Convolutional Neural Networks

Slides by Prof. Ben Leong

Let’s take a step back….

How do we solve AI problems?

1. Formulate the problem
   - Data/state representation
   - Goal state

Data has structure

Spatial vs Temporal

2. Apply some known algorithm to well-defined problem

Credit:Internet meme original source unknown

Chihuahua or muffin?

Computer sees: $\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{2,500} \end{bmatrix} = \begin{bmatrix} \text{colour of pixel 0} \\ \text{colour of pixel 1} \\ \vdots \\ \text{colour of pixel 2,500} \end{bmatrix}$

You see 

50 pixels

50 pixels

Question of the Day:

Suppose we restrict terms to max degree 2, how many features do we have?

“Traditional” approach

Domain Knowledge → Feature Definition → Classification based on features detected

Difficult!

Scaling & Rotations

Deformation

Occlusion

Illumination /shadows

“Automated” Feature Extraction

Naïve approach

Using pixel values directly as an input vector:

- Lose information on spatial structure
- Huge number of weights for the input layer

Exploit Spatial Structure

Images as 2D matrices

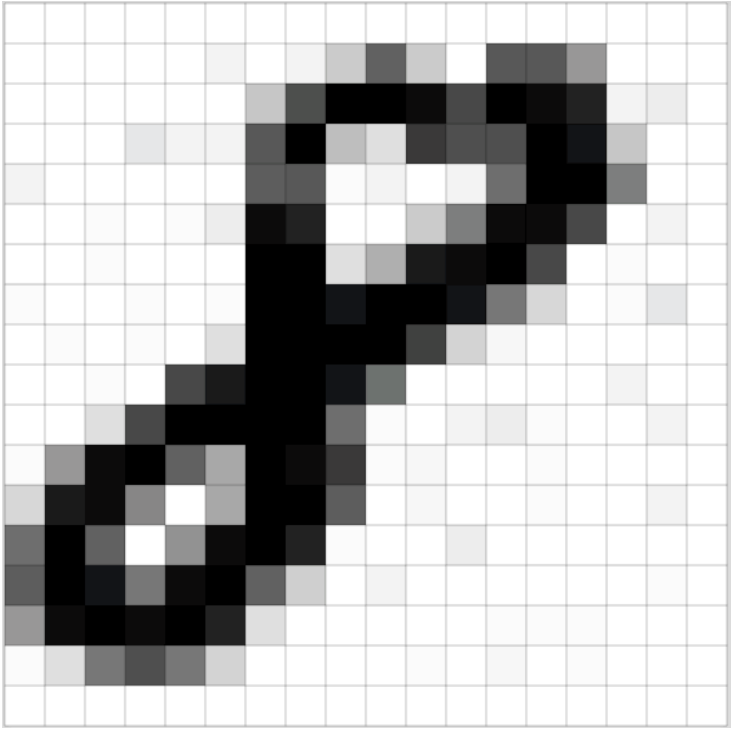
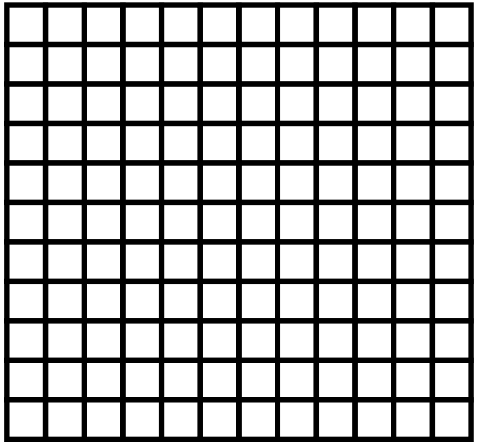


Image credit: https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721

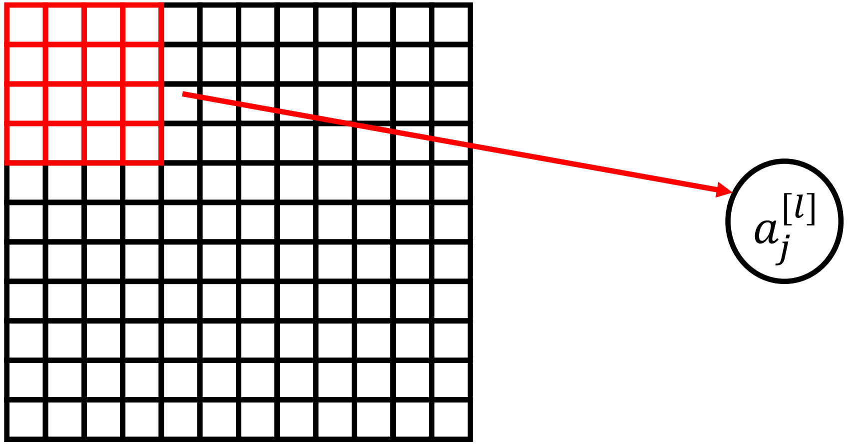
Convolution

Key Idea: Consider not one pixel at a time, but a group of pixels



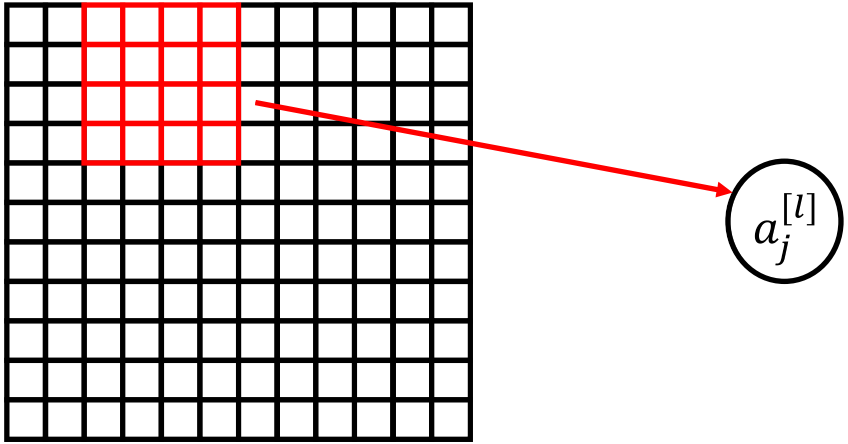
Convolution

Key Idea: Consider not one pixel at a time, but a group of pixels



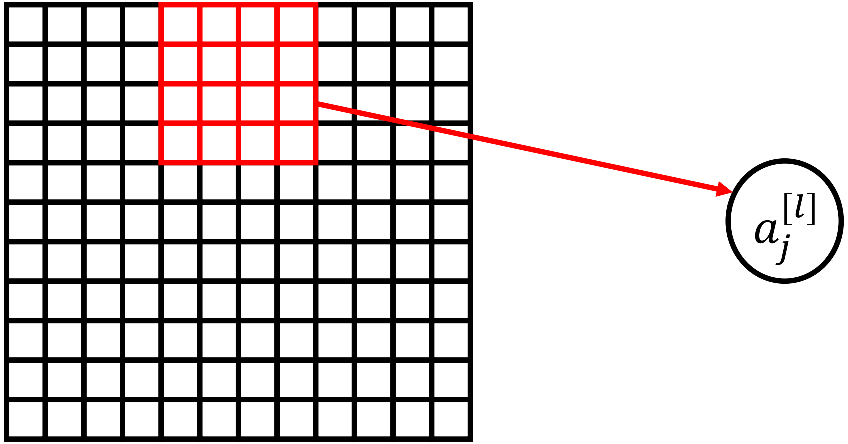
Convolution

Key Idea: Consider not one pixel at a time, but a group of pixels



Convolution

Key Idea: Consider not one pixel at a time, but a group of pixels



How does convolution work?

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

image

*

0	-1	0
-1	5	-1
0	-1	0

kernel

=

3		

feature map

Multiply and sum

How does convolution work?

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

image

*

0	-1	0
-1	5	-1
0	-1	0

kernel

=

3	3	

feature map

Multiply and sum

How does convolution work?

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

image

*

0	-1	0
-1	5	-1
0	-1	0

kernel

=

3	3	-2

feature map

Multiply and sum

How does convolution work?

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

image

*

0	-1	0
-1	5	-1
0	-1	0

kernel

=

3	3	-2
-3		

feature map

Multiply and sum

How does convolution work?

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

image

*

0	-1	0
-1	5	-1
0	-1	0

kernel

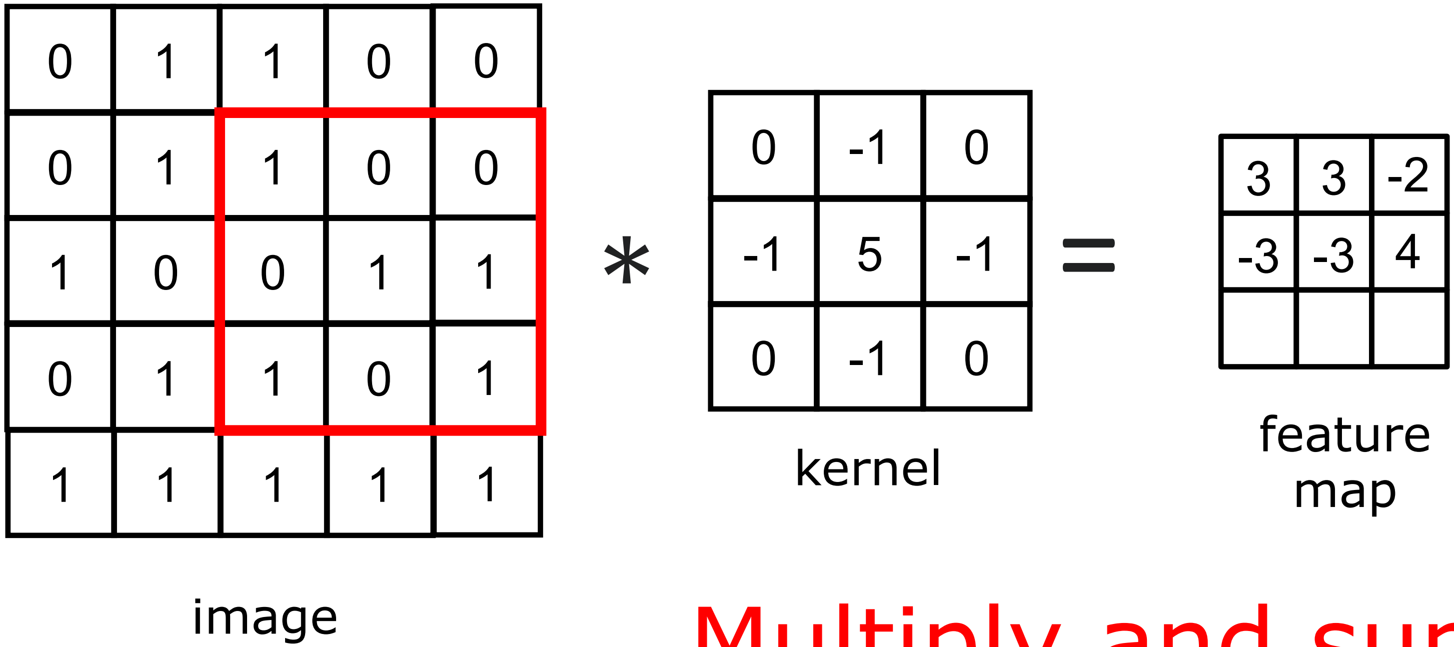
=

3	3	-2
-3	-3	

feature map

Multiply and sum

How does convolution work?



Multiply and sum

How does convolution work?

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

image

*

0	-1	0
-1	5	-1
0	-1	0

kernel

=

3	3	-2
-3	-3	4
3		

feature map

Multiply and sum

How does convolution work?

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

image

*

0	-1	0
-1	5	-1
0	-1	0

kernel

=

3	3	-2
-3	-3	4
3	3	

feature map

Multiply and sum

How does convolution work?

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

image

$*$

0	-1	0
-1	5	-1
0	-1	0

kernel

$=$

3	3	-2
-3	-3	4
3	3	-4

feature
map

Multiply and sum

How does convolution work?

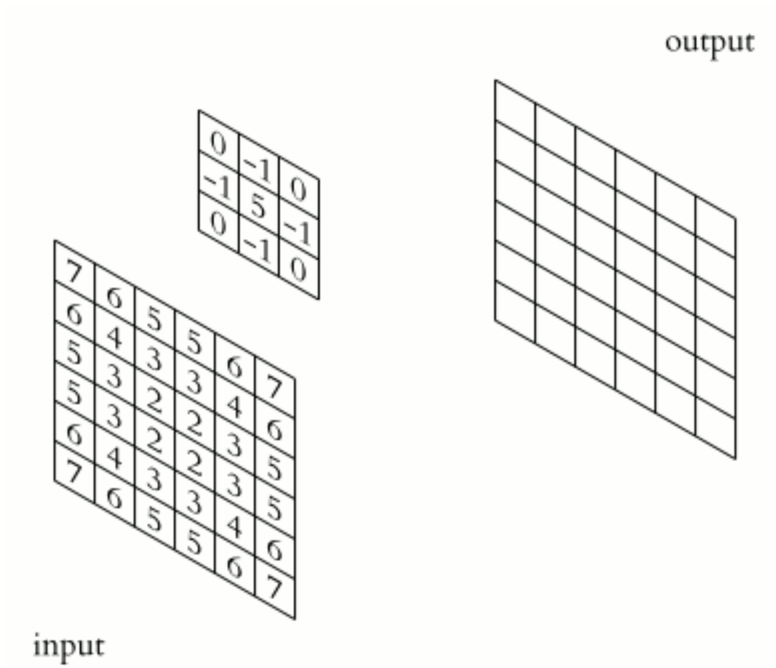


Image credit: https://towardsdatascience.com/types-of-convolution-kernels-simplified-f040cb307c37

Clarification: Convolution

$$(f * g)(t) = \int_{-\infty}^{\infty} f(x)g(t-x)dx$$

Discrete form (2D):

$$x[m,n] * h[m,n] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x[i,j] * h[m-i, n-j]$$

How does convolution work?

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

image

$*$

1	2	3
4	5	6
7	8	9

kernel

$=$

feature map

Flip first

How does convolution work?

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

image

9	8	7
6	5	4
3	2	1

kernel

=

feature map

Flip first

Multiply and sum

How does convolution work?

$$x[m,n] * h[m,n] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x[i,j] * h[m+i, n+j]$$

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

image

1	2	3
4	5	6
7	8	9

kernel

=

feature map

If we don’t flip first

$\Rightarrow$2D Correlation

Convolution Filters

original

sharpen

0	-1	0
-1	5	-1
0	-1	0

Laplacian

0	1	0
1	-4	1
0	1	0

emboss

-2	-1	0
-1	1	1
0	1	2

Convolution Filters

original

outline

-1	-1	-1
-1	8	-1
-1	-1	-1

top sobel

1	2	1
0	0	0
-1	-2	-1

left sobel

1	0	-1
2	0	-2
1	0	-1

Different filters can extract different features!

Padding & Stride

- We lose pixels at the edge of our image $\Rightarrow$padding
- Computing adjacent filters might be slow $\Rightarrow$stride

Padding & Stride

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

image

*

0	-1	0
-1	5	-1
0	-1	0

kernel

=

3	3	-2
-3	-3	4
3	3	-4

feature map

Padding & Stride

$$
\underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\text{image}} * \underbrace{\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}}_{\text{kernel}} = \underbrace{\begin{bmatrix} -1 & 3 & 3 & -1 & 0 \\ -2 & 3 & 3 & -2 & -1 \\ 5 & -3 & -3 & 4 & 3 \\ -3 & 3 & 3 & -4 & 3 \\ 5 & 2 & 2 & 3 & 3 \end{bmatrix}}_{\text{feature map}}
$$

\+ padding p_h and p_w

Padding & Stride

$$\underset{\text{image}}{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}} * \underset{\text{kernel}}{\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}} = \underset{\text{feature map}}{\begin{bmatrix} -1 & \boxed{3} & 3 & -1 & 0 \\ -2 & 3 & 3 & -2 & -1 \\ 5 & -3 & -3 & 4 & 3 \\ -3 & 3 & 3 & -4 & 3 \\ 5 & 2 & 2 & 3 & 3 \end{bmatrix}}$$

+ padding p_h and p_w

Padding & Stride

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

image

$*$

0	-1	0
-1	5	-1
0	-1	0

kernel

$=$

-1	3	3	-1	0
-2	3	3	-2	-1
5	-3	-3	4	3
-3	3	3	-4	3
5	2	2	3	3

feature map

+ padding p_h and p_w

Padding & Stride

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

image

*

0	-1	0
-1	5	-1
0	-1	0

kernel

=

-1	3	3	-1	0
-2	3	3	-2	-1
5	-3	-3	4	3
-3	3	3	-4	3
5	2	2	3	3

feature map

+ padding p_h and p_w

Padding & Stride

2 steps!

$$
\underbrace{\begin{array}{|c|c|c|c|c|c|c|}
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
0 & 0 & 1 & 1 & 0 & 0 & 0 \\ \hline
0 & 0 & 1 & 1 & 0 & 0 & 0 \\ \hline
0 & 1 & 0 & 0 & 1 & 1 & 0 \\ \hline
0 & 0 & 1 & 1 & 0 & 1 & 0 \\ \hline
0 & 1 & 1 & 1 & 1 & 1 & 0 \\ \hline
0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
\end{array}}_{\text{image}}
*
\underbrace{\begin{array}{|c|c|c|}
\hline
0 & -1 & 0 \\ \hline
-1 & 5 & -1 \\ \hline
0 & -1 & 0 \\ \hline
\end{array}}_{\text{kernel}}
=
\underbrace{\begin{array}{|c|c|c|}
\hline
-1 & 3 & 0 \\ \hline
5 & -3 & 3 \\ \hline
5 & 2 & 3 \\ \hline
\end{array}}_{\text{feature map}}
$$

stride = 2

2D matrix

Linear Layer: $\boldsymbol{a}^{[l]} = g^{[l]}\left(\left(\boldsymbol{W}^{[l]}\right)^{\mathsf{T}}\boldsymbol{a}^{[l-1]}\right)$

could be same

Activation is a 3D Matrix! Concatenation of Feature Maps

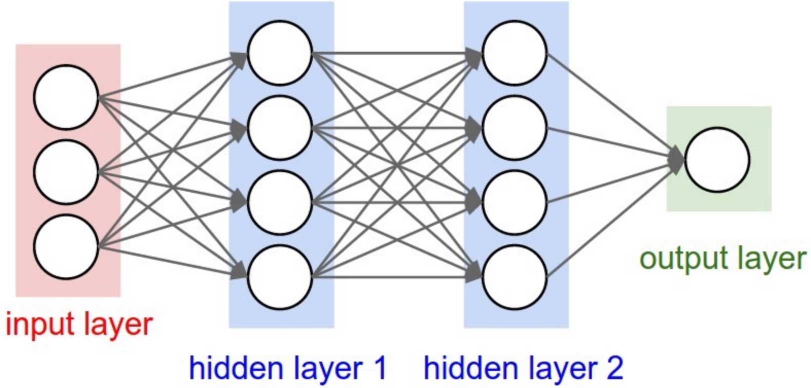
Convolution Layer: $\boldsymbol{A}^{[l]} = g^{[l]}\left(\boldsymbol{W}^{[l]} * \boldsymbol{A}^{[l-1]}\right)$

Weights = **Kernels** (3D matrix)

MLP vs CNN

Fully Connected Linear Layers

- Each layer has multiple **neurons**
- Neuron output: **0D scalar** activation
- Neuron input: **1D vector** of activations
    - Each *element* is a different neuron
- Each layer is a **1D vector**



Convolutional Layers

- Each layer has multiple **kernels**
- Kernel output: **2D matrix** feature map
- Kernel input: **3D matrix** of feature maps
    - Each *depth position* is a different kernel
    - Analogy: filters are “stacked” together
- Each layer is a **3D matrix**

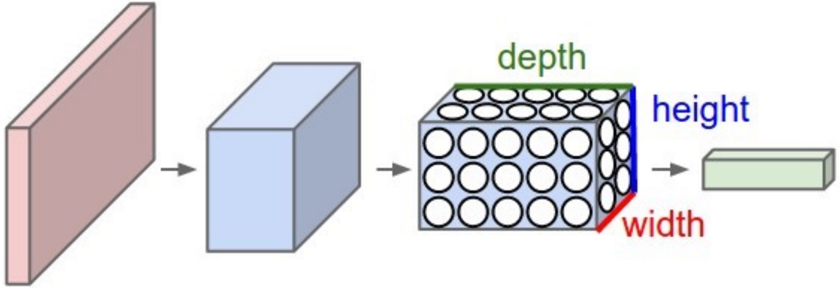
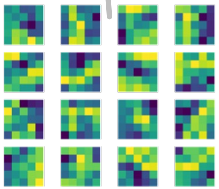


Image credit: https://cs231n.github.io/convolutional-networks/

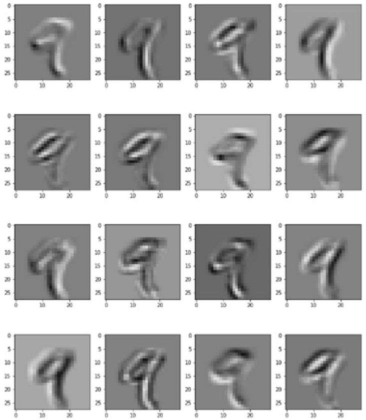
Convolution Layer

$$\boldsymbol{X}^{[0]} \to g^{[1]}\left(\boldsymbol{W}^{[1]} * \boldsymbol{X}^{[0]}\right) = \boldsymbol{A}^{[1]} \xrightarrow{\text{Pooling}} g^{[2]}\left(\boldsymbol{W}^{[2]} * \boldsymbol{X}^{[1]}\right) = \boldsymbol{A}^{[2]}$$

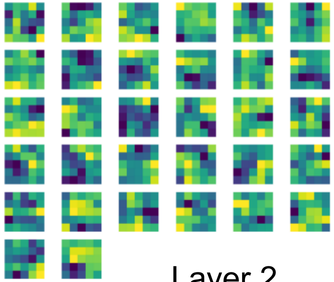
Input



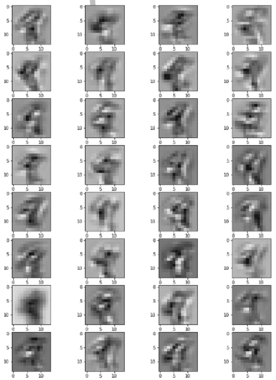
Layer 1 Feature Kernels



Layer 1
Feature Maps



Layer 2
Feature Kernels



Layer 2
Feature Maps

Hyperparameters

1. Number of kernels k
2. Kernel size $\boldsymbol{\kappa}$
3. Padding $\boldsymbol{p}$
4. Stride $\boldsymbol{s}$

Chosen manually, or automatically with [hyperparameter tuning]

Kernels are learned *automatically* through **weight updates**.

Interpretability: do you know what these kernel mean?

Image credit: https://medium.com/dataseries/visualizing-the-feature-maps-and-filters-by-convolutional-neural-networks-e1462340518e

Image credit: https://cs231n.github.io/convolutional-networks/

Feature Detectors: Intuition of Neuron Kernels in Layers

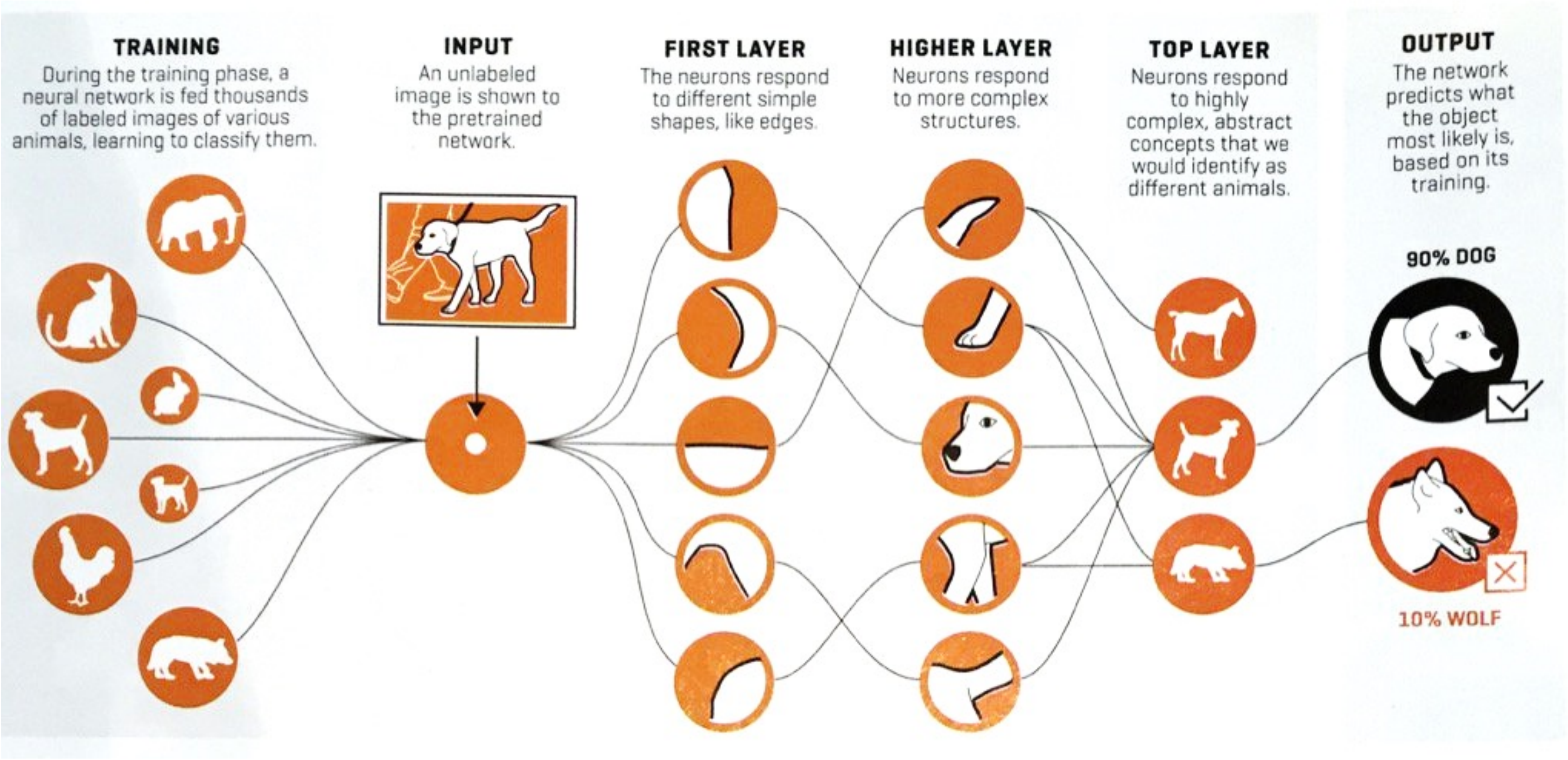


Image credit: https://fortune.com/longform/ai-artificial-intelligence-deep-machine-learning/

Analogy: activations of different filters learned by CNNs is like seeing the image through different lens filters

Image credit: https://www.amazon.com/Godefa-Samsung-Andriod-Smartphone-Universal/dp/B07RQRLQYH
https://www.yankodesign.com/2020/02/17/this-retro-inspired-camera-records-dreamy-looking-gifs-that-replicate-vintage-8mm-film/

Convolutional Neural Network

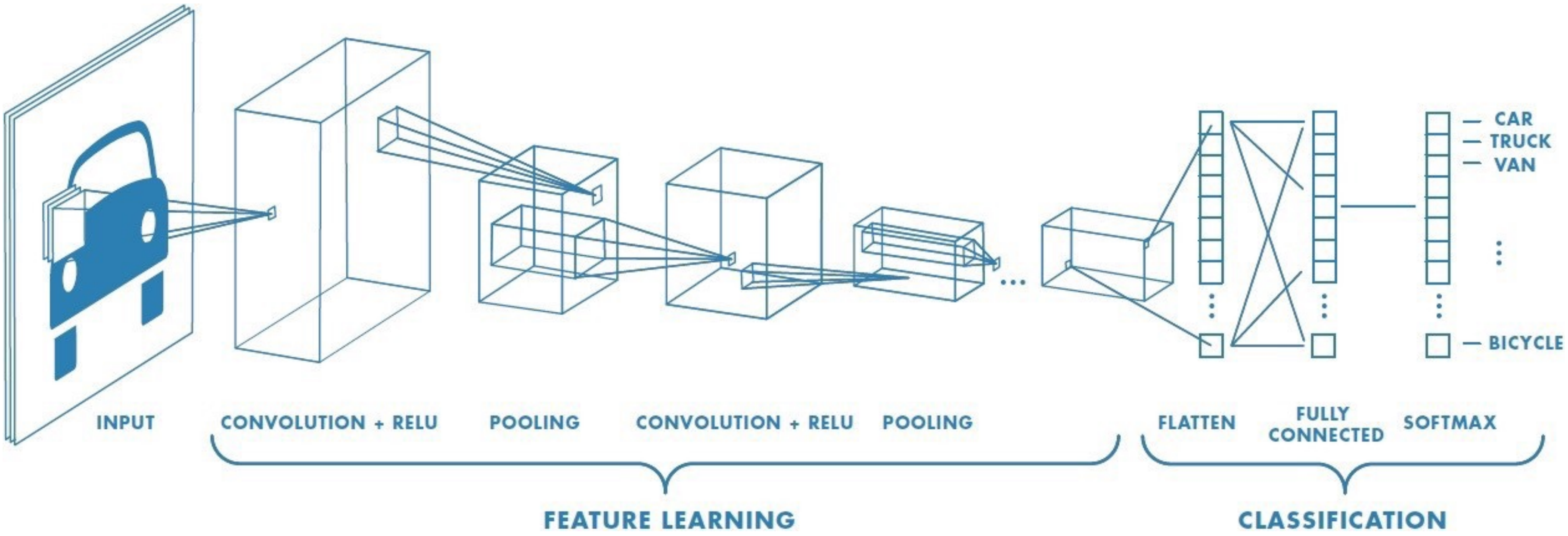


Image credit: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

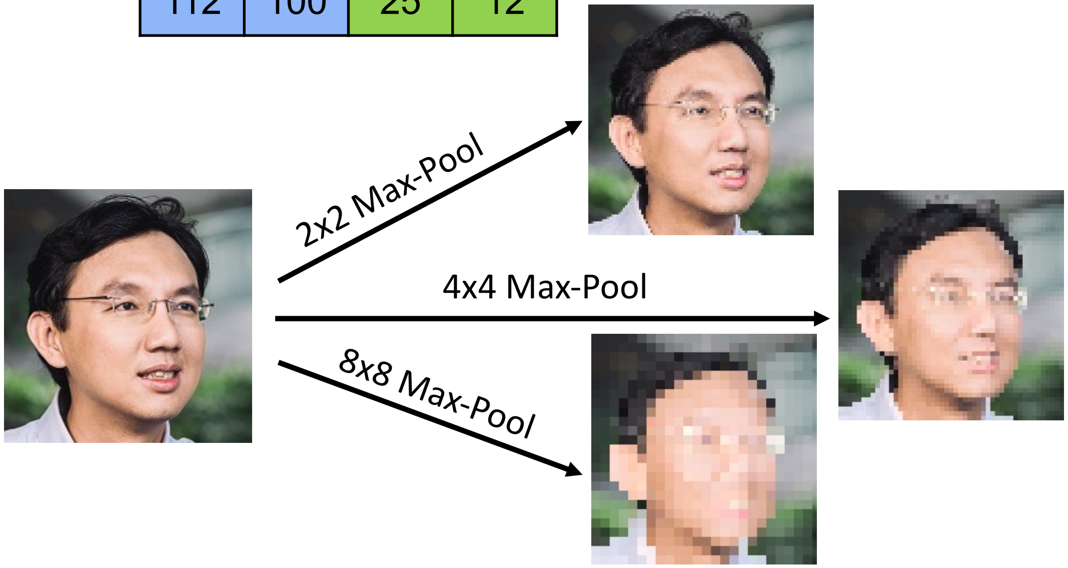
Pooling Layer

- **Downsamples** Feature Maps
- Helps to train later kernels to detect **higher-level** features
- Reduces **dimensionality**
- Aggregation methods
  - Max-Pool (most common)
  - Average-Pool
  - Sum-Pool

12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

2x2 Max-Pool

20	30
112	37



How do we decide on the final answer?

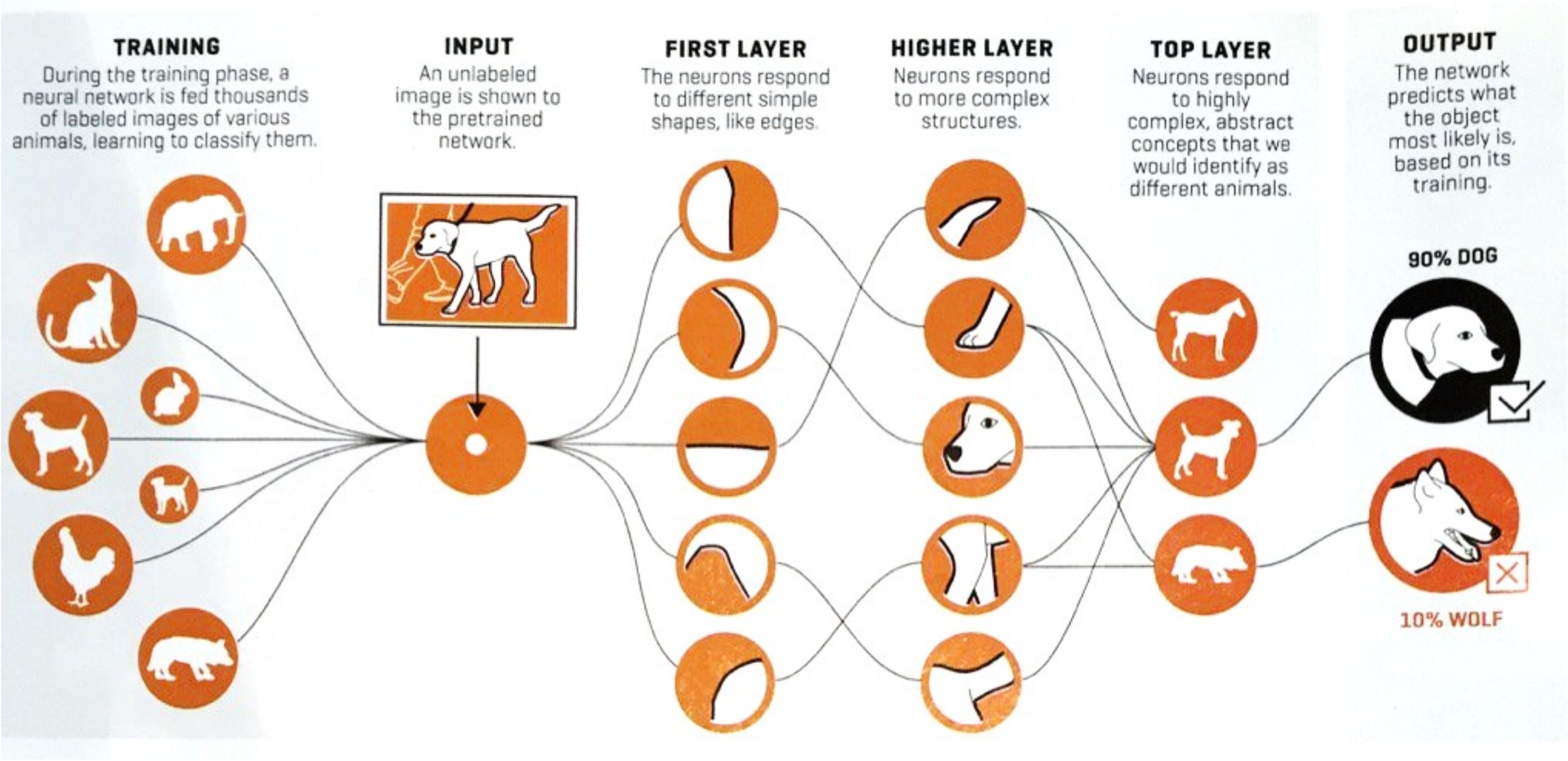


Image credit: https://fortune.com/longform/ai-artificial-intelligence-deep-machine-learning/

Why is convolution better?

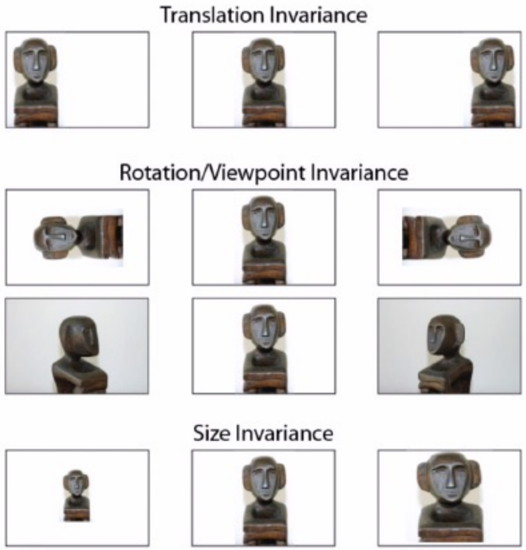
- Sparse Connections:
  - Each output is connected only to inputs within receptive field vs all inputs
  - Fewer parameters
  - Less overfitting

- Weight sharing vs unique weights
  - Regularization
  - Less overfitting

- Location or Spatial Invariance
  - Function transformation should not depend on location within the image
  - Make the same prediction no matter where the object is in the image

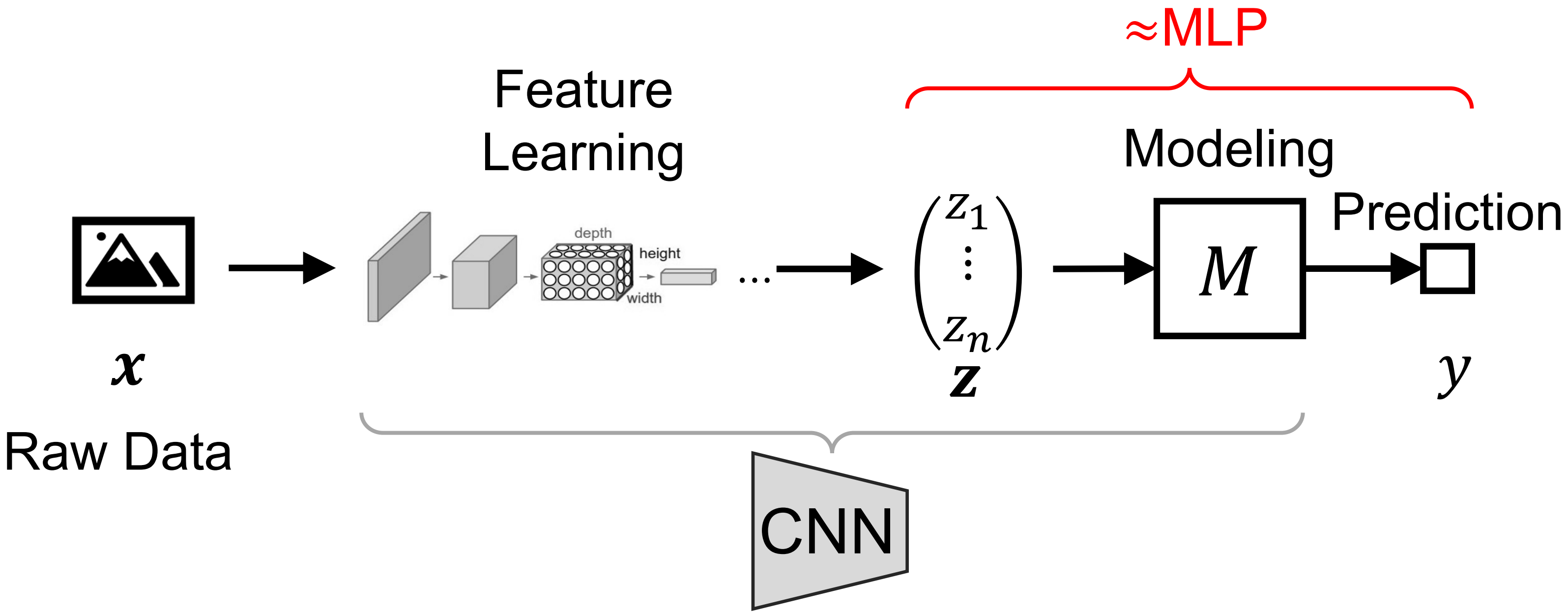
Location or Spatial Invariant

- You can recognize an object even if its appearance varies in some way
- Convolution operation commutes with respect to translation
    - If you convolve f with g, it doesn't matter if you translate the convolved output $f * g$, or you translate f or g first, then convolve them.
    - https://en.wikipedia.org/wiki/Convolution



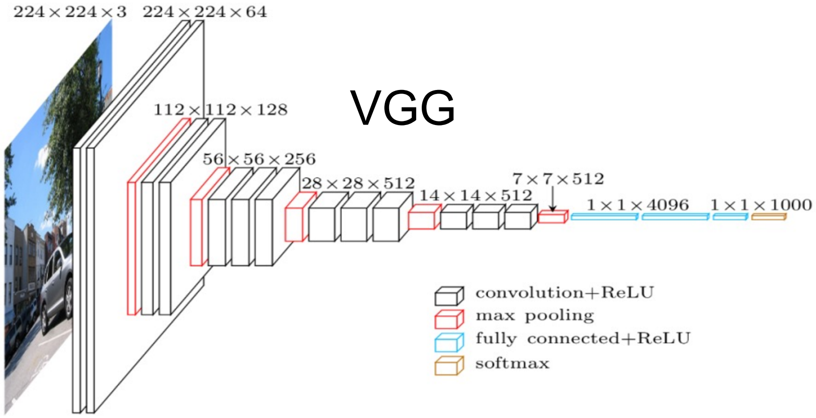
Slide credit: Matt Krause

Summary: CNN



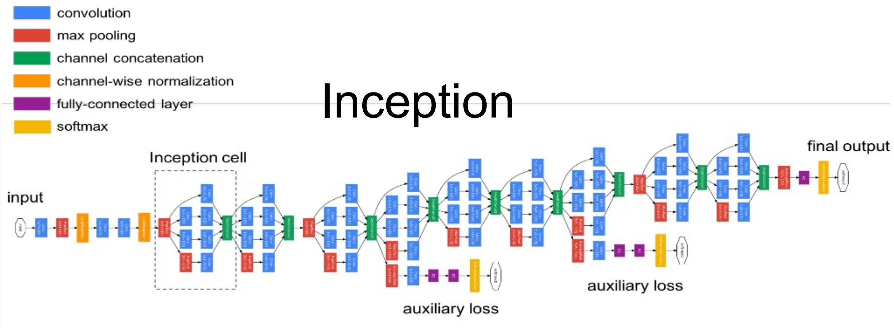
≈MLP
Feature Learning
depth
height
width
Modeling
Prediction
M
z_1
z_n
$\boldsymbol{x}$
$\boldsymbol{z}$
y
Raw Data
CNN

Other popular CNN architectures

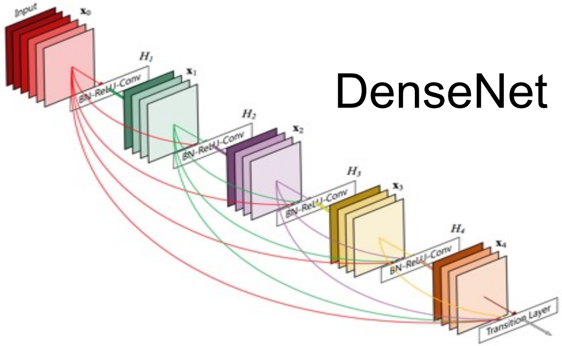


VGG

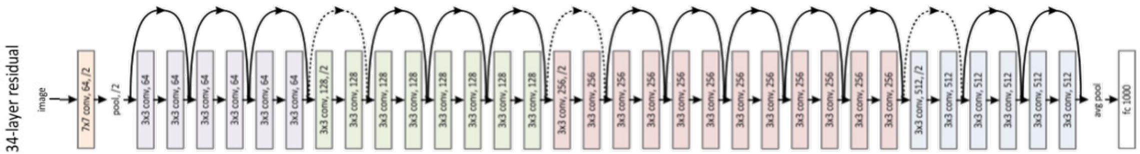
Inception



DenseNet



ResNet



Further reading: https://www.jeremyjordan.me/convnet-architectures/

Applications of CNN

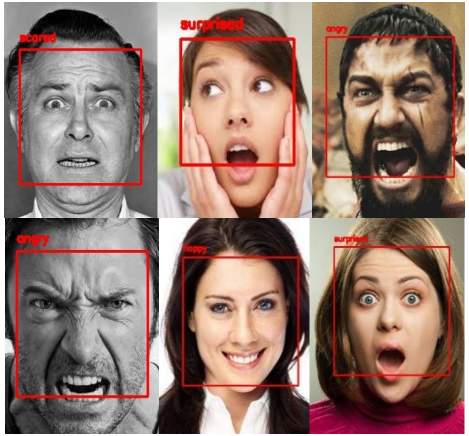
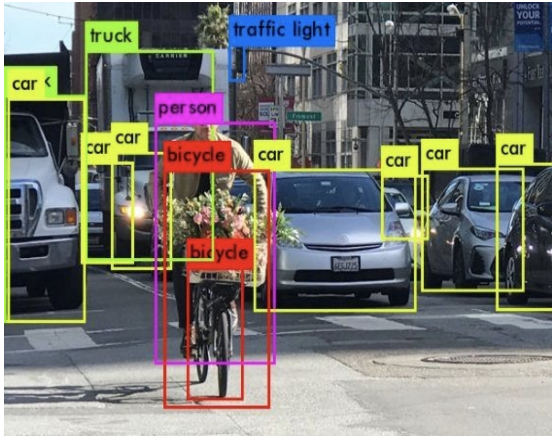


Image Classification
e.g., face emotions



Object Detection
e.g., self-driving cars

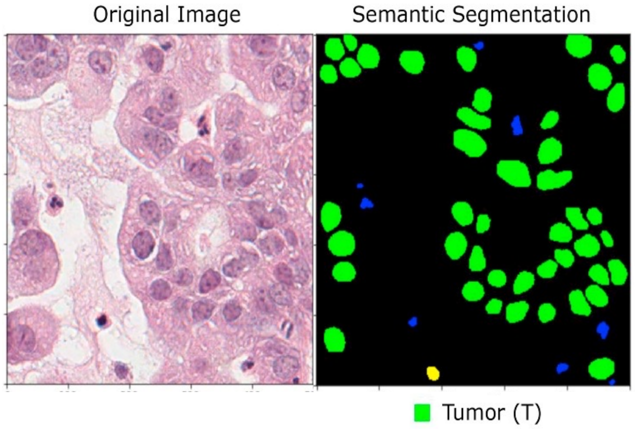


Image Segmentation
e.g., cancer cell detection

Image credit:
https://monica-dommaraju.medium.com/analysis-of-deep-learning-based-object-detection-f14d5138148
https://ajp.amjpathol.org/article/S0002-9440(18)31121-0/fulltext
https://appliedmachinelearning.blog/2018/11/28/demonstration-of-facial-emotion-recognition-on-real-time-video-using-cnn-python-keras/