# Day 8: Convolutional Neural Networks

## Summer STEM: Machine Learning

Department of Electrical and Computer Engineering
NYU Tandon School of Engineering
Brooklyn, New York

July 1, 2020

# Outline

# Better performance with images

- Encoding locality
- How does an MLP see an image?
- Is this how we see images?

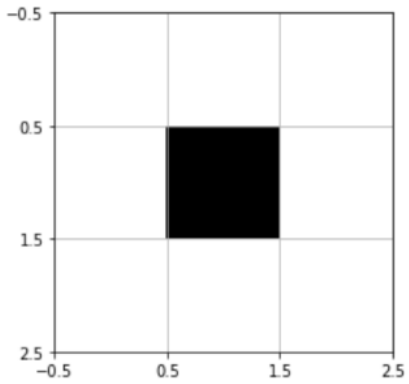
# Examples: Lena & Mandrill

# Outline

# Images in Computer

- Images are stored as arrays of quantized numbers in computers
- Gray scale image: 2D matrices with each entry specifying the intensity (brightness) of a pixel
  - Pixel values range from 0 to 255, 0 being the darkest, 255 being the brightest

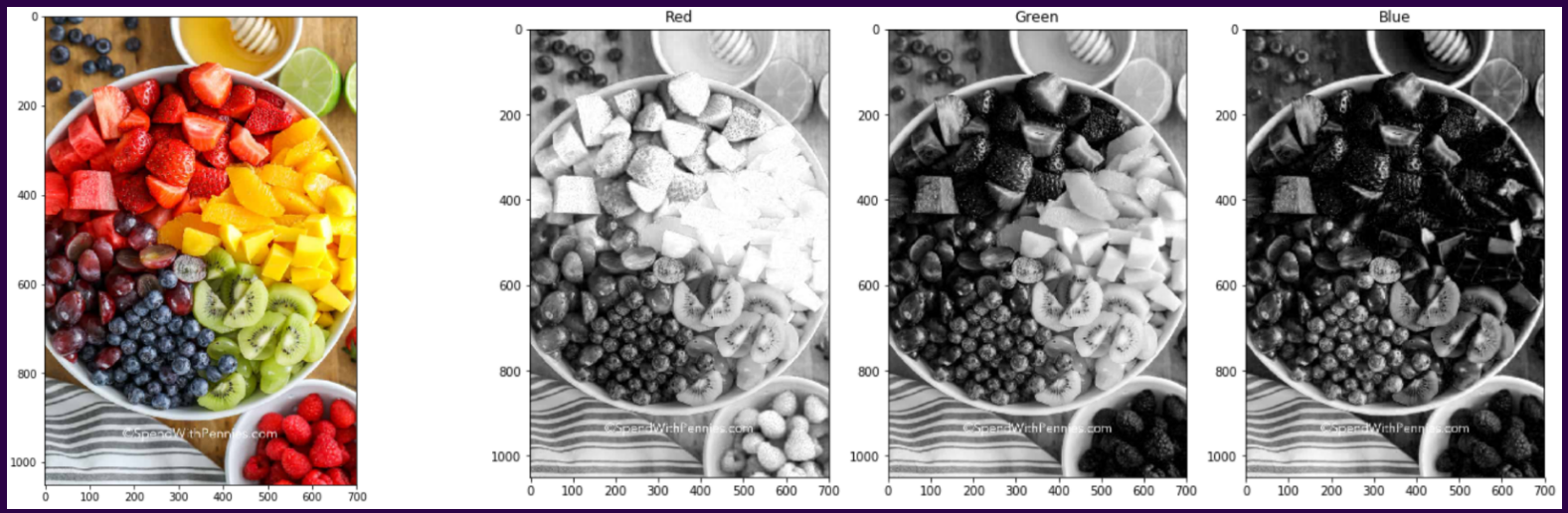
```
[[255 255 255]
 [255   0 255]
 [255 255 255]]
```




NYU TANDON SCHOOL OF ENGINEERING

# Color Images

- Color image: 3D array, 2 dimensions for space, 1 dimension for color
  - Can be thought of as three 2D matrices stacked together into a cube, each 2D matrix specify the amount of each color: Red ,Green ,Blue value at each pixel




- Shape of this image: (1050,700,3)
- There are 1050x700 pixels, 3 channels: R,G,B

# Outline

# Limitations of Fully Connected Network

- In Fashion MNIST, we used a fully connected network, in which each neuron in the hidden layer is connected to all 28x28 = 784 pixels

# Limitations of Fully Connected Network

- In Fashion MNIST, we used a fully connected network, in which each neuron in the hidden layer is connected to all 28x28 = 784 pixels
- Higher definition images often contain millions of pixels → It is not practical to use fully connected network

# Limitations of Fully Connected Network

- In Fashion MNIST, we used a fully connected network, in which each neuron in the hidden layer is connected to all 28x28 = 784 pixels
- Higher definition images often contain millions of pixels → It is not practical to use fully connected network
- Fully connected network treat each individual pixel as a feature, it does not utilize the positional relationship between pixels

# Convolution

- Introducing a new operation: Convolution

# Convolution

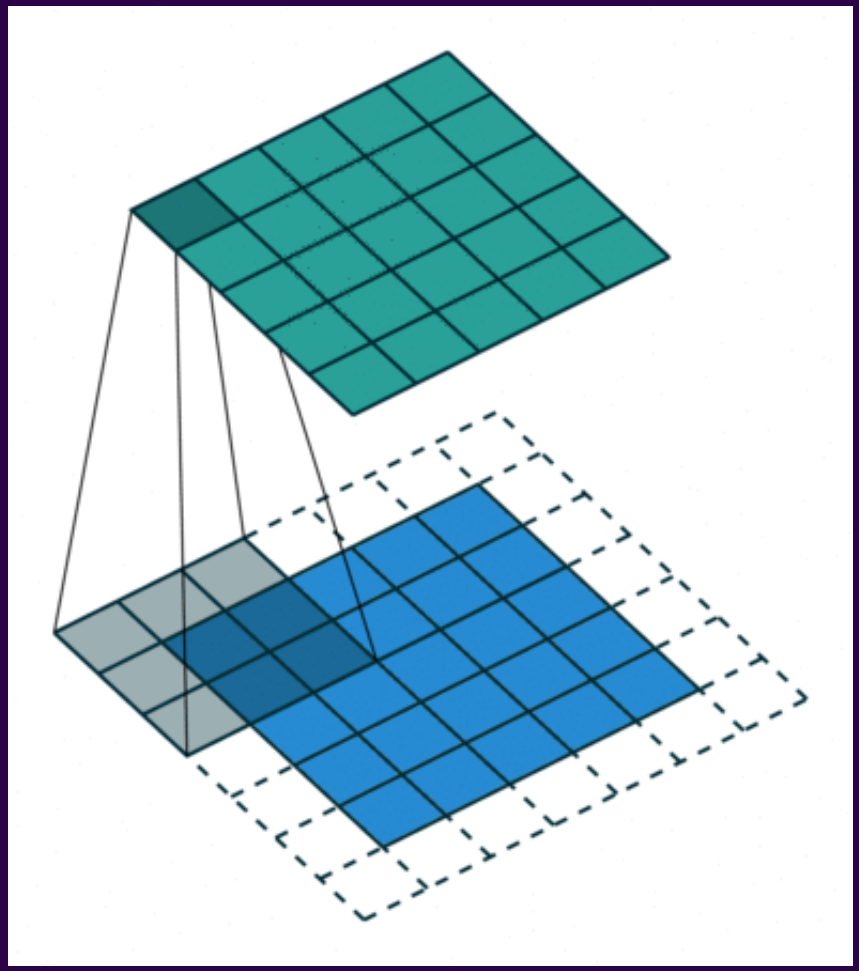
- Introducing a new operation: Convolution
- An operation on an image(matrix) $X$ with a kernel $W$

# Convolution

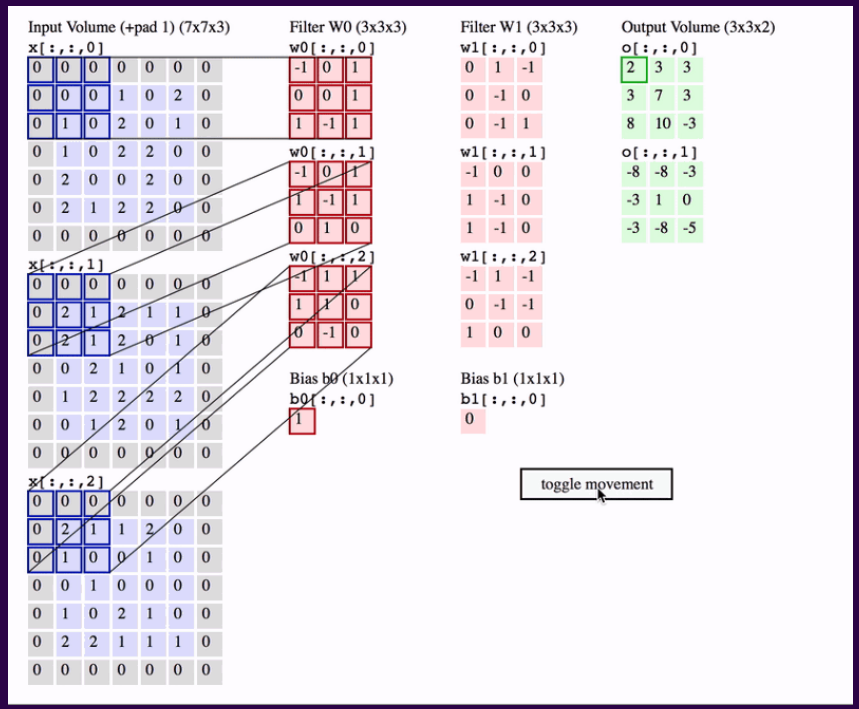
- Introducing a new operation: Convolution

- An operation on an image(matrix) $X$ with a kernel $W$
- $Z = X \circledast W$

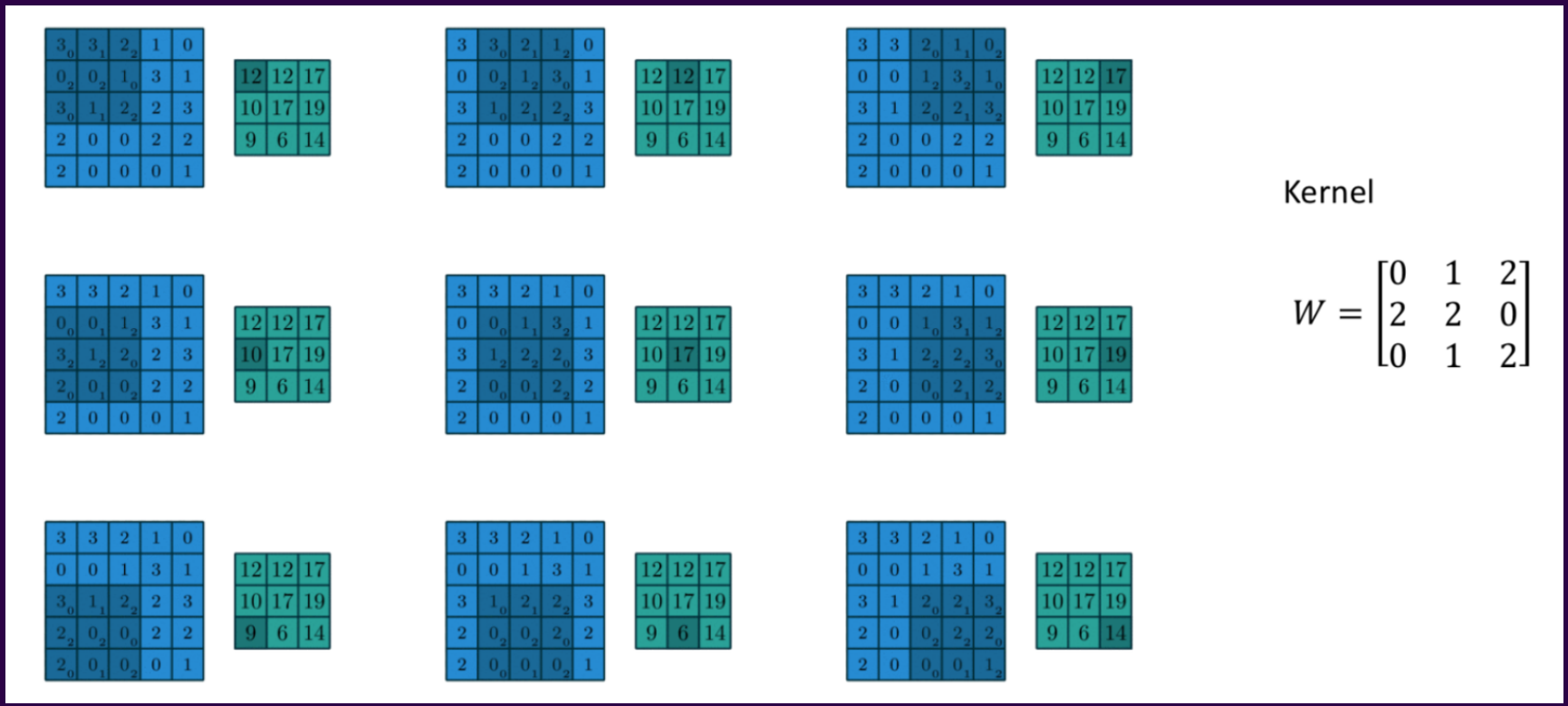
# Some Animations, Source: https://towardsdatascience.com



# Some Animations, Source: https://cs231n.github.io/convolutional-networks/



# Example of a Convolultion



# Why Convolution?

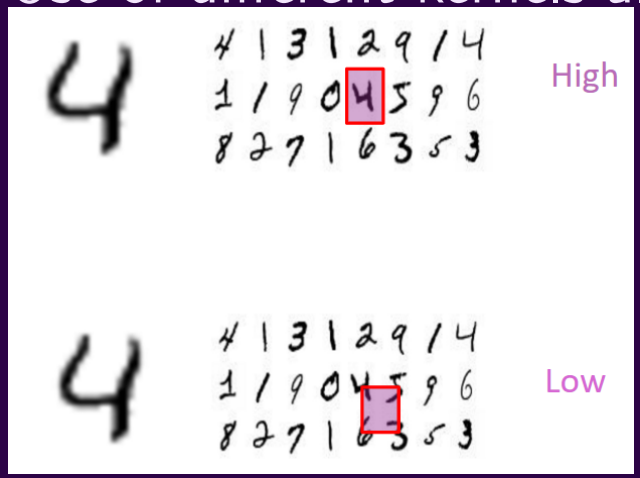
- With convolution, each output pixel depends on only the neighboring pixels in the input

# Why Convolution?

- With convolution, each output pixel depends on only the neighboring pixels in the input
- This allows us to learn the positional relationship between pixels

# Why Convolution?

- With convolution, each output pixel depends on only the neighboring pixels in the input
- This allows us to learn the positional relationship between pixels
- Use of different kernels allows us to detect features



# Convolution for Multiple Channels

- A kernel for each channel. Could be same kernel, or different
- Perform a convolution for each of the channel, with the respective kernel
- Sum the results

# Feature Maps

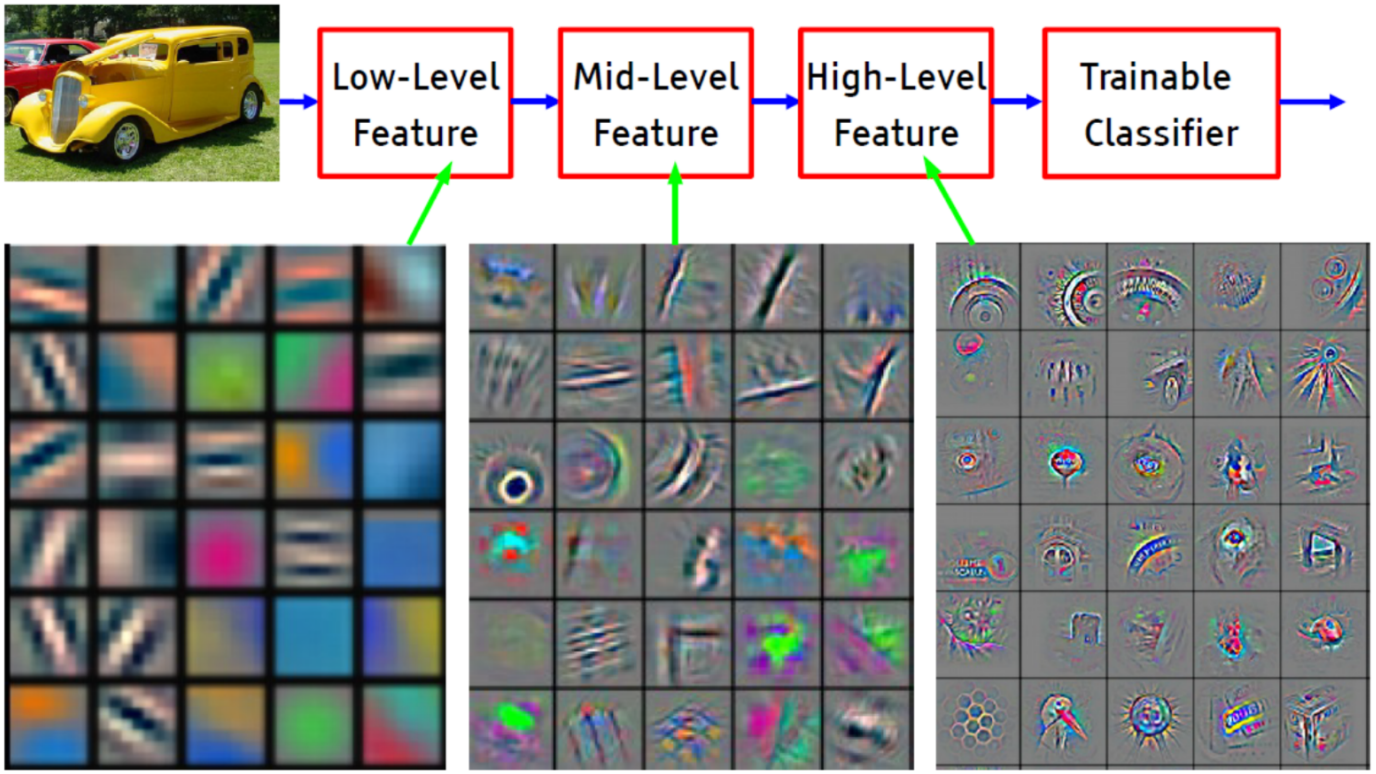



Figure: Figure from Y. LeCun Presentation, "What's wrong with deep learning?". Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013].

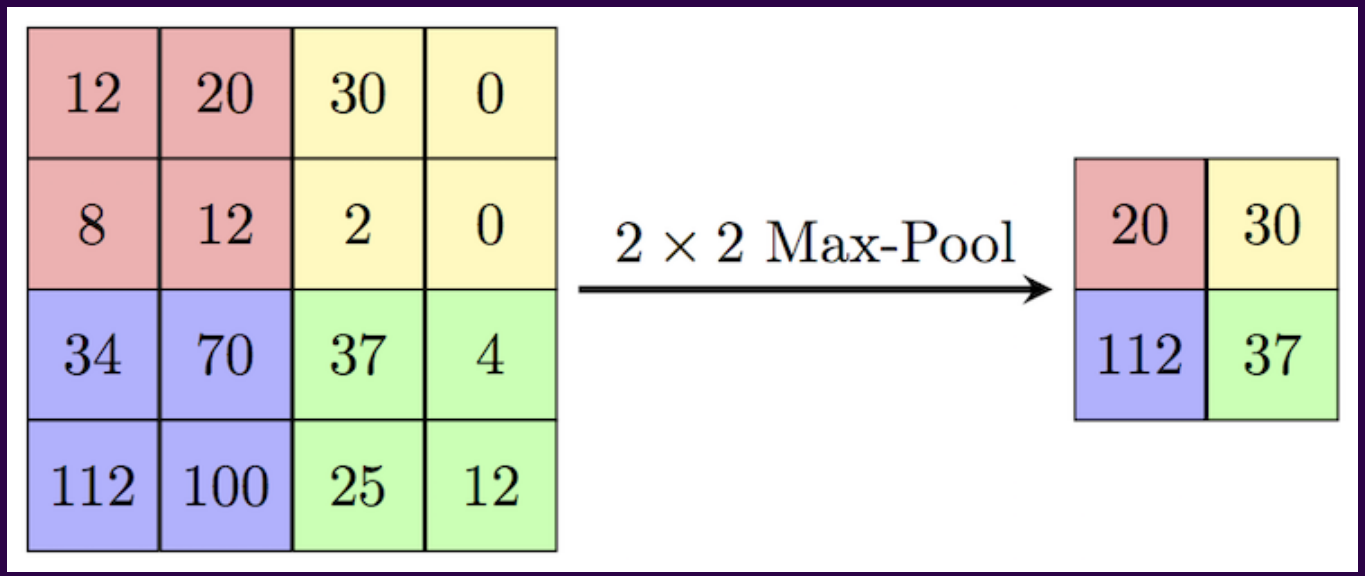
# Demo: Fashion MNIST

Open `demo_fashion_mnist.ipynb`

# Outline

# Max-Pooling

- Down-samples the inputs
- Provides translation invariance. Why?
- Apply after activation!

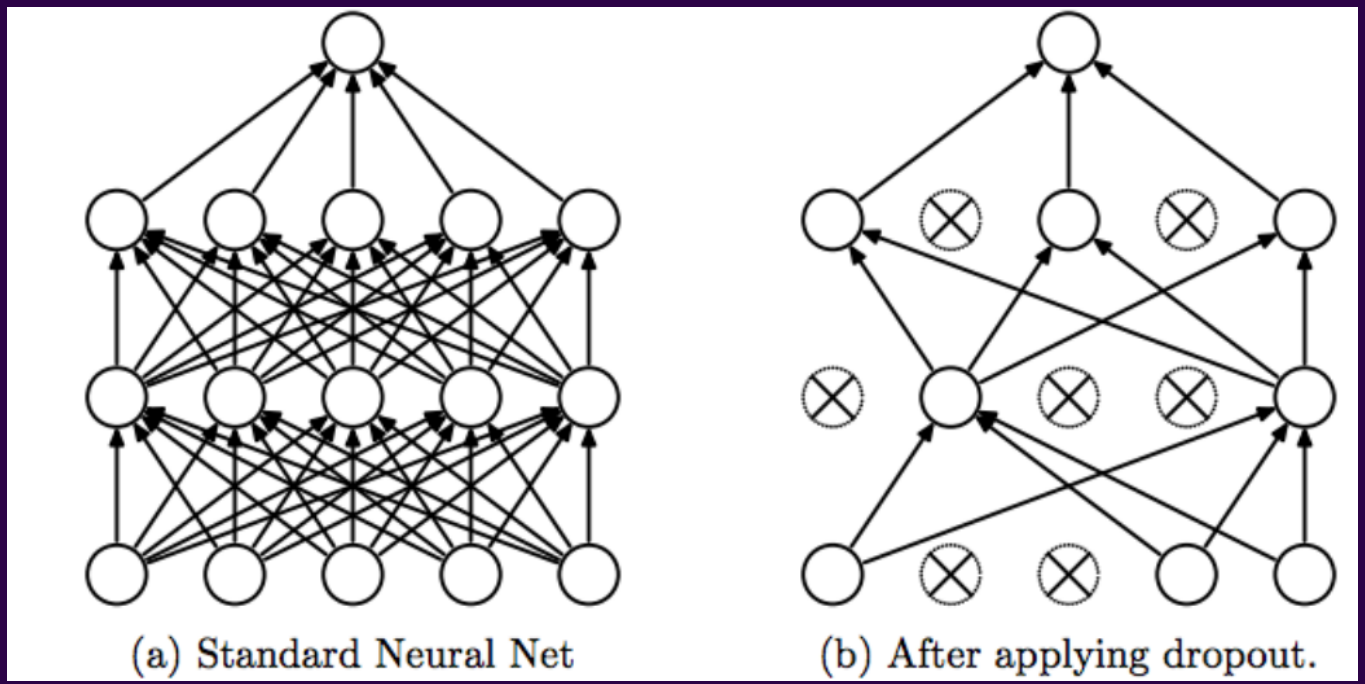


# Batch Normalization

- We normalize the inputs to the network. Why not do that for the inputs to the hidden layers?
- Batch norm: normalize the inputs to a layer for each mini-batch.
- Apply before activation!

# Dropout

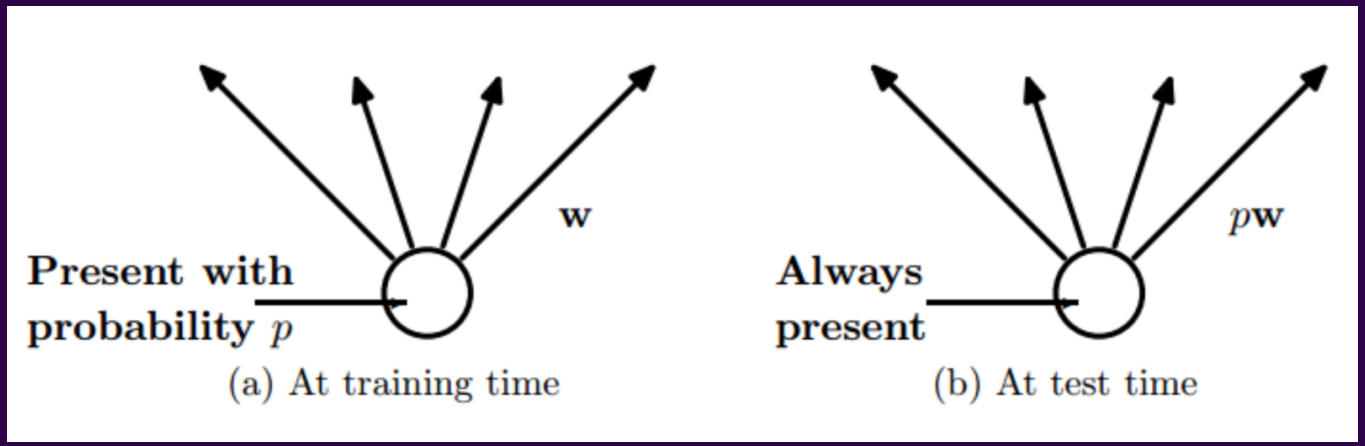
- Patented by Google
- Randomly disable neurons and their connections between each other.



(a) Standard Neural Net (b) After applying dropout.

# Dropout

- In train() mode, each neuron is present with probability $p$.
- In eval() mode, multiply the weights with $p$.
- Apply after activation!



# Lab: Cats vs. Dogs

Open `lab_cats_vs_dogs.ipynb`