# Deep Learning for images

# Lenses of Deep Learning

# Overall architecture of CNN

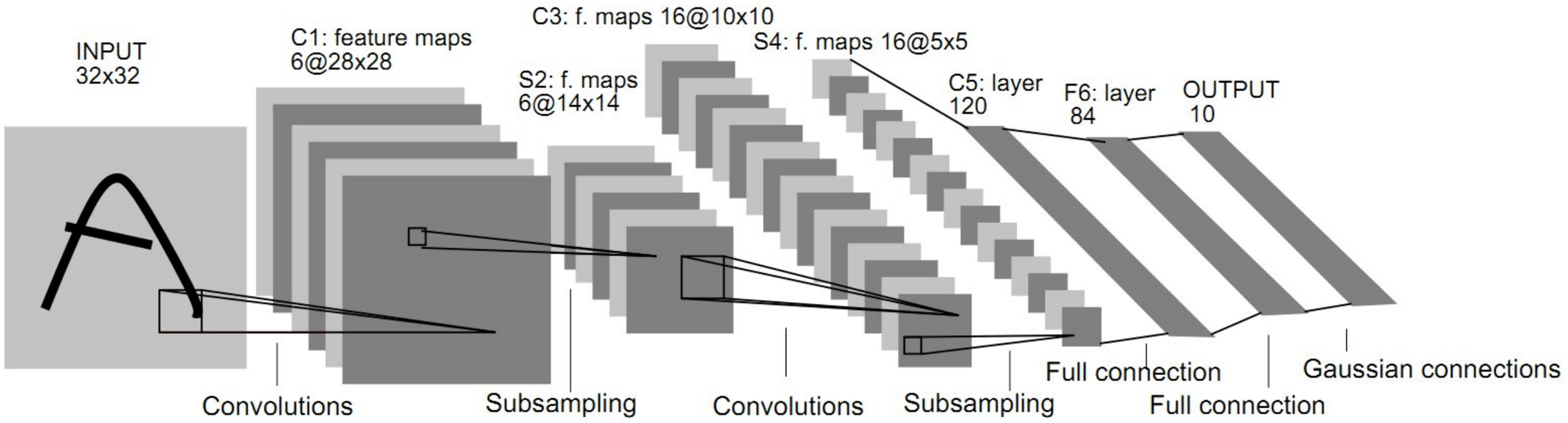
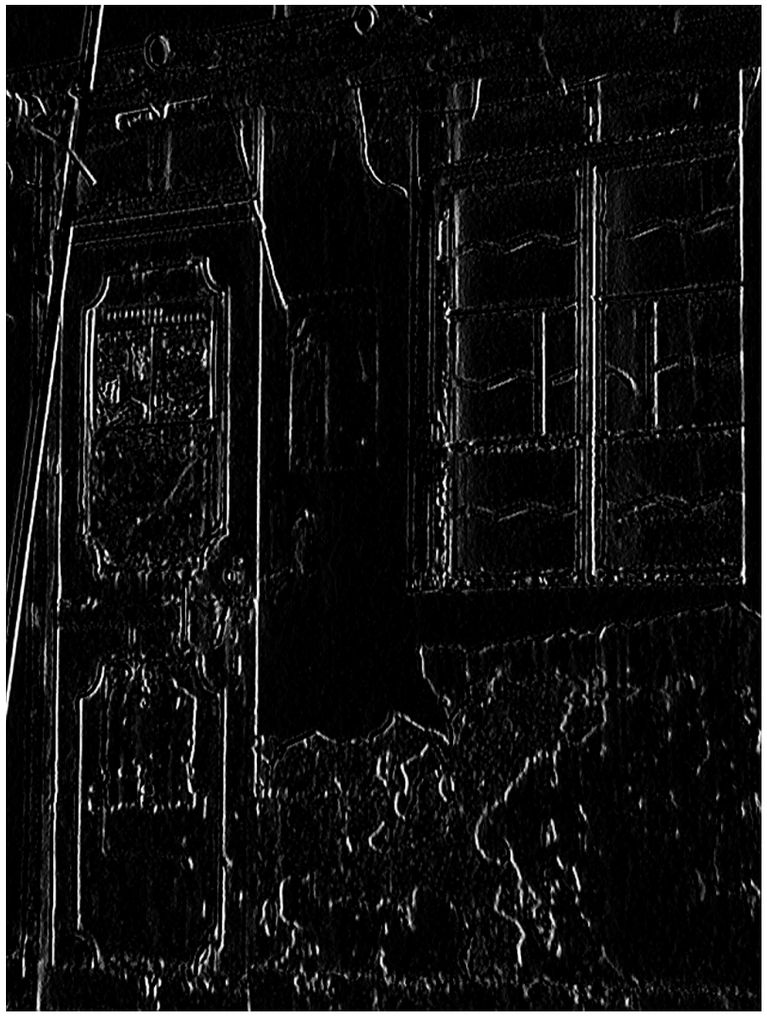
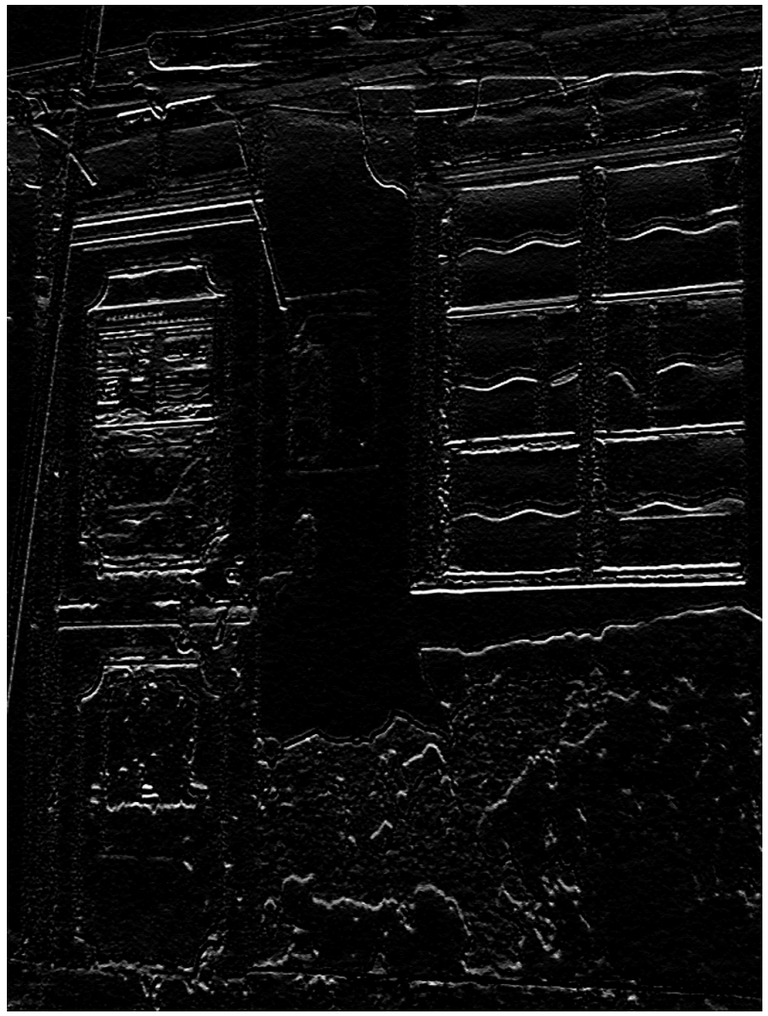



Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

https://world4jason.gitbooks.io/research-log/content/deepLearning/CNN/Model%20&%20ImgNet/lenet

# First we learn about
# Image filtering

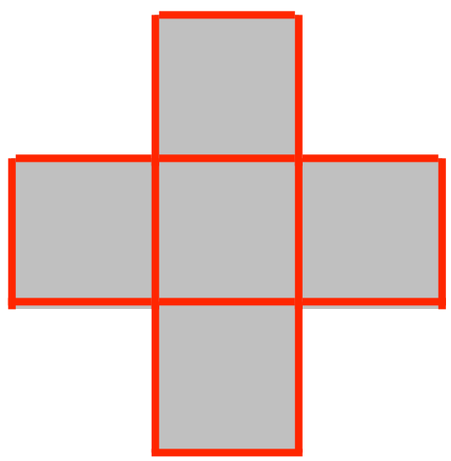




# The Sliding window - image filtering

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

| | | | | | | |
|---|---|---|---|---|---|---|
| 13 | 16 | 18 | 10 | 18 | 28 | 26 |
| 21 | 37 | 25 | 26 | 16 | 35 | 33 |
| 27 | 28 | 35 | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

our lens 

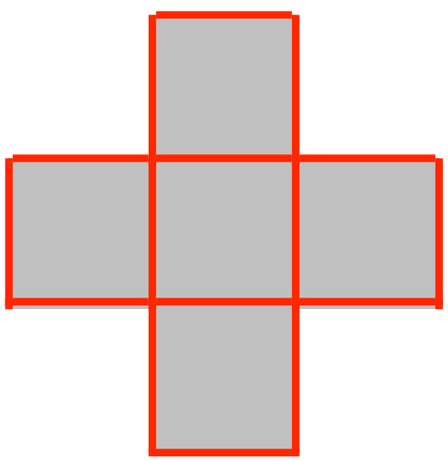
operations = sum of pixels

# The Sliding window - image filtering

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

| | | | | | | |
|---|---|---|---|---|---|---|
| 6 | 8 | 9 | 4 | 8 | 9 | 9 |
| 8 | 9 | 9 | 9 | 7 | 9 | 9 |
| 9 | 9 | 9 | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

our lens



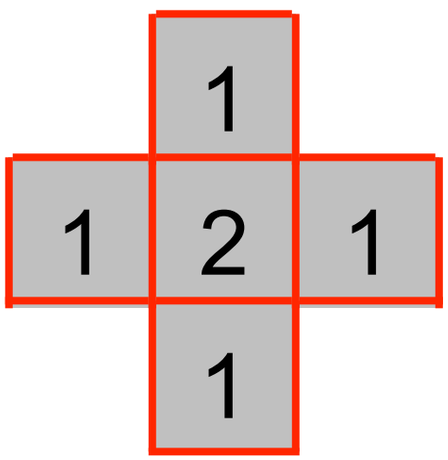
operations = max of pixels

# The Sliding window - image filtering

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

| | | | | | | |
|---|---|---|---|---|---|---|
| 9 | 9 | 5 | 5 | 8 | 7 | 3 |
| 11 | 10 | 9 | 8 | 12 | 10 | 4 |
| 6 | 5 | 8 | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

our lens



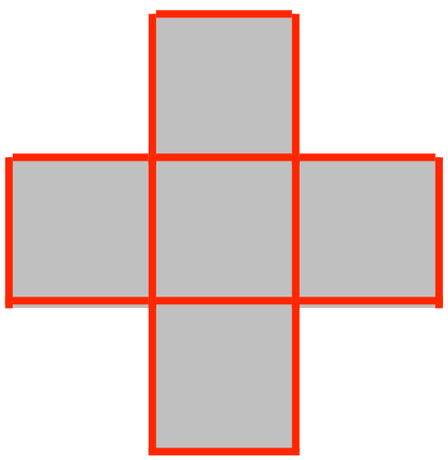

operations = weighted sum

# Dilation as an image filter

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |

our lens



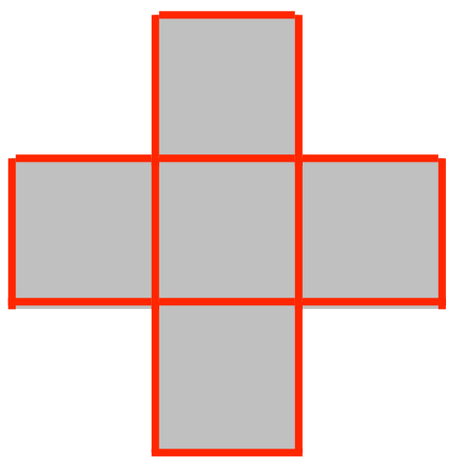
operations = max pixel

# Mean filter

1 = white pixel, 0 = black pixel

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 1/4 | 0 | 0 | 0 | 0 |
| 0 | 2/5 | 2/5 | 2/5 | 0 | 0 | 0 |
| 1/4 | 2/5 | 1 | 3/5 | 1/5 | 0 | 0 |
| 0 | 2/5 | 2/5 | 4/5 | 1/5 | 0 | 0 |
| 0 | 0 | 2/5 | 3/5 | 1/5 | 0 | 0 |
| 0 | 0 | 1/5 | 2/5 | 1/5 | 0 | 0 |
| 0 | 0 | 0 | 1/5 | 0 | 0 | 0 |

our lens 

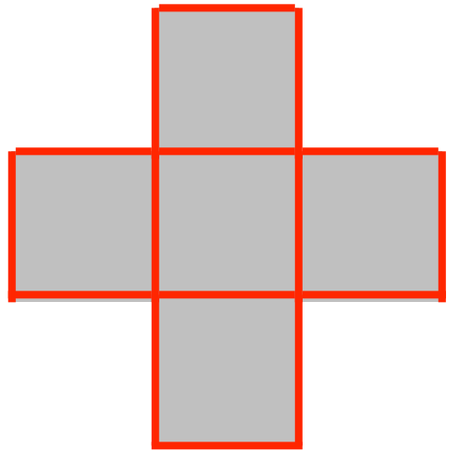
operations = mean

# Mean filter

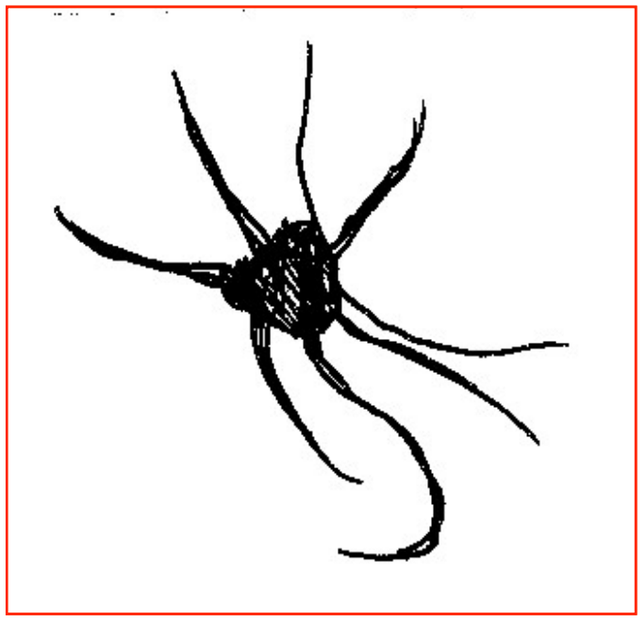
| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 1/4 | 0 | 0 | 0 | 0 |
| 0 | 2/5 | 2/5 | 2/5 | 0 | 0 | 0 |
| 1/4 | 2/5 | 1 | 3/5 | 1/5 | 0 | 0 |
| 0 | 2/5 | 2/5 | 4/5 | 1/5 | 0 | 0 |
| 0 | 0 | 2/5 | 3/5 | 1/5 | 0 | 0 |
| 0 | 0 | 1/5 | 2/5 | 1/5 | 0 | 0 |
| 0 | 0 | 0 | 1/5 | 0 | 0 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0.16 | 0.36 | 0.27 | 0 | 0 | 0 |
| 0.27 | 0.24 | 0.49 | 0.28 | 0.12 | 0 | 0 |
| 0.36 | 0.49 | 0.52 | 0.6 | 0.2 | 0.04 | 0 |
| 0.15 | 0.24 | 0.4 | 0.52 | 0.28 | 0.04 | 0 |
| 0 | 0.16 | 0.32 | 0.48 | 0.24 | 0.04 | 0 |
| 0 | 0.04 | 0.2 | 0.32 | 0.16 | 0.04 | 0 |
| 0 | 0 | 0.1 | 0.15 | 0.1 | 0 | 0 |

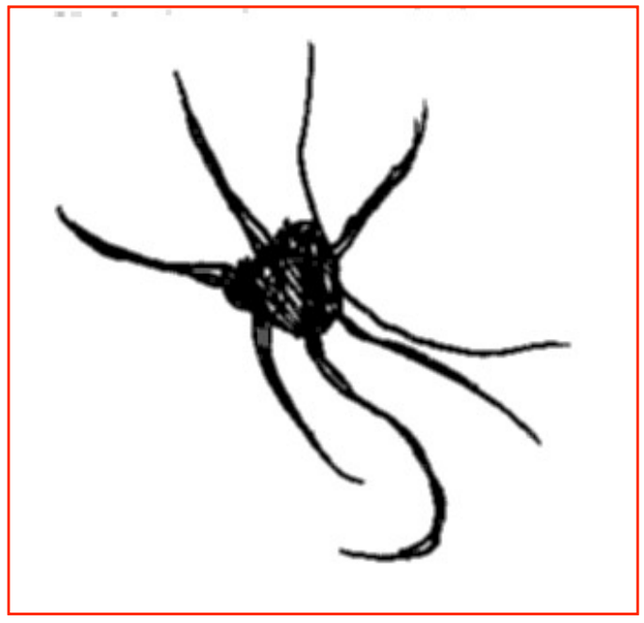
our lens



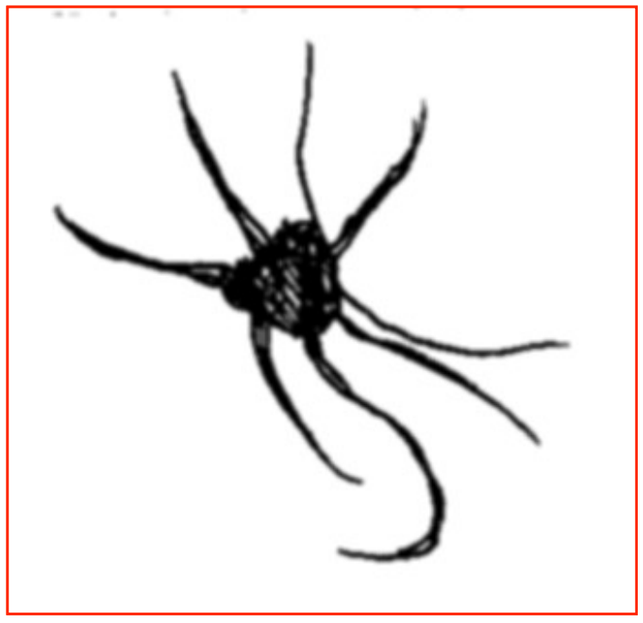
operations = mean



original image
344 x 332 pixels



mean 1 time



mean 2 times

mean 3 times

mean 10 times

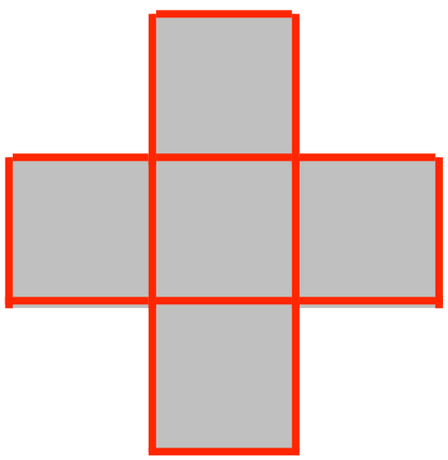
mean 1 time with radius 20

# The Sliding window - image filtering

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

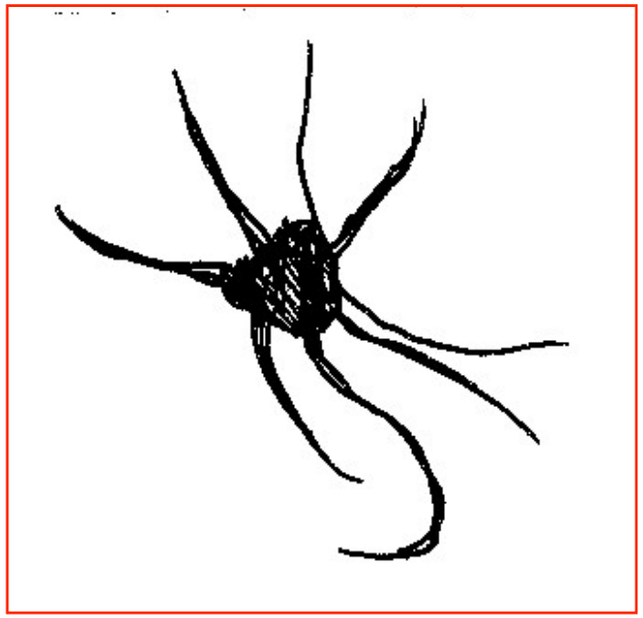
| | | | | | | |
|---|---|---|---|---|---|---|
| 2 | 2 | 1 | 1 | 1 | 1 | 1 |
| 2 | 2 | 2 | 1 | 2 | 2 | 1 |
| 1 | 1 | 1 | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

our lens

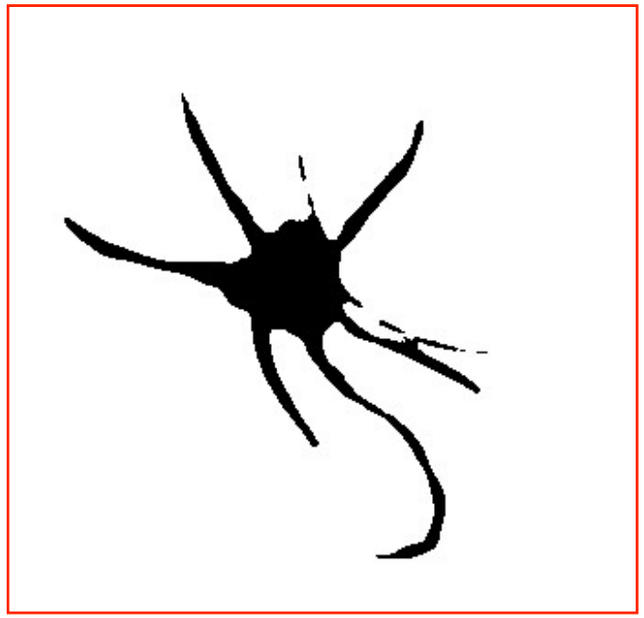


operations = median filter

# Median filter



original image
344 x 332 pixels



median filter
radius 4

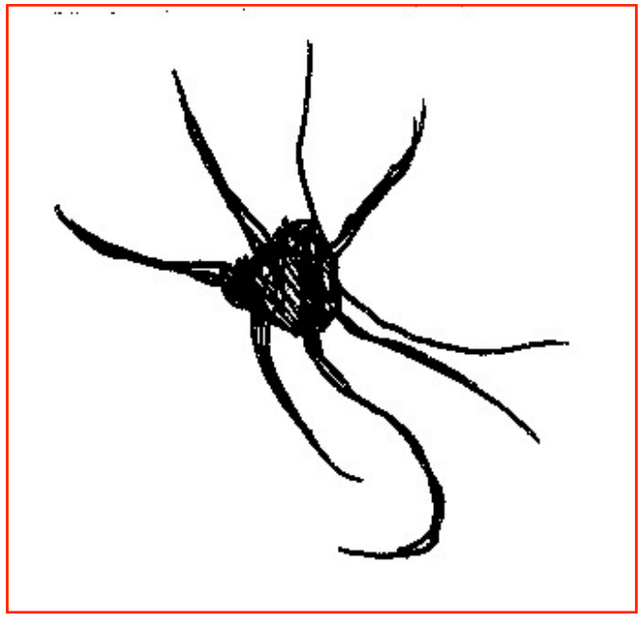
mean filter
radius 4

# Gaussian filter

## A weighted window

| | | | | |
|---|---|---|---|---|
| 0.0075 | 0.01 | 0.02 | 0.01 | 0.0075 |
| 0.01 | 0.04 | 0.1 | 0.04 | 0.01 |
| 0.02 | 0.1 | 0.25 | 0.1 | 0.02 |
| 0.01 | 0.04 | 0.1 | 0.04 | 0.01 |
| 0.0075 | 0.01 | 0.02 | 0.01 | 0.0057 |

Put in weights with a gaussian profile -> a Gaussian filter



original image
344 x 332 pixels

filtered with sigma = 2 pixels

filtered with sigma = 8 pixels

Edge filter

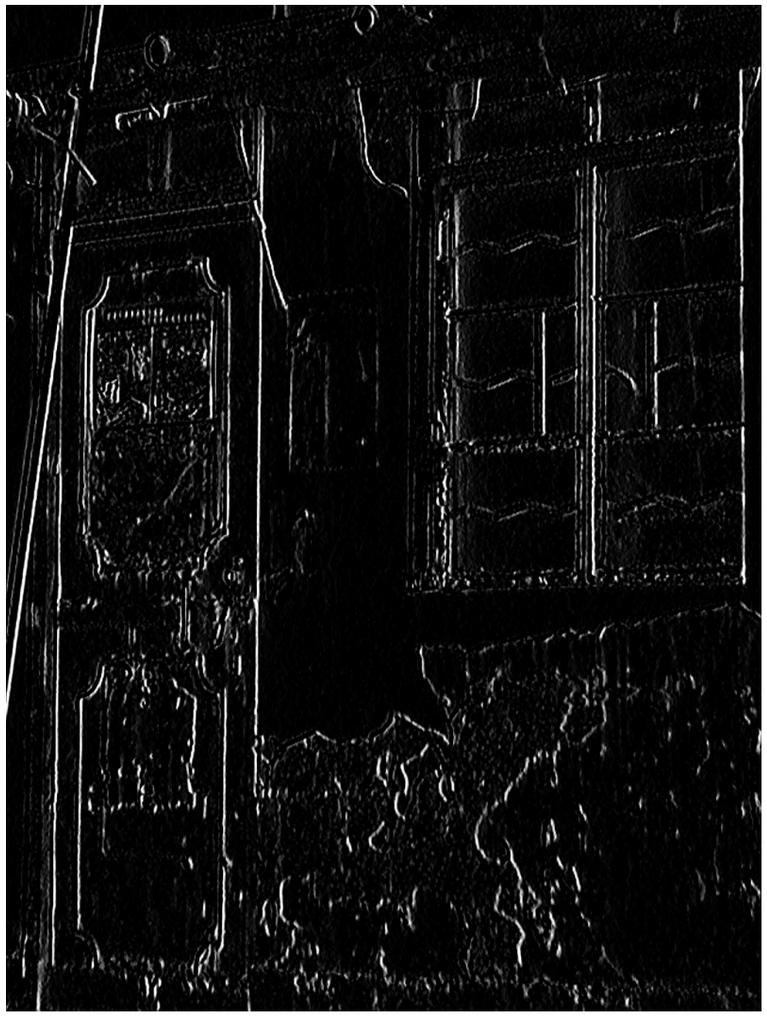
| | | |
|---|---|---|
| -1 | 0 | 1 |
| -1 | 0 | 1 |
| -1 | 0 | 1 |

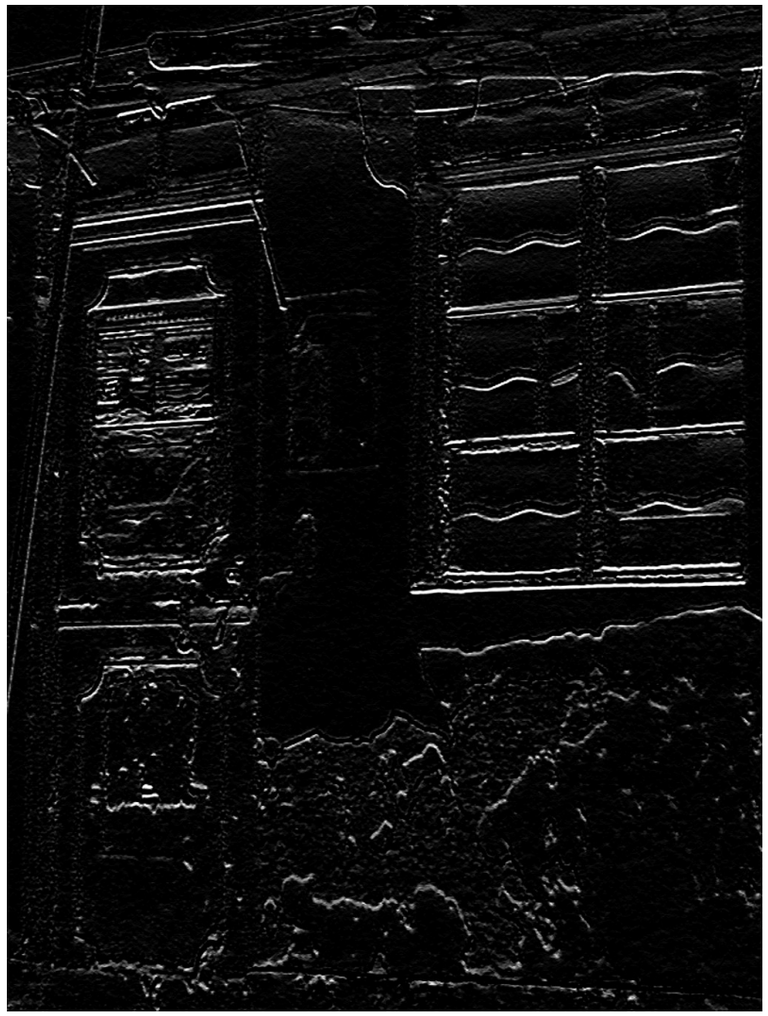
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 |
| 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 |
| 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 |
| 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 |
| 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 |
| 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 |
| 1 | 1 | 1 | 1 | 5 | 5 | 5 | 5 |

| | | | | | |
|---|---|---|---|---|---|
| 0 | 0 | 4 | 4 | 0 | 0 |
| 0 | 0 | 4 | 4 | 0 | 0 |
| 0 | 0 | 4 | 4 | 0 | 0 |
| 0 | 0 | 4 | 4 | 0 | 0 |
| 0 | 0 | 4 | 4 | 0 | 0 |
| 0 | 0 | 4 | 4 | 0 | 0 |

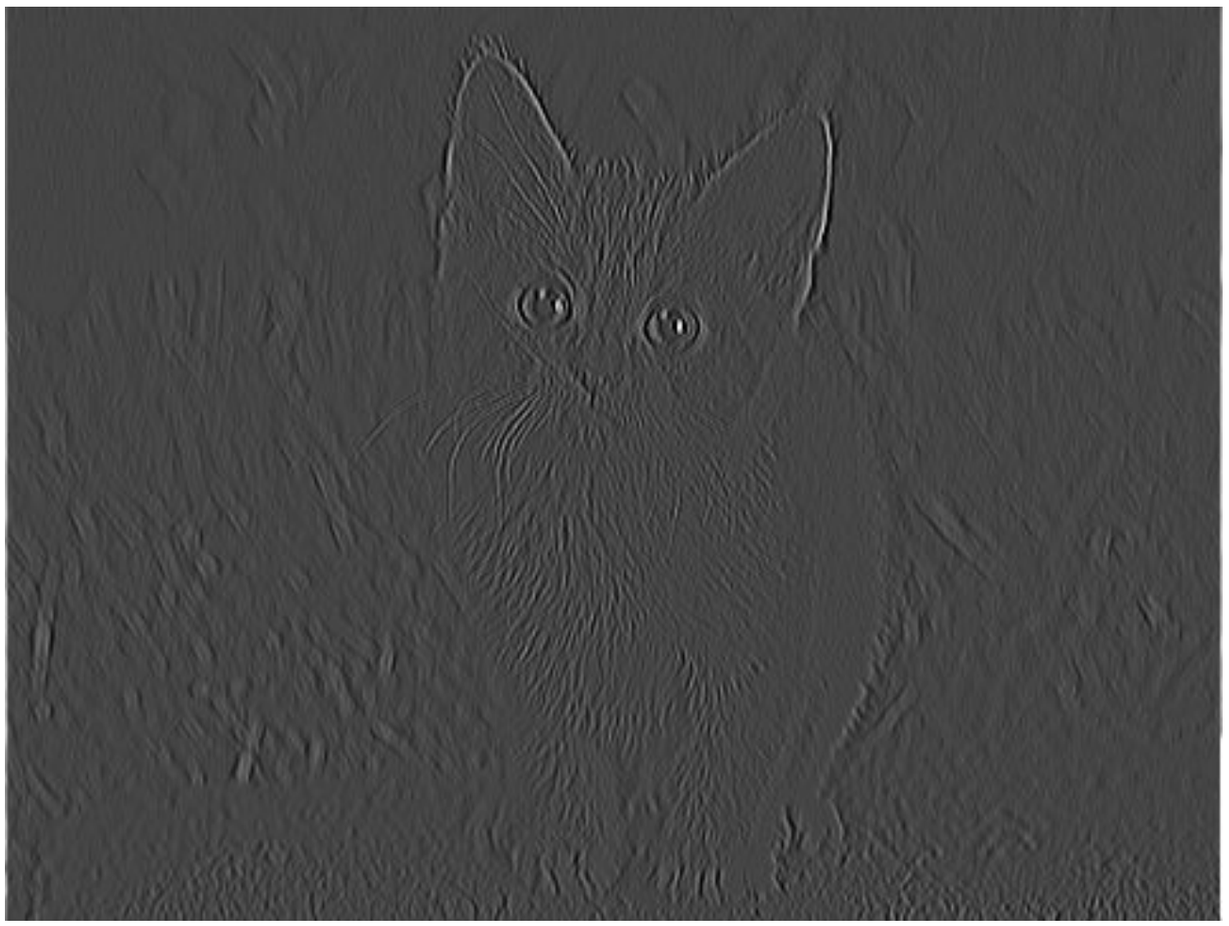
| | | |
|---|---|---|
| -1 | 0 | 1 |
| -1 | 0 | 1 |
| -1 | 0 | 1 |

| | | |
|---|---|---|
| -1 | -1 | -1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

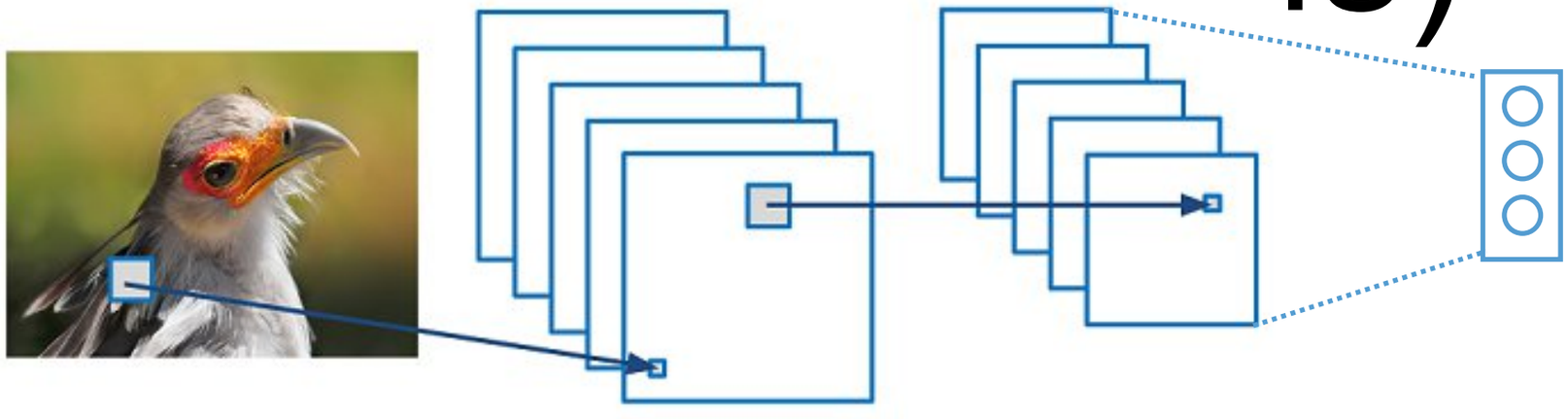




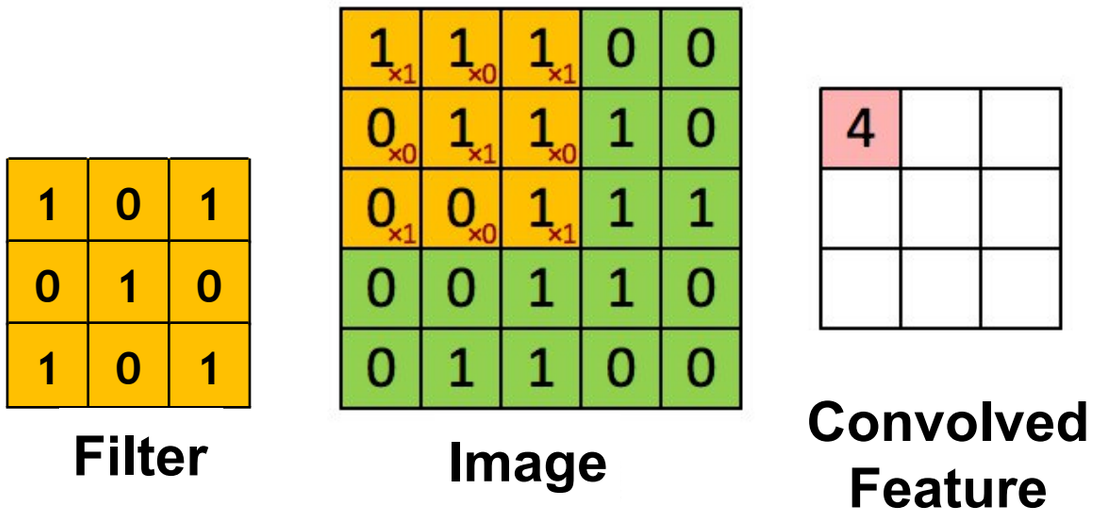
| | | |
|---|---|---|
| -1 | 2 | -1 |
| -1 | 2 | -1 |
| -1 | 2 | -1 |



# Convolutional Neural Networks (CNNs)



**Convolution operation**




**Hyperparameters**

- Filter height
- Filter width
- Stride height
- Stride width
- # of filters
- # of layers

# Effect of Filter Size

3x3 Filter

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

7x7 Filter

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

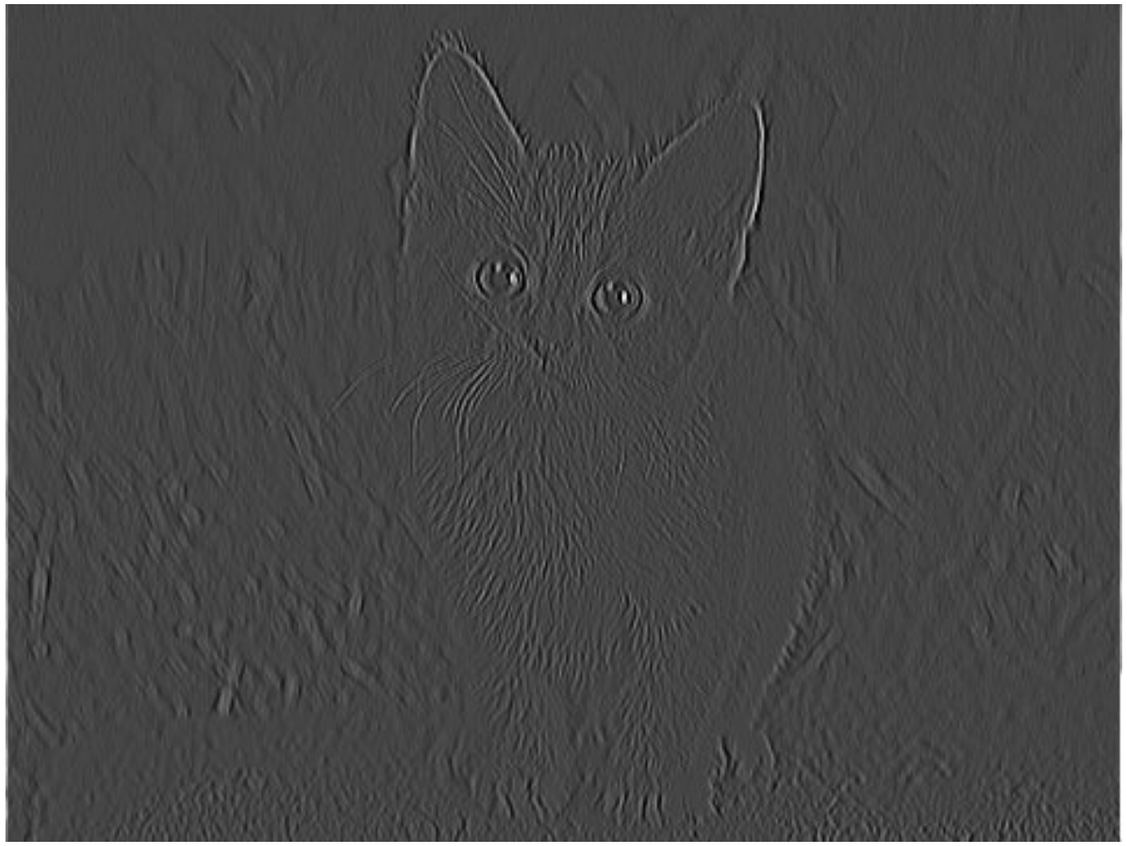
# Learning Filter Weights

Random Weights

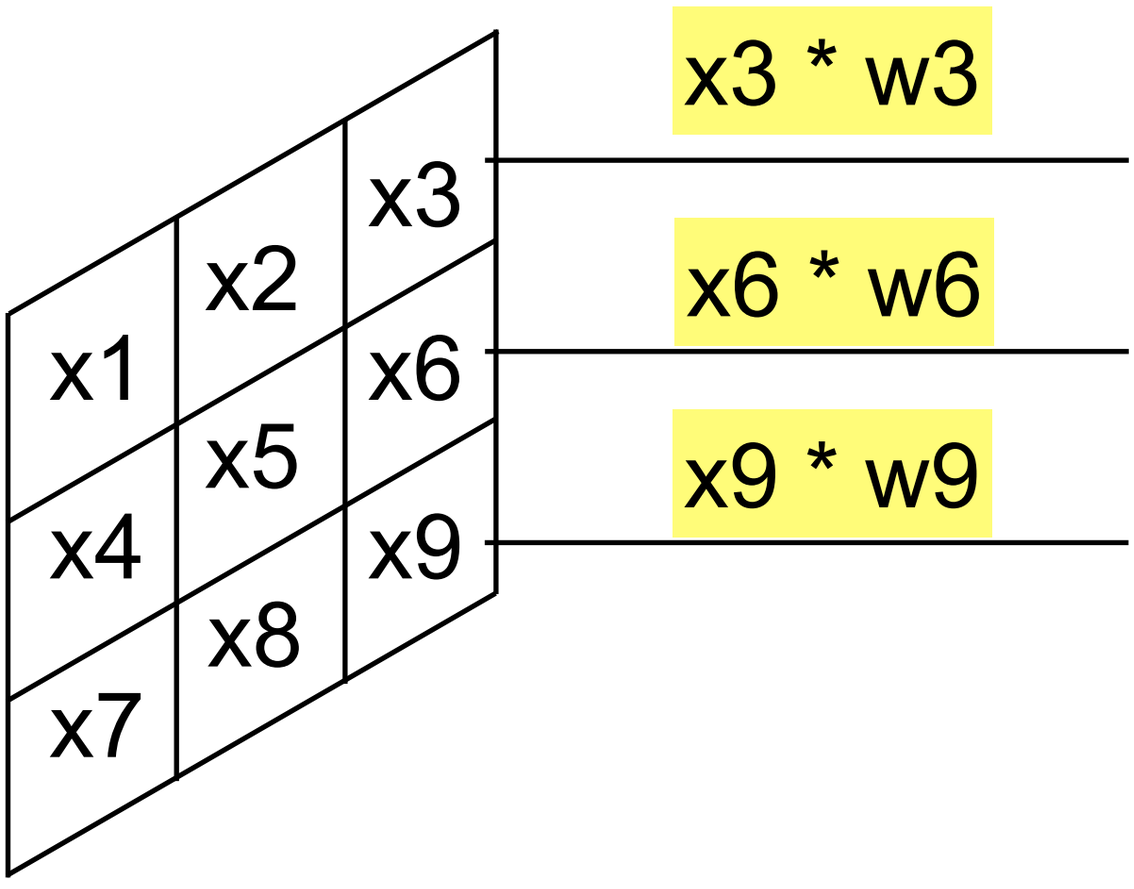
| | | |
|---|---|---|
| 0.5 | 1.1 | -1.6 |
| -1.0 | 0.1 | -0.6 |
| 1.0 | 0.3 | -1.9 |

Learned Weights

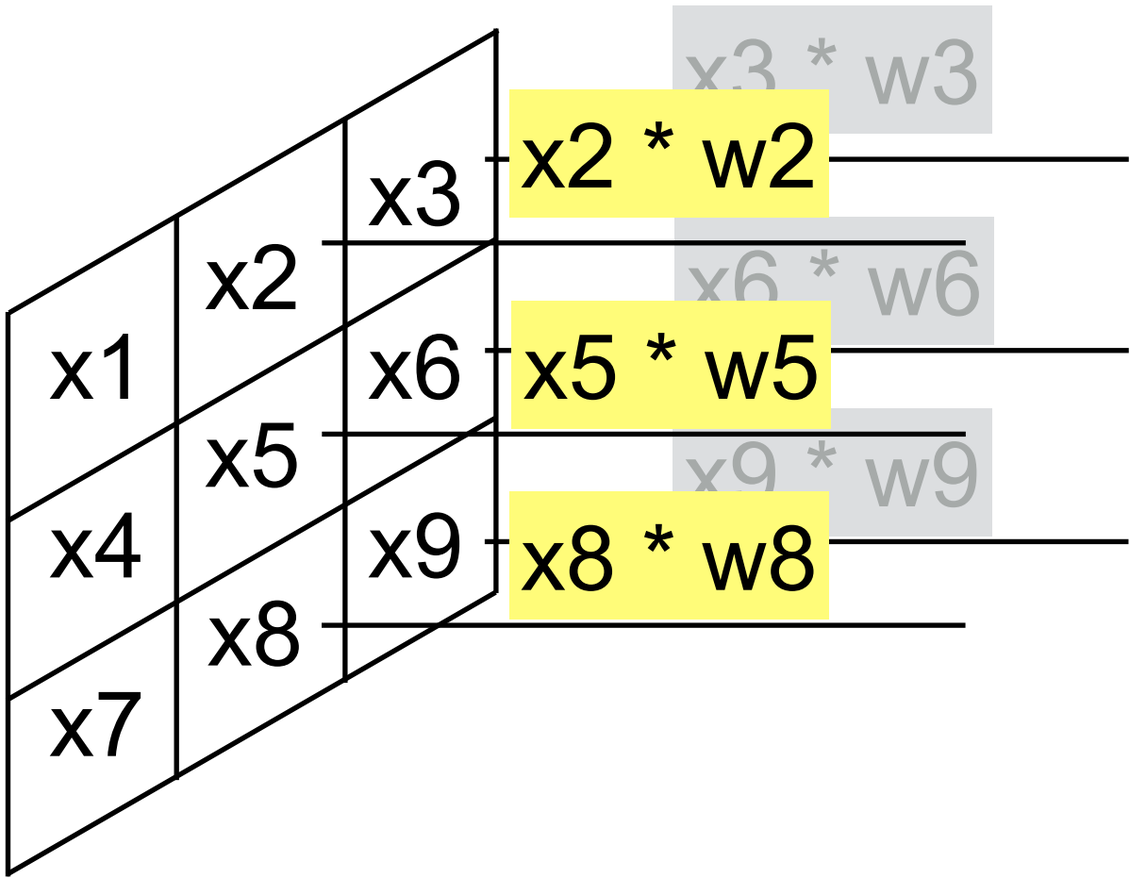
| | | |
|---|---|---|
| -1.0 | 2.0 | -1.0 |
| -1.0 | 2.0 | -1.0 |
| -1.0 | 2.0 | -1.0 |



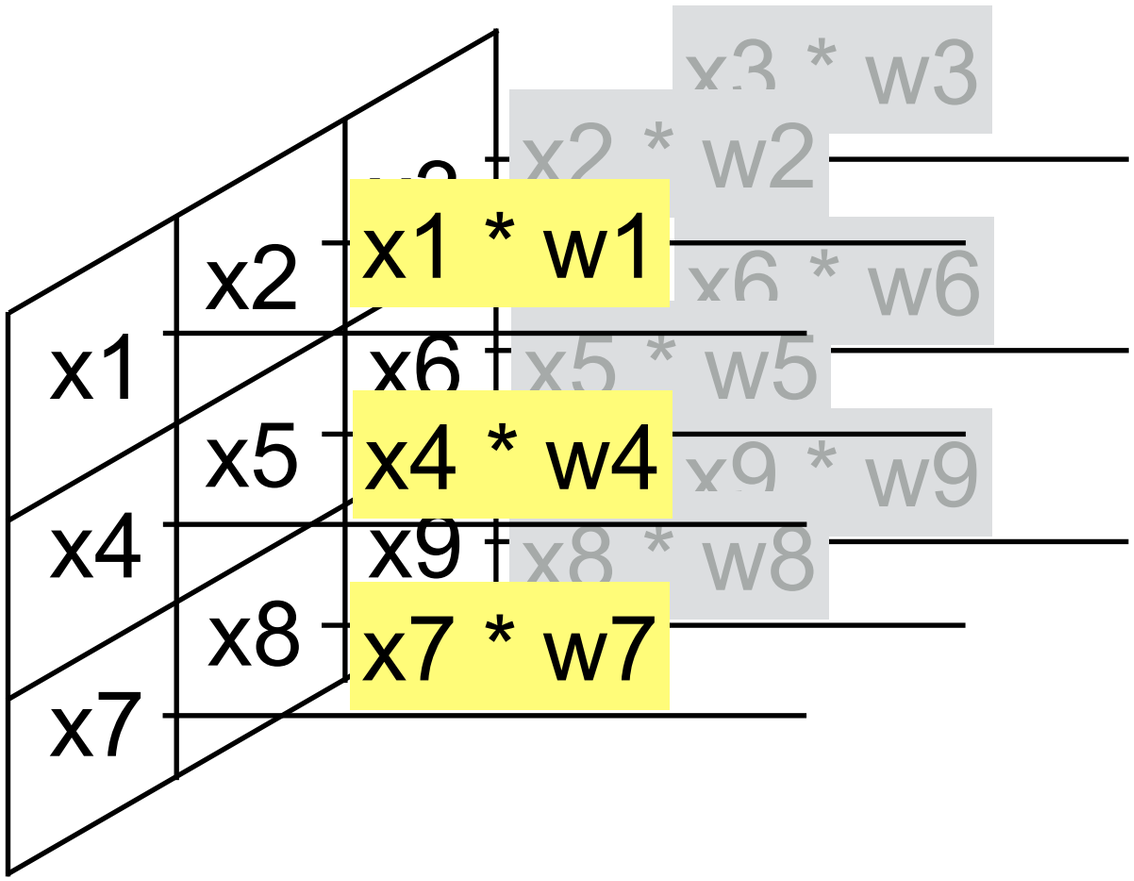
# Simplest convolutional neural network



# Simplest convolutional neural network

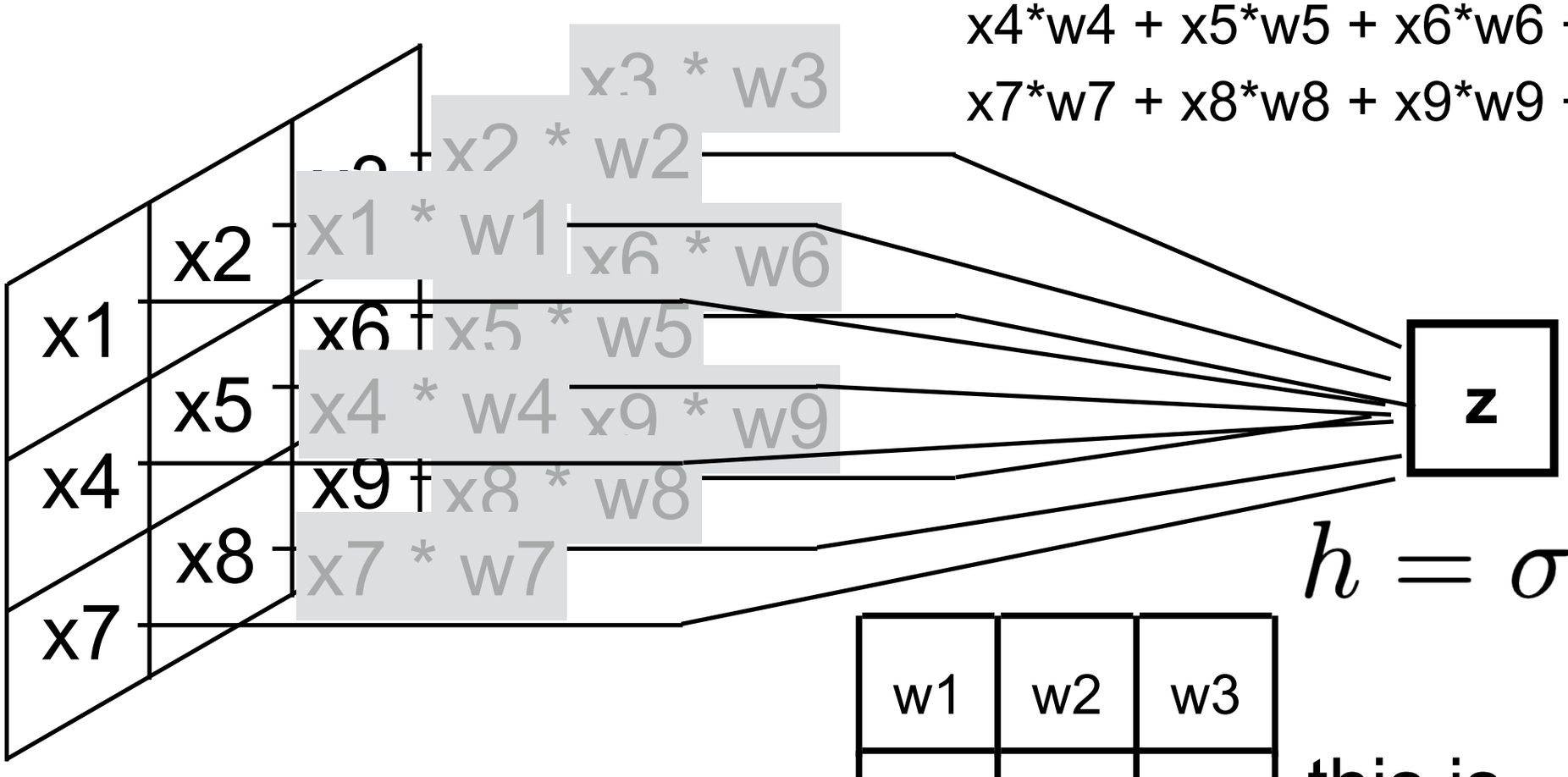


# Simplest convolutional neural network



# Simplest convolutional neural network

x1*w1 + x2*w2 + x3*w3 +
x4*w4 + x5*w5 + x6*w6 +
x7*w7 + x8*w8 + x9*w9 + b = **z**

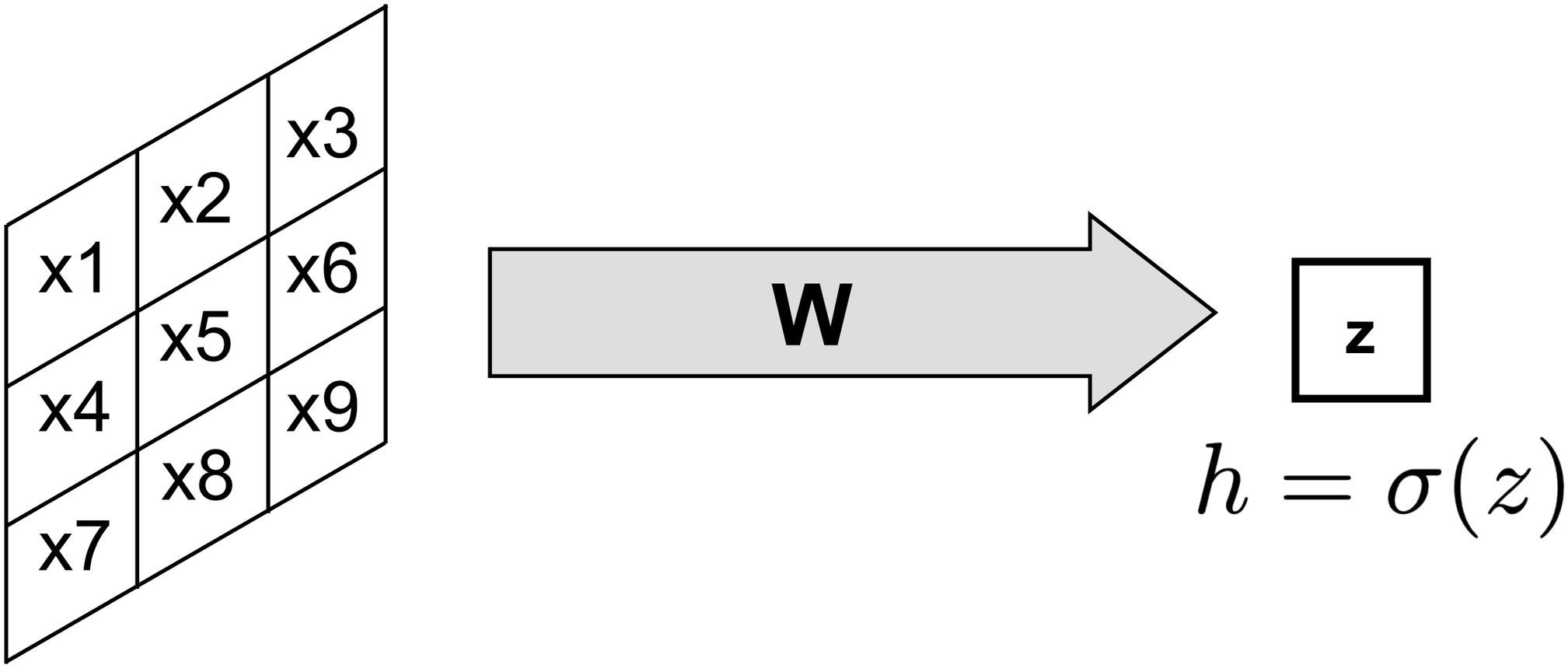



$$h = \sigma(z)$$

| | | |
|---|---|---|
| w1 | w2 | w3 |
| w4 | w5 | w6 |
| w7 | w8 | w9 |

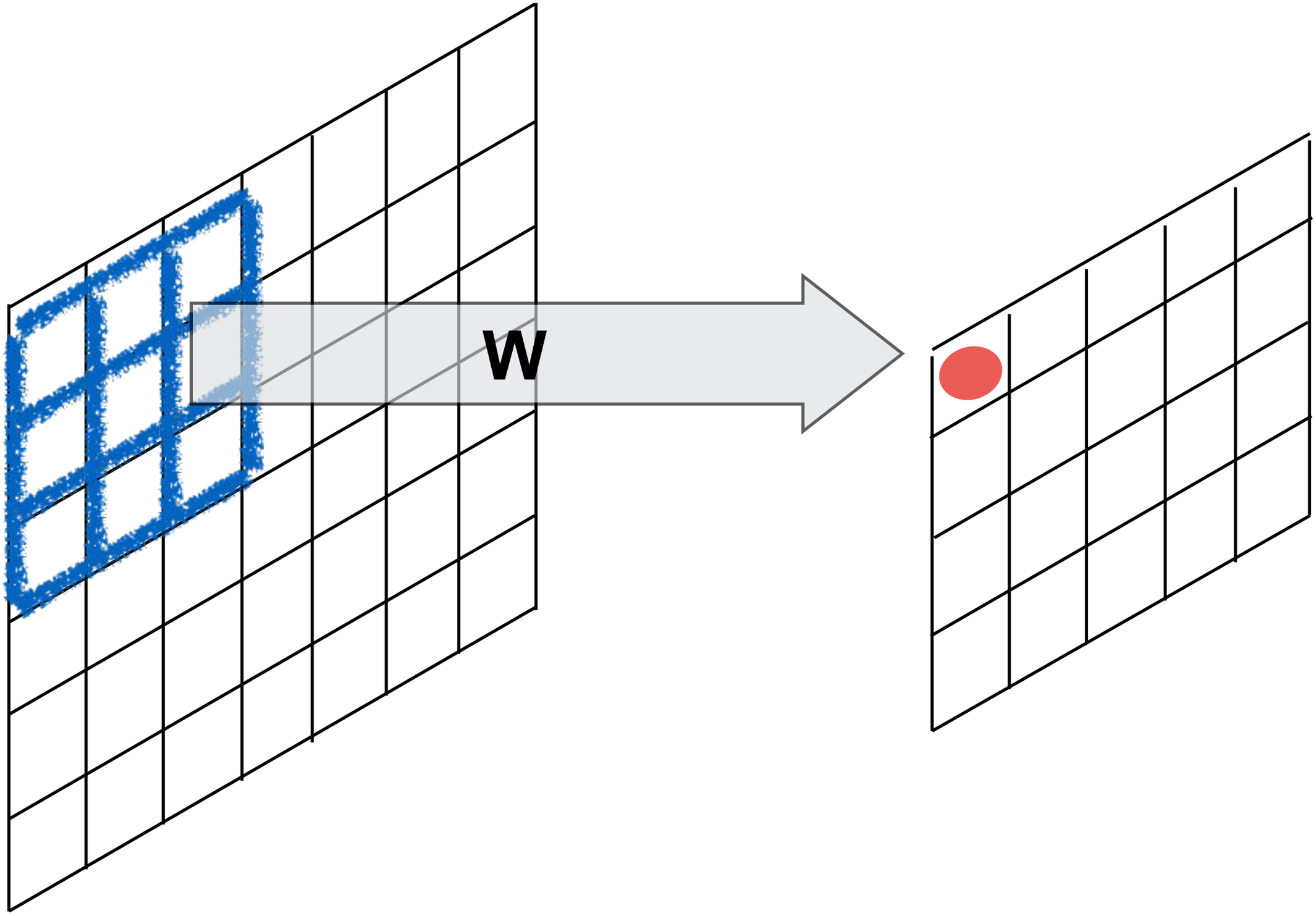
this is
our lens

# Simplest convolutional neural network

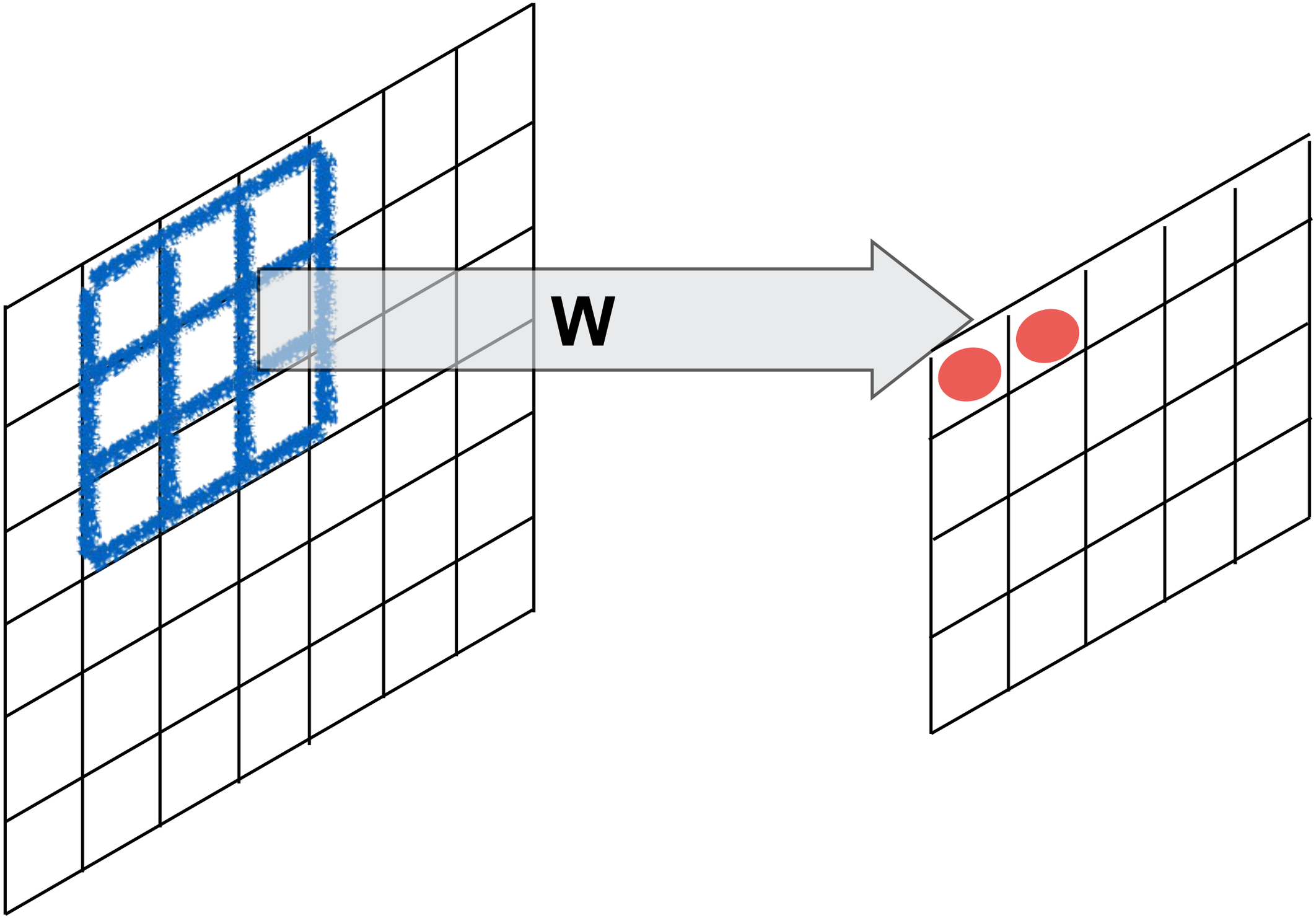



$$h = \sigma(z)$$

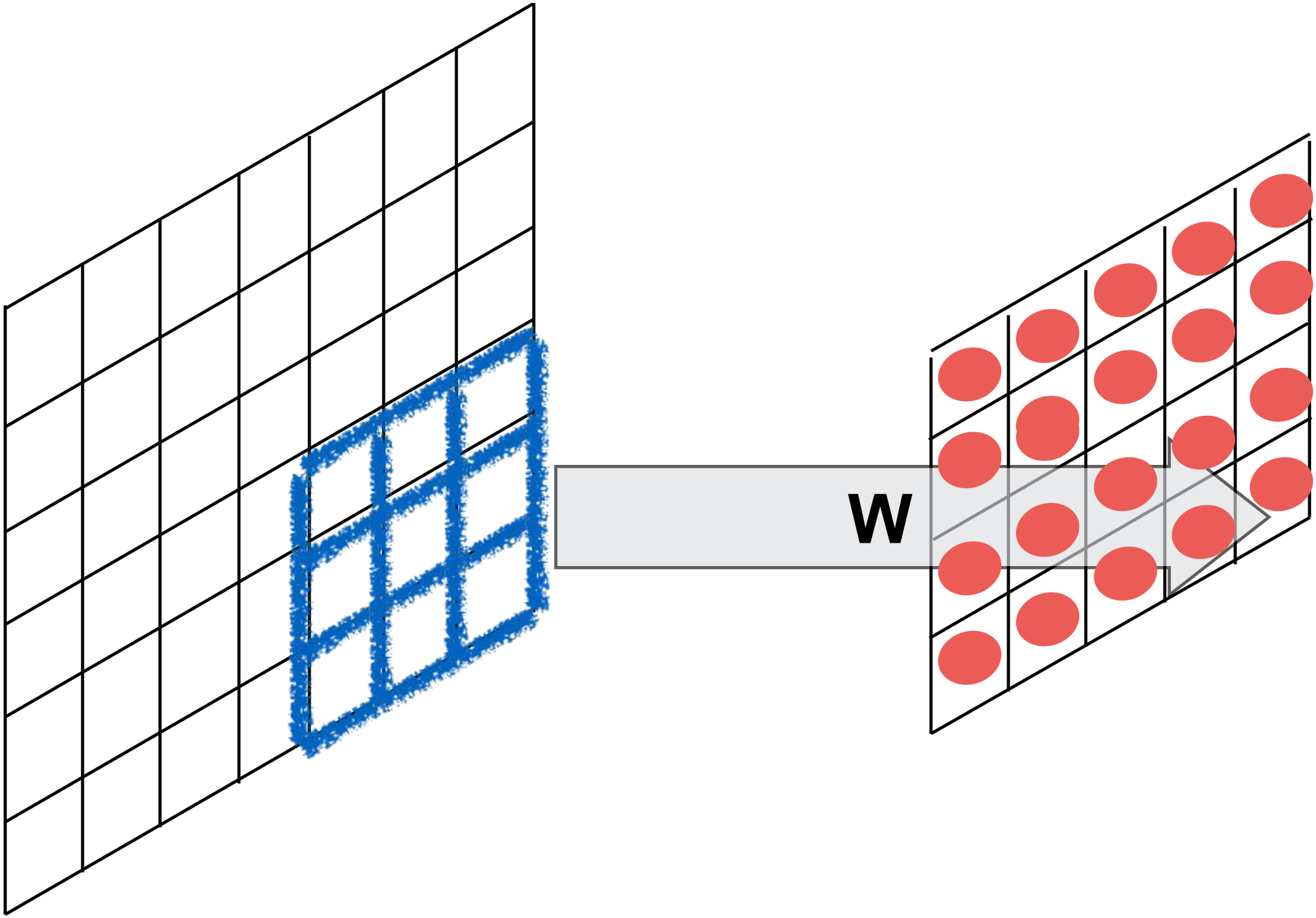
# Simplest convolutional neural network



# Simplest convolutional neural network

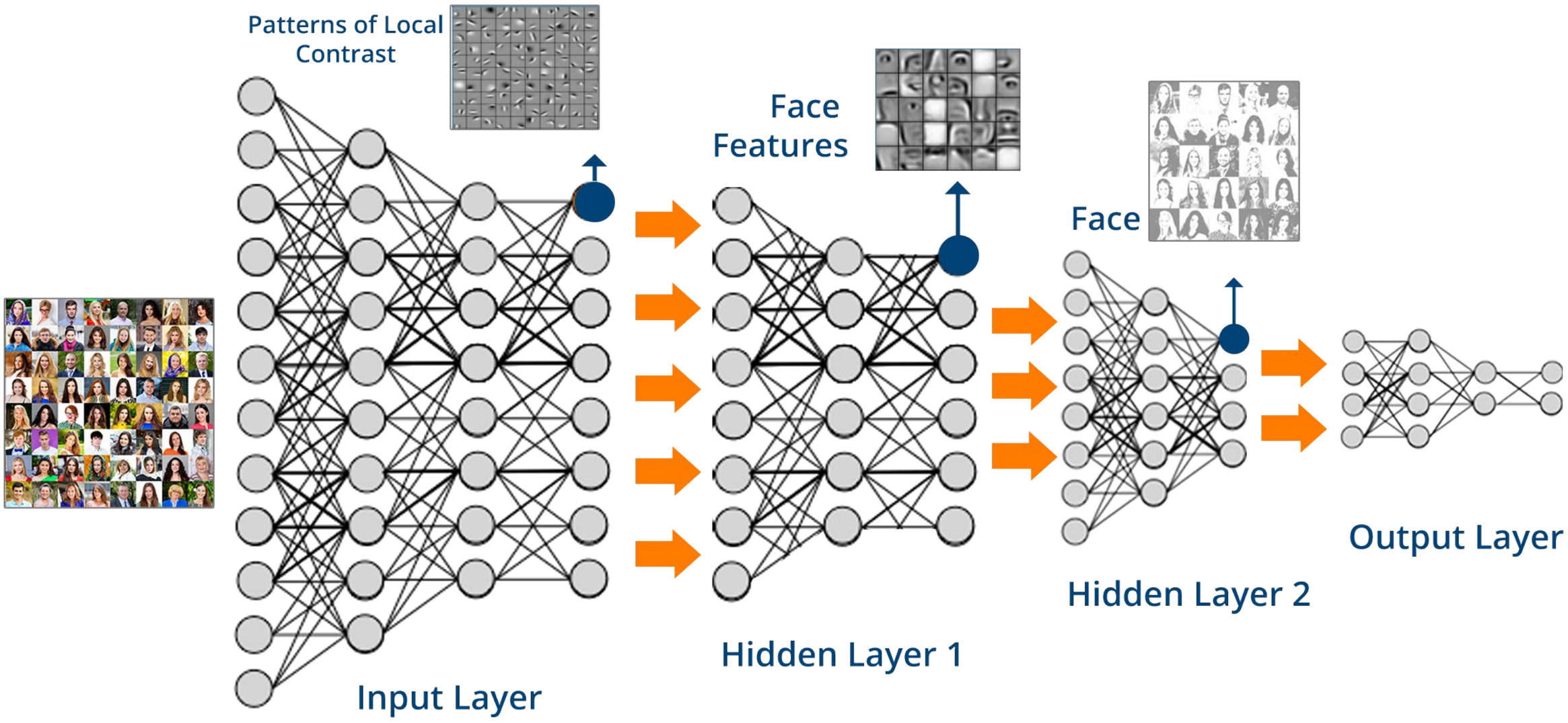


# Simplest convolutional neural network



# What is Deep Learning?

- Multi-level representation learning



# Overall architecture of CNN

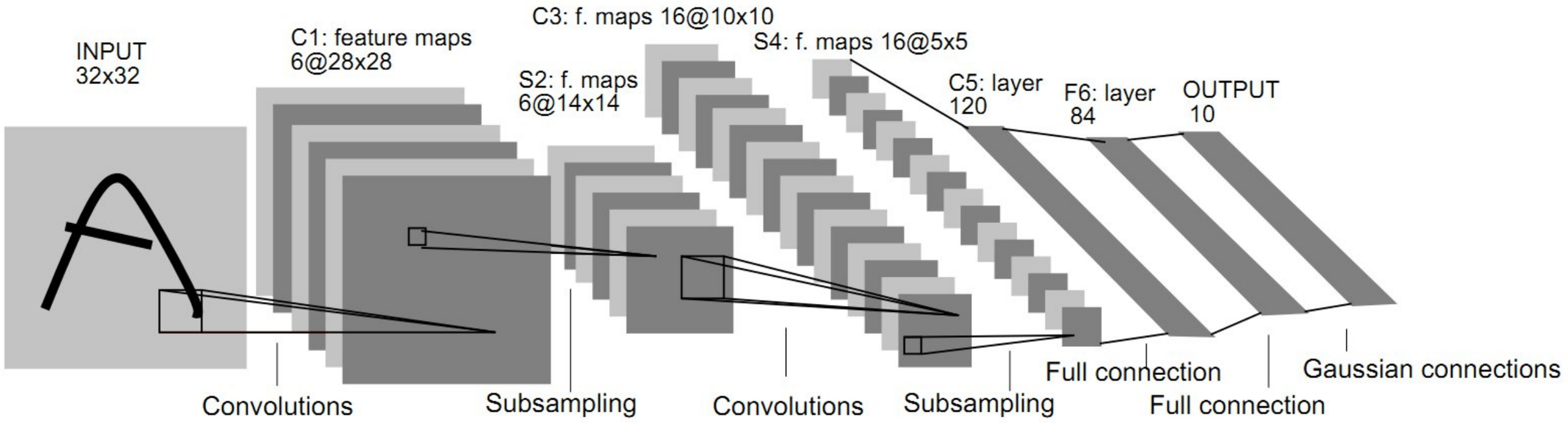
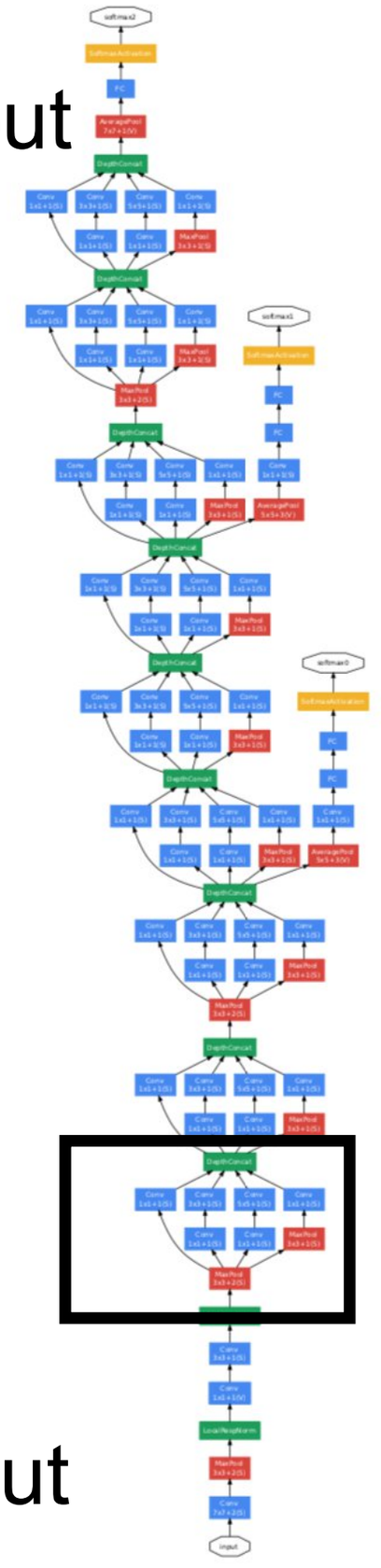



Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

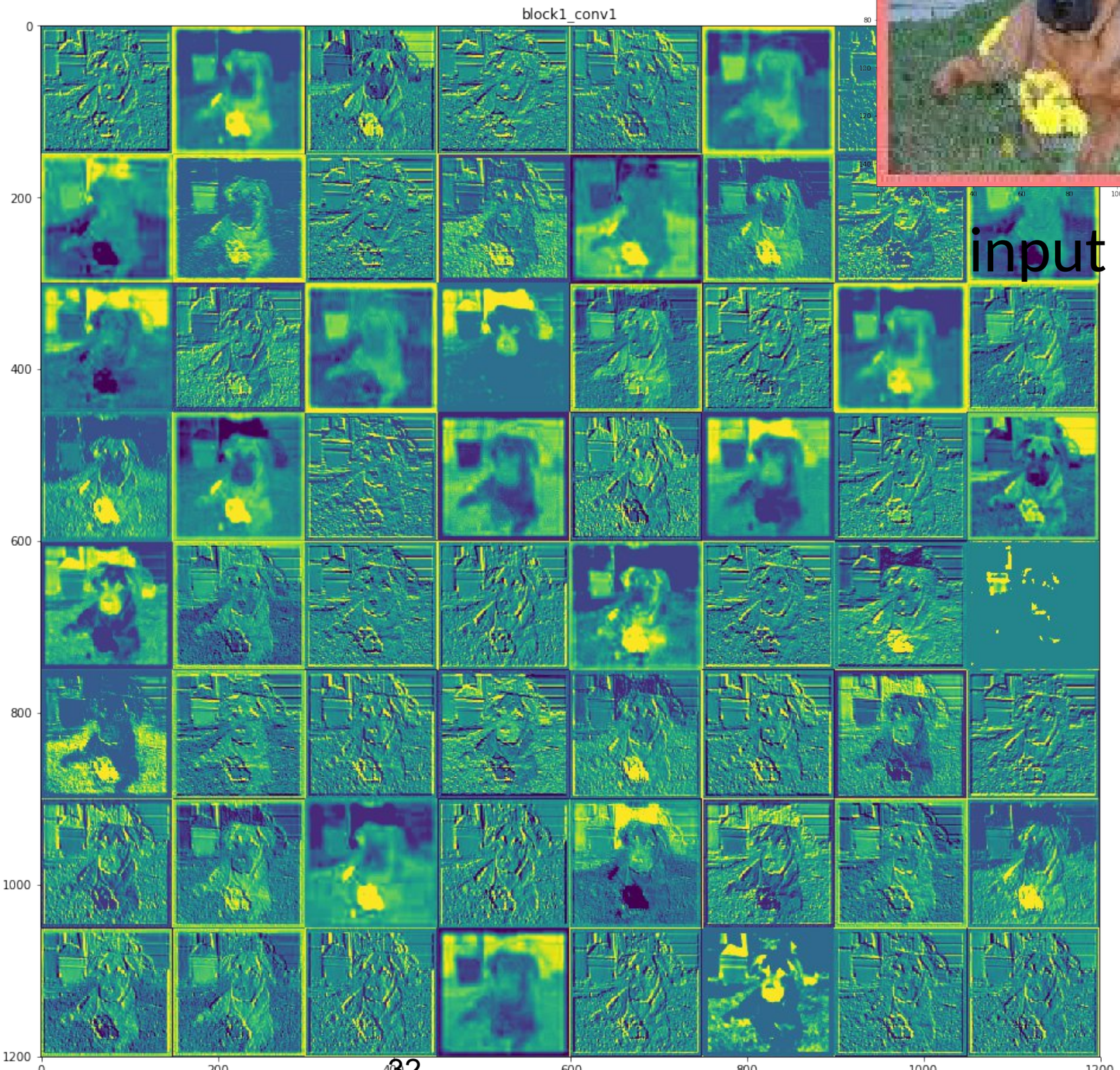
https://world4jason.gitbooks.io/research-log/content/deepLearning/CNN/Model%20&%20ImgNet/lenet

output



input

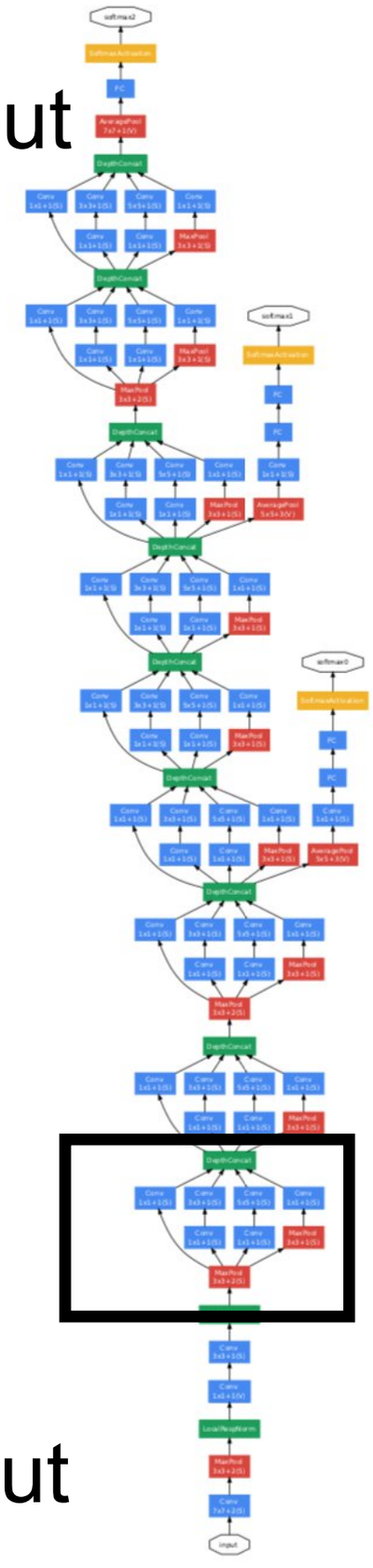
# Block1_conv1



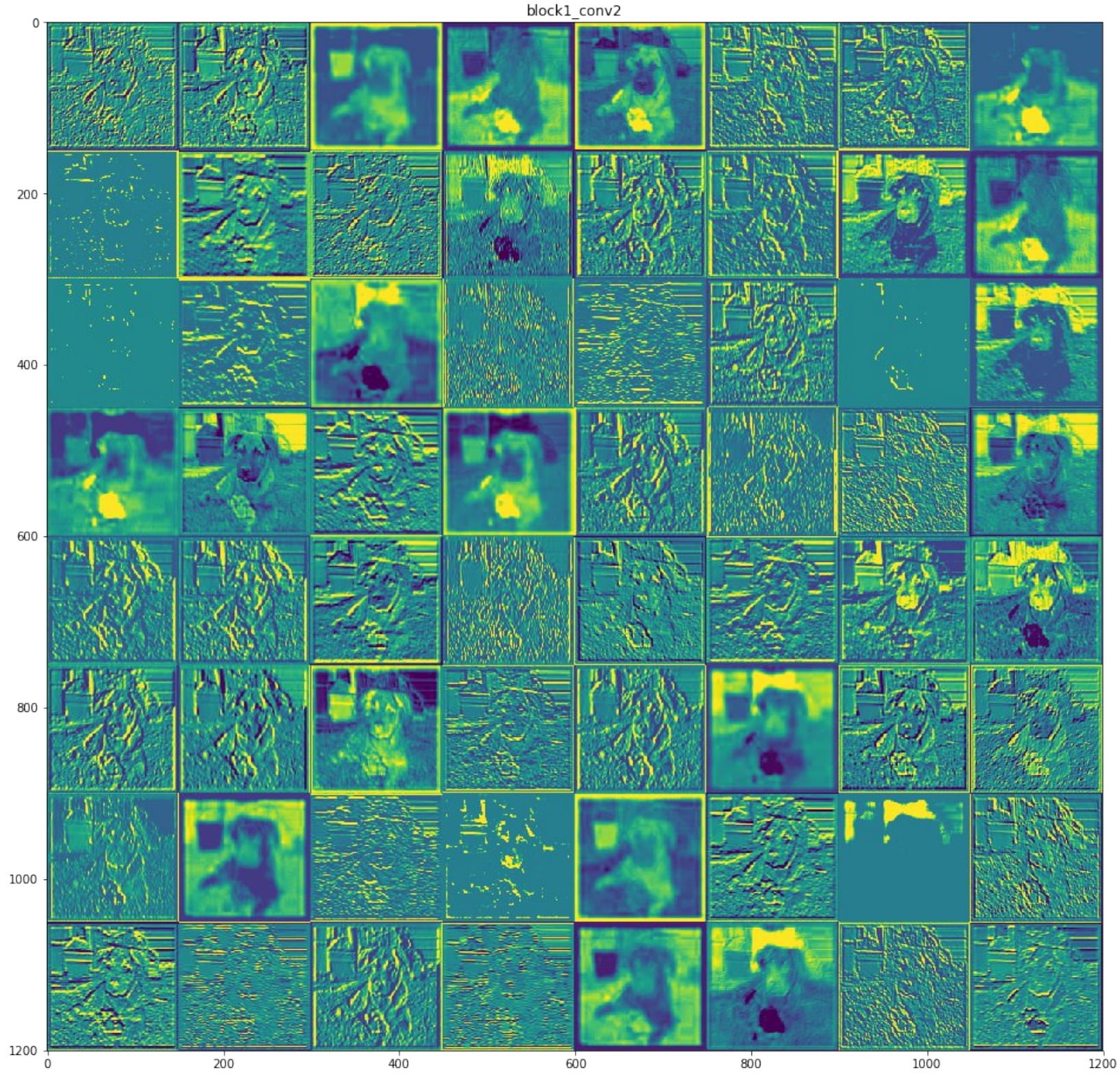


input

output

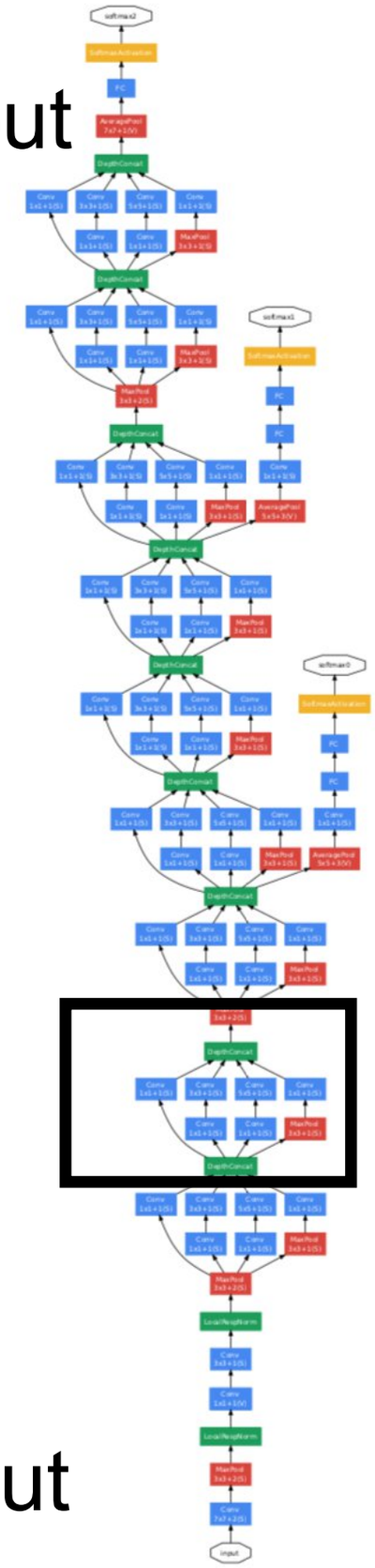


input

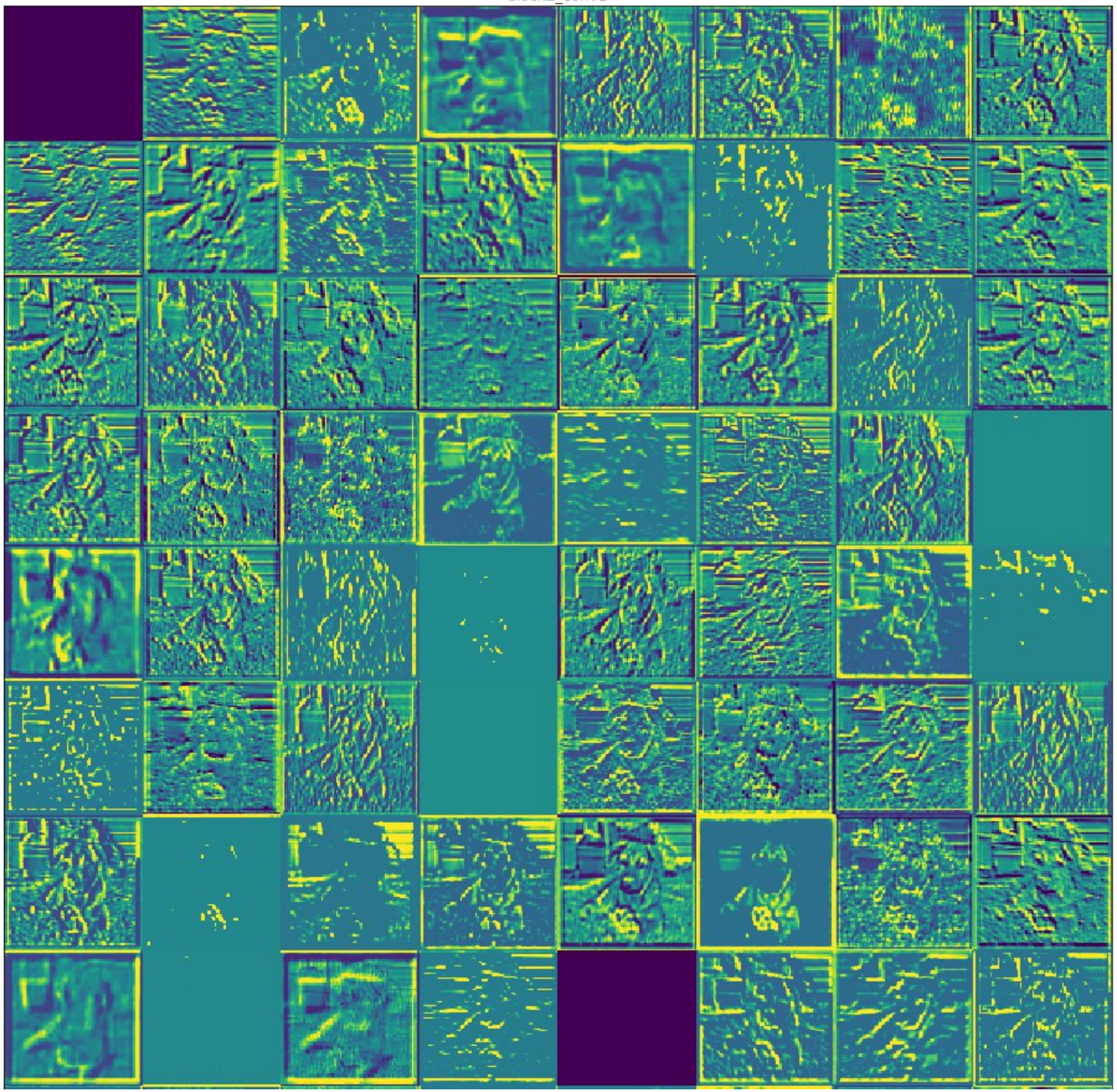



Block1_conv2

output

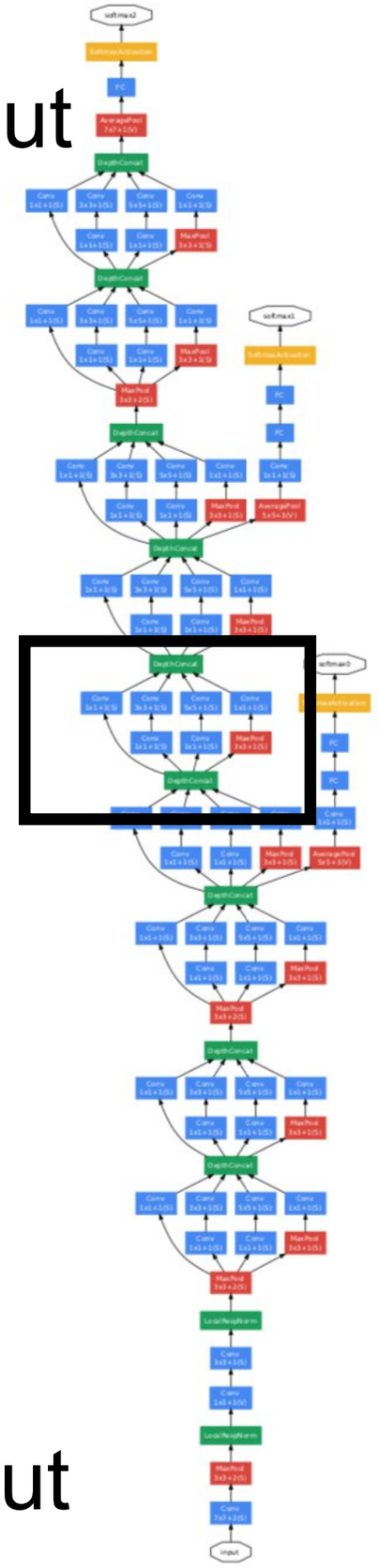


input

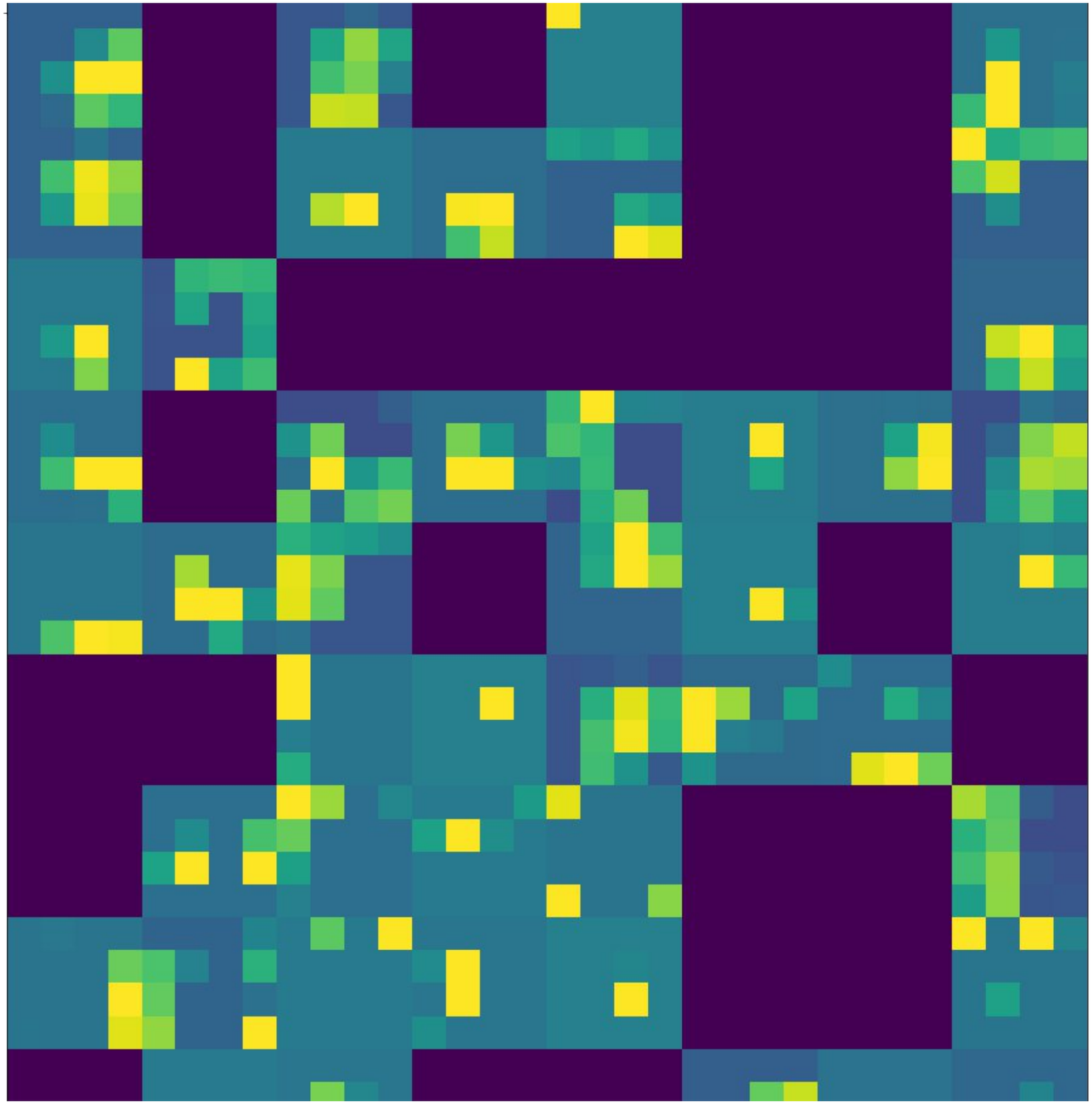



Block2_conv1

output



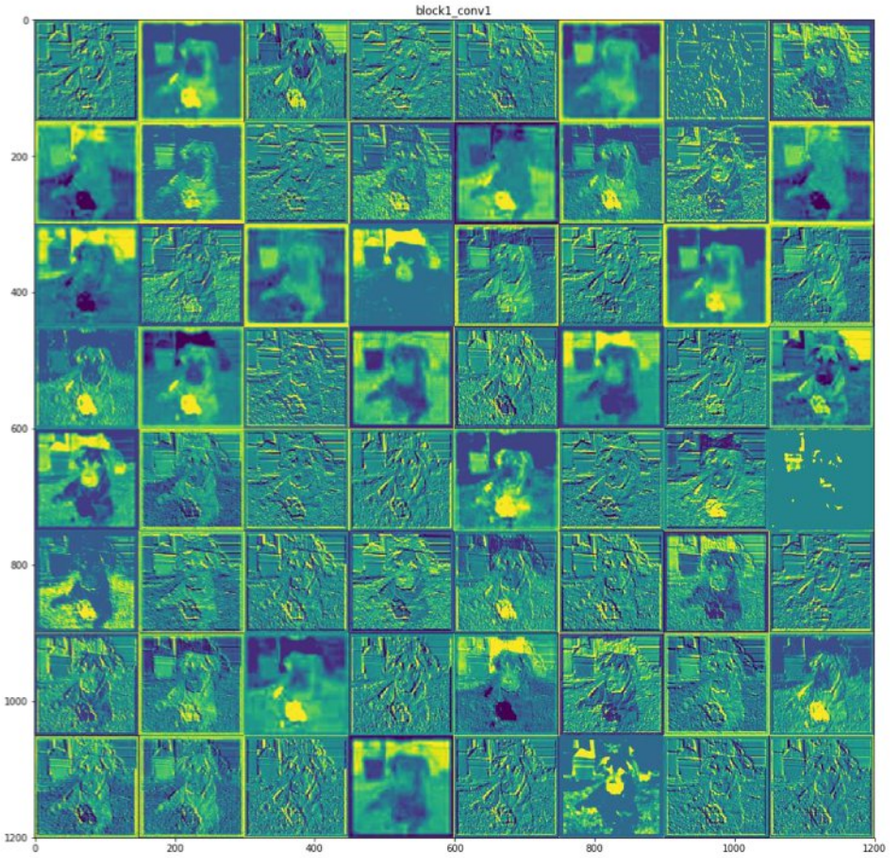
input



Block5_conv5

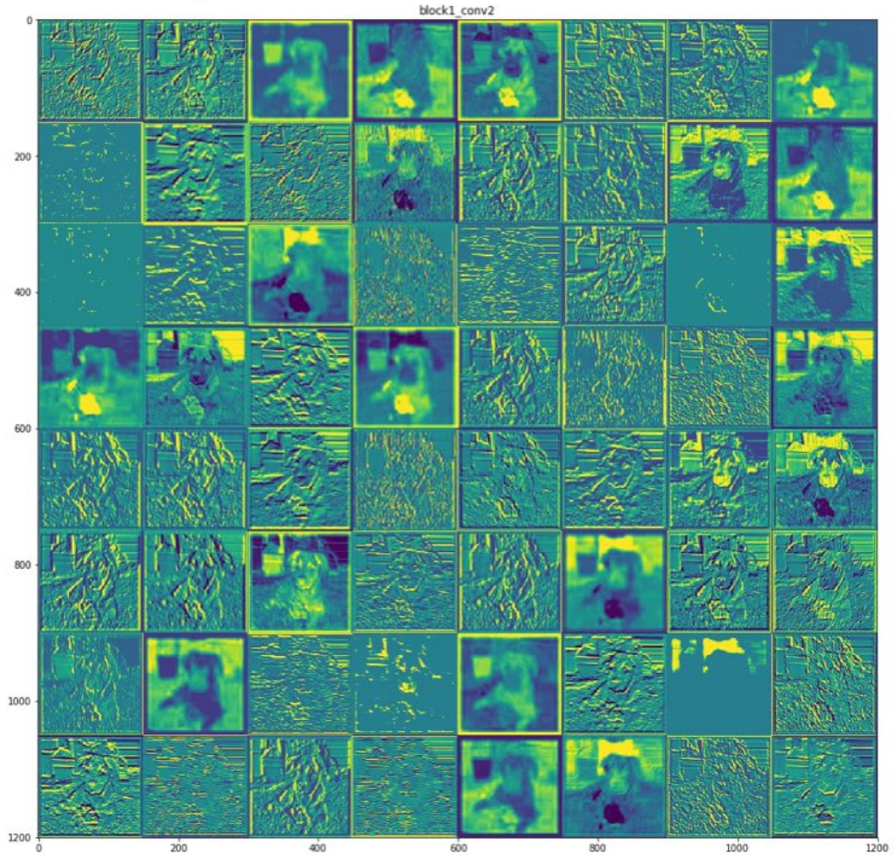
# 2) LAYERWISE

input

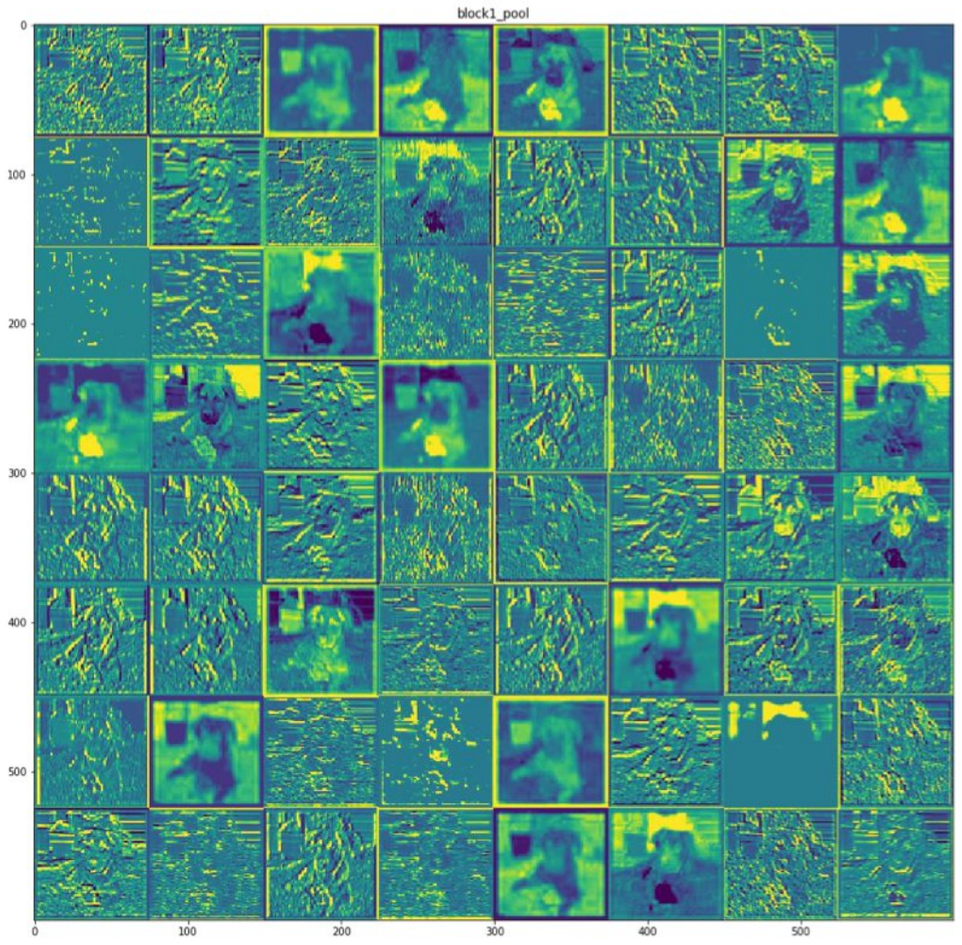



Block1_conv1

# 2) LAYERWISE

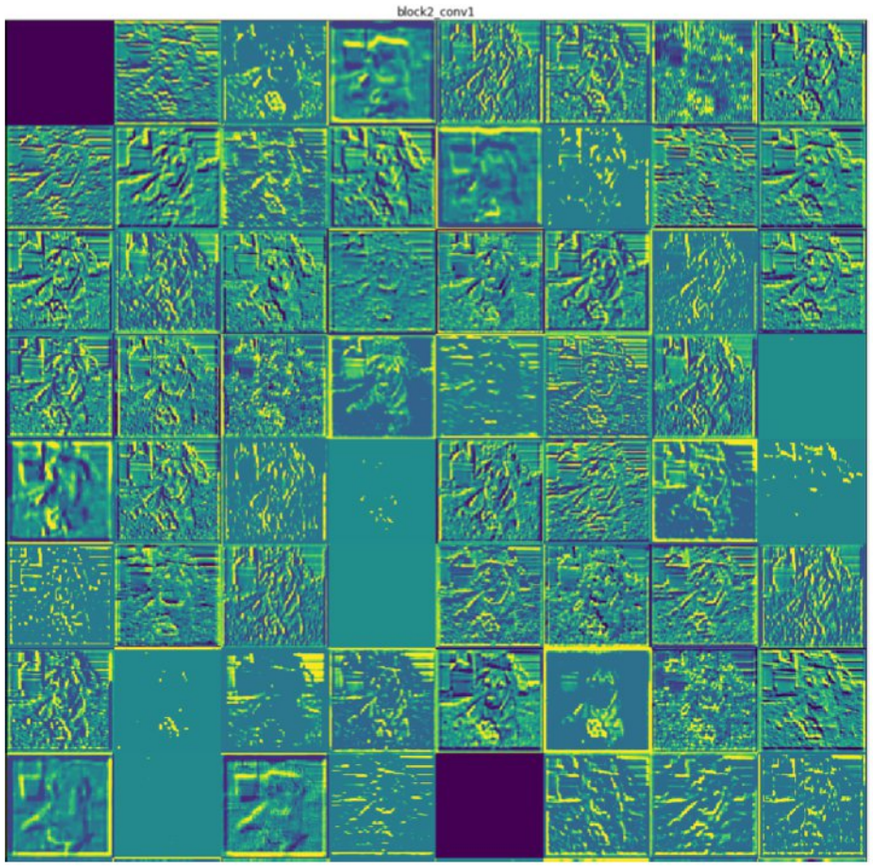



Block1_conv2

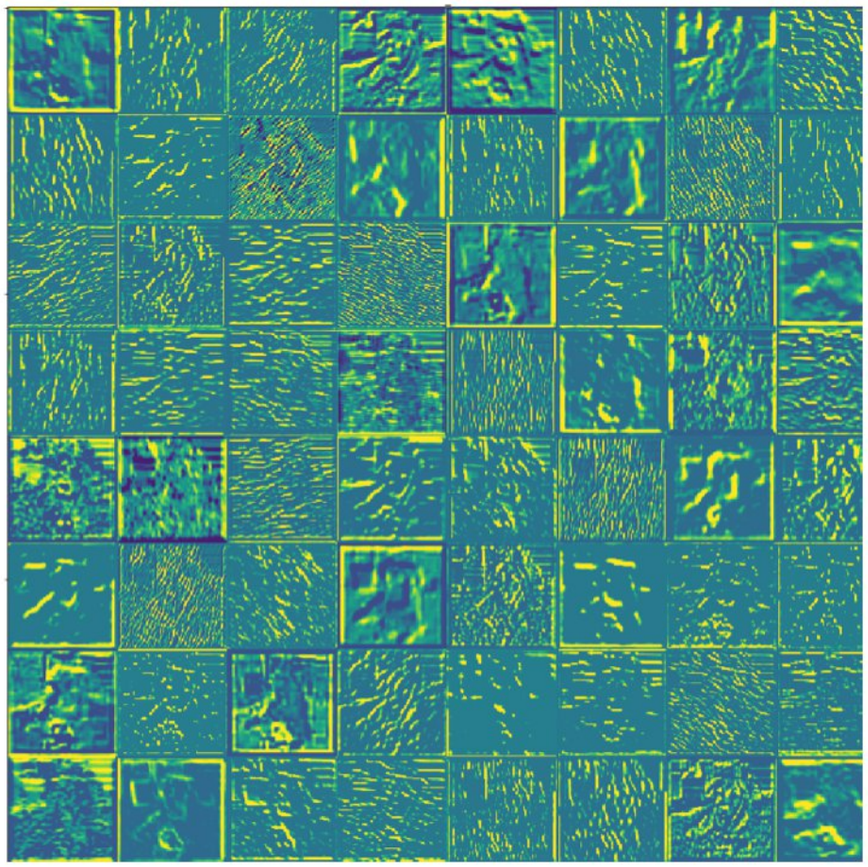



Block1_maxpool

# 2) LAYERWISE

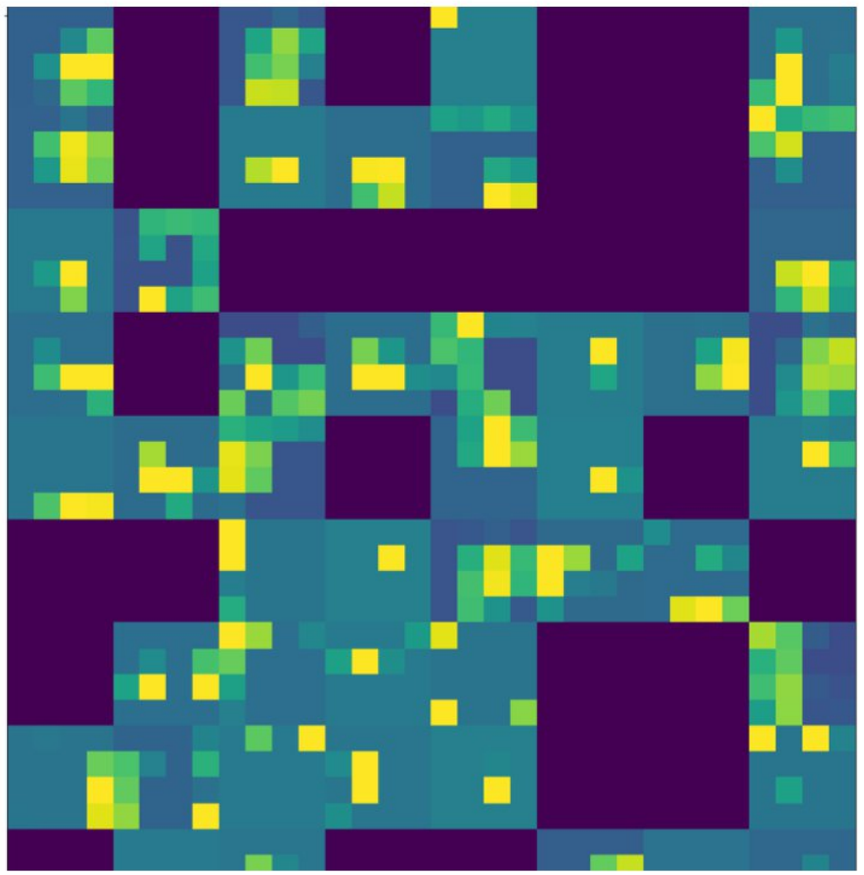



Block2_conv1



Block2_maxpool

# 2) LAYERWISE



Block5_conv5

We have 5 blocks, is has two to three conv layer and a maxpool layer.

The pattern keeps going on like this: zooming in effect.

People said that, the blank square (feature map) implies that this particular input don't have this feature.

# Overall architecture of CNN

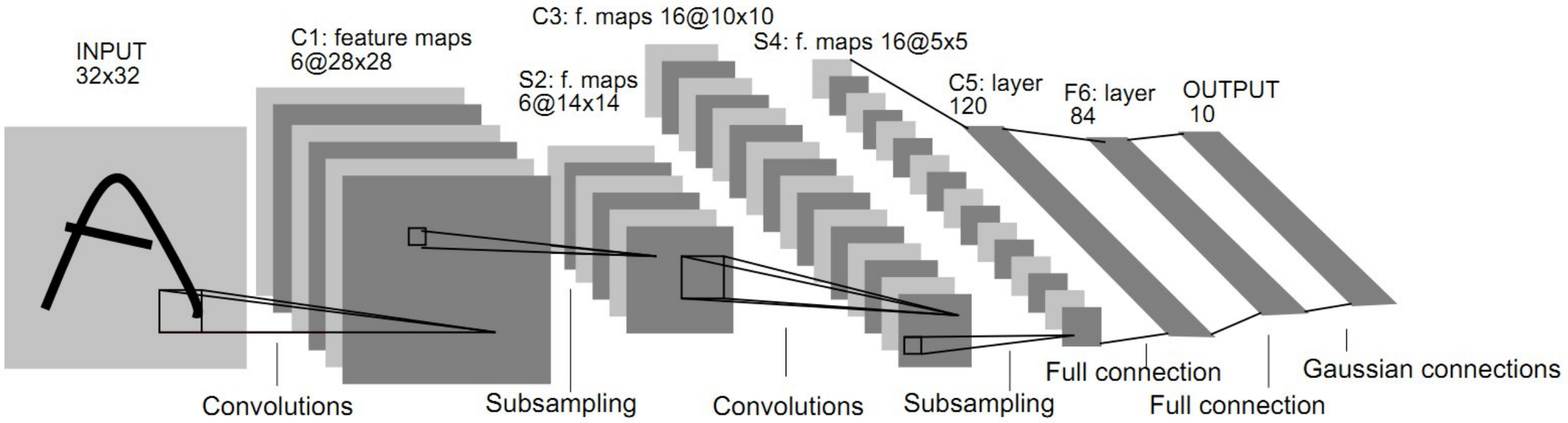



Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

https://world4jason.gitbooks.io/research-log/content/deepLearning/CNN/Model%20&%20ImgNet/lenet

# Lets get our hands wet through coding using MNIST data set

How to set up google colab

https://towardsdatascience.com/getting-started-with-google-colab-f2fff97f594c

28x28 pixels grayscale images

60,000 images for training
10,000 images for testing