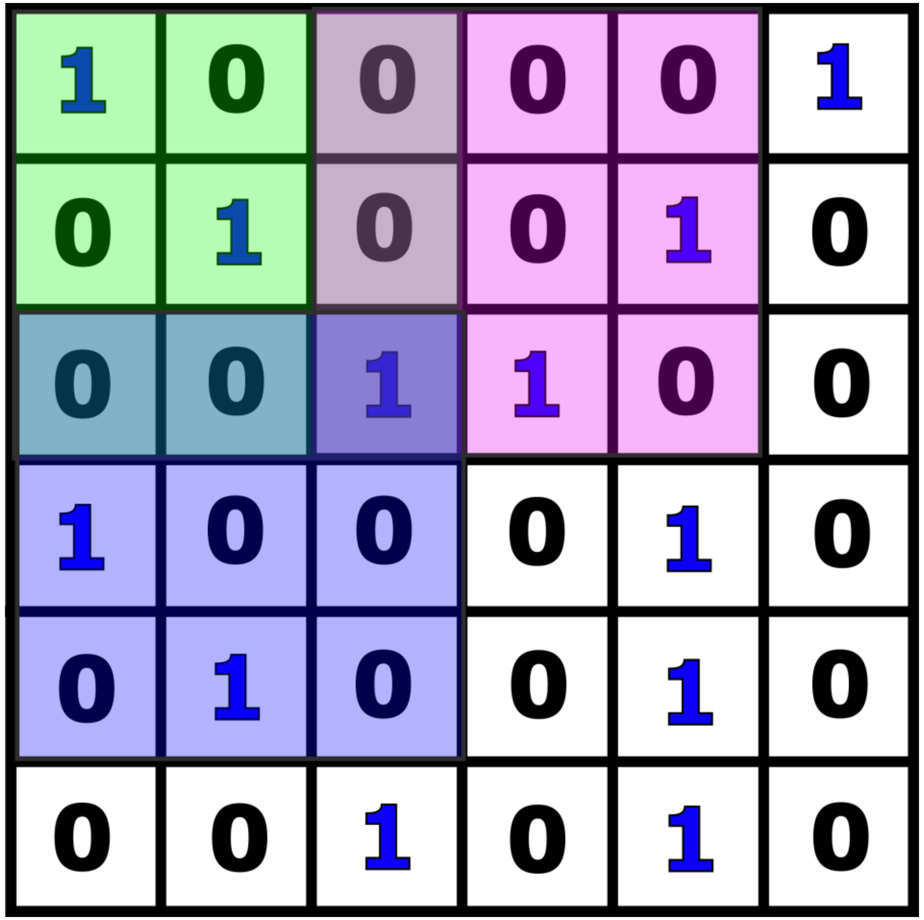


1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

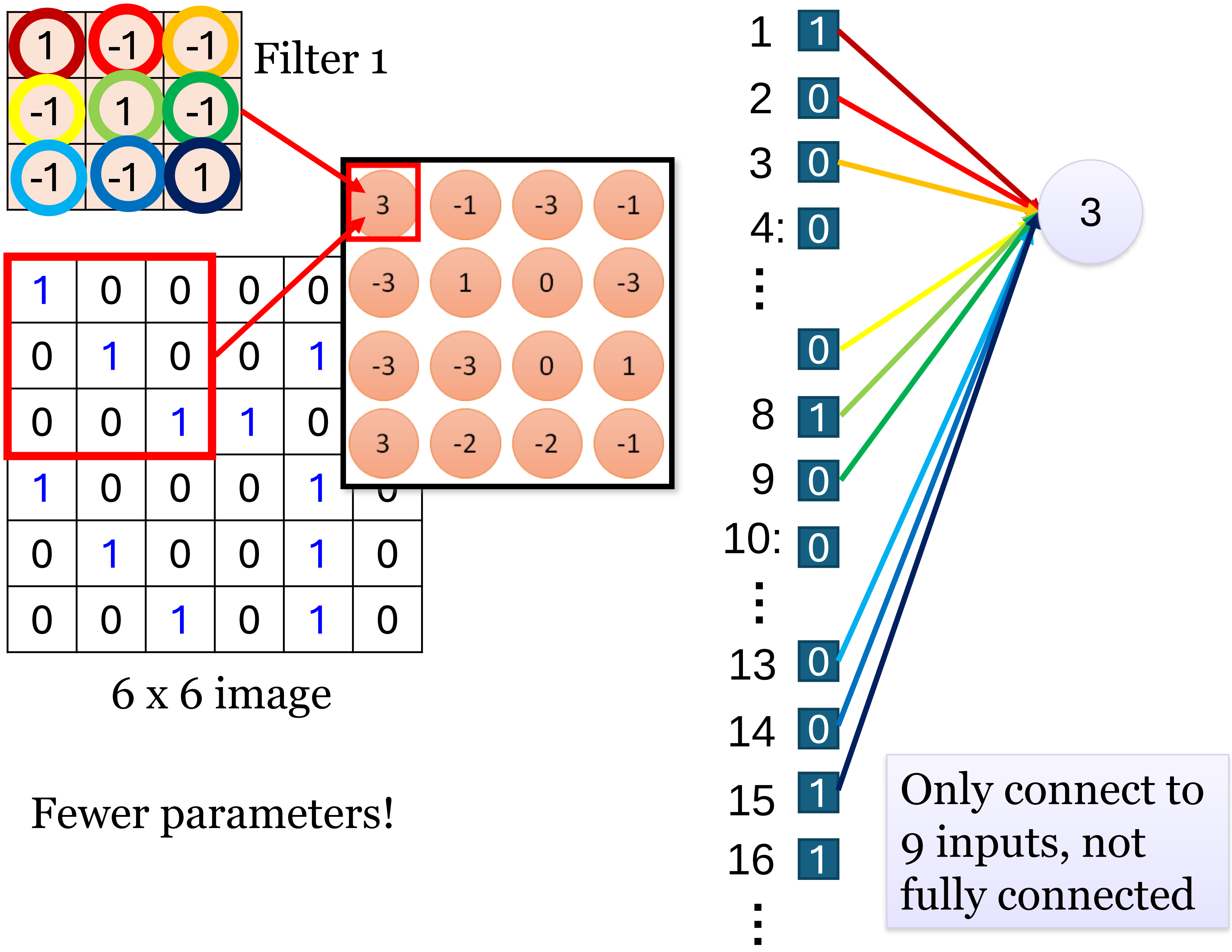
Stride



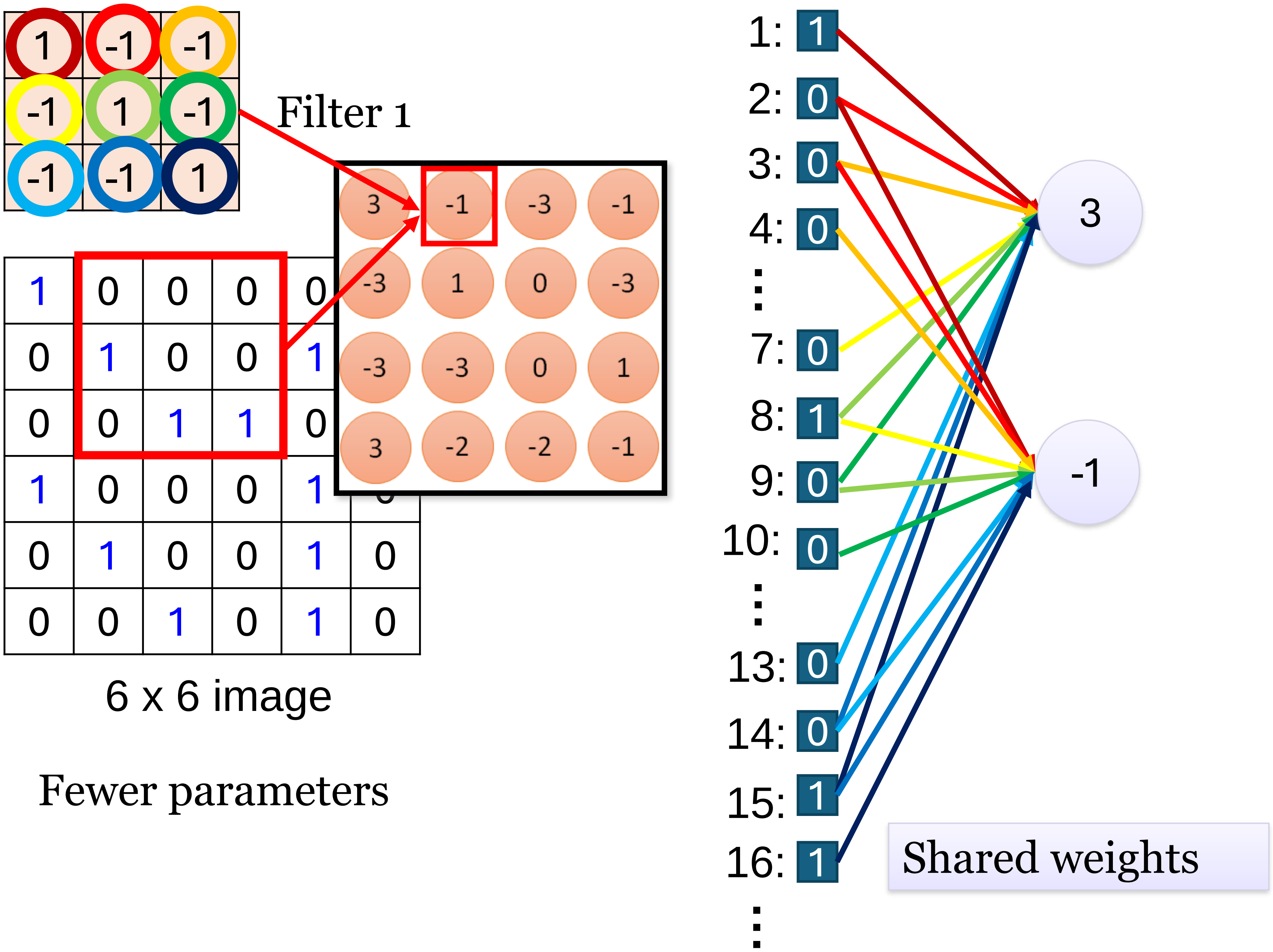
Stride = 1



Stride = 2



1 -1 -1
-1 1 -1
-1 -1 1
Filter 1
1 0 0 0 0
0 1 0 0 1
0 0 1 1 0
1 0 0 0 1
0 1 0 0 1 0
0 0 1 0 1 0
6 x 6 image
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
1 1
2 0
3 0
4: 0
0
8 1
9 0
10: 0
13 0
14 0
15 1
16 1
3
Fewer parameters!
Only connect to 9 inputs, not fully connected



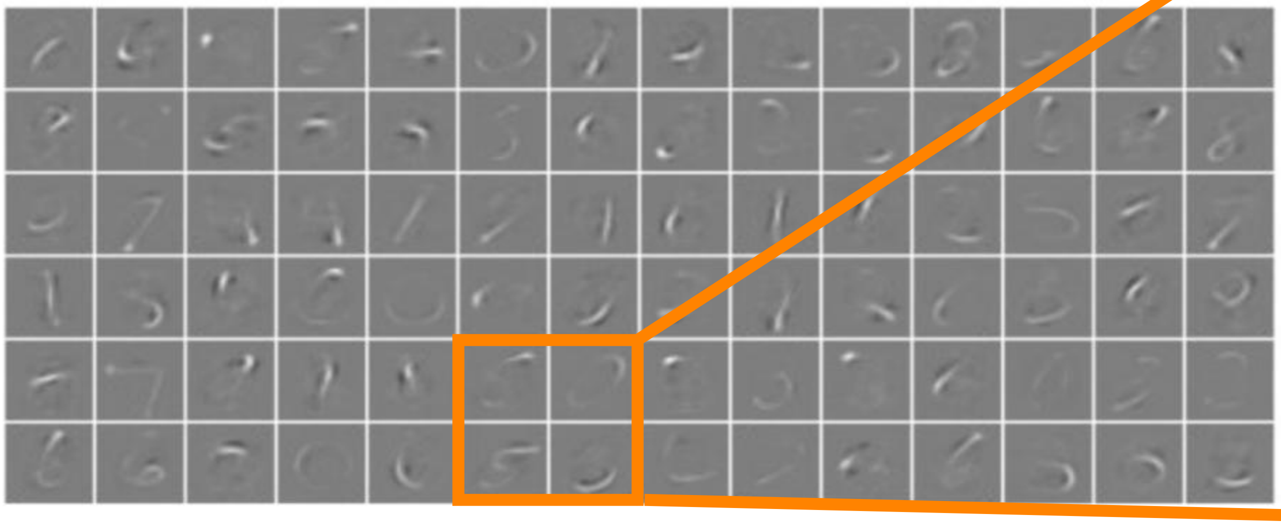
Filter 1
6 x 6 image
Fewer parameters
1: 1
2: 0
3: 0
4: 0
7: 0
8: 1
9: 0
10: 0
13: 0
14: 0
15: 1
16: 1
3
-1
Shared weights

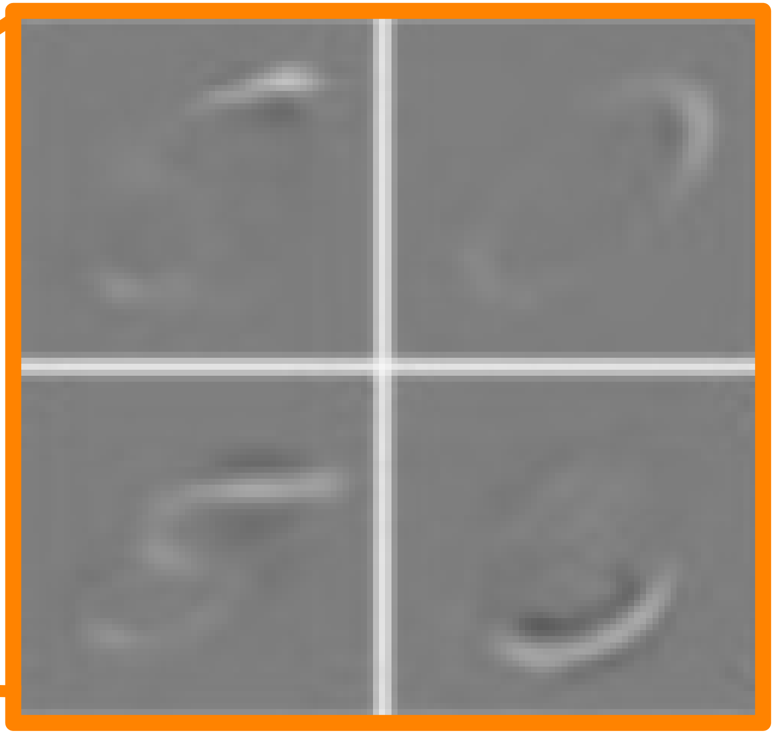
Result of Convolution - MNIST

Example training data – handwritten digits

- A CNN trained on the MNIST database of handwritten characters produces “activations” that contain the characteristics of the individual digits

Result of First Convolution





From Nash et al. “An Introduction to Convolutional Neural Networks”

POOLING

Pooling Layers

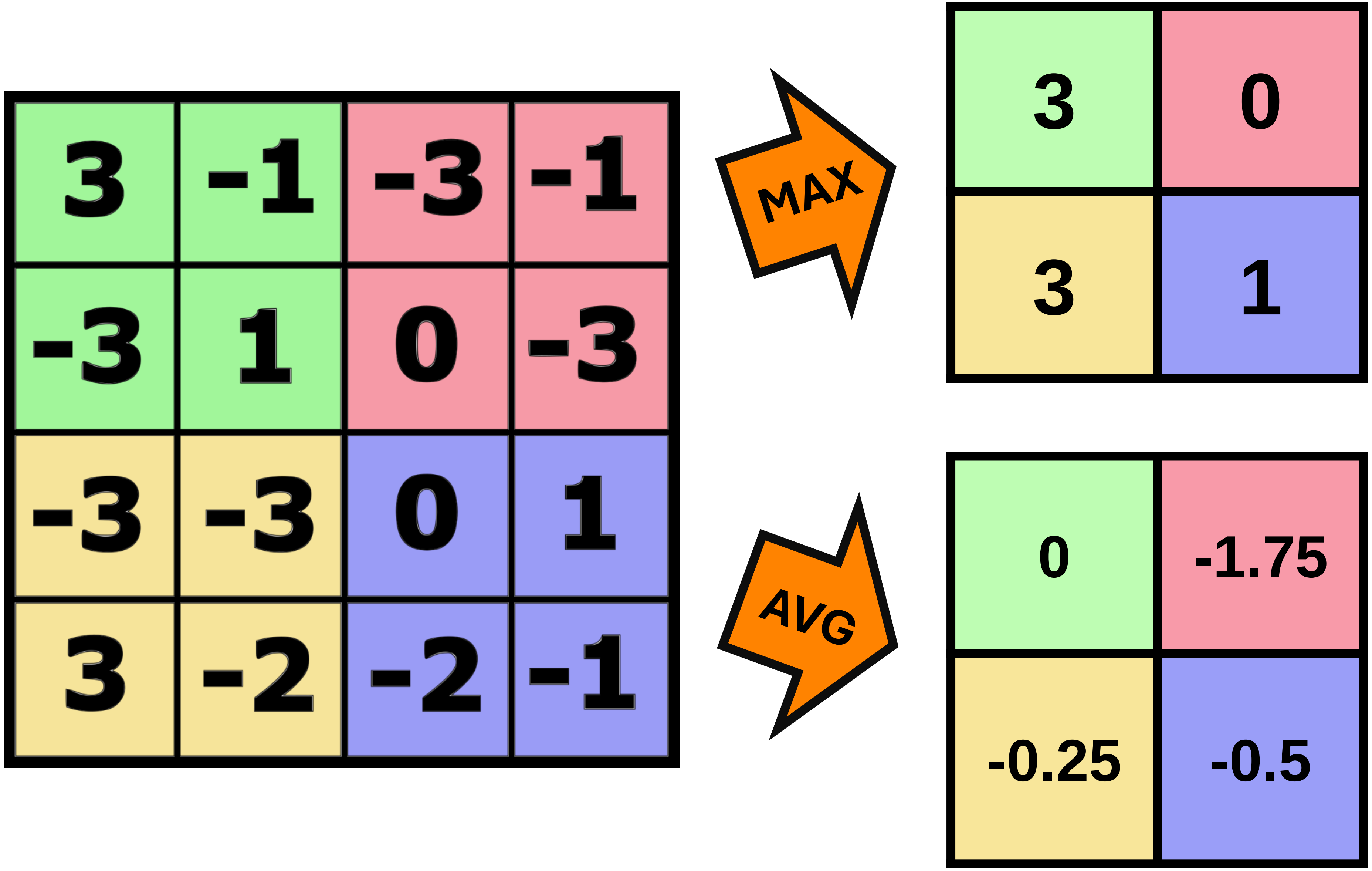
- Pooling makes an image smaller, so that the network requires fewer parameters

- Divide the image into regions. Create a new pixel value from each region, resulting in a smaller output array.

    - Max Pooling: Output for each region is max of pixel values in region
    - Average Pooling: Output for each region is average of pixel values in region

Pooling layers make the image smaller without changing the object

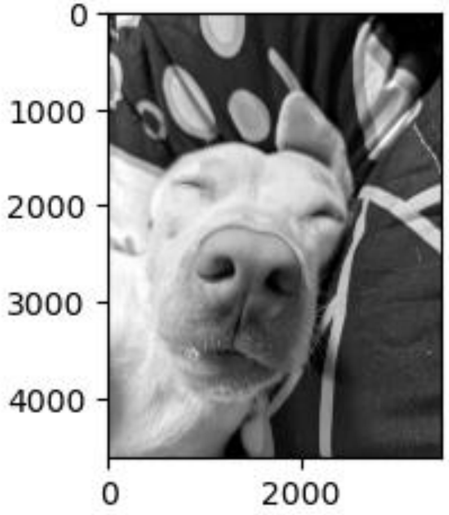
Max Pool and Average Pool



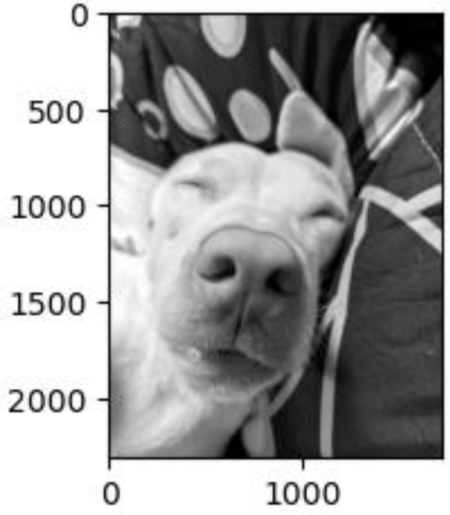
Pooling a Real Image

- Subsampling pixels does not change the object

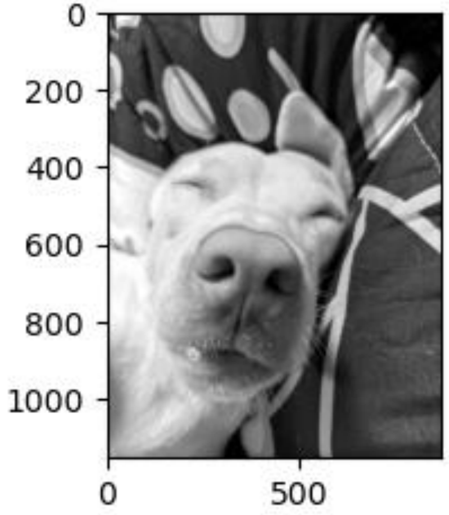
3742*4624



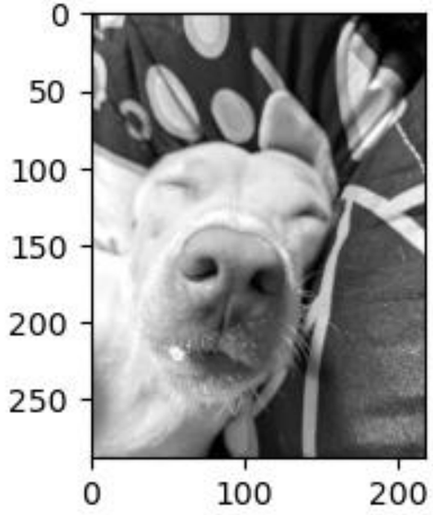
2312*1736



1156*868

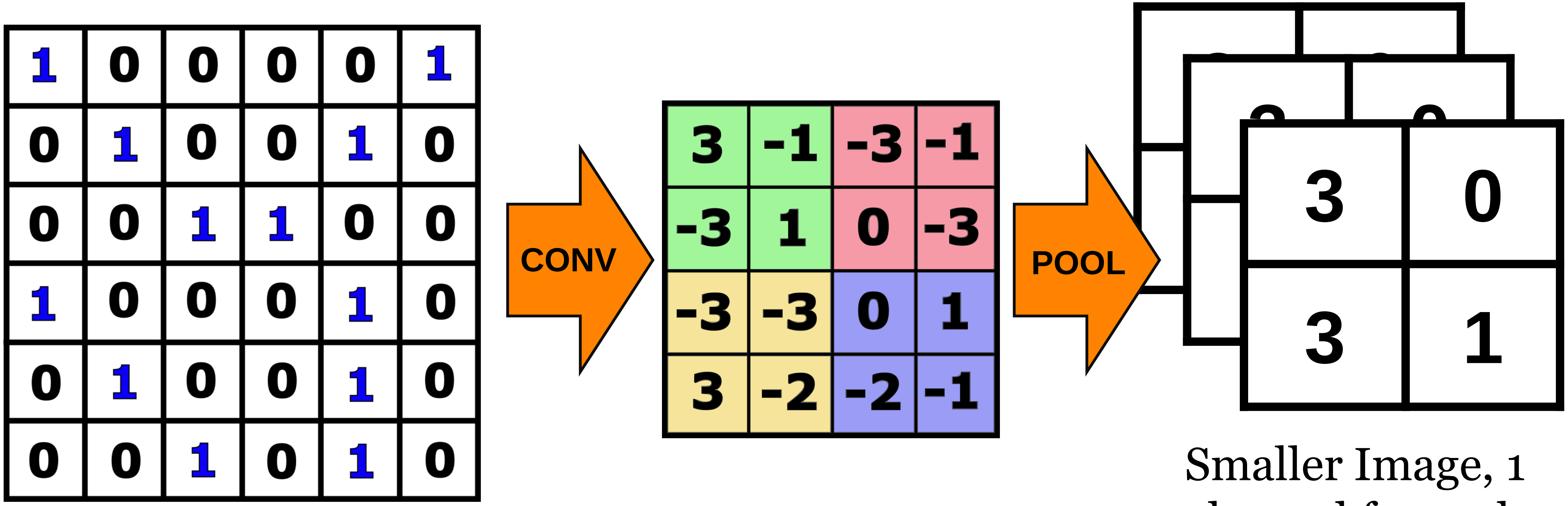


289*217



ASSEMBLING THE CNN

Moving Through the Layers

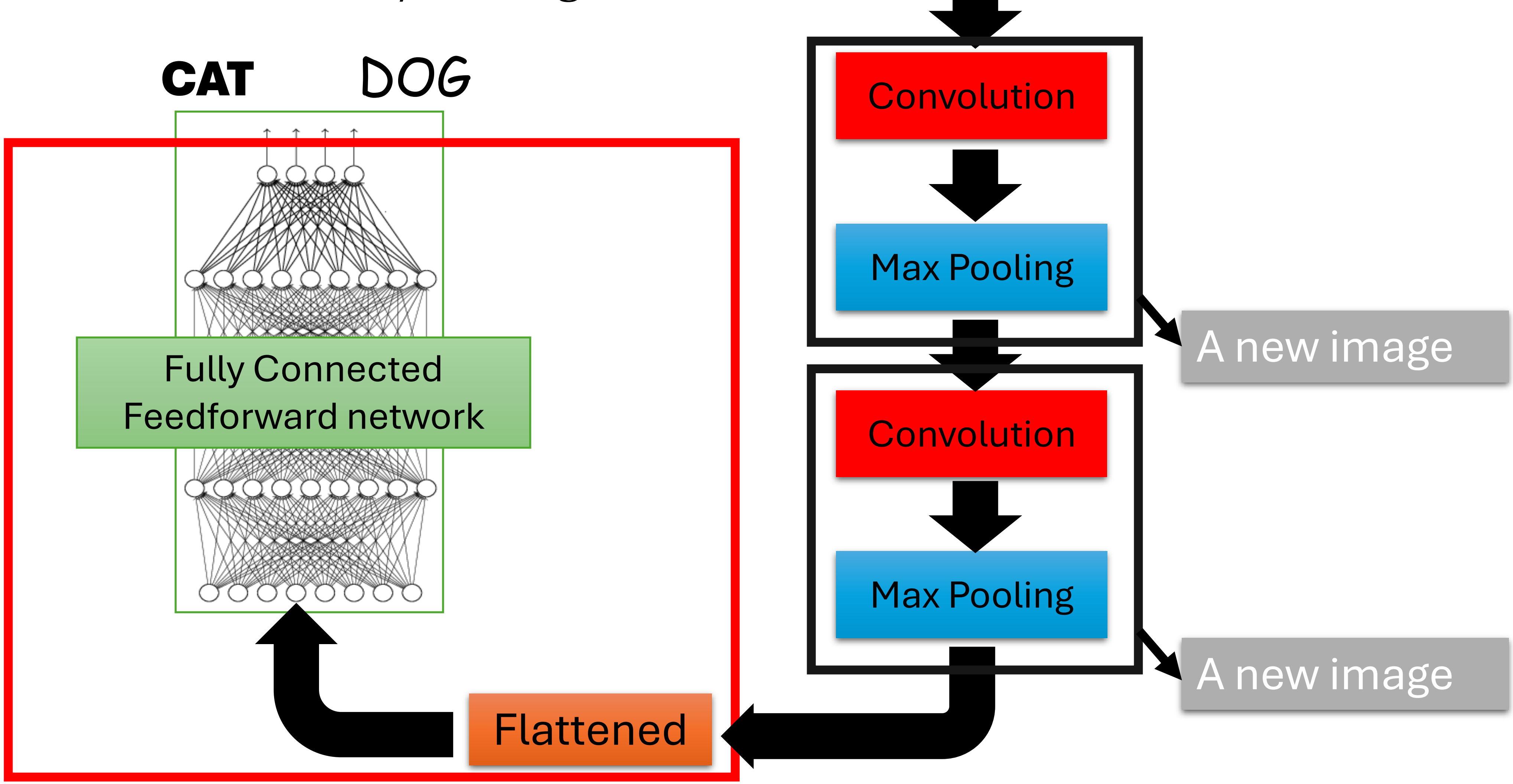


Input Image

Smaller Image, 1 channel for each filter

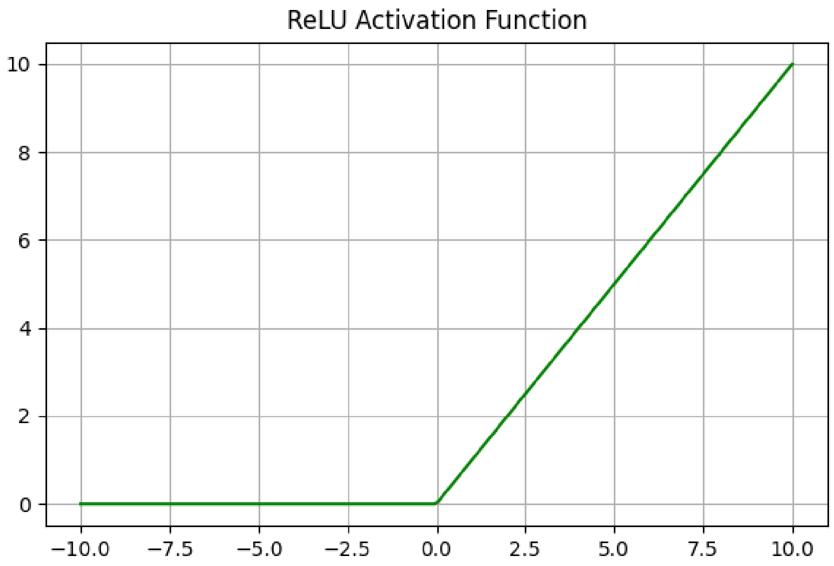
Assembling CNN

Can have a variable number of Convolution/Pooling units



Fully Connected Layers

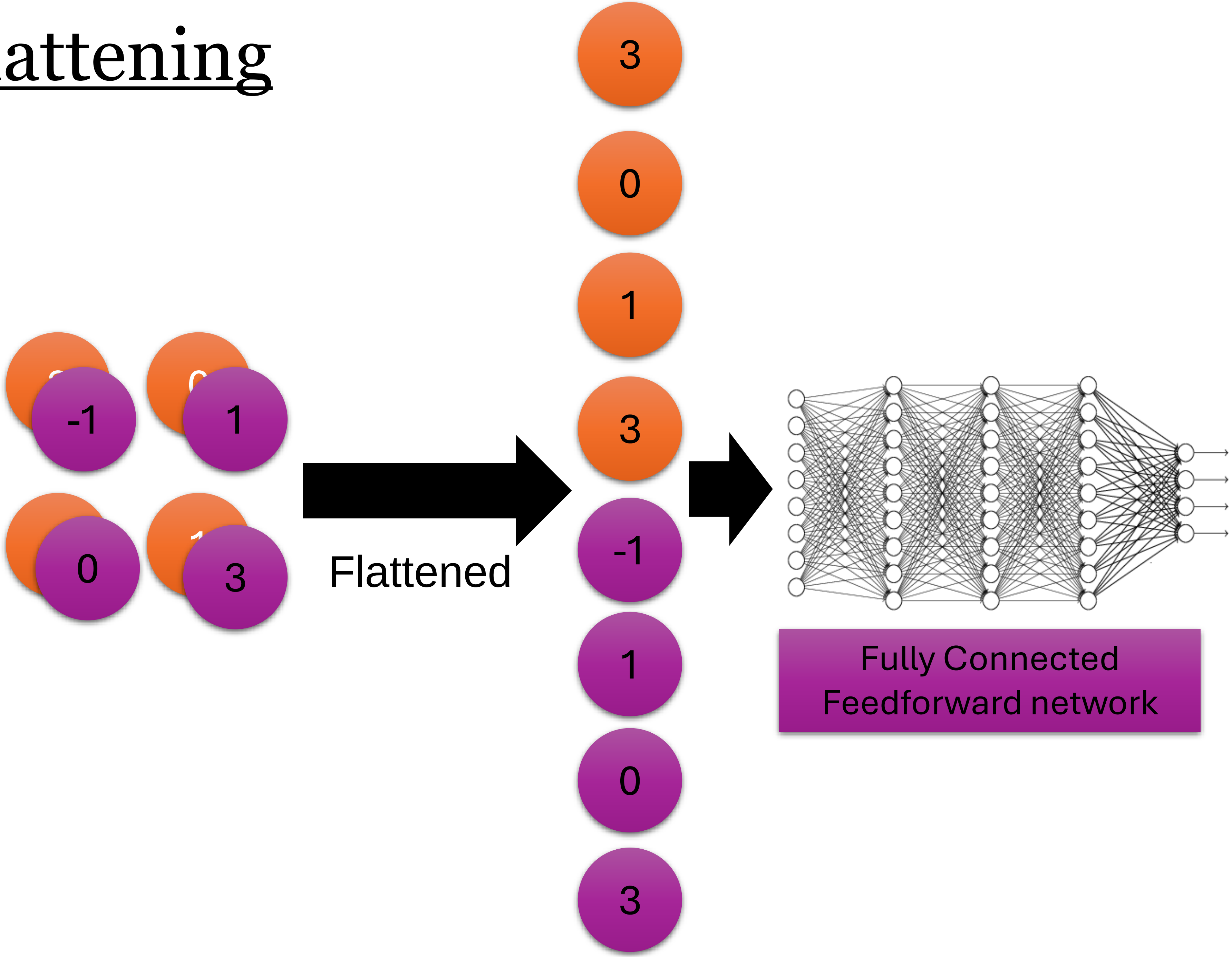
- The last layers in the CNN are fully connected layers which form a standard feedforward network
  - Feedforward – data only moves forward. Nodes do not send data back to previous layers
- Data must be flattened before sending it to the dense network



- It is suggested to use an activation layer (often with ReLU) between these layers to improve performance

The feature maps produced by convolution and pooling are flattened and fed into a feedforward neural network that determines the final image classifications

Flattening


-1
1
0
3
Flattened
3
0
1
3
-1
1
0
3
Fully Connected
Feedforward network

KERAS

Layer Types

1. **Convolutional Layer:** Extracts features from the input image through convolution operations. Key Parameters: Number of filters, kernel size, strides, padding.

2. **Activation Layer:** Introduces non-linearity to the model, allowing it to learn complex patterns. Common Types: ReLU (Rectified Linear Unit), Sigmoid, Tanh.

3. **Pooling Layer:** Reduces the spatial dimensions (height and width) of the input volume, making the model more computationally efficient and less sensitive to the exact location of features. Types: Max Pooling, Average Pooling.

4. **Batch Normalization Layer:** Normalizes the output of a previous layer by subtracting the batch mean and dividing by the batch standard deviation. Helps in speeding up training and reducing the sensitivity to network initialization.

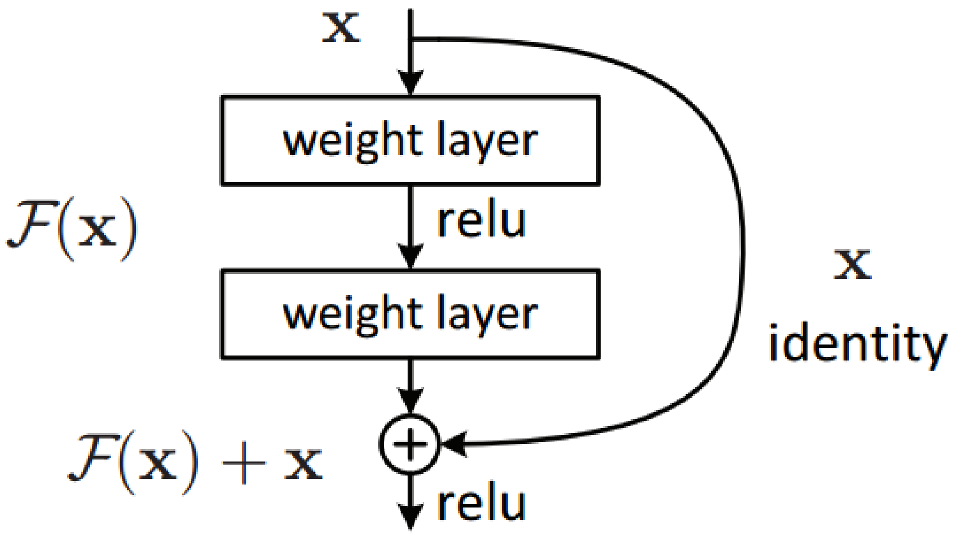
5. **Dropout Layer:** Randomly sets a fraction of input units to zero at each update during training, helping to prevent overfitting.

Layer Types

6. **Fully Connected (Dense) Layer:** Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular neural networks. Used to classify or regress based on features extracted by the convolutional layers.

7. **Flatten Layer:** Flattens the input from a multi-dimensional tensor to a 1D tensor, making it possible to connect convolutional and pooling layers with dense layers.

8. **Residual Connections** (as in ResNet):Allows the output of one layer to bypass one or more layers and be added to the output of a later layer, improving gradient flow through the network.

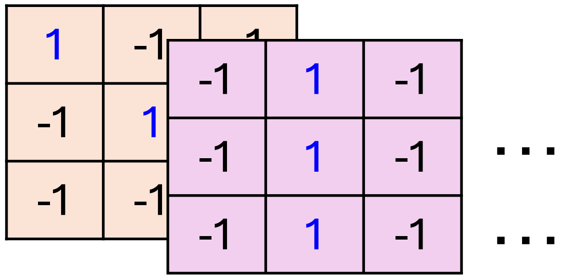


9. **Transposed Convolutional Layer (Deconvolution):** Used in generative models and some segmentation tasks; it upscales the input feature map.

Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D tensor)***

```
model2.add( Convolution2D( 25,3,3,
         input_shape=(28,28,1)) )
```



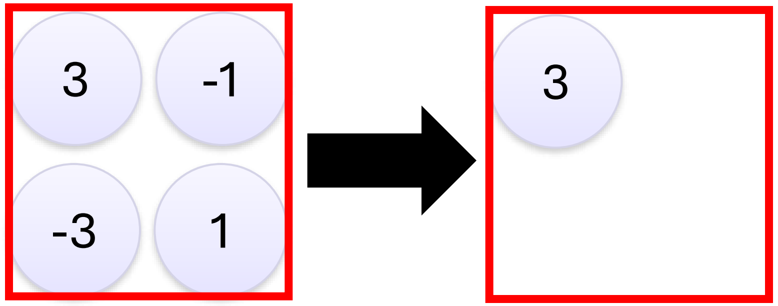
There are 25 3x3 filters.

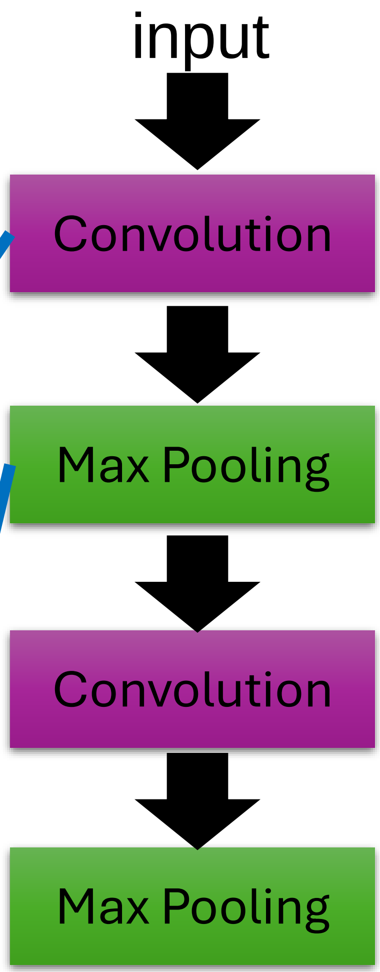
Input_shape = (28 , 28 , 1)

28 x 28 pixels

1: black/white, 3: RGB

```
model2.add(MaxPooling2D((2,2)))
```





Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D array)***

Input

1 x 28 x 28

```
model2.add( Convolution2D( 25,3,3,
            input_shape=(28,28,1)) )
```

Convolution

25 x 26 x 26

```
model2.add(MaxPooling2D((2,2)))
```

Max Pooling

25 x 13 x 13

```
model2.add(Convolution2D(50,3,3))
```

Convolution

50 x 11 x 11

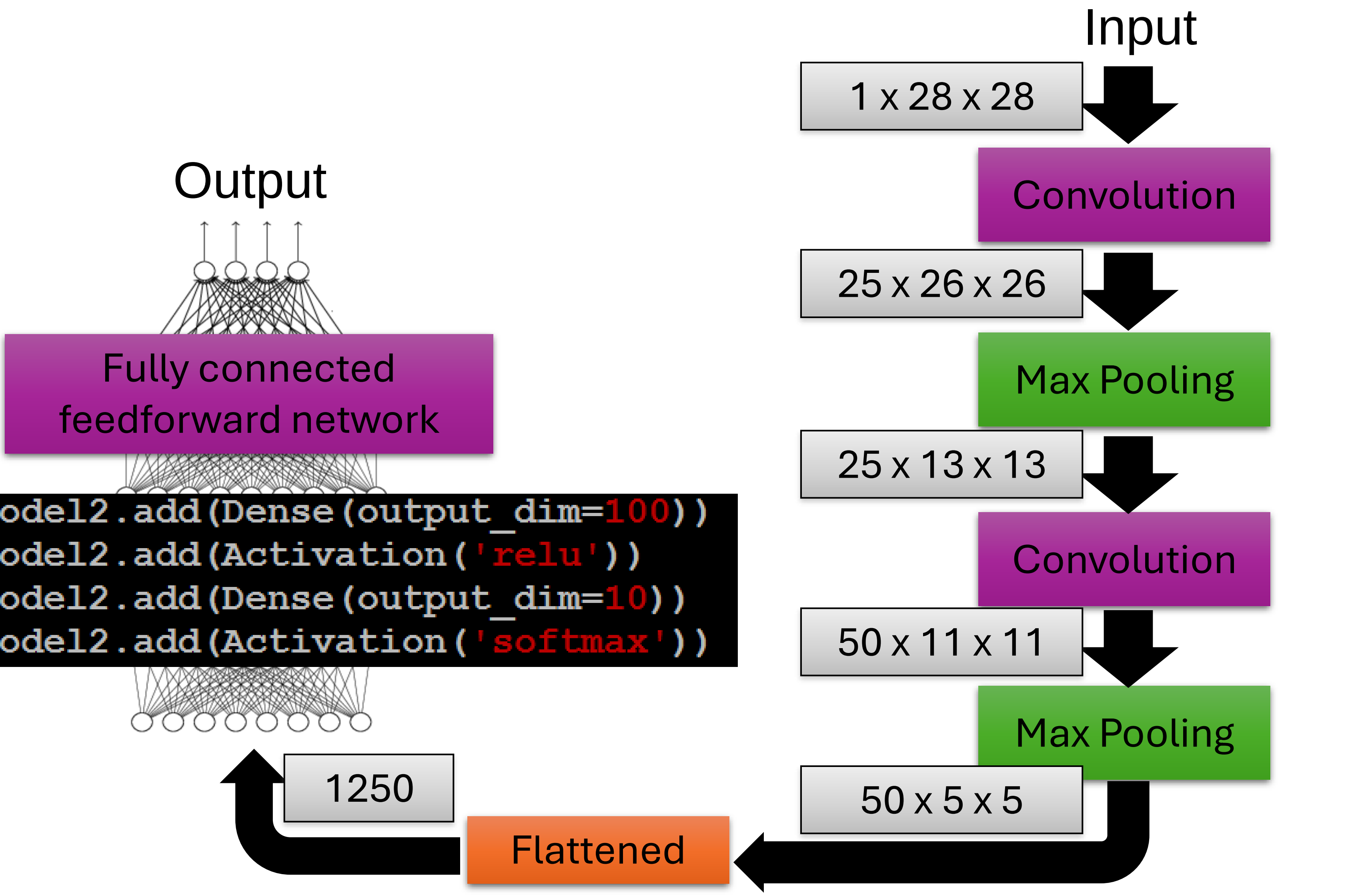
```
model2.add(MaxPooling2D((2,2)))
```

Max Pooling

50 x 5 x 5

Keras

Only modified the *network structure* and *input format (vector -> 3-D array)*




```
model2.add(Dense(output_dim=100))
model2.add(Activation('relu'))
model2.add(Dense(output_dim=10))
model2.add(Activation('softmax'))
```

```
model2.add(Flatten())
```

DCNN vs. Multi-Layer Perceptron (MLP)

1. Parameter Efficiency

- Fewer Parameters: CNNs require significantly fewer parameters than FCNs. They use shared weights and convolutional filters, reducing the total number of trainable parameters. This makes CNNs more efficient and less memory-intensive.
- Reduces Overfitting: With fewer parameters, CNNs are less prone to overfitting, especially with image data.

2. Exploitation of Spatial Structure

- Local Connectivity: CNNs exploit the spatial structure of the data by applying convolutional filters that capture local features (like edges, textures) in early layers and more complex features (like patterns or object parts) in deeper layers.
- Preservation of Spatial Relationships: Unlike FCNs that lose spatial relationships by flattening the input, CNNs maintain the spatial hierarchy and relationships between different parts of the input.

3. Translation Invariance

- Robust to Translation: Due to pooling layers and the nature of convolution operations, CNNs are inherently more robust to the translation of input data. This means that if an object shifts in an image, a CNN can still detect it effectively.

Coding Task