# Image Featurization CovNet style

# Objectives

- Be able to explain what a convolution is, and how it works
- Understand the basic structure of a convolutional neural network
- Comprehend the three basic ideas behind convolutional networks :

    (1) Local receptive field
    (2) Shared weights
    (3) Pooling

- Be aware of general strategies for building convolutional neural networks

# Convolutions

In image processing, a kernel, convolution matrix, or mask is a small matrix useful for blurring, sharpening, embossing, edge-detection, and more. This is accomplished by means of **convolution** between a kernel and an image.

-Wikipedia

# Convolutions

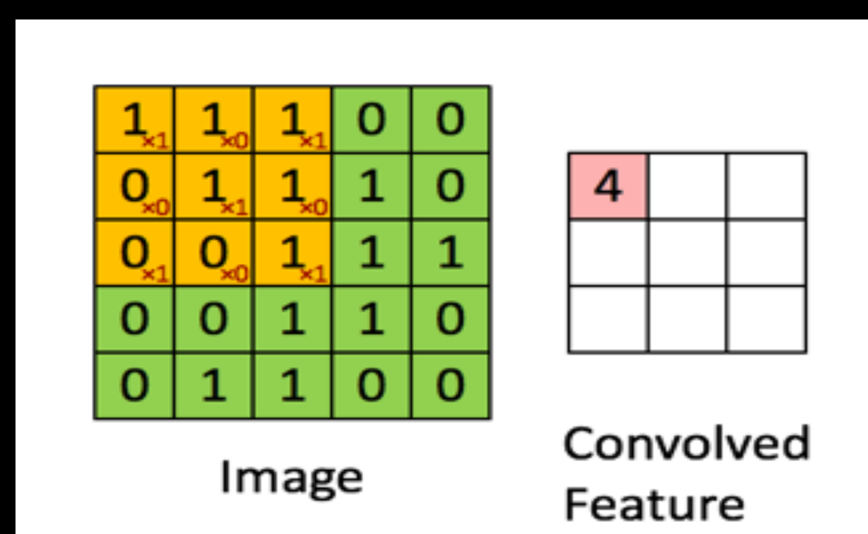
A Kernel

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

Applied Over

A Simple Image

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



# Example Kernels - Edge Detectors

Vertical Edge Detector

| | | |
|---|---|---|
| -1 | 0 | +1 |
| -2 | 0 | +2 |
| -1 | 0 | +1 |

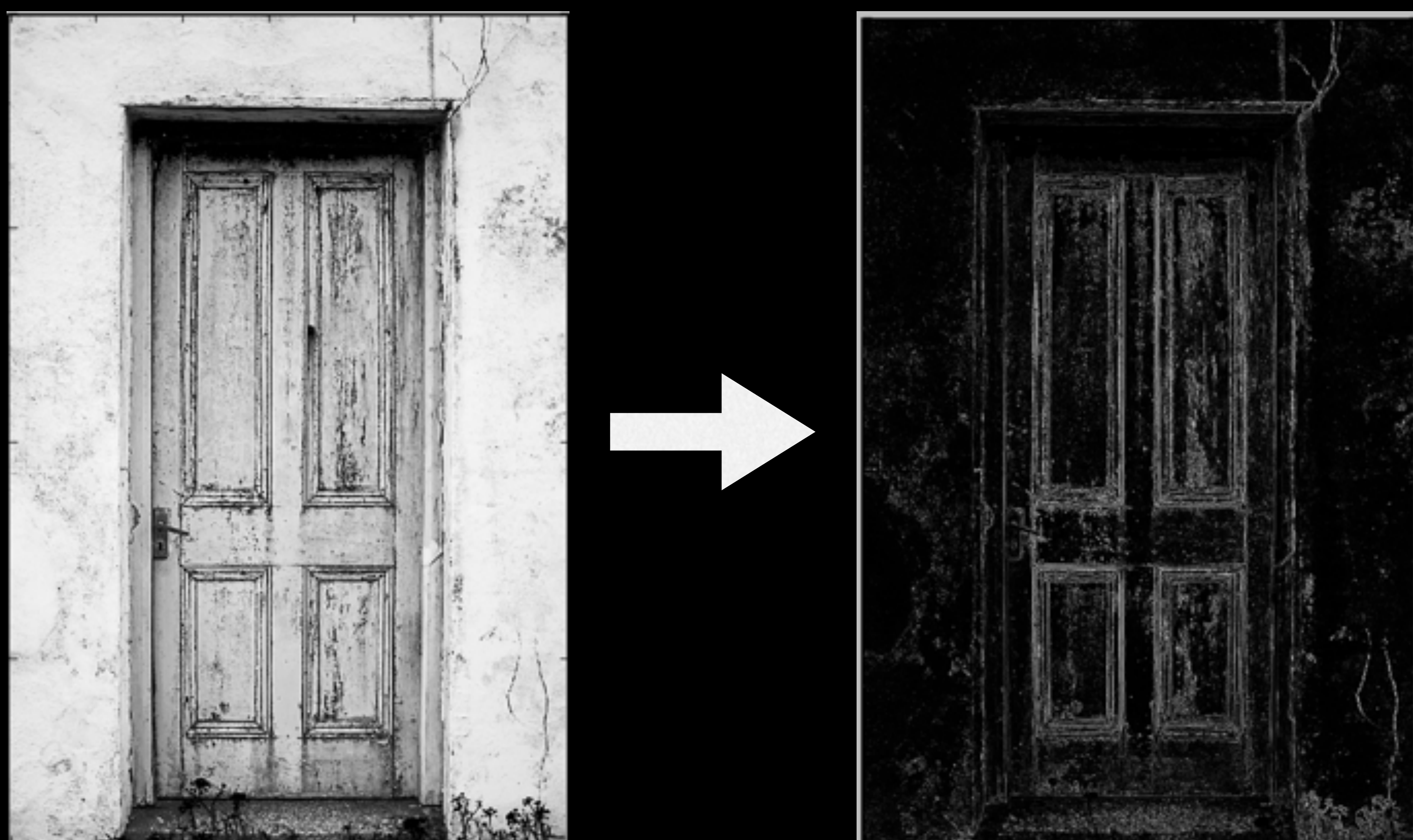
Horizontal Edge Detector

| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| +1 | +2 | +1 |

**Sobel filter**

# How do we tell that this is a door?

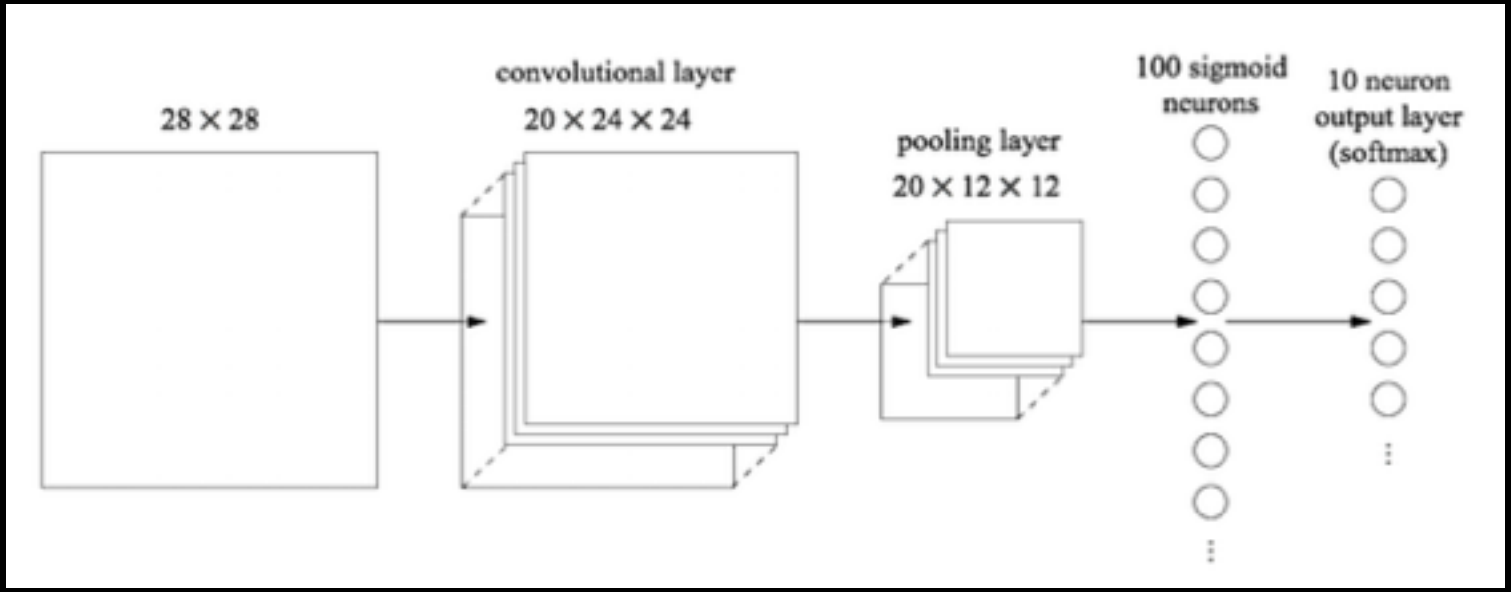
# What if we applied our edge detectors to the image?



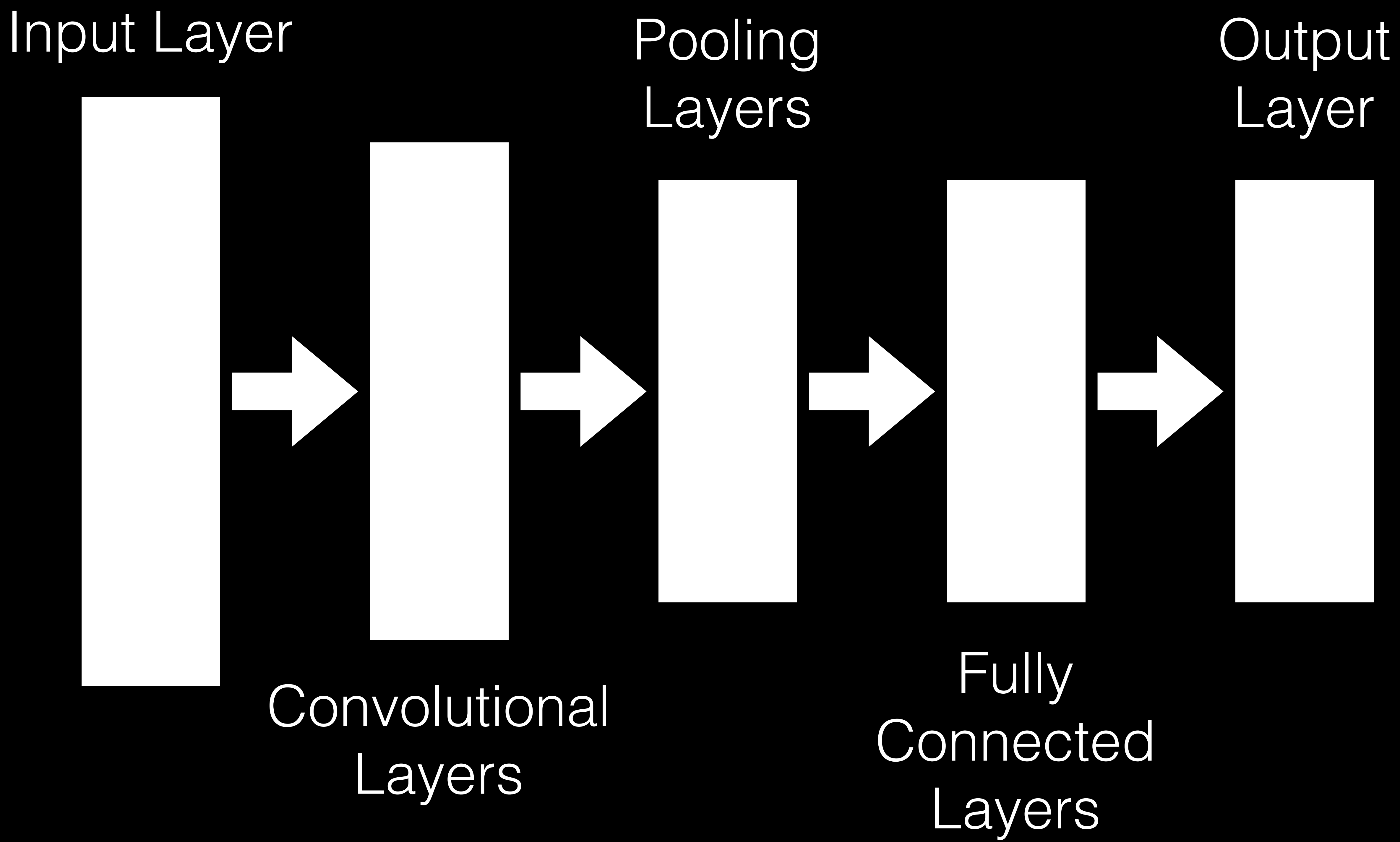
# Convolutional Neural Networks (CNNs)

What if we could get a computer to build it's own kernels, apply those to images, and then interpret those results to perform object recognition?

Enter convolutional neural networks….



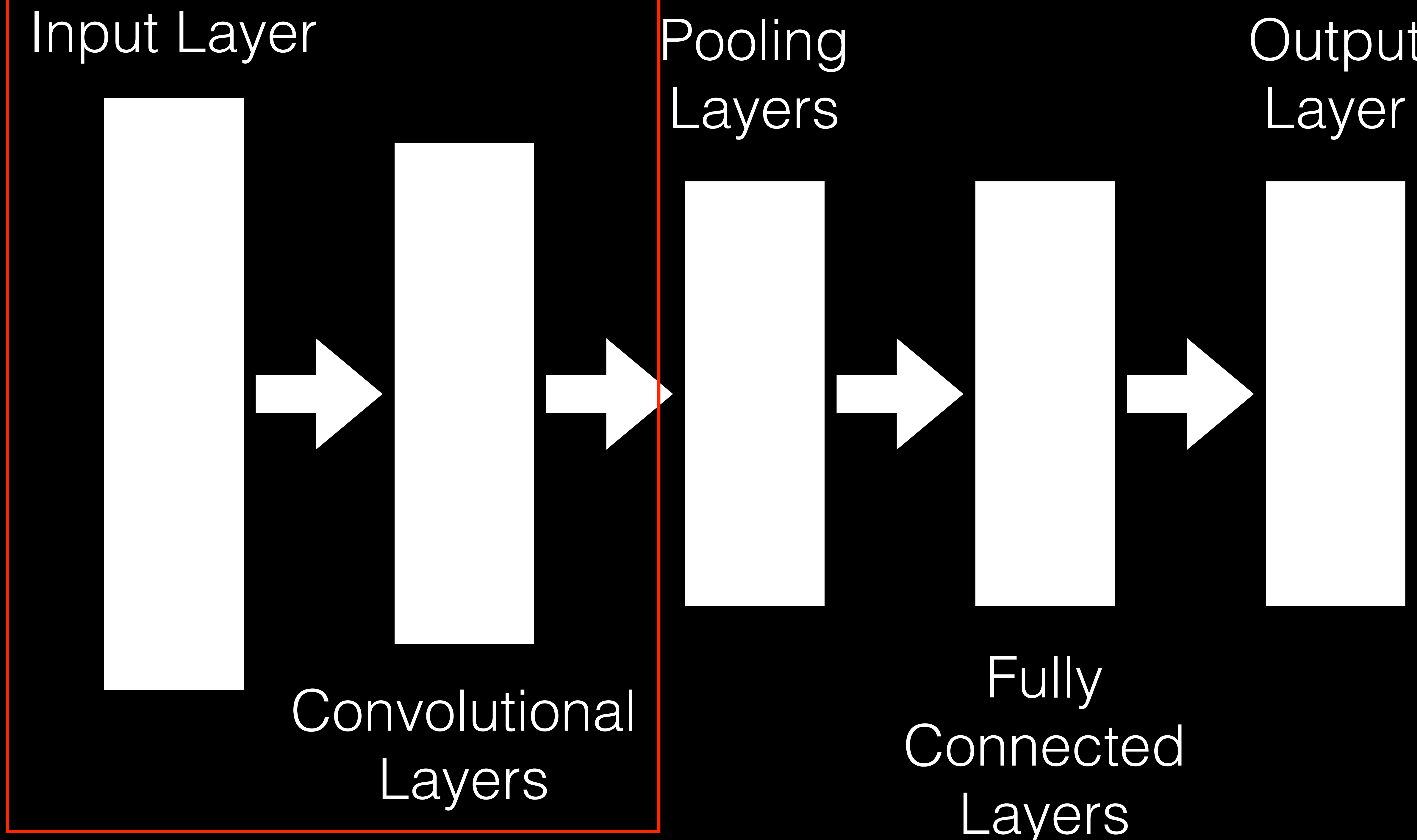
# General Structure



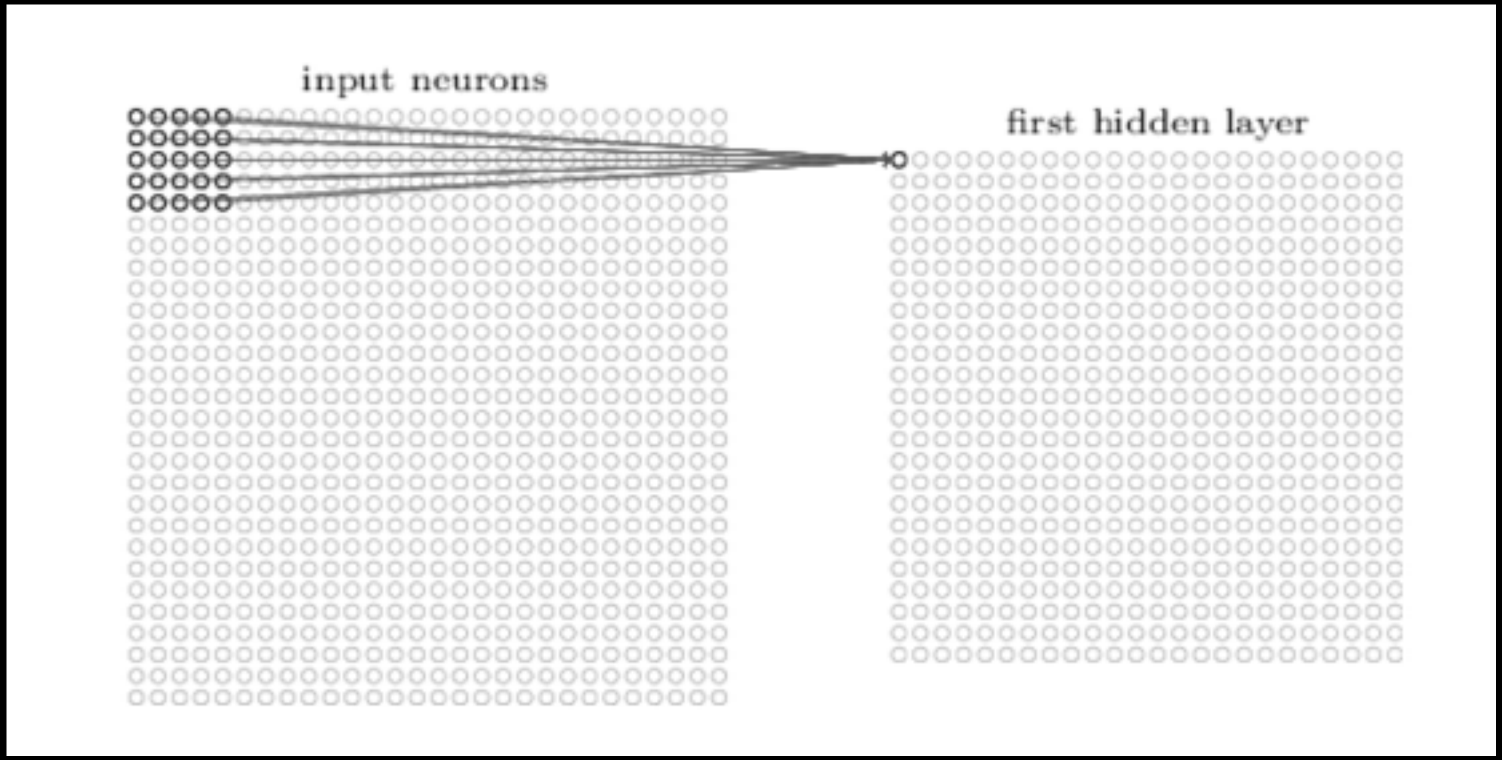
# Three Main Ideas…

- Local Receptive Fields
- Shared Weights
- Pooling

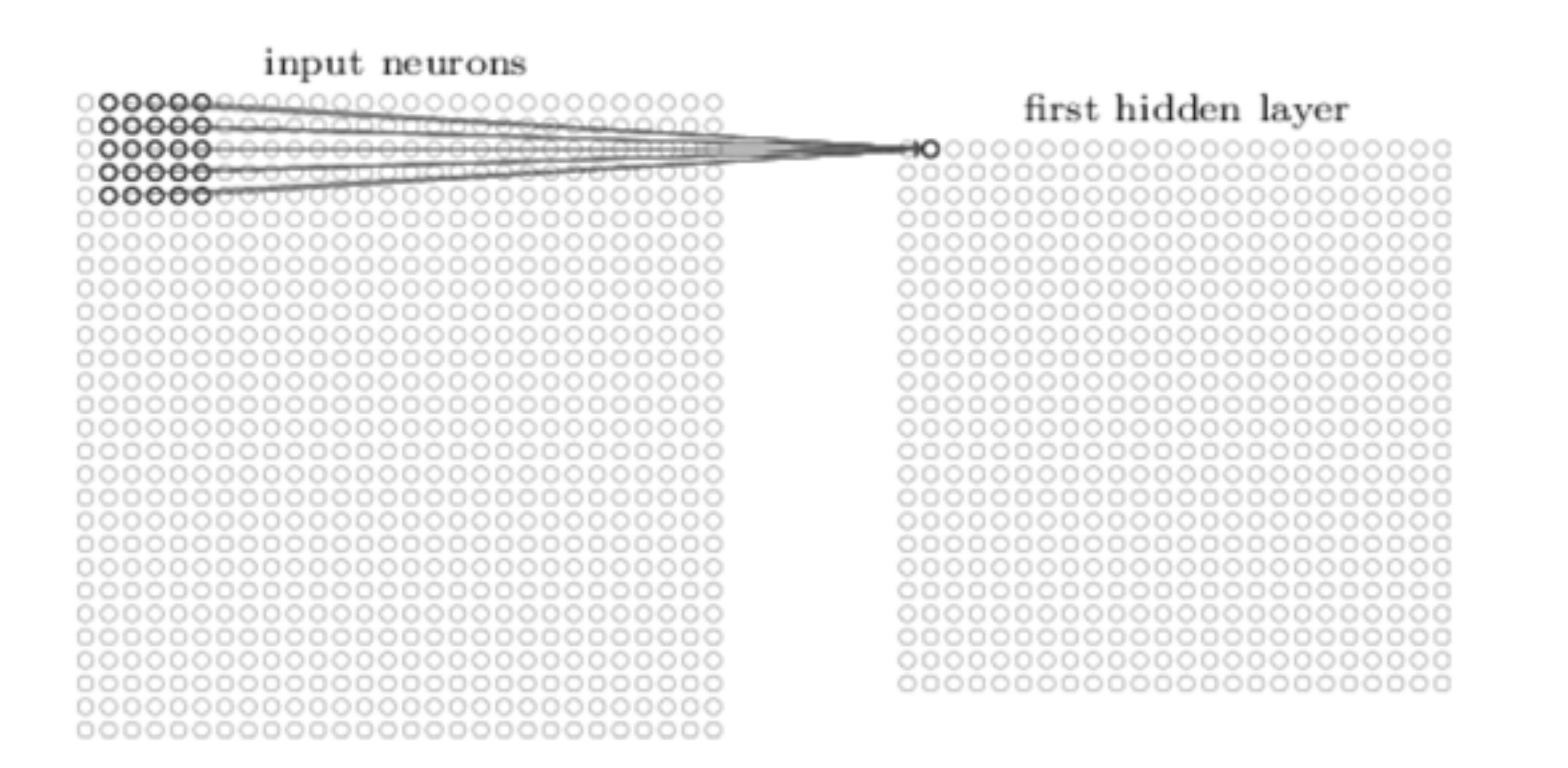
# General Structure



# Input Image



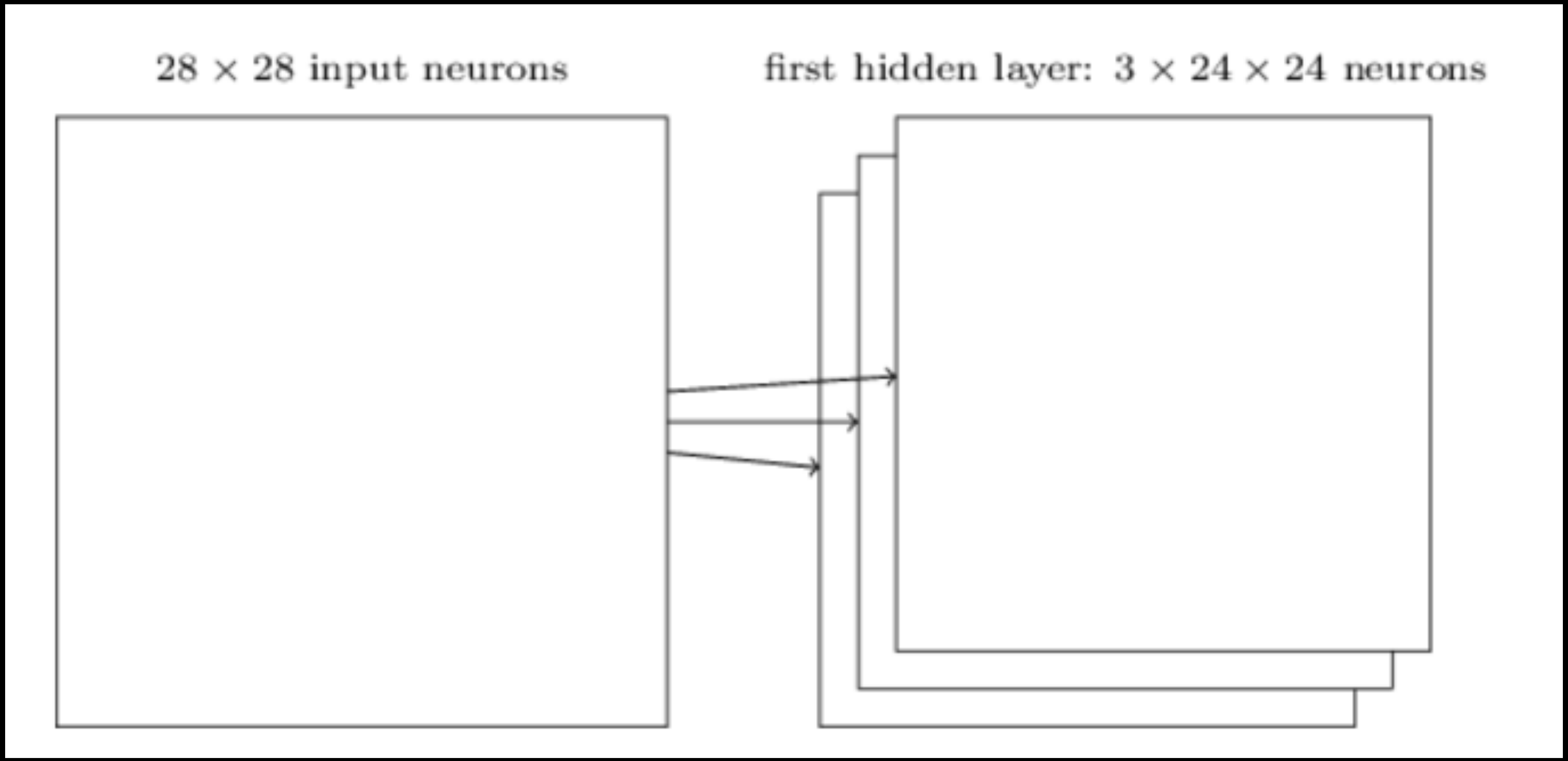
# Local Receptive Fields



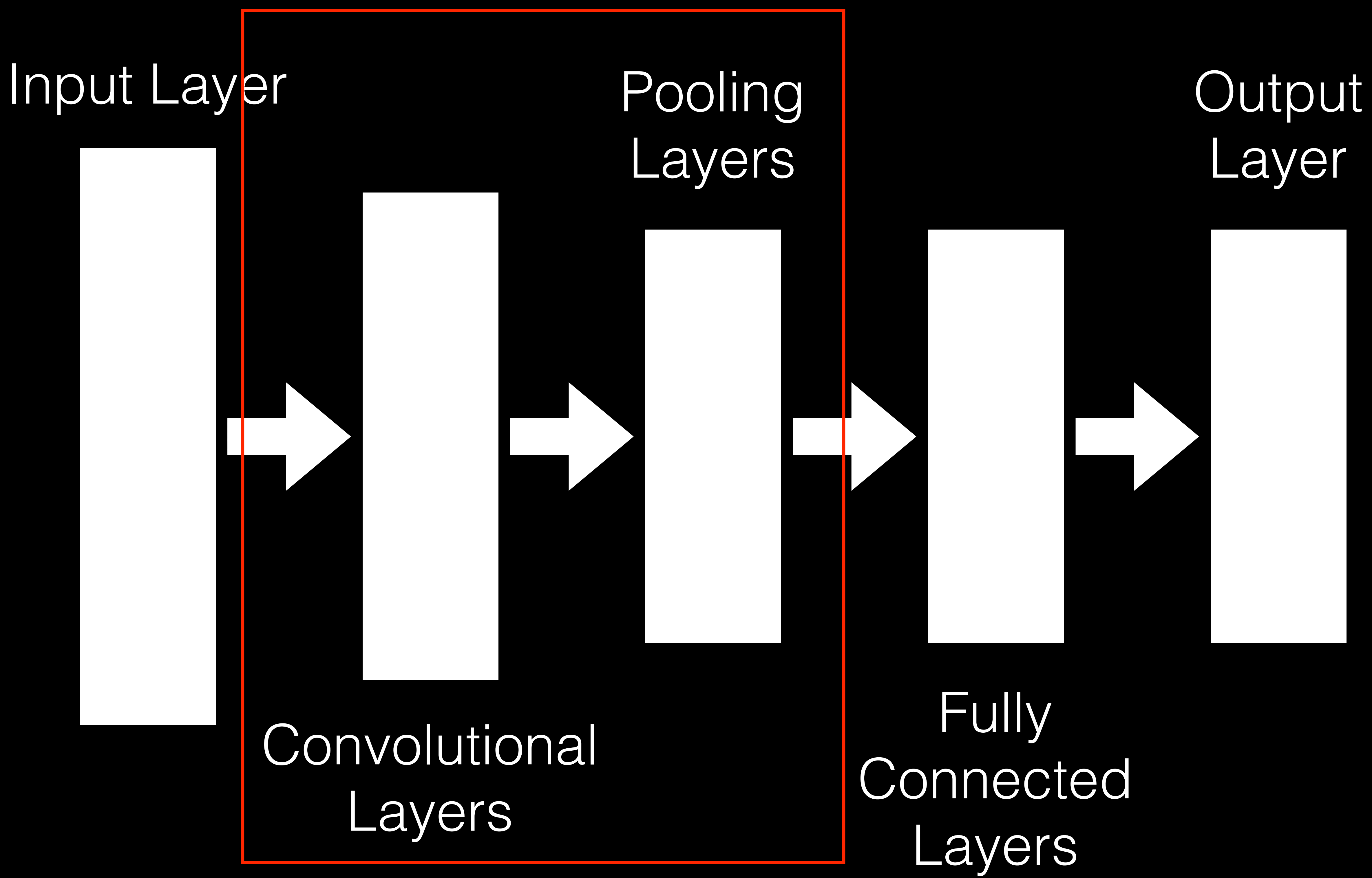

- This group of pixels is a **local receptive field**, and it's size is defined by the size of your kernel
- We then slide this kernel across the entire image

# Convolutional Layers

We apply multiple kernels to the image, which results in multiple learned kernels per hidden layer



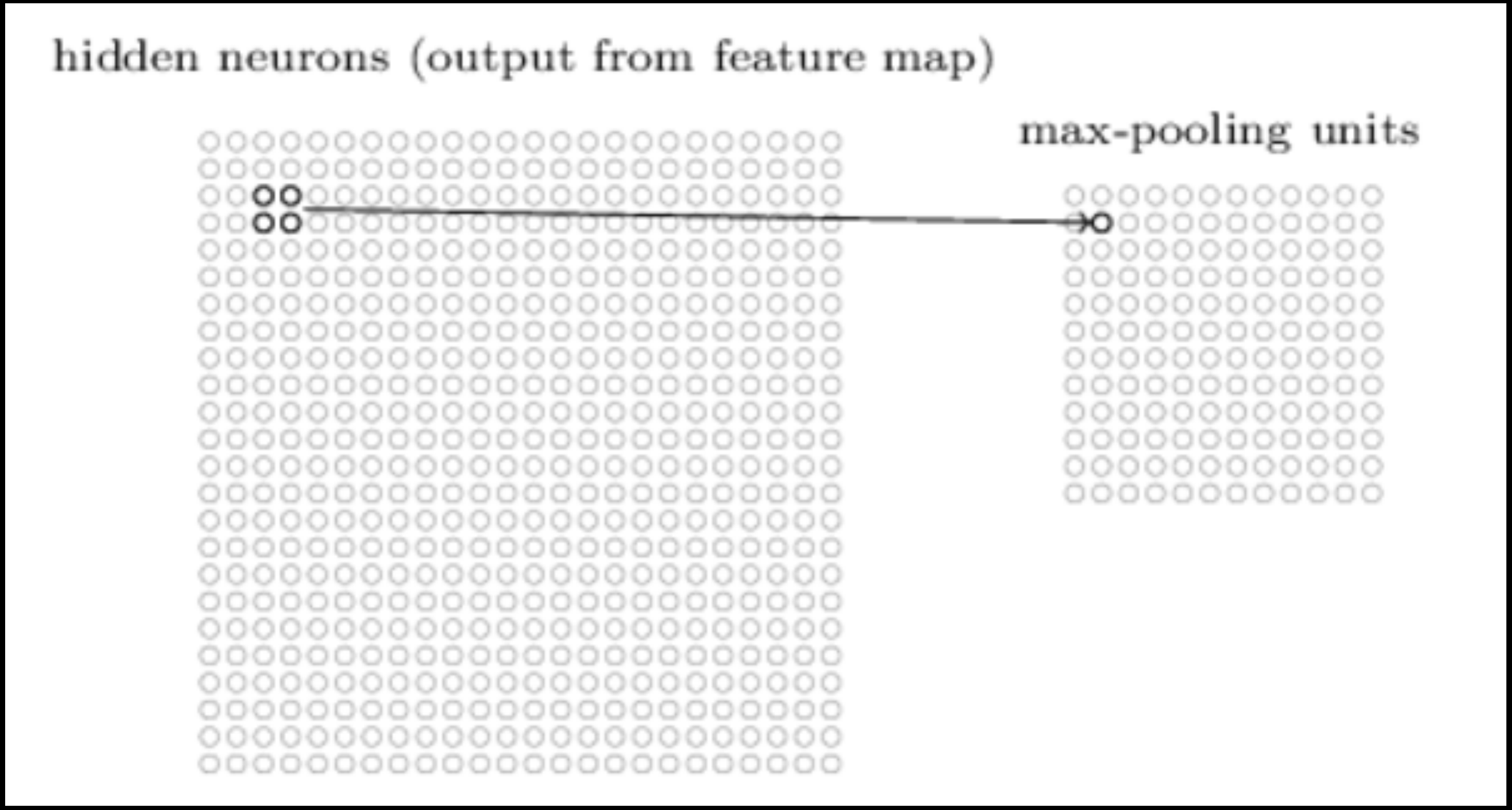
# General Structure



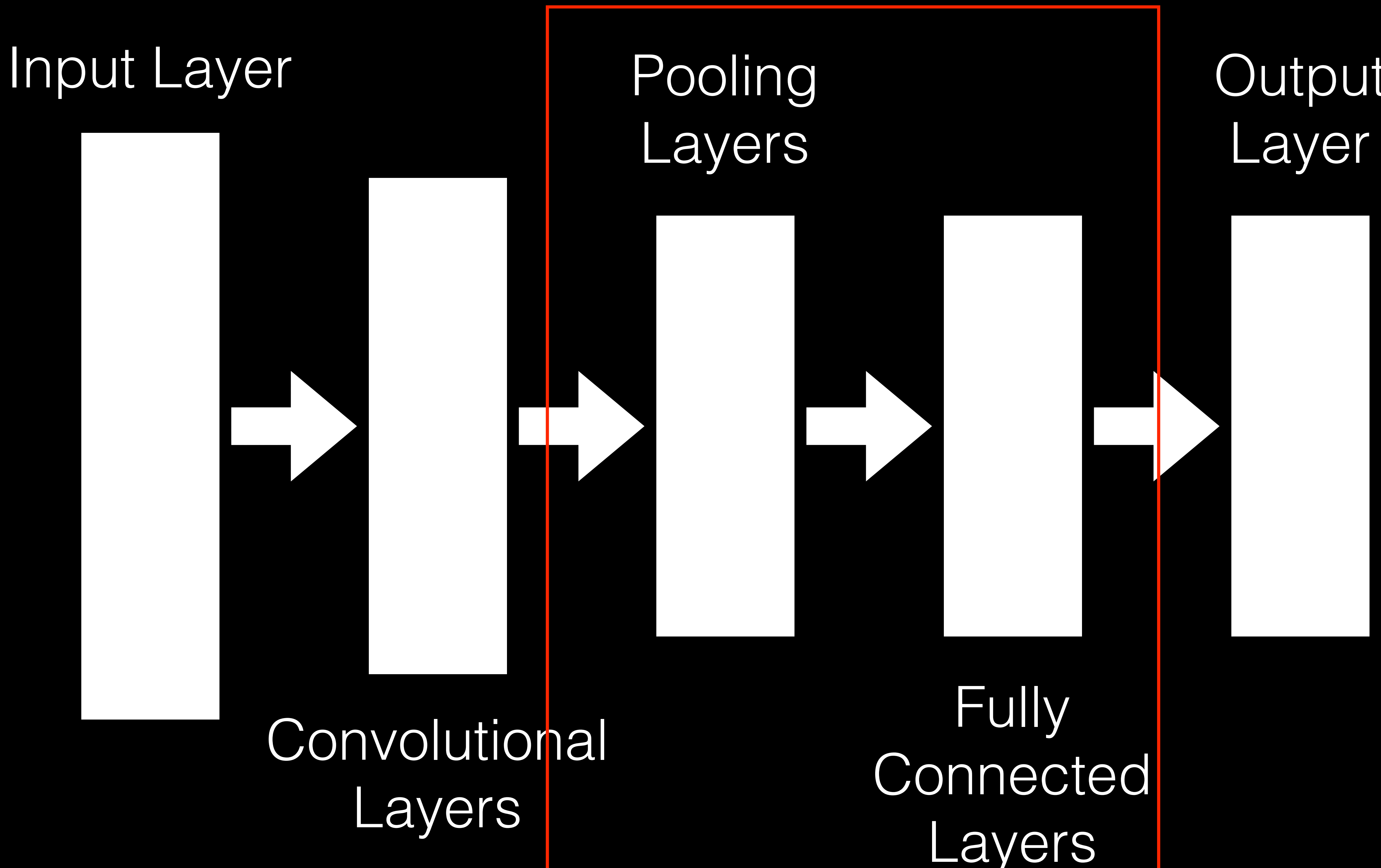
# Pooling Layers

- Used immediately after convolutional layers, and simplify the information in the output from the convolutional layer.
    - Reduces the computational complexity for later layers
    - Provides a form of translational invariance

# Max Pooling



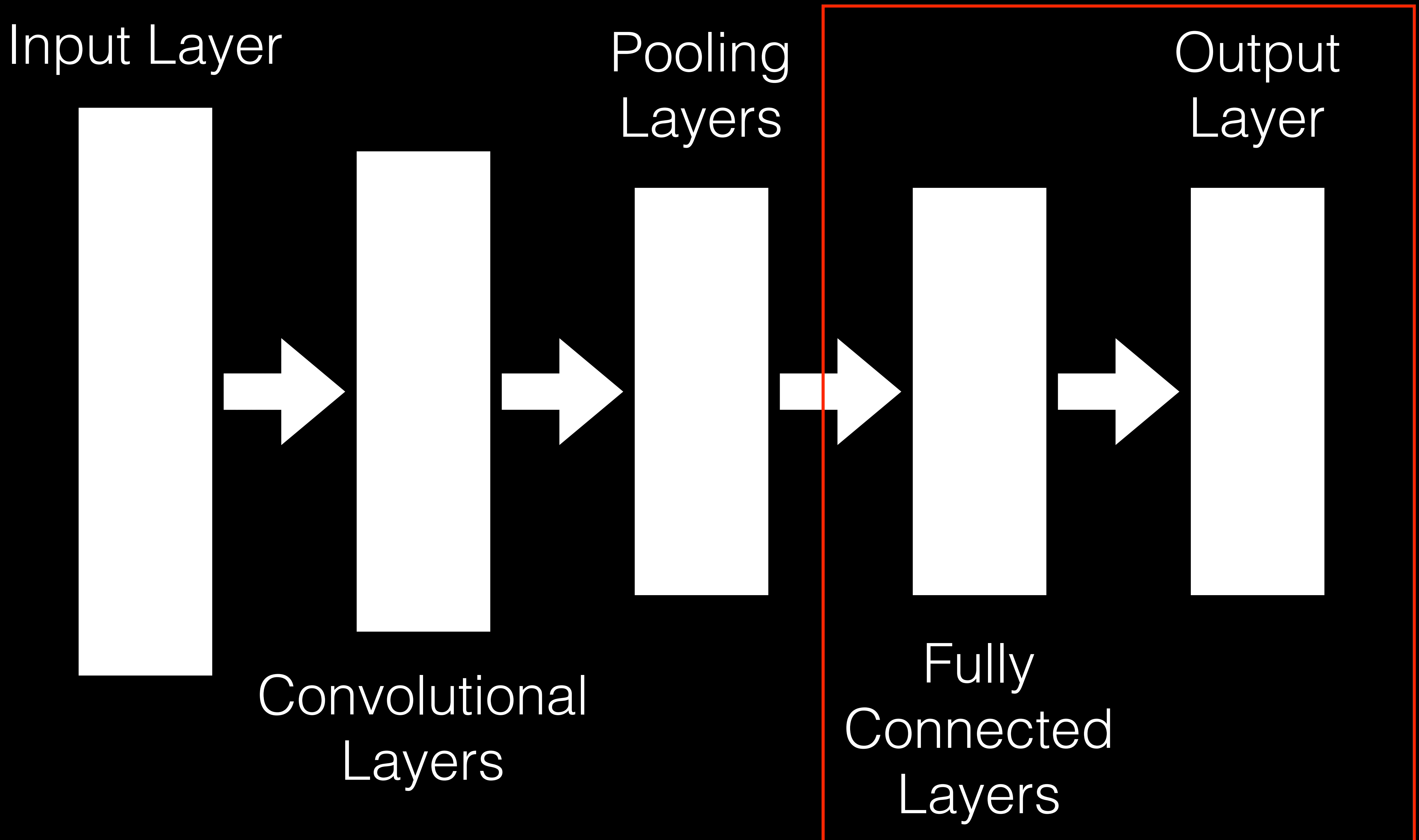
# General Structure



# Fully Connected Layers

Used to aggregate all of the information that has been learned in the convolutional and pooling layers.

# General Structure



# Output Layer

- For object recognition, always composed of softmax activation units
- Outputs the probability that your image is of a certain class

# Tips and Tricks

- We of course have all the image processing techniques that we learned about - resizing, denoising, etc. (denoising is not actually common to use with CNN's, but available)

- New set of image processing techniques for getting more images - rotate images, flip images, etc.

# General Structure Normalities

- It is not too common to use dropout after convolutional layers (but it is common to use it after you're fully connected layers)
- It's common to have multiple convolutional layers in between pooling layers
- RelU activation units are incredibly popular with CNN's

# Don’t know where to start?

If you’re confused about the general CNN structure that you should start off, you should find a research paper in your domain space that uses CNN’s.  Start off trying to get something working that uses the same structure they did, and go from there.