PadhAI: CNN Architectures

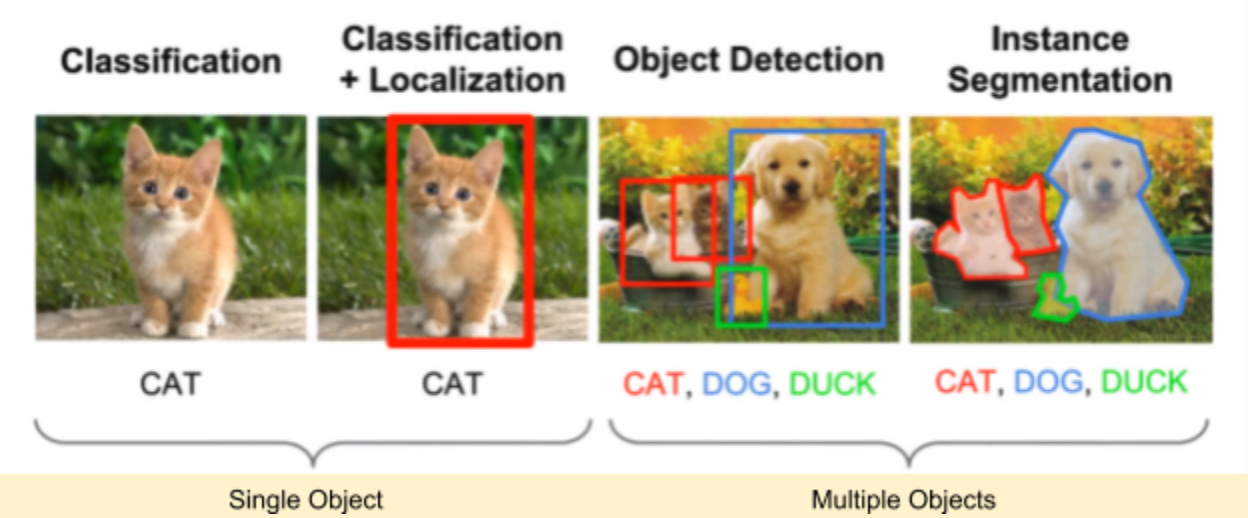
One Fourth Labs

CNN Architectures I

Setting the context

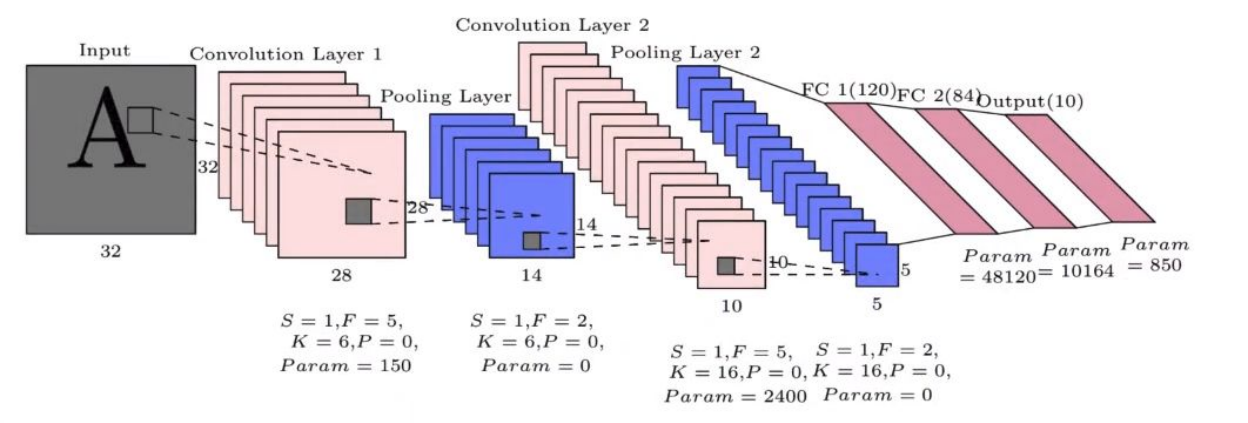
What are CNNs used for?

1. First, let's see **what kind of tasks** are CNNs used for
   a. Consider the following image for an overview of the tasks that CNNs are used for



   b. Using popular datasets like Imagenet, which contains 1000 classes of objects, ranging from vehicles and sceneries, up to the different dog & cat breeds.
   c. **Classification**: In the first image, the task at hand is to correctly predict the class to which the object belongs to. This example uses an "Iconic Photo", i.e. where the class-object of the photo occupies most of the photograph area.
   d. **Classification + Localization**: In the second image, we are predicting the class of the object and also precisely where it is located in the image. Given a starting point outside the object, we detect the width and height of the bounding box enclosing the object. This is both a classification problem and a regression problem.
   e. **Object Detection**: In the third image, we have multiple objects. Therefore, we must correctly detect each object and classify them respectively. It involves multiple Classification + Localization operations on the same image.
   f. **Instance Segmentation**: In the fourth image, we are moving a step further from object detection. Here, we are identifying the precise bounding area around each of the objects present in the image. This is commonly performed in autonomous driving etc, where we have to detect the presence of multiple objects at any given point in time.
   g. These are the four applications that commonly use CNNs. In fact, even our capstone-project of Character-detection and recognition also requires the above mentioned processes.
2. A typical recipe for solving image-related tasks is as follows
   a. First passing the input images through as series of convolutional layer
   b. We obtain a 3D tensor which we flatten to a single dimensional vector
   c. We pass the vector through a number of fully connected layers, which culminate in output prediction
   d. In these cases, we either perform Classification or Regression
3. Now, if we are going to use CNNs for these tasks, we have to make a few choices with regards to the design of the CNN layout.

4. What are some of the decisions that need to be taken?
   a. Let's look at a sample CNN architecture



   b. Some of the factors under our control are as follows:
   c. Number of layers
   d. Number of filters in each layer
   e. Filter Size
   f. Max pooling
5. With the amount of choice we have, designing a CNN architecture could become a very messy process.
6. So a **standard practice is to use tried and tested architectures**.
7. In this chapter, we will be looking at the most popular CNN architectures.

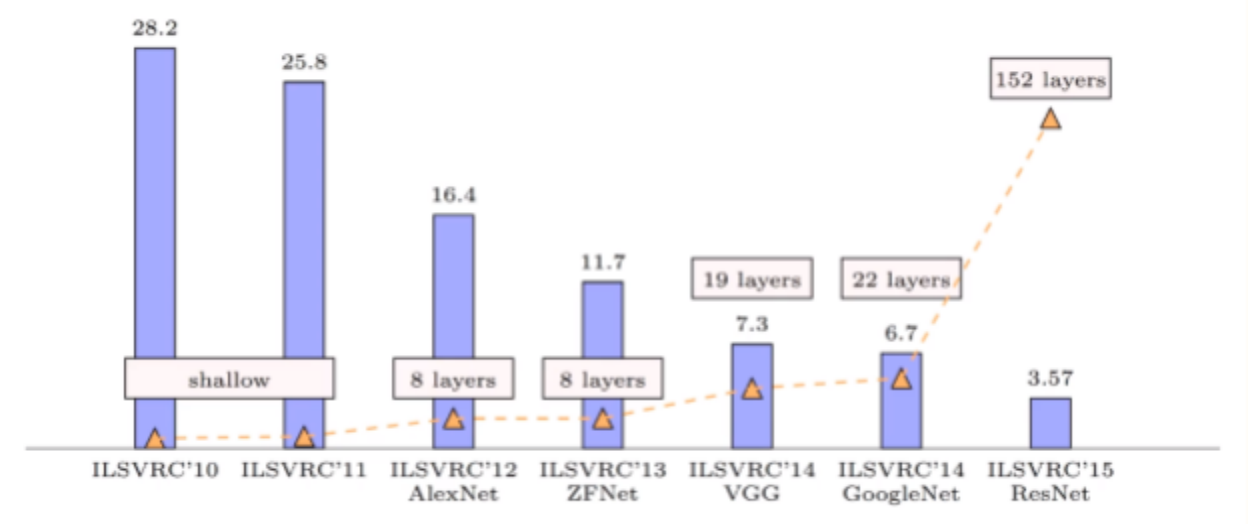
PadhAI: CNN Architectures

One Fourth Labs

The Imagenet Challenge

The Imagenet challenge over the years

1. The Imagenet dataset is a 1000-class, 1,000,000-image dataset (1000 images per class)
2. The **I**magenet **L**arge **S**cale **V**isual **R**ecognition **C**hallenge (**ILSVRC**) or the **Imagenet Challenge** is an annual contest for contender's models to correctly classify the images in the dataset
3. Let us look at the Challenge results between 2010-2015



4. Let us analyse the graph briefly
   a. The metric used to measure performance was Top-5 accuracy. I.e. If any of the top-five predicted class probabilities matched the true class, it was considered correct.
   b. **2010-2011**: This was the pre-DL era, and the Machine Learning models that were submitted had an error between 25-28 %
   c. **2012**: The AlexNet (CNN) architecture smashed existing records with 16.4% error. It kickstarted the Deep Learning Era.
   d. **2013-2014**: Significant improvements were made using models such as ZFNet, VGG and GoogLeNet. Error was brought down to ~6.7%
   e. **2015**: Microsoft's ResNet successfully brought the error down to 3.57% which is lower than the error scored by humans!
   f. One of the reasons for beating the human-error was because some of the classes in the Imagenet Dataset were very fine-grained, i.e. distinguishing between the different dog breeds.
   g. Another interesting point to note is the consistently increasing depth of the Networks used. From shallow networks in the ML era right up to 152 layers in ResNet

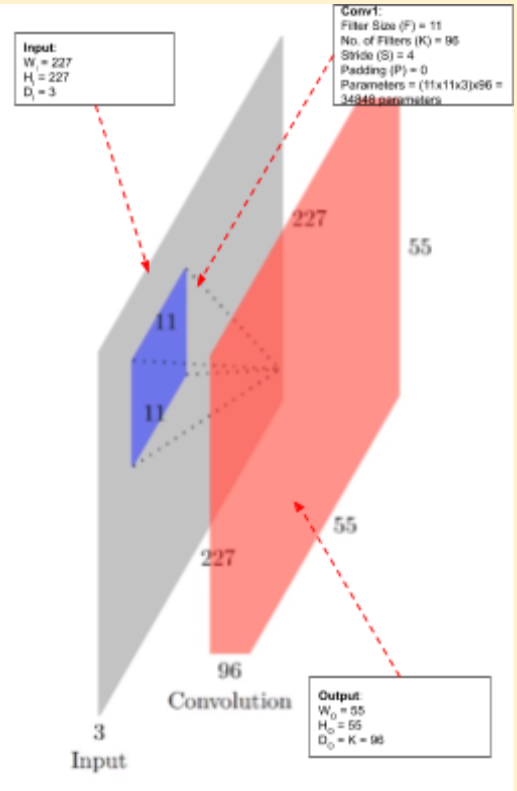
PadhAI: CNN Architectures

One Fourth Labs

Understanding the first layer of AlexNet

Let's break down the first layer of the AlexNet architecture

1. AlexNet was the winning architecture for the 2012 Imagenet Challenge
2. Let us look at the convolutional layer



3. The details are as follows
   a. **Input images**: 227x227x3 (colour images of 227x227 Width x Height)
      i. $W_I = 227$
      ii. $H_I = 227$
      iii. $D_I = 3$
   b. **Filter/Conv1** layer:
      i. Filter Size (F) = 11 (i.e. FxFxD_I or 11x11x3)
      ii. No. of Filters (K) = 96
      iii. Stride (S) = 4
      iv. Padding (P) = 0
      v. Parameters = (11x11x3) x 96 = 34,848
      vi. These values were determined through extensive experimentation
   c. **Output**:
      i. $W_O = \frac{W_I - F + 2P}{S} + 1 = 55$
      ii. $H_O = \frac{H_I - F + 2P}{S} + 1 = 55$
      iii. $D_O = K = 96$

4. This was a standard architecture and can be used for a variety of tasks.

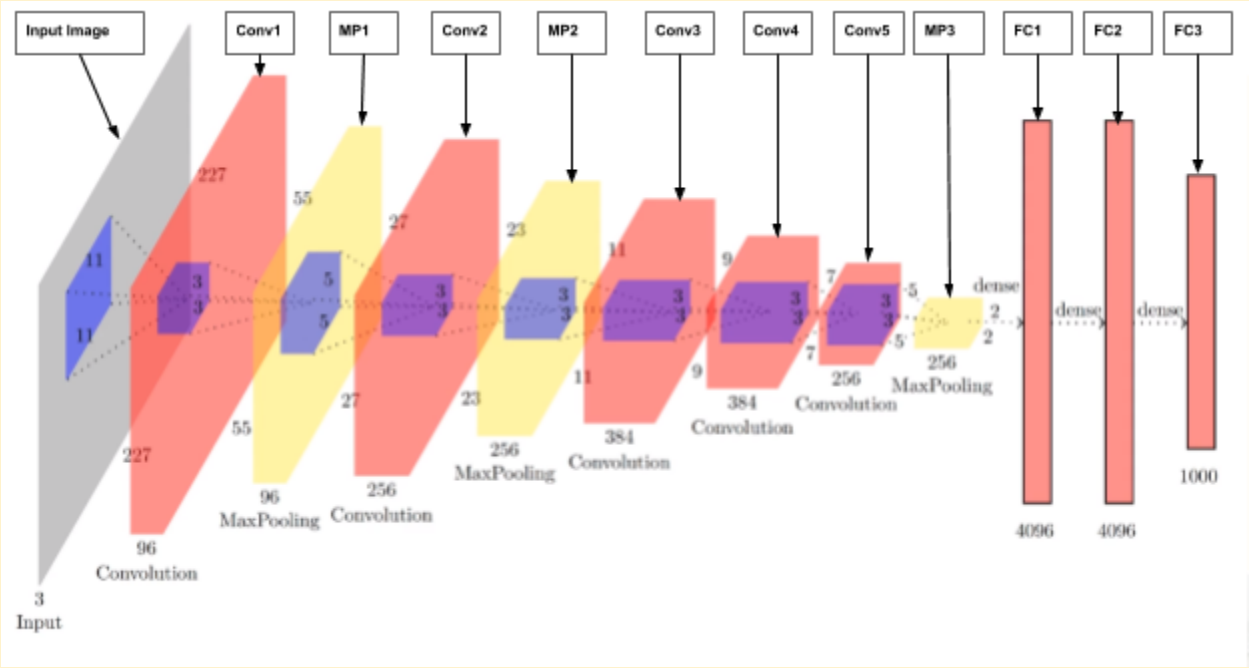
PadhAI: CNN Architectures

One Fourth Labs

Understanding all the layers of AlexNet

Let's now look at the entire AlexNet

1. Before moving into the AlexNet architecture, let us understand why we have decided to use a Convolutional layer instead of a Fully-connected layer
   a. In the convolutional layer, the number of parameters is (11x11x3) x 96 = **34,848**
   b. However, in a FC layer, the number of parameters would be
      i. Input:(227x227x3) x Output:(55x55x96) = **4.49×10^{10}**
   c. Thus, because of sparse-connectivity and weight sharing, we are able to achieve a similar degree of complexity with a significantly smaller number of parameters with a Convolutional layer.
2. Now, let us **break down the entire AlexNet architecture**



3. Let's look at each of the layers in depth
4. **Input Layer**: 227x227x3 (colour images of 227x227 Width x Height)
   a. W_{In} = 227
   b. H_{In} = 227
   c. D_{In} = 3
5. **Convolutional Layer 1**: Input is 227x227x3
   a. Filter Size (**F**) = 11 (11x11x3)
   b. No. of Filters (**K**) = 96
   c. Stride (**S**) = 4
   d. Padding (**P**) = 0
   e. **Parameters** = (11x11x3) x 96 = 34,848
   f. $\mathbf{W_1}$ = 55
   g. $\mathbf{H_1}$ = 55
   h. $\mathbf{D_1}$ = K = 96
   i. **ReLU** Non-linearity function is applied to every 2D area in the output volume.
6. **Max-Pooling Layer 1**: Input is 55x55x96
   a. Filter Size (**F**) = 3 (i.e. 3x3x96)
   b. Stride (**S**) = 4

c. **Parameters** = 0 (no parameters in max pooling)
d. $\mathbf{W_{1m}}$ = 27
e. $\mathbf{H_{1m}}$ = 27
f. $\mathbf{D_{1m}}$ = 96

7. **Convolutional Layer 2**: Input is 27x27x96
   a. Filter Size (**F**) = 5 (5x5x96)
   b. No. of Filters (**K**) = 256
   c. Stride (**S**) = 1
   d. Padding (**P**) = 0
   e. **Parameters** = (5x5x96) x 256 = 614,400
   f. $\mathbf{W_2}$ = 23
   g. $\mathbf{H_2}$ = 23
   h. $\mathbf{D_2}$ = K = 256
   i. **ReLU** Non-linearity function is applied.
8. **Max-Pooling Layer 2**: input is 23x23x256
   a. Filter Size (**F**) = 3 (3x3x256)
   b. Stride (**S**) = 3
   c. **Parameters** = 0
   d. $\mathbf{W_{2m}}$ = 11
   e. $\mathbf{H_{2m}}$ = 11
   f. $\mathbf{D_{2m}}$ = 256
9. **Convolutional Layer 3**: input is 11x11x256
   a. Filter Size (**F**) = 3 (3x3x256)
   b. No. of Filters (**K**) = 384
   c. Stride (**S**) = 1
   d. Padding (**P**) = 0
   e. **Parameters** = (3x3x256) x 384 = 884,736
   f. $\mathbf{W_3}$ = 9
   g. $\mathbf{H_3}$ = 9
   h. $\mathbf{D_3}$ = K = 384
   i. **ReLU** Non-linearity function is applied.
10. **Convolutional Layer 4**: input is 9x9x384
    a. Filter Size (**F**) = 3 (3x3x384)
    b. No. of Filters (**K**) = 384
    c. Stride (**S**) = 1
    d. Padding (**P**) = 0
    e. **Parameters** = (3x3x384) x 384 = 1,327,104
    f. $\mathbf{W_4}$ = 7
    g. $\mathbf{H_4}$ = 7
    h. $\mathbf{D_4}$ = K = 384
    i. **ReLU** Non-linearity function is applied.
11. **Convolutional Layer 5**: input is 7x7x384
    a. Filter Size (**F**) = 3 (3x3x384)
    b. No. of Filters (**K**) = 256
    c. Stride (**S**) = 1
    d. Padding (**P**) = 0
    e. **Parameters** = (3x3x384) x 256 = 884,736
    f. $\mathbf{W_5}$ = 5
    g. $\mathbf{H_5}$ = 5

h. $\mathbf{D_5}$ = K = 256
i. **ReLU** Non-linearity function is applied.

12. **Max-Pooling Layer 3**: input is 5x5x256
    a. Filter Size (**F**) = 3 (3x3x256)
    b. Stride (**S**) = 2
    c. **Parameters** = 0
    d. $\mathbf{W_{5m}}$ = 2
    e. $\mathbf{H_{5m}}$ = 2
    f. $\mathbf{D_{5m}}$ = 256
13. **Fully Connected Layer 1**: input is 2x2x256 = 1024
    a. Number of Neurons = 4096
    b. Parameters = (2x2x256) x 4096 = 4,194,304
14. **Fully Connected Layer 2**: input = 4096
    a. Number of Neurons = 4096
    b. Parameters = 4096 x 4096 = 16,777,216
15. **Fully Connected Layer 3**: input = 4096
    a. Number of Neurons/output-classes = 1000
    b. Parameters = 4096 x 1000 = 4,096,000

16. Totally, there are around 27.55 Million parameters, out of which roughly 25 Million parameters were in the last 3 Fully-connected layers.
17. When counting the total number of layers, we do not include the max-pooling layers as they do not carry weights. Thus, we say that **AlexNet has 8 layers**

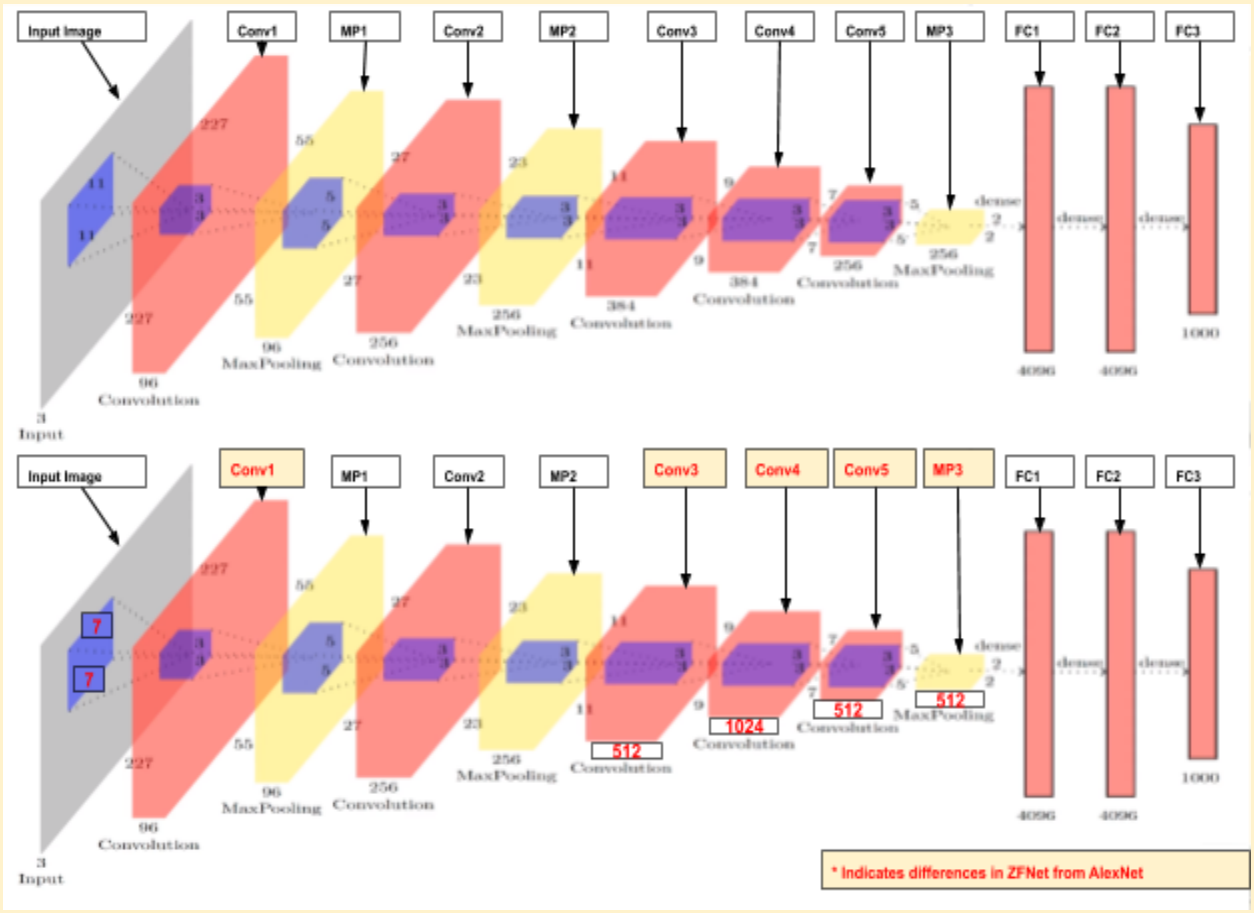
PadhAI: CNN Architectures

One Fourth Labs

ZFNet

Let's now look at the entire AlexNet

1. ZFNet is another 8-layer CNN architecture. Let's understand it better with a side-by-side comparison with AlexNet.



2. ZFNet is largely similar to AlexNet, with the exception of a few of the layers. Let us highlight those differences.
3. **Convolutional Layer 1**: Input is 227x227x3
   a. Filter Size (**F**) = 7 (7x7x3)
   b. No. of Filters (**K**) = 96
   c. Stride (**S**) = 4
   d. Padding (**P**) = 0
   e. **Parameters** = (7x7x3) x 96 = 14,112
   f. $\mathbf{W_1}$ = 55
   g. $\mathbf{H_1}$ = 55
   h. $\mathbf{D_1}$ = K = 96
   i. **ReLU** Non-linearity function is applied to every 2D area in the output volume.

4. **Convolutional Layer 3**: input is 11x11x256
   a. Filter Size (**F**) = 3 (3x3x256)
   b. No. of Filters (**K**) = 512
   c. Stride (**S**) = 1
   d. Padding (**P**) = 0
   e. **Parameters** = (3x3x256) x 512 = 1,179,648
   f. $\mathbf{W_3}$ = 9
   g. $\mathbf{H_3}$ = 9
   h. $\mathbf{D_3}$ = K = 512
   i. **ReLU** Non-linearity function is applied.
5. **Convolutional Layer 4**: input is 9x9x512
   a. Filter Size (**F**) = 3 (3x3x512)
   b. No. of Filters (**K**) = 1024
   c. Stride (**S**) = 1
   d. Padding (**P**) = 0
   e. **Parameters** = (3x3x512) x 1024 = 4,718,592
   f. $\mathbf{W_4}$ = 7
   g. $\mathbf{H_4}$ = 7
   h. $\mathbf{D_4}$ = K = 1024
   i. **ReLU** Non-linearity function is applied.
6. **Convolutional Layer 5**: input is 7x7x1024
   a. Filter Size (**F**) = 3 (3x3x1024)
   b. No. of Filters (**K**) = 512
   c. Stride (**S**) = 1
   d. Padding (**P**) = 0
   e. **Parameters** = (3x3x1024) x 512 = 4,718,592
   f. $\mathbf{W_4}$ = 5
   g. $\mathbf{H_4}$ = 5
   h. $\mathbf{D_4}$ = K = 512
   i. **ReLU** Non-linearity function is applied.
7. **Max-Pooling Layer 3**: input is 5x5x512
   a. Filter Size (**F**) = 3 (3x3x512)
   b. Stride (**S**) = 2
   c. **Parameters** = 0
   d. $\mathbf{W_{2m}}$ = 2
   e. $\mathbf{H_{2m}}$ = 2
   f. $\mathbf{D_{1m}}$ = 512
8. **Fully Connected Layer 1**: input is 2x2x512 = 2048
   a. Number of Neurons = 4096
   b. Parameters = (2x2x512) x 4096 = 8,388,608
9. The **total difference in the number of parameters** ZFNet - AlexNet = 1.45 Million

10. There are other variants of ZFNet where we use a stride of 2 in the first convolutional layer, thereby changing the subsequent layer dimensions.

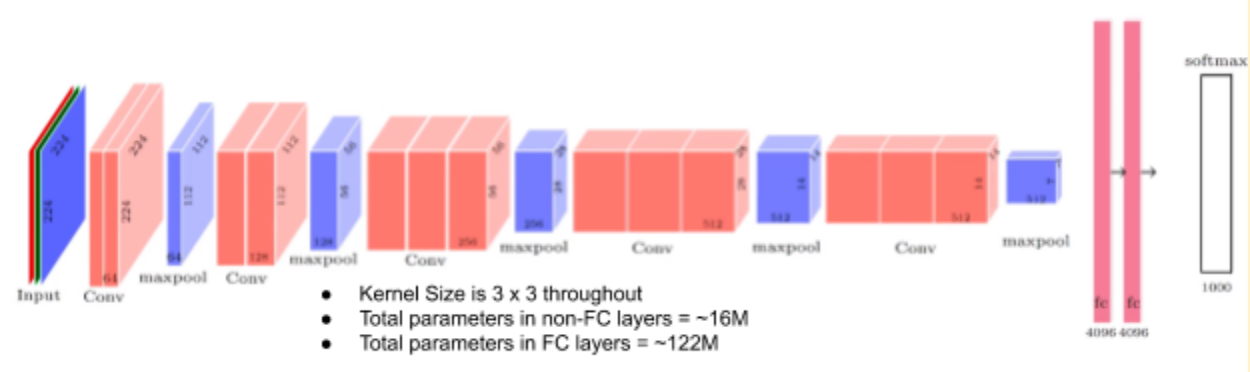
PadhAI: CNN Architectures

One Fourth Labs

VGGNet

Let's look at the architecture of VGGNet

1. During the design of the VGGNet, it was found that alternating convolution & pooling layers were not required. So VGGnet uses multiple of Convolutional layers in sequence with pooling layers in between.
2. Let us break down the VGGNet architecture



3. A few points to note
4. The kernel size 3x3 is maintained throughout the network, only the depth is changed between layers
5. Appropriate padding is provided to maintain the dimensions across the layers
6. **Convolutional Bundle 1**: There are **2 convolutional layers** of size **224x224x64**
7. **Max Pool Layer 1**: The size is **112x112x64**
8. **Convolutional Bundle 2**: There are **2 convolutional layers** of size **112x112x128**
9. **Max Pool Layer 2**: The size is **56x56x128**
10. **Convolutional Bundle 3**: There are **3 convolutional layers** of size **56x56x256**
11. **Max Pool Layer 3**: The size is **28x28x256**
12. **Convolutional Bundle 4**: There are **3 convolutional layers** of size **28x28x512**
13. **Max Pool Layer 4**: The size is **14x14x512**
14. **Convolutional Bundle 5**: There are **3 convolutional layers** of size **14x14x512**
15. **Max Pool Layer 5**: The size is **7x7x512**
16. The number of **parameters in the Non-FC layers is ~16 Million**
17. **FC Layer 1** has **4096** Neurons
18. **FC Layer 2** has **4096** Neurons
19. **FC Layer 3** is a softmax with **1000** Neurons/Output-classes
20. The number of **parameters in the FC layers is ~122 Million** (Most in FC Layer 1: ~102 Million)
21. Though the number of parameters in this network seems very large, it would have been exponentially larger if we had chosen an entirely Fully-Connected network.
22. The above shown VGGNet is a 16 layer network called VGG16. There are also other versions like the 19 layered VGG19

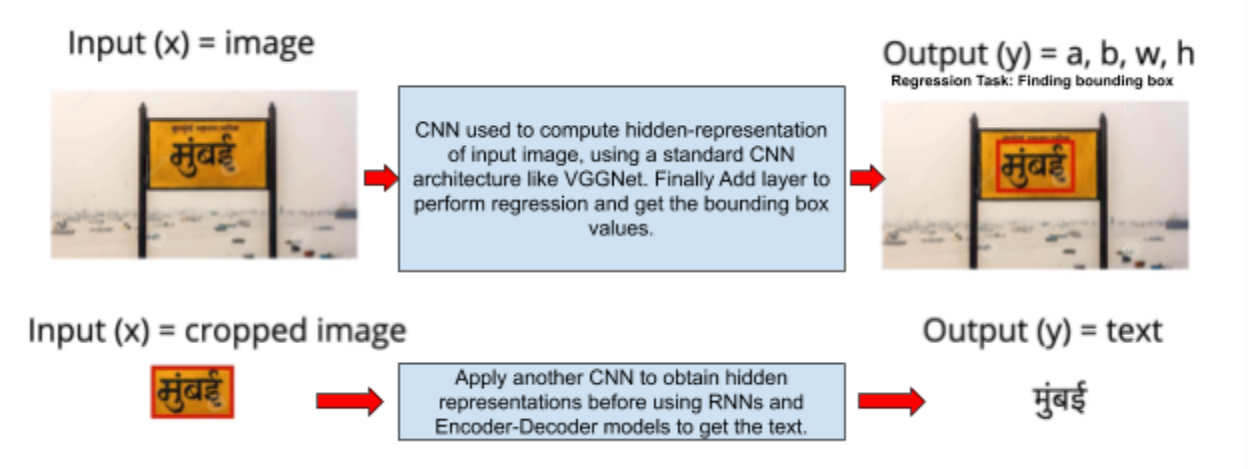
PadhAI: CNN Architectures

One Fourth Labs

Summary

Connecting this to the capstone project

1. Let's see how CNNs come into play for our Signboard translation capstone project



2. The above diagram shows how we use CNNs in our capstone project.