Convolutional Neural Networks

COSC 410: Applied Machine Learning

Spring 2022

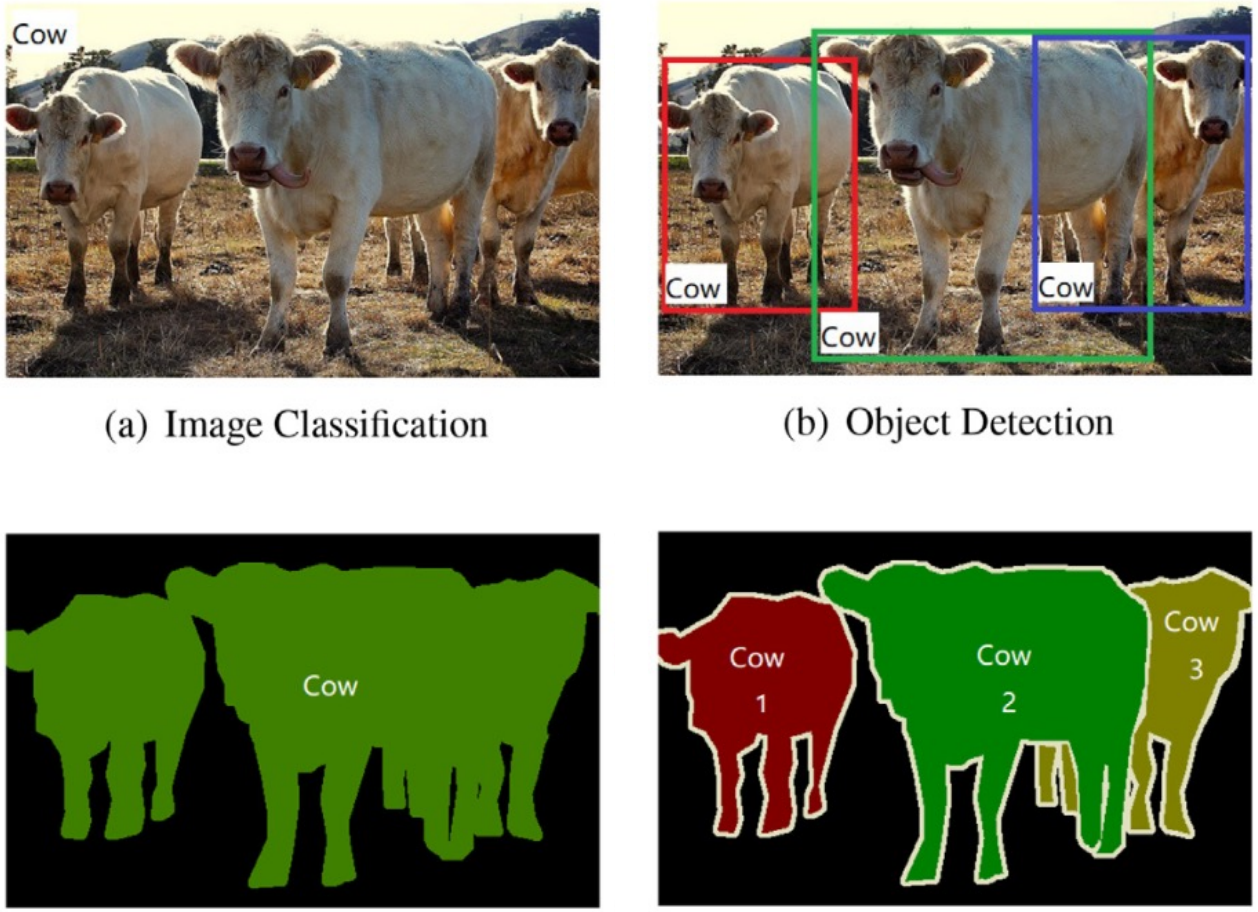
Prof. Apthorpe

Outline

- Why CNNs?
- 1D & 2D Convolutions
- Convolution hyperparameters
  - Filter size
  - Padding
  - Stride length
- Convolutional Networks (CNNs)

Motivating Tasks

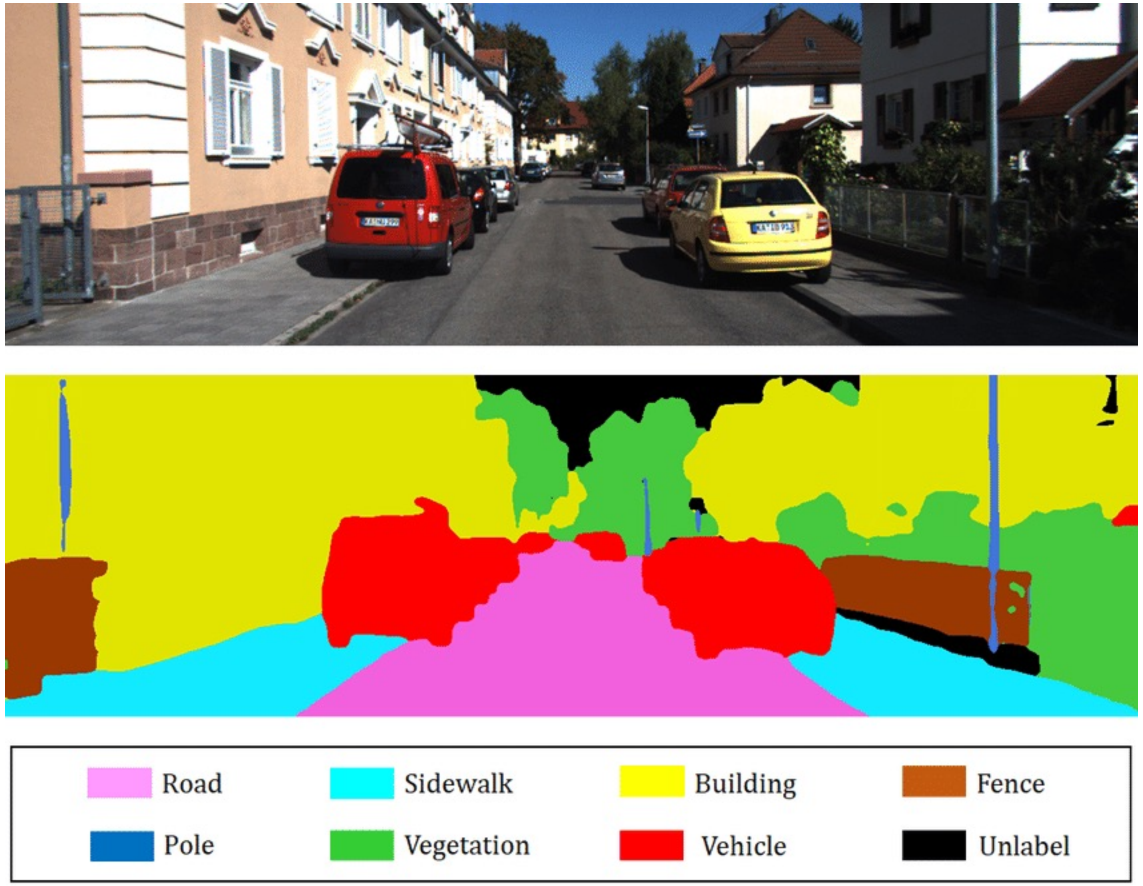
- **Image classification**
  - Predict labels per **image**
- **Object Detection**
  - Predict labels & bounding boxes
- **Image segmentation**
  - Predict labels per **pixel**



(a) Image Classification (b) Object Detection
(c) Semantic Segmentation (d) Instance Segmentation

Motivating Tasks

- **Image classification**
  - Predict labels per **image**
- **Object Detection**
  - Predict labels & bounding boxes
- **Image segmentation**
  - Predict labels per **pixel**

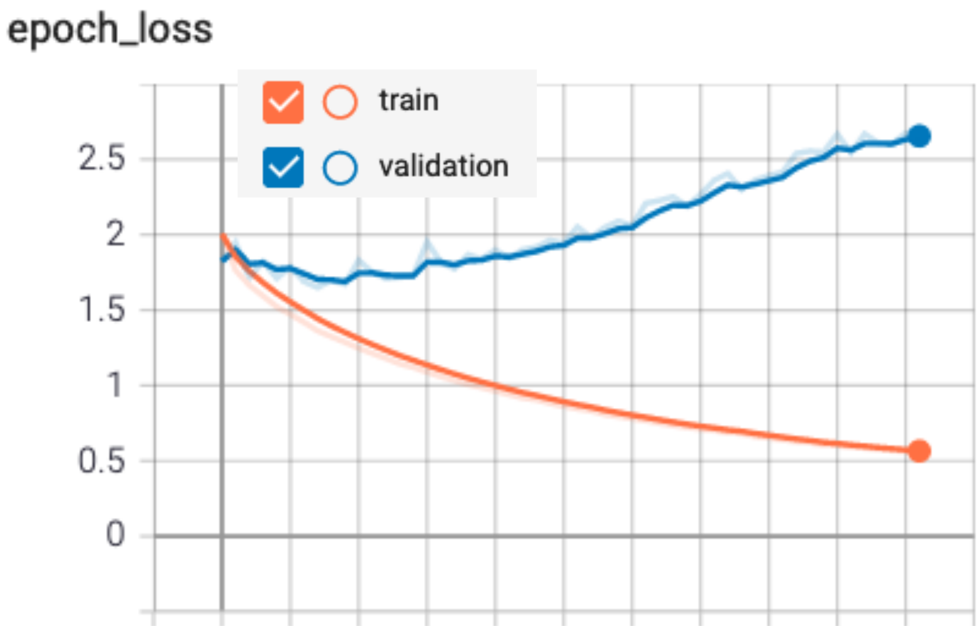


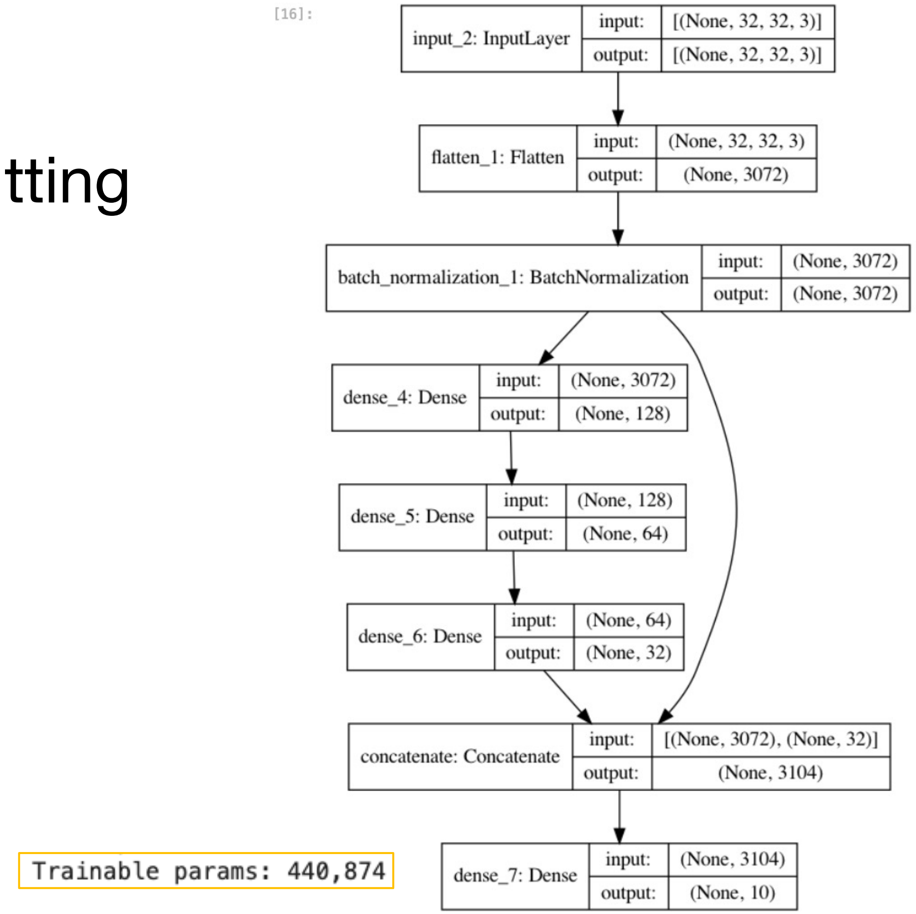
Problems with FNNs

- Thoughts?

Problems with FNNs

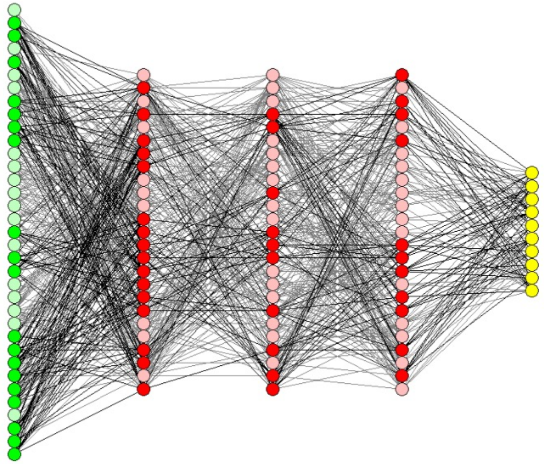
- Too many parameters
  - Even "small" FNNs are prone to overfitting

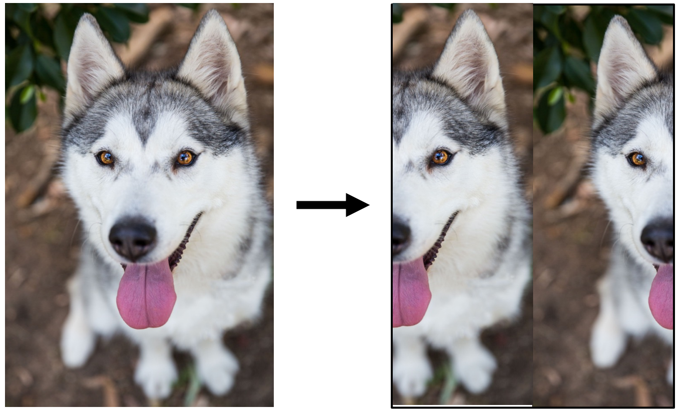




Problems with FNNs

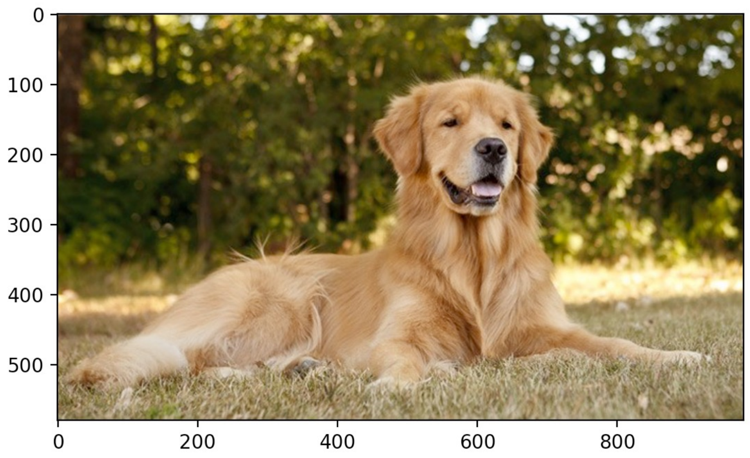
- Does not account for feature **locality**
  - *Assumes feature order is arbitrary*
  - All features treated equivalently with respect to each other
  - Sensitive to shifting data via rotation or translation

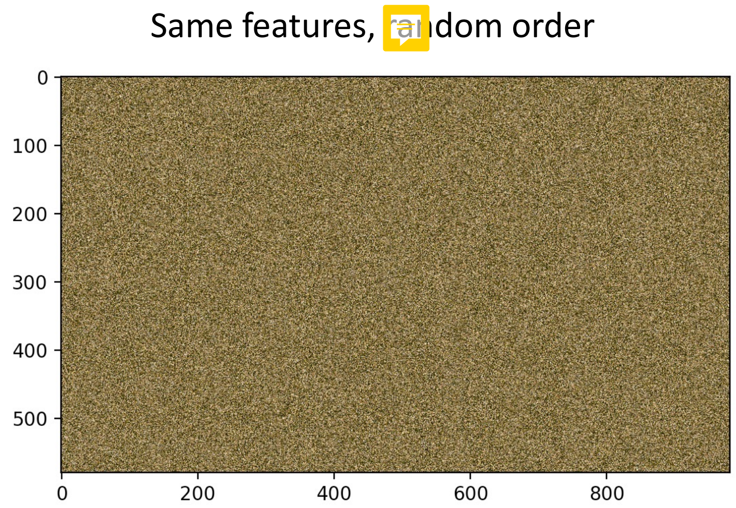


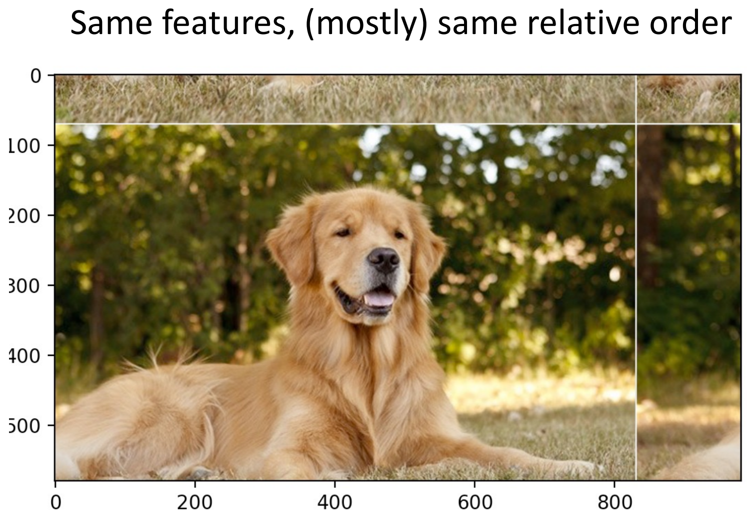


Key Ideas of CNNs

- **Locality**
  - Images encode information in the **order of features**
  - Nearby features should be considered together

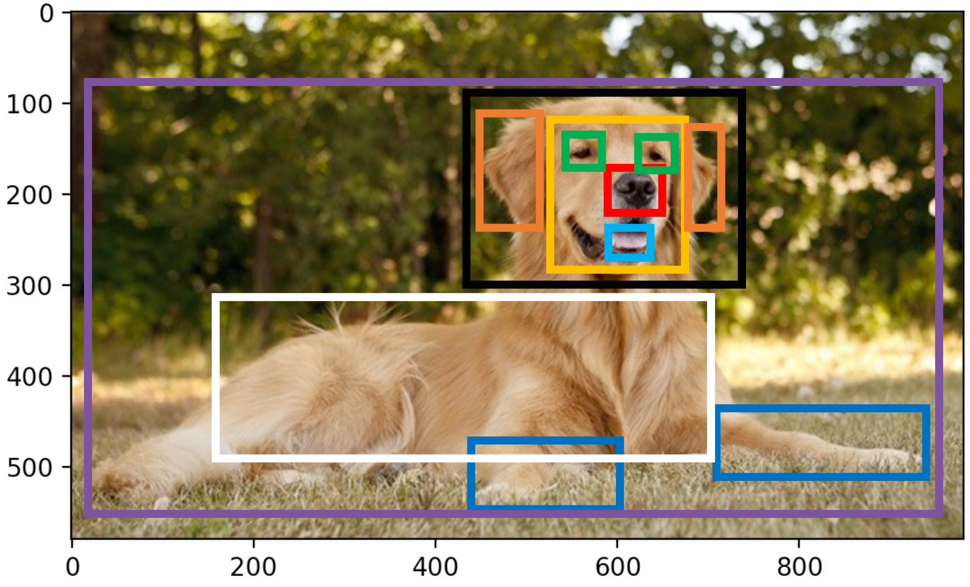






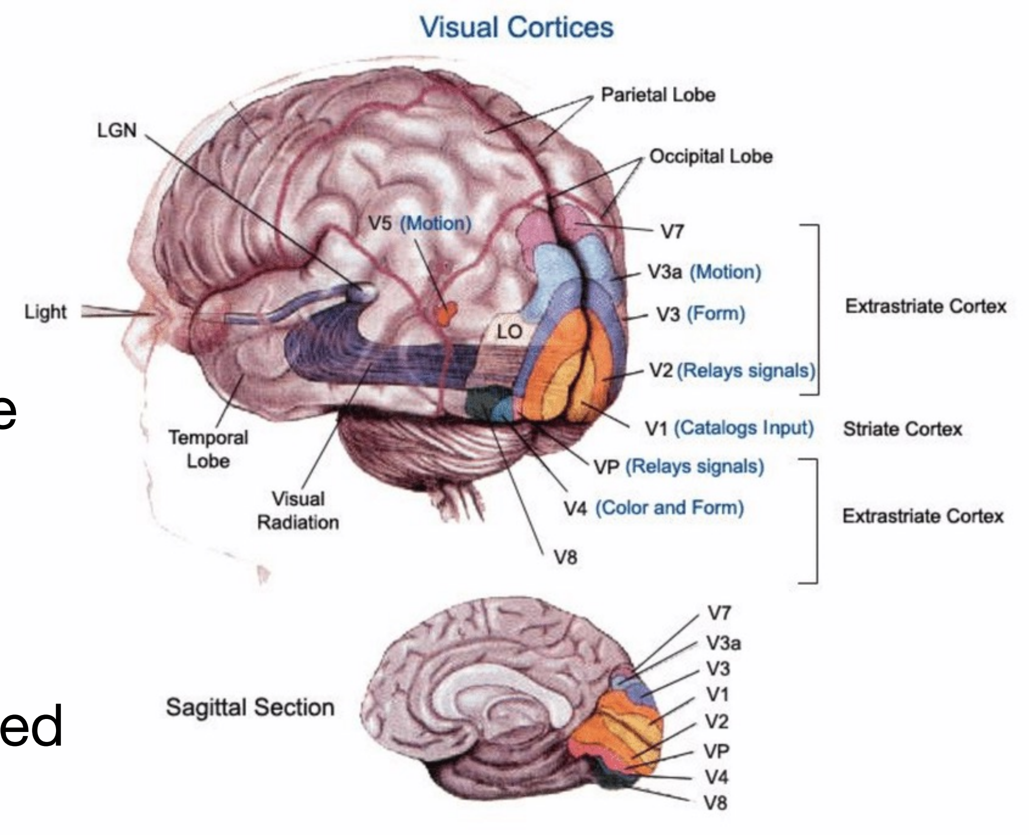
Key Ideas of CNNs

- **Hierarchy**
  - Many data types consist of patterns of patterns
  - Each layer of patterns involves a "wider" view of the data



Key Ideas of CNNs

- The human visual system is hierarchical
  - Neurons in the retina and V1 recognize simple localized light patterns
  - Neurons in higher visual cortices recognize combinations of patterns
  - Process repeats until entire scene is recognized



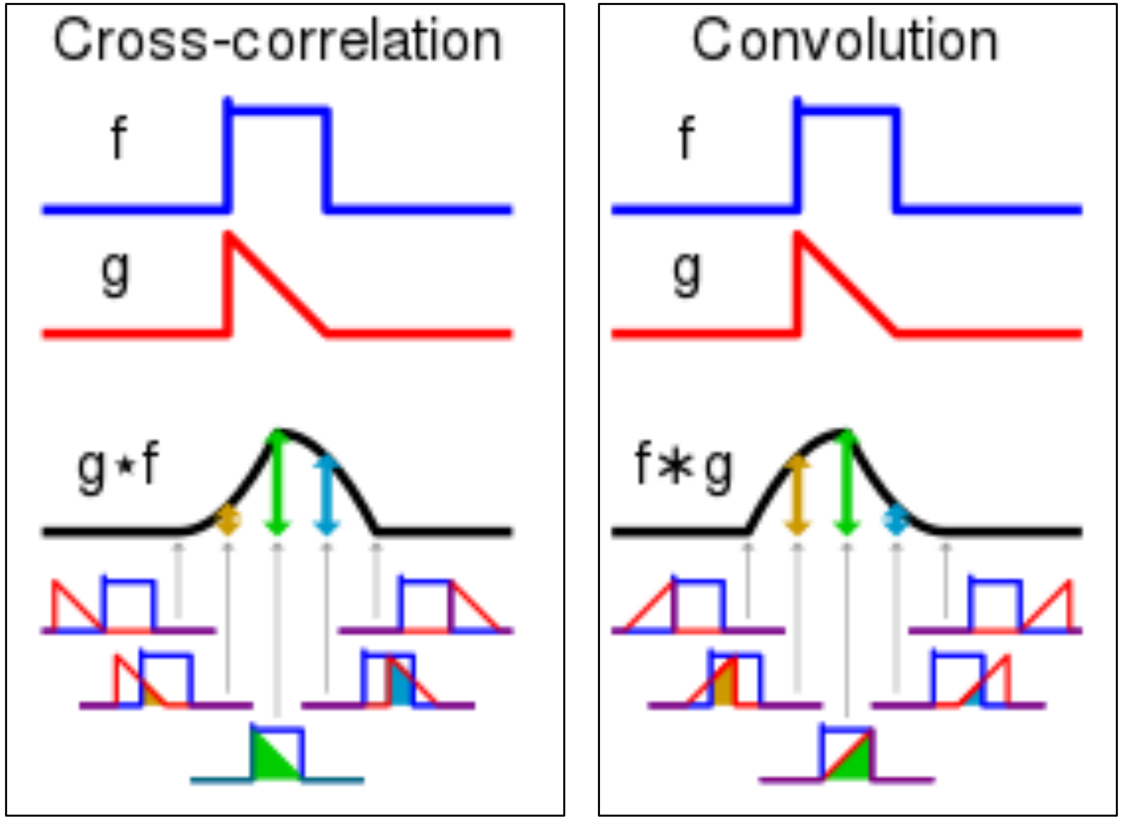
★ CNNs use **feature locality** & **pattern hierarchy** to create models with

- Fewer overall parameters
- Insensitivity to data translation

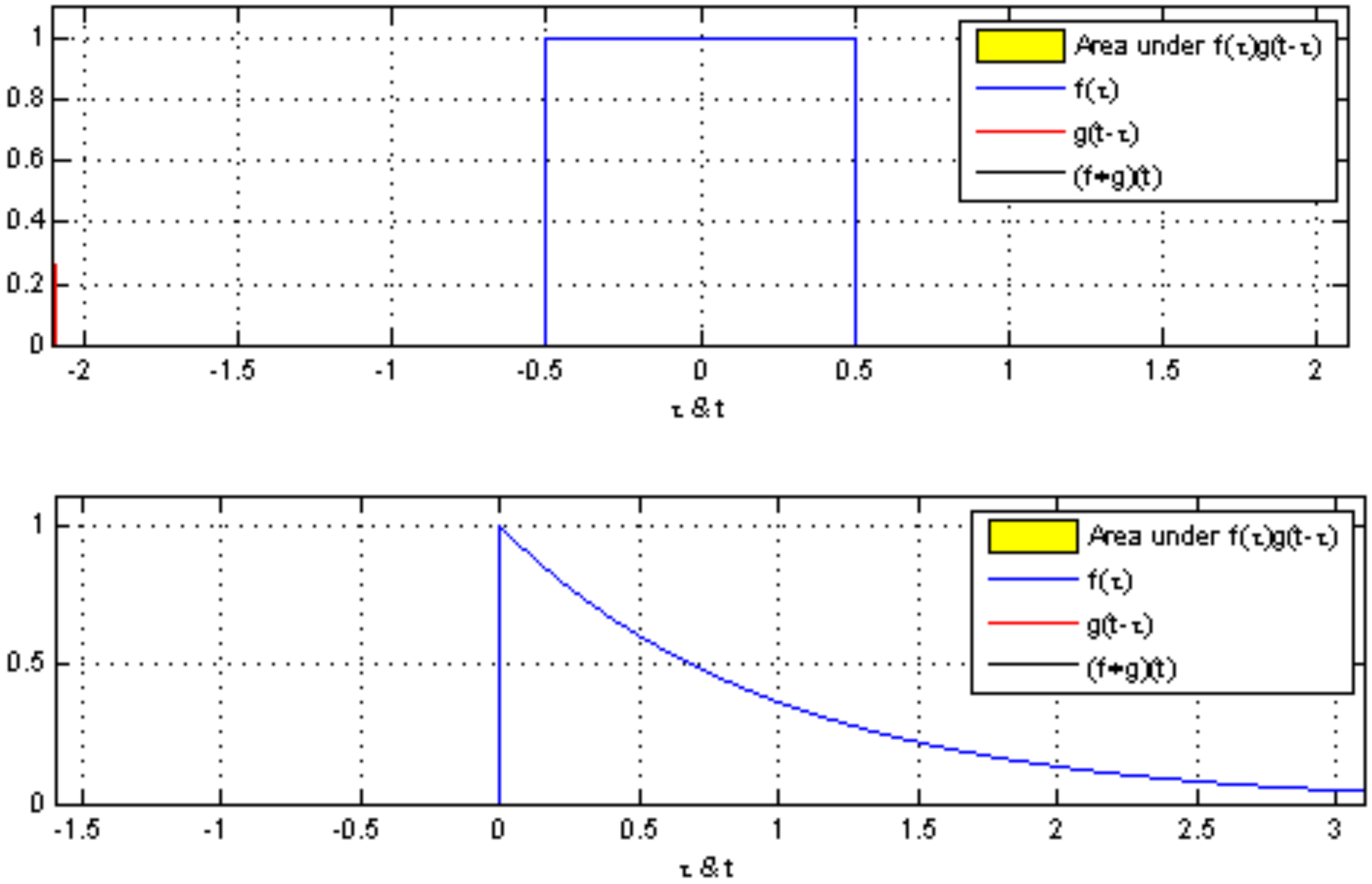
Convolutions

Convolutions & Cross-Correlations

- **Convolution** and **cross-correlation** are operations on functions
  - Measure **similarity** between two functions across their domains
  - Computed by **sliding** one function over the other and integrating **overlap**
  - *Convolution is just cross-correlation with one function reflected over y-axis*
    - "Convolutional" neural networks actually use cross-correlation for simplicity



1D Cross-Correlation Examples



2D Convolutions

3x3 **Filter**

$w_{0,0}$	$w_{0,1}$	$w_{0,2}$
$w_{1,0}$	$w_{1,1}$	$w_{1,2}$
$w_{2,0}$	$w_{2,1}$	$w_{2,2}$

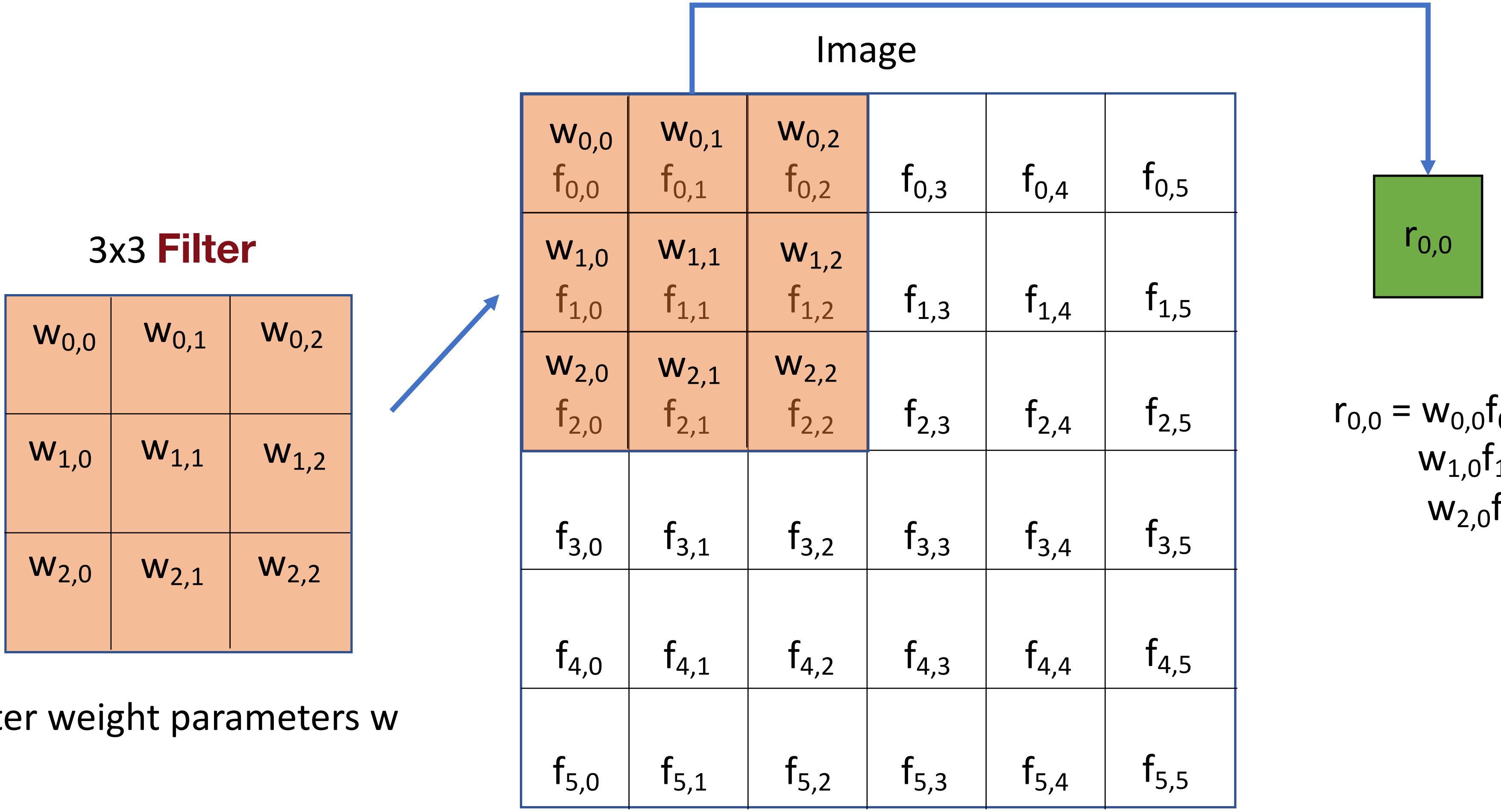
filter weight parameters w

Image

$f_{0,0}$	$f_{0,1}$	$f_{0,2}$	$f_{0,3}$	$f_{0,4}$	$f_{0,5}$
$f_{1,0}$	$f_{1,1}$	$f_{1,2}$	$f_{1,3}$	$f_{1,4}$	$f_{1,5}$
$f_{2,0}$	$f_{2,1}$	$f_{2,2}$	$f_{2,3}$	$f_{2,4}$	$f_{2,5}$
$f_{3,0}$	$f_{3,1}$	$f_{3,2}$	$f_{3,3}$	$f_{3,4}$	$f_{3,5}$
$f_{4,0}$	$f_{4,1}$	$f_{4,2}$	$f_{4,3}$	$f_{4,4}$	$f_{4,5}$
$f_{5,0}$	$f_{5,1}$	$f_{5,2}$	$f_{5,3}$	$f_{5,4}$	$f_{5,5}$

Result

2D Convolutions

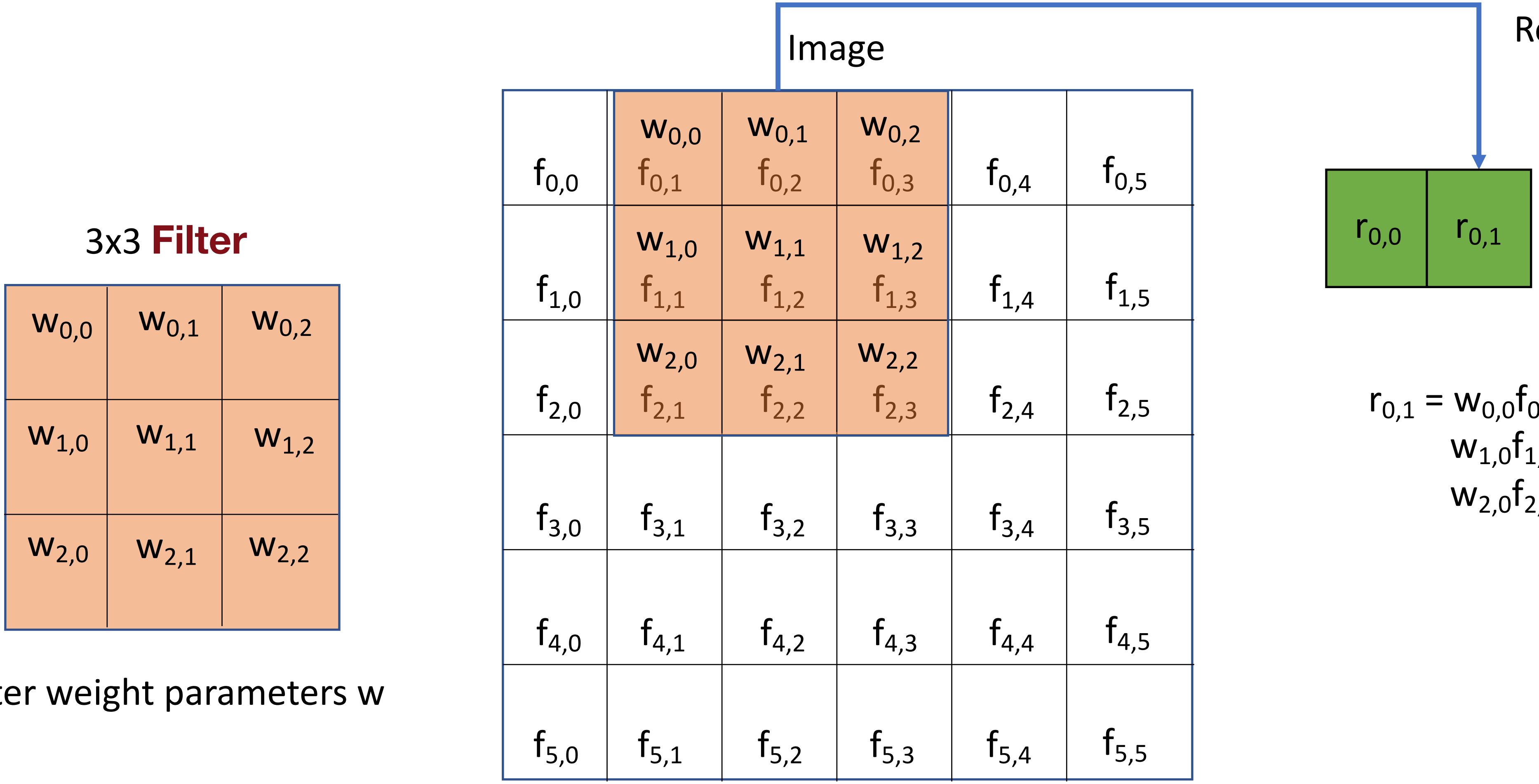


filter weight parameters w

$$r_{0,0} = w_{0,0}f_{0,0} + w_{0,1}f_{0,1} + w_{0,2}f_{0,2} + w_{1,0}f_{1,0} + w_{1,1}f_{1,1} + w_{1,2}f_{1,2} + w_{2,0}f_{2,0} + w_{2,1}f_{2,1} + w_{2,2}\, f_{2,2}$$

2D Convolutions

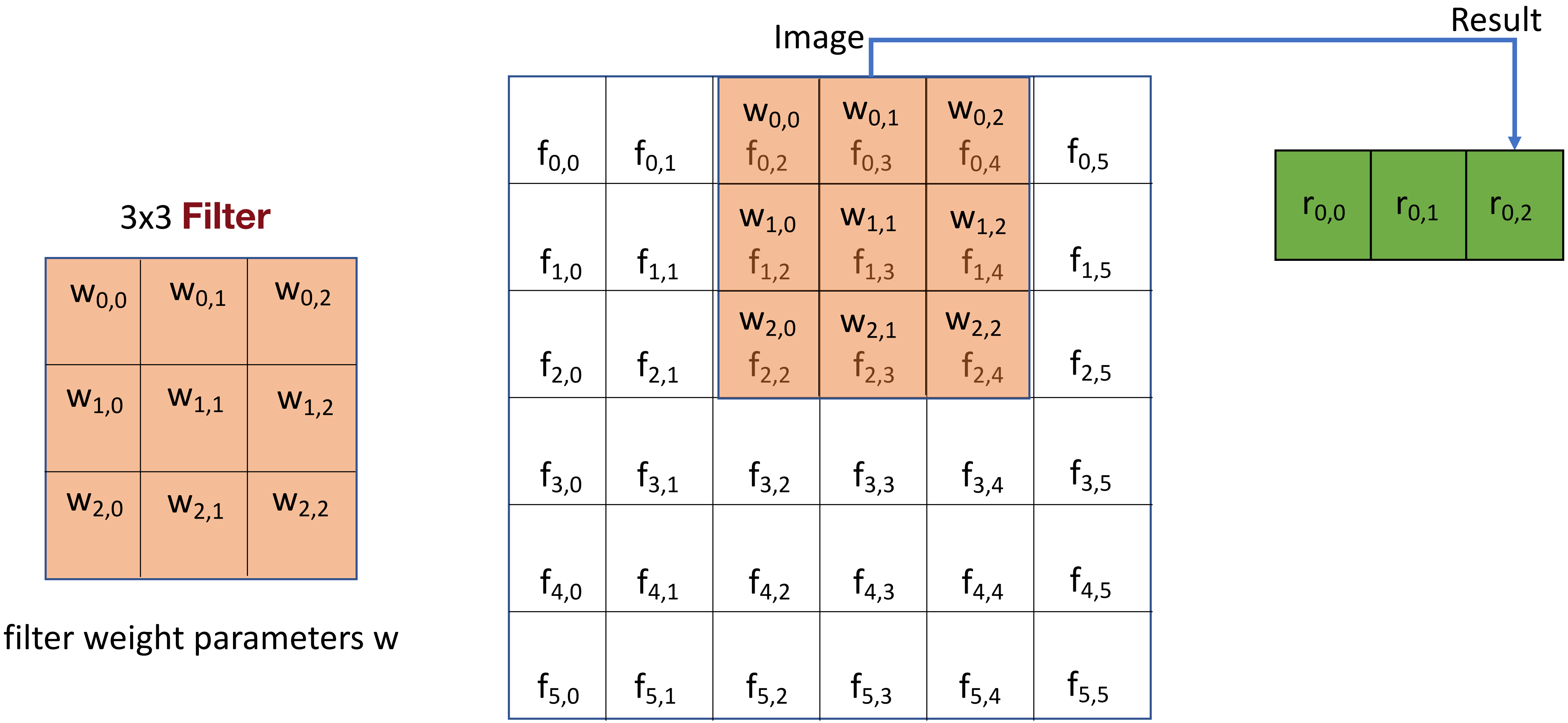
3x3 **Filter**



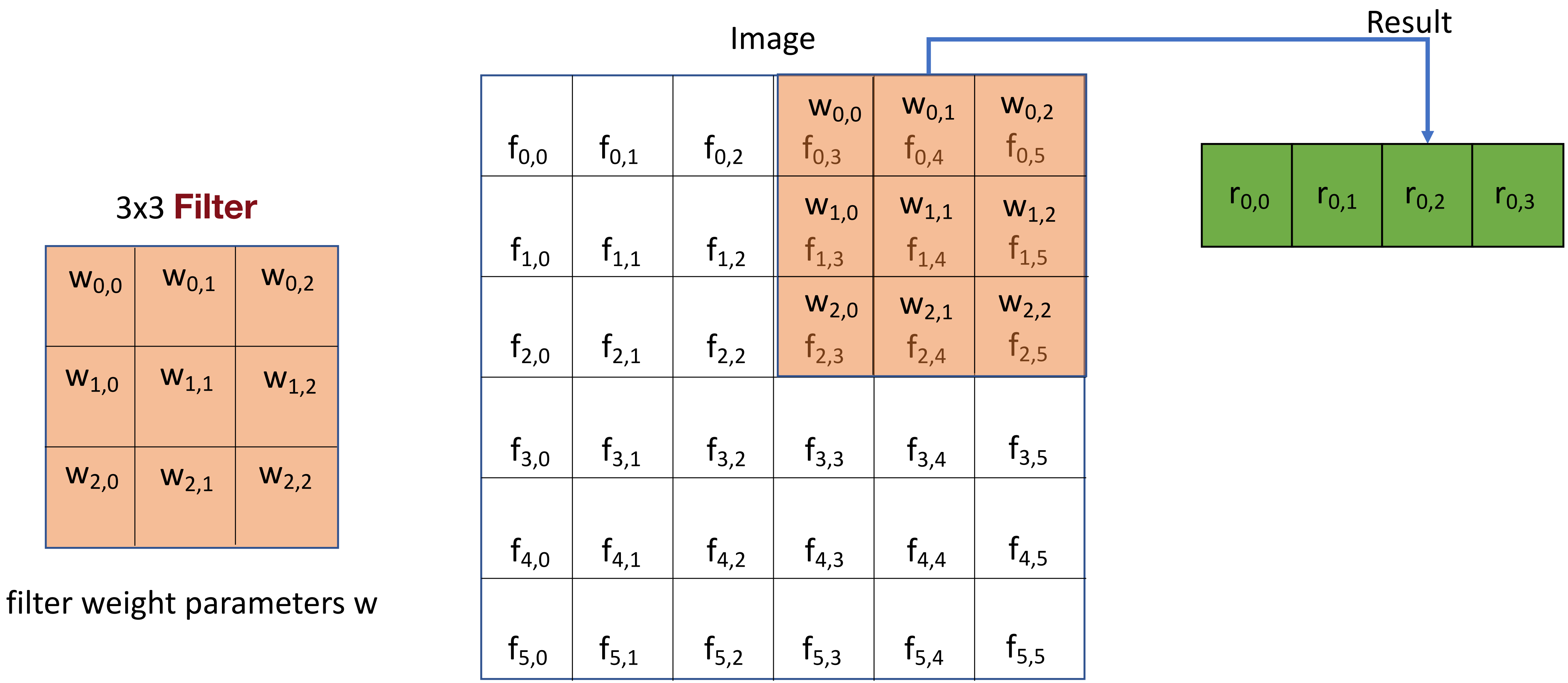
filter weight parameters w

$$\begin{aligned} r_{0,1} = {} & w_{0,0}f_{0,1} + w_{0,1}f_{0,2} + w_{0,2}f_{0,3} + \\ & w_{1,0}f_{1,1} + w_{1,1}f_{1,2} + w_{1,2}f_{1,3} + \\ & w_{2,0}f_{2,1} + w_{2,1}f_{2,2} + w_{2,2}\,f_{2,3} \end{aligned}$$

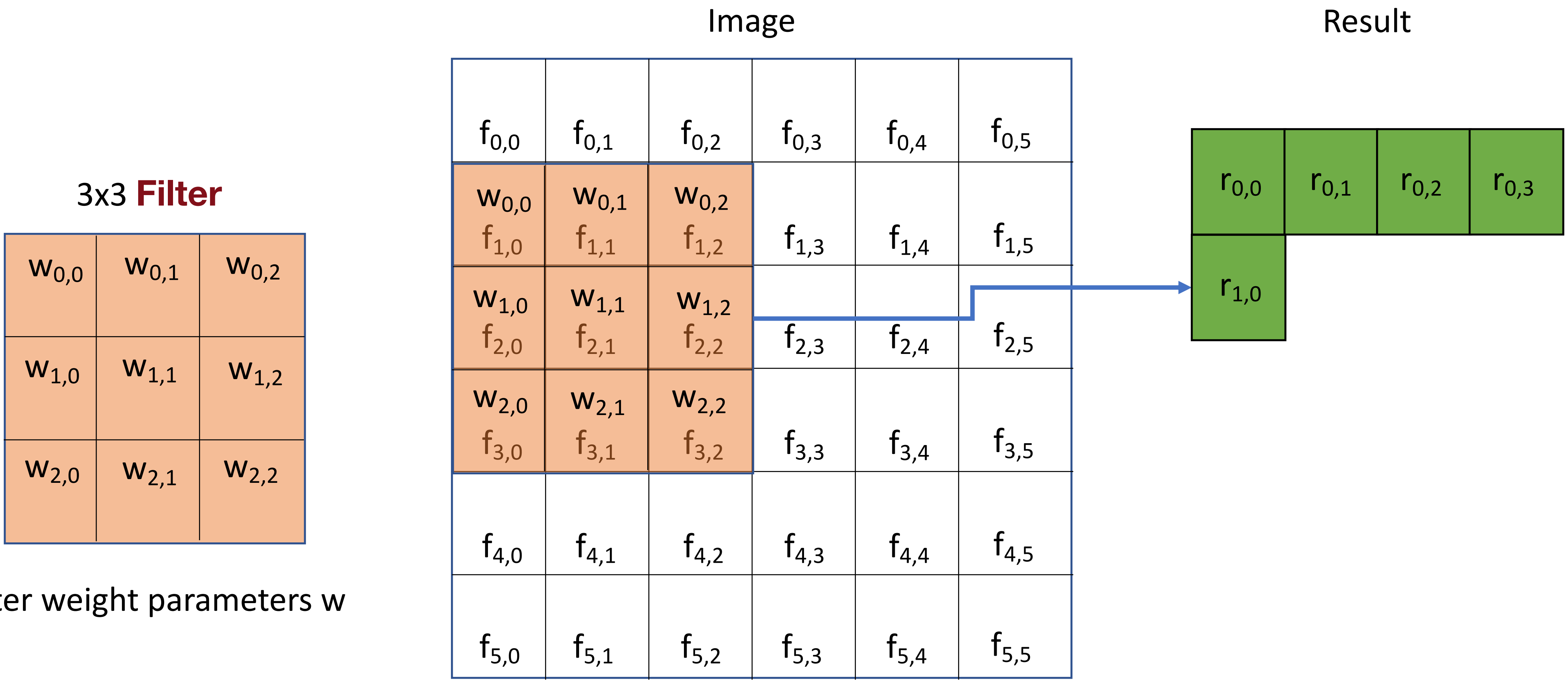
2D Convolutions



2D Convolutions



2D Convolutions



filter weight parameters w

2D Convolutions

Convolution Hyperparameters

Filter Size

- Convolutional filters identify areas of the image with similar patterns
- Why might you want larger or smaller filters?

Example convolutional filters

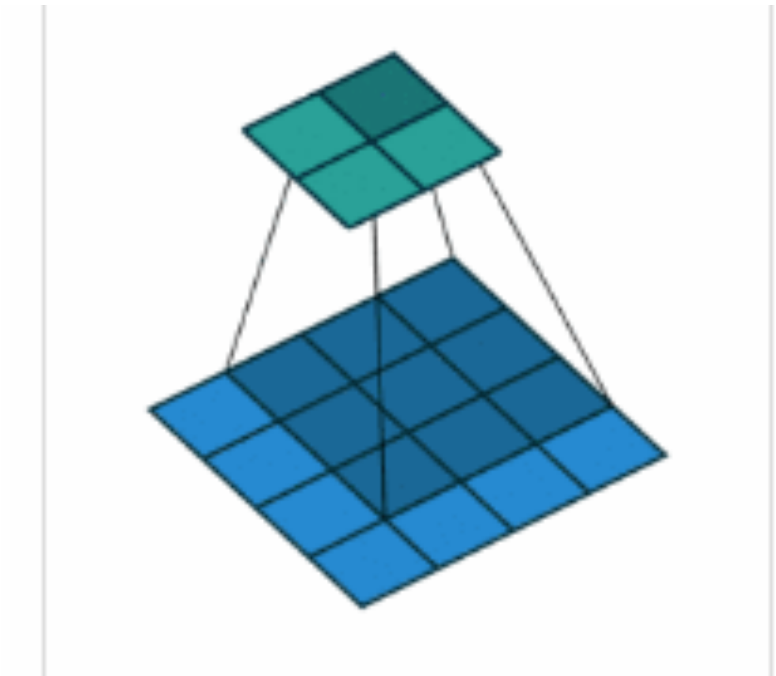
Filter Size

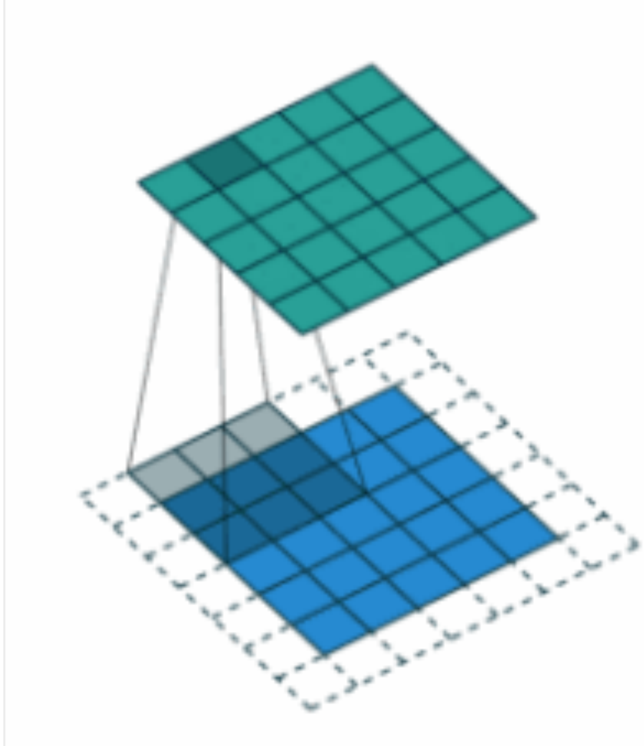
- Convolutional filters identify areas of the image with similar patterns
- Larger filters can identify larger patterns, but have more parameters and less locality benefit

Example convolutional filters

Padding

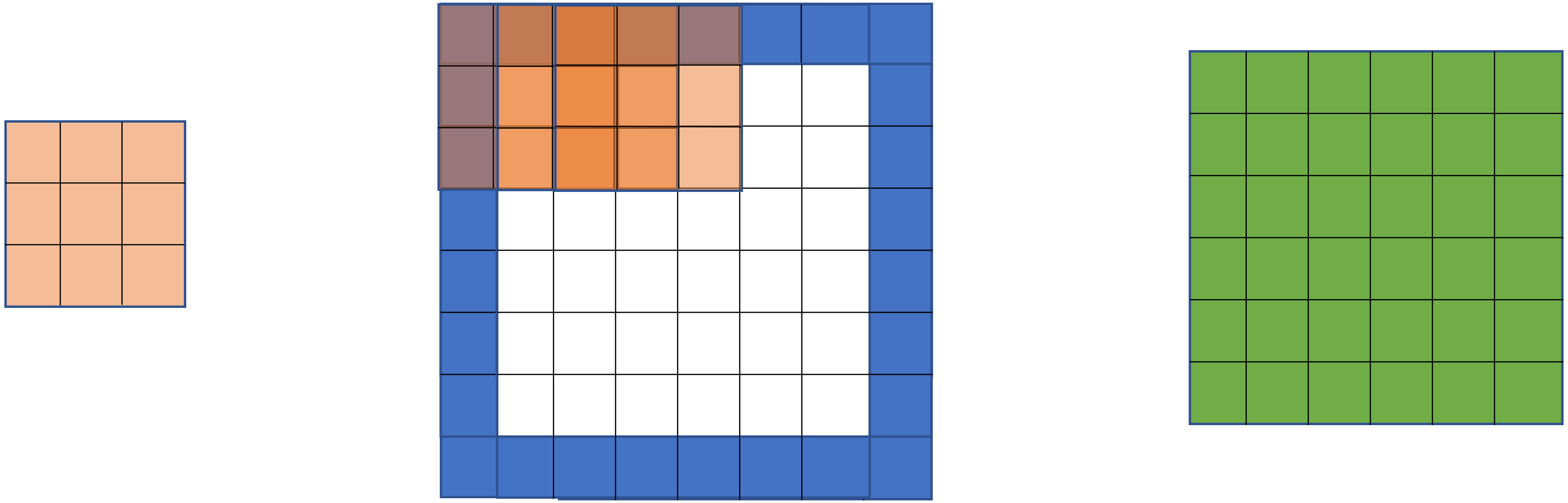
- In our convolution example, the result was smaller than the original image
- If we add **padding** to the edges of the input, the result will be the same size





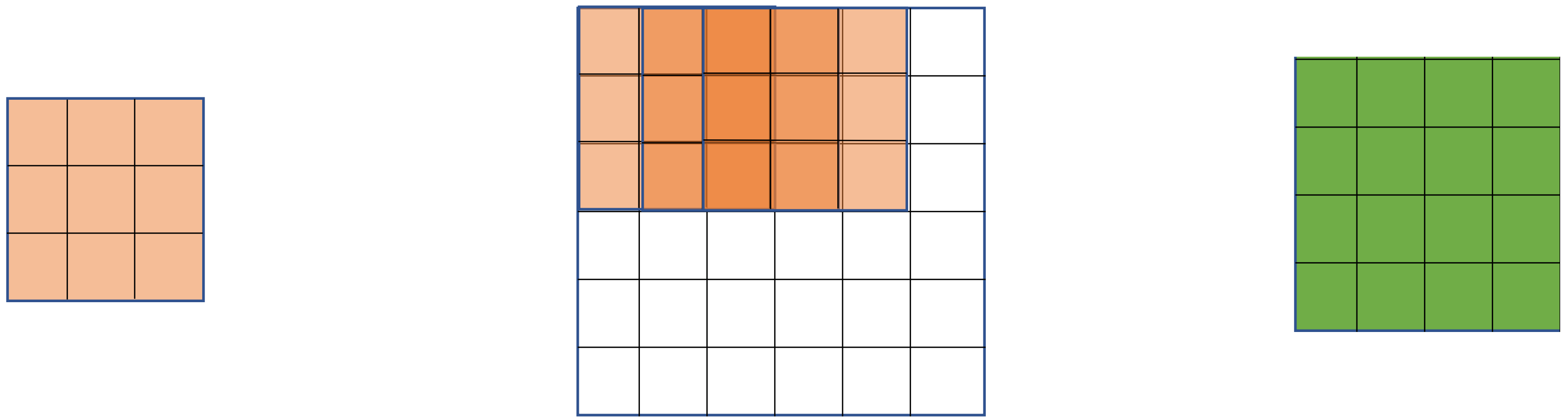
Padding

- **Padding** options
  - **“Same”** – Add padding (usually 0s) to edges of data so convolution results are the same size as the input examples
  - Width of padding depends on filter size



Padding

- **Padding** options
  - **“Valid”** – Do NOT add padding, so convolution results are is smaller than the input example
  - Size difference depends on filter size

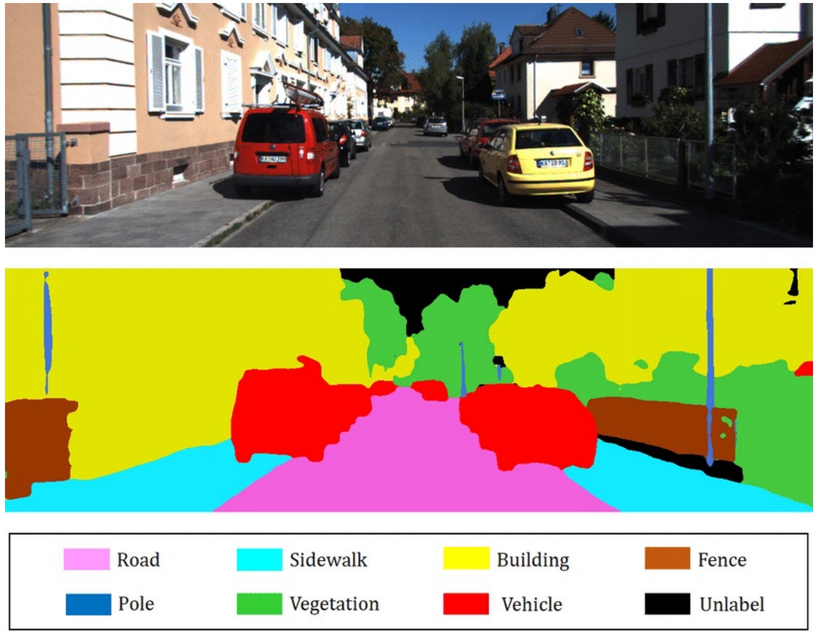


Padding

- Why might you want "same" vs. "valid" padding or vice versa?

Padding

- When to use different padding options
  - **“Same” (padding):** Use for **image segmentation** when each pixel in input needs a label in the output – or generally when the edges have important info



Don’t cut off the edges of the scene!

Padding

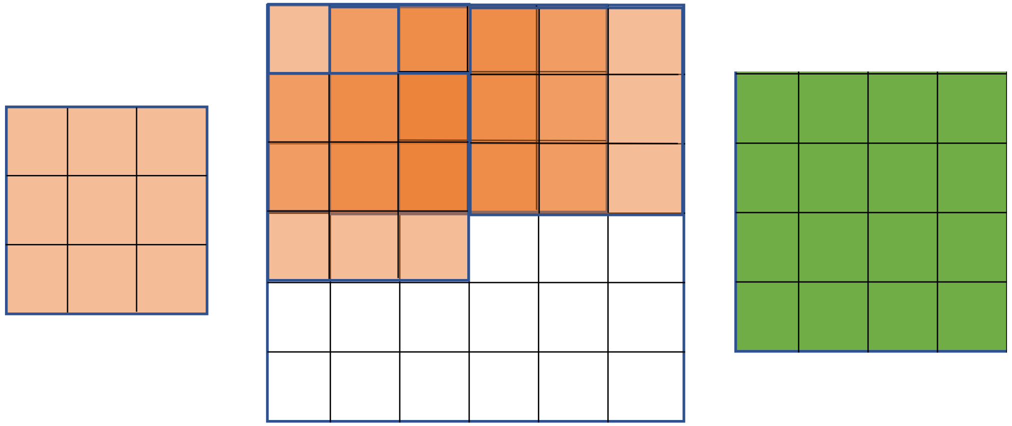
- When to use different padding options
  - **"Same" (padding):** Use for **image segmentation** when each pixel in input needs a label in the output – or generally when the edges have important info
  - **"Valid" (NO padding):** Use for **image classification** when you only need label(s) for the entire image – or generally when the edges are irrelevant

Label: Dog

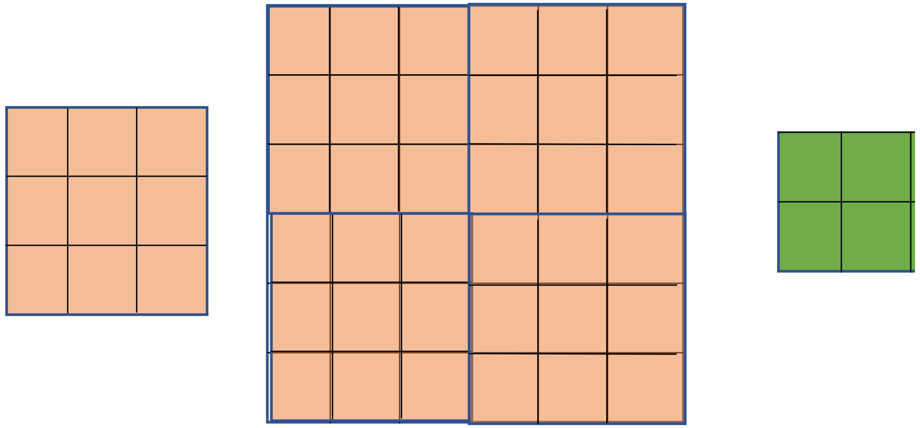
Label: Cat

Stride

- In our 2D convolution example, we slid the filter by one “space” each time
- We can also choose to slide the filter by multiple spaces
- Tradeoff between identifying overlapping patterns & computation cost



Stride = 1

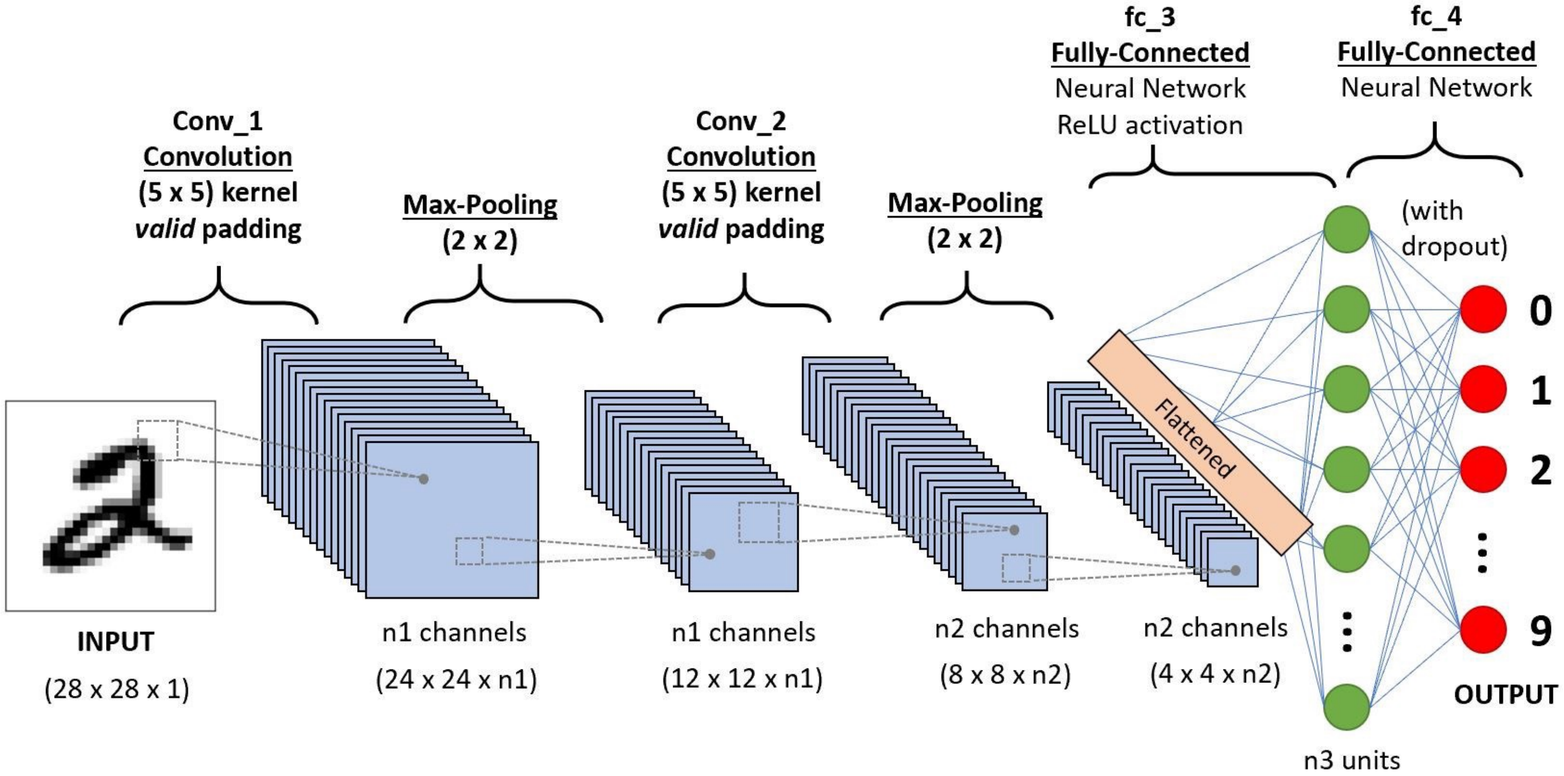


Stride = 3

Convolutional Neural Networks

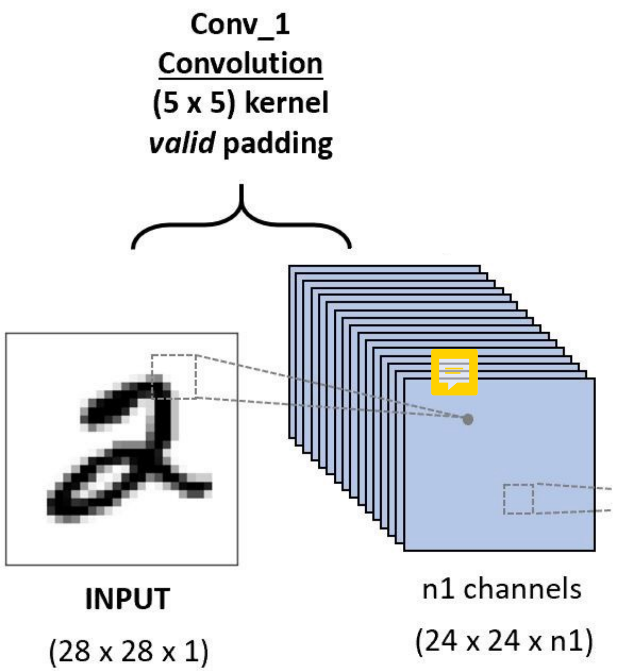
Convolutional Neural Networks (CNNs)

- Combine many convolutional filters in parallel and in sequence to identify localized & hierarchical patterns in data



Convolutional Layers

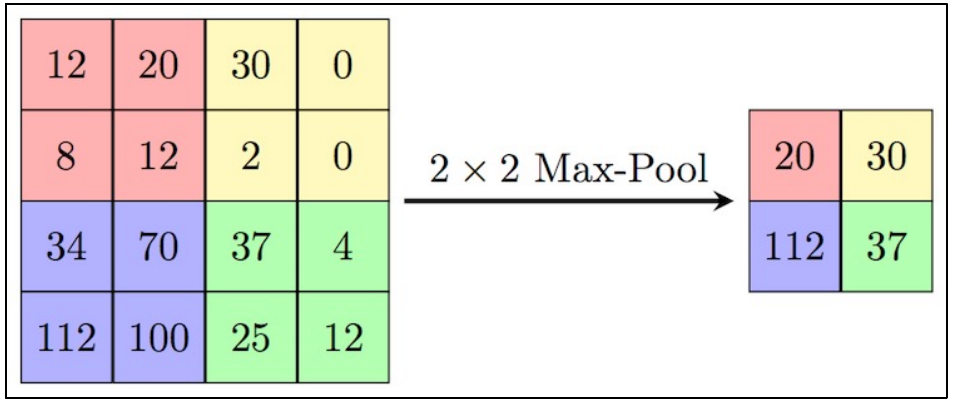
- Primary building block of CNNs
  - *N* independent convolutional filters
    - *N* is a hyperparameter
    - All filters have same size, padding, and stride
    - All filters initialized to different random starting weights
  - Activation function applied to each filter output

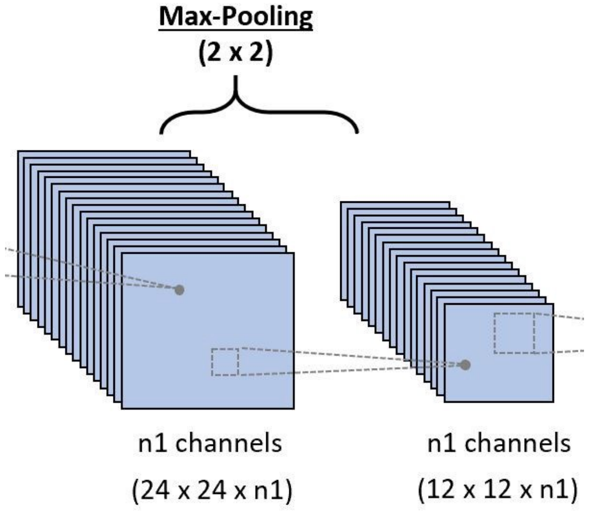


  - Produces N unique outputs called **"feature maps"** for each input

Pooling Layers

- Downsampling
  - Allows following convolution layers to find patterns over wider areas
  - Reduces # of computations
- Pros & cons
  - Information loss can reduce performance
  - Computation is less of a bottleneck than it used to be



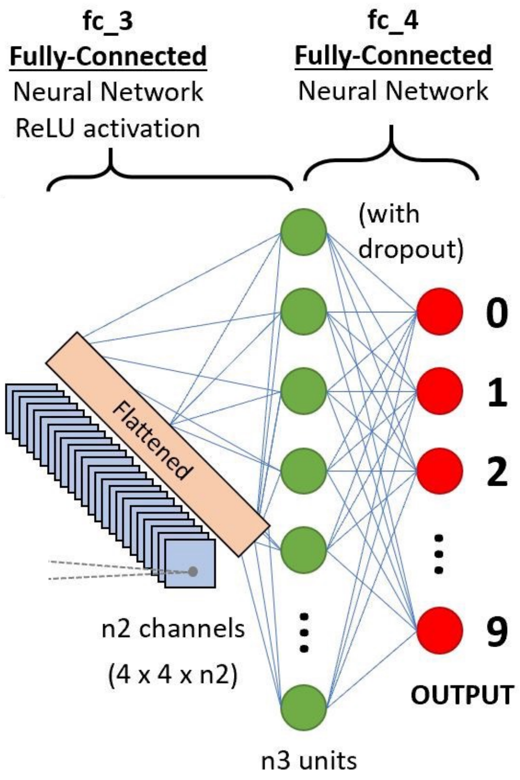


Fully Connected Layers

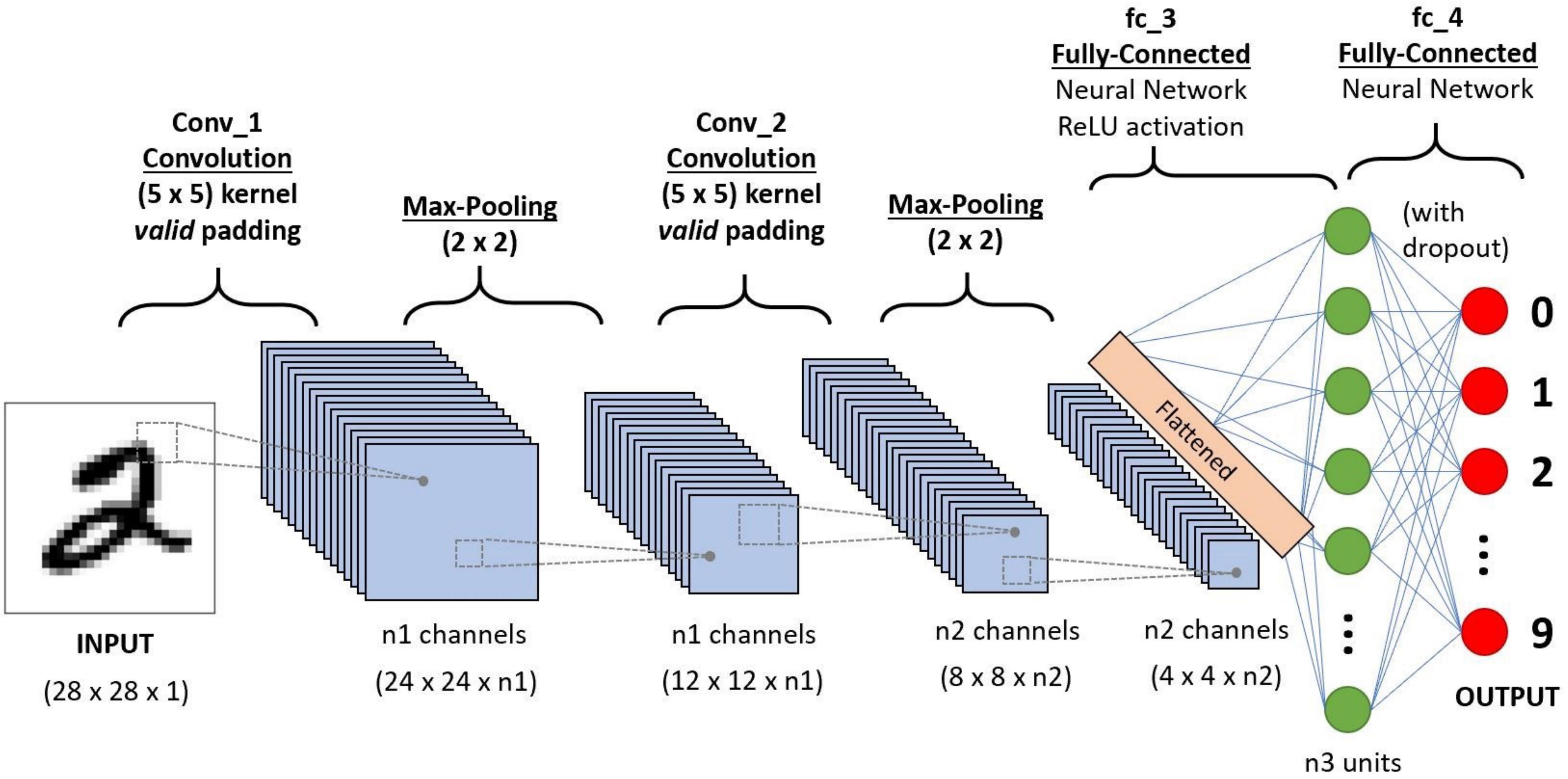
- CNNs for image classification often end with one (or more) fully-connected feedforward layers

- Effectively a FNN that treats the outputs of the last convolutional layers as input features

- Produces class probabilities or regression values



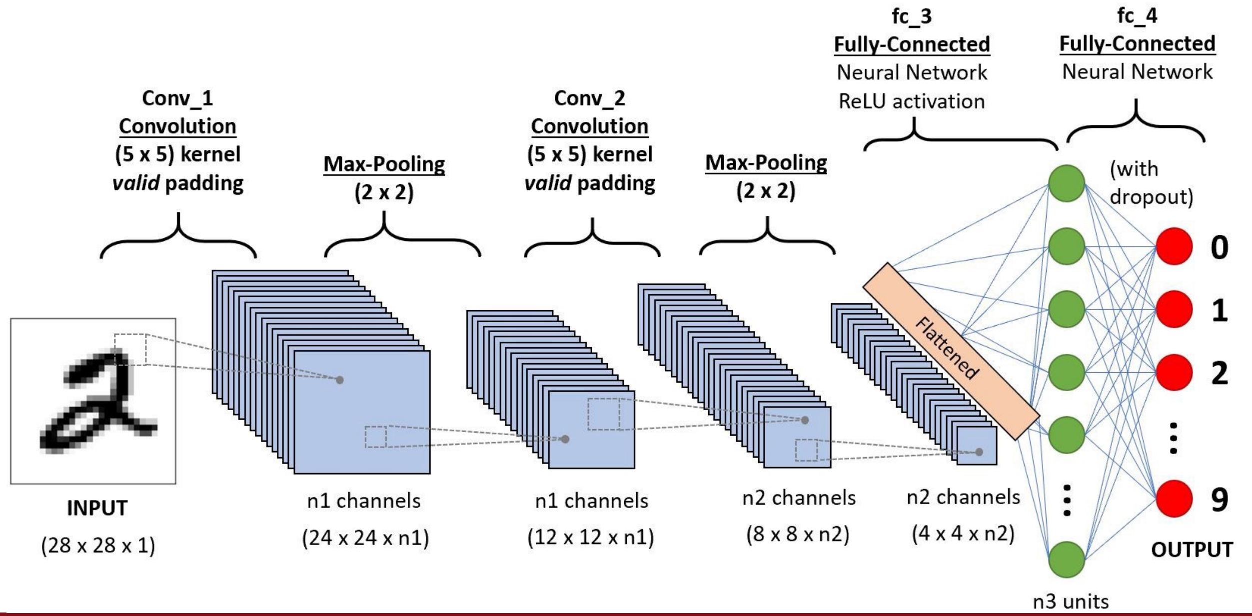
CNN Architecture



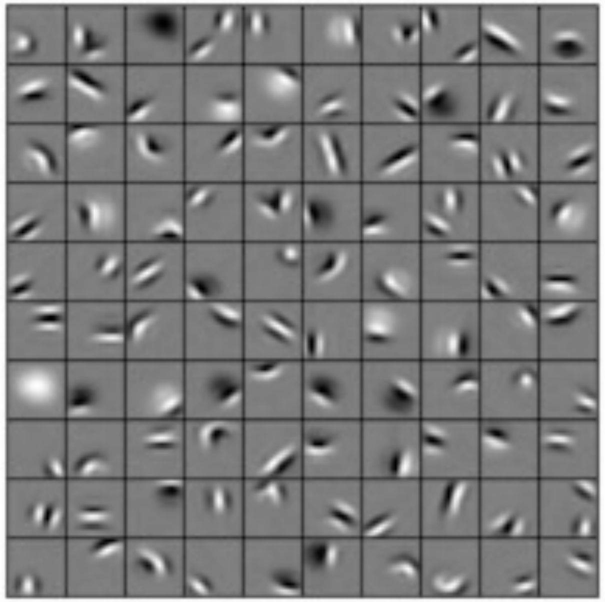
Training CNNs

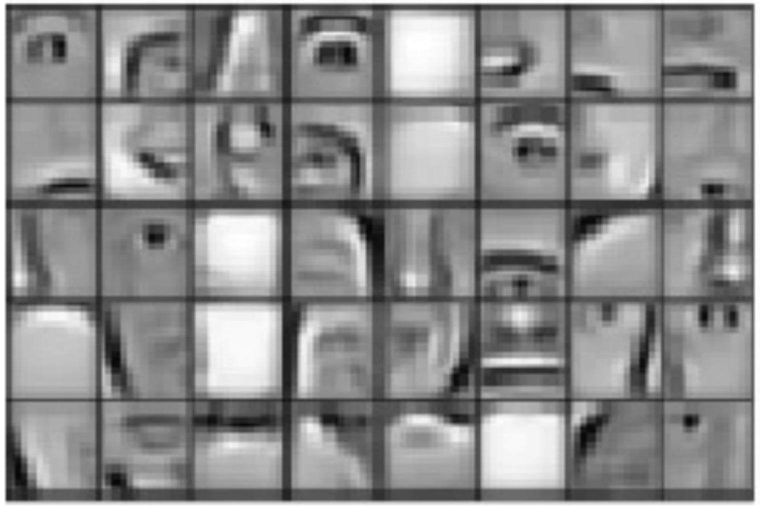
Training CNNs

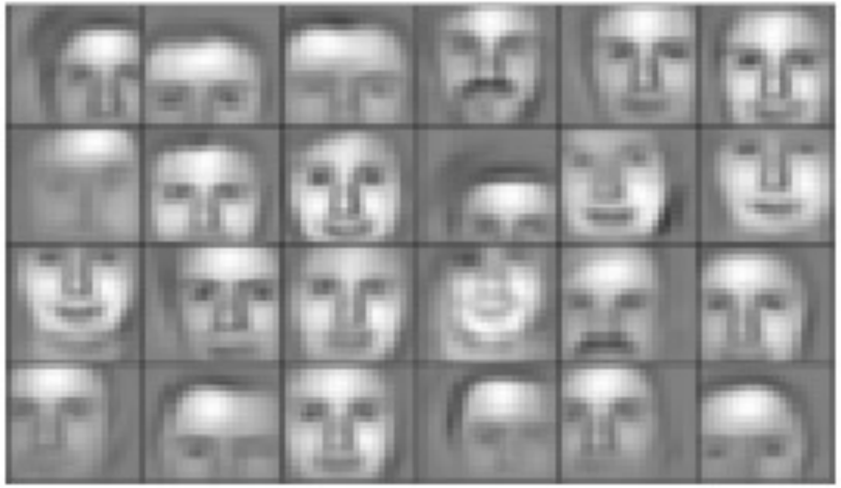
- ***Goal:*** Find **convolutional filter weights** (and feedforward connection weights if applicable) that minimize training error
- ***Approach:*** Gradient descent with backpropagation



Example Trained CNN Filters







Lee, et al.

Conv layers early in network ⟶ Conv layers later in network

Questions?