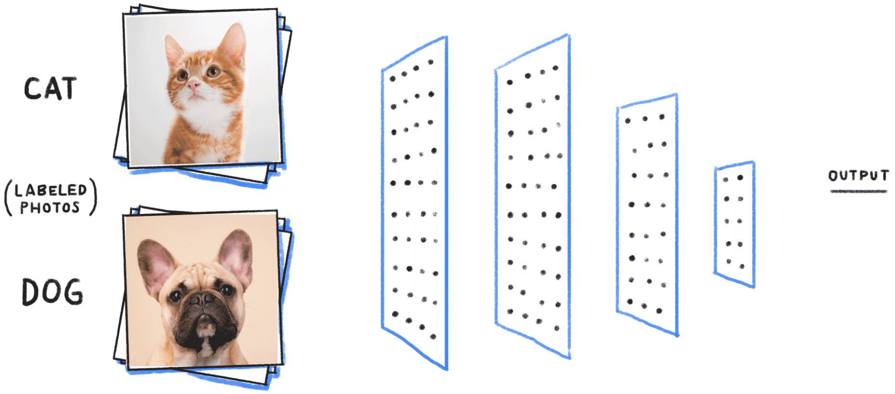
An introduction to Convolutional Neural Networks

Matthew Vue and Maryam Vazirabad

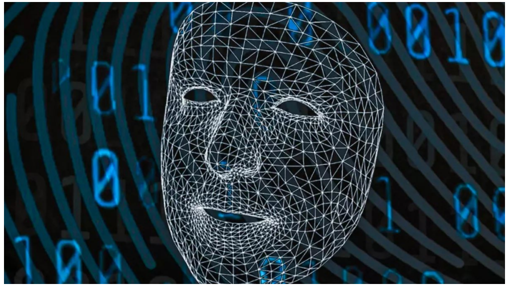
What are convolutional neural networks?

- Convolutional neural networks are primarily used as a form of image classification or image recognition



Where are they used?

- Convolutional networks can used for a wide range of tasks such as facial recognition or analyzing documents (Natural Language Processing)



General CNN architecture (the layers)

1) Convolutional layer
2) ReLU Layer
3) Pooling Layer
4) Fully Connected Layer

Convolutional layer

- The convolutional Layer is used for detecting patterns in an image
  - Edges or curves
- Filters
  - Detects patterns in an image by scanning over it

0	0	1	1	0	0
0	1	0	0	1	0
0	0	1	1	0	0
0	1	0	0	1	0
0	0	1	1	0	0

Represented as 0’s and 1’s

3x3 filter

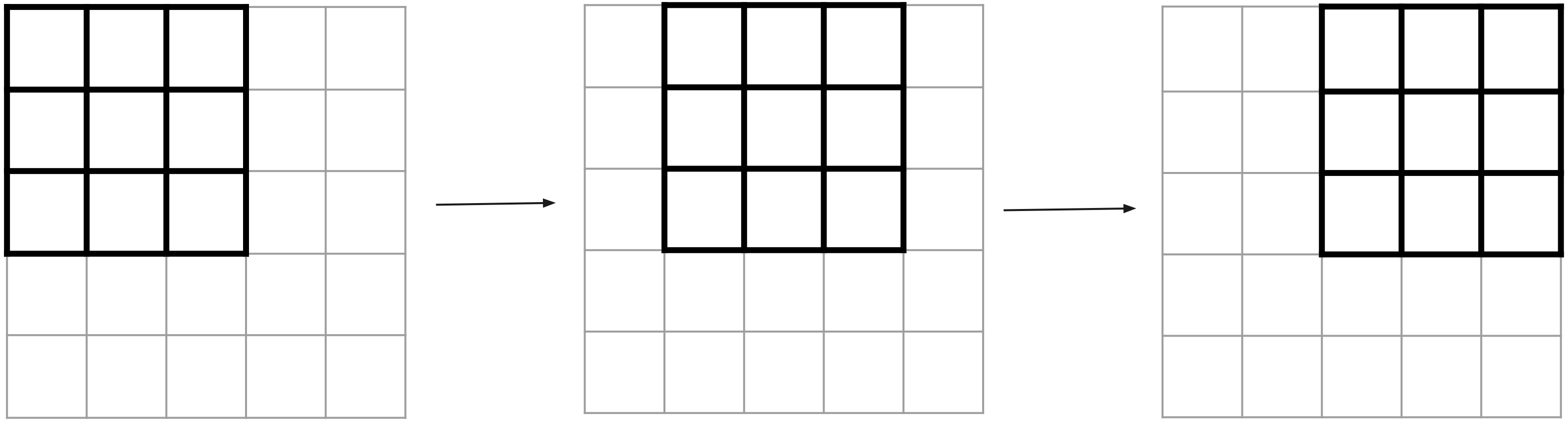
1	0	0
0	1	0
0	0	1

Scans the image according to a specified stride

Image

0	0	1	1	0	0
0	1	0	0	1	0
0	0	1	1	0	0
0	1	0	0	1	0
0	0	1	1	0	0

Stride = 1



1	0	0
0	1	0
0	0	1

Filter

0	0	1	1	0	0
0	1	0	0	1	0
0	0	1	1	0	0
0	1	0	0	1	0
0	0	1	1	0	0

Image represented as 0's and 1's

- As the filters scans the image, the sum of element wise matrix multiplication is calculated

1*0 + 0*0 + 0*1 + 0*0 + 1*1 + 0*0 + 0*0 + 0*0 + 1*1 = 2

2				

Examples of Filters used to detect patterns

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

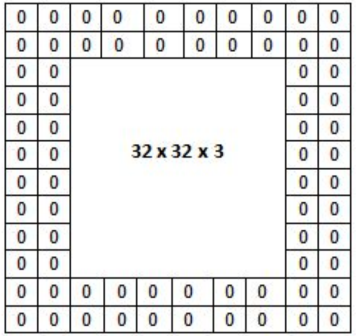
Can be used to detect top edges

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

Can be used to detect bottom edges

Hyperparameters and Common Configurations

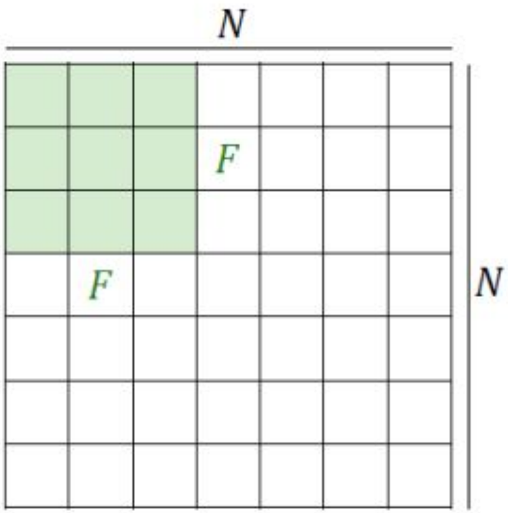
- Hyperparameters
  - Number of filters (K)
    - How many times you are scanning over the image with different filters
  - Filter Size (F)
  - Stride (S)
    - How much the filter moves across the image
  - Zero Padding (P)
    - Puts P borders of zeros around the picture
    - To preserve as much information about the original input



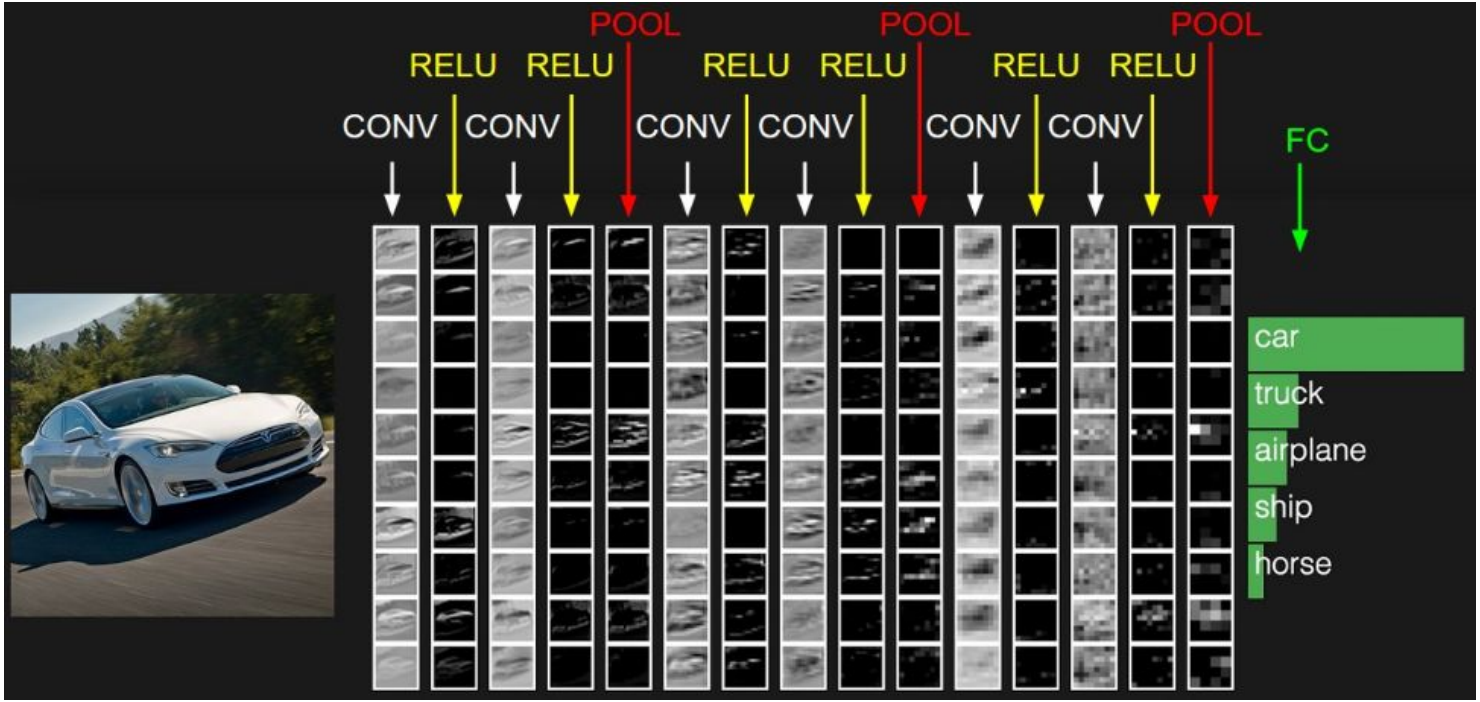
Common Configurations:

- K = 32, 64, 128, ...
- F = 3, S = 1, P = 1
- F = 5, S = 1, P = 2
- F = 5, S = 2, P = 2

Calculating Output Size

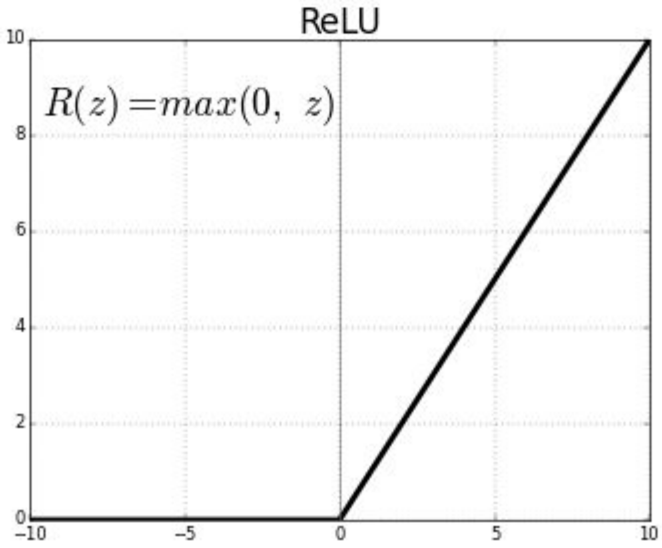


- Output size = (N-F) / S + 1
  - Stride = 1, N =6, F = 3
  - Output size = (6-3) / 1 + 1 = 4

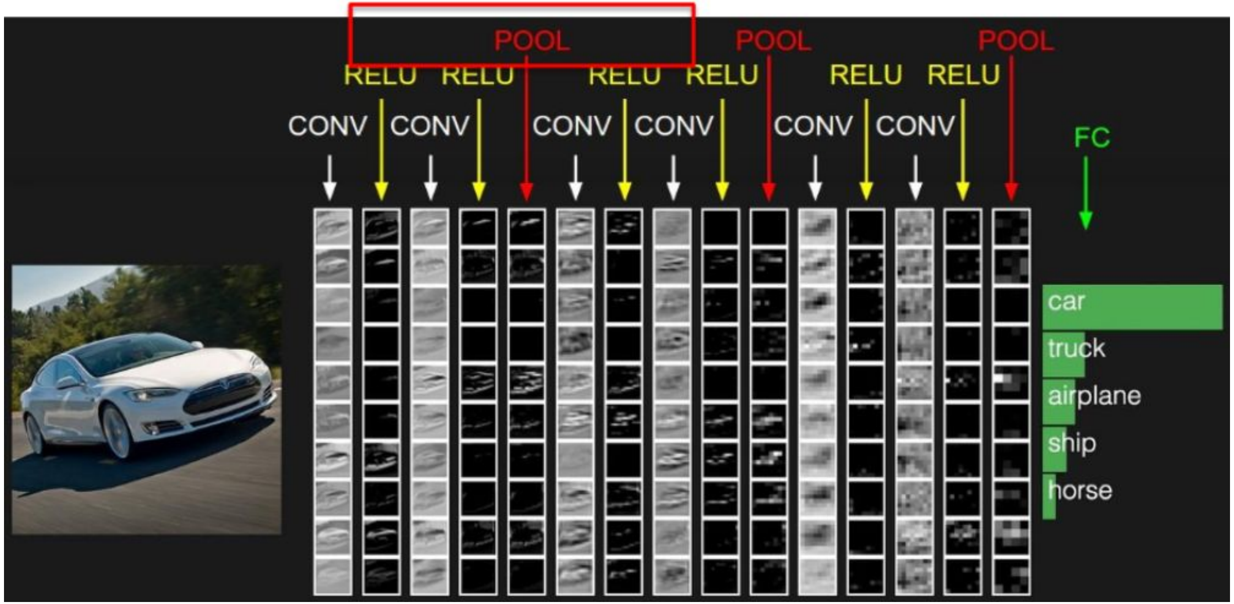

POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

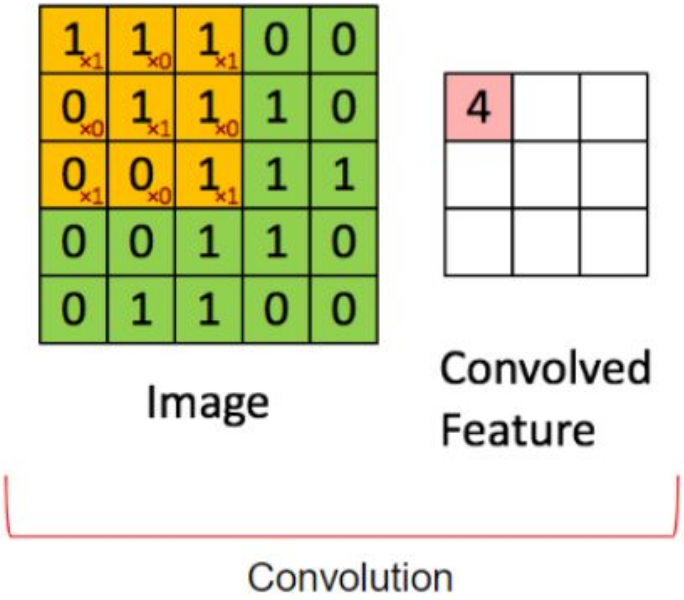
Beyond Convolution: Activation Layer

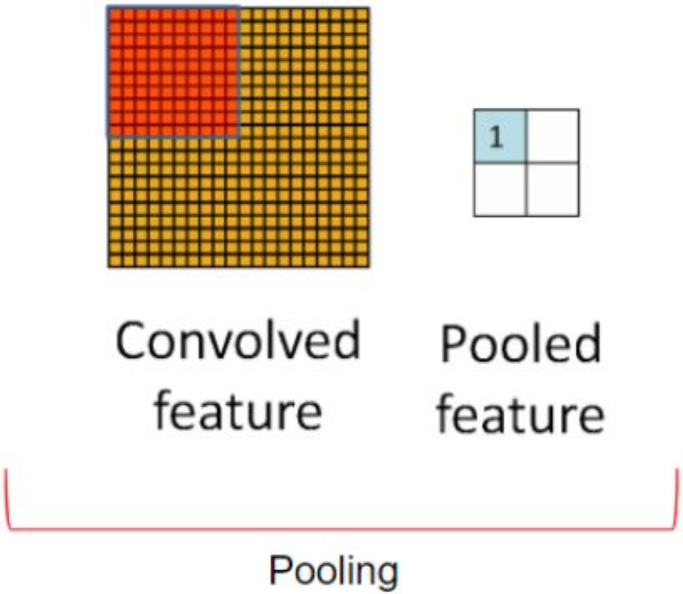
- Applying the ReLu (Rectified Linear Unit) function to hidden layers
- Images are made of different objects that are not linear to each other.
- We apply ReLu to increase non-linearity in the CNN



Pooling


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse


1 ×1 1 ×0 1 ×1 0 0
0 ×0 1 ×1 1 ×0 1 0
0 ×1 0 ×0 1 ×1 1 1
0 0 1 1 0
0 1 1 0 0
Image
4
Convolved Feature
Convolution


1
Convolved feature
Pooled feature
Pooling

Pooling

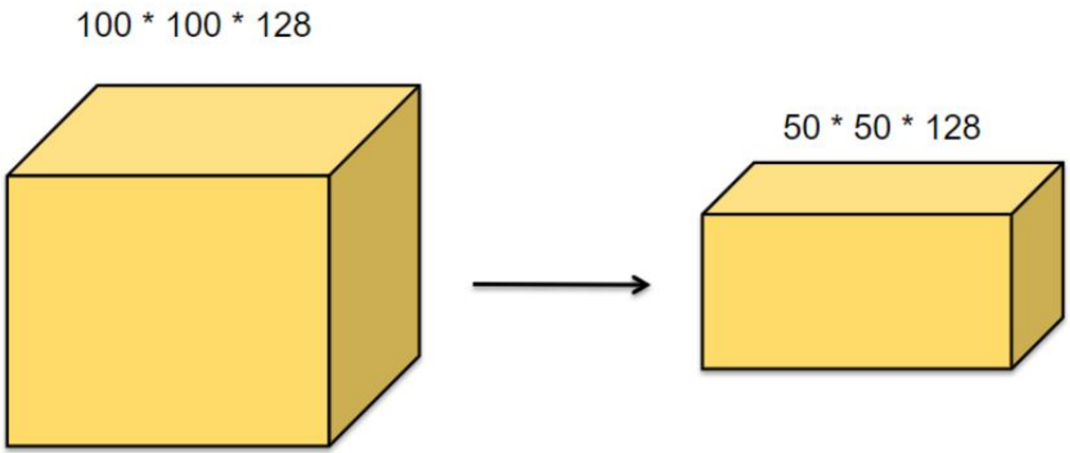
- The pooling layer operates upon each feature map separately to create a new set of the same number of pooled feature maps.

- its main function is to reduce the size of volume (number of pixels in the image) which makes the computation fast, reduces memory, and prevents from overfitting
- Still retains important information about the image

almost always 2×2 pixels applied with a stride of 2 pixels.

Pooling

It performs downsampling across the spatial dimensions (width, height). The representation would be smaller and more manageable.



Types of pooling

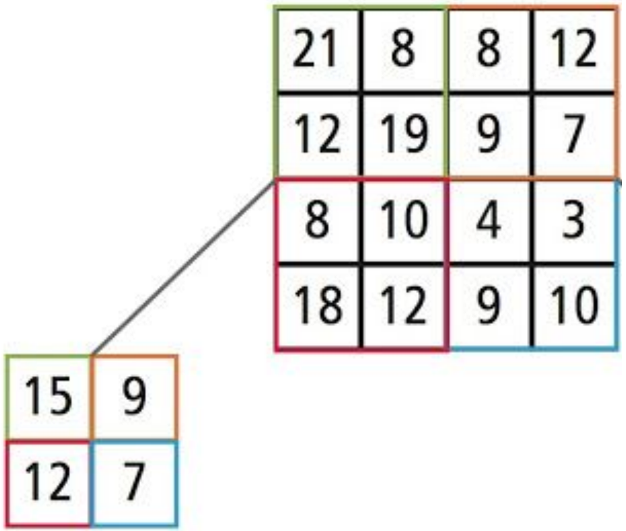
Average Pooling: Calculate the average value for each patch on the feature map.

Sum pooling: Calculate the sum for each patch on the feature map.

Min pooling: Calculate the minimum value for each patch of the feature map.

Max pooling: Calculate the maximum value for each patch of the feature map.

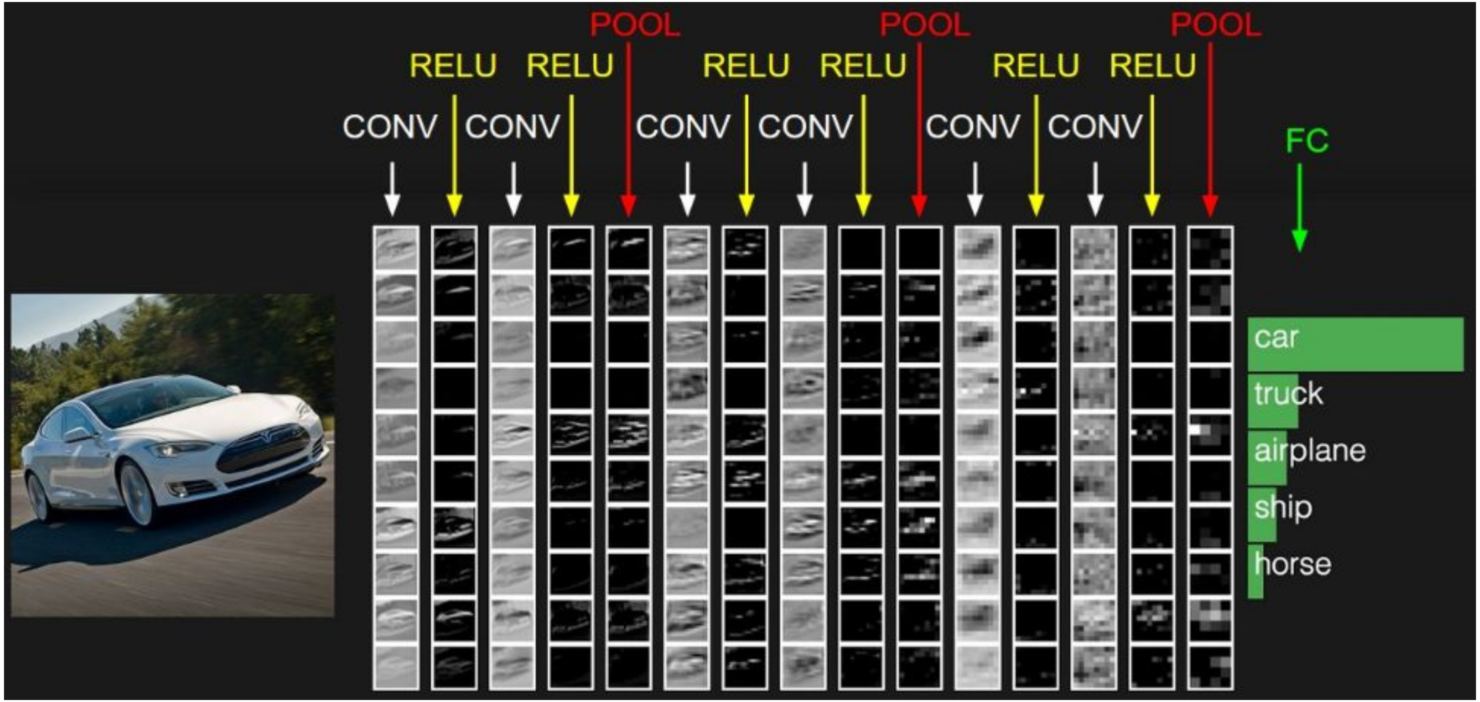
Average pooling



Average Pooling

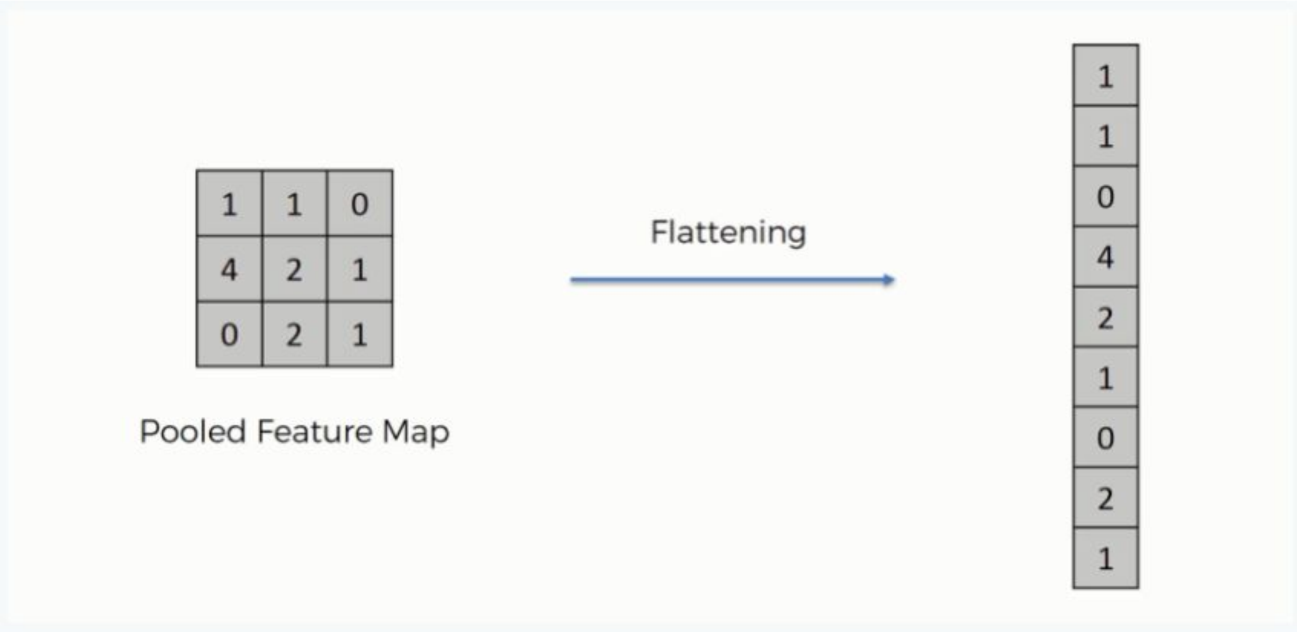
Max pooling

Filter Size: 2 * 2
Stride: 2


POOL
RELU
RELU
POOL
RELU
RELU
POOL
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

Fully Connected Layer

Flattening our pooled layer into a column so we can insert it into an artificial neural network

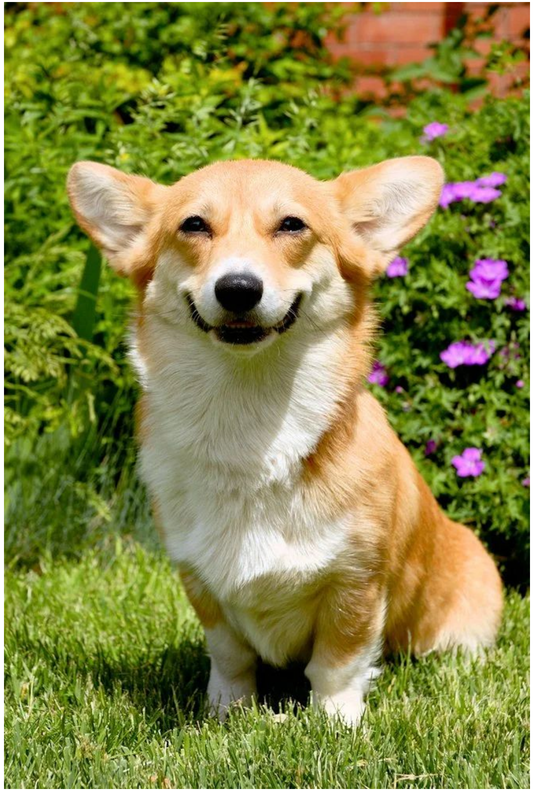


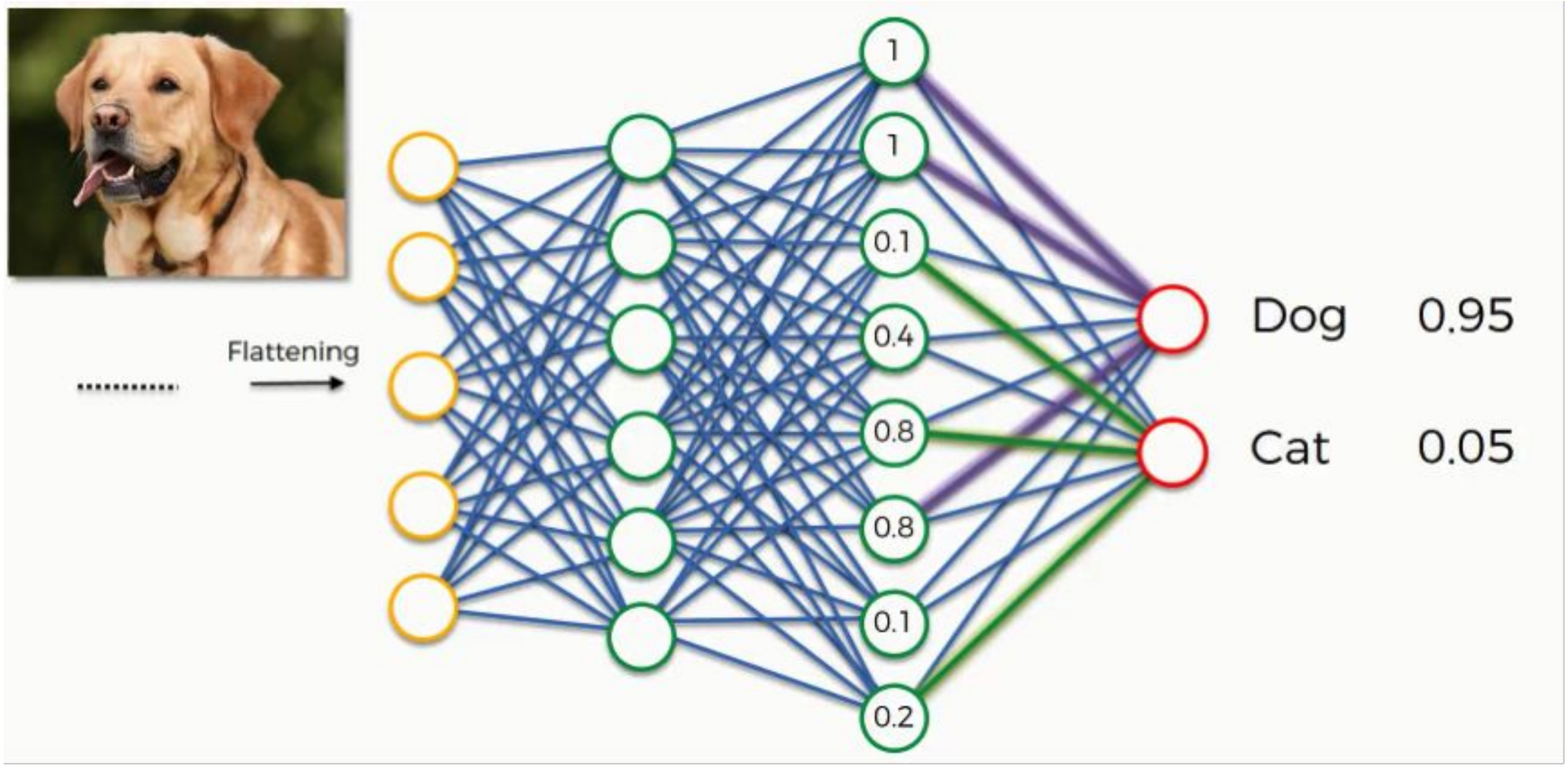
Fully Connected Layer

Pass the flattened layer through the ANN

The ANN outputs an N dimensional vector, where N is the number of classes we want to choose.

Example: Image detection of cats or dogs, N would equal 2. Each number in the N dimensional vector shows the probability of a class. If the resulting vector is [.05,.95], then this shows a 5% chance the image is a cat and 95% chance the image is a dog





Flattening
1
1
0.1
0.4
0.8
0.8
0.1
0.2
Dog
0.95
Cat
0.05

CNN Architecture: Review

- **Input:** In our scenario, it holds the raw pixel values of an image (e.g., an image of width 32, height 32, and with three color channels R,G,B).

- **Convolutional Layer:** This layer filters (convolve) the inputs to provide very useful information appropriate for object modeling. These convolutional layers help to automatically extract the most valuable information for the task at hand without human designed feature selection. This layer will result in data volume such as [32 * 32 * 16] if we used for example 16 filters.

- **ReLU Layer:** will apply a pixelwise activation function, such as the $max(0,x)$ thresholding at zero. This layer keeps the size of the data volume unchanged (e.g., [32 * 32 * 16]).

- **Pooling Layer**: It does a downsampling operation across the spatial dimensions (width, height), and will result in data volume such as [16 * 16 * 16].

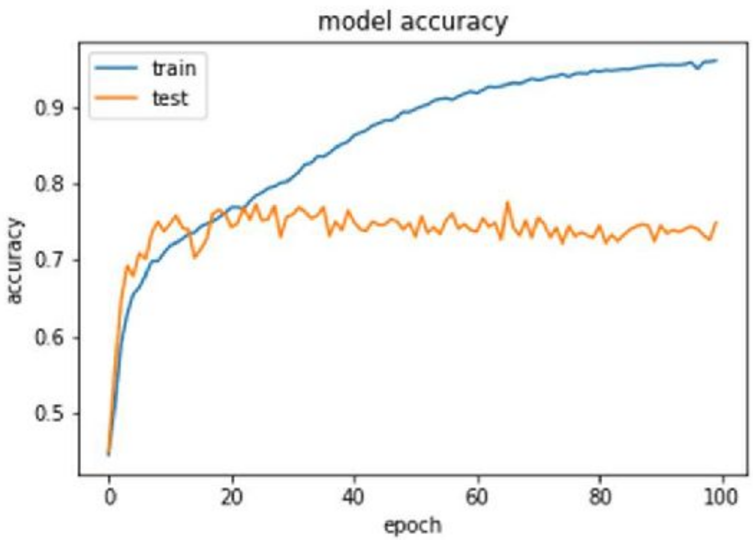
- **Fully Connected Layer:** This layer computes the class scores, and it will result in volume of size [1 * 1 * 3], where each of those 3 numbers correspond to a class score, such as among the 3 categories (doors, stairs, signs).

Improving the Model

Overfitting

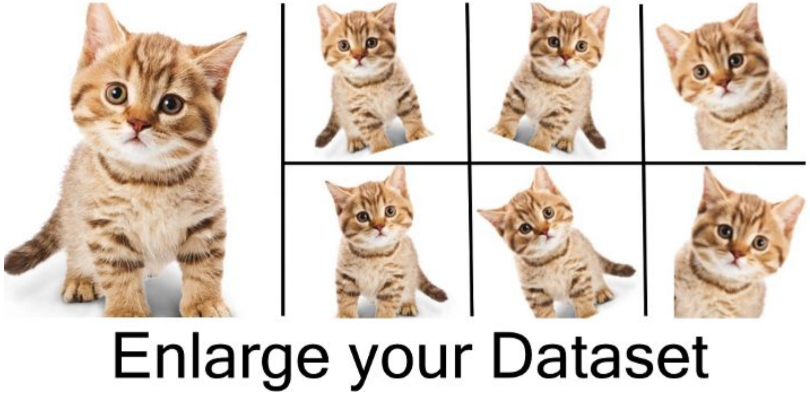
- Model fits too well to the training set
- Can't generalize to new examples in testing

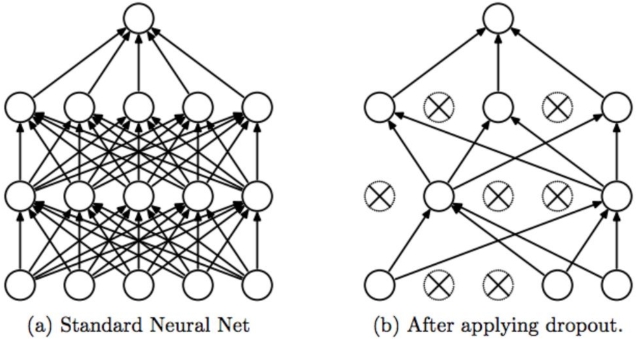
Ex: Recognizing specific images in training set instead of learning general patterns



How to reduce overfitting

- Add more data, data augmentation, reduce architecture complexity, add dropout





Let's check out a CNN implementation!

Flowers Recognition CNN