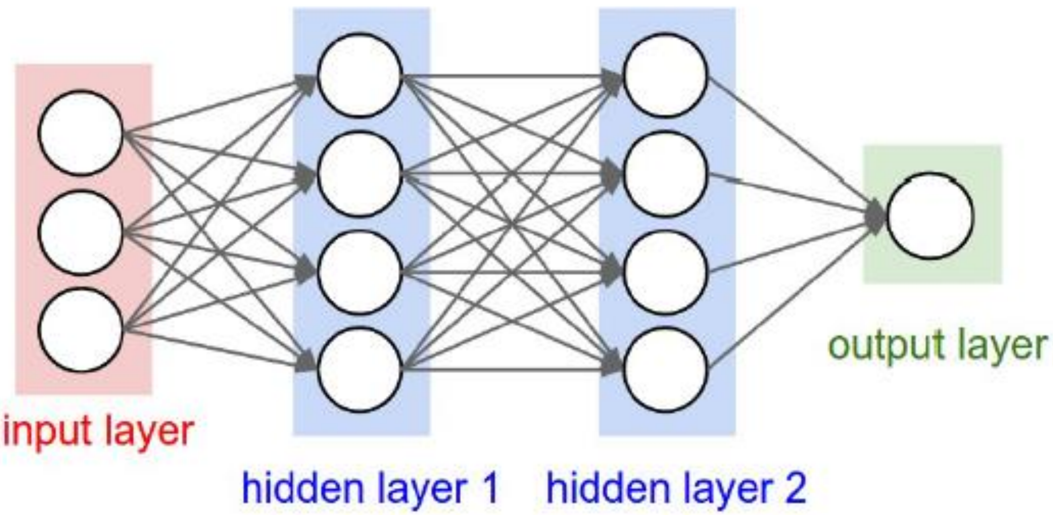
Deep learning

Azim Dehghani Amirabad

ANN



input layer: $x \in \mathbb{R}^{3*1}$
activation function: $f(x) = \frac{1}{1+exp(-x)}$
hidden layer 1: $h_1 = f(W_1^T x + b_1)$
hidden layer 2: $h_2 = f(W_2^T h_1 + b_2)$
output layer: $out = W_3^T h_2 + b_3$

Model Parameters:

- weights $W_1 \in \mathbb{R}^{3*4}$, $W_2 \in \mathbb{R}^{4*4}$, $W_3 \in \mathbb{R}^{4*1}$
- biases $b_1 \in \mathbb{R}^{4*1}$, $b_2 \in \mathbb{R}^{4*1}$, $b_3 \in \mathbb{R}$

Fig. courtesy to Andrej Karpathy [Karpathy, 2017]

How we can an architecture which tries to take advantage of the spatial structure?

Answer: CNN

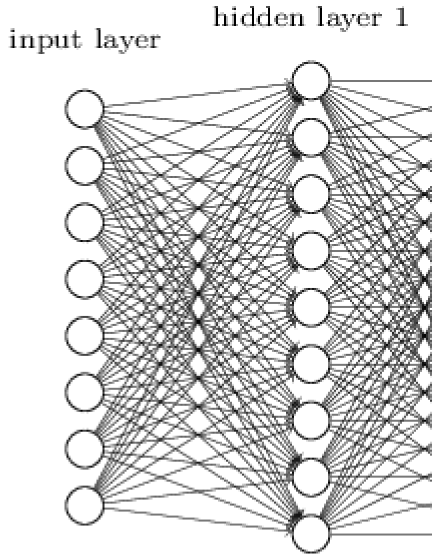
CNN = Neural Network with a convolutional function in at least one of the layers

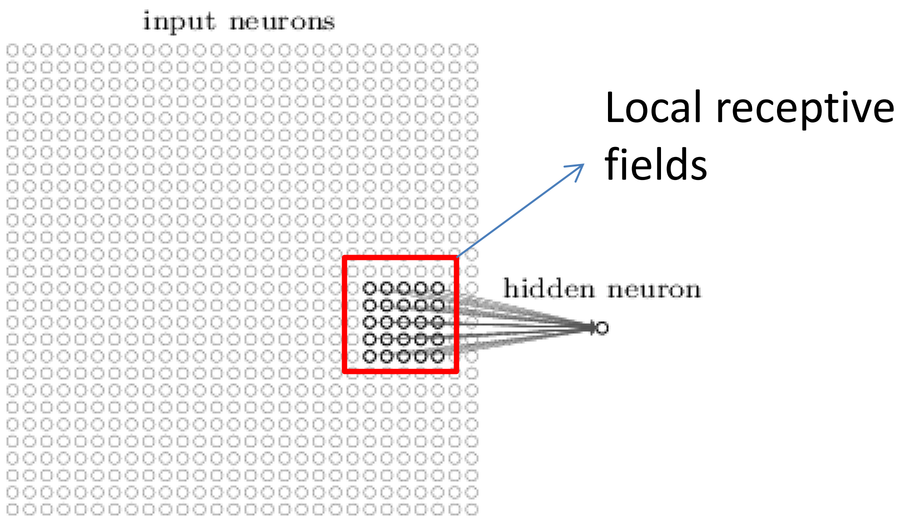
Convolutional neural networks

three basic ideas of CNN :

- *local receptive fields*,
- *shared weights*,
- *pooling*

local receptive fields





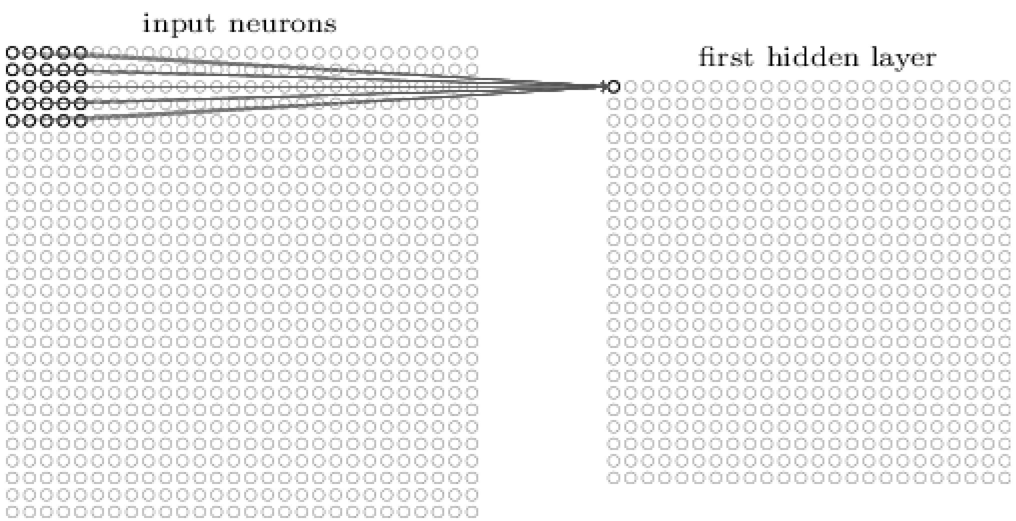
make connections in small, localized regions

Fully connected layer

won't connect every input pixel to every hidden neuron. Instead, we only make connections in small, localized regions of the input image.

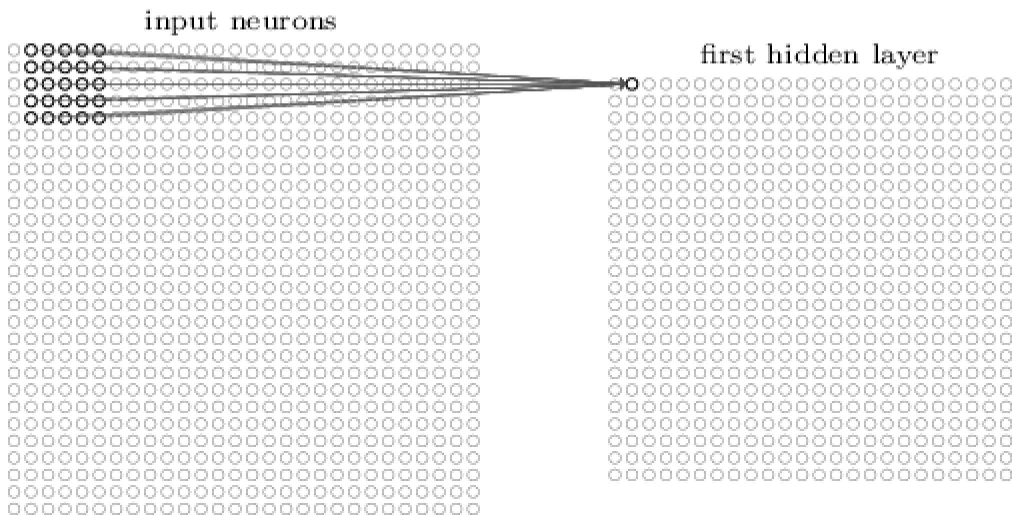
You can think of that particular hidden neuron as learning to analyze its particular local receptive field.

local receptive fields



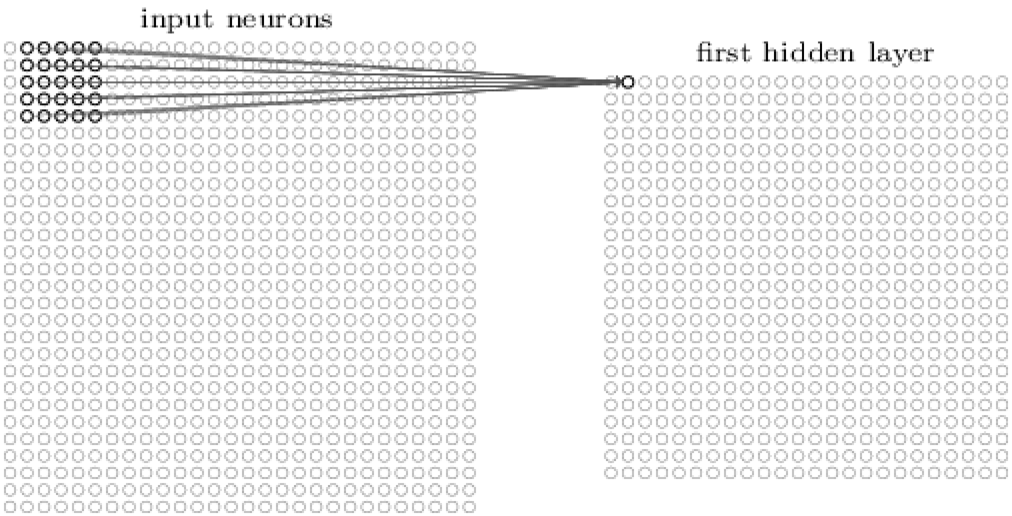
You can think of that particular hidden neuron as learning to analyze its particular local receptive field.

local receptive fields



stride length: total size of the pixel that we move local receptive fields right or down

Shared weights and biases

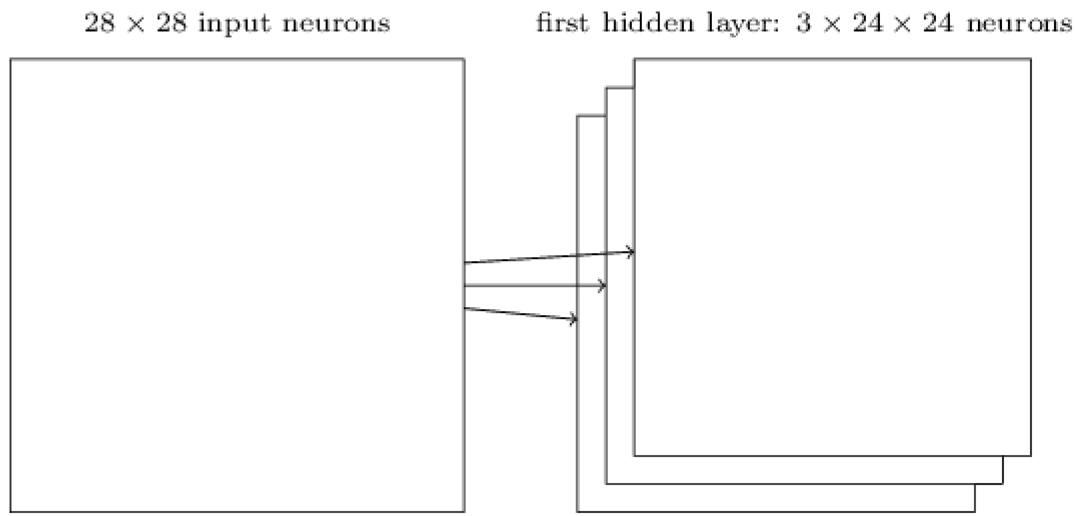


use the *same* weights and bias for each of the 24×2424×24 hidden neurons

This means that all the neurons in the first hidden layer detect exactly the same feature**

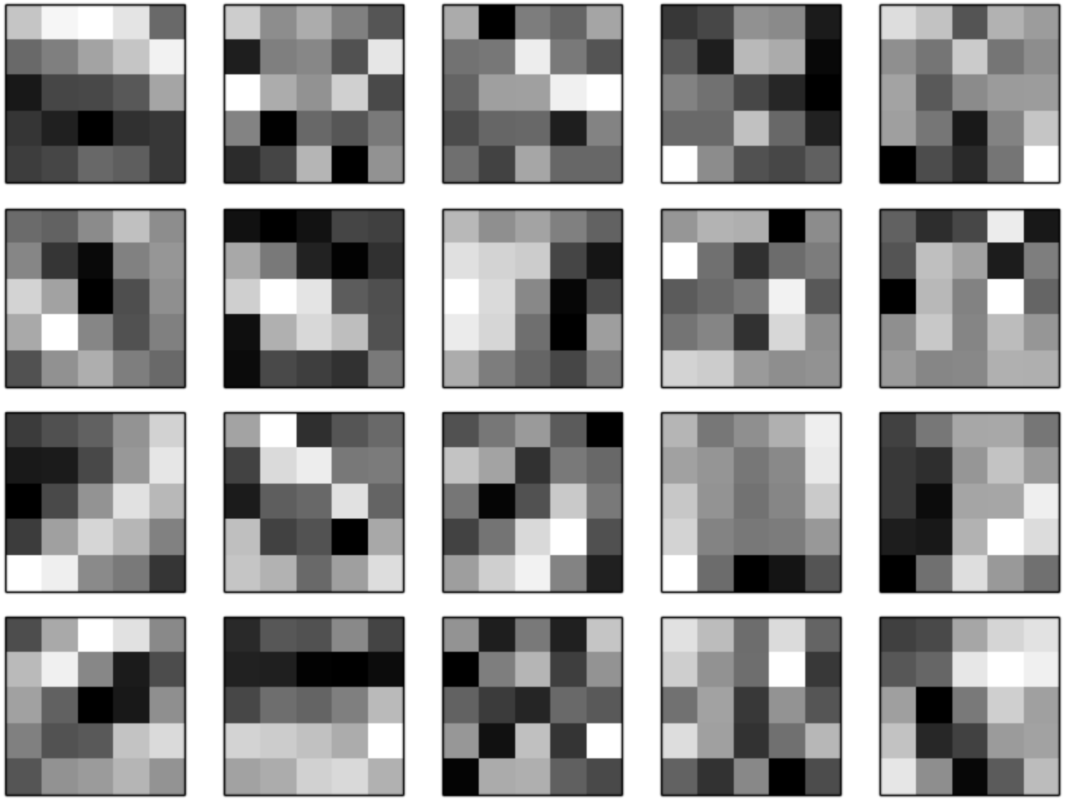
For this reason, we sometimes call the map from the input layer to the hidden layer a ***feature map***

Complete convolutional layer consists of several different feature maps



In the example shown, there are 33 feature maps. Each feature map is defined by a set of 5×55×5 shared weights, and a single shared bias. The result is that the network can
detect 33 different kinds of features, with each feature being detectable across the entire image.

Some of the features which are learned

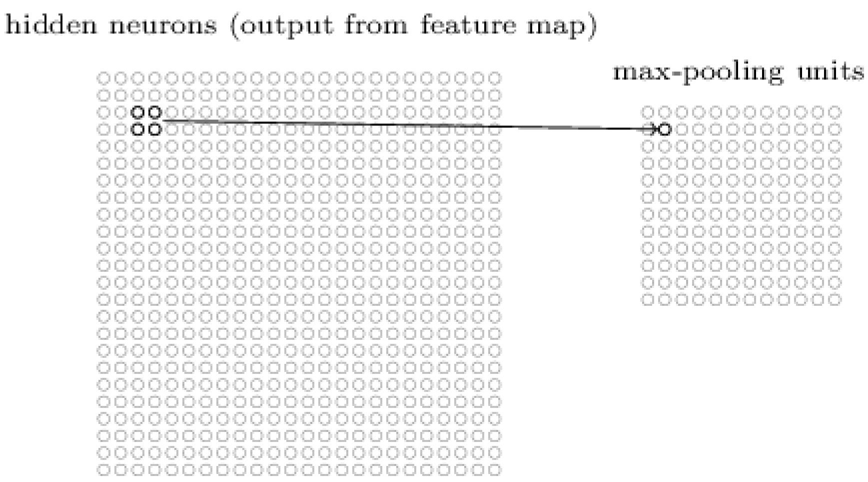


Each map is represented as a 5×5 block image

A big advantage of sharing weights and biases is that it greatly reduces the number of parameters involved in a convolutional network.

Pooling layers

Pooling layers are usually used immediately after convolutional layers.
What the pooling layers do is simplify the information in the output from the convolutional layer.



Pooling Layer

- Method -

- sampling technique
- makes the network more robust to noise or shifts
- decreases input size with **strides**
- different types:
  - **Max Pooling**
  - **Average Pooling**
- in practice, **Max Pooling** works best in practice [Scherer et al., 2010]

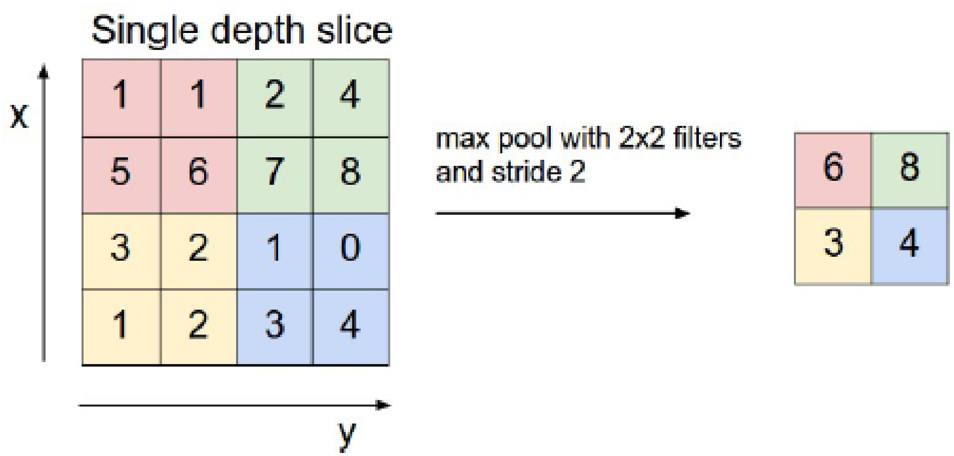
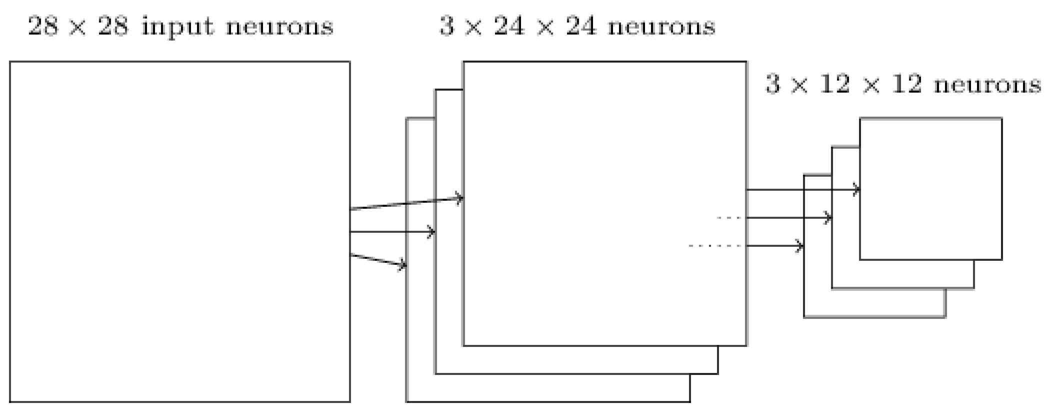


Fig. courtesy to Andrej Karpathy [Karpathy, 2017]

Pooling layers



helps reduce the number of parameters
needed in later layers

CNN architecture

- **Vanilla Architecture**:
  - Input Layer
  - Convolutional Layer
  - Activation Layer
  - Pooling Layer
  - Fully Connected Layer
- **Additional Layers**:
  - Dropout Layer
  - Batch Normalization Layer

Activation Layer

- Method -

- makes the network learn a non-linear model
- different types:
  - **ReLU**: $relu(x) = max(0, x)$
  - **Sigmoid**: $\sigma(x) = 1/(1+e^{-x})$
  - **Tanh**: $tanh(x) = 2\sigma(2x) - 1$
- in practice, **ReLU** converges faster with Stochastic Gradient Descent [Krizhevsky et al., 2012]

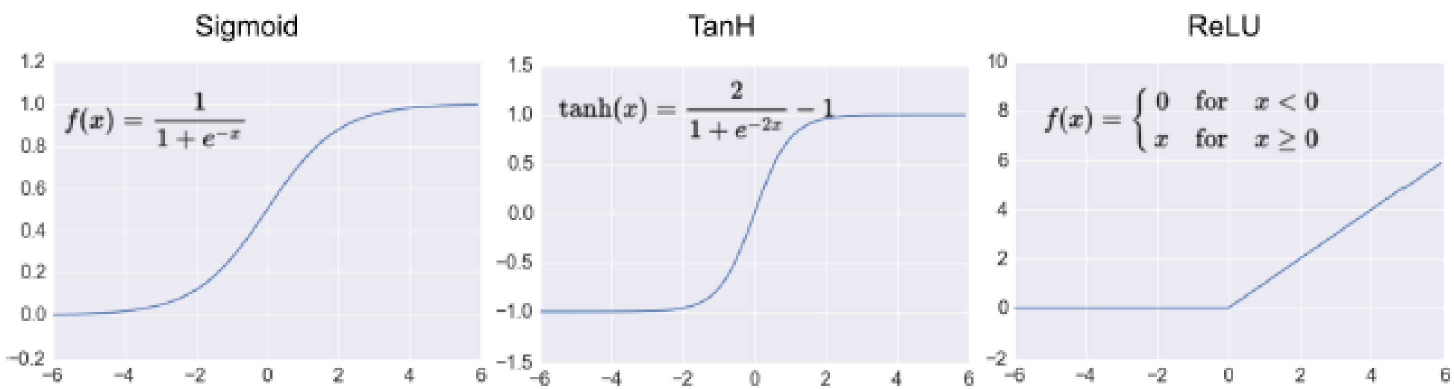
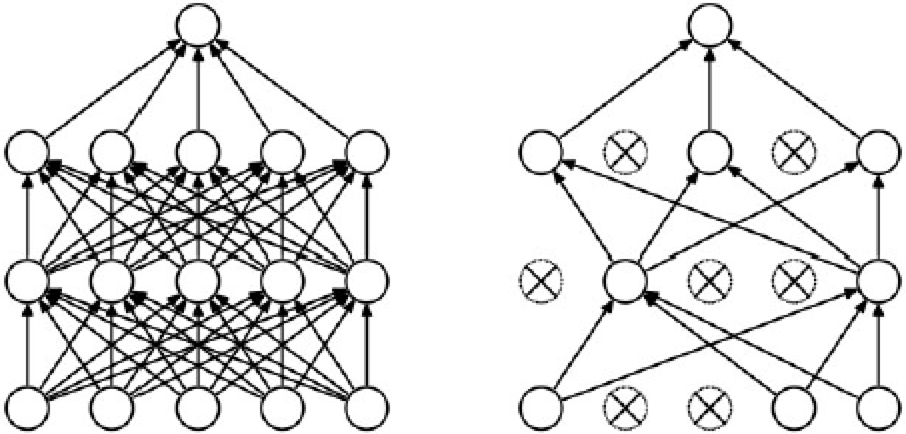


Fig. courtesy to Adil Moujahid [Moujahid, 2017]

Dropout and Batch Normalization Layers

- Method -

- **Dropout**: addresses overfitting
- **Batch Normalization**: helps with the issue of vanishing gradients

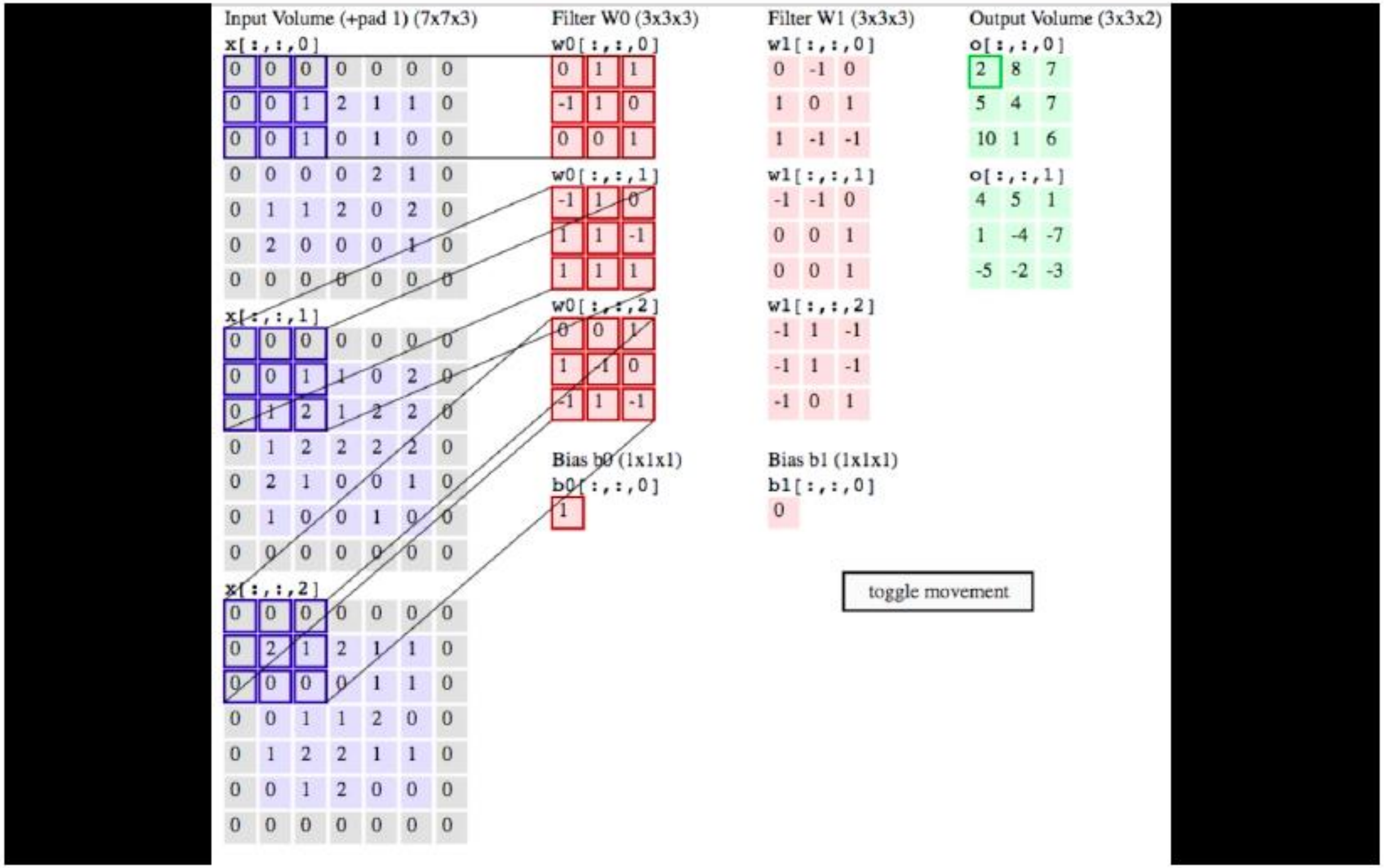


(a) Standard Neural Net (b) After applying dropout.

Fig. courtesy to [Srivastava et al., 2014]

Convolutional Layer [1]

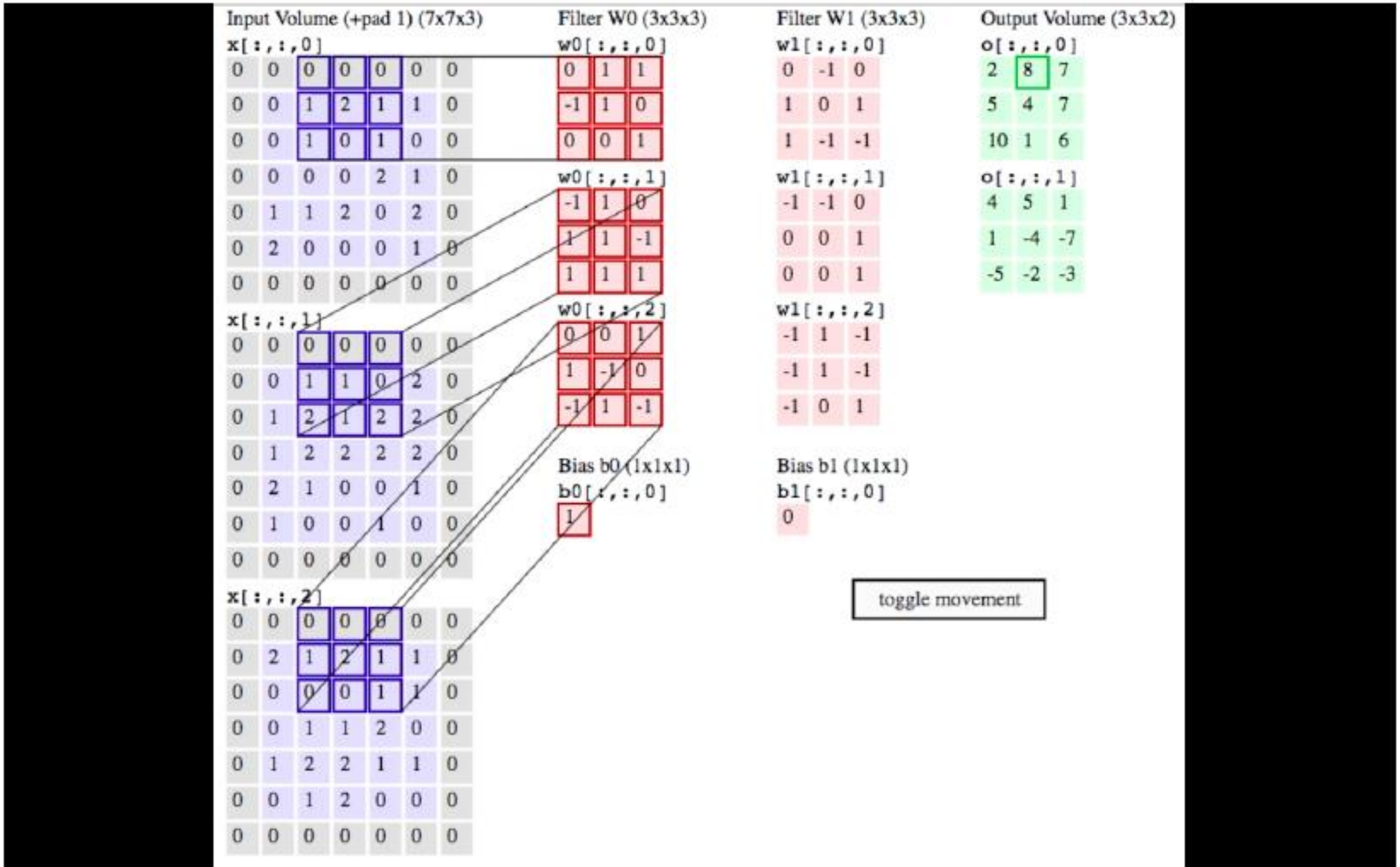
- Method -



Demo courtesy to Andrej Karpathy [Karpathy, 2017]

Convolutional Layer [1]

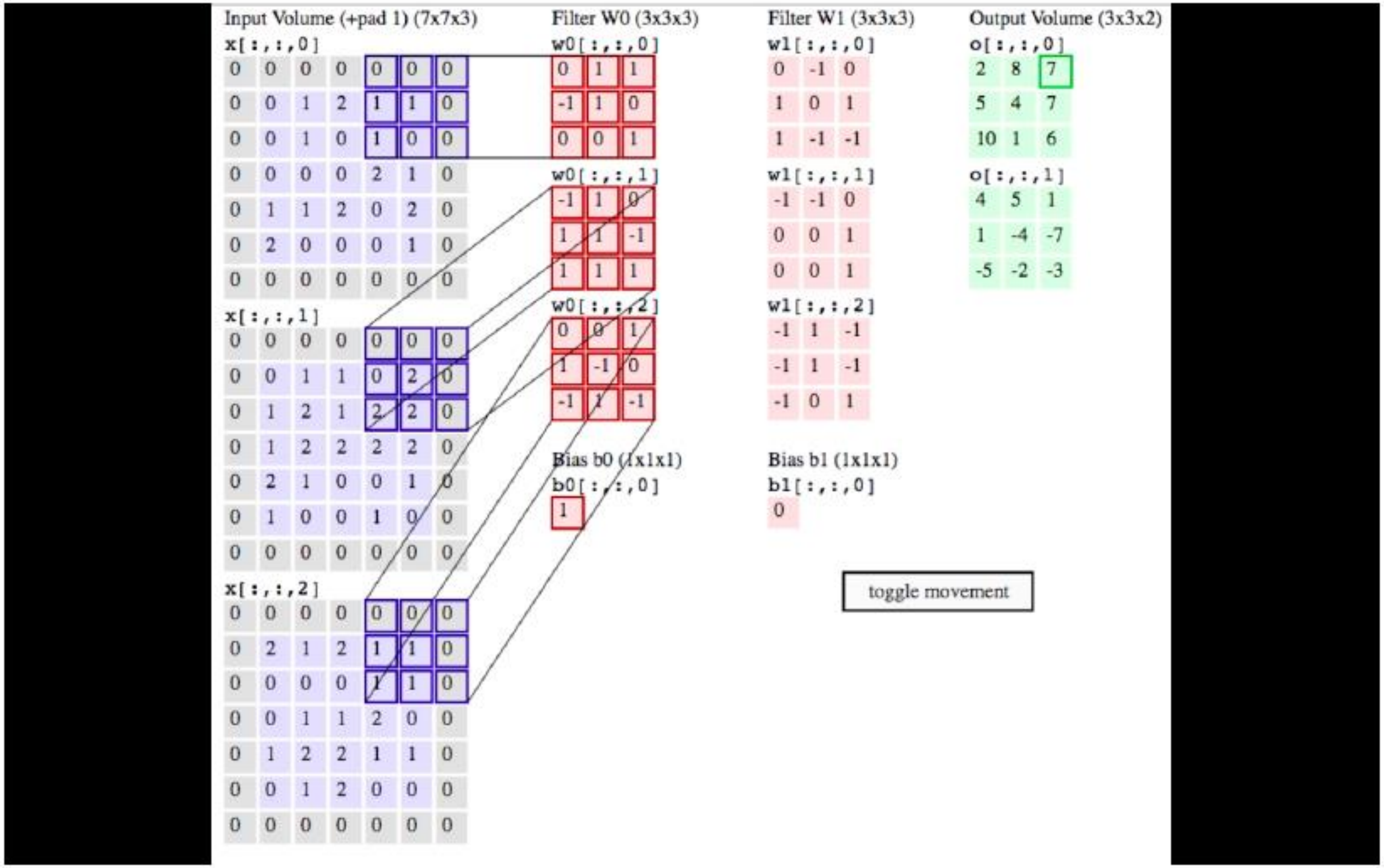
- Method -



Demo courtesy to Andrej Karpathy [Karpathy, 2017]

Convolutional Layer [1]

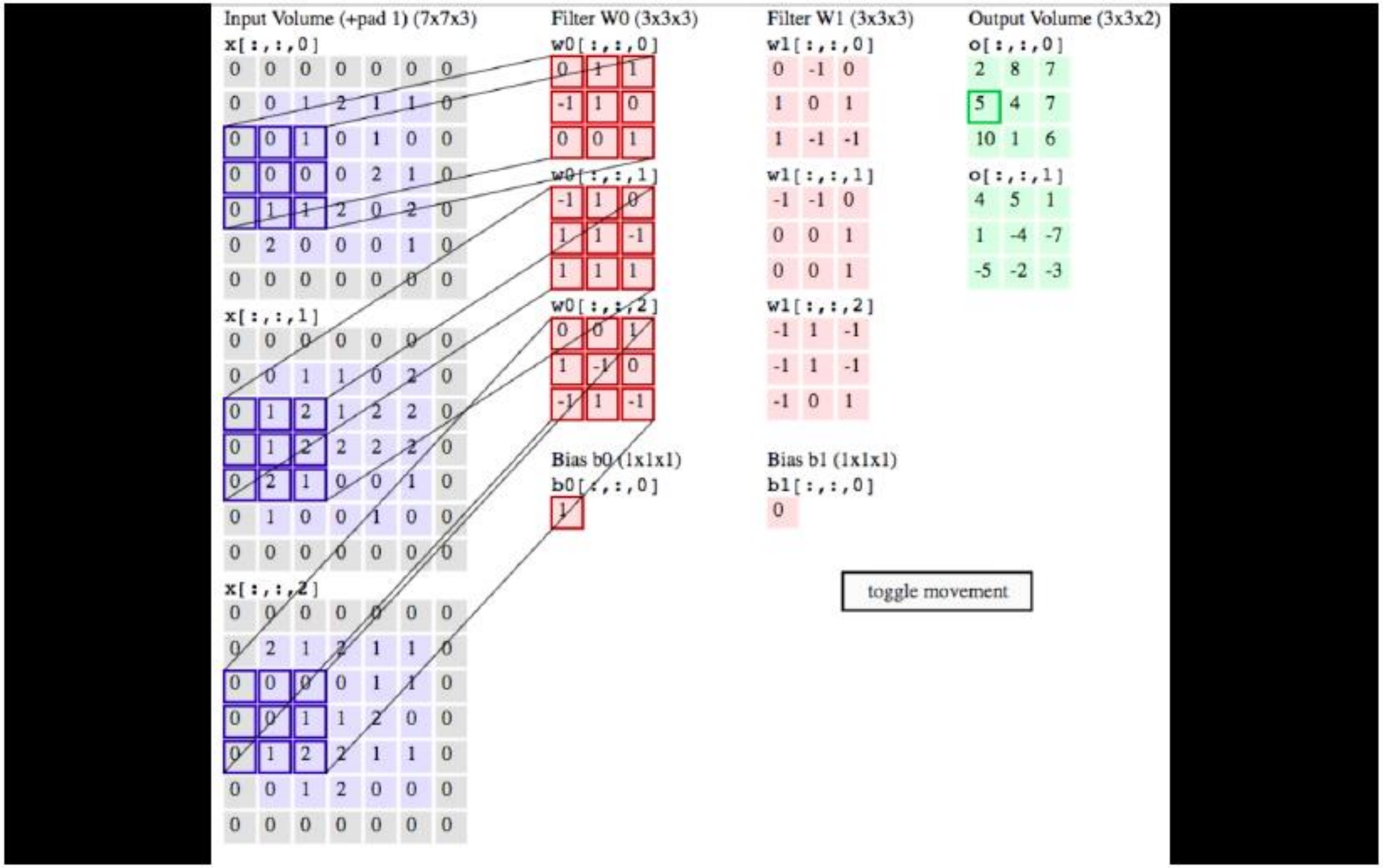
- Method -



Demo courtesy to Andrej Karpathy [Karpathy, 2017]

Convolutional Layer [1]

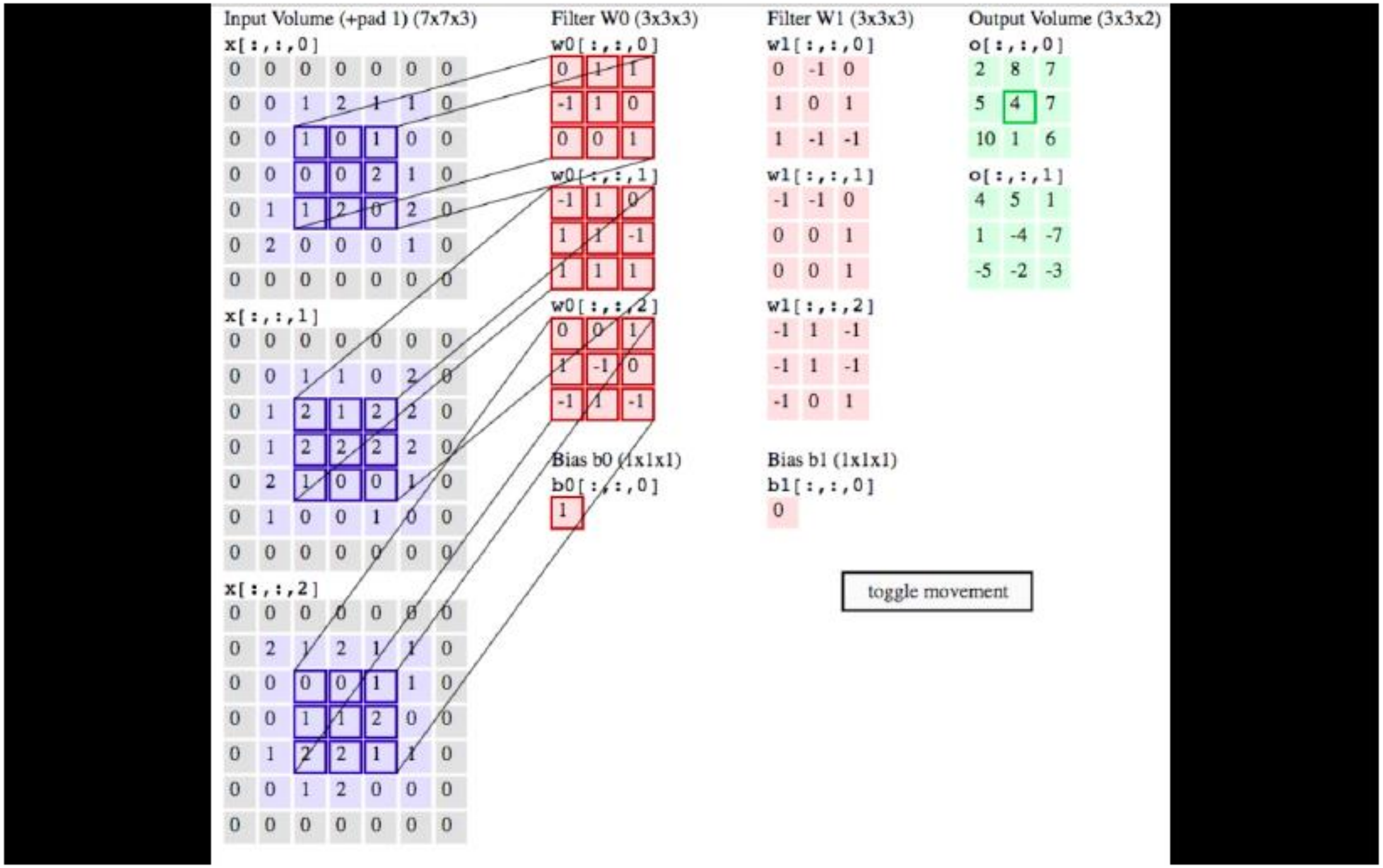
- Method -



Demo courtesy to Andrej Karpathy [Karpathy, 2017]

Convolutional Layer [1]

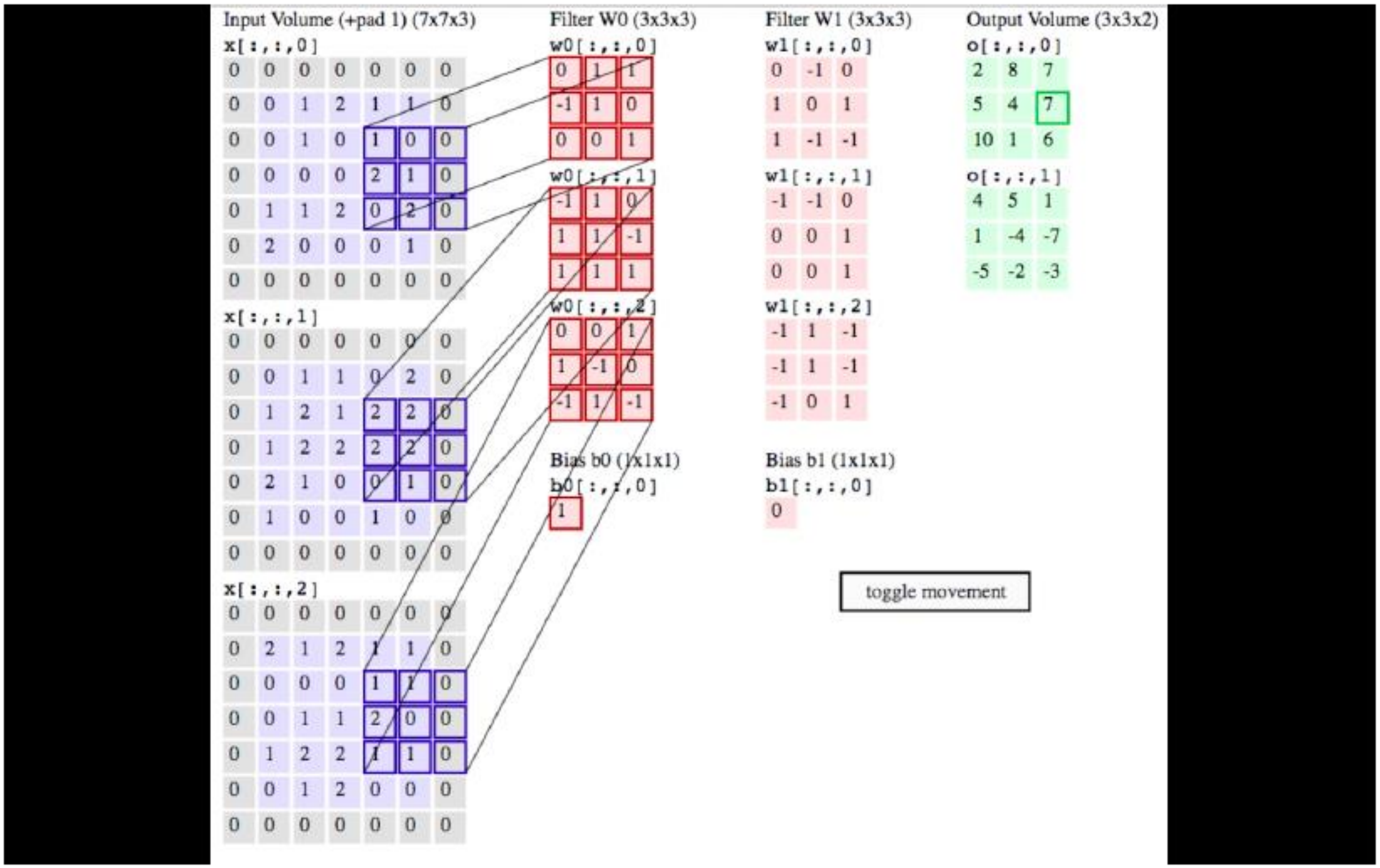
- Method -



Demo courtesy to Andrej Karpathy [Karpathy, 2017]

Convolutional Layer [1]

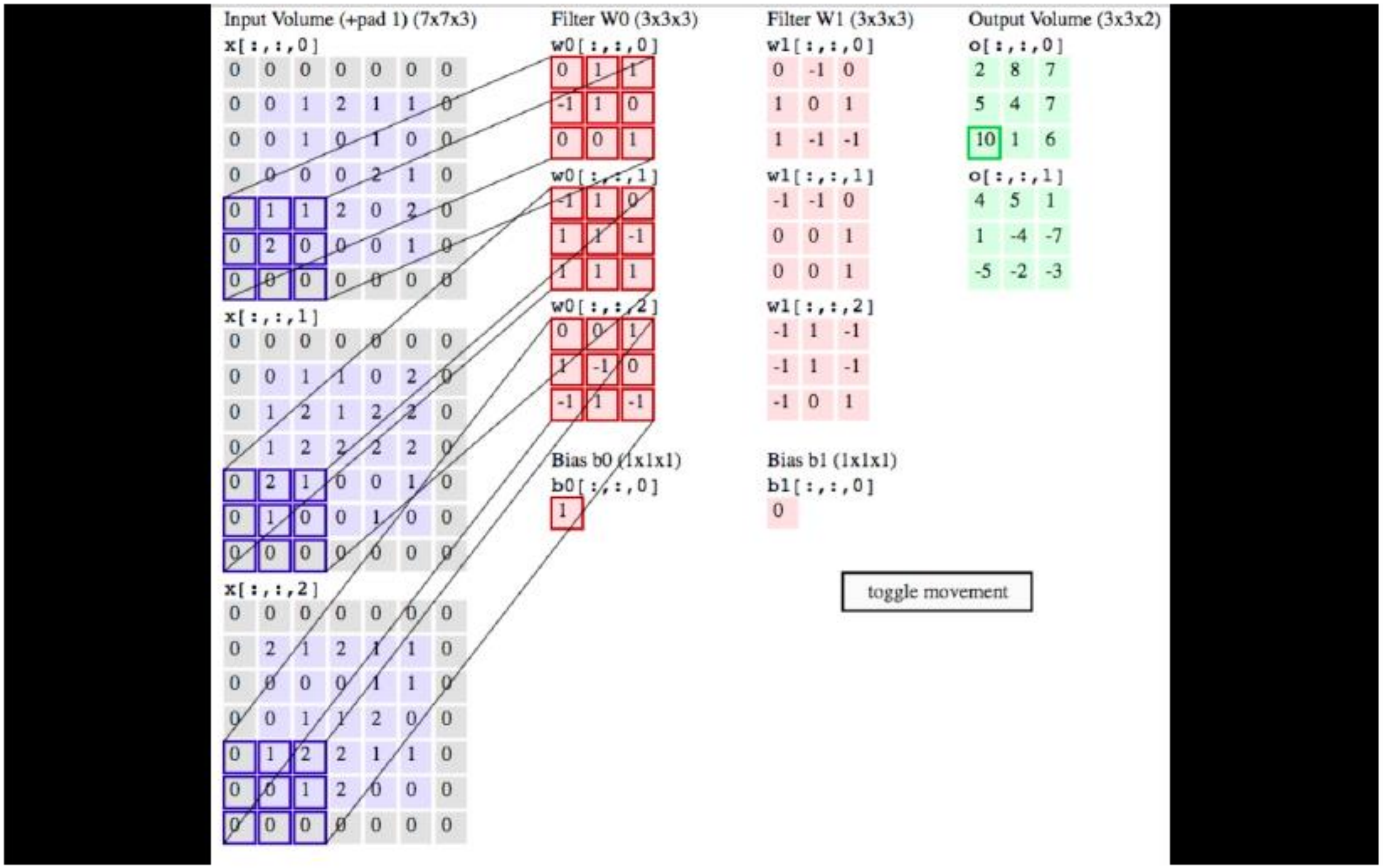
- Method -



Demo courtesy to Andrej Karpathy [Karpathy, 2017]

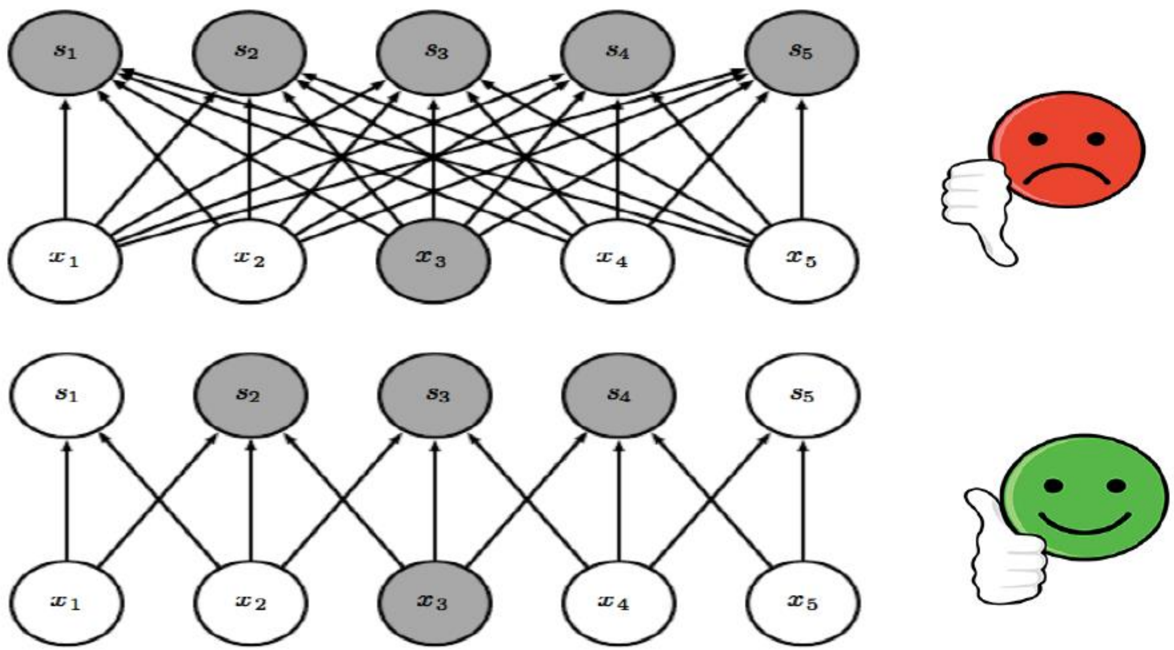
Convolutional Layer [1]

- Method -

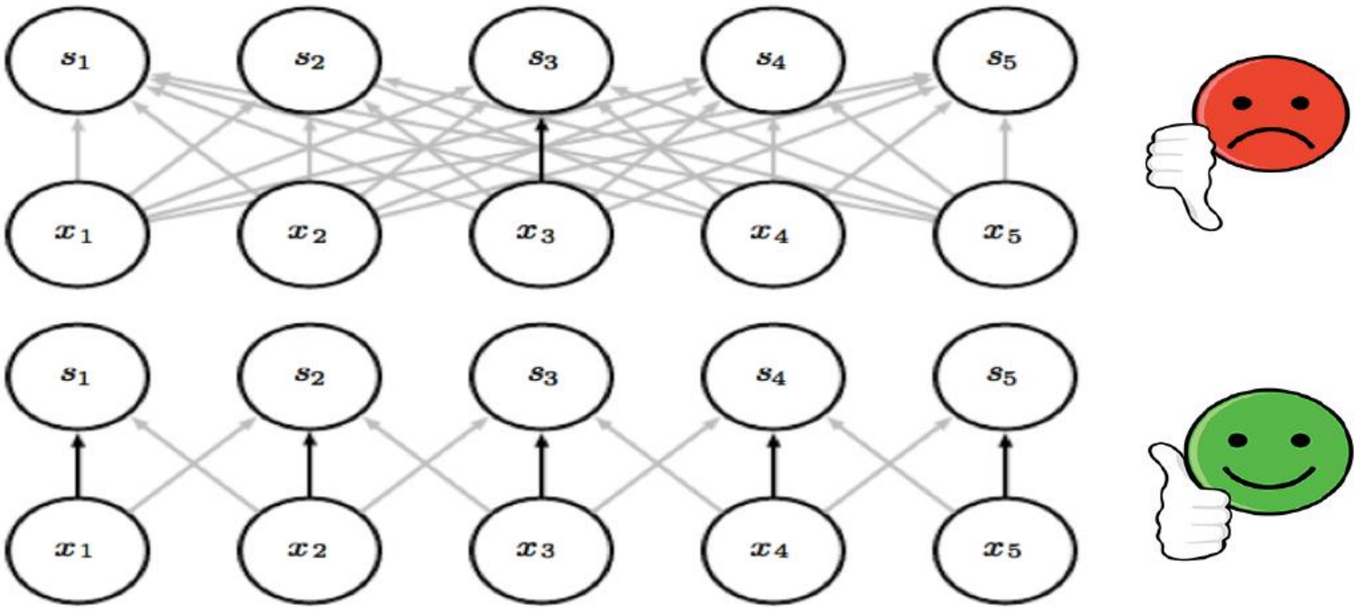


Demo courtesy to Andrej Karpathy [Karpathy, 2017]

3 reasons why convolution is cool



s_1 s_2 s_3 s_4 s_5
x_1 x_2 x_3 x_4 x_5
s_1 s_2 s_3 s_4 s_5
x_1 x_2 x_3 x_4 x_5



s_1 s_2 s_3 s_4 s_5
x_1 x_2 x_3 x_4 x_5
s_1 s_2 s_3 s_4 s_5
x_1 x_2 x_3 x_4 x_5

Reason 3 : Equivariant Representations

When the input changes -> output changes in the same way

Feature Visualization in CNNs
- Method -

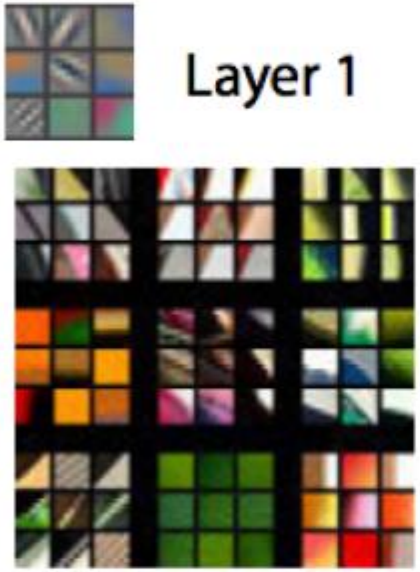


Figure: Visualization of filters and the inputs for which they obtain the highest activation (result of dot product operation)

Fig. courtesy to [Zeiler and Fergus, 2014]

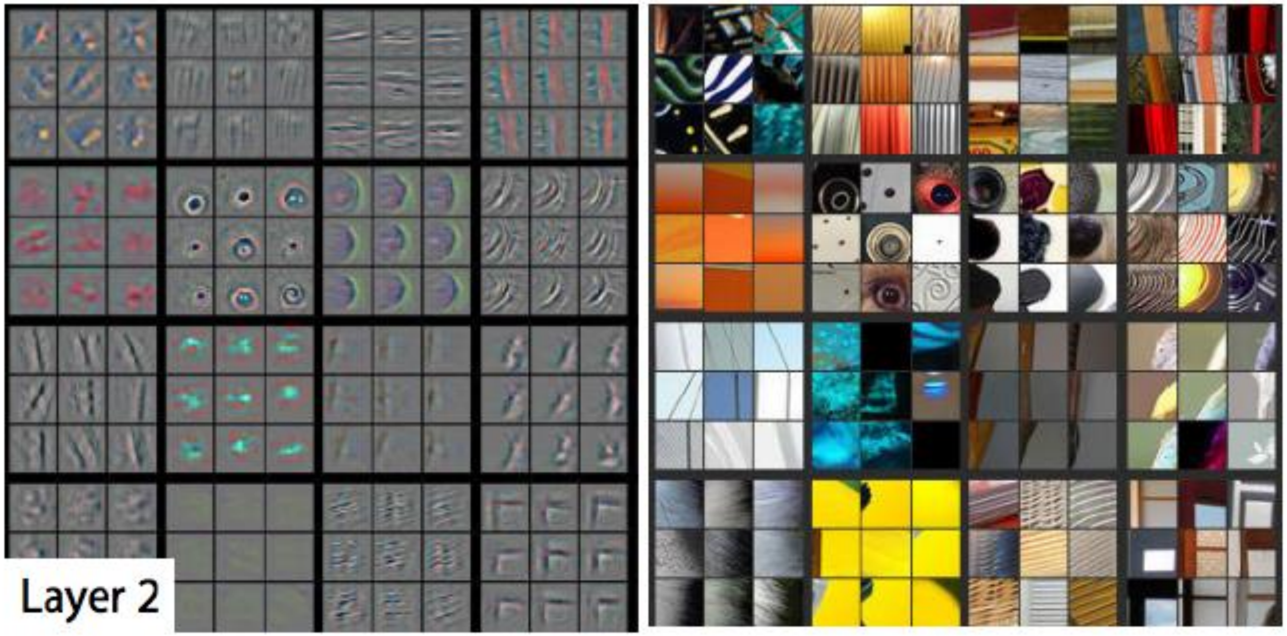


Figure: Visualization of filters and the inputs for which they obtain the highest activation (result of dot product operation)

Fig. courtesy to [Zeiler and Fergus, 2014]

References

- Coming soon!