Using Machine Learning Tools

Convolutional Neural Networks

University of Adelaide

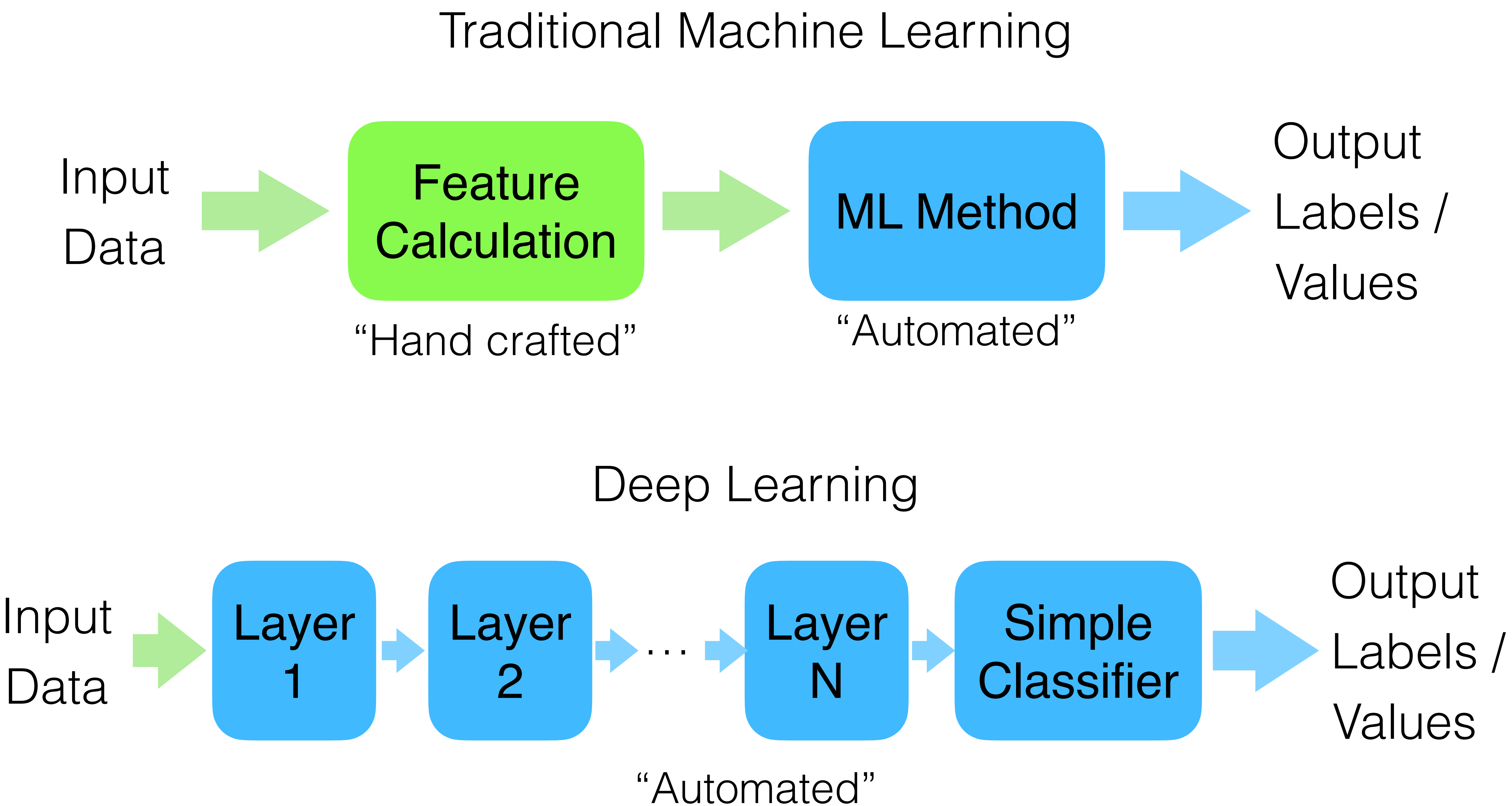
Previously …

- Network architecture: fully connected layers, one to next
- Neurons: importance of nonlinear activation functions (ReLU)
- Parameters: weights and biases, number of parameters
- Loss functions, epochs, batches, optimisers
- Regression:
  - loss = mean squared error ; mean absolute error
  - activation = None, ReLU, sigmoid
- Classification:
  - loss = cross entropy variants
  - activation = softmax (with one-hot representation)

Today

- Convolutional Neural Networks (CNNs)
- Architecture:
    - convolution layer
    - pooling layer
    - stride
    - number of filters
- Convergence and overfitting
- Ensembles

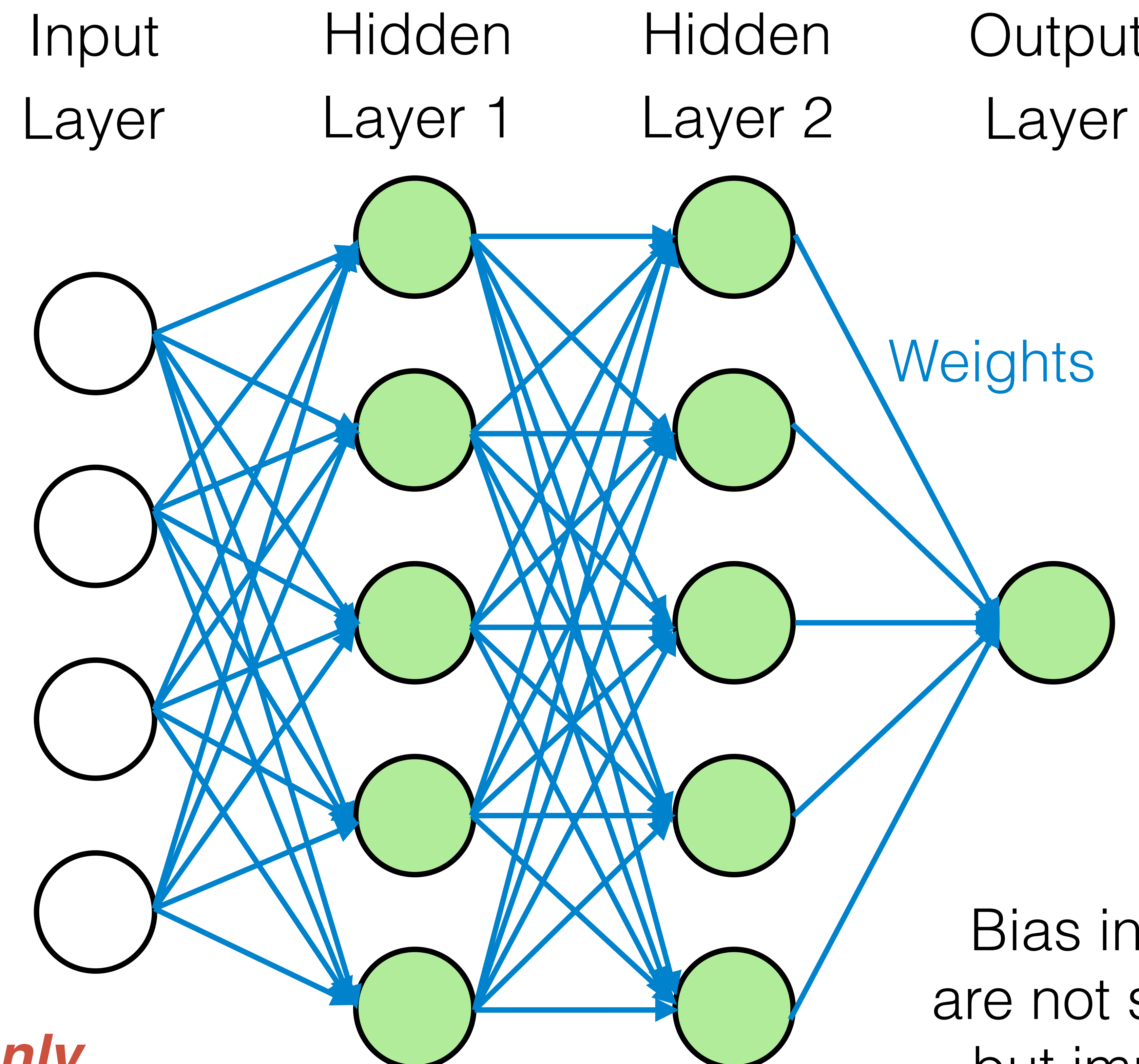
Introduction to Neural Networks



Neural Network

Input values as 1D vector

Fully Connected Layout



Bias inputs are not shown but implicit

Each layer ***only*** connects to next layer

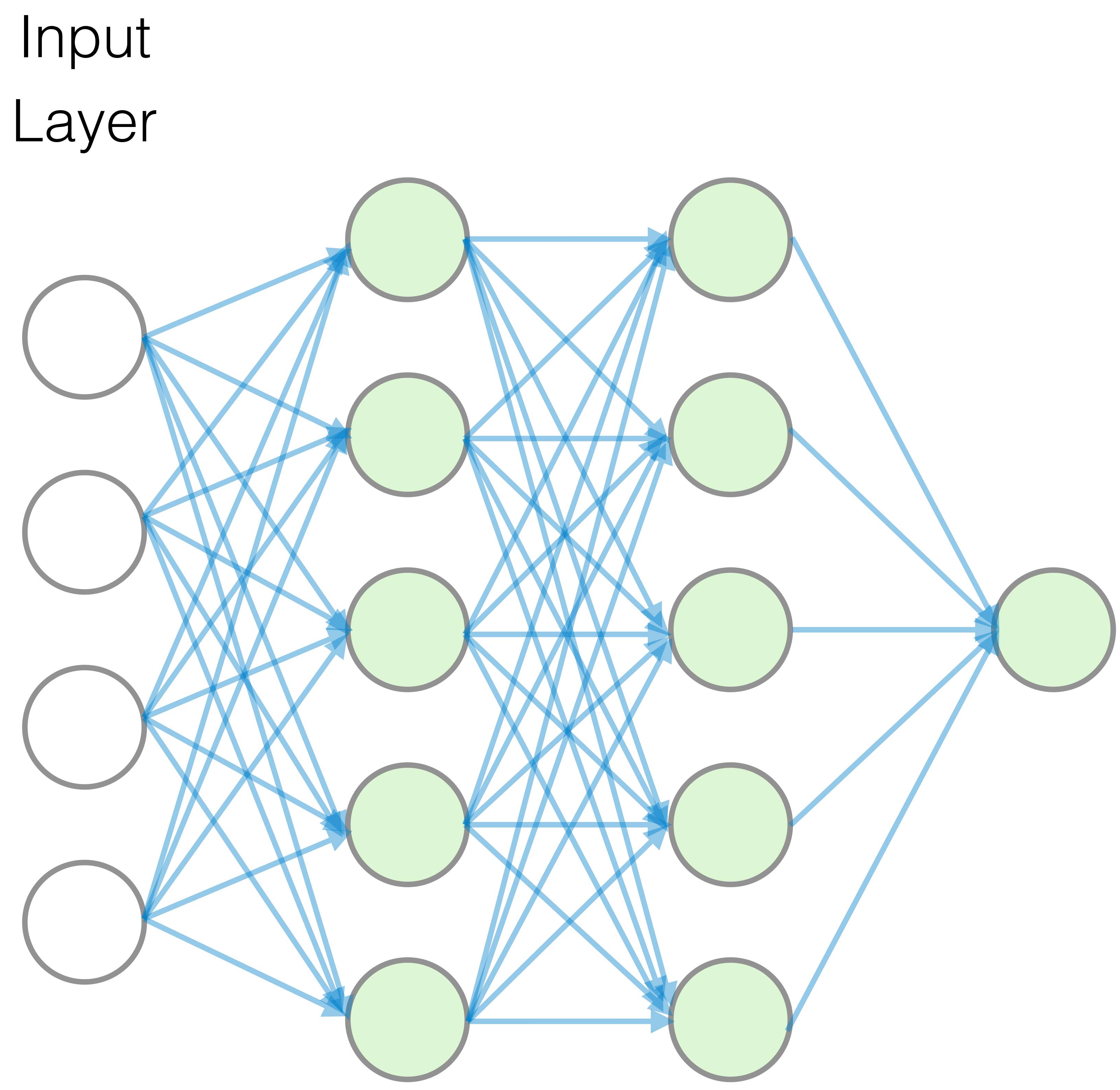
Neural Network

Input values as 1D vector

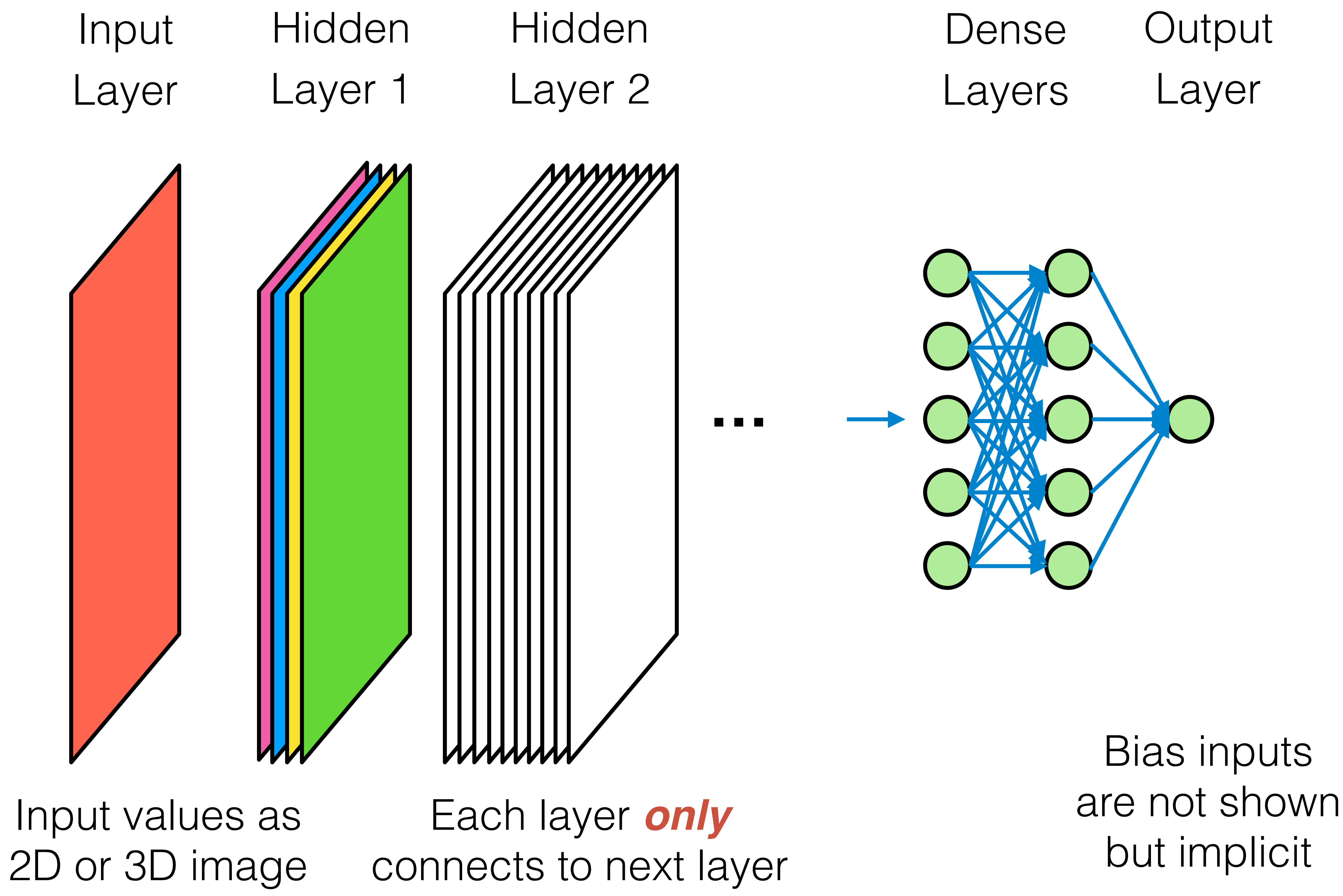
Reshaping into a 1D vector means that spatial relationships are lost

Instead, use a linear operation that works with images and has few parameters:

convolution



Convolutional Neural Network

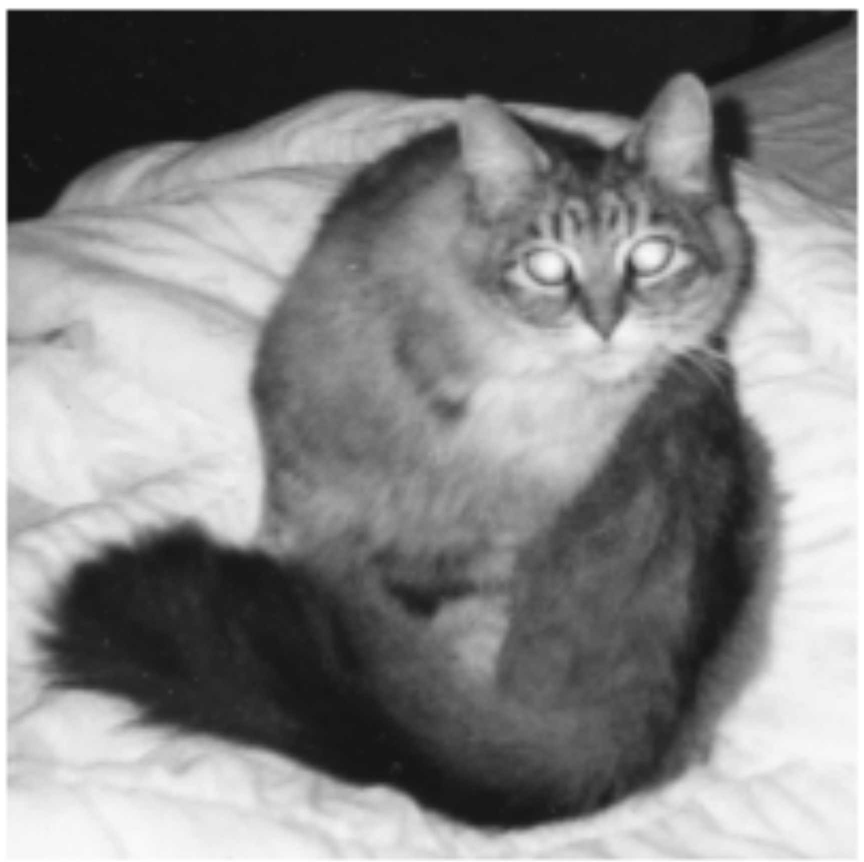


Convolution Layer

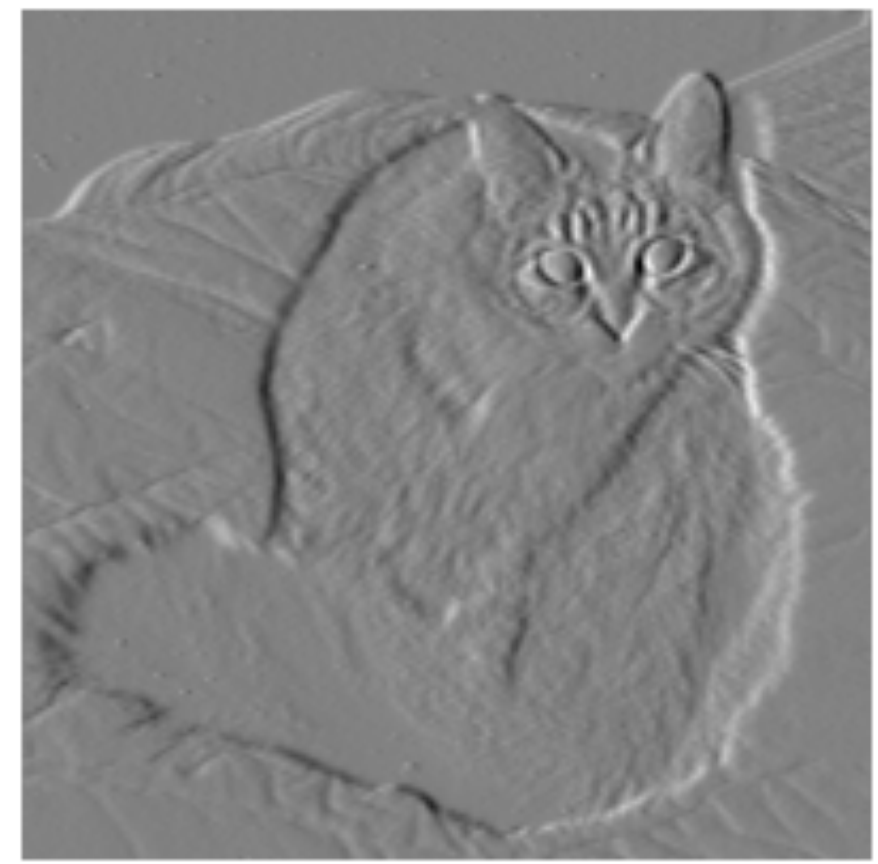
Image

Filter

x derivative



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



Convolution Layer

Image

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

Convolution Layer

Image

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

Filter

-1	2	-1
-2	4	-2
-1	2	-1

Convolution Layer

Image

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

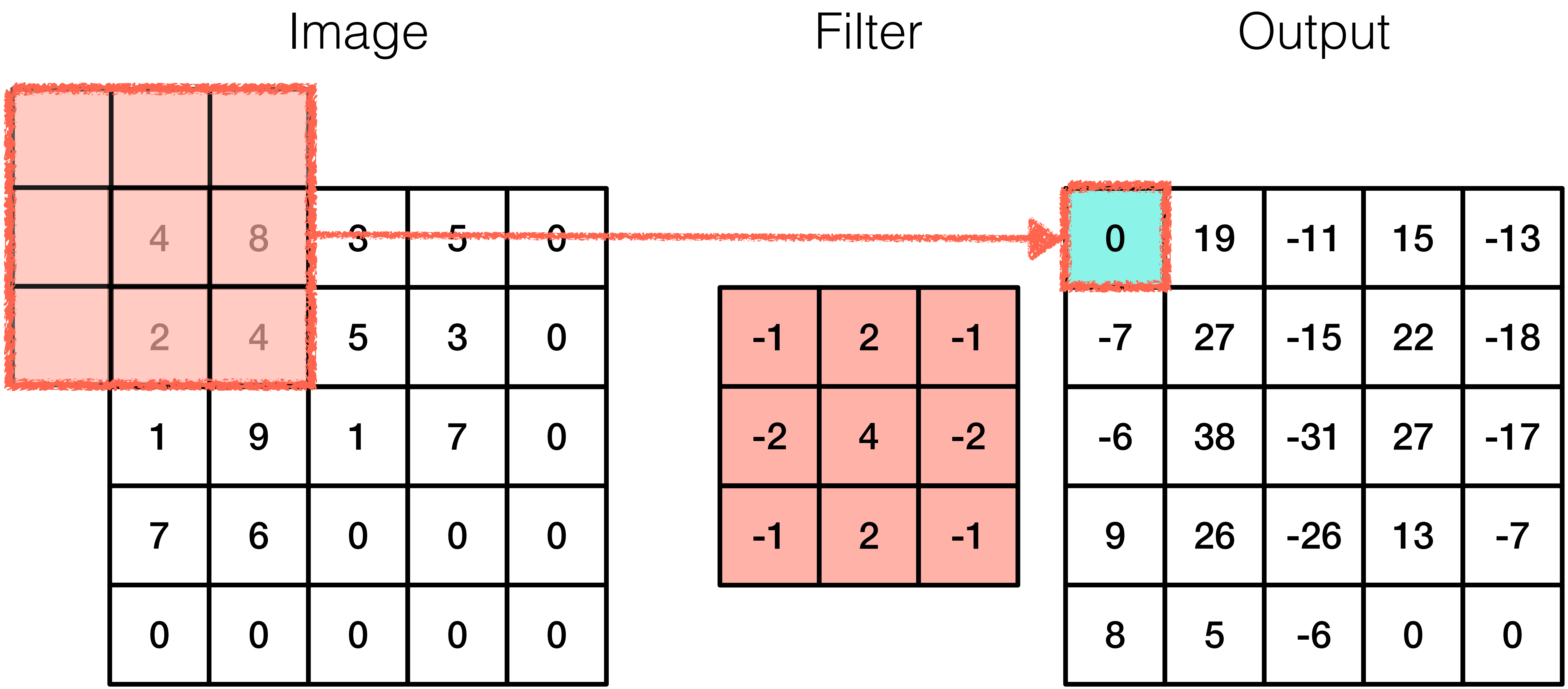
Filter

-1	2	-1
-2	4	-2
-1	2	-1

Output

0	19	-11	15	-13
-7	27	-15	22	-18
-6	38	-31	27	-17
9	26	-26	13	-7
8	5	-6	0	0

Convolution Layer



Convolution Layer

Image

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

Filter

-1	2	-1
-2	4	-2
-1	2	-1

Output

0	19	-11	15	-13
-7	27	-15	22	-18
-6	38	-31	27	-17
9	26	-26	13	-7
8	5	-6	0	0

Convolution Layer

Image

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

Filter

-1	2	-1
-2	4	-2
-1	2	-1

Output

0	19	-11	15	-13
-7	27	-15	22	-18
-6	38	-31	27	-17
9	26	-26	13	-7
8	5	-6	0	0

Convolution Layer

Image

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

Filter

-1	2	-1
-2	4	-2
-1	2	-1

Output

0	19	-11	15	-13
-7	27	-15	22	-18
-6	38	-31	27	-17
9	26	-26	13	-7
8	5	-6	0	0

Convolution Layer

Image

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

Filter

-1	2	-1
-2	4	-2
-1	2	-1

Output

0	19	-11	15	-13
-7	27	-15	22	-18
-6	38	-31	27	-17
9	26	-26	13	-7
8	5	-6	0	0

Convolution Layer

Image

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

Filter

-1	2	-1
-2	4	-2
-1	2	-1

Output

0	19	-11	15	-13
-7	27	-15	22	-18
-6	38	-31	27	-17
9	26	-26	13	-7
8	5	-6	0	0

Convolution Layer

Image

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

Filter

-1	2	-1
-2	4	-2
-1	2	-1

Output

0	19	-11	15	-13
-7	27	-15	22	-18
-6	38	-31	27	-17
9	26	-26	13	-7
8	5	-6	0	0

Convolution Layer

Image

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

Filter

-1	2	-1
-2	4	-2
-1	2	-1

Output

0	19	-11	15	-13
-7	27	-15	22	-18
-6	38	-31	27	-17
9	26	-26	13	-7
8	5	-6	0	0

Convolution Layer

Image

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

Filter

-1	2	-1
-2	4	-2
-1	2	-1

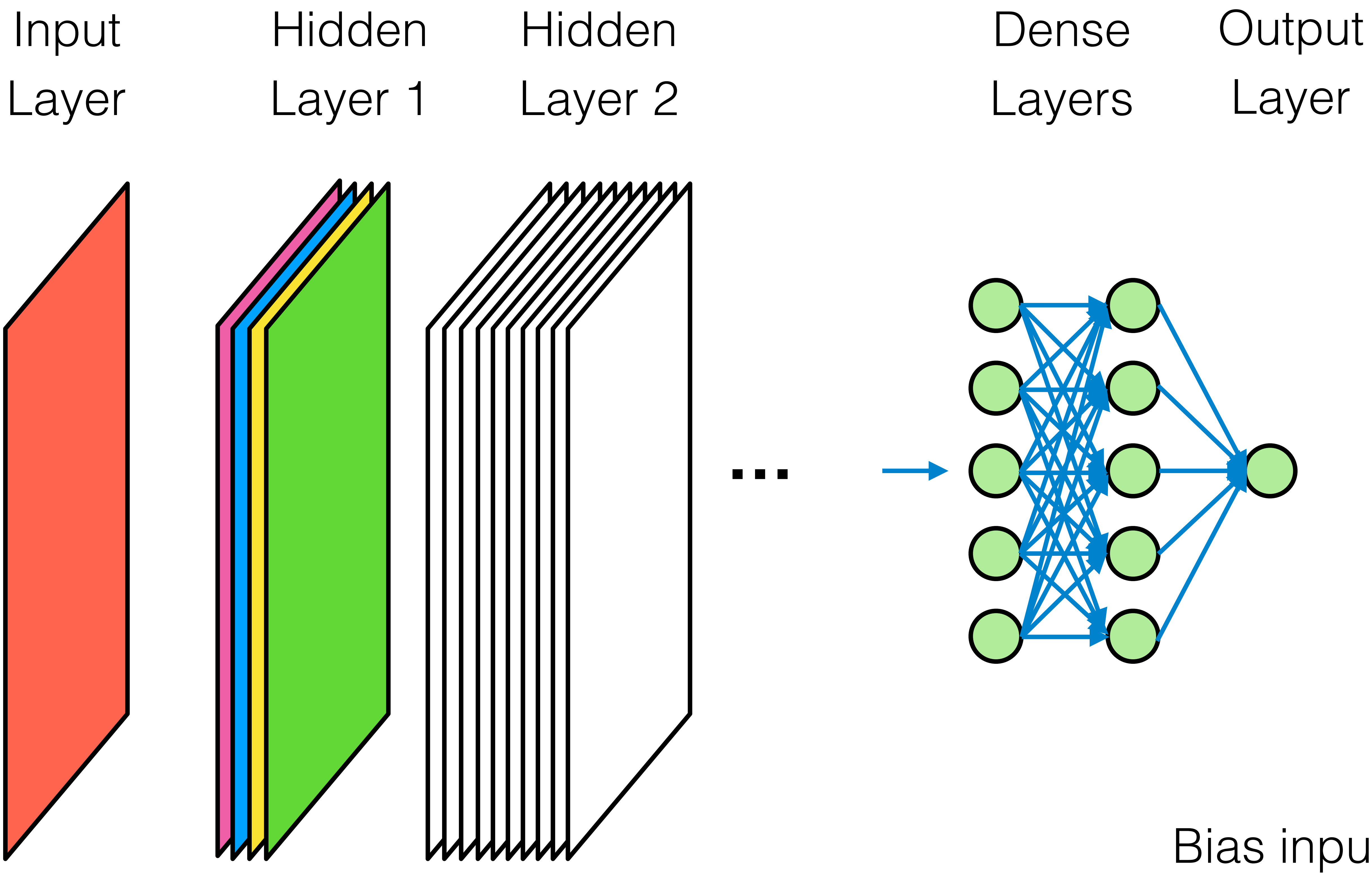
Output

0	19	-11	15	-13
-7	27	-15	22	-18
-6	38	-31	27	-17
9	26	-26	13	-7
8	5	-6	0	0

Activation is also applied to outputs

Bias inputs are not shown but implicit

Convolutional Neural Network



Input values as 2D or 3D image

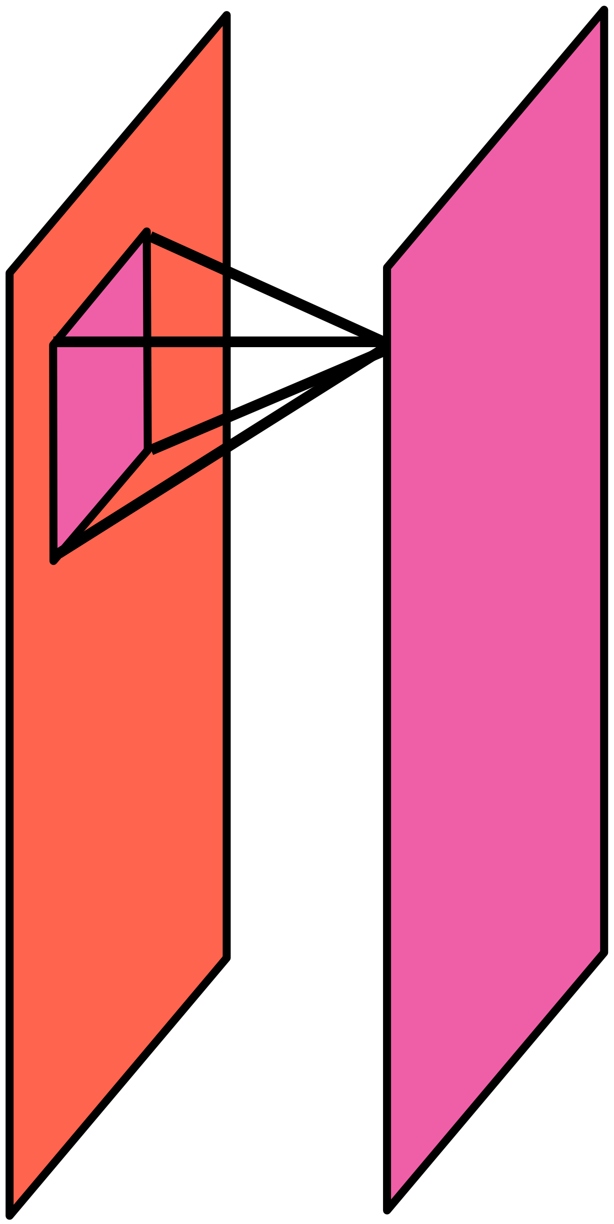
Each layer ***only*** connects to next layer

Bias inputs are not shown but implicit

Convolutional Neural Network

Input Layer

Hidden Layer 1



Have many filters (kernels)

Each filter used to make another output image, stacked together

Apply each convolutional filter in term

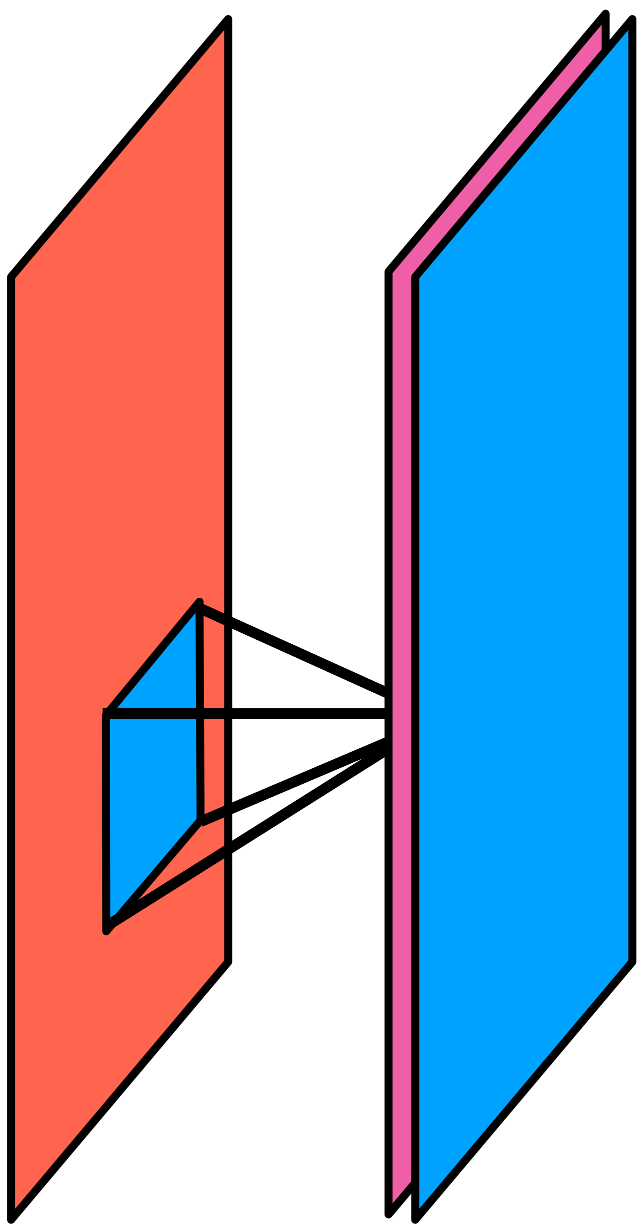
Input values as 2D or 3D image

Each layer ***only*** connects to next layer

Convolutional Neural Network

Input Layer

Hidden Layer 1



Input values as 2D or 3D image

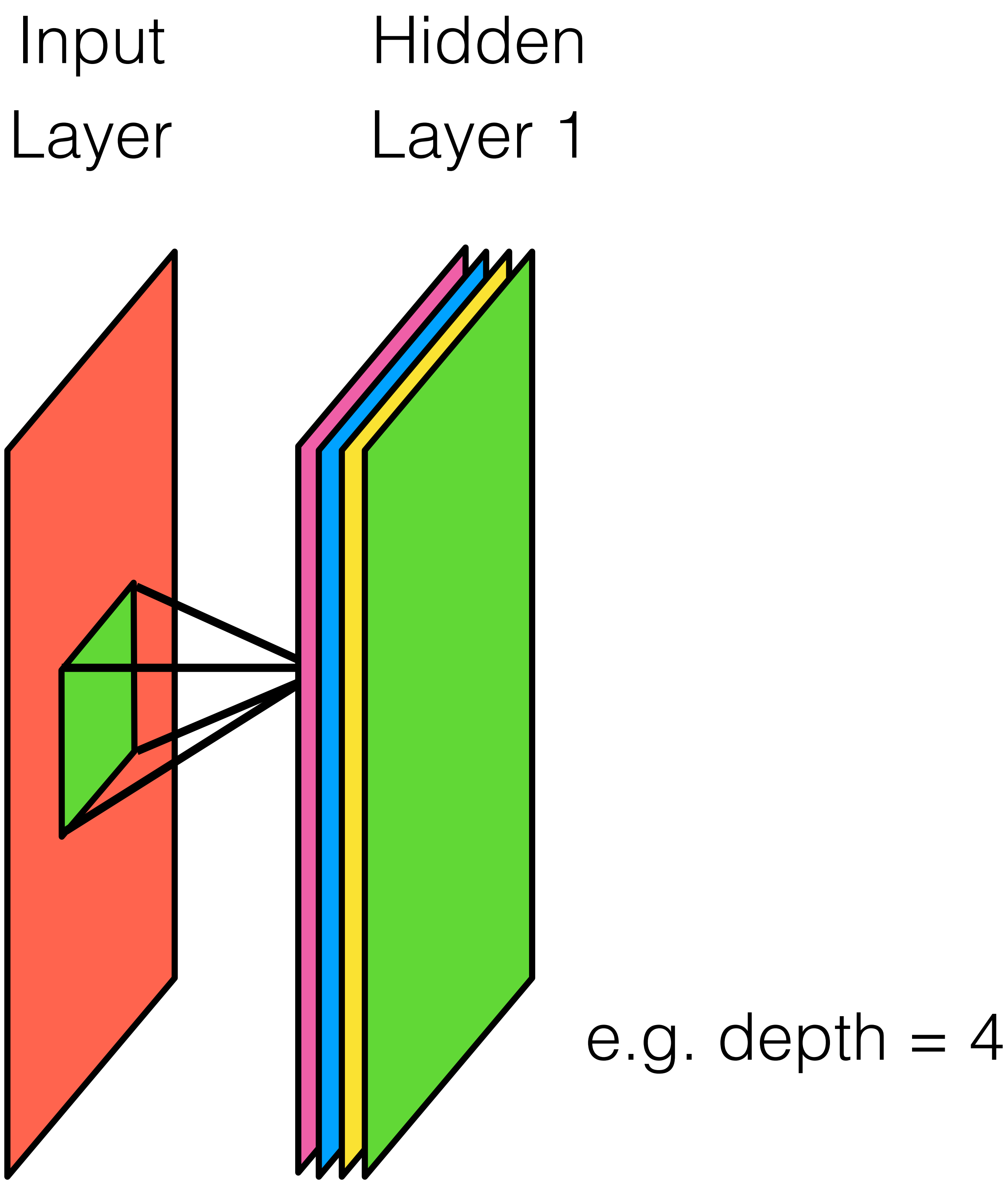
Each layer ***only*** connects to next layer

Have many filters (kernels)

Each filter used to make another output image, stacked together

Apply each convolutional filter in term

Convolutional Neural Network



Input values as
2D or 3D image

Each layer ***only***
connects to next layer

Have many
filters (kernels)

Each filter used
to make another
output image,
stacked together

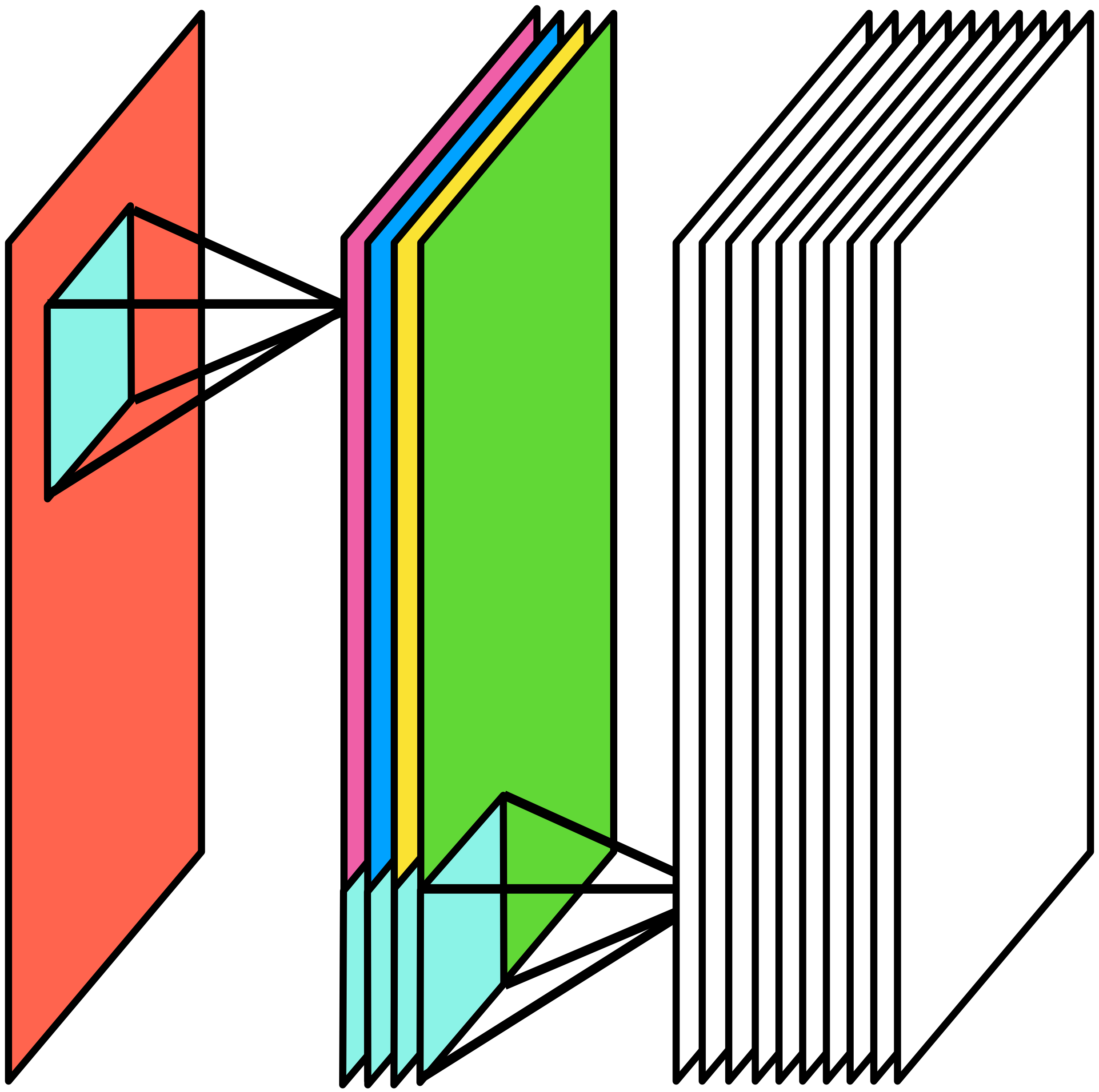
Apply each
convolutional
filter in term

Convolutional Neural Network

Input Layer

Hidden Layer 1

Hidden Layer 2



Input values as 2D or 3D image

Each layer ***only*** connects to next layer

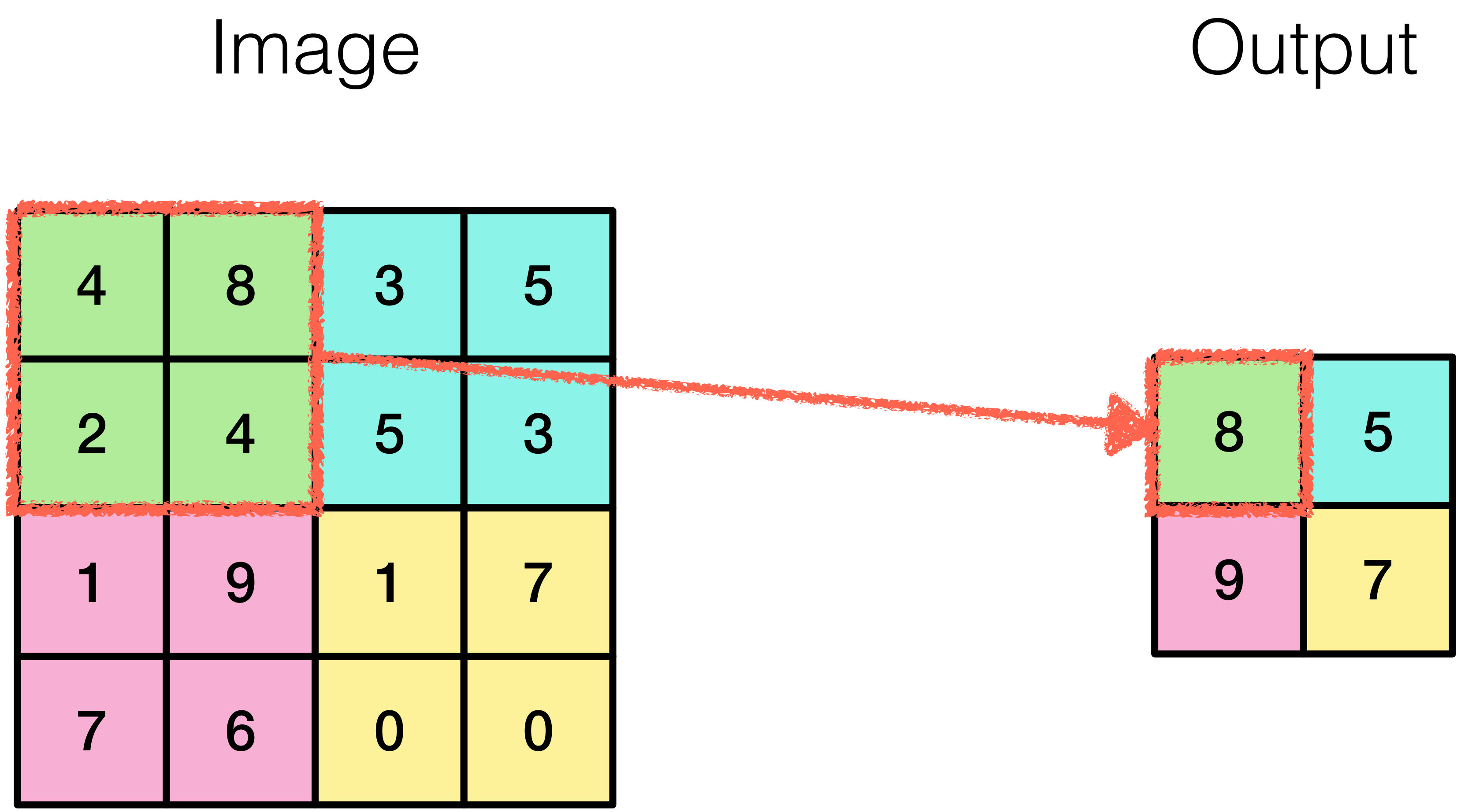
Have many filters (kernels)

Each filter used to make another output image, stacked together

Apply each convolutional filter in term

Each filter acts across all depths

Max Pooling Layer

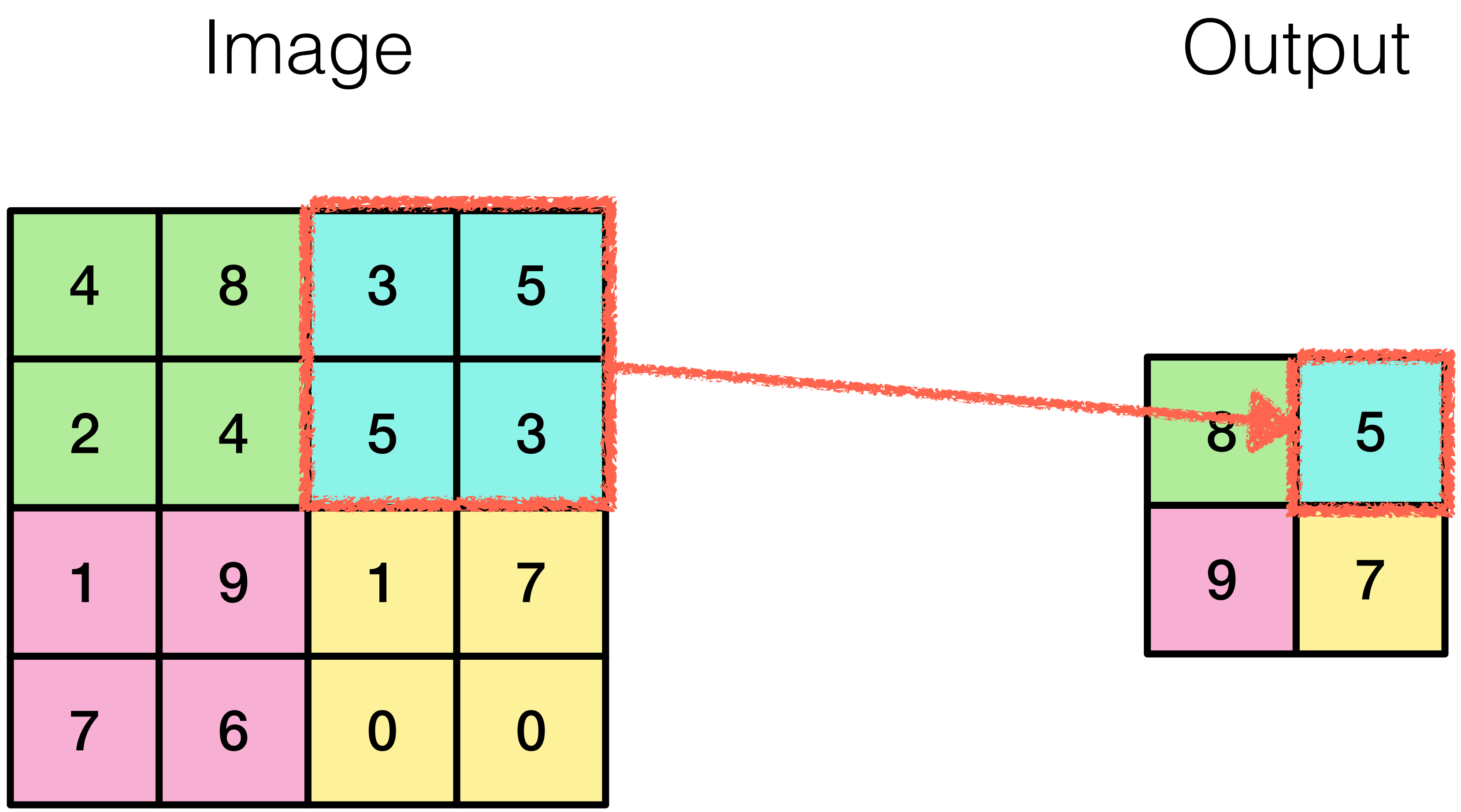


Take maximum value within 2x2 filter

Move in steps of 2 ***(stride = 2)***

Zero parameters

Max Pooling Layer



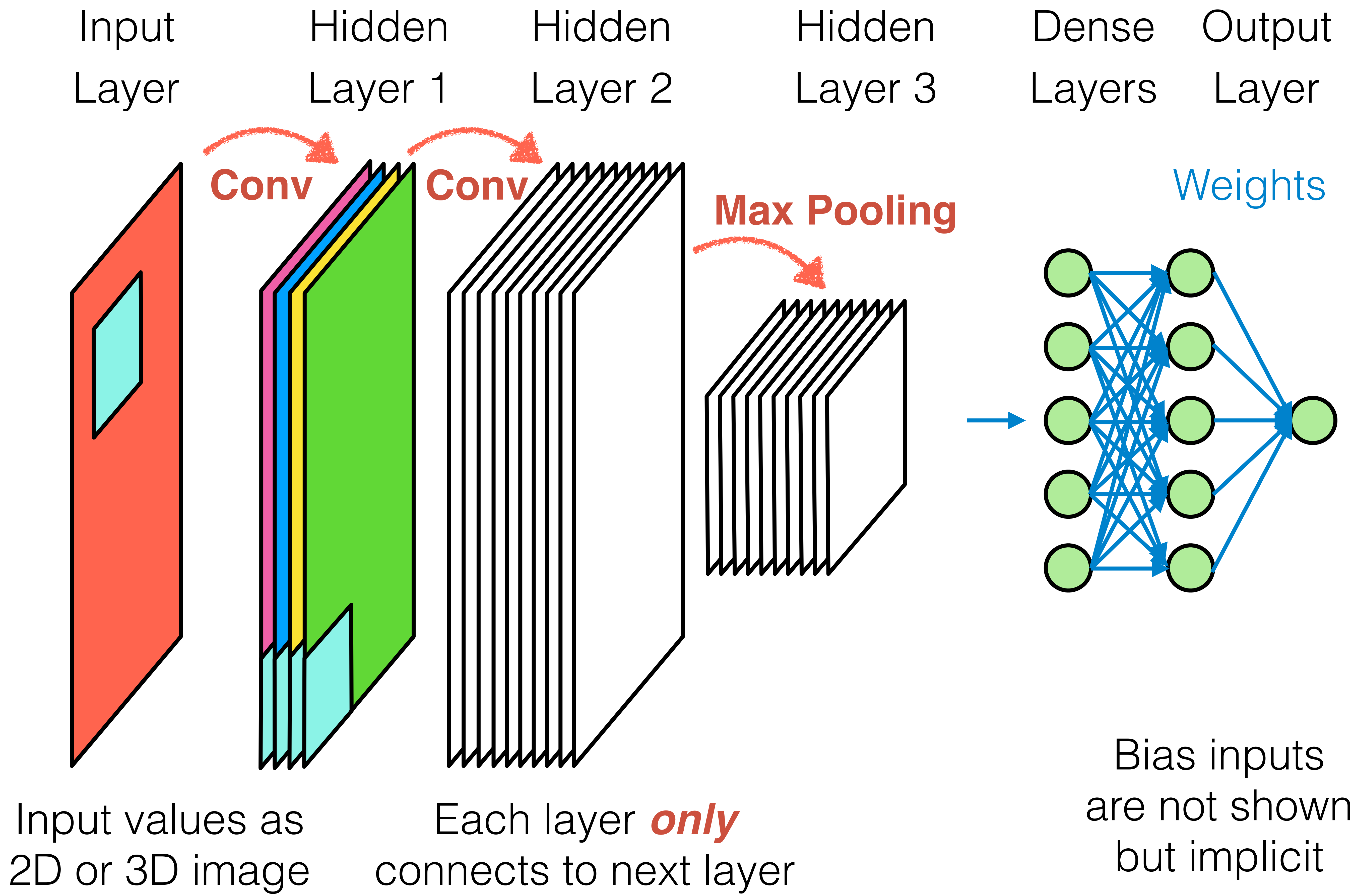
Take maximum value within 2x2 filter

Move in steps of 2 ***(stride = 2)***

Zero parameters

Average Pooling is also an option

Convolutional Neural Network



Parameters

Filter

-1	2	-1
-2	4	-2
-1	2	-1

Size of 3x3 is most common

Image

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

Output

0	19	-11	15	-13
7	27	-15	22	-18
-6	38	-31	27	-17
9	26	-26	13	-7
8	5	-6	0	0

Number of parameters is 3x3 = 9

+ 1 bias input (implicit)

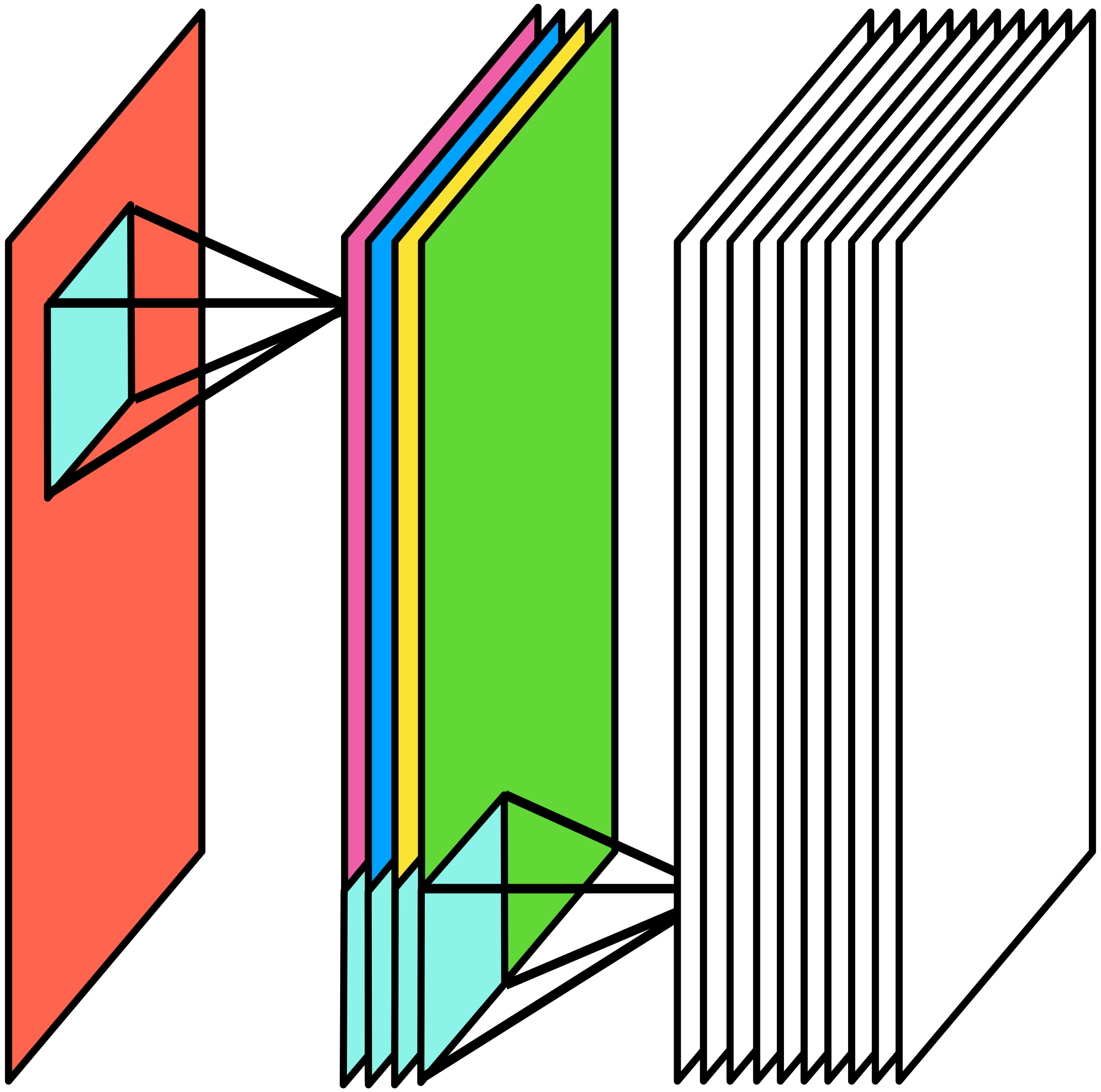
= 10 parameters

Independent of image size!

… for a scalar input image (depth=1)

NB: depth = 3 for colour channels

Convolutional Neural Network



Have many filters (kernels)

Each filter used to make another output image, stacked together

Apply each convolutional filter in term

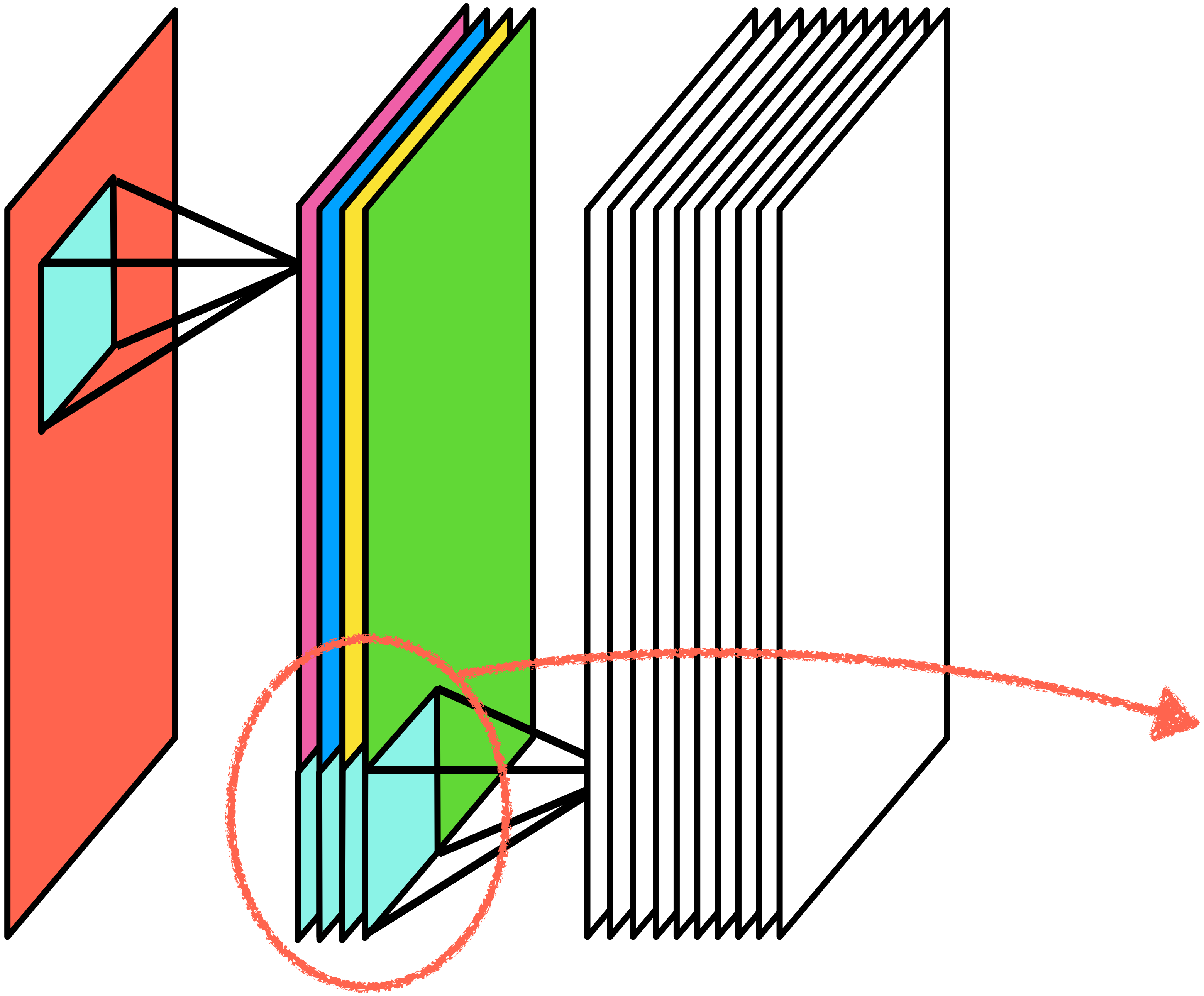
Each filter acts across all depths

For deeper stacks, the number of parameters in *each filter* is:

depth * filter size + bias

e.g. 4 * 3*3 + 1 = 37

Convolutional Neural Network



Number of filters = depth of next layer

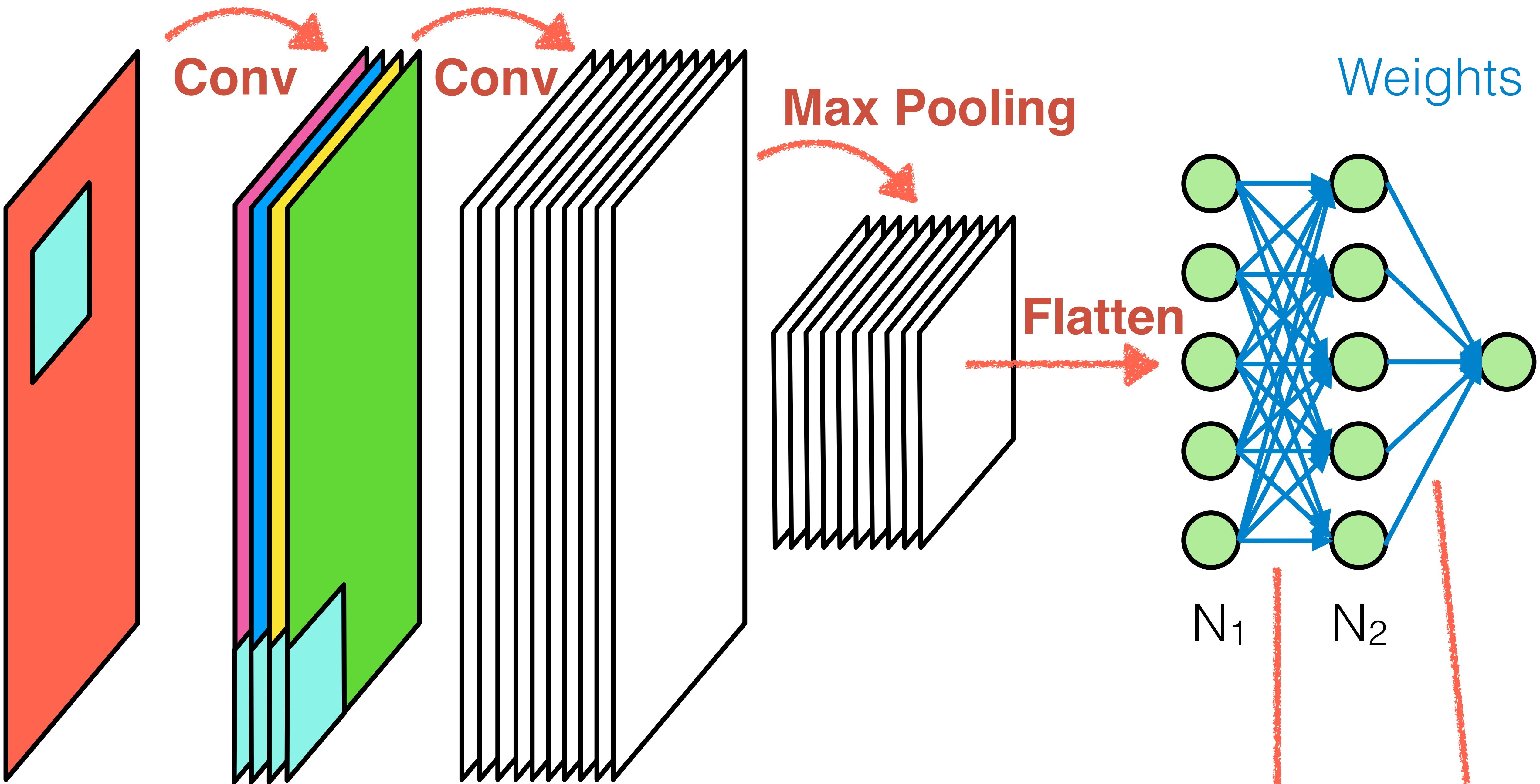
e.g. 10 filters = 370 parameters

For deeper stacks, the number of parameters in *each filter* is:

depth * filter size + bias

e.g. 4 * 3*3 + 1 = 37

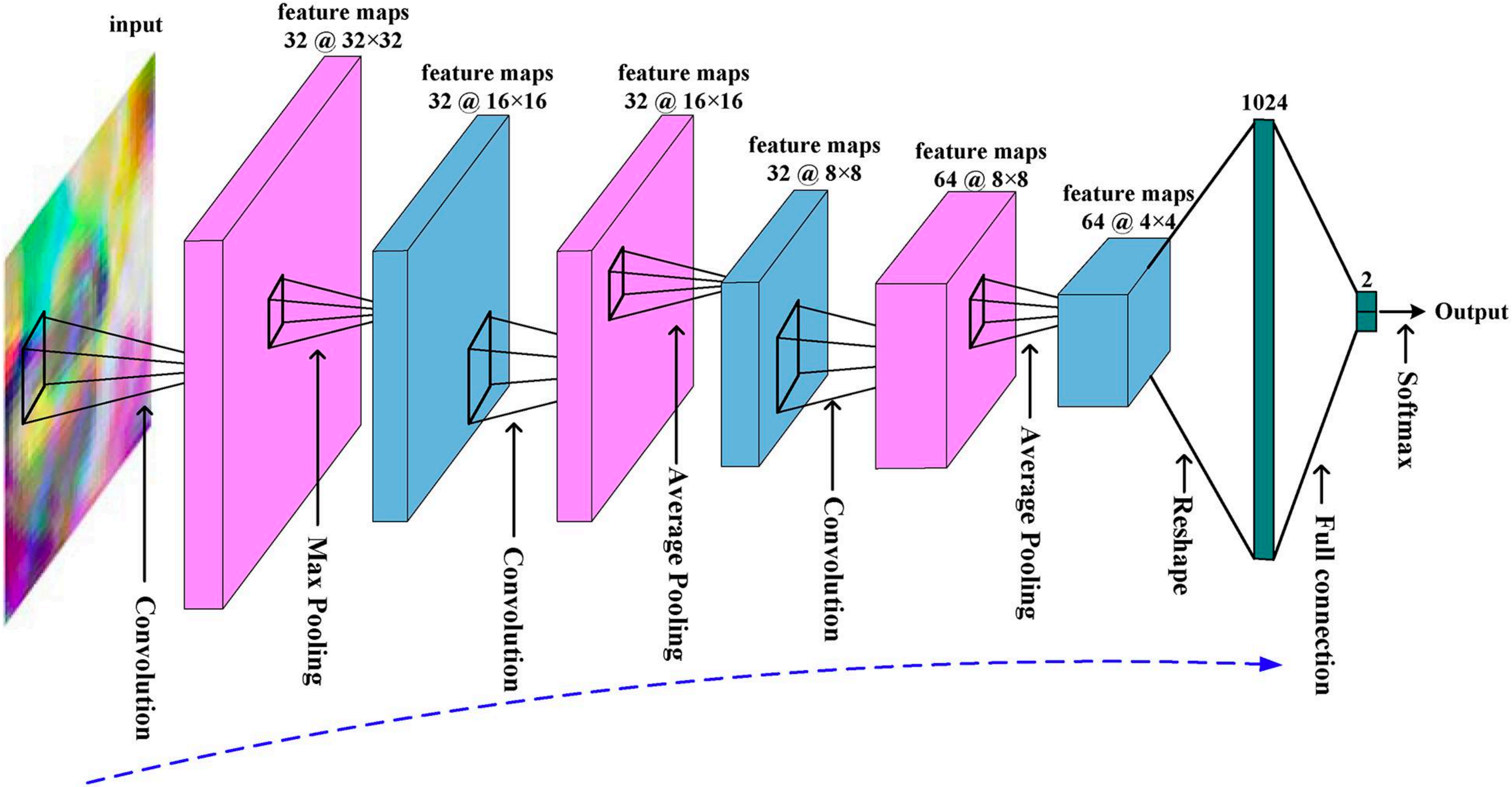
Parameters



Total number of parameters:

$(3*3+1)*4 = 40$ | $(3*3*4+1)*10 = 370$ | 0 | 0 | $(N_1+1)*N_2$ | N_2+1

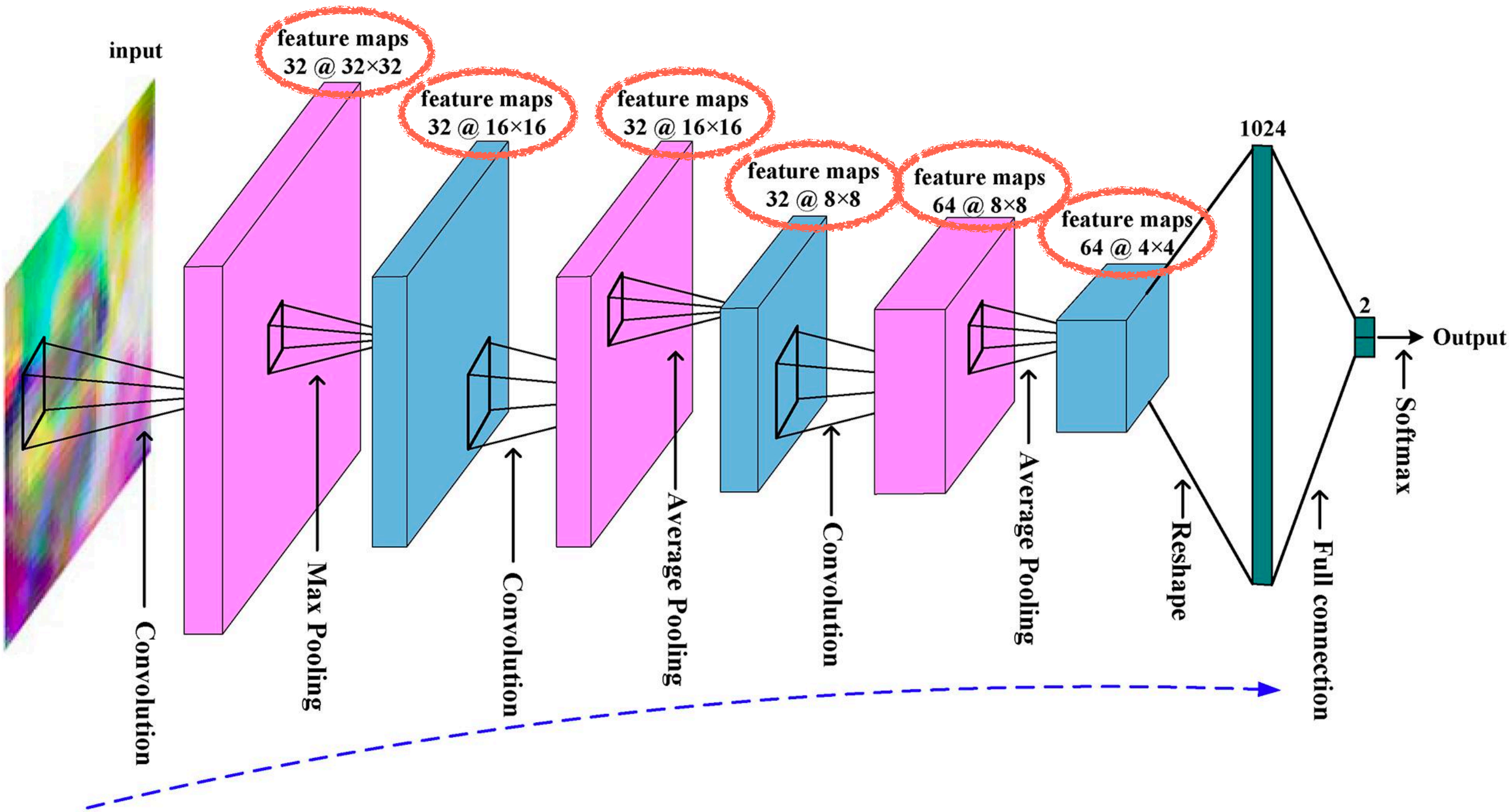
Example



Lin et al., *Convolutional Neural Networks-Based MRI Image Analysis for the Alzheimer's Disease Prediction From Mild Cognitive Impairment,* Front. Neurosci., 2018

Example

Lots of decisions about architecture

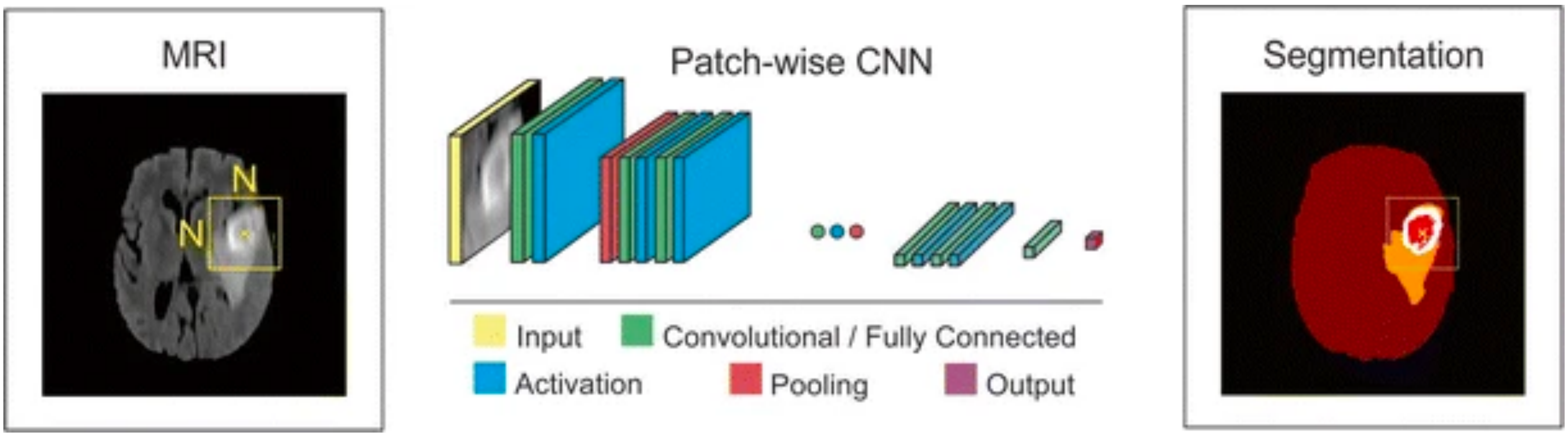


Lin et al., *Convolutional Neural Networks-Based MRI Image Analysis for the Alzheimer's Disease Prediction From Mild Cognitive Impairment,* Front. Neurosci., 2018

Example

Patch-based application for segmentation

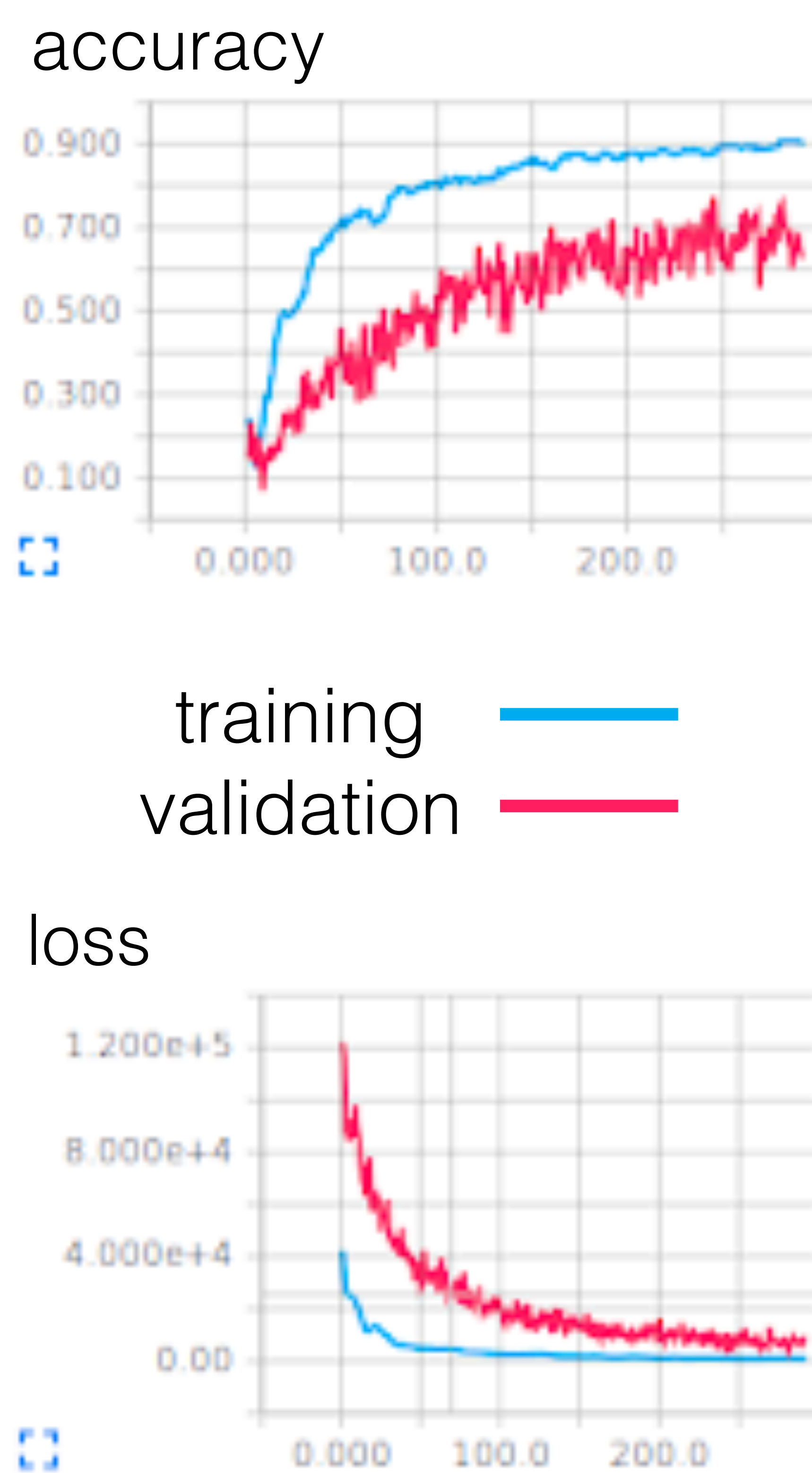
Apply separately at each location



Akkus et al., Journal of Digital Imaging, 2017

Convergence and Overfitting

- Batch size, epochs and convergence are the same as before
- Overfitting is more likely when number of parameters exceeds number of training data inputs
    - validation acc/loss much worse than training acc/loss
- Underfitting occurs when insufficient number of parameters or insufficient richness in data
    - training acc/loss is poor
- Can help guide network choices
- Empirical selection is still most common



Ensembles

- Results of optimisation (training) will depend on many things (e.g. initialisation, number of epochs, order of data, etc.)
- Most of the time it will end up in a local minima
  - Though that might be good enough
- Can combine together different trained models and average outputs to get an improved model
  - Initialised differently
  - Trained differently
  - Different architectures / parameters
- Well studied in traditional machine learning
  - Random forests = ensemble of decision trees
  - This is one of the best models in traditional ML
- Approach is also common and powerful for Deep Learning

Summary

- Convolutional Neural Networks (CNNs)
- Architecture:
    - convolution layer (size & number of filters)
    - pooling layer (max or average - downsizes)
    - stride (can also apply to convolution)
    - number of filters and number of parameters
- Convergence and overfitting
- Ensembles