

Day 7: Convolutional Neural Networks

Summer STEM: Machine Learning

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

June 28, 2022

Outline

- 1 Motivation
- 2 Dealing with Images in Computers
- 3 Convolution
- 4 Data augmentation
- 5 Normalization
- 6 Dropout
- 7 Transfer Learning

Better performance with images

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

Examples: Lena & Mandrill



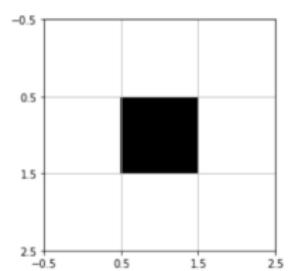
Outline

- 1 Motivation
- 2 Dealing with Images in Computers
- 3 Convolution
- 4 Data augmentation
- 5 Normalization
- 6 Dropout
- 7 Transfer Learning

Images in Computer

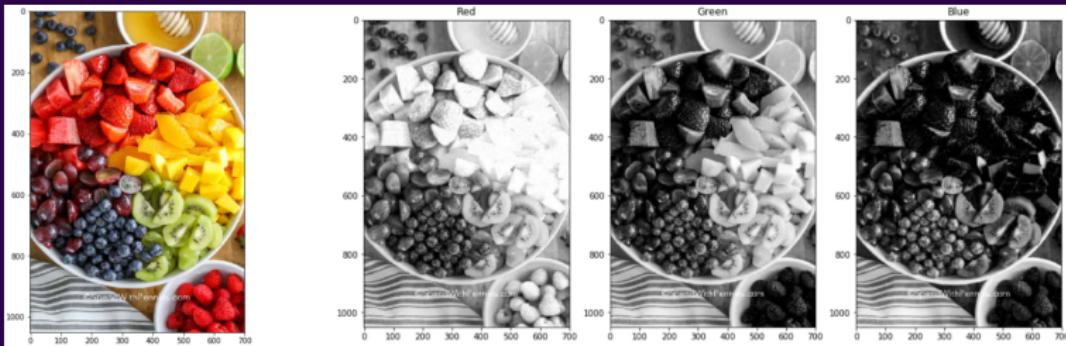
- Images are stored as arrays of quantized numbers in computers
- Gray scale image: 2D matrices with each entry specifying the intensity (brightness) of a pixel
 - Pixel values range from 0 to 255, 0 being the darkest, 255 being the brightest

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



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

- 1 Motivation
- 2 Dealing with Images in Computers
- 3 Convolution
- 4 Data augmentation
- 5 Normalization
- 6 Dropout
- 7 Transfer Learning

Limitations of Fully Connected Network

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

Limitations of Fully Connected Network

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

Limitations of Fully Connected Network

- In MNIST, we used a fully connected network, in which each neuron in the hidden layer is connected to all $28 \times 28 = 784$ 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
- $Z = X \circledast W$

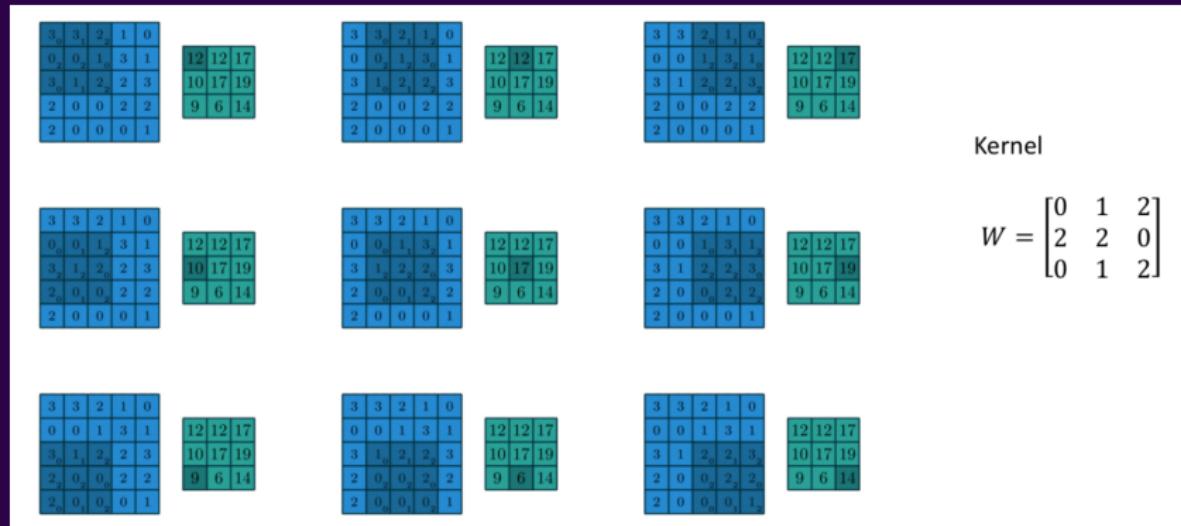
Some Animations, Source:

<https://towardsdatascience.com>

Some Animations, Source:

<https://cs231n.github.io/convolutional-networks/>

Example of a Convolution



Why Convolution?

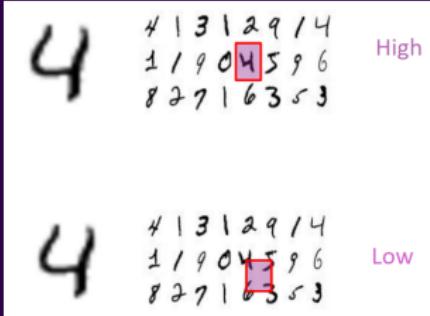
- With convolution, each output pixel depends on only the neighboring pixels in the input

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?

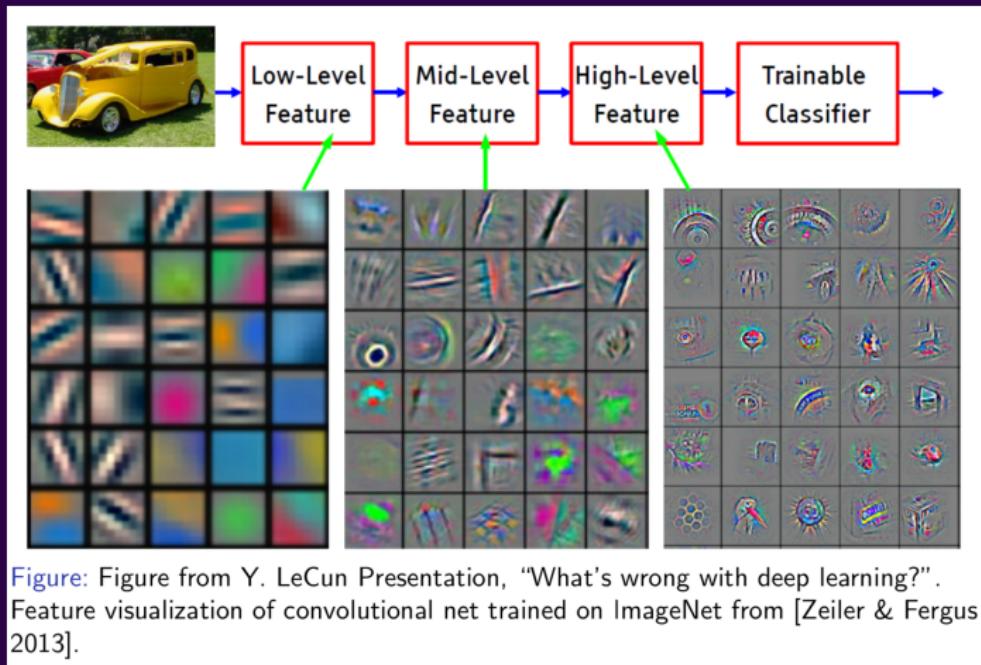
- With convolution, each output pixel depends on only the neighboring pixels in the input
- This allows us to learn the positional relationship between pixels
- Use of different kernels allows us to detect features



Convolution for Multiple Channels

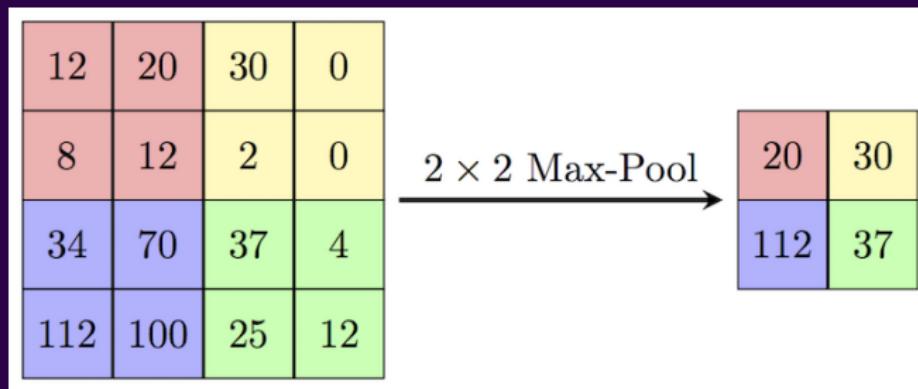
- A kernel for each channel. Could be same kernel, or different
- Perform a convolution for each of the channel, with the respective kernel
- Sum the results

Feature Maps



Max-Pooling

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



Outline

- 1 Motivation
- 2 Dealing with Images in Computers
- 3 Convolution
- 4 Data augmentation
- 5 Normalization
- 6 Dropout
- 7 Transfer Learning

Data augmentation

- Image classification is a difficult task
- We need more data !
- Labeling is expensive and time-consuming.
- How can we create new images ?

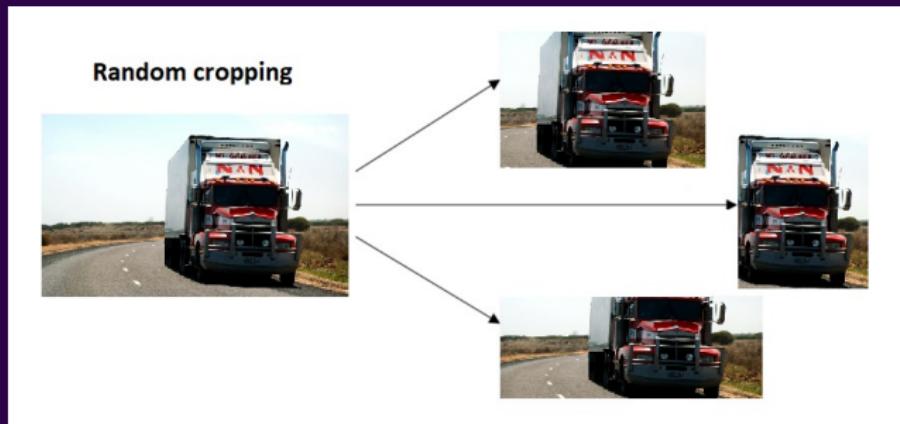
Data augmentation



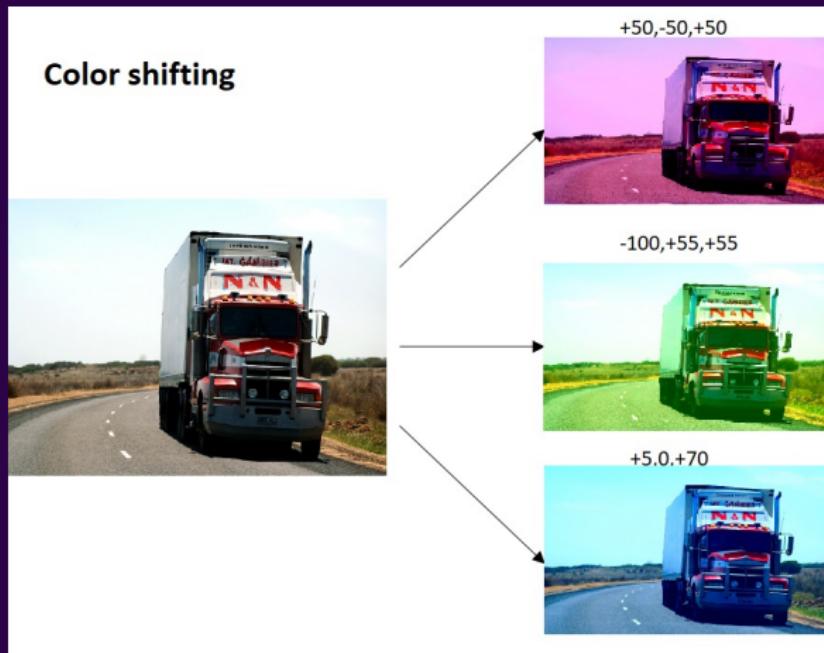
Mirroring



Data augmentation



Data augmentation



<http://datahacker.rs/deep-learning-data-augmentation>

Data augmentation



<https://medium.com/@wolframalphav1.0/easy-way-to-improve-image-classifier-performance-part-1-mixup-augmentation-with-codes-33288db9f8e0>

Outline

- 1 Motivation
- 2 Dealing with Images in Computers
- 3 Convolution
- 4 Data augmentation
- 5 Normalization
- 6 Dropout
- 7 Transfer Learning

Data Normalization

- Given the dataset (x_i, y_i) for $i = 1, 2, \dots, N$

- Mean: $\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$

- Variance: $\sigma^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2$

- Standard deviation : σ ($= \sqrt{\sigma^2}$)

Data Normalization

- Given the dataset (x_i, y_i) for $i = 1, 2, \dots, N$
- Mean: $\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$
- Variance: $\sigma^2 = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2$
- Standard deviation : σ ($= \sqrt{\sigma^2}$)
- **Normalization** : Replace each x_i by $x'_i = \frac{x_i - \bar{x}}{\sigma}$
- The new dataset will have a mean of 0 and a variance of 1.

Data Normalization

■ Proof :

$$\begin{aligned}\bar{x}' &= \frac{1}{N} \sum_{i=1}^N x'_i \\ &= \frac{1}{N} \sum_{i=1}^N \frac{x_i - \bar{x}}{\sigma} \\ &= \frac{1}{N\sigma} \left(\sum_{i=1}^N x_i - \sum_{i=1}^N \bar{x} \right)\end{aligned}$$

Data Normalization

■ Proof :

$$\begin{aligned}\bar{x}' &= \frac{1}{N} \sum_{i=1}^N x'_i \\ &= \frac{1}{N} \sum_{i=1}^N \frac{x_i - \bar{x}}{\sigma} \\ &= \frac{1}{N\sigma} \left(\sum_{i=1}^N x_i - \sum_{i=1}^N \bar{x} \right) \\ &= \frac{1}{N\sigma} (N\bar{x} - N\bar{x}) = 0\end{aligned}$$

Data Normalization

■ Proof :

$$\sigma_{x'}^2 = \frac{1}{N} \sum_{i=1}^N (x'_i - \bar{x}')^2$$

Data Normalization

■ Proof :

$$\begin{aligned}\sigma_{x'}^2 &= \frac{1}{N} \sum_{i=1}^N (x'_i - \bar{x}')^2 \\ &= \frac{1}{N} \sum_{i=1}^N (x'_i - 0)^2\end{aligned}$$

Data Normalization

■ Proof :

$$\begin{aligned}\sigma_{x'}^2 &= \frac{1}{N} \sum_{i=1}^N (x'_i - \bar{x}')^2 \\ &= \frac{1}{N} \sum_{i=1}^N (x'_i - 0)^2 \\ &= \frac{1}{N} \sum_{i=1}^N \left(\frac{x_i - \bar{x}}{\sigma} \right)^2\end{aligned}$$

Data Normalization

■ Proof :

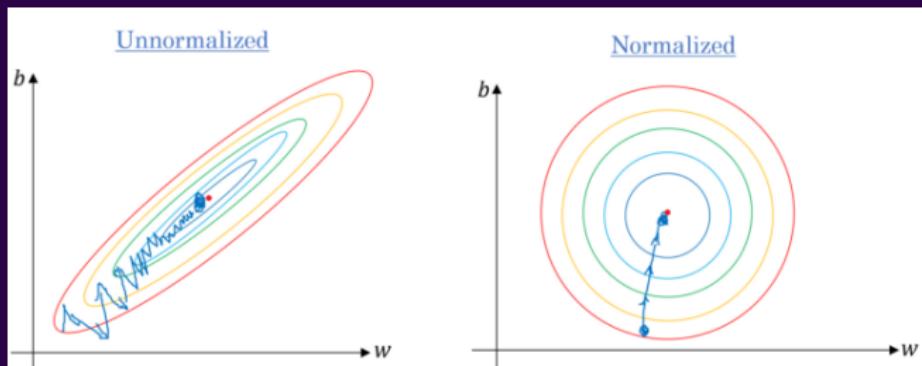
$$\begin{aligned}\sigma_{x'}^2 &= \frac{1}{N} \sum_{i=1}^N (x'_i - \bar{x}')^2 \\ &= \frac{1}{N} \sum_{i=1}^N (x'_i - 0)^2 \\ &= \frac{1}{N} \sum_{i=1}^N \left(\frac{x_i - \bar{x}}{\sigma} \right)^2 \\ &= \frac{1}{\sigma^2} \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2\end{aligned}$$

Data Normalization

■ Proof :

$$\begin{aligned}\sigma_{x'}^2 &= \frac{1}{N} \sum_{i=1}^N (x'_i - \bar{x}')^2 \\ &= \frac{1}{N} \sum_{i=1}^N (x'_i - 0)^2 \\ &= \frac{1}{N} \sum_{i=1}^N \left(\frac{x_i - \bar{x}}{\sigma} \right)^2 \\ &= \frac{1}{\sigma^2} \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2 \\ &= \frac{1}{\sigma^2} \sigma^2 = 1\end{aligned}$$

Data Normalization



<https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3>

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!

Batch Normalization

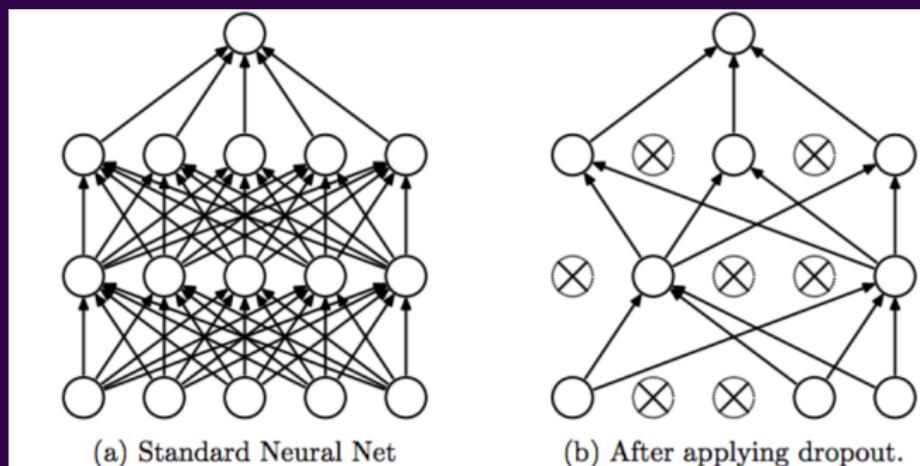
```
model = models.Sequential()  
# More layers  
model.add(layers.Conv2D(64, (3, 3)))  
model.add(layers.BatchNormalization())  
model.add(layers.Activation('relu'))  
# More layers  
model.add(layers.Flatten())  
model.add(layers.Dense(64))  
model.add(layers.BatchNormalization())  
model.add(layers.Activation('relu'))  
model.add(layers.Dense(1, activation='sigmoid'))
```

Outline

- 1 Motivation
- 2 Dealing with Images in Computers
- 3 Convolution
- 4 Data augmentation
- 5 Normalization
- 6 Dropout
- 7 Transfer Learning

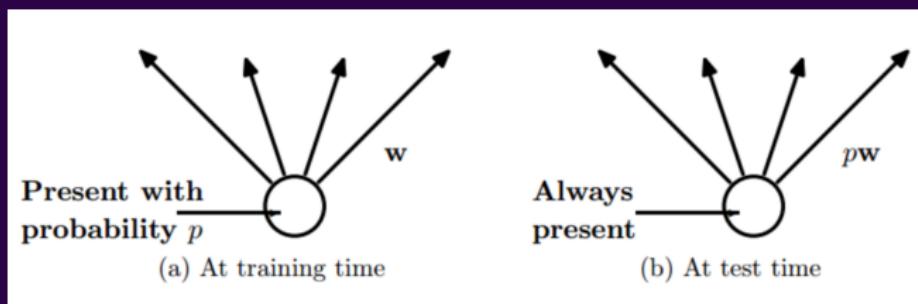
Dropout

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



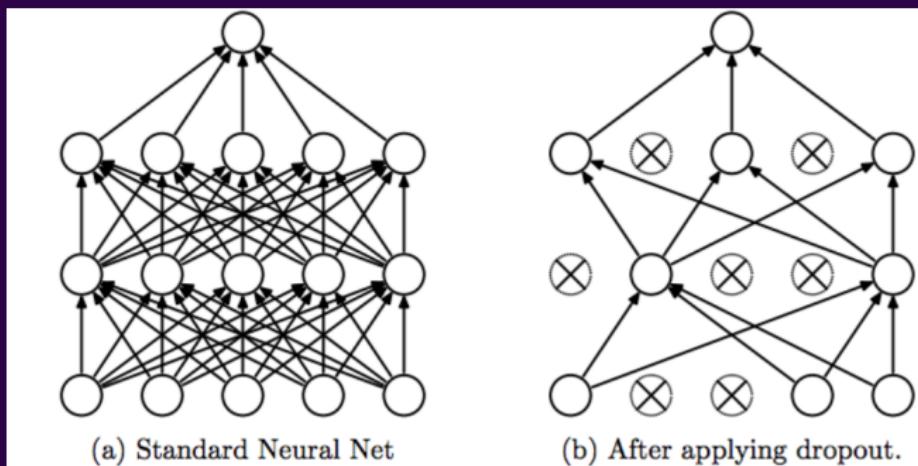
Dropout

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



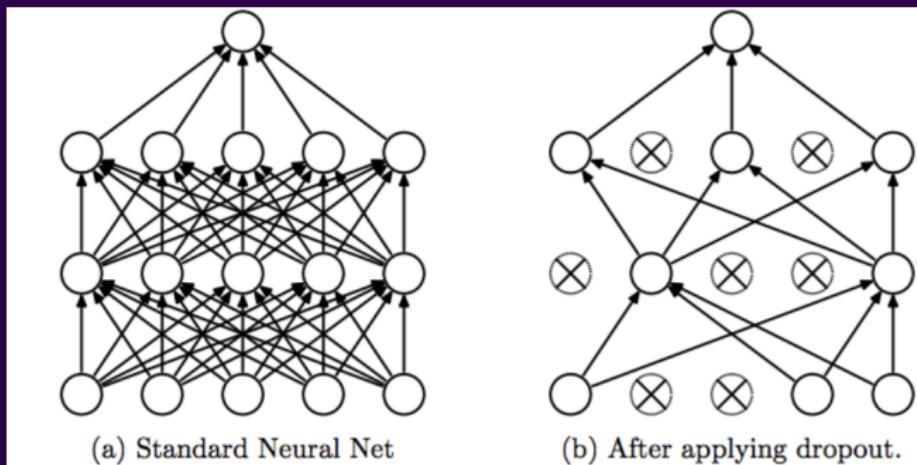
Dropout

- This is the same as using a neural network with the same amount of layers but less neurons per layer.
- The more neurons the more powerful the neural network is, and the more likely it is to overfit.



Dropout

- This also means that the model can not rely on any single feature, therefore would need to spread out the weights.
- When spreading out the weights the size also shrinks, thus giving a similar effect to L2 regularization



Outline

- 1 Motivation**
- 2 Dealing with Images in Computers**
- 3 Convolution**
- 4 Data augmentation**
- 5 Normalization**
- 6 Dropout**
- 7 Transfer Learning**

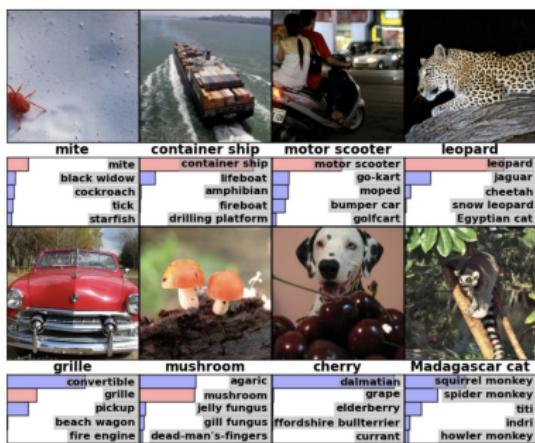
Open Source Implementation

- If you are interested in building on top of a deep learning research paper it is a good idea to first go online and see if there is a nice open source implementation instead of starting from scratch.
- If you are in the field of computer vision, many networks would require an extensive hyperparameter search and multiple GPUs to train, this process might take weeks.
- People now open source their results along with the weights and these can be seen as a nice initialization to your application.

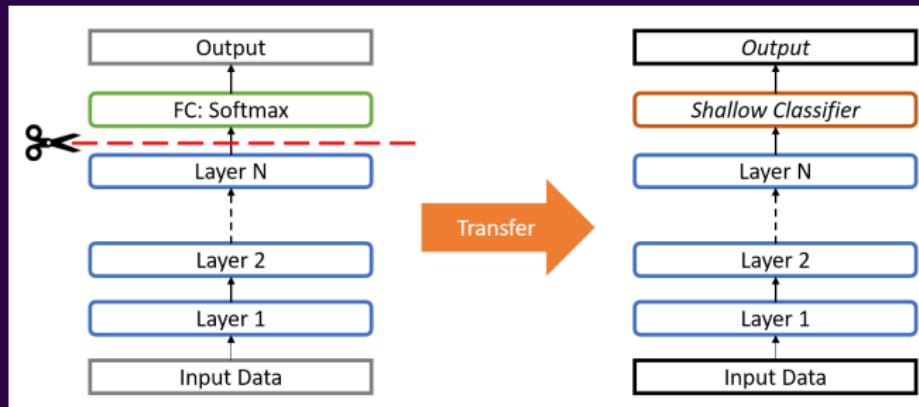
Imagenet Challenge



- 1,000 object classes (categories).
- Images:
 - 1.2 M train
 - 100k test.



Transfer Learning



<https://www.oreilly.com/library/view/hands-on-transfer-learning/9781788831307/d94586c6-1c46-4794-aded-22442a4f81d8.xhtml>

Transfer Learning

- You can freeze the early layers and replace the last few layers to match your own application needs (e.g. different number of classes, different activation functions).
- Only train the replaced layers and use the weights of the early layers "as-is".
- This is similar to transferring the knowledge from one network to another, thus the name transfer learning.

Lab: Cats vs. Dogs

Open lab_transfer_learning_dog_cat.ipynb