Using Machine Learning Tools PG

Week 9 – Convolutional Neural Networks

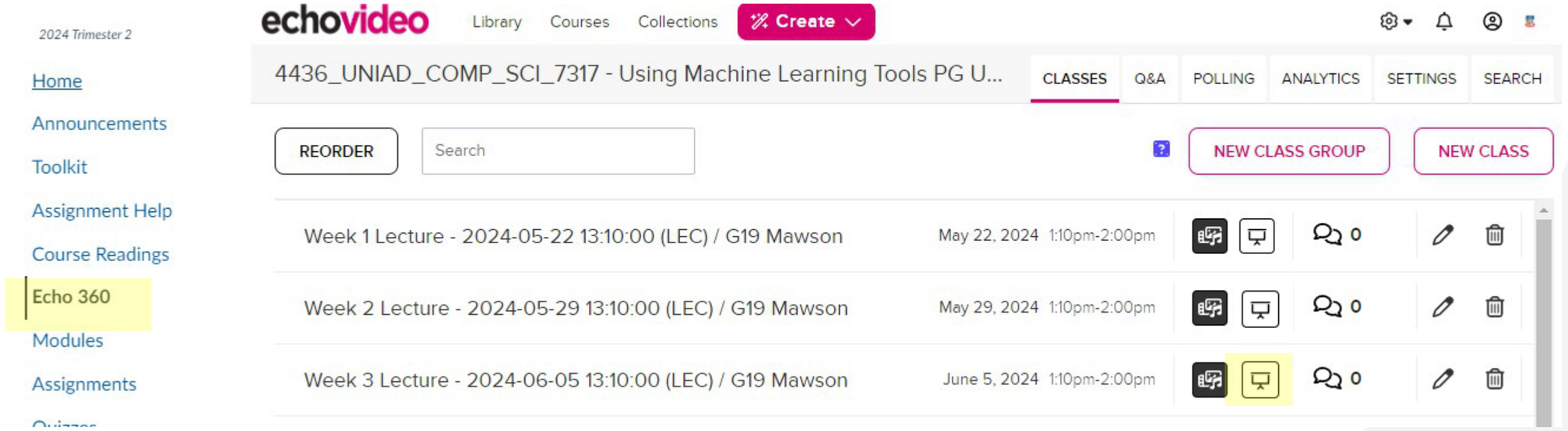
COMP SCI 7317

Trimester 2, 2024

Before we start…

There will be some **interactive elements** throughout this lecture.

Please participate by heading to **MyUni** > **4436_COMP_SCI_7317** > **Echo360** > **Lecture Week 9** > 



THE UNIVERSITY of ADELAIDE
150 YEARS

Last week… Neural Networks & Deep Learning

1. **Components of a Neural Network**
   - Neurons: non-linearities and activation functions
   - Network architecture
   - Parameters: connections, weights and biases
   - Number of parameters
   - Deep Neural Networks
2. **Loss functions, epochs, batches, optimisers and training**
3. **Classification and regression variants**

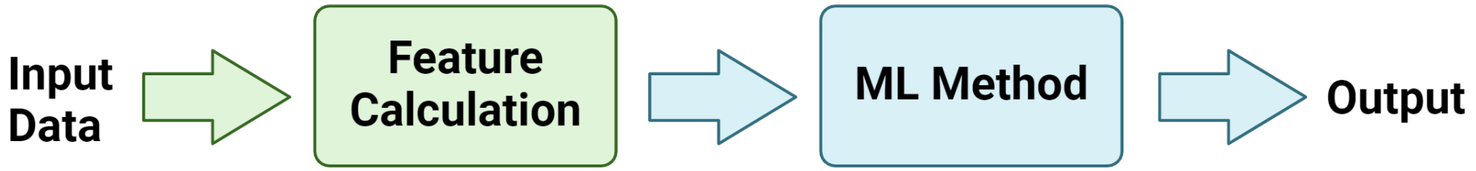
This week

1. Convolutional Neural Networks
2. CNN Structures
3. Pooling + Paramaters
4. Applications + Considerations

Convolutional Neural Networks

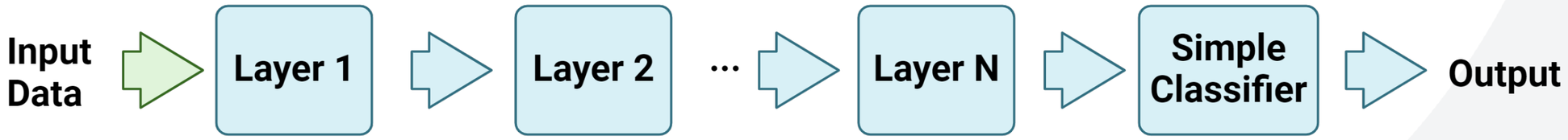
Introduction to Neural Networks

Traditional Machine Learning



- "Hand crafted features" i.e. pre-define the relevant features

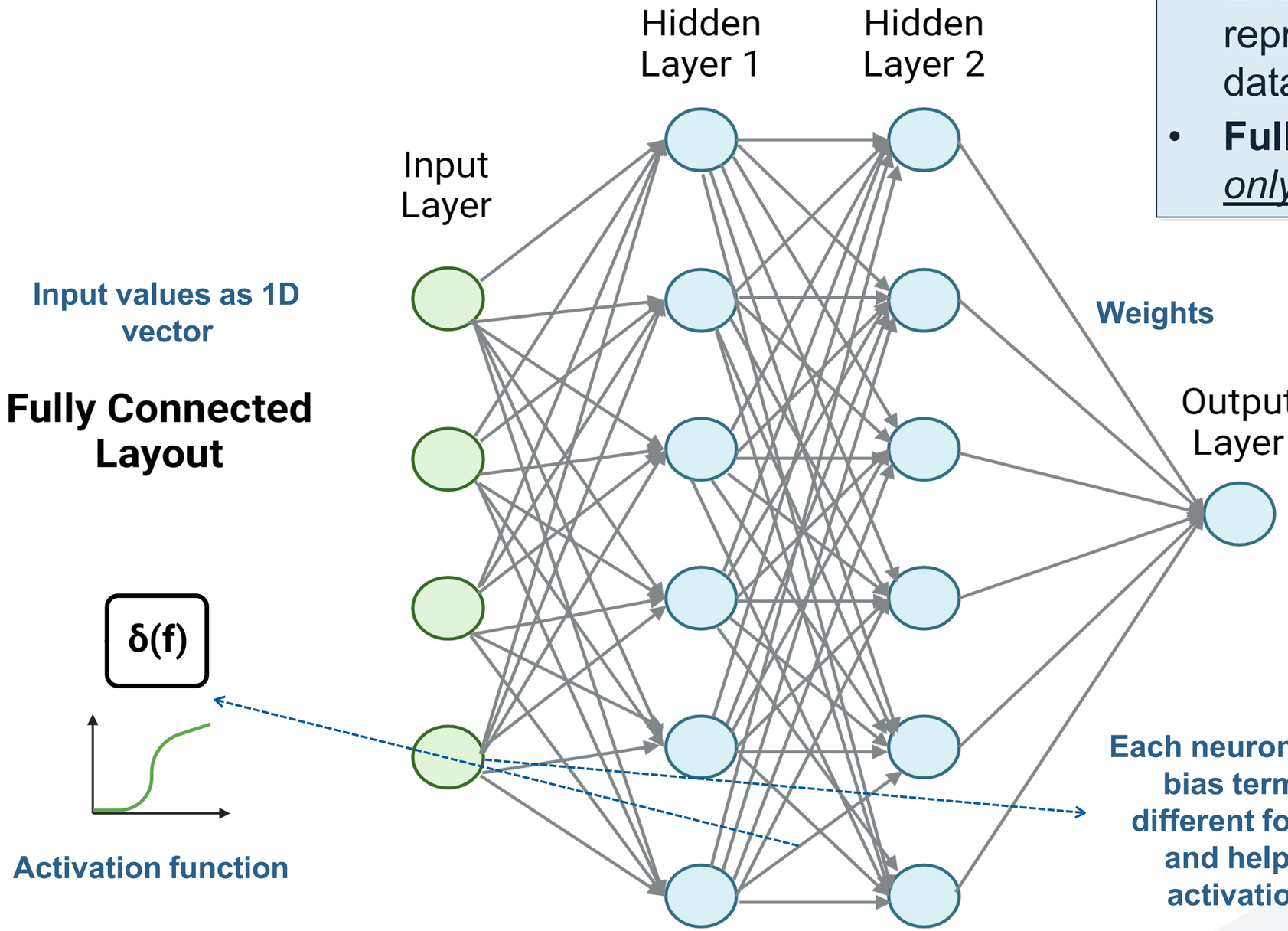
Deep Learning



- "Learned features": Network automatically learns relevant features
- Tend to have many more parameters + need much more data

Neural Network

- Commonly used architecture.
- Each green circle/input node represents a single value (i.e. data sample, pixel).
- **Fully connected:** Each layer *only* connects to next layer.

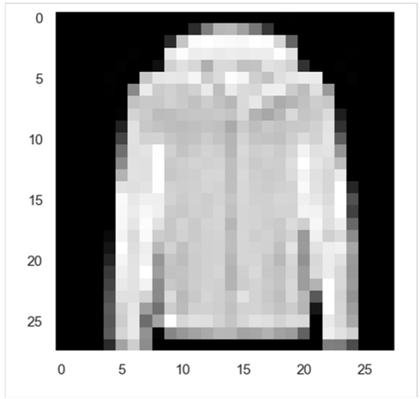


THE UNIVERSITY of ADELAIDE 150 YEARS

Neural Network: Image ‘Flattening’/Reshaping

Fully Connected Network

Input
28x28 pixel 2D image



Flattened 1D vector
(784 pixels/units)

Hidden layer 1
(300 nodes)

Hidden layer 1
(100 nodes)

Output
(10 nodes)

P1

P2

P3

P784

"T-shirt/top"

"Trouser"

"Ankle boot"

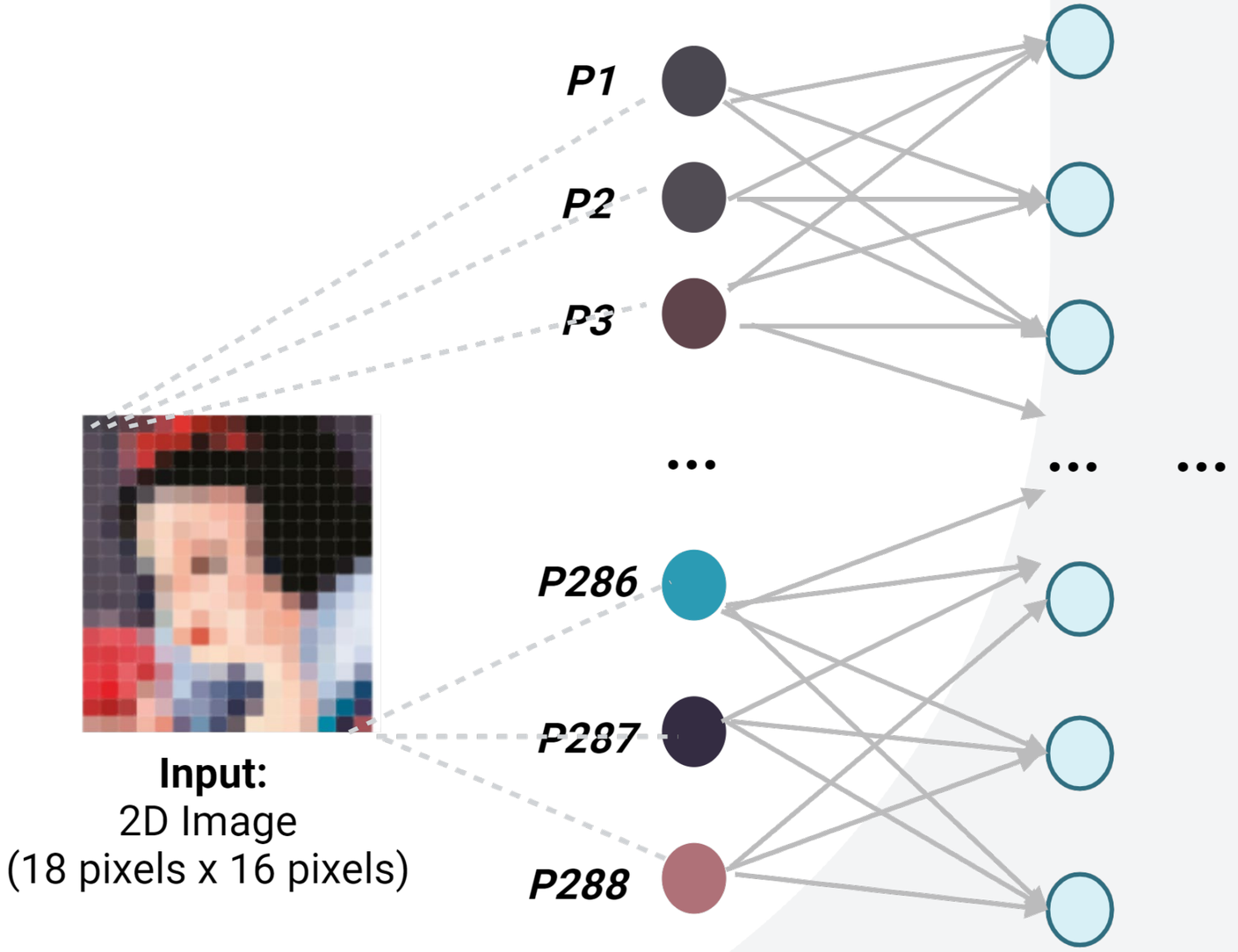
Loss fn: sparse categorical entropy

Loss of spatial information

Network loses spatial information about the image. Relationships between adjacent pixels, which could represent important features like edges or textures, are not directly encoded in the input.

Impact: Network may struggle to learn spatial patterns, as it treats each pixel independently rather than considering their spatial relationships.

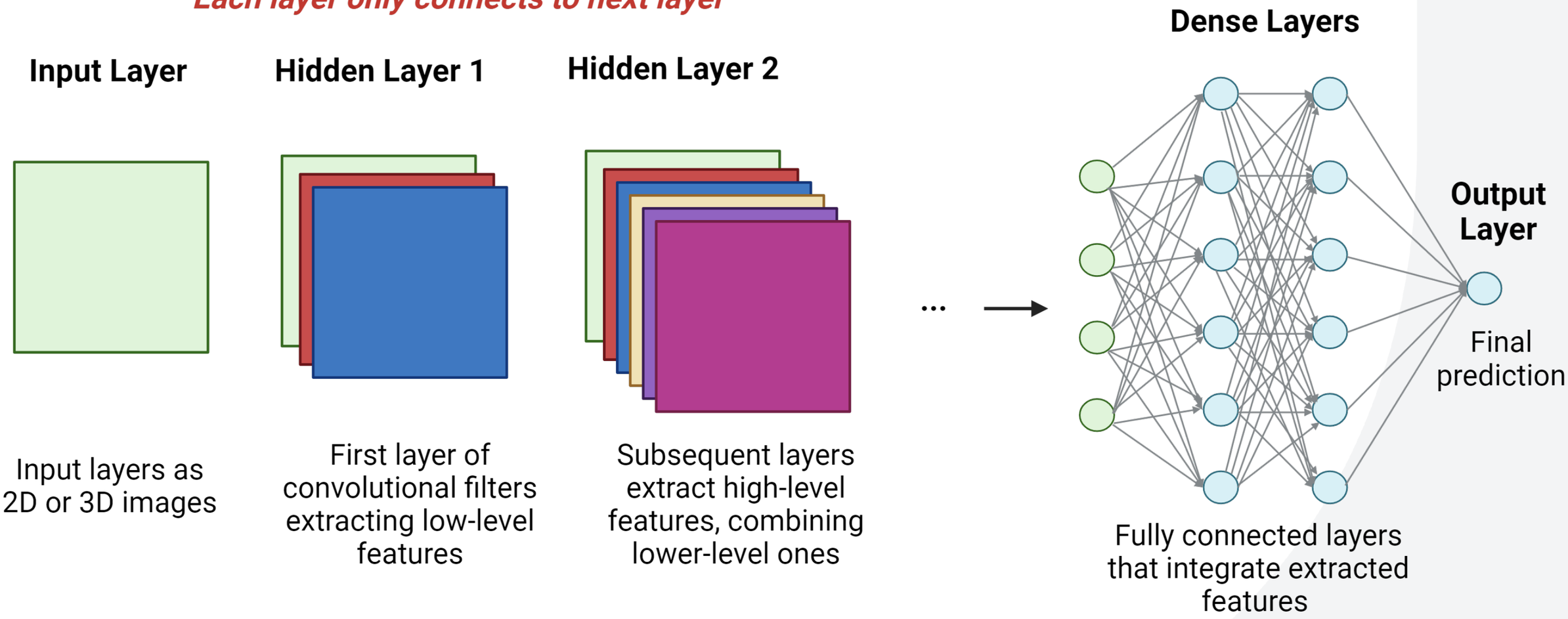
Solution: Use a linear operation that works with images and few parameters → ***Convolution***



Convolutional Neural Network

- Implicit bias term b not shown
- Activation function still applied

Each layer only connects to next layer

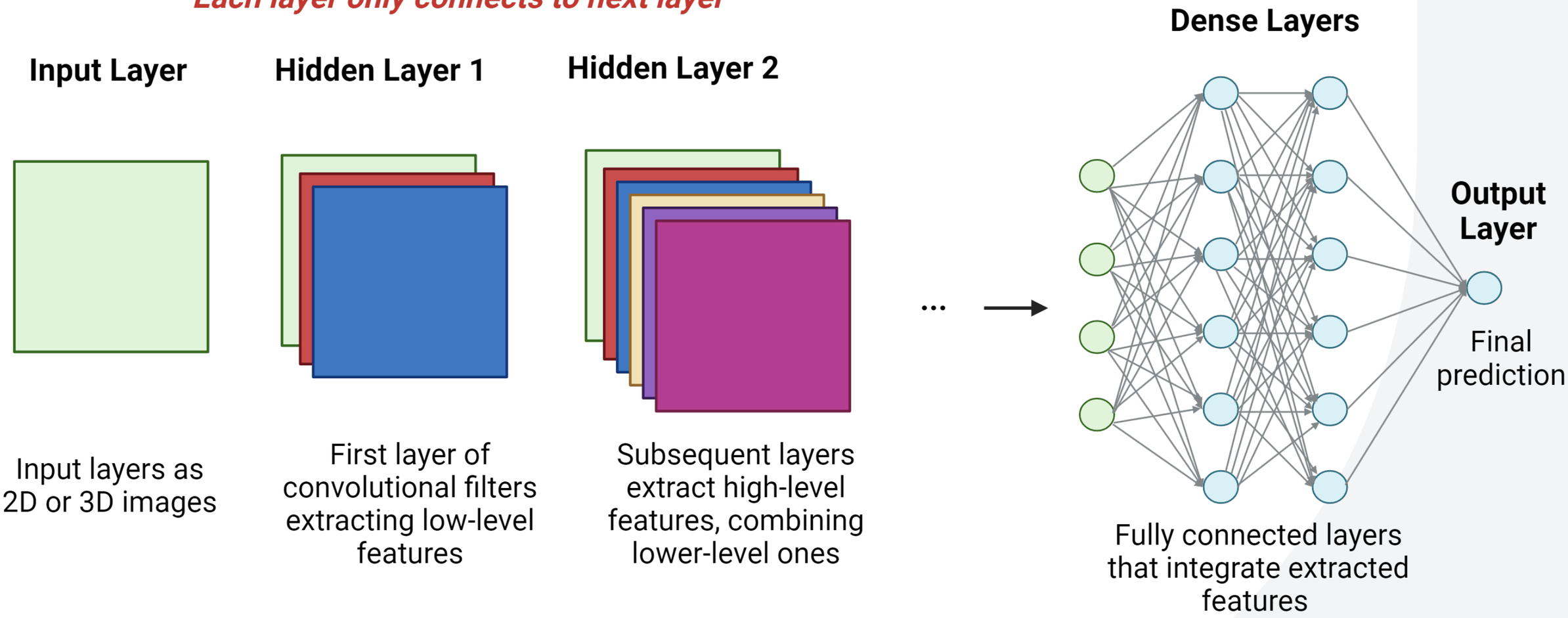


- A type of deep learning algorithm designed to process structured grid data such as images.
- Primarily used for image recognition, classification, and processing.

Convolutional Neural Network

- Implicit bias term b not shown
- Activation function still applied

Each layer only connects to next layer



- Convolutional and pooling layers act as feature extractors.
- Fully connected layers act as a classifier based on the extracted features.

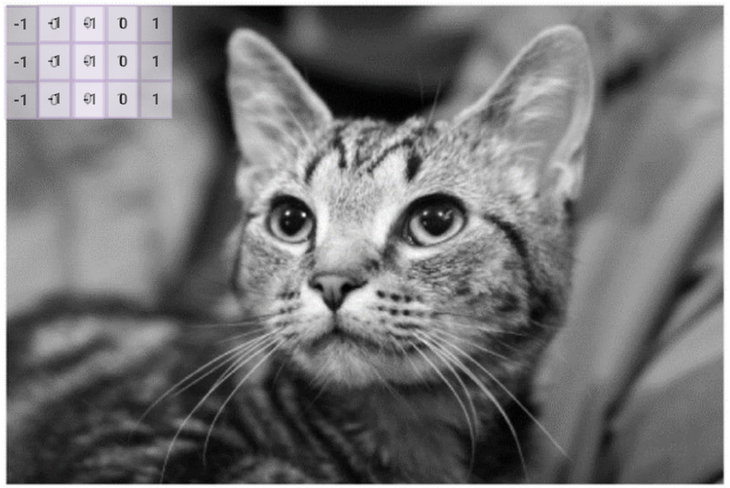
CNN Structures

Convolution

Convolution: A process that uses a small matrix (filter or kernel) to scan over an image and extract features. It involves sliding the filter over the input image, multiplying the overlapping values, and summing them to produce a new value in the output image.

- **Example:** edge detection (shown).

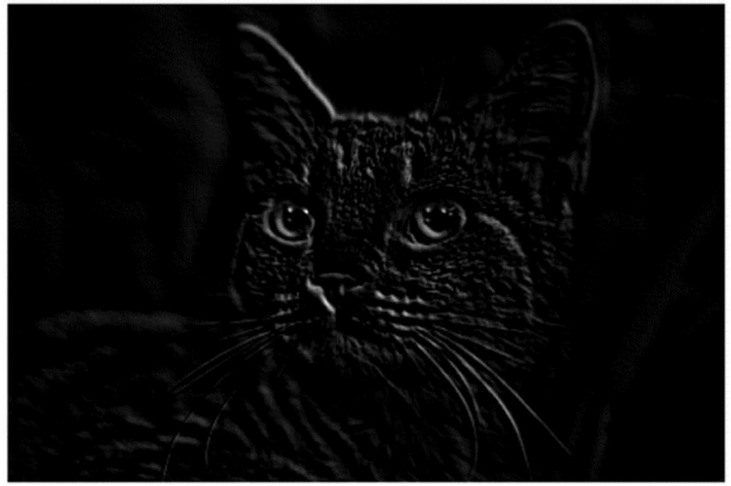
Original



Filter

-1	0	1
-1	0	1
-1	0	1

Convolved



Many different types of filters are available!

Convolutional Layer

Image (5x5 pixels)

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

Zero values clustered together
→ Spatial relationship

Filter

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

Generally chosen by NN

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

Convolutional layer uses a set of filters (also known as kernels) that slide over the input data. Output of each convolution produces a feature map

Convolutional Layer

Image (5x5 pixels)

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

(4*4) + (8*-2) + (2*2) + (4*-1) = 0

Starting with centre value of the filter/kernel aligned with the top-left most pixel in the image

Convolutional Layer

Image (5x5 pixels)

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

Values outside of the filter/kernel ignored → Same effect as 0 padding

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

(4*4) + (8*-2) + (2*2) + (4*-1) = 0

Convolutional Layer

Image (5x5 pixels)

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

(4*4) + (8*-2) + (2*2) + (4*-1) = 0

Alternative is to constrain filter/kernel to inside the image, but this makes output smaller as you can't align the centre with the edge pixel.

Convolution Layer

Image (5x5 pixels)

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

(4*-2)+(8*4)+(3*-2)+(2*-1)+(4*2)+(5*-1) = 19

Moving filter/kernel so centre is aligned with next pixel in the image (to the right)

Convolutional Layer

Image (5x5 pixels)

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

Repeat until the image is fully convolved with the filter/kernel

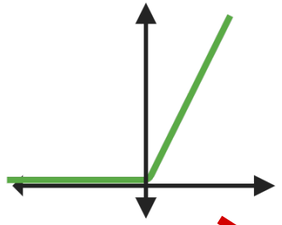
Convolutional Layer

Image (5x5 pixels)

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

Applying convolution is a linear operation. To introduce non-linearity into the model to learn more complex patterns, an activation function (e.g., ReLU) is applied element-wise after convolution is performed.

i.e. ***Convolution → Output feature map → Activation function***

Convolutional Layer

Image (5x5 pixels)

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

Bias

+1

Output

0+1	19+1	-11+1	15+1	-13+1
-7+1	27+1	-15+1	22+1	-18+1
-6+1	38+1	-31+1	27+1	-17+1
9+1	26+1	-26+1	13+1	-7+1
8+1	5+1	-6+1	0+1	0+1

We can also have bias terms, which helps the model learn an additional offset to improve its ability to fit the data (similarly to other neural networks). This adds a value to each element in output feature map. Example on previous slides equivalent to bias of 0.

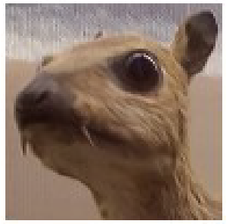
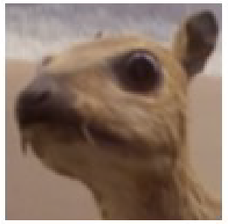
Which answer is correct?

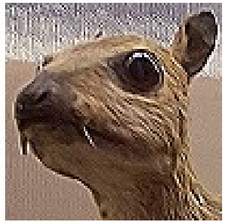
What is the main purpose of an activation function in a neural network?

- To initialise weights
- To introduce non-linearity
- To normalise input data
- To optimise the loss function

Convolutional Layer

- Only uses a small portion of the image at a time, so tends to be quite computationally efficient operation.
- Many types of filters/kernels exist, depending on what is most suitable to extract features.
  - **Examples:** edge detectors, blur filters, etc.
- ***Stride*** determines how much the filter moves at each step (usually 1 for convolution), and ***padding*** helps in controlling the output size by adding extra pixels around the border of the input.
- Size of filter/kernel affects the granularity of the feature extraction (usually 3x3)
  - Smaller filters capture fine details
  - Larger filters capture broader patterns
  - Better to have smaller filters + more layers

Original	*Gaussian Blur*
$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$
	

Sharpen	*Edge Detection*
$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$
	

Convolutional Neural Network

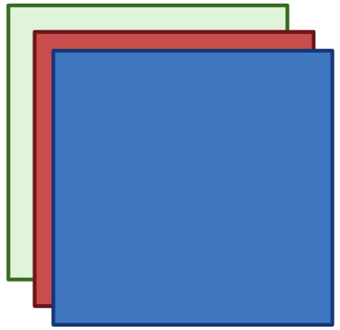
- Implicit bias term b not shown
- Activation function still applied

Each layer only connects to next layer

Input Layer

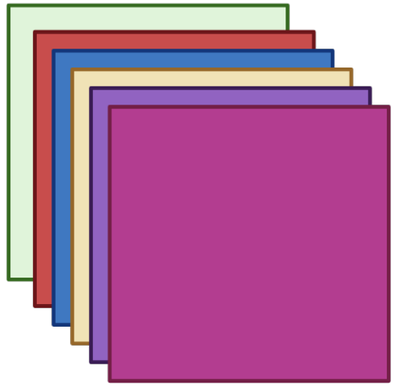
Input image

Hidden Layer 1



Convolution applied to generate (low-level) feature maps. Different convolution filters/kernels applied to generate multiple different feature maps

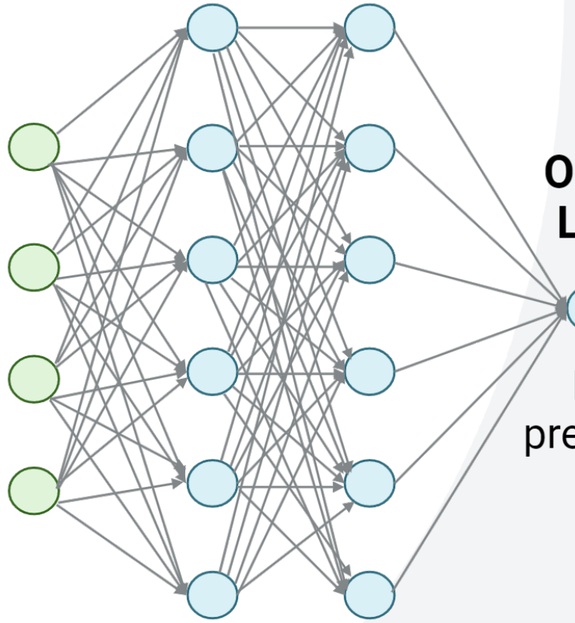
Hidden Layer 2



Subsequent layers extract high-level features, combining lower-level ones

...

Dense Layers

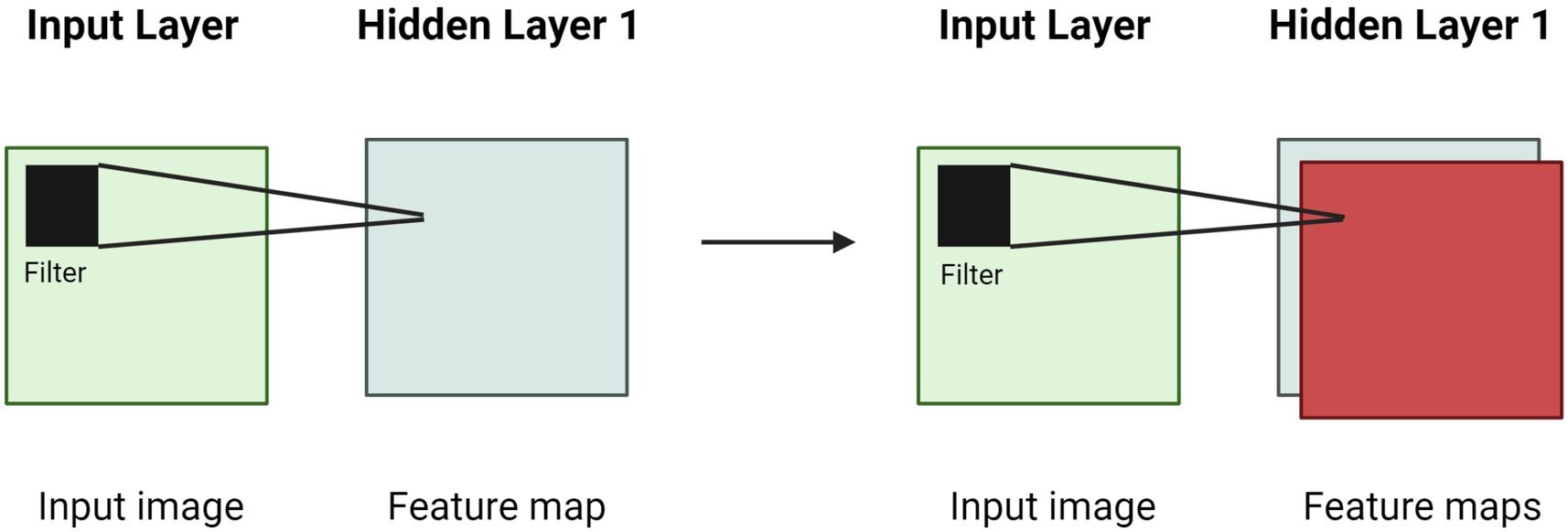


Output Layer

Final prediction

Fully connected layers that integrate extracted features

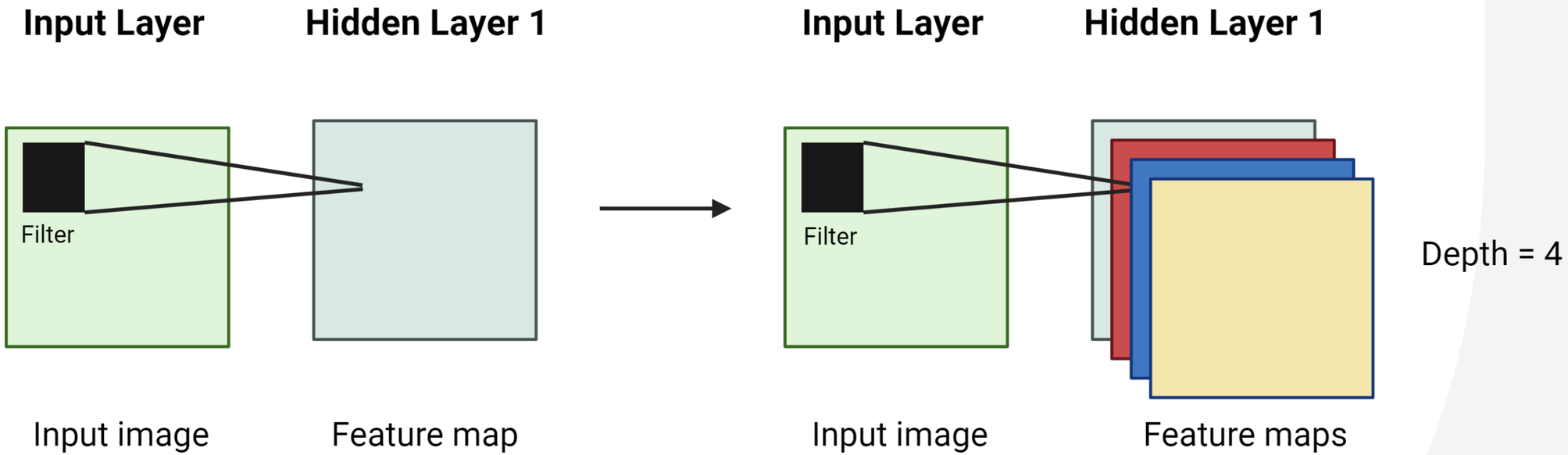
Convolutional Neural Network



Convolution operation performed by sliding same filter/kernel over image to produce output feature map

Different convolution filters/kernels applied to generate multiple different feature maps

Convolutional Neural Network



Convolution operation performed by sliding same filter/kernel over image to produce output feature map

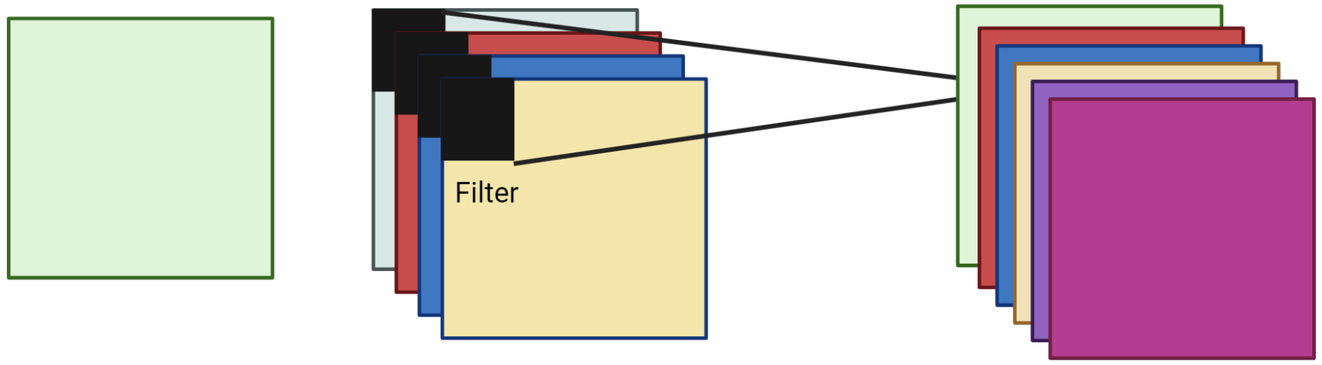
Different convolution filters/kernels applied to generate multiple different feature maps

Values in each hidden layer are analogous to values we would have for each node in a neural network, which are used in next layer to do calculations

Convolutional Neural Network

The input might not be scalars but instead RGB values or a 3D image (a stack of ‘channels’). In this case, we treat it as a layer with a depth of 3 (e.g., filter dimensions 3x3x3).

Input Layer **Hidden Layer 1** **Hidden Layer 2**



Input image

Feature maps

Filters slides over input image to create feature maps.

In hidden layer 1, layers have extra dimension (depth) equal to number of feature maps.

In hidden layer 2, convolutions are applied across all feature maps from hidden layer 1, producing new feature maps that combine information from all previous maps.

Convolution applied to hidden layers: Each filter applied across all depths. Each filter results in a new feature map in next layer.

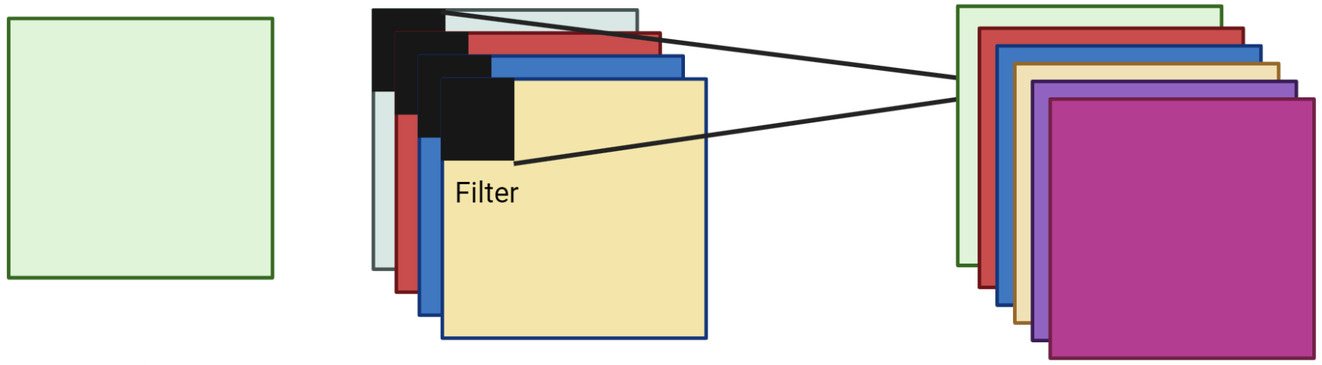
- Depth 4 in hidden layer 1 → 4 filters applied
- Depth 6 in hidden layer 2 → 6 filters applied

Once we get to the next layer (hidden layer 2), we are applying convolution to multiple feature maps from the previous layer. This allows the network to learn more complex features by combining information from different feature maps.

Convolutional Neural Network

The input might not be scalars but instead RGB values or a 3D image (a stack of ‘channels’). In this case, we treat it as a layer with a depth of 3 (e.g., filter dimensions 3x3x3).

Input Layer **Hidden Layer 1** **Hidden Layer 2**



Input image

Feature maps

Filters slides over input image to create feature maps.

In hidden layer 1, layers have extra dimension (depth) equal to number of feature maps.

In hidden layer 2, convolutions are applied across all feature maps from hidden layer 1, producing new feature maps that combine information from all previous maps.

Convolution applied to hidden layers: Each filter applied across all depths. Each filter results in a new feature map in next layer.

- Depth 4 in hidden layer 1 → 4 filters applied
- Depth 6 in hidden layer 2 → 6 filters applied

Once we get to the next layer (hidden layer 2), we are applying convolution to multiple feature maps from the previous layer. This allows the network to learn more complex features by combining information from different feature maps.

Pooling + Parameters

Pooling Layer

Pooling layers (down-sample layers) perform a down-sampling operation, reducing the spatial dimensions (width and height) of the input feature map while retaining the important features.

Max pooling layer

Original image

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

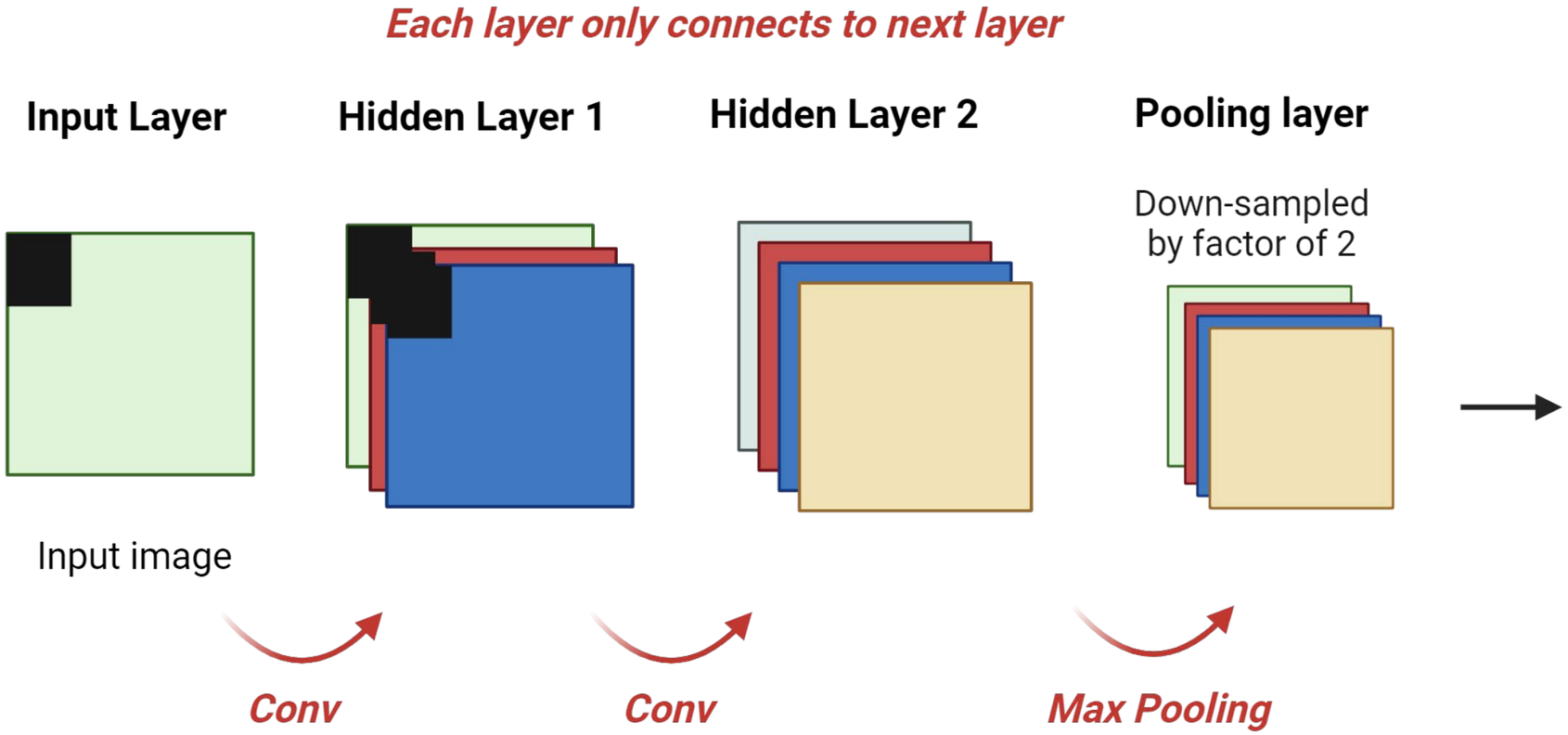
Output

8	5
9	7

Take maximum value within a 2x2 filter
Move in steps of 2 (***stride***)
Down samples image by factor of 2

- **Useful for reducing the number of parameters required in a CNN.**
- **Pooling layers have no parameters.**
  - The values in the filters (or kernels) for convolution are parameters that are learned during training.
  - These are updated through optimisation/backpropagation, similar to weights in traditional neural networks.
  - Different feature maps are generated during optimisation due to updates in the filters.
- **Pooling doesn't have to be max pooling; average pooling is also an option.**

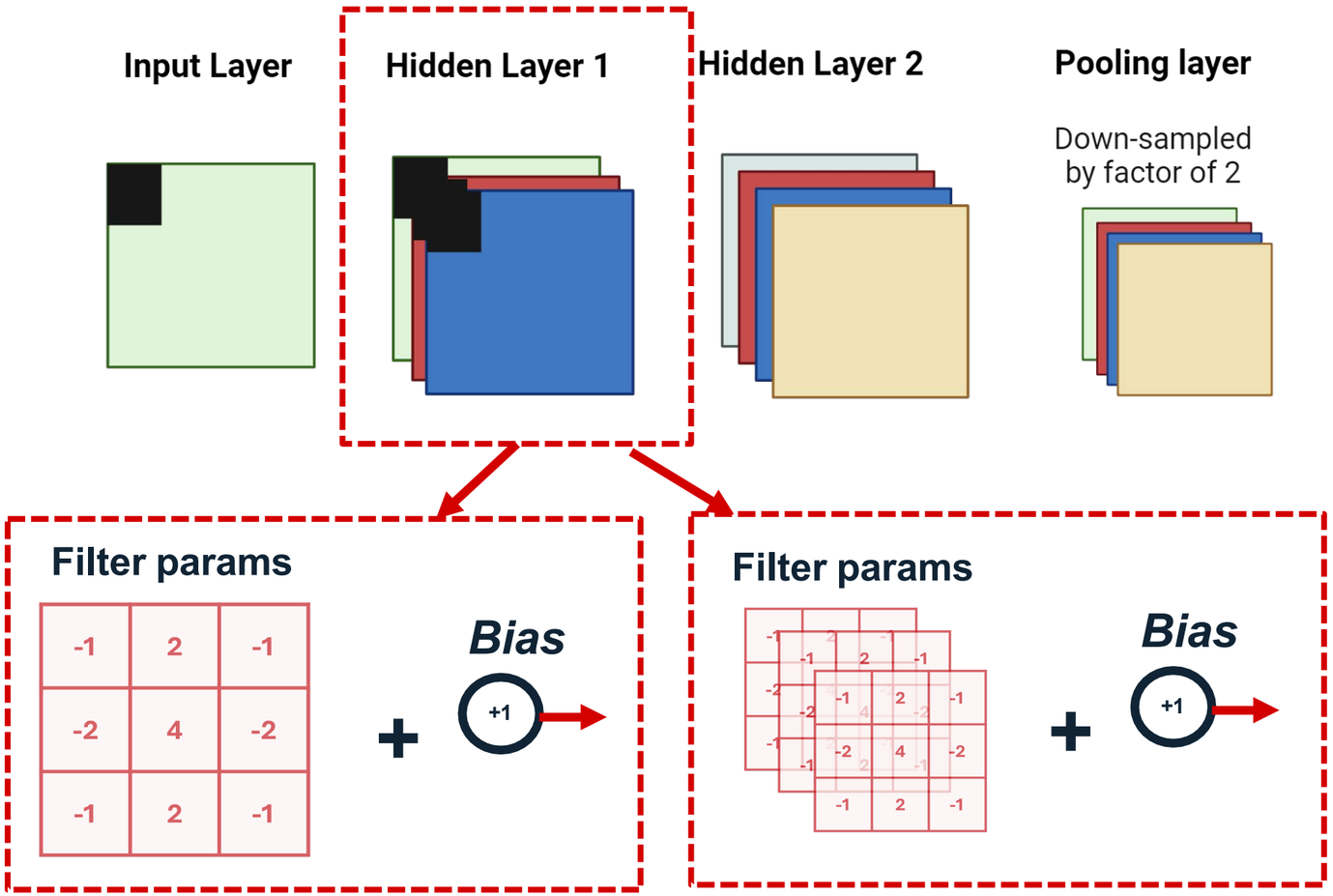
Convolutional Neural Network



Convolutional layers are often interspersed with max pooling layers

Parameters

- Values in the filters/kernels for convolution = parameters learnt during training.
- Updated through optimisation/backpropagation, generating different feature maps



For a 3x3 filter

Scalar input image (depth = 1)

Params = Filter Size + Bias

3*3*1+1 = 10

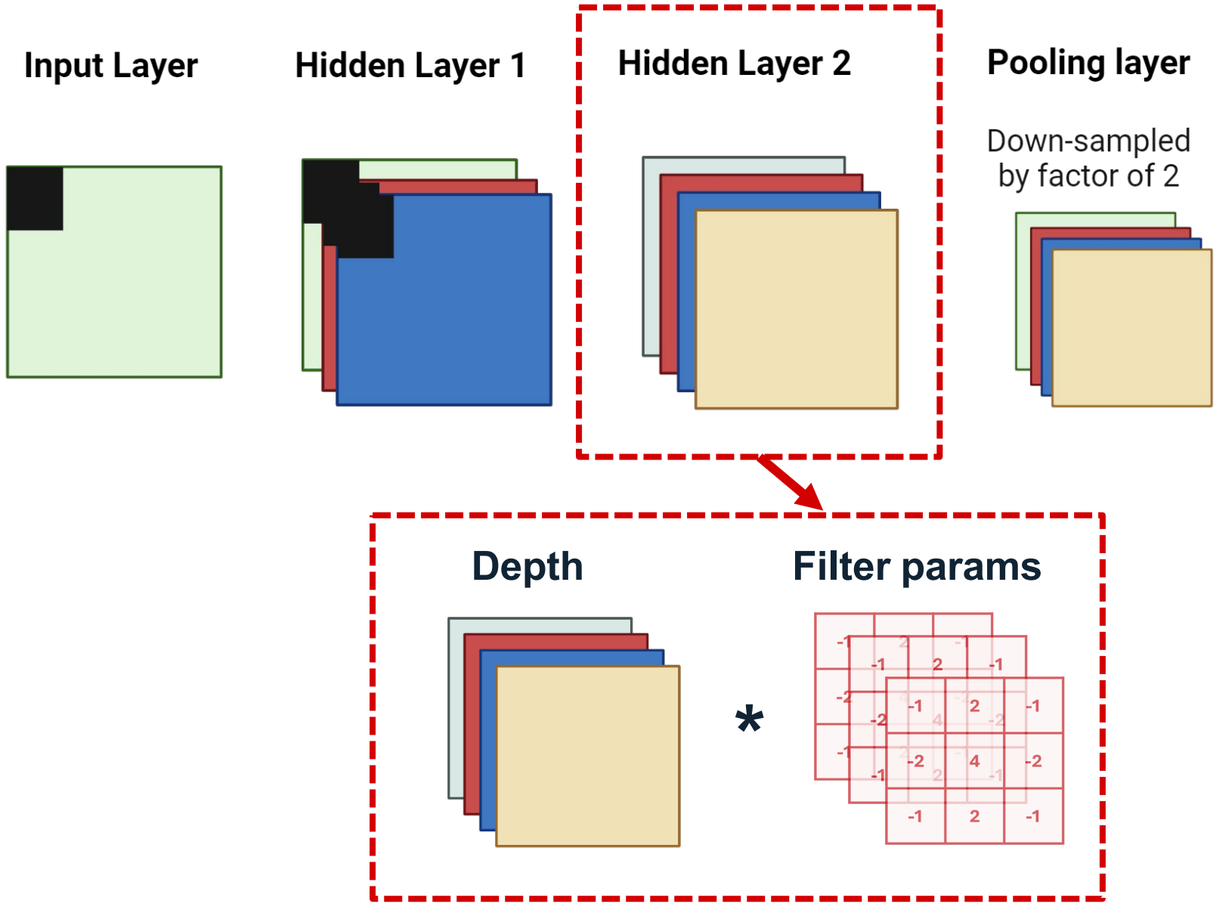
Hidden layer 1

Parameters in each filter:

(3*3*3+1) = 28

Parameters

- Values in the filters/kernels for convolution = parameters learnt during training.
- Updated through optimisation/backpropagation. Different feature maps are generated during optimisation due to updates in the filters.



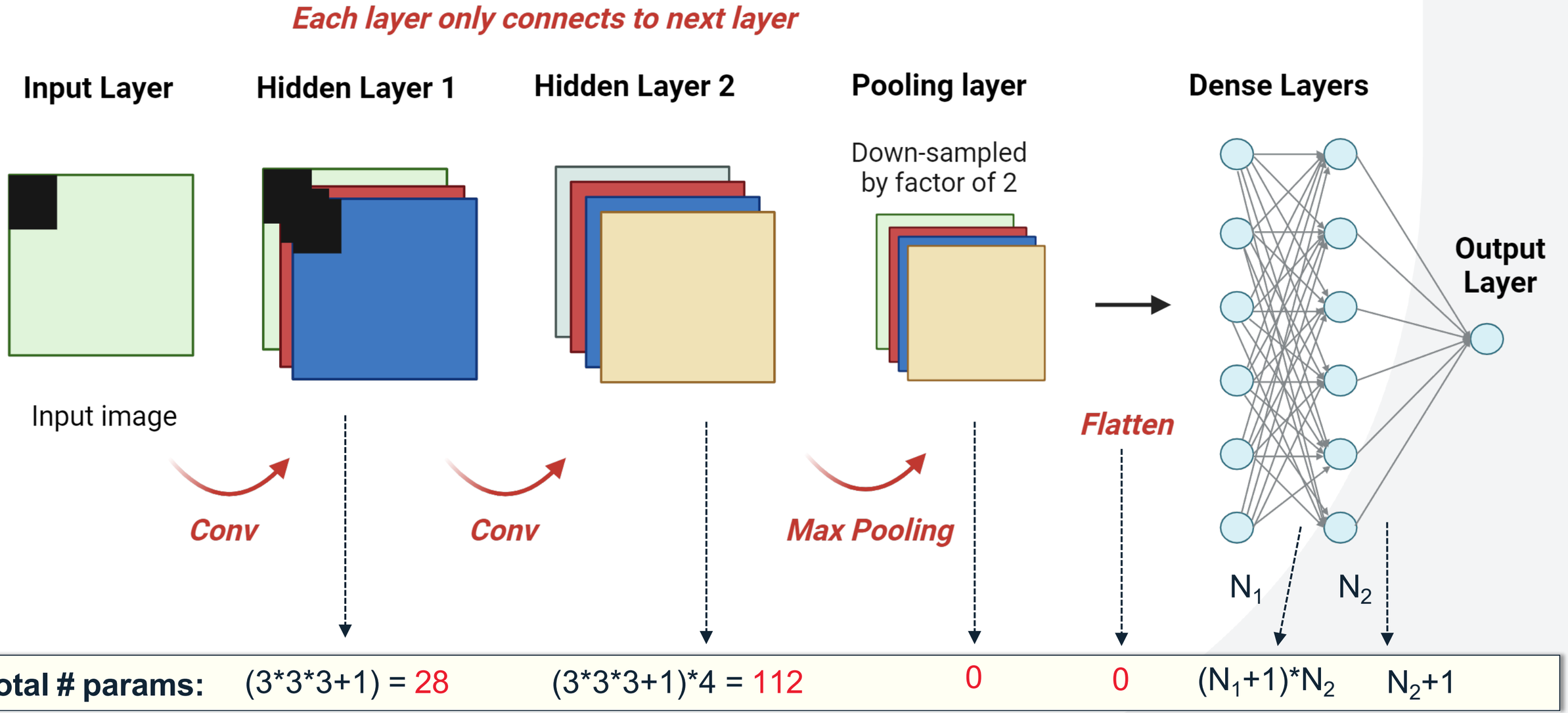
For a 3x3 filter in hidden layers

Hidden layer 2

filters/depth * #params in each filter:

(3*3*3+1)*4 = 28*4 =112

Convolutional Neural Network



Total # params:	(3*3*3+1) = 28	(3*3*3+1)*4 = 112	0	0	$(N_1+1)*N_2$	N_2+1

Which answer is correct?

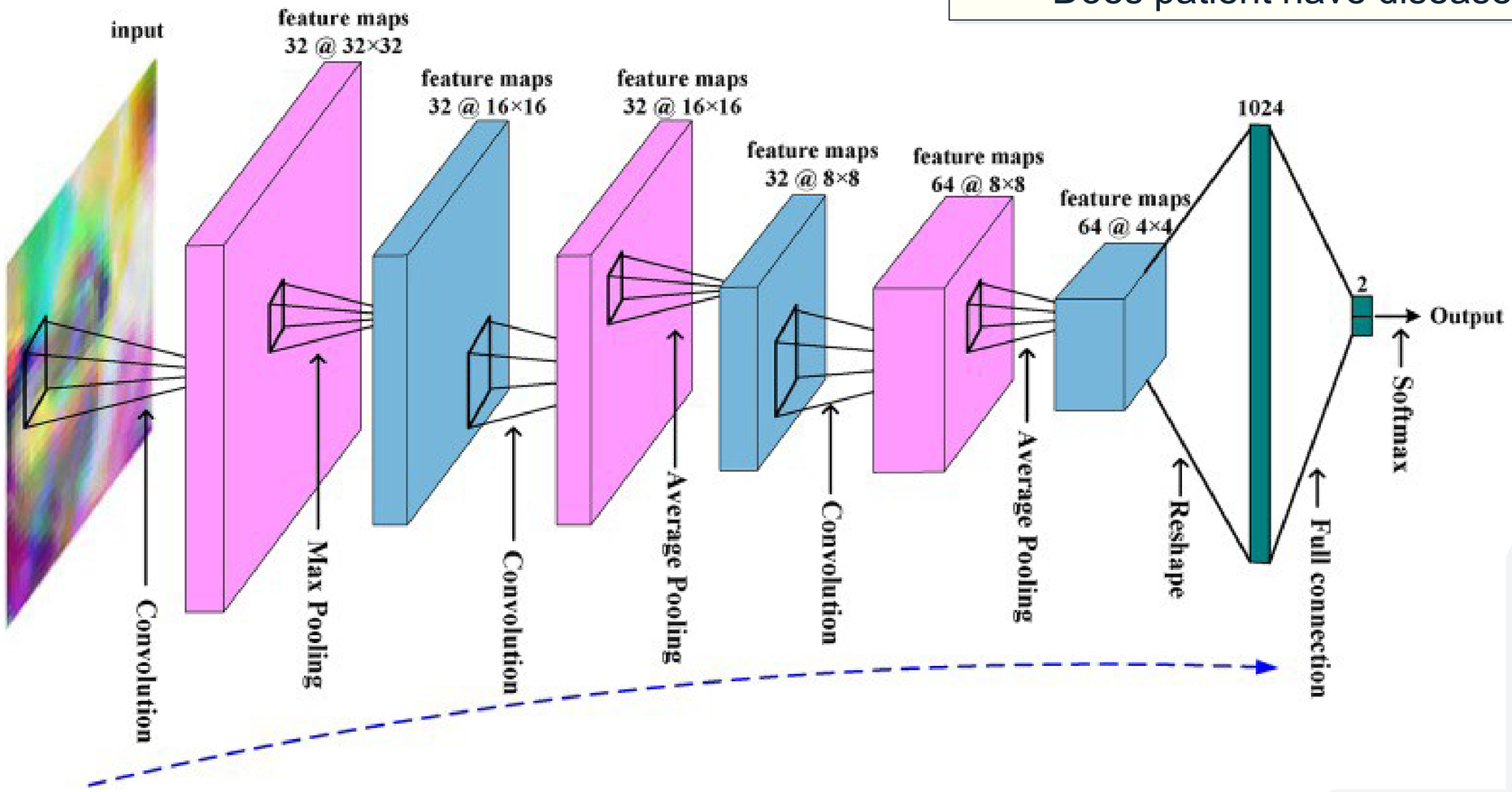
What is the main purpose of an activation function in a neural network?

- To initialise weights
- To introduce non-linearity
- To normalise input data
- To optimise the loss function

Applications + Considerations

Example: Classification

Does patient have disease or not?

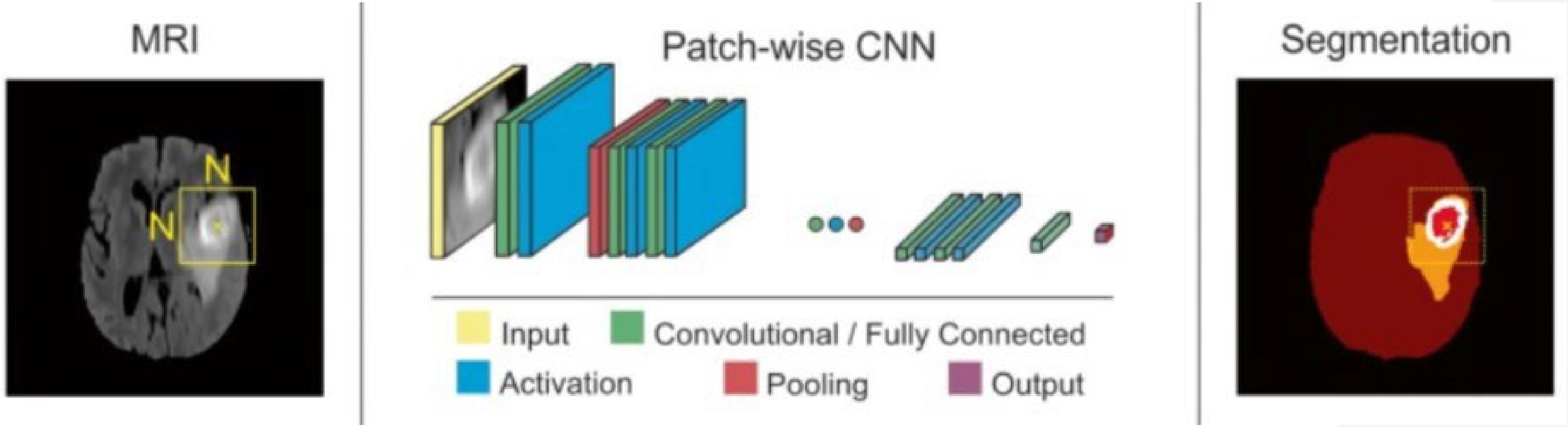


Lin et al., *Convolutional Neural Networks-Based MRI Image Analysis for the Alzheimer's Disease From Mild Cognitive Impairment*, Front. Neurosci,m 2018

Prediction

Example: Segmentation

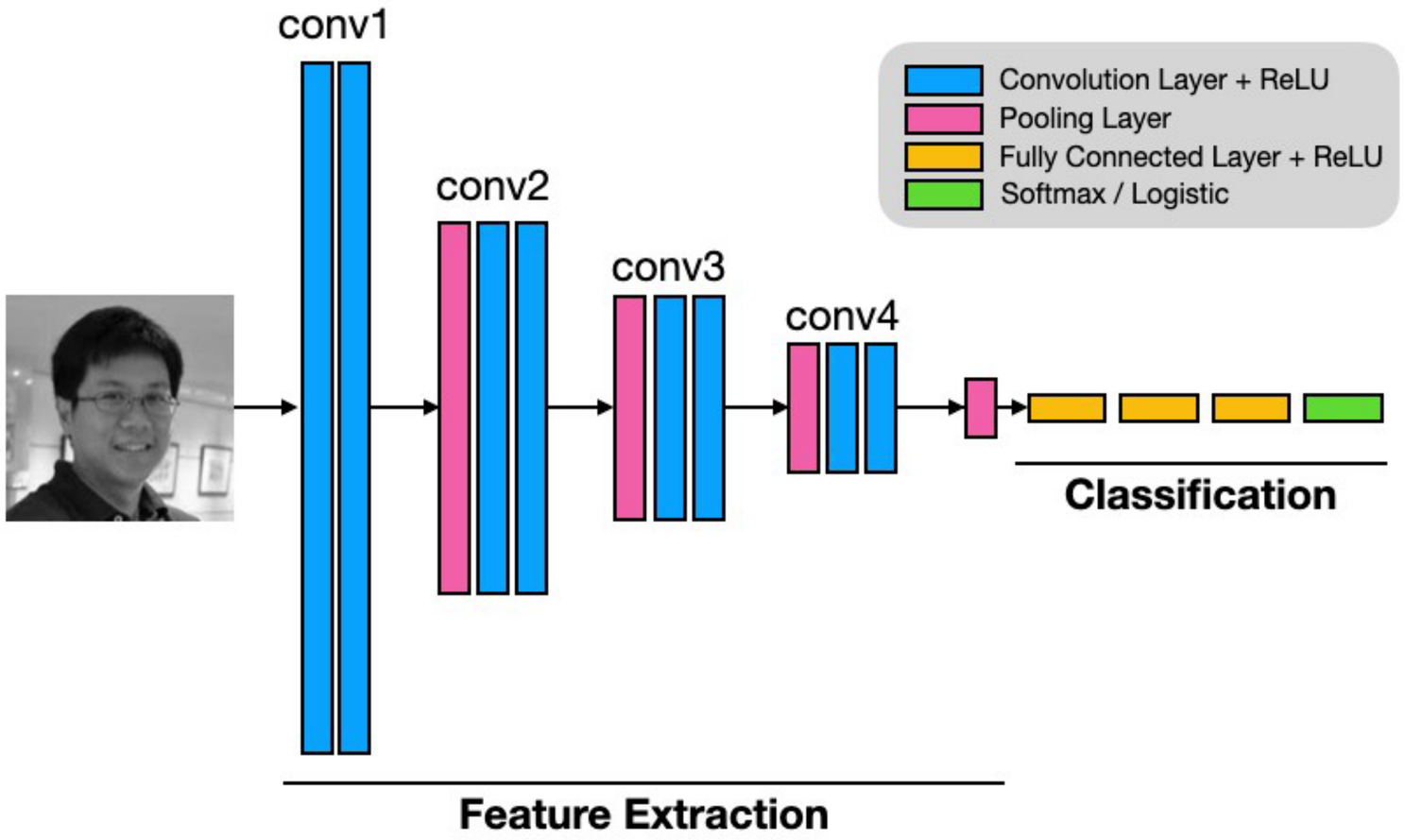
- What type of brain tissue (coloured) is present at each point/location in image?
  - Like a point-by-point classification
- Patch-based application, meaning this methodology applied across different patches/chunks (NxN) of the image at a time and give label for each centre point



Akkus et al., *Deep Learning for Brain MRI Segmentation: State of the Art and Future Directions*, 2017

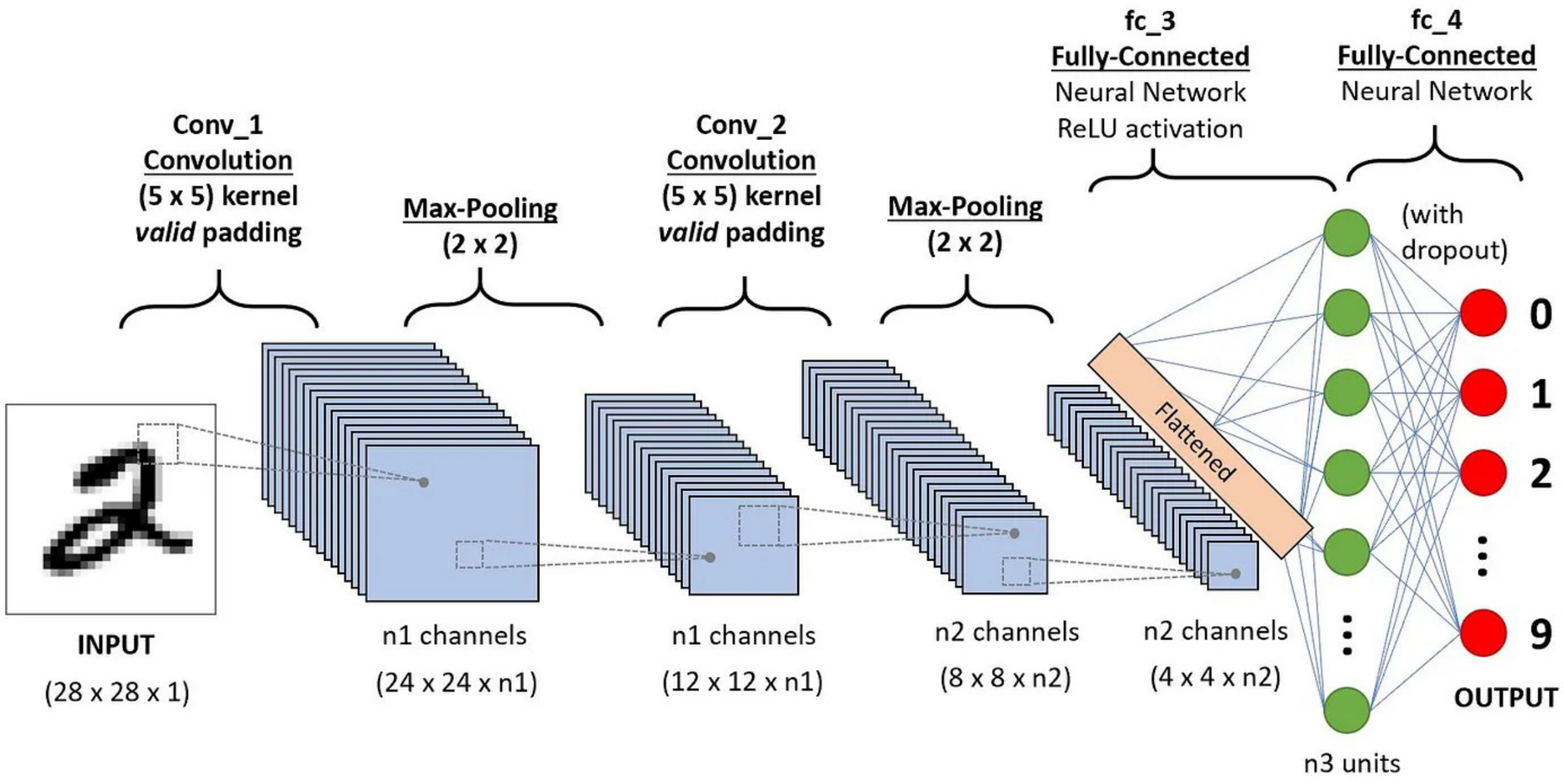
Example: Facial Recognition

- Extracts unique facial features from image and decides if it belongs to person A, B or C.



Lee., *Code Magazine*, 2022

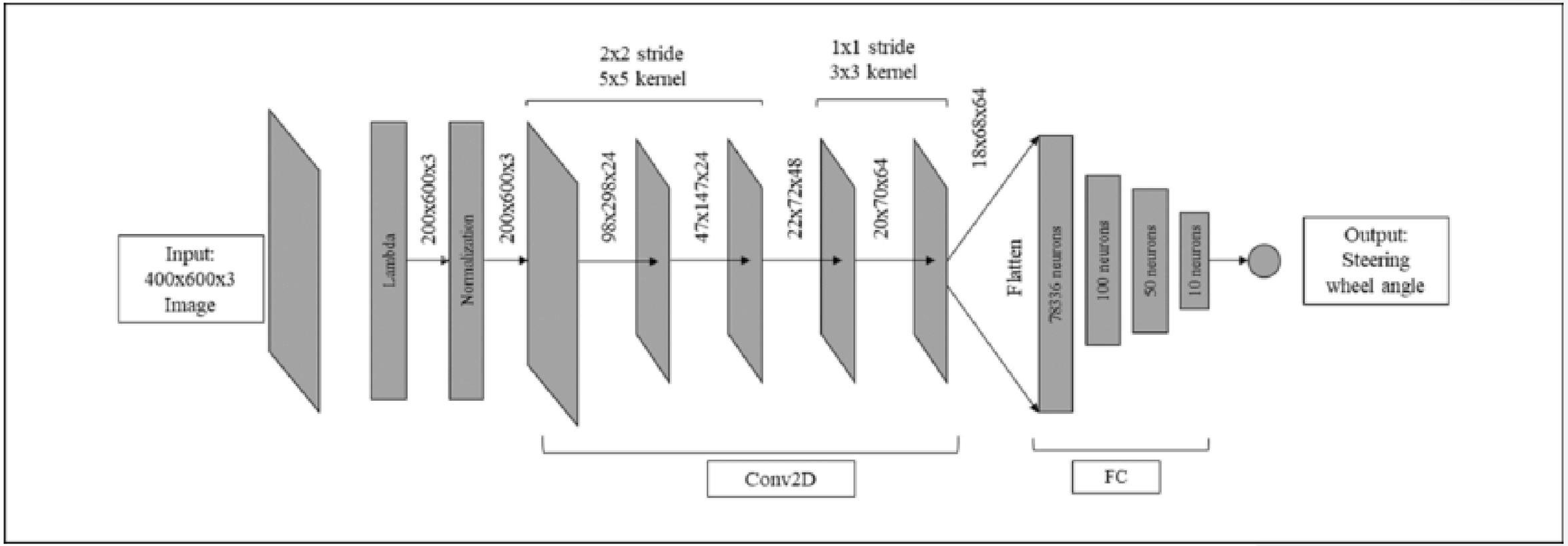
Example: Text Recognition



Kelta., *DataCamp*, 2023

Example: Autonomous Vehicles

Uses CNN to detects hazards on the road and produces a segmented image to assist the vehicle in navigating safely



Islam et al., *TRR*, 2019

Convergence and Overfitting

- Batch size, epochs and convergence are the same as discussed before
- Overfitting 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 poor
- Can help guide network choices
- Empirical selection most common

Ensembles

- Results of optimisation (training) will depend on many things (initialisation, number of epochs, order of data, etc.)
- Most of the time it will end up in a local minima (this 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
  - i.e random forest = ensemble of decision trees (one of the best ML models)
- Approach also common and powerful for deep learning
  - Usually, would ensemble just a few in deep learning

Which answer is correct?

What is the main purpose of an activation function in a neural network?

- To initialise weights
- To introduce non-linearity
- To normalise input data
- To optimise the loss function

Summary

1. Convolutional Neural Networks
2. CNN Structures
3. Pooling + Paramaters
4. Applications + Considerations

Questions?

dhani.dharmaprani@adelaide.edu.au