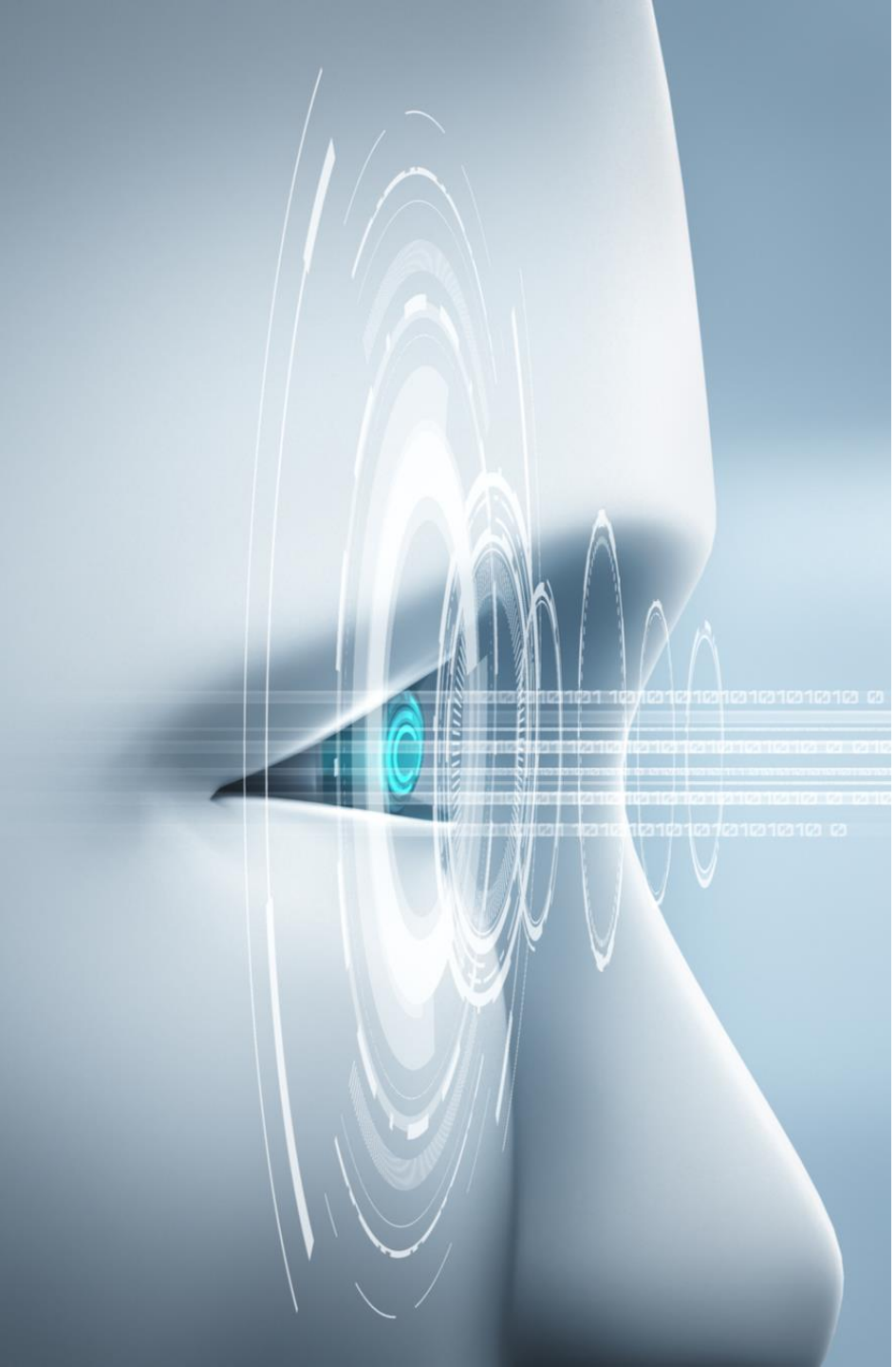


Computer Vision Applications

BY QUADEER SHAIKH

About me

Work Experience

- Risk Analyst
  - Morgan Stanley (Jan 2023 – Present)
- Data Science Intern
  - AkzoNobel Coatings International B.V. Netherlands (Feb 2022 – Dec 2022)
- Data Science Intern
  - EzeRx Health Tech Pvt. Ltd. (Jan 2022 – July 2022)
- Associate Engineer
  - Tata Communications Ltd. (July 2019 – Aug 2020)
- Network Automation and Analysis Engineer Intern
  - Cisco (June 2018 – July 2018)

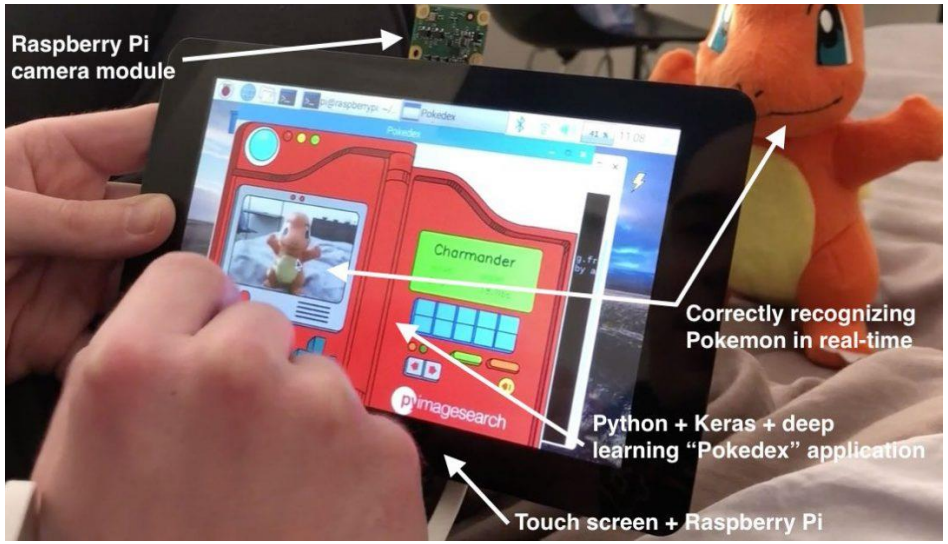
Education

- M.Tech – Artificial Intelligence
  - NMIMS (2021 - 2023, currently pursuing)
- B.E. – Computer Engineering
  - Mumbai University (2015 - 2019)

The Deep Learning Era

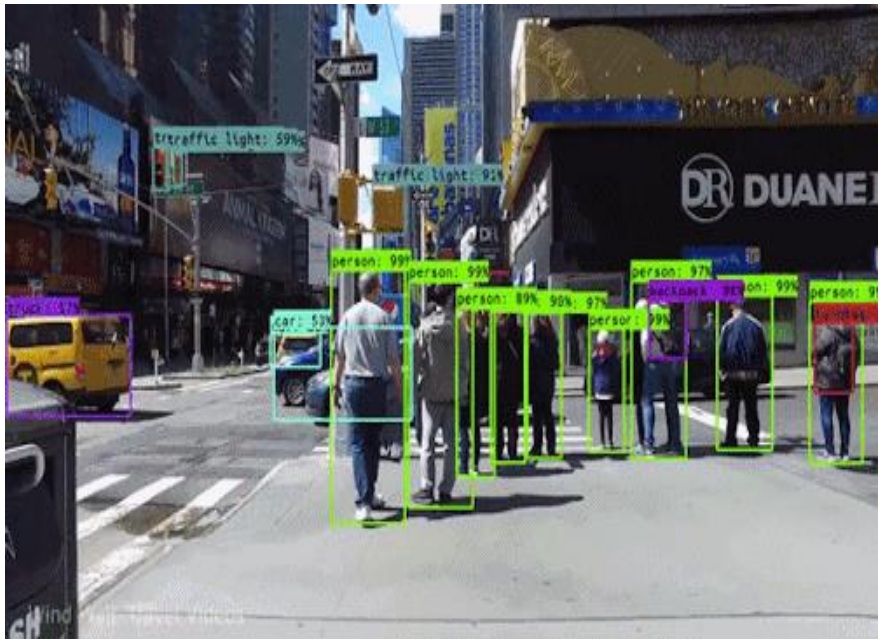
Deep Learning Phase Outline

1. A revisit to an old friend: Neural Networks
2. A visit to a new friend: Convolutional Neural Networks (CNNs)
3. Image Augmentation
4. Optimization and Improvement of CNNs
5. Diagnosis of CNNs
6. CNNs Architectures and Transfer Learning
    1. AlexNet
    2. VGG16/19
    3. InceptionNet/GoogleNet
    4. ResNet
    5. MobileNet
    6. EfficientNet



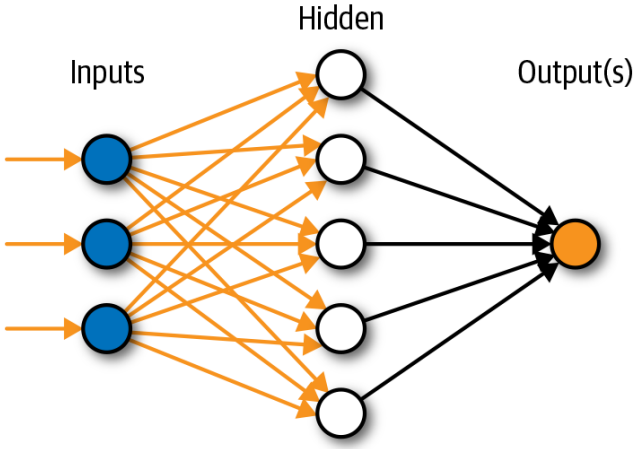
Deep Learning Phase Outline

1. What do CNNs Learn ?
2. One Shot Learning (Face Recognition)
    1. Siamese Network
3. Image Segmentation
    1. FCN
    2. U-Net (Optional)
4. Object Detection (Architecture details and training framework)
    1. RCNN
    2. Fast RCNN
    3. Faster RCNN
    4. YOLO



A revisit to an old friend: NNs

1. An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain
2. Used for non linear pattern recognition
3. Always count the hidden layers + output layers
   - This is a 2 layers neural network
4. Hyperparameters need to be configured:-
   1. No. of layers
   2. No. of neurons in each layer
   3. No. of epochs
   4. Learning rate
   5. Loss function, Optimizer
   6. Batch size
5. How do you count the total parameters in your neural network ?



A revisit to an old friend: NNs

1. How do you calculate the total parameters in the neural network ?

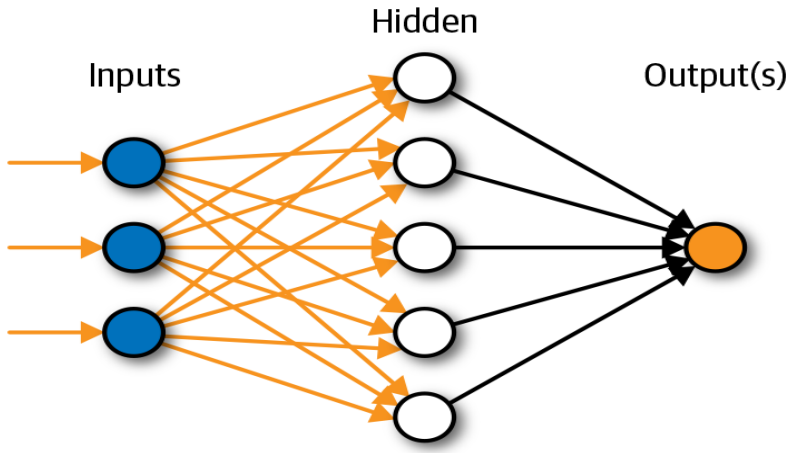
Ans. A neural network has 2 kind of params: weights and biases

Params for a given layer l

Weights = No. of units in current layer (l) x No. of units in prev layer (l-1)

Biases = No. of units in current layer (l) x 1

Params = no. of weights + no. of biases



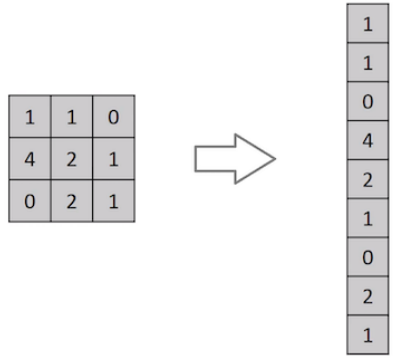
A revisit to an old friend: NNs

How to perform image classification using ANNs ?

1. Flatten the image
2. Pass the flattened image to the neural network

Problems with this for image related tasks:-

1. Images have a 2D spatial structure, therefore considering only the flattened pixel values are not very useful features for tasks like classification
   - Pixels that spatially separated are treated the same way as pixels that are adjacent
2. No way for network to learn features at different places in an input image
3. Imagine flattening an image of resolution 224x224x3, every image will be represented using a vector of size 150,528.
4. Now imagine using a neural network which has 2 hidden layers of 512 and 128 neurons each and one output layer of 1 neuron. Can you calculate the number of parameters required ?



Params

1. 512 x 150,528 + 512 x 1 = 77,070,848
2. 128 x 512 + 128 x 1 = 65,664
3. 1 x 128 + 1 x 1 = 129

Total = 77,136,641

A revisit to an old friend: NNs

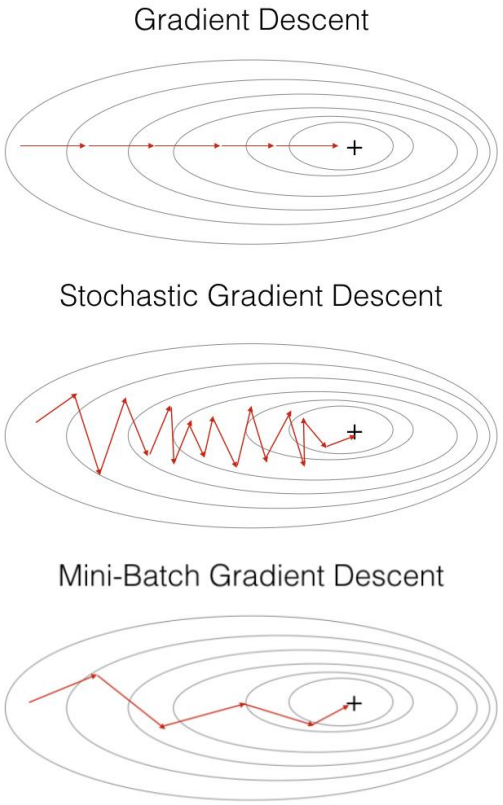
Fantastic optimization algorithms and how to use them

1. Batch Gradient Descent
   - A misleading name given by the ML community, the batch gradient descent updates the weight of the neural network by performing forward & backward propagation on the entire dataset i.e. all the training samples/records in a single epoch/iteration

2. Stochastic Gradient Descent
   - In batch gd we consider all the training records, but what if the training dataset is huge. You cannot fit huge amount of data into your memory (RAM) for the training process. We take one example each time from the records to update the weights using forward-backward prop (done for all records one by one in an entire epoch). Cost fluctuates a lot during the minimization and starts decreasing in the long run. Has a problem with convergence to minima.

3. Mini Batch Gradient Descent
   - SGD has a minima convergence problem and also cannot be implemented with vectorized computation (due to one sample taken each time for a forward-backward prop. Therefore we take mini batches from the dataset and update the weights repeatedly batch wise for an entire epoch. Convergence to minima is smoother than SGD.

Other optimizers like RMSProp, AdaGrad, Adam, etc also use mini batch implementations



A visit to a new friend: CNNs

Remember Convolution operation ?

Kernels/Filters were used to detect features in an image by convolving them on the image

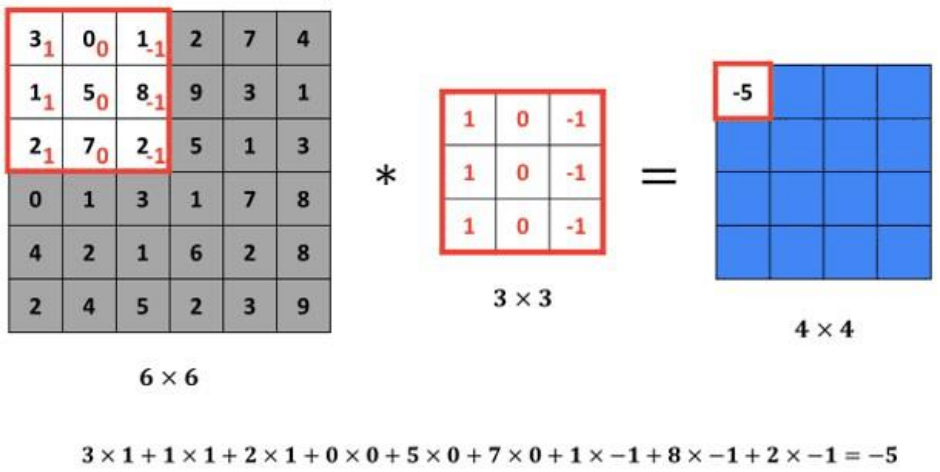
How to decide weights of kernels ?

Make the kernels learnable

Treat the kernels as trainable weights

What about nonlinear patterns in images ?

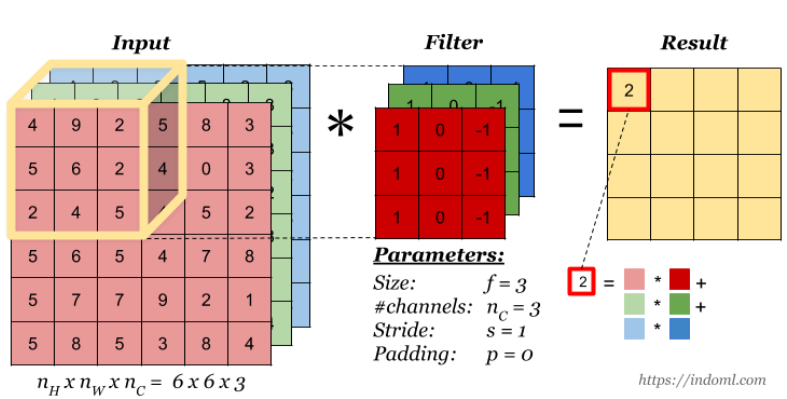
Add activation functions to the kernels

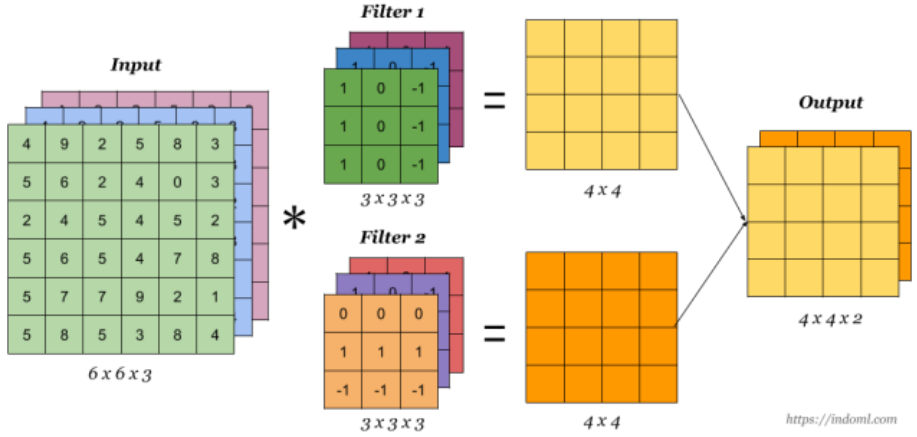


Convolutions over Volumes

Convolutions on RGB image/over volumes

Note: Filter/Kernel depth should be the same as input image depth





CNNs: Understanding Dimensions

Input Dimensions: Width (W1) x Height (H1) x Depth (D1) [e.g. 224x224x3]

Spatial Extent of Filter/Kernel: F (depth of the filter is same as of input)

Output Dimensions: W2 x H2 x D2

Strides: S

Kernels: K

Padding: P

CNNs: Understanding Dimensions

Lets just consider the W1 and H1 for now. Suppose we have an image of size 5x5 (W1xH1) and kernel of size 3x3.

The result of convolution is 3x3

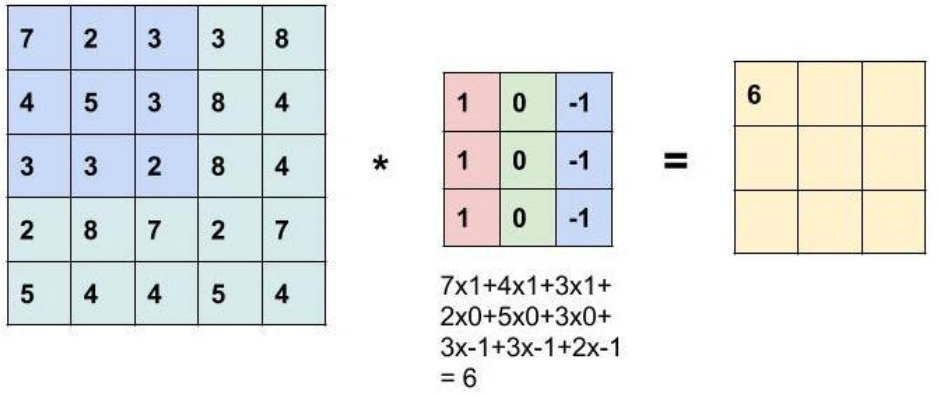
W2: W1 – F + 1 -> 5 – 3 + 1 = 3

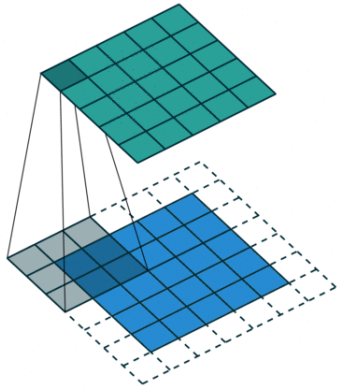
H2: H1 – F + 1 -> 5 – 3 + 1 = 3

Lets say we have a 7x7 image and kernel of size 3x3

7 – 3 + 1 = 5

Note: The output result dimensions are smaller than input dimensions





CNNs: Understanding Dimensions

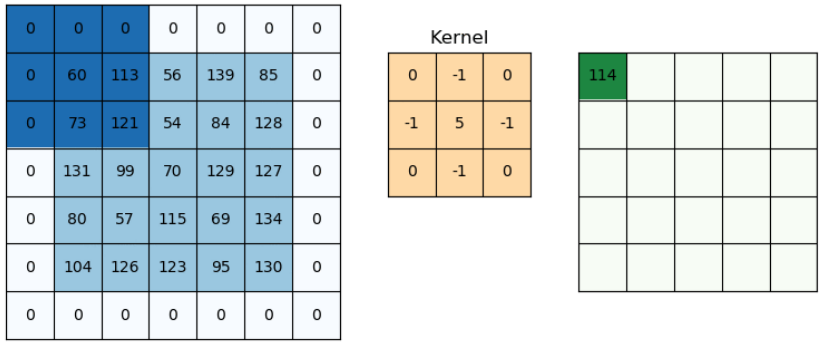
What if we want the output of convolution to be of same size as input ?

Pad input image with appropriate number of inputs so that you can now apply the kernel on input image corners as well. There are different methods for padding but the most commonly used one is zero padding. No. of Padding units (P)

If P is set to 1 then it will pad rows and columns of 0's to the top, bottom, left and right of the image

W2: W1 – F + 2P +1, **H2:** H1 – F + 2P +1

5 – 3 + 2x1 + 1 = 5



CNNs: Understanding Dimensions

What does stride do ?

Defines the intervals at which the kernel is applied to the image

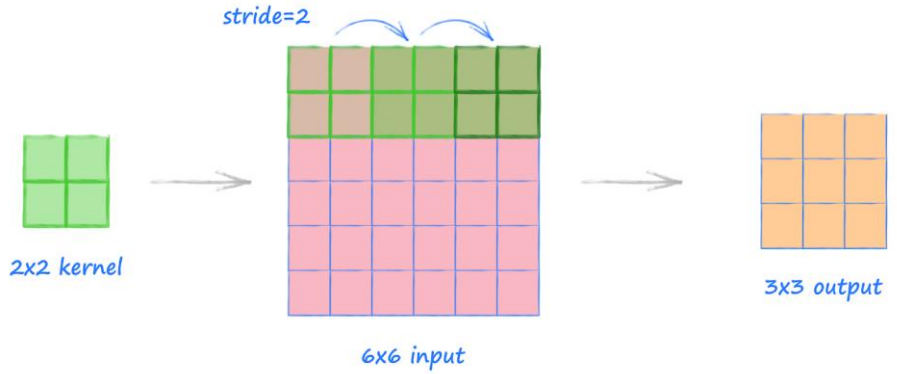
Results in an output of smaller dimensions

W2: [(W1 – F + 2P)/S] + 1

H2: [(H1 – F + 2P)/S] + 1

6x6 image, 2x2 kernel and stride 2

[(6 – 2 + 2x0)/2] + 1 = 3



CNNs: Understanding Dimensions

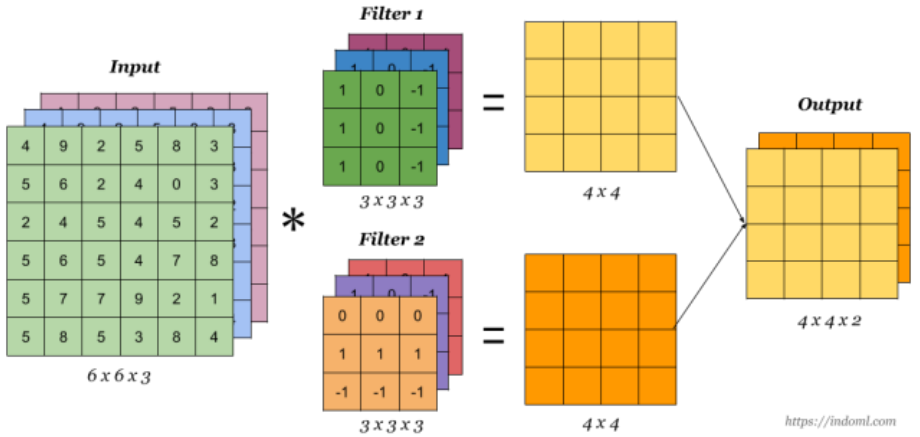
Lets talk about the depth of the output now.

Each kernel/filter gives us a 2D output

If we use K filters then we will get K such 2D outputs

The resulting dimensions would then be W2xH2xK

Thus, D2 = K



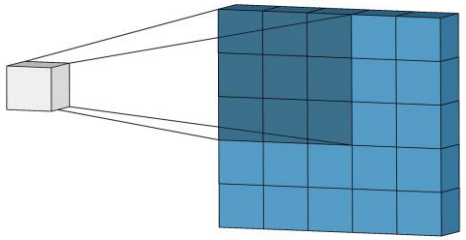
What were the challenges faced by ANNs aka FNNs in image tasks ?

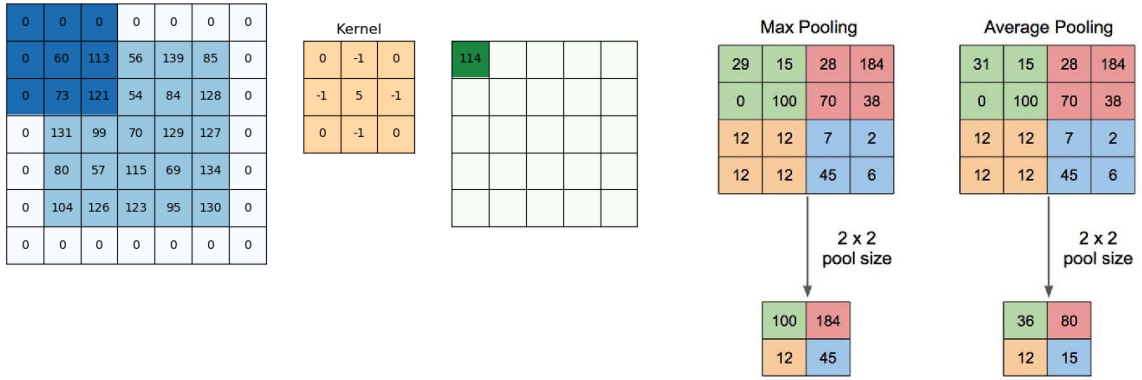
Problems with this for image related tasks:-

1. Images have a 2D spatial structure, therefore considering only the flattened pixel values are not very useful features for tasks like classification
   - Pixels that spatially separated are treated the same way as pixels that are adjacent
2. No obvious way for network to learn features at different places in an input image (similar edges, corners or blobs)
3. Can get computationally expensive for large images

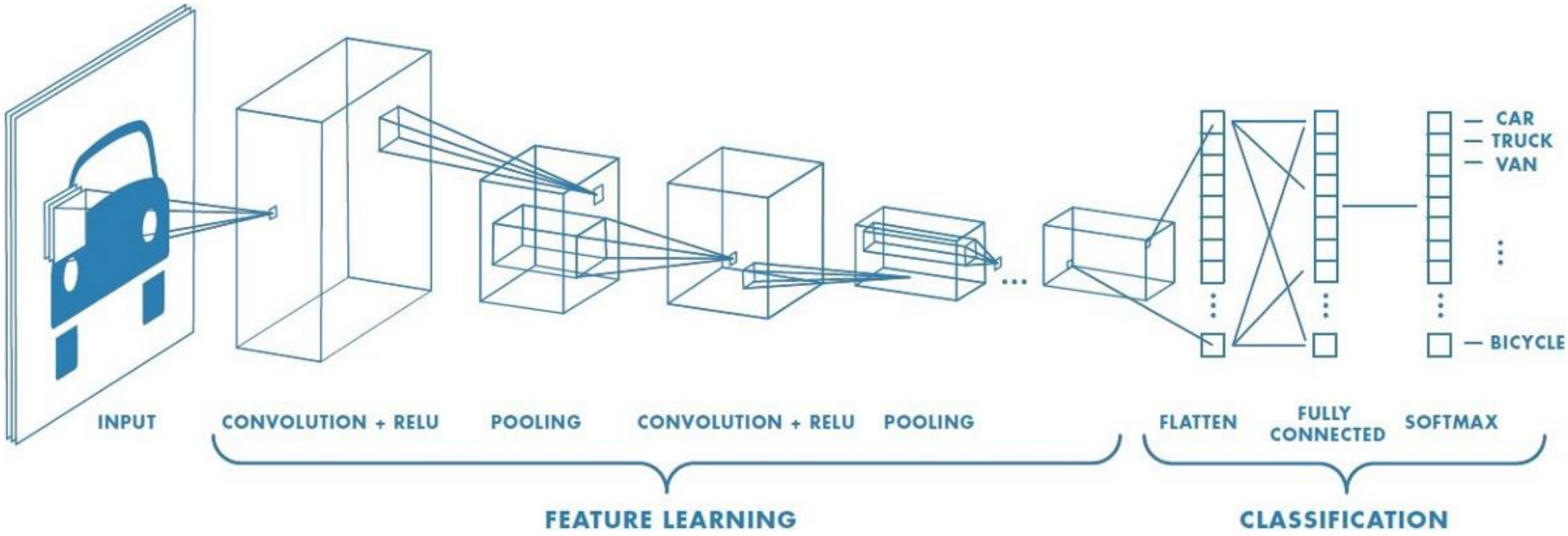
How do CNNs solve these challenges ?

1. Local Receptive Fields: Hidden units are connected to the local patches from the previous layer. This serves 2 main purposes:-
   - Captures local spatial relationships in pixels
   - Reduces the number of parameters significantly
2. Weight Sharing: Also serves 2 purposes
   - Enables rotation, scale and translational invariance to objects in images (no more usage of the painful SIFT algo)
   - Reduces number of parameters in the model
3. Pooling: condenses info from previous layer
   - Aggregates info, especially the minor variations
   - Reduces size of output from previous layer, which reduces the number of computations in the previous layer





What a typical CNN architecture looks like ?



How do CNNs go about learning patterns ?

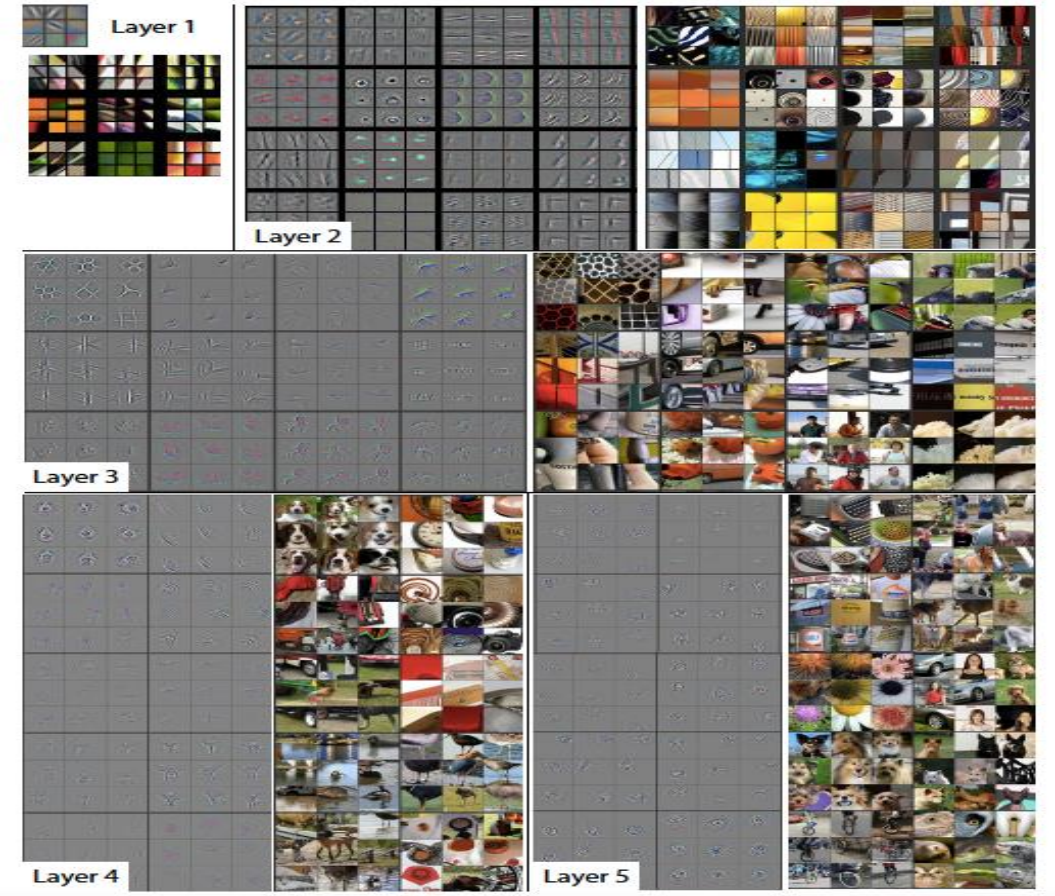
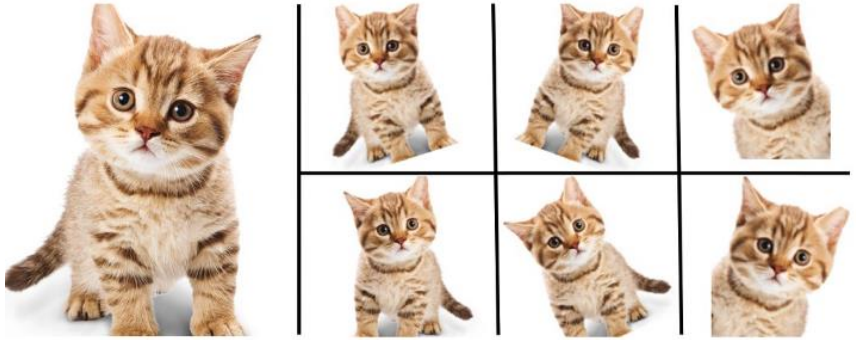


Image Augmentation

Image augmentation is **a technique that is used to artificially expand the data-set**. This is helpful when we are given a data-set with very few data samples. In case of Deep Learning, this situation is bad as the model tends to over-fit when we train it on limited number of data samples.

We use both image filtering and image transformations (affine transformations) to expand the dataset.

Since we know that the weight sharing property of the CNNs make the network rotation, scale and translational invariant we can easily leverage these transformation methods.



Diagnosis of CNNs

1. Use Classification Metrics
2. Observe the high and low probability scores
3. Observe the feature maps
4. Extract the features from the second last layer
5. Perform 2D or 3D plots to visualize them in the x-y(z) coordinate space
    1. Requires dimensionality reduction
6. Advanced visualization methods
    1. Maximal activated image patches
    2. Gradient based class activation maps

Thank you

For any queries drop an email at: quadeershaikh15.8@gmail.com