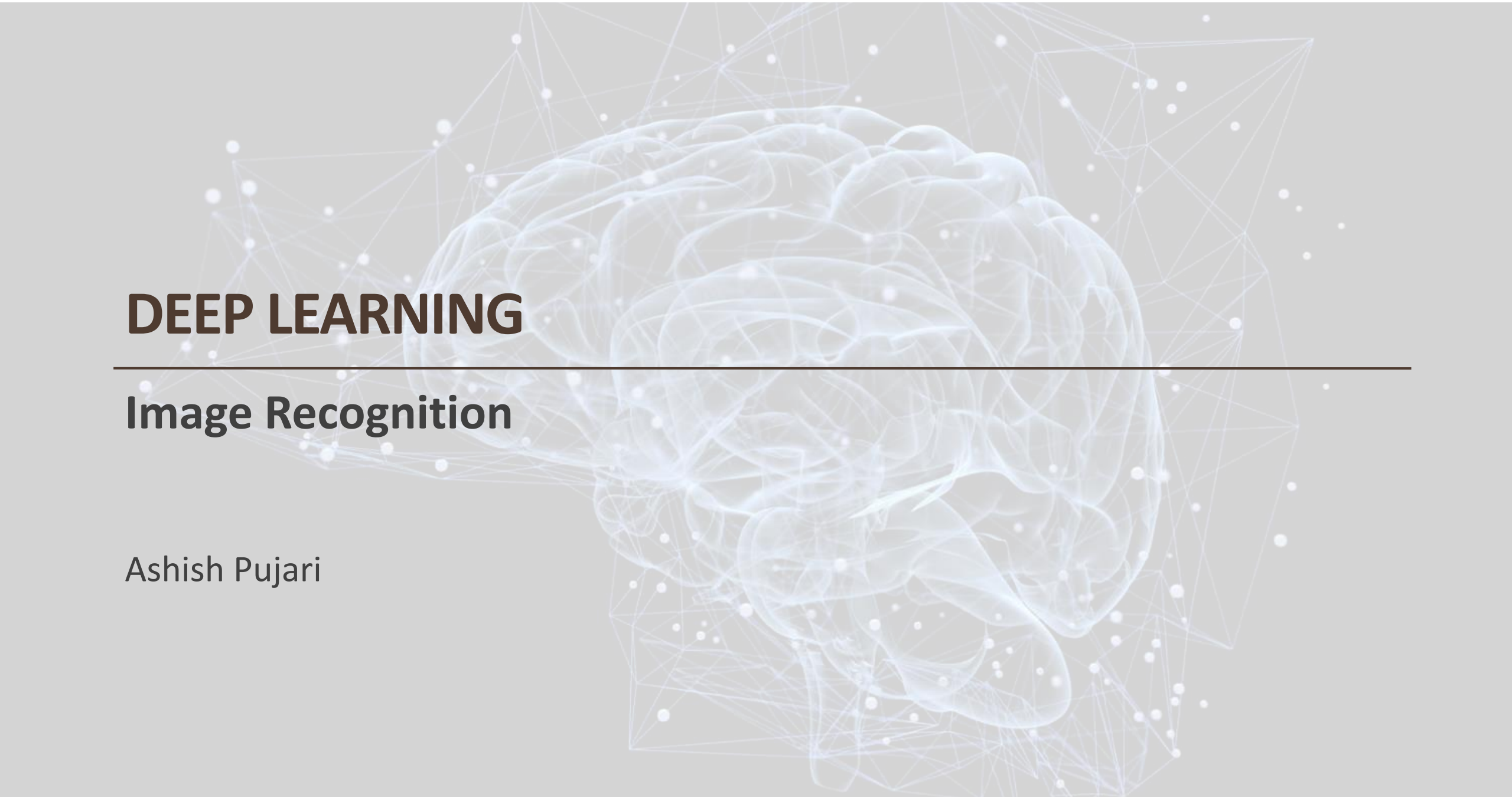


DEEP LEARNING

Image Recognition

Ashish Pujari

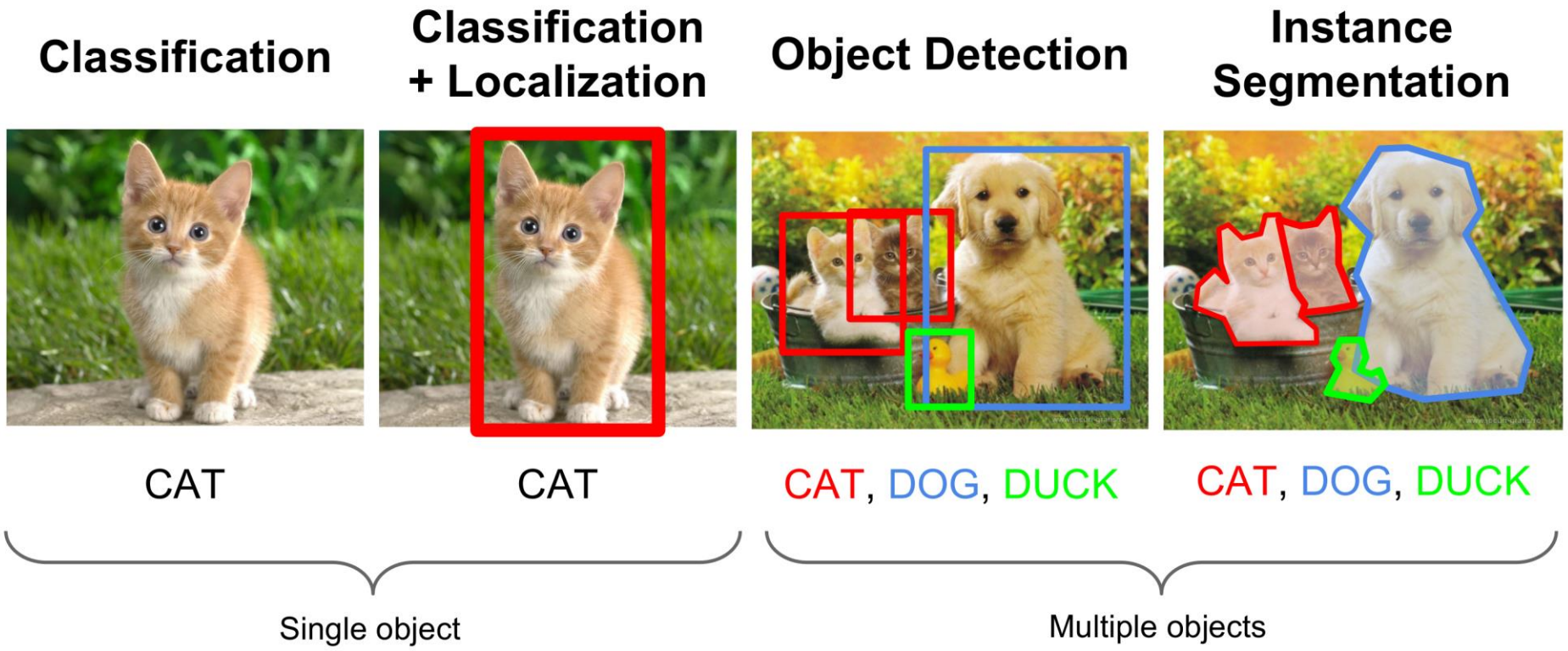
Image Recognition

1. Computer Vision
2. Convolutional Neural Networks
3. CNN Training
4. CNN Applications

COMPUTER VISION

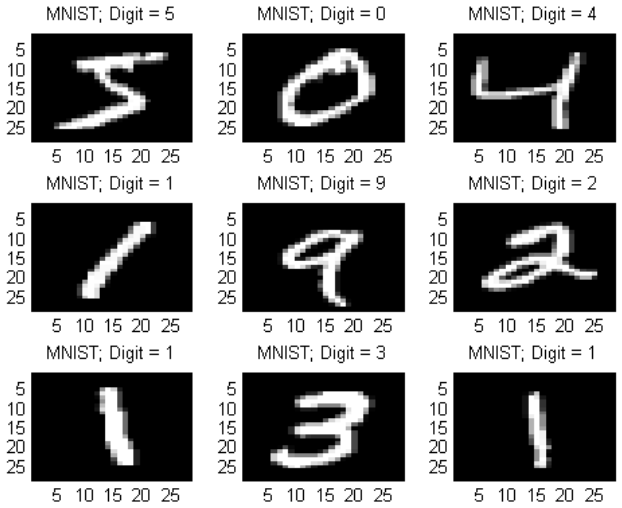
Image Datasets, Computer Vision Problems, Invariance

Computer Vision



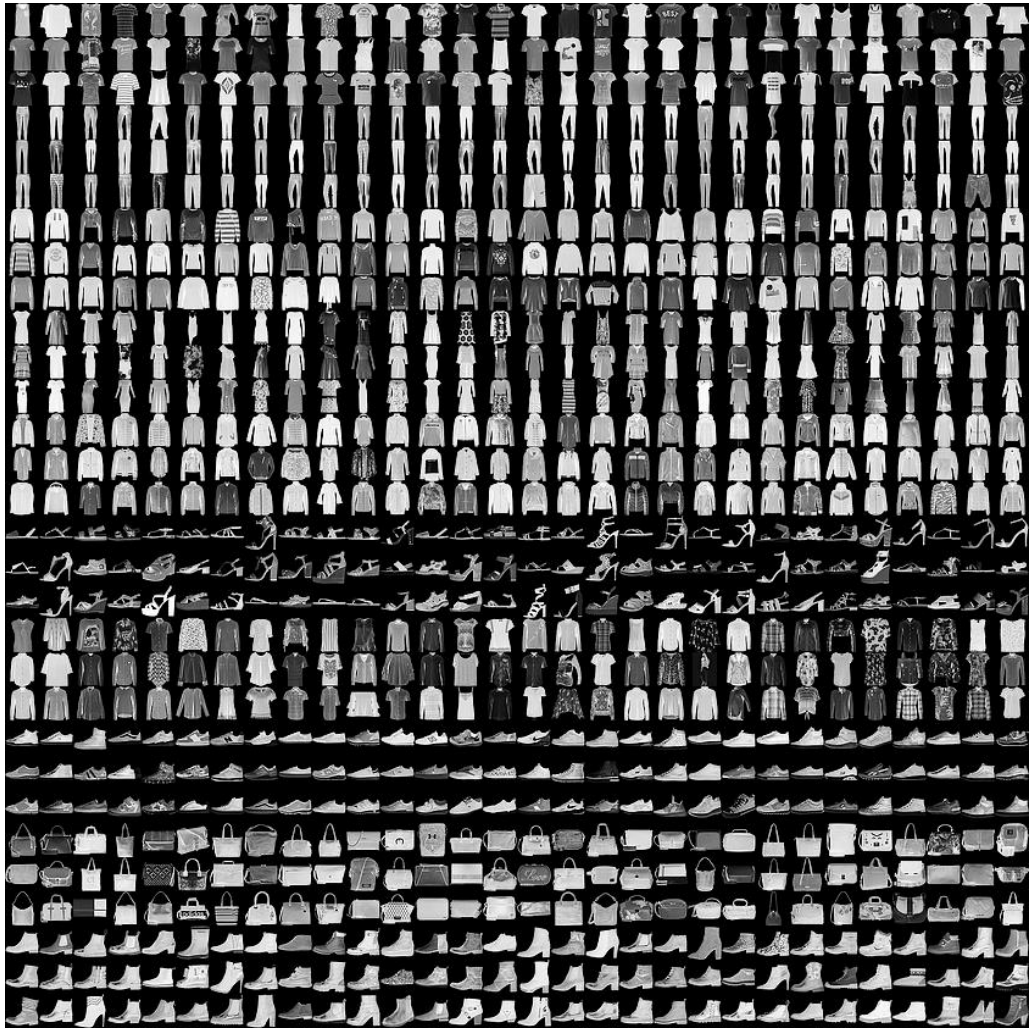
http://cs231n.stanford.edu/

Images - Benchmarking Datasets



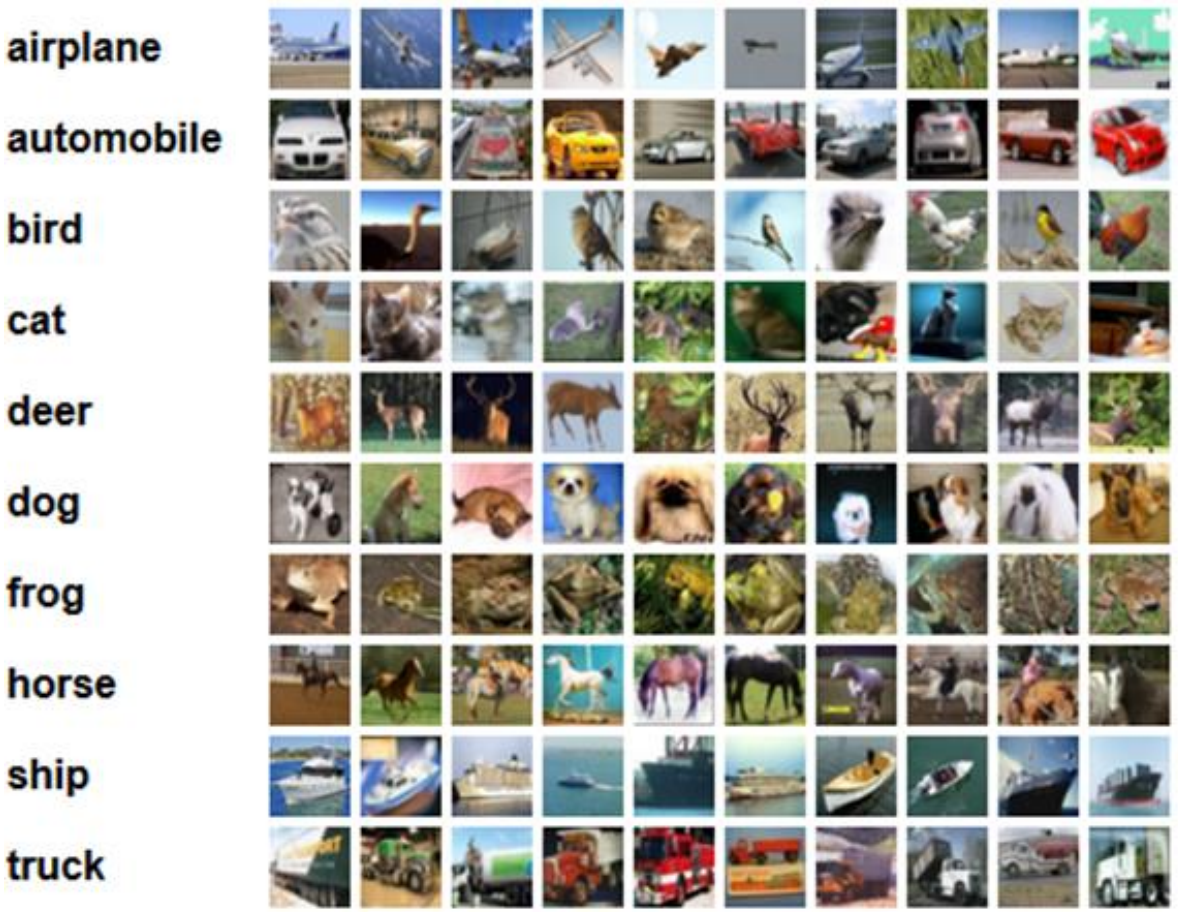
http://yann.lecun.com/exdb/mnist/

http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html

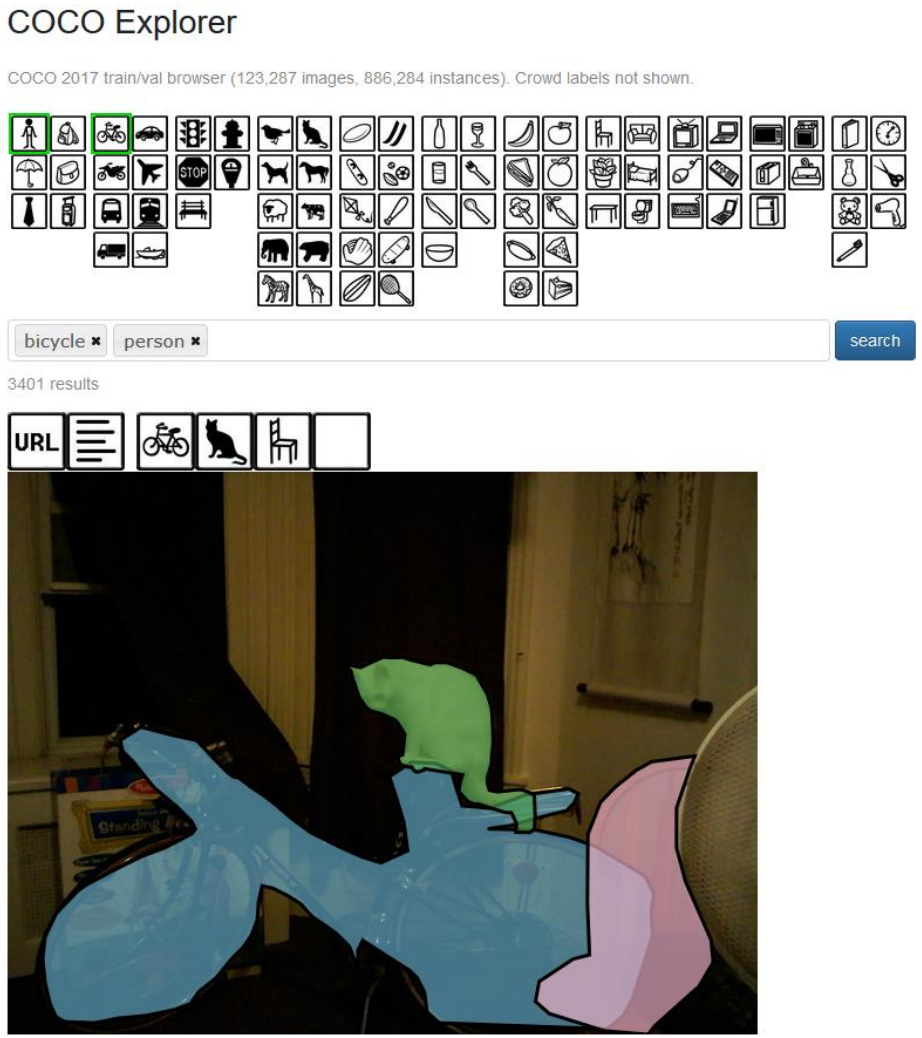


https://github.com/zalandoresearch/fashion-mnist

Images - Benchmarking Datasets

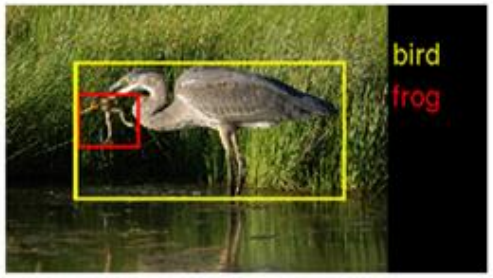


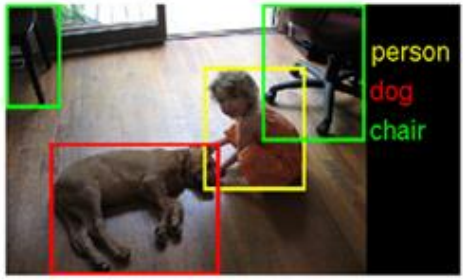
https://www.cs.toronto.edu/~kriz/cifar.html

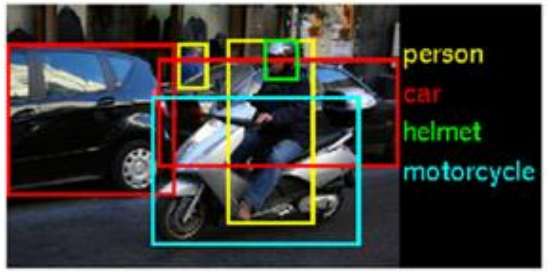


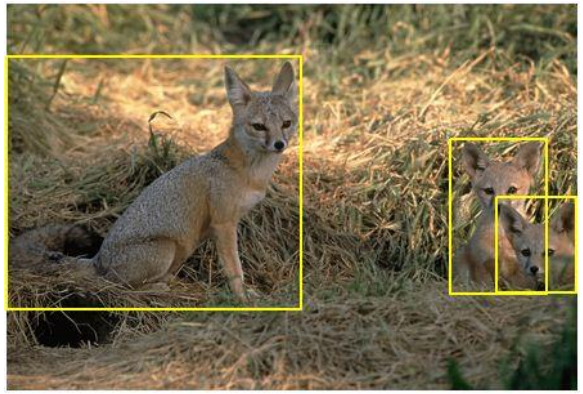
http://cocodataset.org/#explore

Images - Competition Datasets







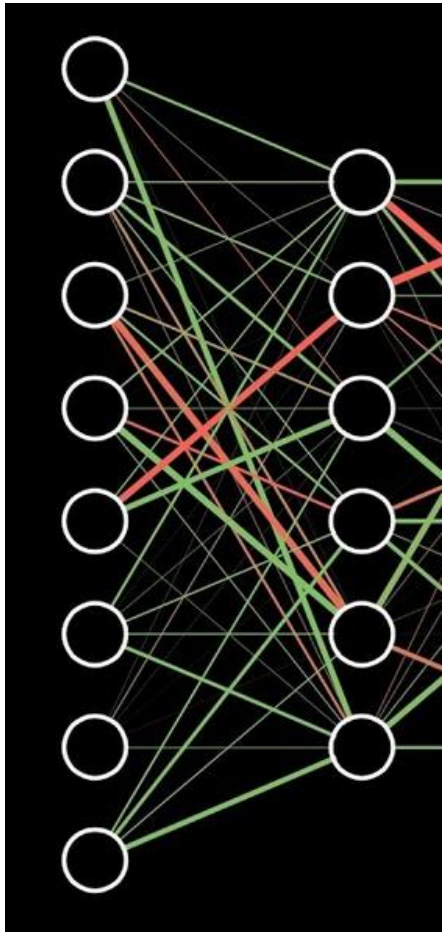


IMAGENET

http://image-net.org/challenges/LSVRC/

MLP Limitations

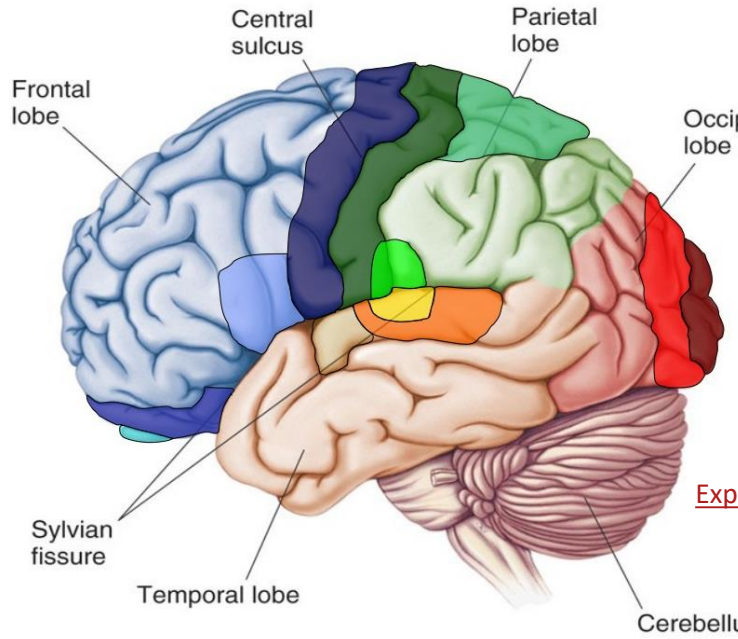
- High Dimensionality
  - 150px X 150px = 22500
  - Color Images: 22500 X 3 RGB = 67500
  - 3D images, video
- Computational Cost
  - MLPs suffer from high dimensionality due to full matrix multiplication between layers
- Spatial Information
  - MLPs are insensitive to order of elements in input vector. A images have intrinsic spatial information that is lost.



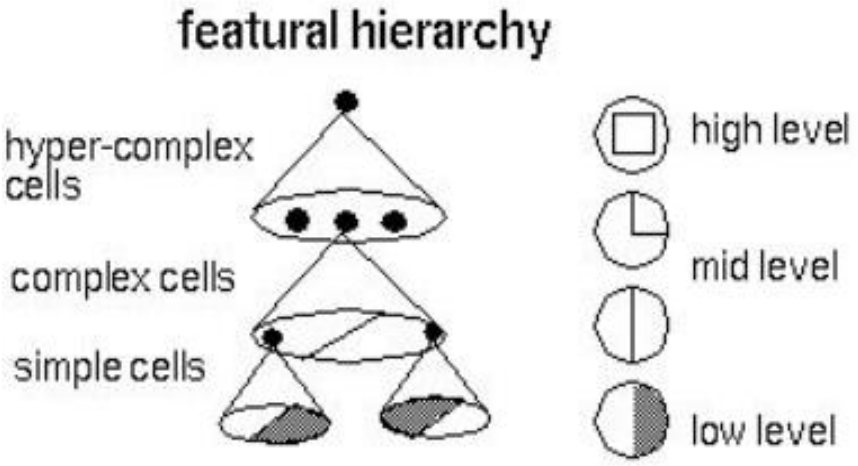
MLP – dense connections

Inspiration - Physiology of Vision

- Animal Visual Cortex
  - Contains complex arrangements of cells
  - Simple cells responsible for detecting edges
  - Complex cells have larger receptive fields with a degree of spatial invariance or contrast-insensitive manner
  - Higher level features respond to patterns of activation in lower level cells, and propagating activation upwards to still higher level cells
  - Experiments on cats showed that different cells responded to only certain kinds of stimuli such as an edge or a particular color

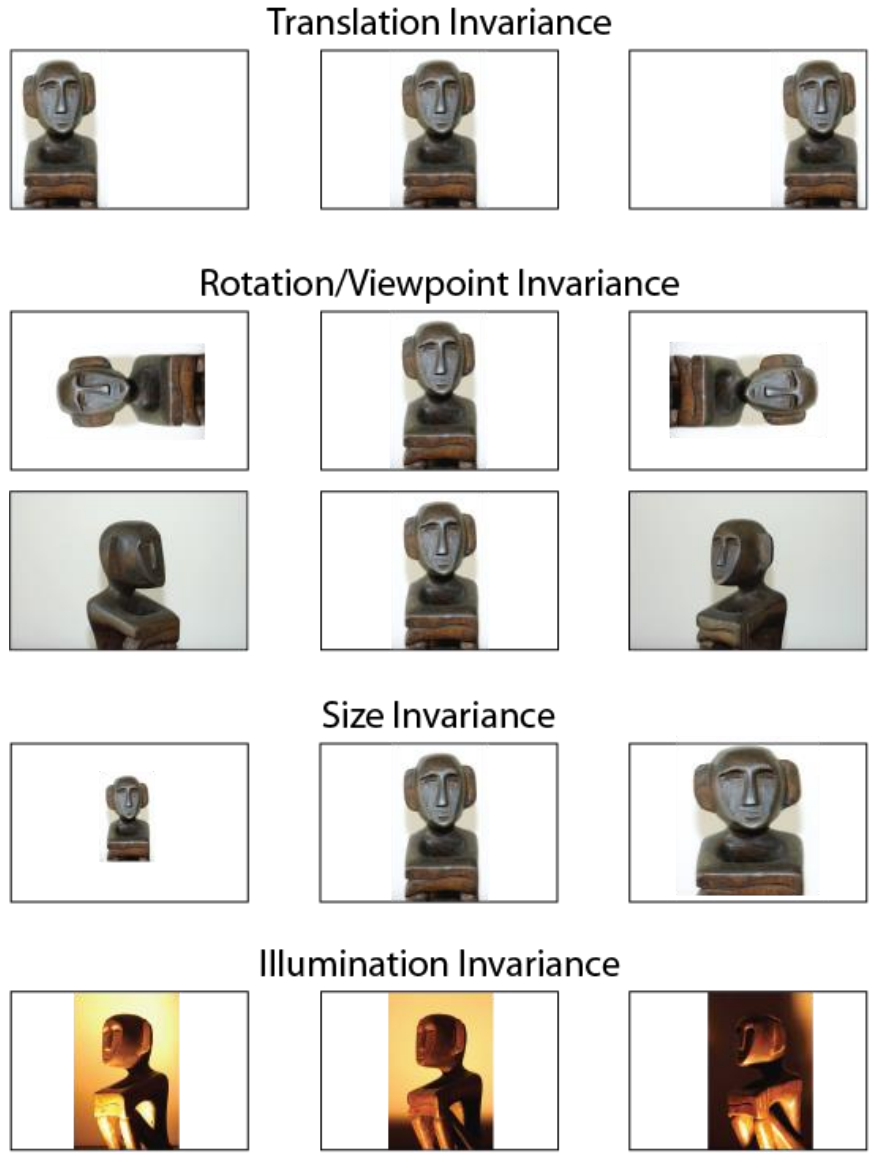


Explore the brain



Statistical Invariance

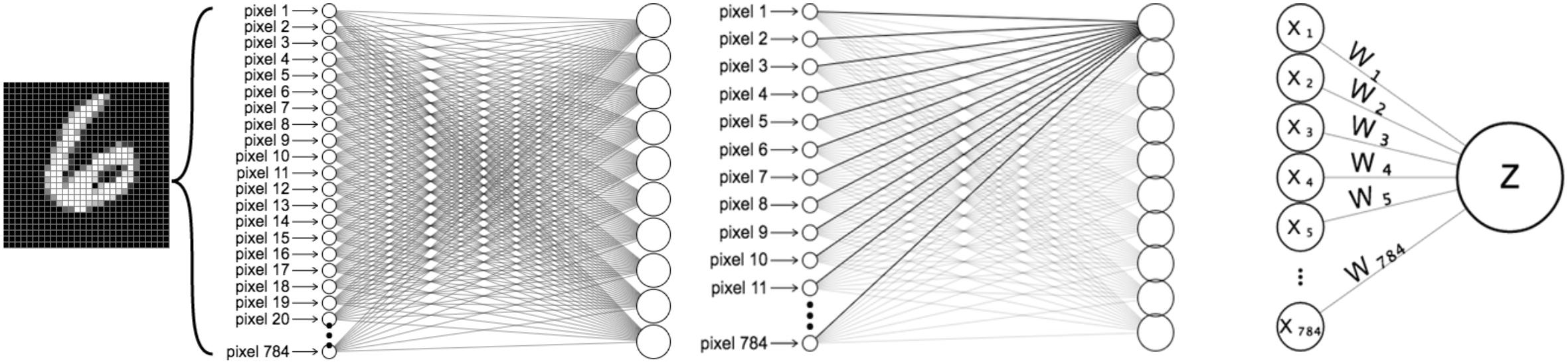
- Invariance
  - Translation/Spatial invariance
  - Orientation/Rotation invariance
  - Size/Scale invariance
  - Illumination invariance
- Challenge
  - Recognition in the presence of common transformations such as translation, rotation, etc.
- Solution
  - Since images have high statistical invariance, we could squeeze out the spatial information and map parameters that focus on content



CONVOLUTIONAL NEURAL NETS (CNN)

MLP Challenges

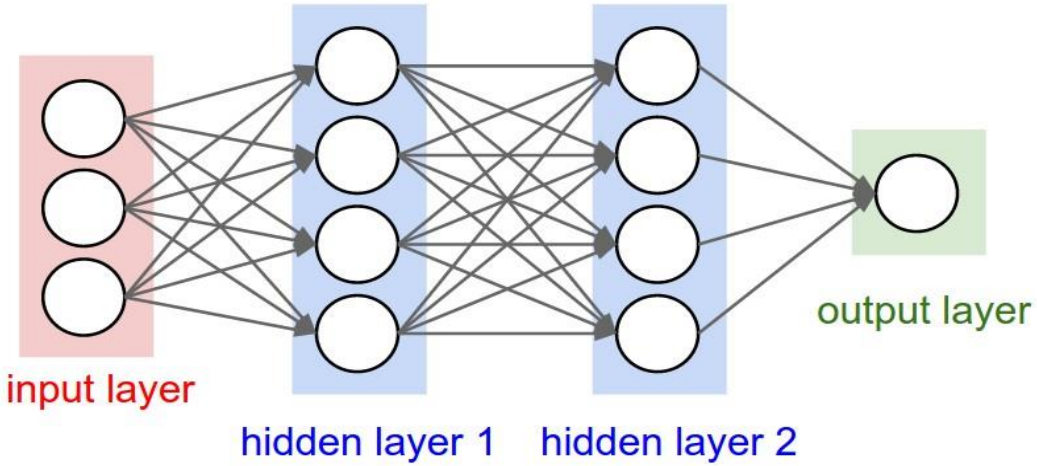
- When we input an image into an MLP, we need to unroll the pixels into a single column of neurons E.g. For a 32x32 MNIST image the number of input pixels will be 784 serially fed into an MLP through the input layer



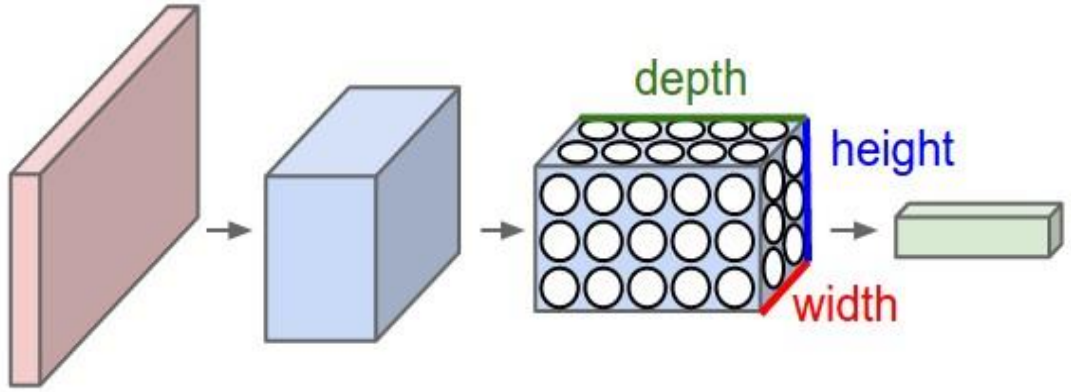
1-layer neural network for MNIST. The 10 output neurons correspond to our classes, the 10 digits from 0 to 9.

http://ml4a.github.io/ml4a/looking_inside_neural_nets/

MLP vs CNN



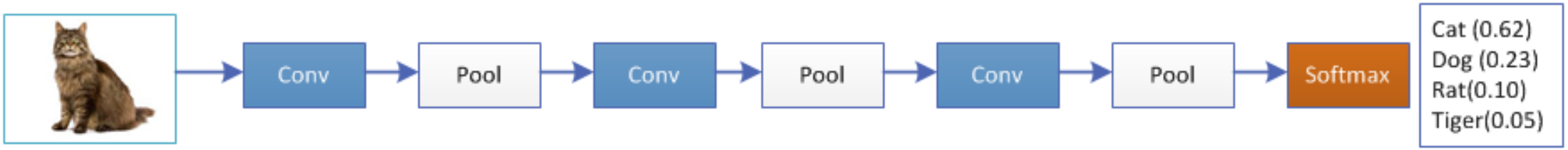
- MLPs suffer from curse of dimensionality due to full connectivity between nodes.
- MLPs do not scale well to higher resolution images



- CNN neurons are represented in 3D (width, height, depth)
- Transforms a 3D input volume to a 3D output volume of neuron activations

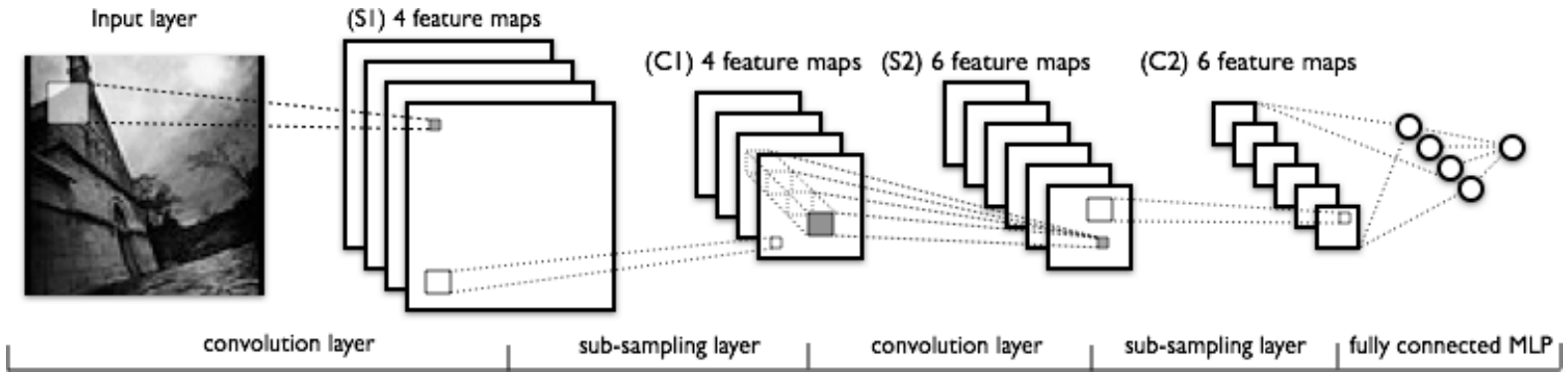
Convolutional Neural Networks (CNN or ConvNets)

- Deep neural networks inspired by a hierarchy of feature detectors in the visual cortex.
- Each layer progressively extracts higher and higher-level features of the image
- Leverage 3 important principles:
  - Sparse Connections
  - Parameter Sharing
  - Equivariant Representations

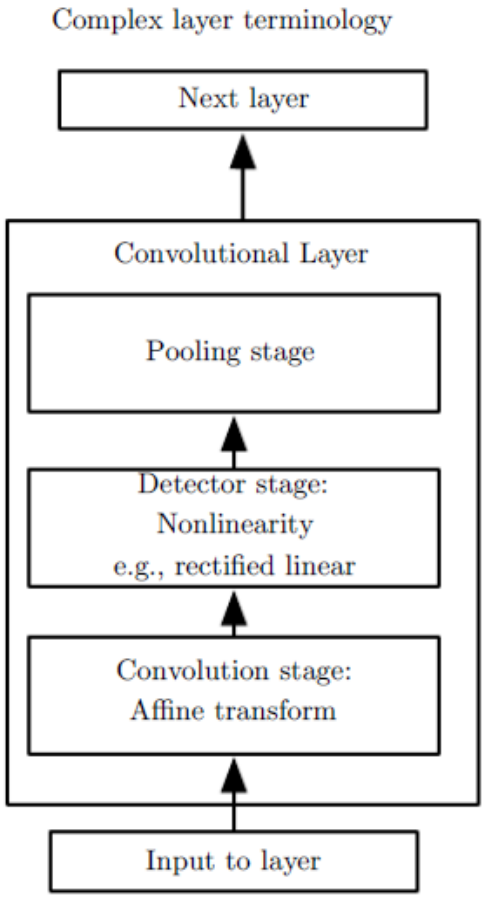


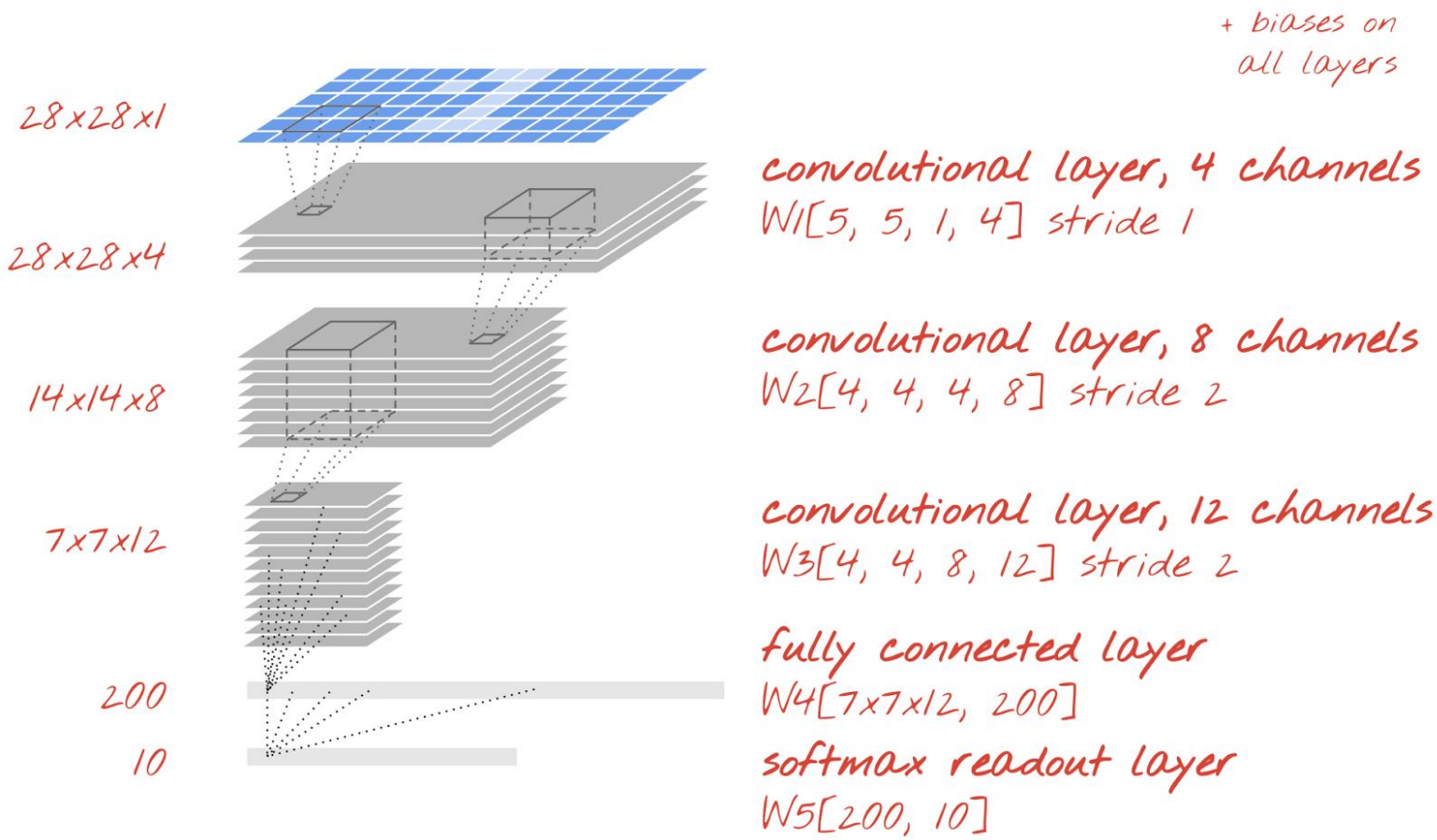
CNN Advantages

- Composability
  - The output of one convolutional layer becomes an input into another. With each layer, the network can detect higher-level, more abstract features.
- Reduced Dimensionality
  - Reduce dimensionality by orders of magnitude over ANNs by learning invariant features by using weight sharing
  - Smaller representations are achieved by shared parameters



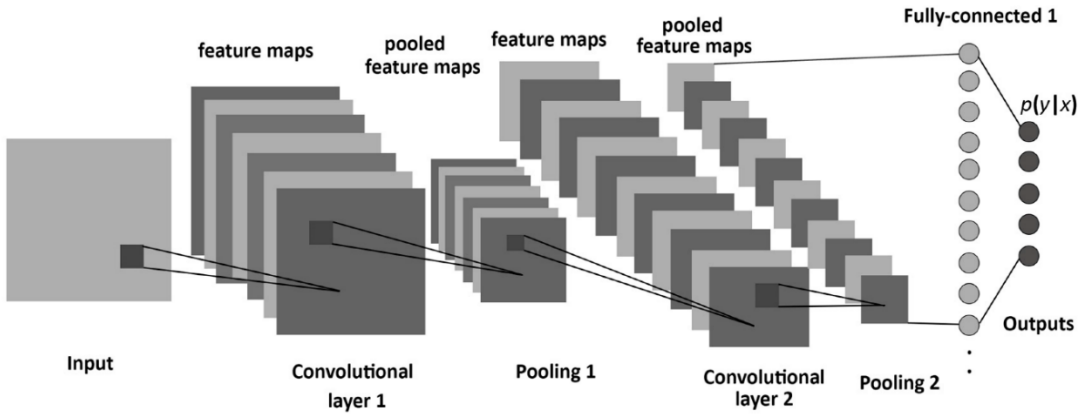
CNN Layers

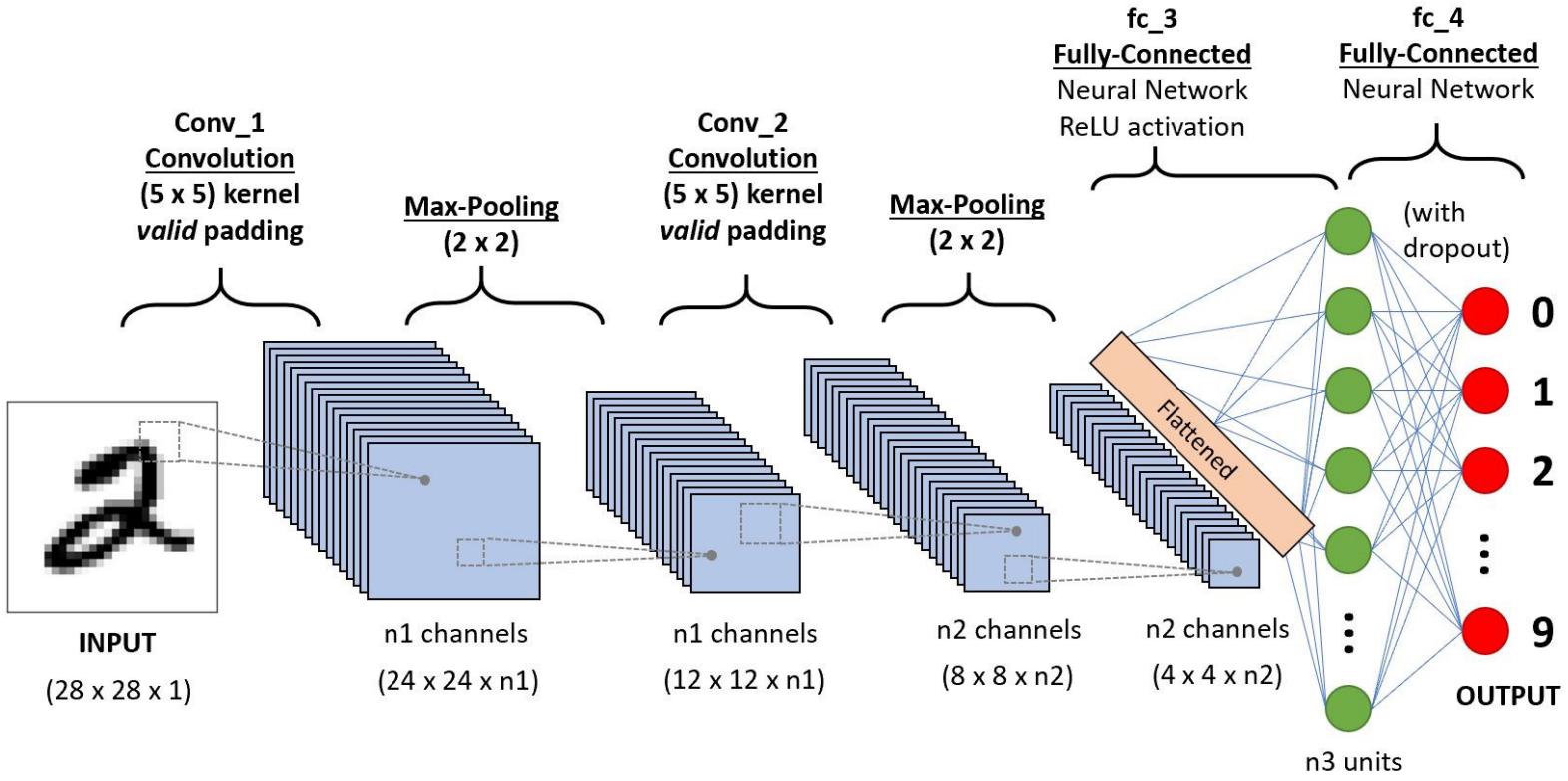




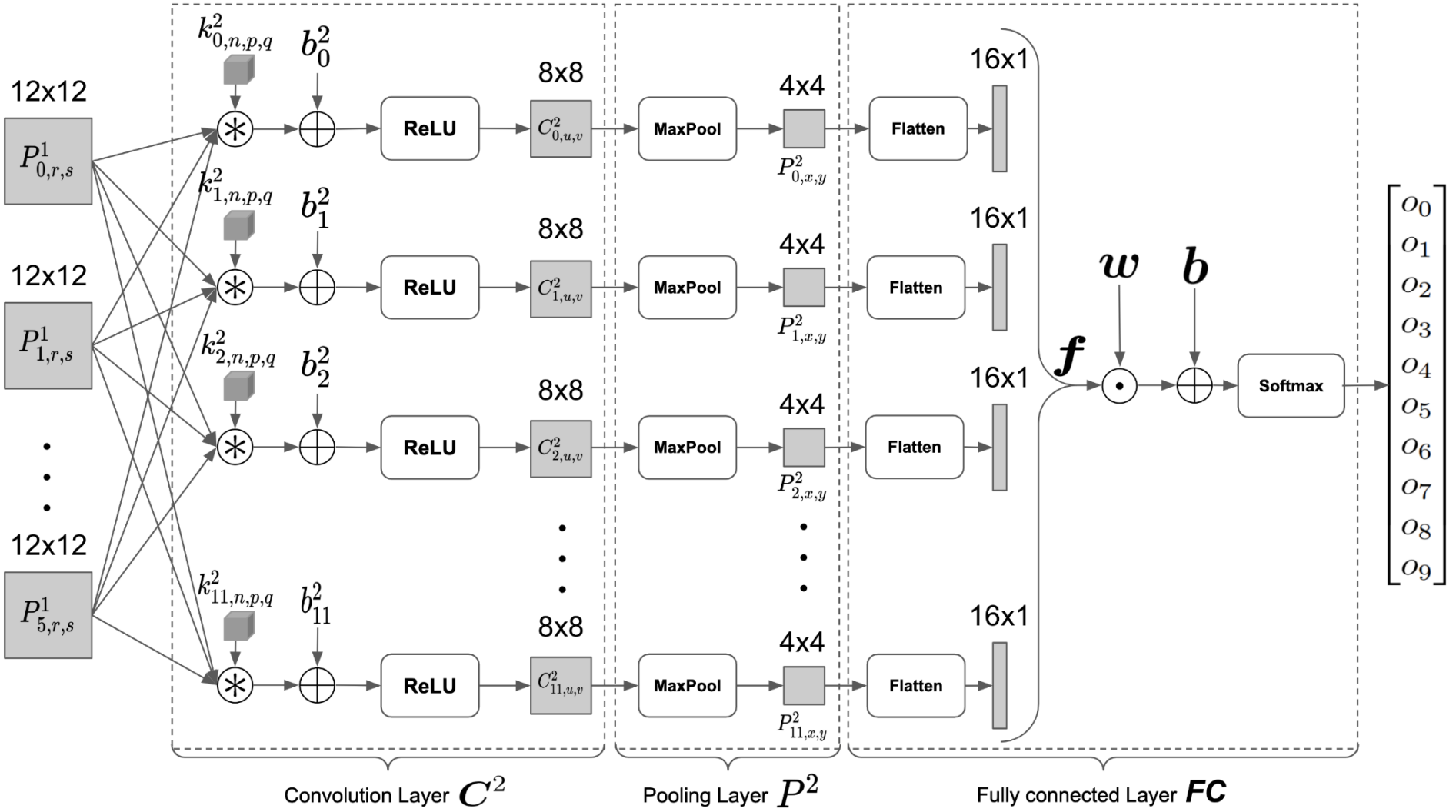
https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#10

CNN Design


feature maps
pooled feature maps
feature maps
pooled feature maps
Fully-connected 1
p(y|x)
Outputs
Input
Convolutional layer 1
Pooling 1
Convolutional layer 2
Pooling 2


Conv_1
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
Conv_2
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
fc_3
Fully-Connected
Neural Network
ReLU activation
fc_4
Fully-Connected
Neural Network
(with dropout)
Flattened
INPUT
(28 x 28 x 1)
n1 channels
(24 x 24 x n1)
n1 channels
(12 x 12 x n1)
n2 channels
(8 x 8 x n2)
n2 channels
(4 x 4 x n2)
n3 units
0
1
2
9
OUTPUT

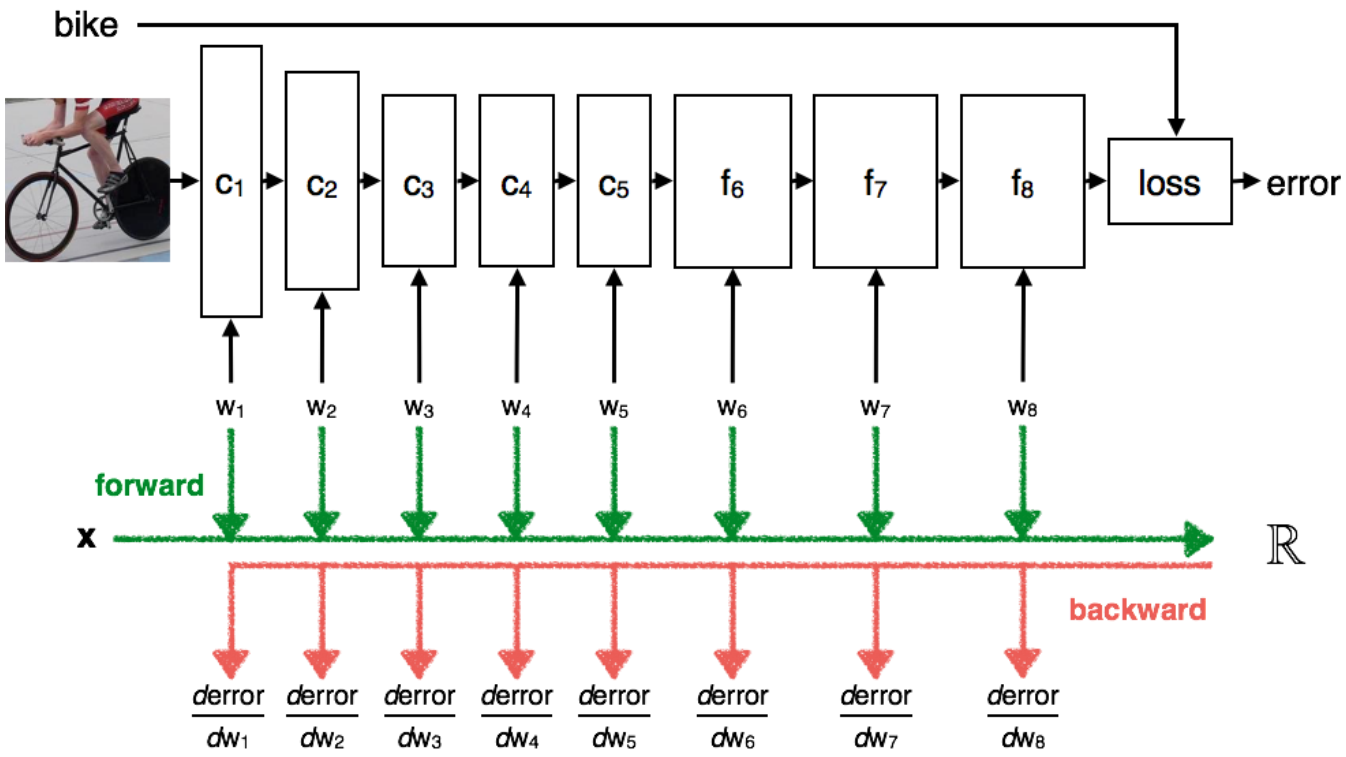
CNN Classifier



CNN TRAINING

CNN Training

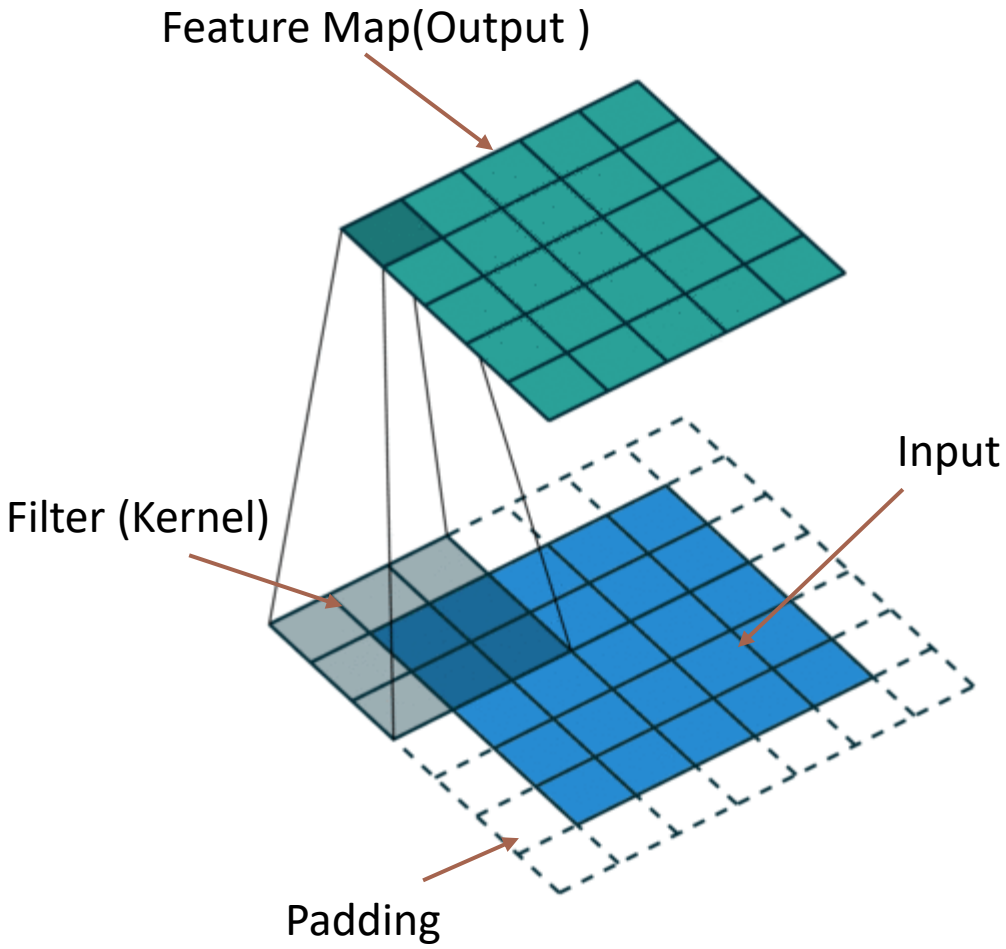
- CNNs train using Backpropagation
- Weights of the CNN are just the filters
- Filters are not known but learned through backpropagation
  - Instead of fixed numbers in the kernel, assign parameters to kernels that train on the data
  - With training, kernels get better at filtering a given image for relevant information



http://andrew.gibiansky.com/blog/machine-learning/convolutional-neural-networks/

Convolution Parameters

- Filter Size (F)
  - Size of convolution filter
- Padding (P)
  - Allows filter to slide through the image
  - Generates output feature map of desired size.
  - Padding with 0 is most common
- Stride (S)
  - No of steps the filter slides
  - S = 1, 1 pixel at a time
  - S ≥ 2 produces smaller output volumes

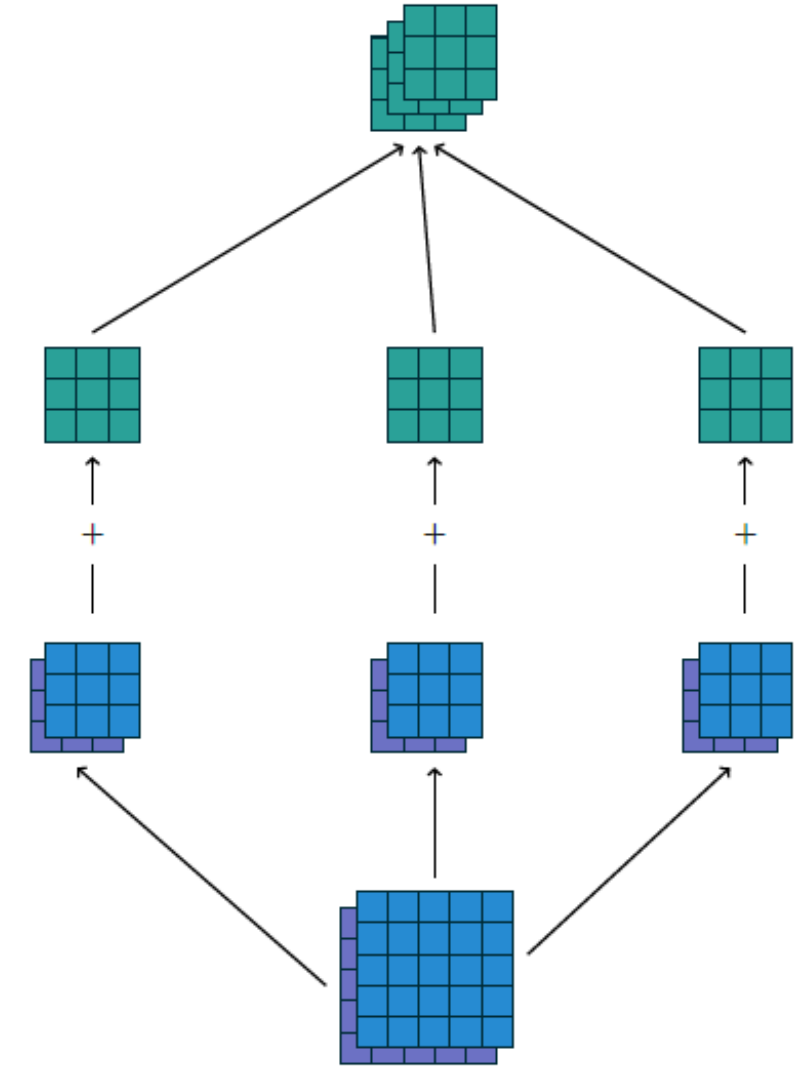


Convolution Mapping

- The size of the output may shrink and the number of channels may grow as more convolutions are applied to the original input

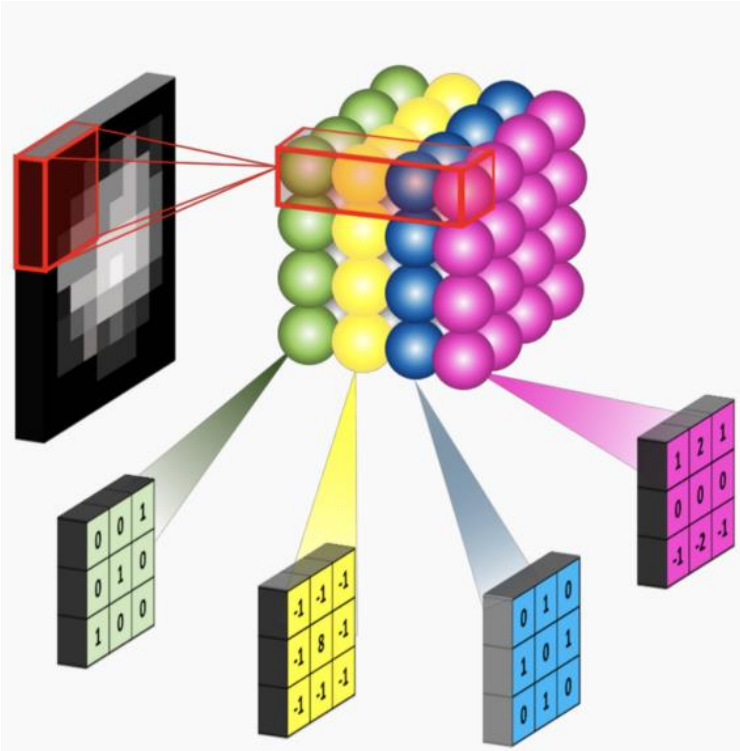
$$o = \left\lfloor \frac{i + 2p - k}{s} \right\rfloor + 1$$

First Layer Representation | Second Layer Representation | Third Layer Representation

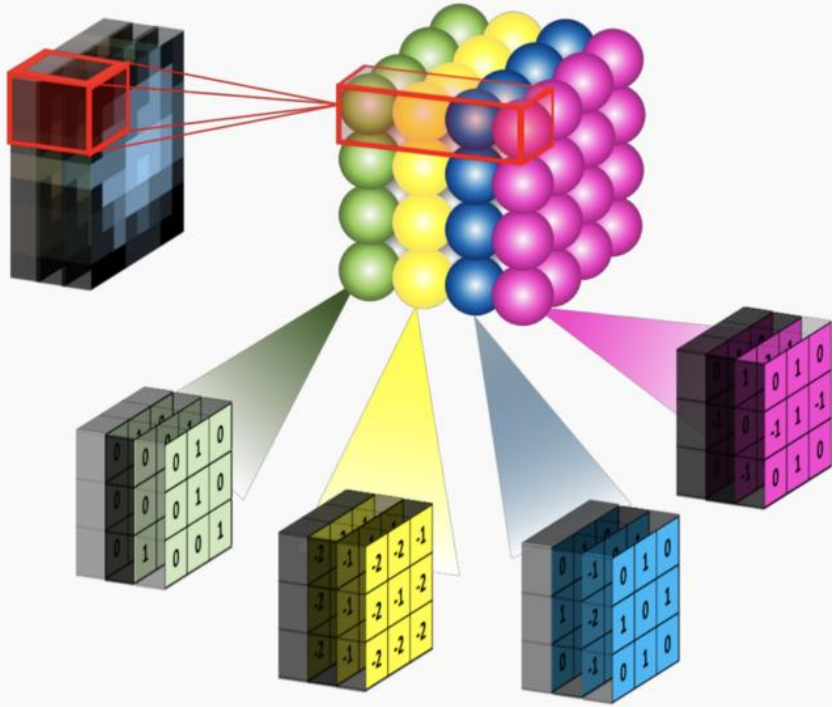


Mapping from two input feature maps to three output feature maps using a 3x2x3x3 collection of kernels w

Multi-channel Convolution



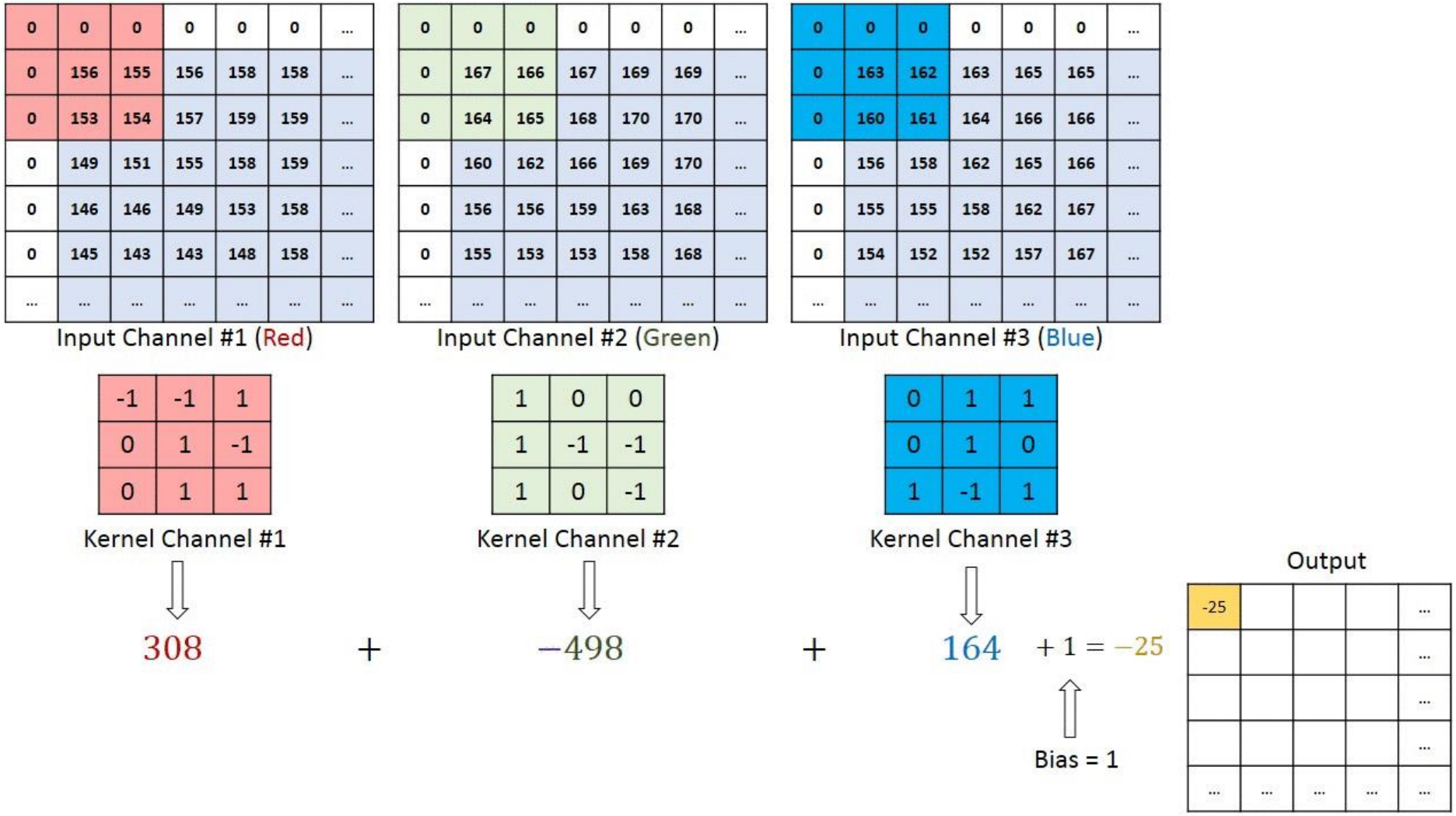
Convolutional layer with four 3x3 filters on a black and white image (just one channel)



Convolutional layer with four 3x3 filters on an RGB image. As you can see, the filters are now cubes, and they are applied on the full depth of the image..

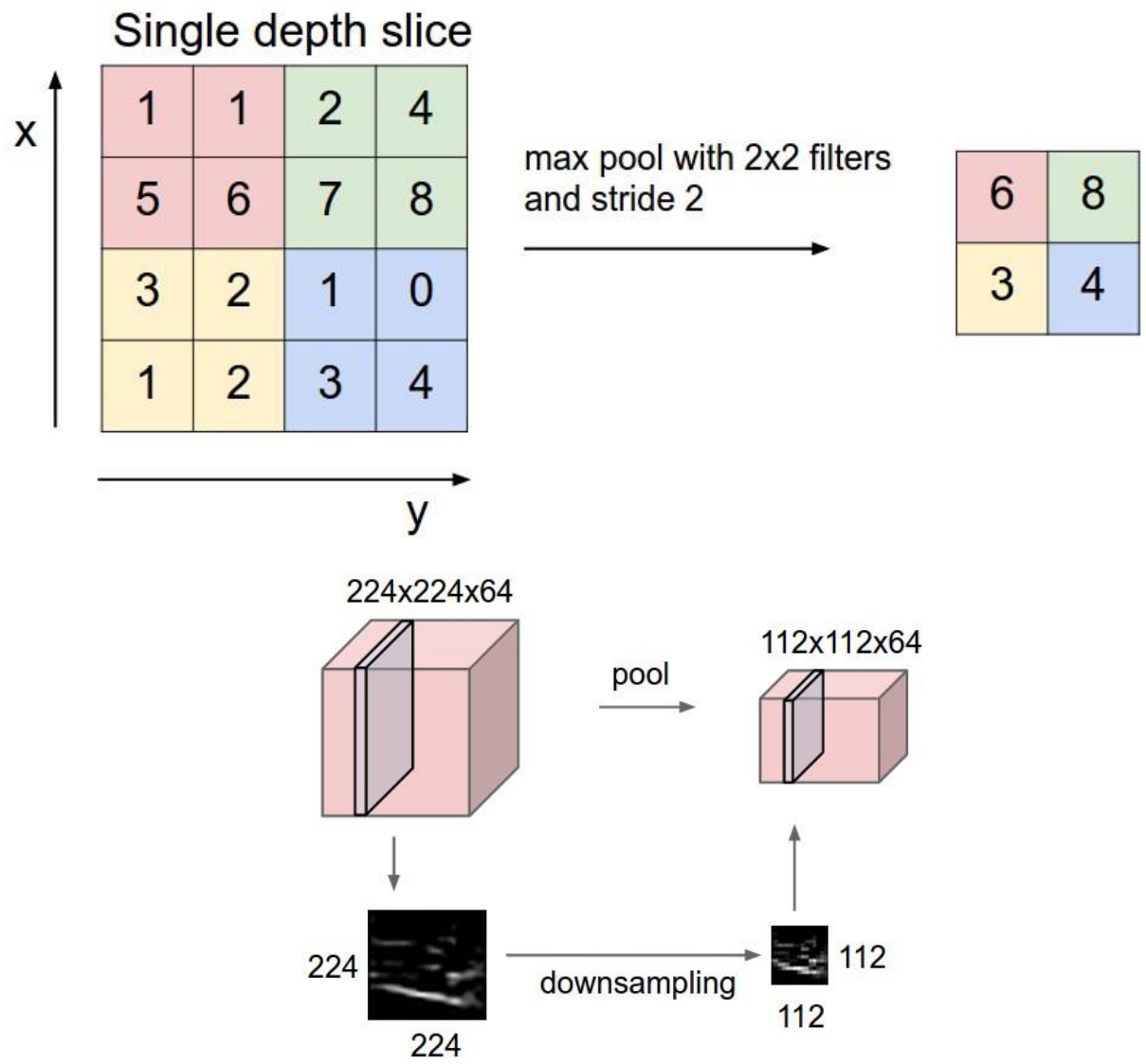
https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac

Multi-channel Convolution



Pooling

- Pooling is a sub-sampling operation
- Reduces size of feature maps by summarizing sub-regions, such as average or the maximum value (max-pooling)
- Max-pooling reduces the dependency of the feature vectors on their exact placement in an image
- Max-pooling layers do not actually do any learning themselves. Instead, then reduce the size of the problem by introducing sparseness

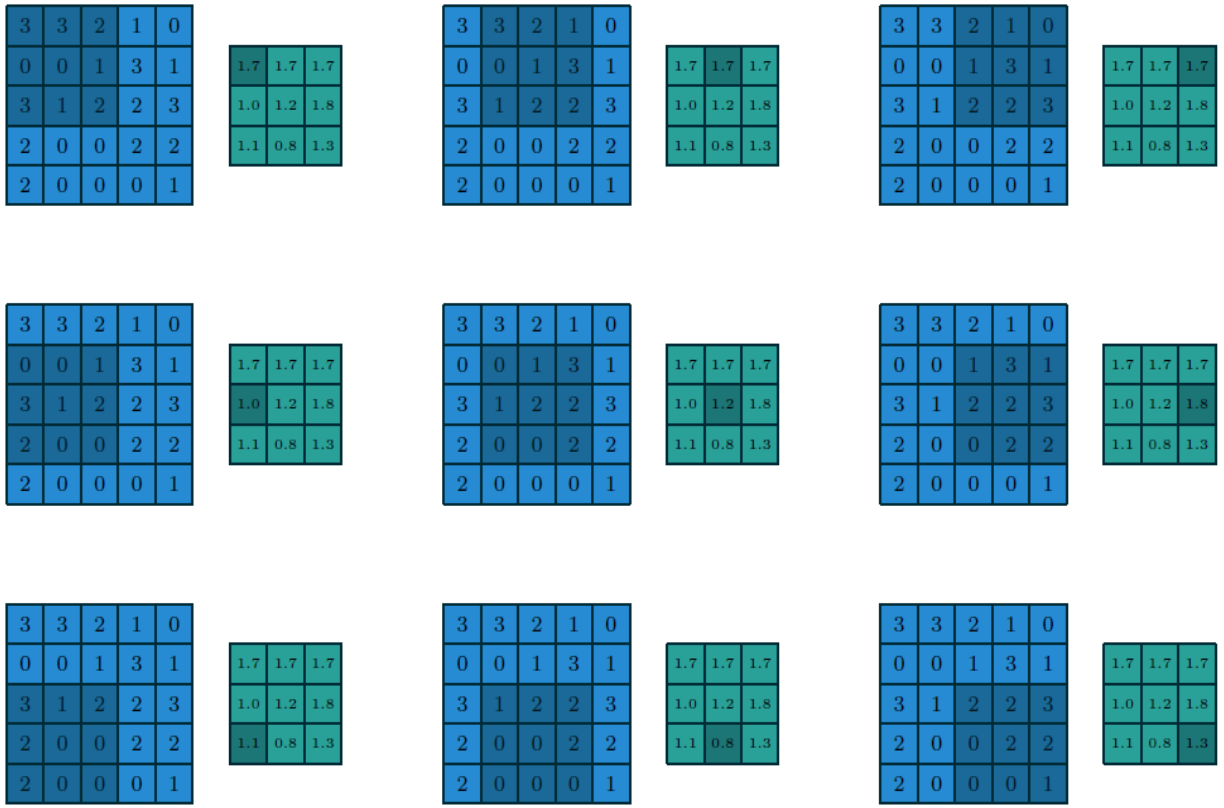


Pooling

Pooling operations reduce the size of feature maps by using some function to summarize sub-regions, such as taking the average or the maximum value

$$o = \left\lfloor \frac{i - k}{s} \right\rfloor + 1$$

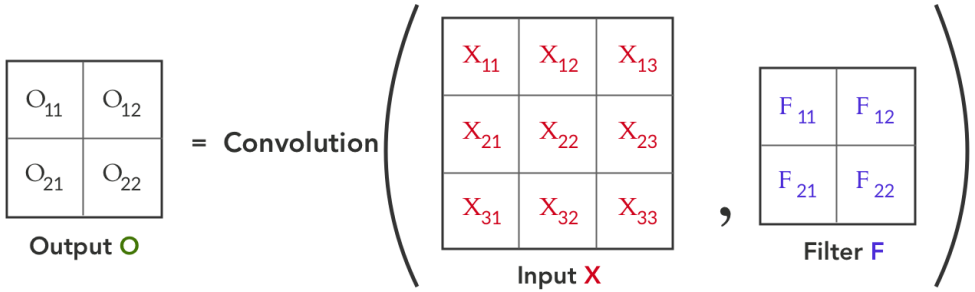
K should be set as powers of 2 for computational efficiency
F is generally taken as odd number
F=1 might sometimes be used and it makes sense because there is a depth component involved

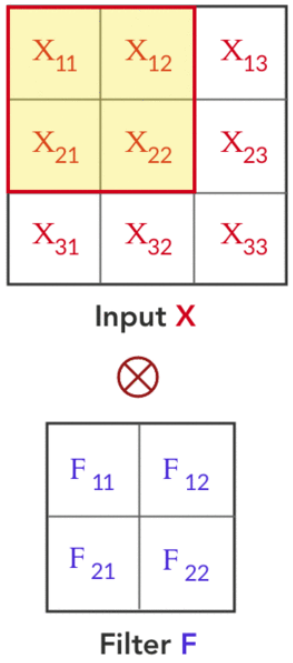


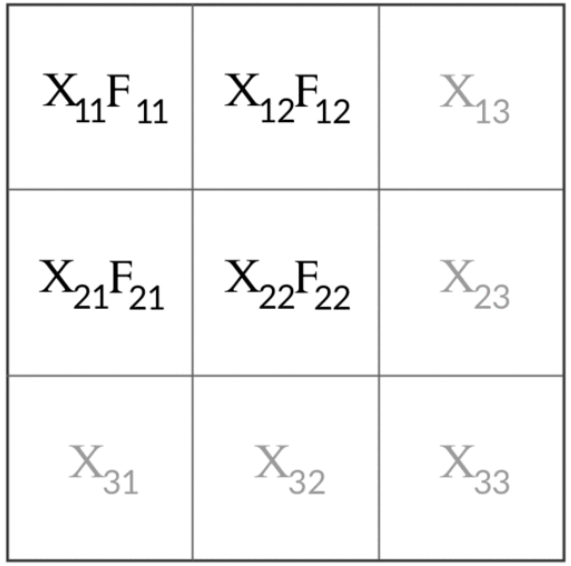
Computing the output values of a 3x3 average pooling operation on a 5x5 input using 1x1 strides

https://github.com/vdumoulin/conv_arithmetic

Convolution



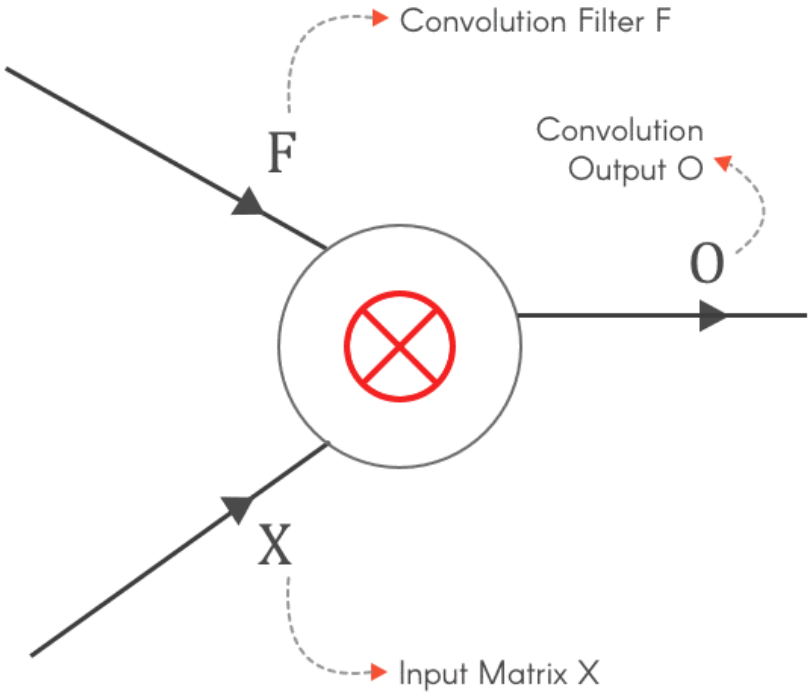


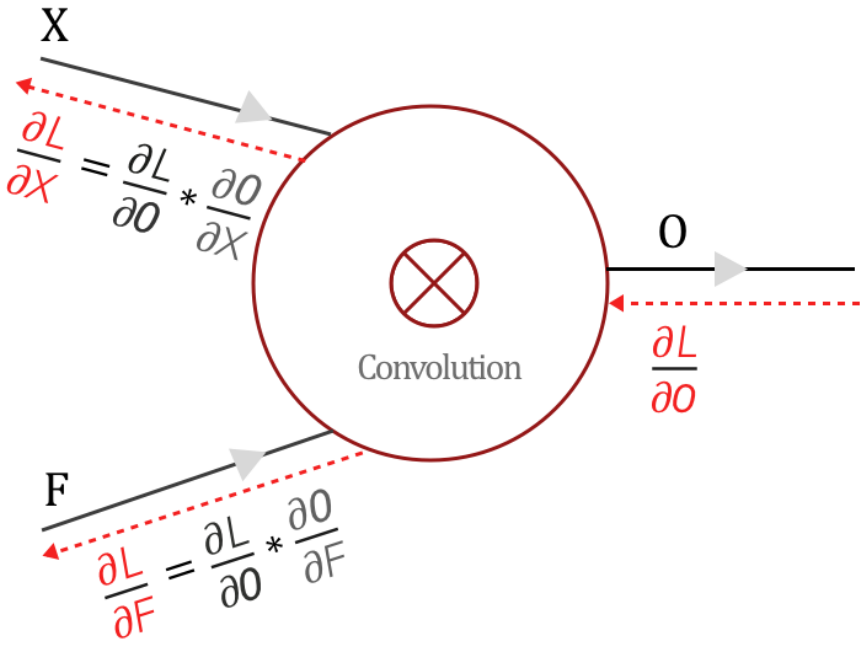


$$O_{11} = X_{11}F_{11} + X_{12}F_{12} + X_{21}F_{21} + X_{22}F_{22}$$

CNN Backpropagation

We can find the local gradients **∂O/∂X** and **∂O/∂F** with respect to Output O. And with loss gradient from previous layers — **∂L/∂O** and using chain rule, we can calculate **∂L/∂X** and **∂L/∂F**





$\frac{\partial O}{\partial X}$ & $\frac{\partial O}{\partial F}$ *are local gradients*

$\frac{\partial L}{\partial z}$ *is the loss from the previous layer which has to be backpropagated to other layers*

https://medium.com/@pavisj/convolutions-and-backpropagations-46026a8f5d2c

CNN Backpropagation

$$\begin{bmatrix} \frac{\partial L}{\partial F_{11}} & \frac{\partial L}{\partial F_{12}} \\ \frac{\partial L}{\partial F_{21}} & \frac{\partial L}{\partial F_{22}} \end{bmatrix} = \text{Convolution} \left(\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix}, \begin{bmatrix} \frac{\partial L}{\partial O_{11}} & \frac{\partial L}{\partial O_{12}} \\ \frac{\partial L}{\partial O_{21}} & \frac{\partial L}{\partial O_{22}} \end{bmatrix} \right)$$

where

$$\begin{bmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{31} & X_{32} & X_{33} \end{bmatrix} = \text{Input X}$$

$$\begin{bmatrix} \frac{\partial L}{\partial O_{11}} & \frac{\partial L}{\partial O_{12}} \\ \frac{\partial L}{\partial O_{21}} & \frac{\partial L}{\partial O_{22}} \end{bmatrix} = \frac{\partial L}{\partial O}$$

Loss gradient from previous layer

https://medium.com/@pavisj/convolutions-and-backpropagations-46026a8f5d2c

CNN Backpropagation

$$\frac{\partial L}{\partial X} = \text{Full Convolution}\left(\begin{bmatrix} F_{22} & F_{21} \\ F_{12} & F_{11} \end{bmatrix}, \begin{bmatrix} \frac{\partial L}{\partial O_{11}} & \frac{\partial L}{\partial O_{12}} \\ \frac{\partial L}{\partial O_{21}} & \frac{\partial L}{\partial O_{22}} \end{bmatrix} \right)$$

$$\frac{\partial L}{\partial X} = \begin{bmatrix} \frac{\partial L}{\partial X_{11}} & \frac{\partial L}{\partial X_{12}} & \frac{\partial L}{\partial X_{13}} \\ \frac{\partial L}{\partial X_{21}} & \frac{\partial L}{\partial X_{22}} & \frac{\partial L}{\partial X_{23}} \\ \frac{\partial L}{\partial X_{31}} & \frac{\partial L}{\partial X_{32}} & \frac{\partial L}{\partial X_{33}} \end{bmatrix}$$

Filter F

Loss Gradient $\frac{\partial L}{\partial O}$

∂L/∂X can be represented as ‘full’ convolution between a 180-degree rotated Filter F and loss gradient ∂L/∂O

https://medium.com/@pavisj/convolutions-and-backpropagations-46026a8f5d2c

CNN Backpropagation

- Both the Forward pass and the Backpropagation of a Convolutional layer are Convolutions

$$\frac{\partial L}{\partial F} = \text{Convolution}\left(\text{Input } X\text{ , Loss gradient } \frac{\partial L}{\partial O}\right)$$

$$\frac{\partial L}{\partial X} = \begin{matrix}\text{Full}\\ \text{Convolution}\end{matrix}\left(\begin{matrix}180^{\circ}\text{rotated}\\ \text{Filter } F\end{matrix}\text{ , }\begin{matrix}\text{Loss}\\ \text{Gradient}\end{matrix}\ \frac{\partial L}{\partial O}\right)$$

How to calculate **∂L/∂X and ∂L/∂F**

https://medium.com/@pavisj/convolutions-and-backpropagations-46026a8f5d2c

CNN Architecture Considerations & Best Practices

- Architecture
  - Number of channels generally increases or stays the same while we progress through layers
  - Dropout layer is placed after the Max-Pool layer
  - Batch Normalization layer is placed after the activation function.
  - Avoid negative dimensions due to high values of stride length and filter-sizes
  - Choose the architecture(s) for which you wish to do hyper-parameter optimization
  - Track dimensions, activation shapes, and sizes for the architecture

Layer	Number of Filters	Padding	Activation Shape	Activation Size
Input Image	-	-	(28,28,1)	784
Conv2d(f=3,s=1)	8	Same	(28,28,8)	6,272
MaxPool(f=2,s=2)	-	Valid	(14,14,8)	1568
Conv2d(f=5,s=1)	16	Valid	(10,10,16)	1,600
MaxPool(f=2,s=2)	-	Valid	(5,5,16)	400
Flatten	-	-	(400,1)	400
Flatten	-	-	(120,0)	120
Dense	-	-	(64,1)	64
Softmax	-	-	(10,1)	10

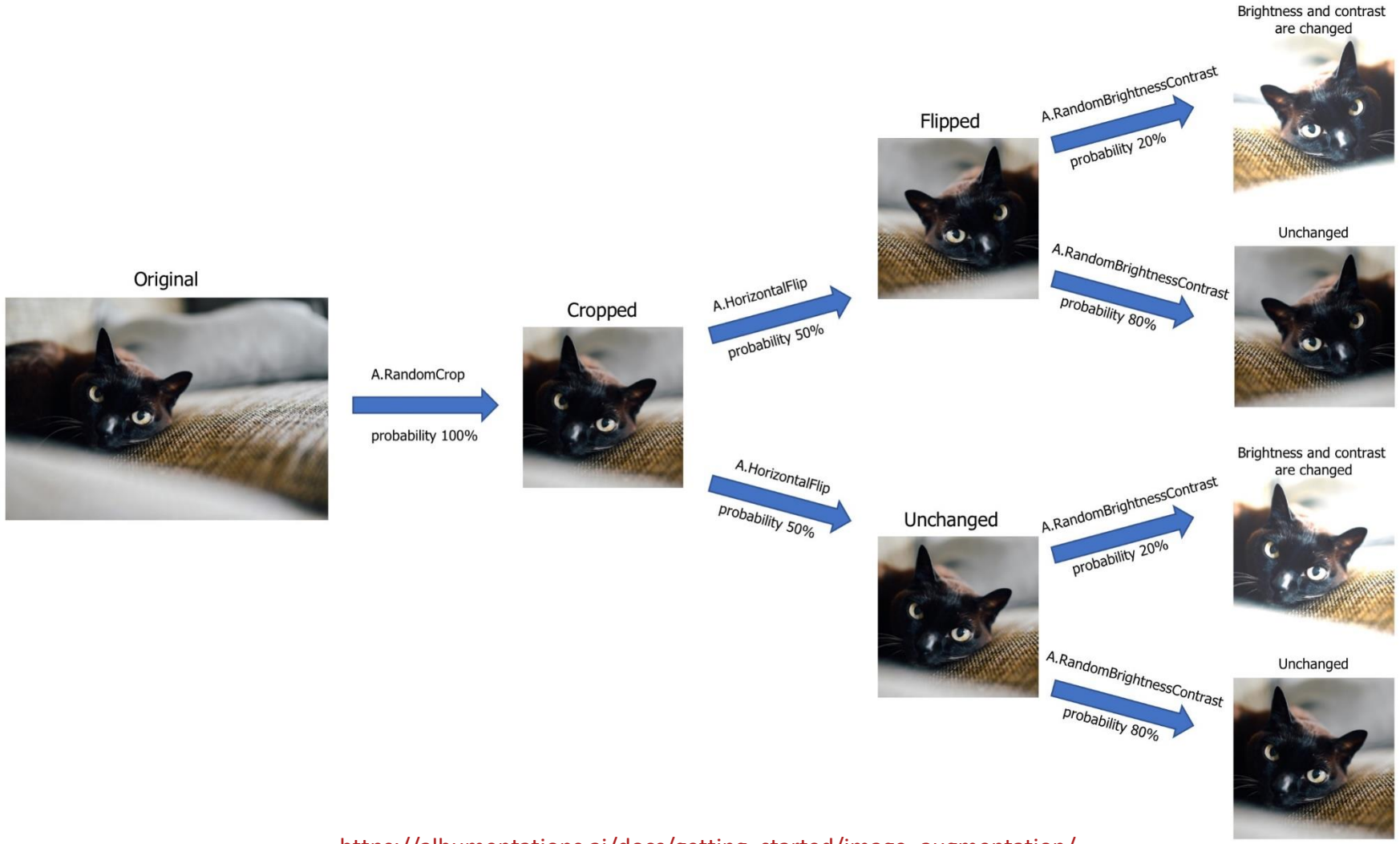
CNN Training Considerations & Best Practices

- Training Data
  - CNNs require large training samples
  - Perform image augmentation - multiple image orientations and noise
- Vanishing & Exploding Gradients
  - Lowering the learning rate can help keep ReLU units from dying
  - Add Batch Normalization to prevent overfitting and exploding gradients
- Regularization
  - Pick dropout values between 0.0 (no dropout) and 1.0 (dropout everything)
- Training Speed
  - GPU/TPUs
  - BLAS libraries

https://developers.google.com/machine-learning/crash-course/training-neural-networks/best-practices

https://www.mosaicml.com/blog/5-best-practices-for-efficient-model-training

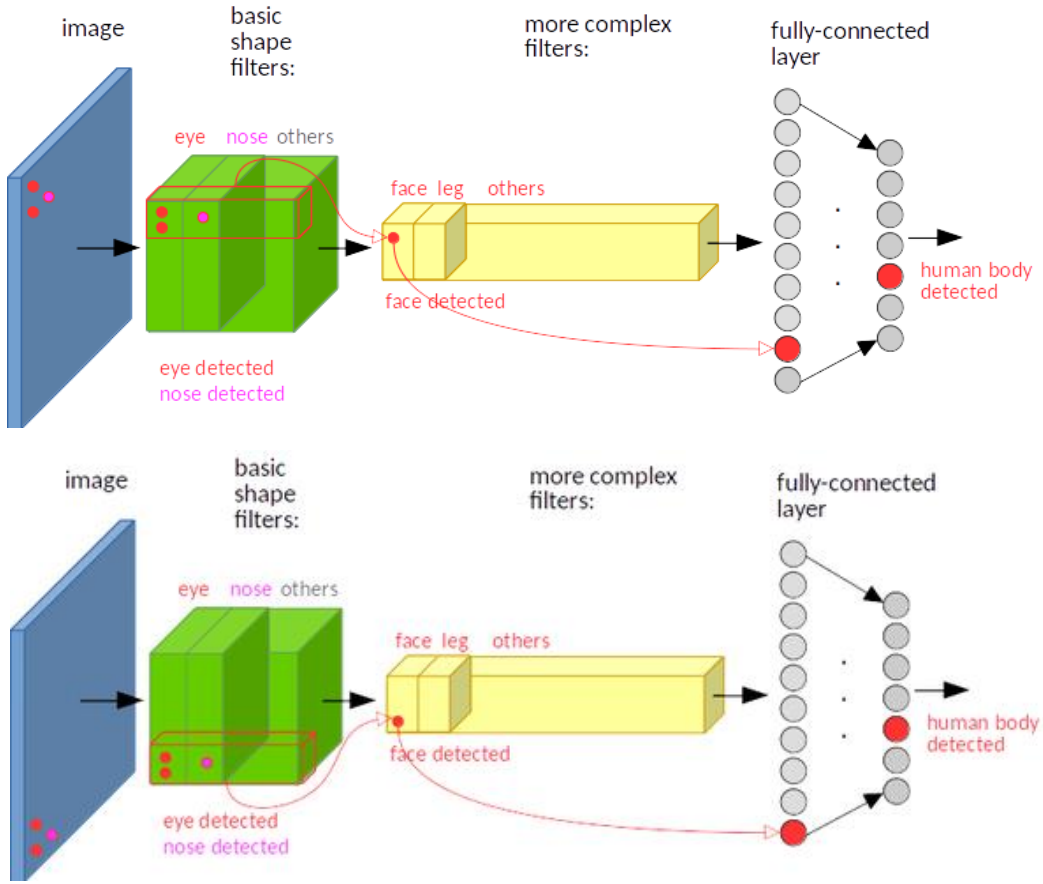
Image Augmentation



https://albumentations.ai/docs/getting_started/image_augmentation/

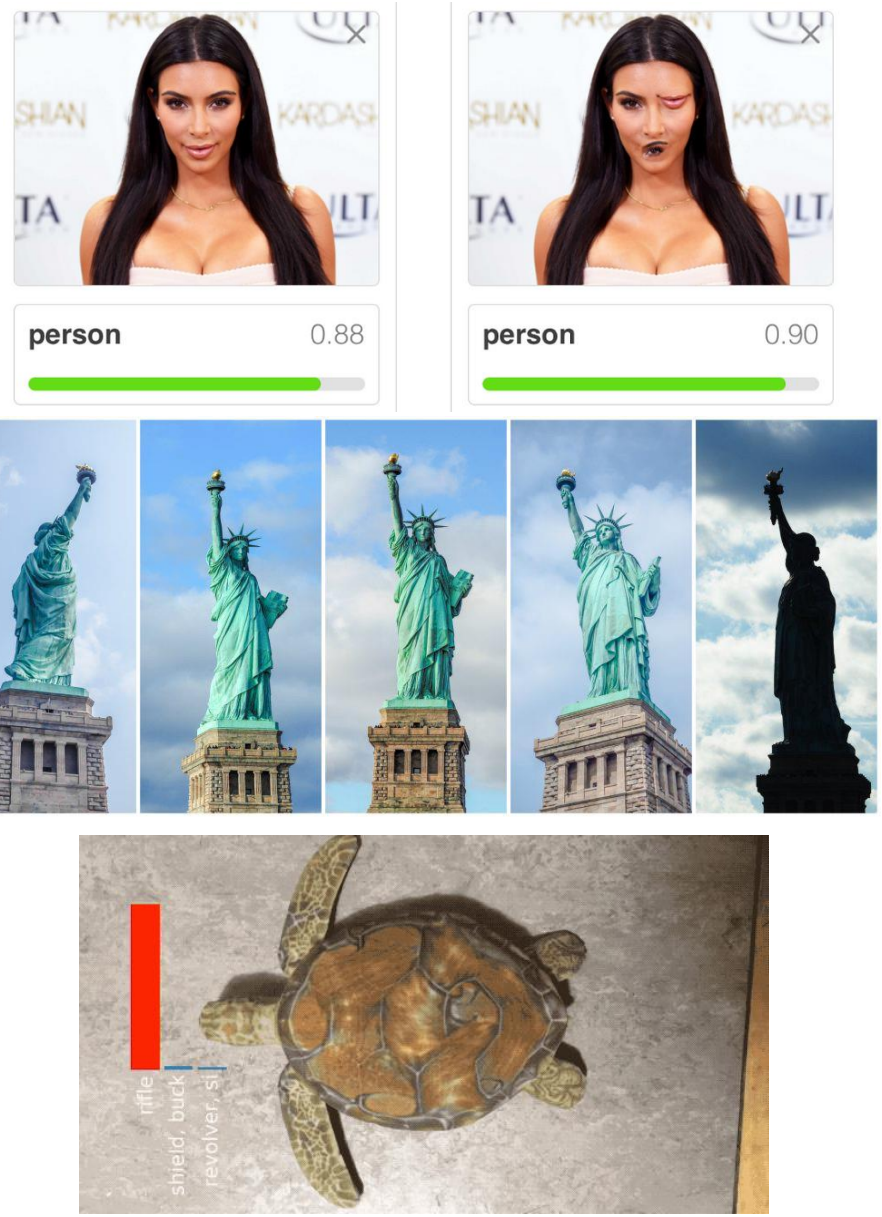
Invariance

- CNNs are invariant to object translation due to convolution filters
- Regular CNNs are not rotation invariant



CNN Issues

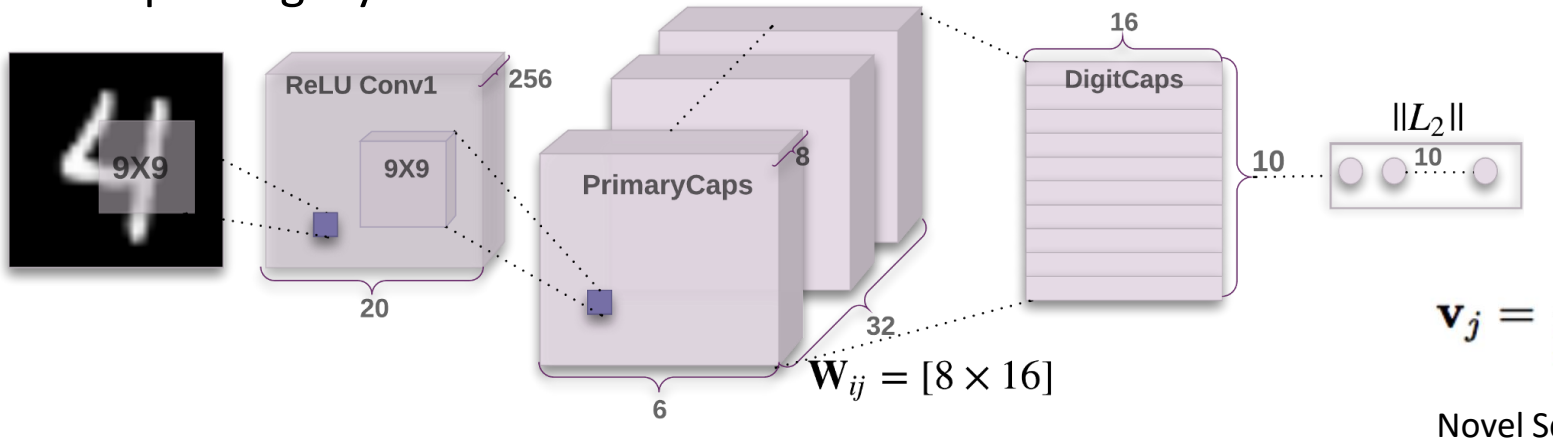
- Fooled by images with features in the wrong place
- Easily confused when viewing an image in a different orientation
  - One way to combat this is with excessive training of all possible angles, but this takes a lot of time
- CNNs can be susceptible to white box adversarial attacks
  - Embedding a secret pattern into an object to make it look like something else



https://towardsdatascience.com/what-is-wrong-with-convolutional-neural-networks-75c2ba8fbd6f

Capsule Networks

- Capsule is a nested set of neural layers - more layers inside a single layer
- Invariance
  - Capsule takes into account the existence of the specific feature like mouth or nose.
- Equivariance
  - Instead of making the feature translation invariant, capsule will make it **translation-equivariant** or viewpoint-equivariant. As the feature moves and changes its position in the image, feature vector representation also changes thus making it equivariant. This property of capsules tries to solve the drawback of max pooling layers.



$$\mathbf{v}_j = \frac{||\mathbf{s}_j||^2}{1 + ||\mathbf{s}_j||^2} \frac{\mathbf{s}_j}{||\mathbf{s}_j||}$$

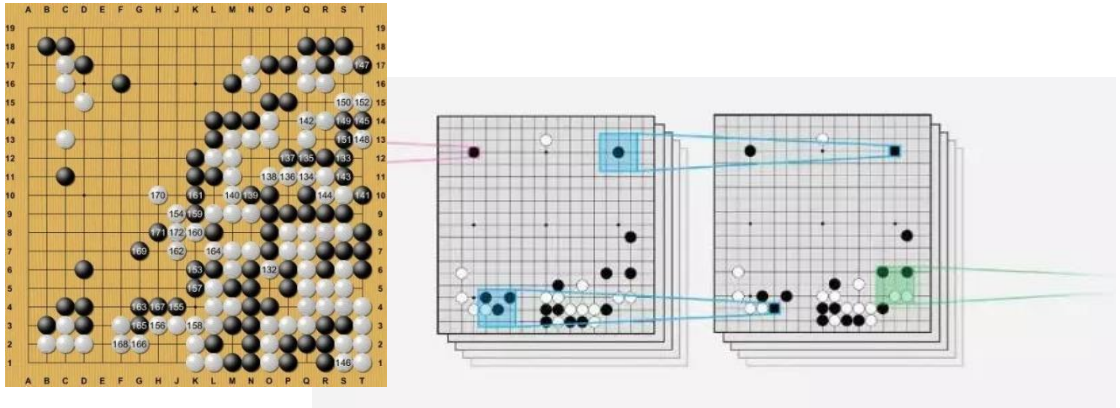
Novel Squashing Function

https://arxiv.org/abs/1710.09829

CNN APPLICATIONS

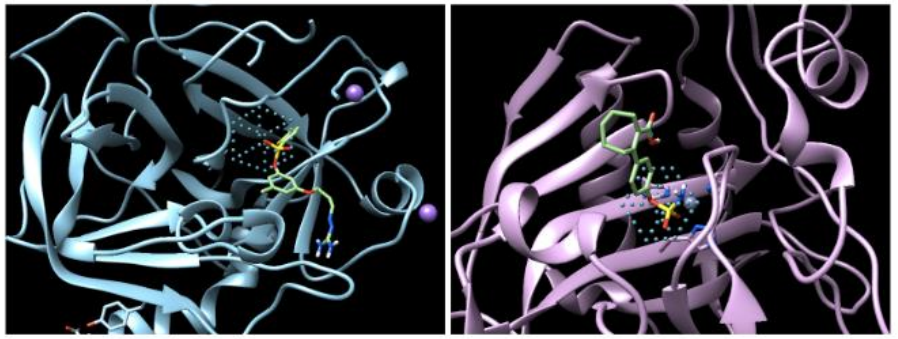
CNN Applications

http://www.cs.cmu.edu/~vramakri/poseMachines.html



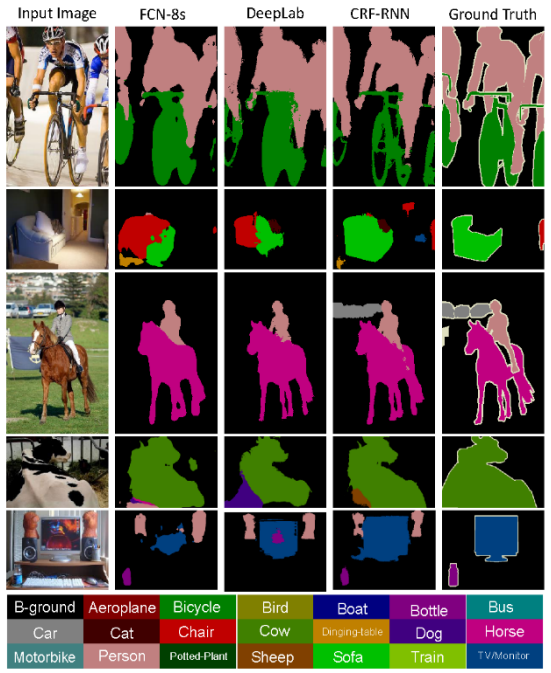
https://deepmind.com/research/alphago/

https://github.com/jcjohnson/neural-style/



https://arxiv.org/pdf/1510.02855.pdf

Figure 5: Sulfonyl/sulfonamide detection with autonomously trained convolutional filters.



http://www.robots.ox.ac.uk/~szheng/CRFasRNN.html

Yoga Pose Detection

- 10 Yoga poses detected
- ~500+ images per pose
- Model: CNN Resnet-50 with Transfer Learning
- Tensorflow on iPhone

https://grahamschool.uchicago.edu/academic-programs/masters-degrees/analytics/yoga

Lab

1. MNIST MLP
2. MNIST CNN - Keras
3. CIFAR CNN - Keras
4. https://transcranial.github.io/keras-js/#/