PadhAI: CNN Architectures

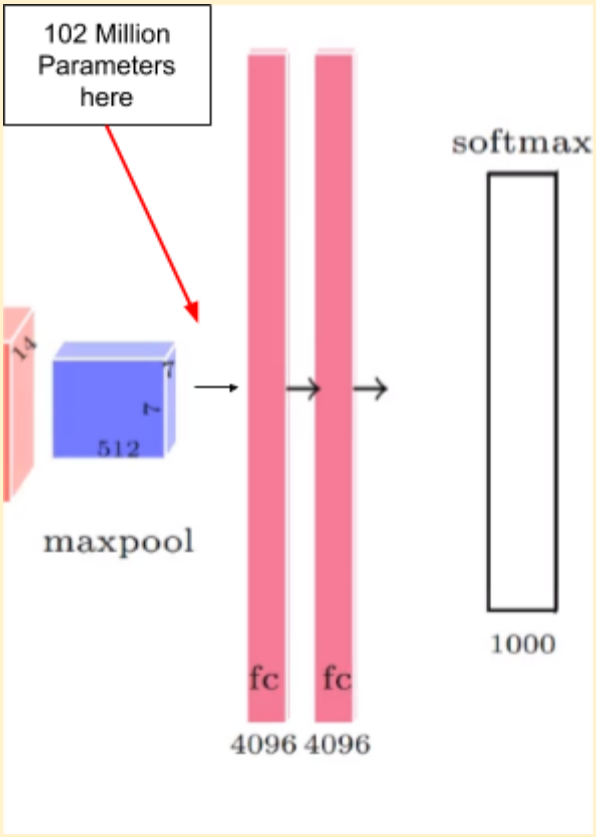
One Fourth Labs

CNN Architectures II

Setting the context

How do we address what's been bothering us so far with CNNs

1. To pick up from VGGNet, can we do something to reduce the huge number of parameters incurred between the non-FC Layer and the FC Layer.



2. Another question we can ask is why we must stop at 16 layers, can we not go deeper.
3. So by combining the above two points, we can make the network deeper to reduce the number of parameters at the interface between the non-FC and the FC layers.
4. We must also make sure that the number of computations (Sliding a filter across the input) is not very high.
5. Our problem points can be summarised as follows
   a. Increase choice of filters
   b. Reduce number of parameters
   c. Reduce number of computations
   d. Make a deeper network

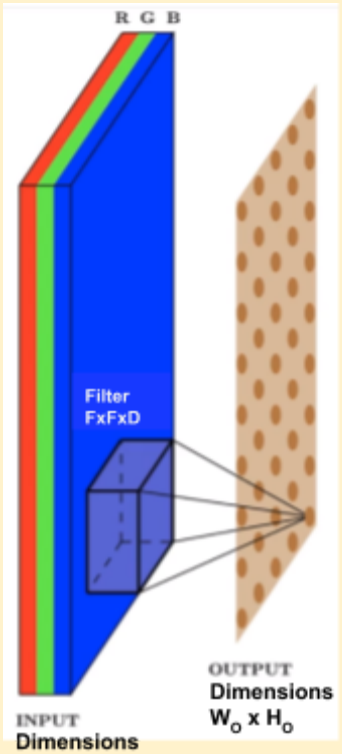
PadhAI: CNN Architectures

One Fourth Labs

Number of computations in a convolution layer

Let's see how many computations are needed in a CNN.

1. We will be looking at the GoogLeNet architecture as an improvement to the VGGNet based on the points discussed in the previous section.
2. However, before that, we must look at two key concepts in the GoogLeNet layout: **1x1 convolution** and an **interesting way to perform max-pooling**.
3. To approach these two, we first need to see how many computations are needed in one convolutional layer



Assume S = 1 and we have used appropriate padding so that W_O = W_I = W and H_O = H_I = H

   a. **Input dimensions**: W_I x H_I x D_I
   b. **Filter size**: F x F x D_I
   c. **Output dimensions**: W_O x H_O
   d. Stride = 1 and appropriate padding so that **W_O = W_I = W and H_O = H_I = H**
4. To calculate the number of computations:
   a. For every pixel of interest, for D layers, we perform FxFxD computations
   b. So for an output area of WxH, we perform (WxH) x (FxFxD) computations
   c. From the previous point, we can observe that the Depth of the output layer will be very large if there is a large number of filters applied on the input layer, as each filter generates a 2D area of unit depth.
   d. So if we use a **large number of filters**, the **output volume will be very deep**, subsequently **increasing the number of computations in the next layer's calculation** (Due to high D value).
   e. We can also try controlling W and H, but they can be more easily regulated using max-pooling. However, depth is directly related to the number of filters used.

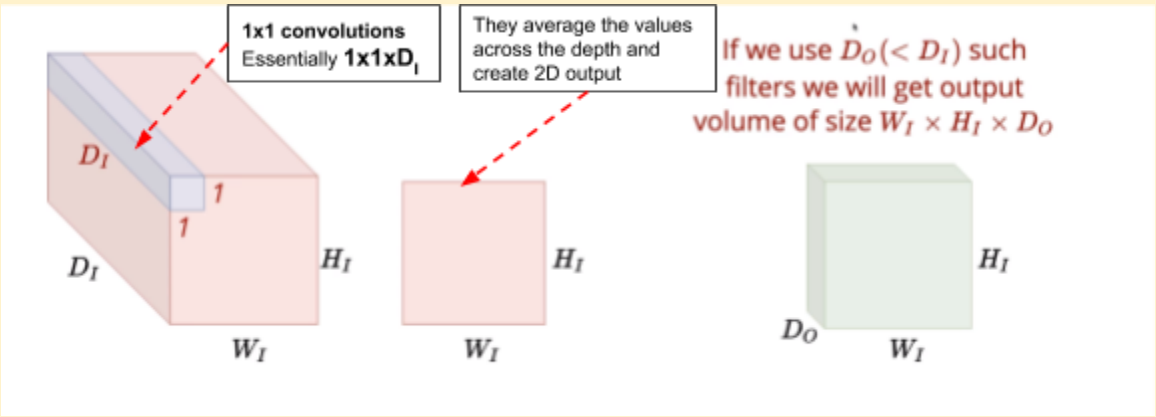
PadhAI: CNN Architectures

One Fourth Labs

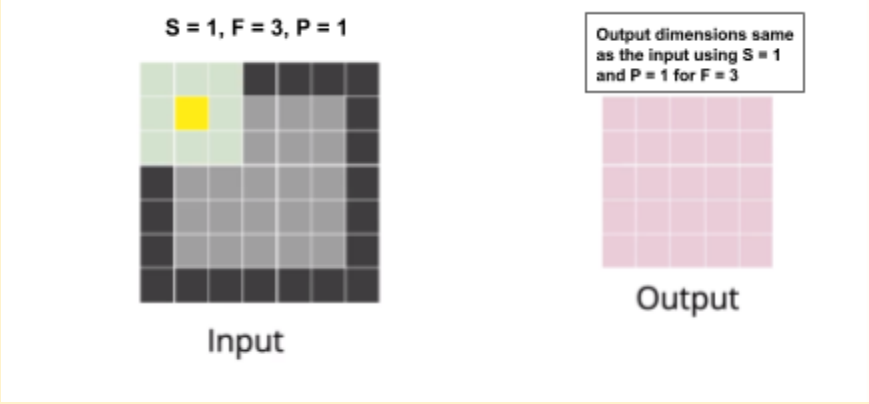
1x1 Convolutions

What is a 1x1 convolution used for?

1. We've mostly worked with 3x3 convolutions so far, i.e. a grid containing 3 rows, 3 columns with 9 cells in total.
2. The result of a single 3x3 operation is the weighted average of all the points in the grid, applied to our selected pixel.
3. Now, let us look at a 1x1 convolution



4. A 1x1 kernel takes a neighborhood of 1 row and 1 column, which is essentially the pixel itself. Since the kernel is of size $1x1xD_I$, it computes the weighted average of all the pixels across the input depth D_I
5. Here, the Input is 3D, the filter is 3D but the operation is 2D, as we are only moving horizontally and vertically. The 3D volume is compressed to a 2D area.
6. If we were to use D_O number of filters, where $(D_O < D_I)$, we will get an output of $W_I \times H_I \times D_O$. Each of the 1x1 kernels will give one 2D output and D_O such kernels gives us an output volume of the dimensions $W_I \times H_I \times D_O$.
7. If D_O is much smaller than D_I, we effectively shrink the input volume while still effectively retaining the depth information (Due to averaging across depth).
8. Now, this output behaves as an input to the next layer, resulting in a much smaller number of computations due to smaller depth.
9. In a nutshell, 1x1 filters are used to compress input volumes across their depth to get a smaller output volume of same Width and Height.
10. Another operation we need to look at is Max Pooling. We usually perform Max-pooling with a Stride=2, resulting in halving the input dimensions. However, we can also perform it with a stride of 1. With S = 1 and appropriate padding, we can preserve input dimensions.



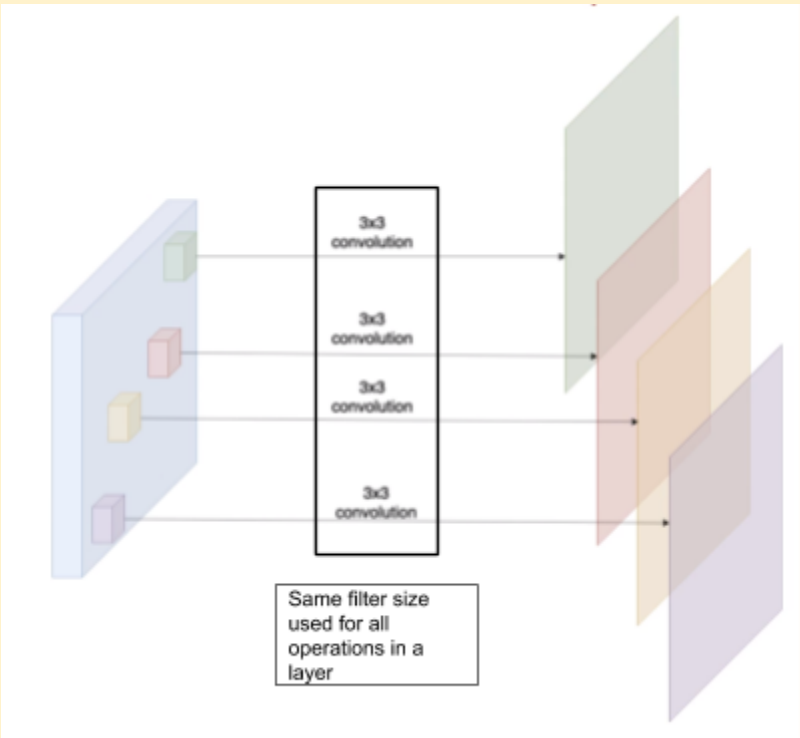
PadhAI: CNN Architectures

One Fourth Labs

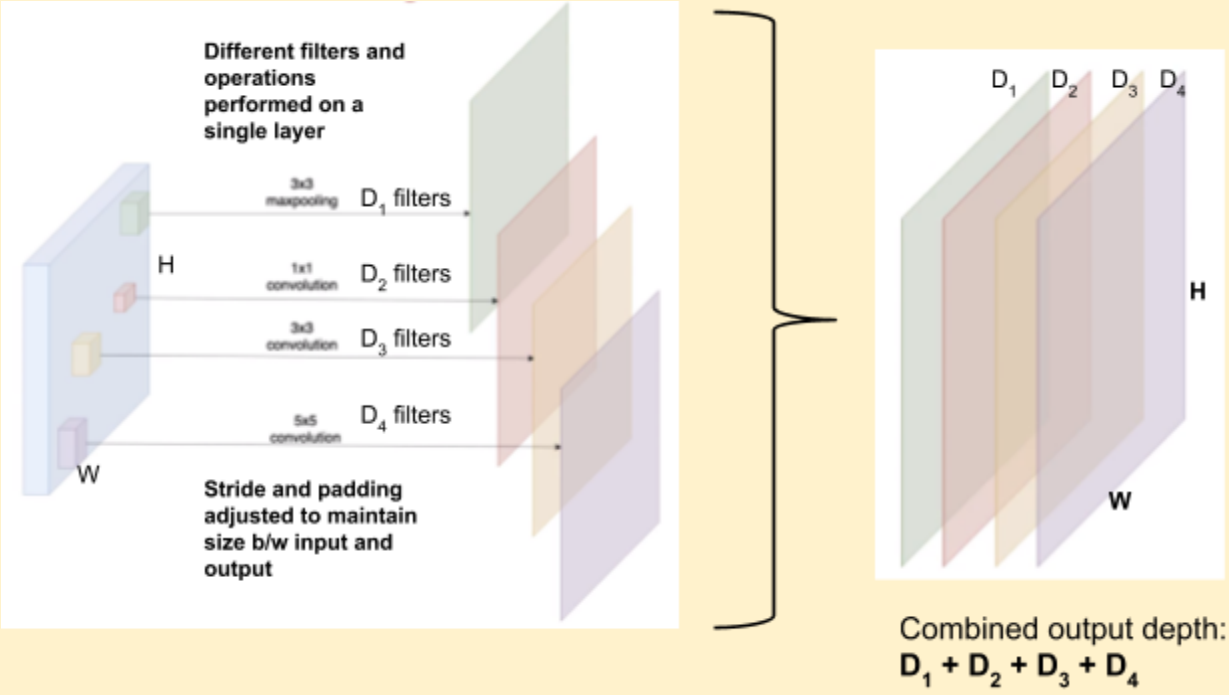
The Intuition behind GoogLeNet

What is the intuition behind GoogLeNet?

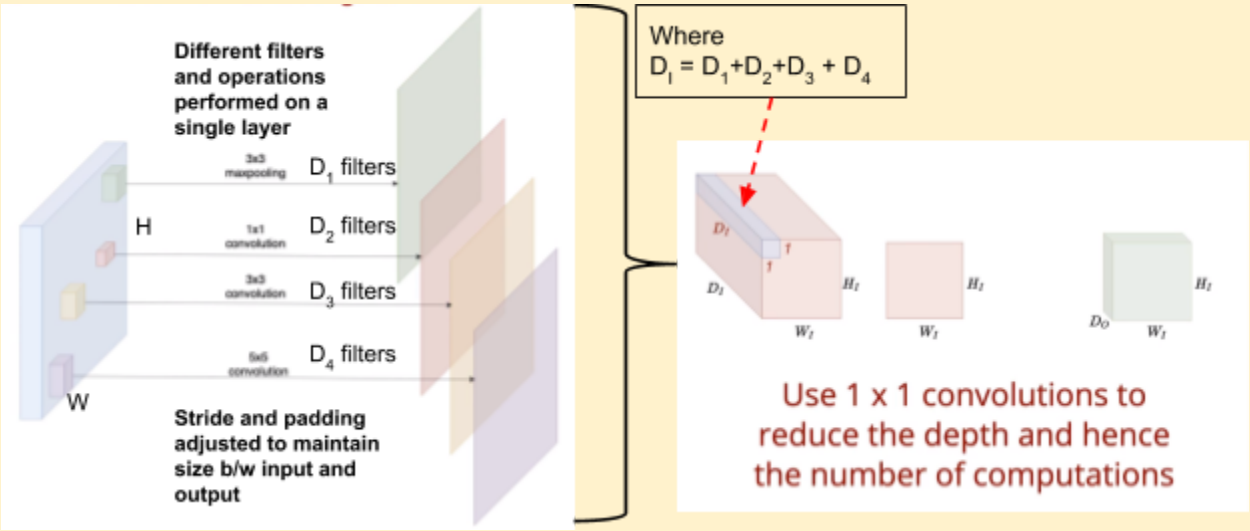
1. One point to note in the architectures used thus far, is that we must always make a choice of a particular filter size for any given layer. For eg, in VGGNet, all filters used were 3x3
2. Another point is the interspersing of Max-pooling layers between convolutional layers. How do we decide the arrangement to follow?



3. In GoogLeNet, the choice was eliminated, instead we are able to apply all our operations whatever combination of filter size and max-pooling/non that we'd like.



4. In GoogLeNet, we can apply multiple filters of varying size to perform either convolutional or max-pooling operations
5. Through experimentation with the older architectures, we have found that when there are multiple convolutional operations in a layer, we needn't use larger filter sizes.
6. Therefore, 5x5 filters are usually the upper limit of size.
7. One constraint is that for each operation in a layer, appropriate padding and stride must be taken so as to preserve the width and height between the input and the output
8. The Number of filter for each operation can vary, (D_0 to D_3 etc)
9. In the output volume, the total depth is the combined depth of the individual volumes from each of the operations. $D = D_0 + D_1 + D_2 + D_3$
10. The problem with combining the depths is that it has the potential to become very large, thus drastically increasing the number of computations.
11. We can mitigate this problem by performing 1x1 convolutions



12. By performing 1x1 convolutions, we can reduce the depth of the output volume, thereby reducing the number of computations to be performed in the subsequent layers.

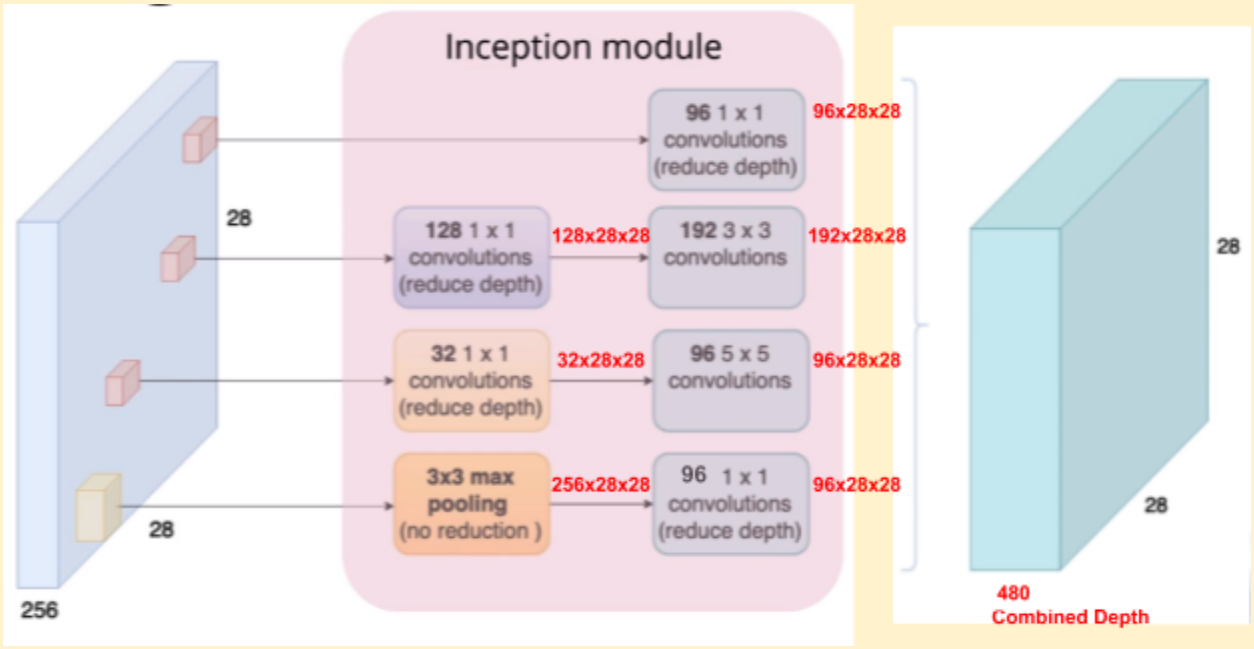
PadhAI: CNN Architectures

One Fourth Labs

The Inception Module

What is the Inception Module?

1. GoogLeNet is also called Inception net. It is made up of multiple modular operation-blocks known as inception modules.
2. Let us break down a single inception module



3. Here, we can see the sequence of operations performed in this inception module.
4. There are a few operations that are a blueprint of inception modules
5. Direct 1x1 convolutions
6. 1x1 convolutions followed by 3x3 and 5x5 convolutions
7. 3x3 convolutions followed by 1x1 convolutions
8. Here, The number of filters can change but the same blueprint repeats itself throughout the network.
9. Summing the output of all of these, we get the output volume as shown in the figure.

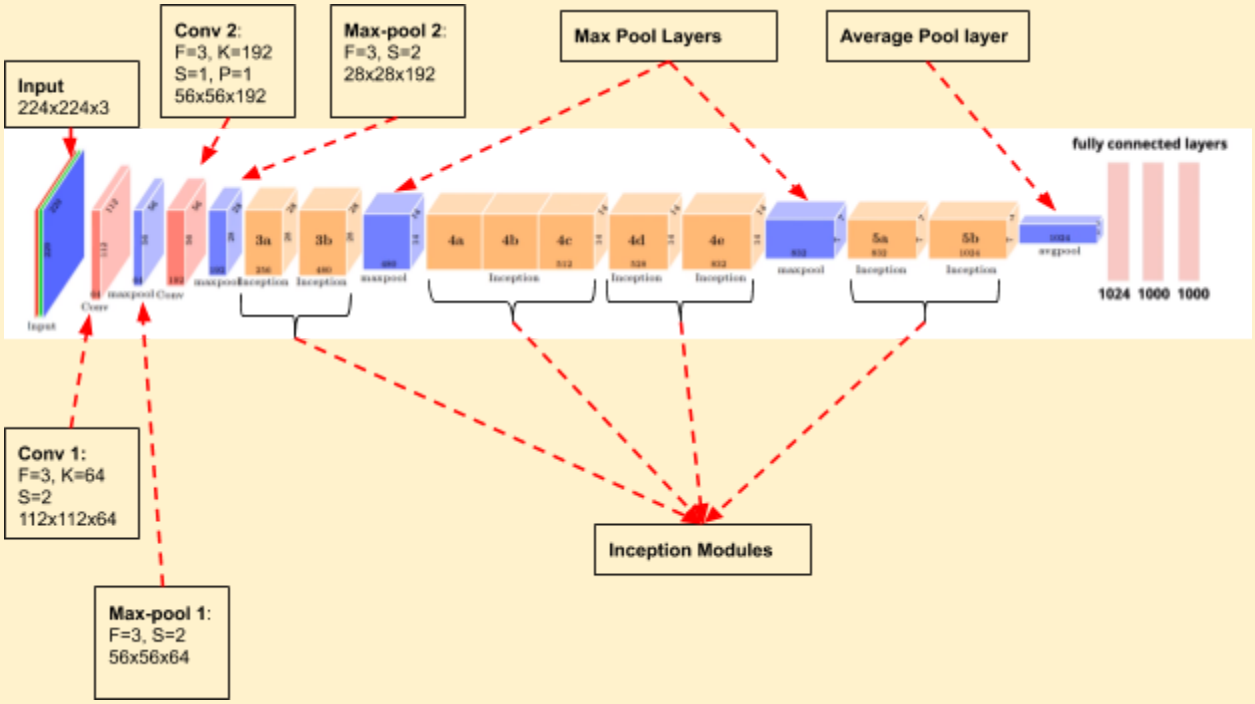
PadhAI: CNN Architectures

One Fourth Labs

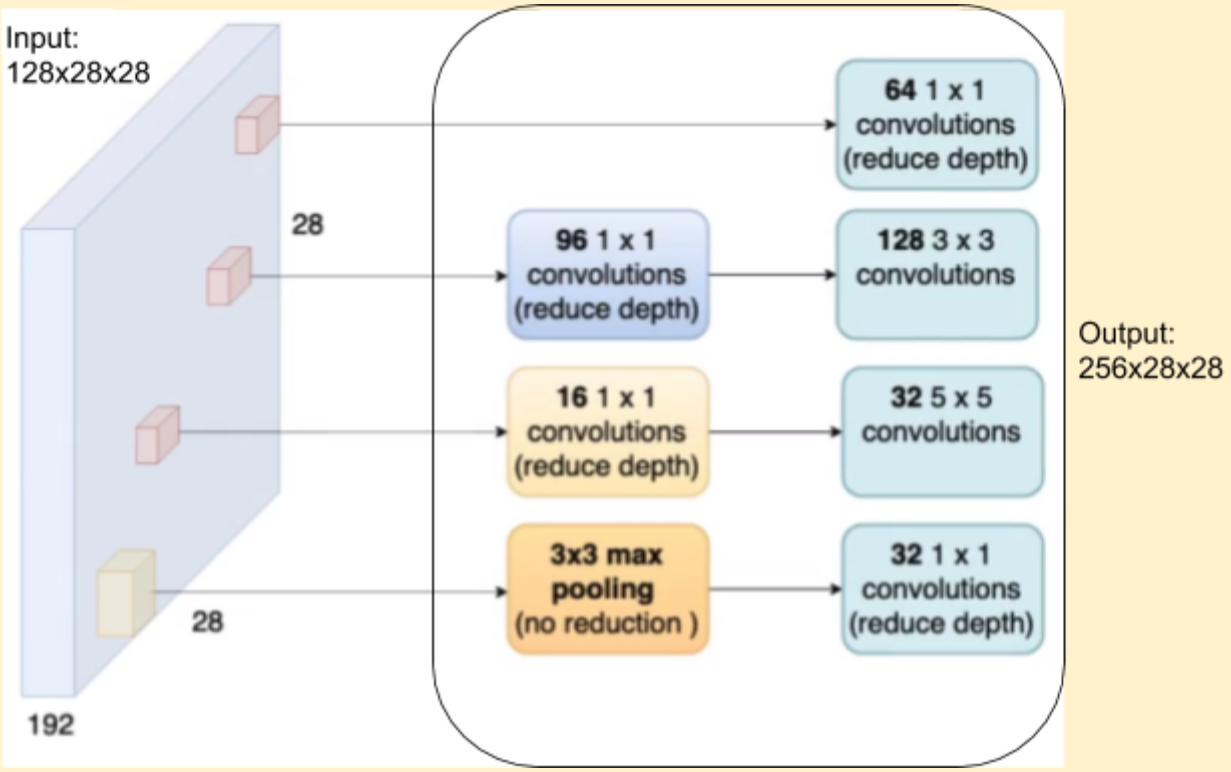
The GoogLeNet Architecture

What does the full network look like?

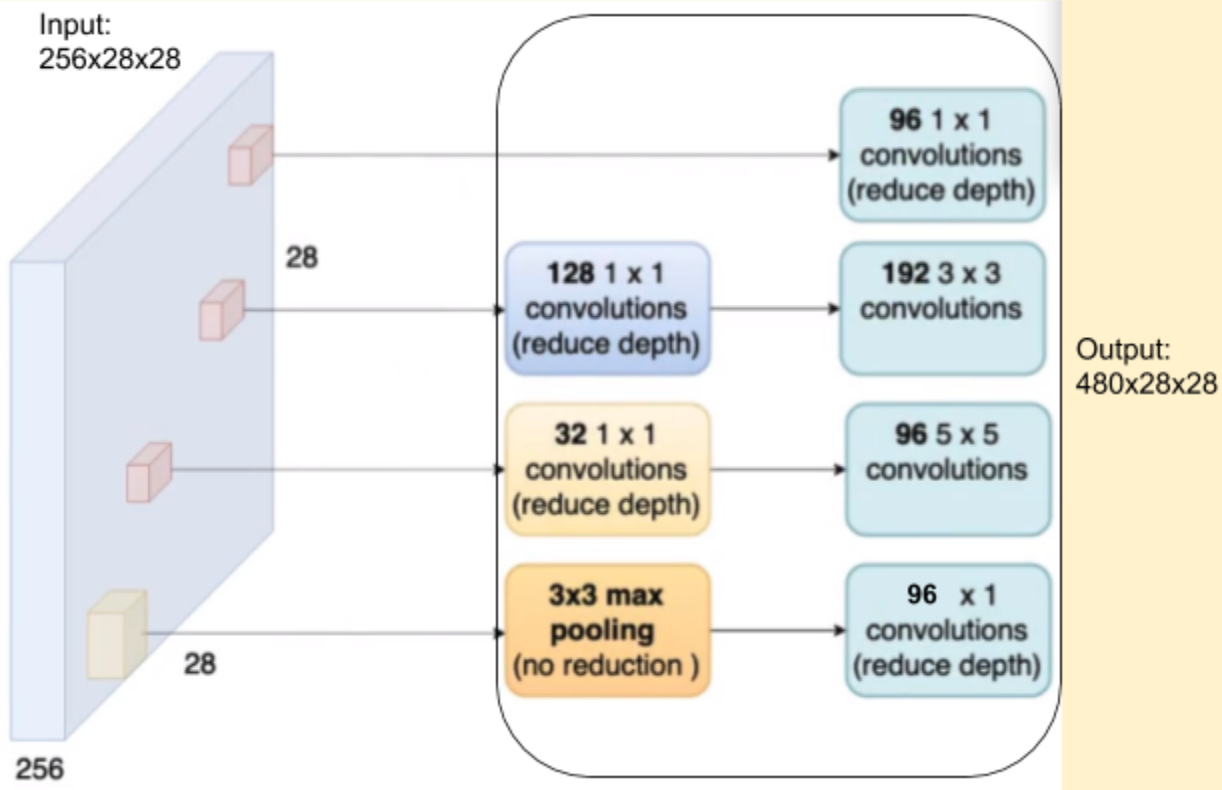
1. Let's take a look at the entire GoogLeNet architecture



2. Up till the second max-pooling layer, the architecture is similar to what we've seen in earlier configurations. Post that, we begin moving into the Inception modules.
3. **Inception Module 1:**



4. **Inception Module 2**:

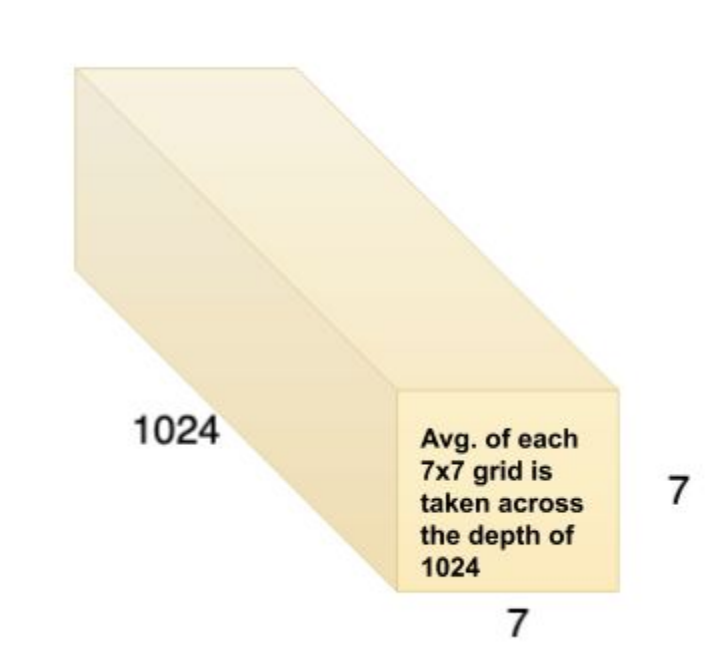


5. Then there is a **Max-pooling layer which reduces the Dimensions by Half (480x14x14)**. This Max-pooling layer is used because the Max-pooling layers in the Inception modules do not reduce the dimensions.
6. **Inception Module 3,4 & 5**:
   a. **Input at Inception Module 3** is: 480x14x14
   b. **Output at Inception Module 5** is: 512x14x14
7. **Inception Module 6**:
   a. **Input**: 512x14x14
   b. **Output**: 528x14x14
8. **Inception Module 7**:
   a. **Input**: 528x14x14
   b. **Output**: 832x14x14
9. Then there is a **Max-pooling layer which reduces the Dimensions by Half (832x7x7)**.
10. **Inception Module 8**:
    a. **Input**: 832x7x7
    b. **Output**: 832x7x7
11. **Inception Module 9**:
    a. **Input**: 832x7x7
    b. **Output**: 1024x7x7
12. **Average Pool layer is used to reduce the output from Inception Module 9**, thereby reducing the number of parameters between the non-FC and FC layer interface.
13. Each Inception Module counts as 2 layers, therefore we have more than 20 layers in GoogLeNet.

Average Pooling

How does average pooling reduce the output size?

1. At the final Inception Module, we have an output dimension of 1024x7x7. If this was to directly interface with a Fully-Connected layer with a 1000 Neurons, we would get ~50 Million parameters
2. To reduce this number, Google added another layer which performs Average-pooling



3. By taking the average value of each 7x7 slice to get a 1x1 value across the depth, we are left with a 1024x1x1 vector.
4. This vector of 1024 values interfaces with the Fully connected layer with a 1000 Neurons.
5. This gives us ~1 Million parameters, as opposed to the earlier seen 50 Million parameters.

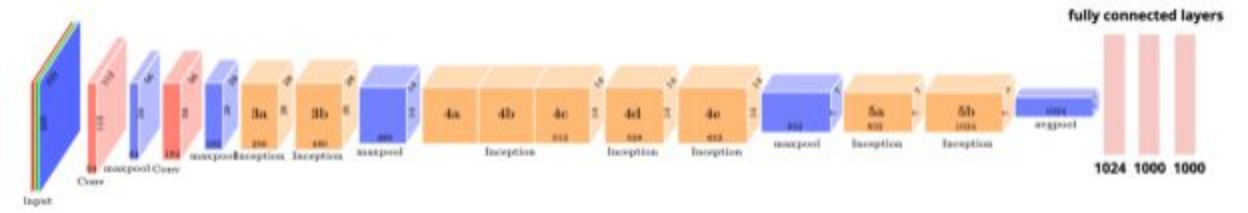
PadhAI: CNN Architectures

One Fourth Labs

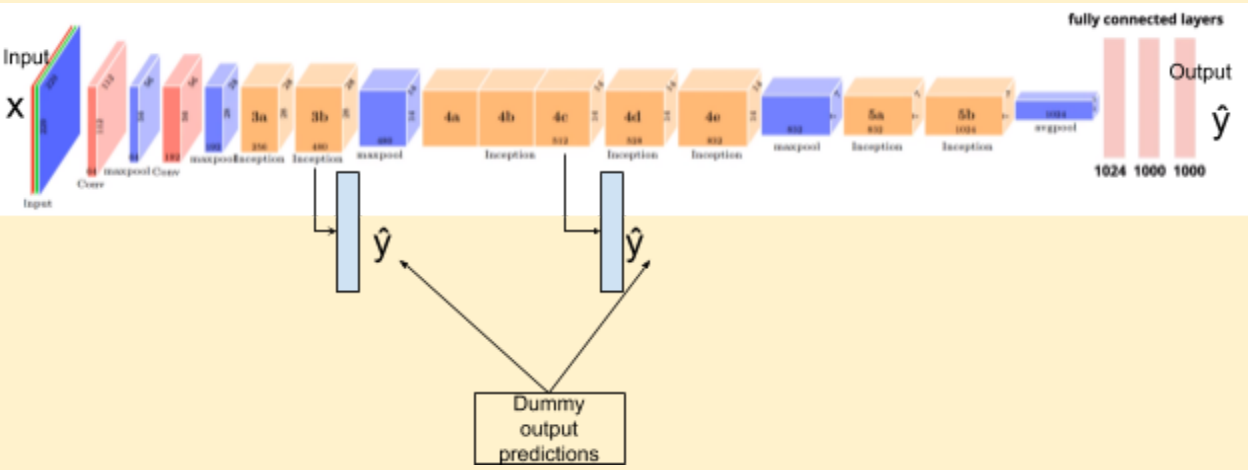
Auxiliary Loss for training a deep network

Can auxiliary loss help to train the network better?

1. Let's look at how GoogLeNet responds to the 4 problem points from the previous CNN architectures



2. **Increase choice of filters**: Parallel convolutions/max-pooling
3. **Reduce number of parameters**: Average Pooling
4. **Reduce number of computations**: 1x1 convolutions
5. **Make a deeper network**: Has 22 layers as opposed to VGG19's 19 layers.
6. Now, since it is a very deep network, there is a possibility for vanishing gradients to occur when backpropagating the Loss. This is mitigated using a technique called Auxiliary Loss



7. In addition to the final output prediction, we are also trying to make partial predictions from the above specified regions in the network.
8. We compute the loss at the final prediction and at both the dummy predictions.
9. Now we can backpropagate from the final loss or from the dummy-losses obtained, thereby shortening the effective depth of the network and lowering the chance of vanishing gradients occurring.
10. Some interesting points to note about GoogLeNet
    a. 12x less parameters than AlexNet
    b. 2x more computations than AlexNet
    c. Improved performance on ImageNet

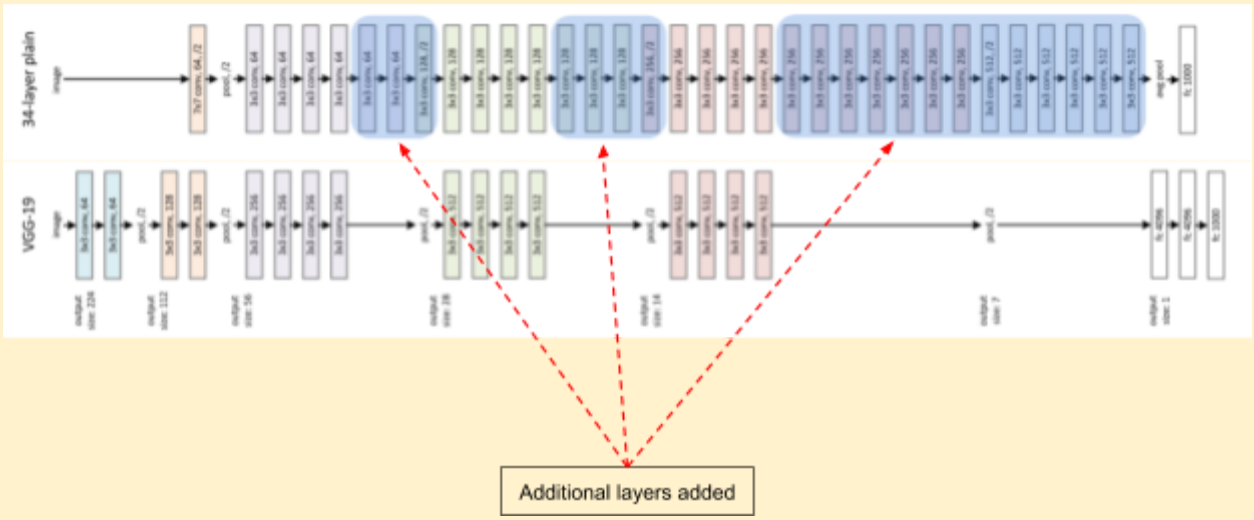
PadhAI: CNN Architectures

One Fourth Labs

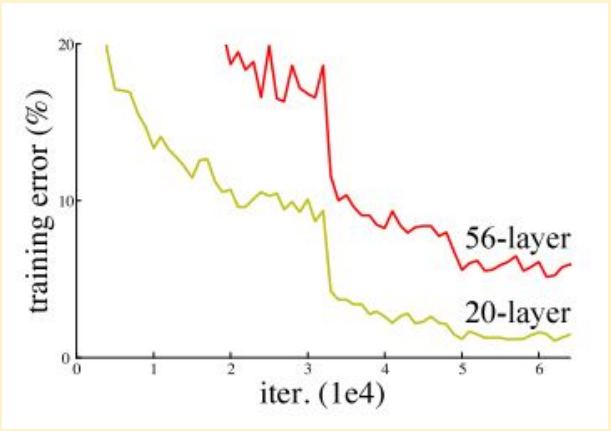
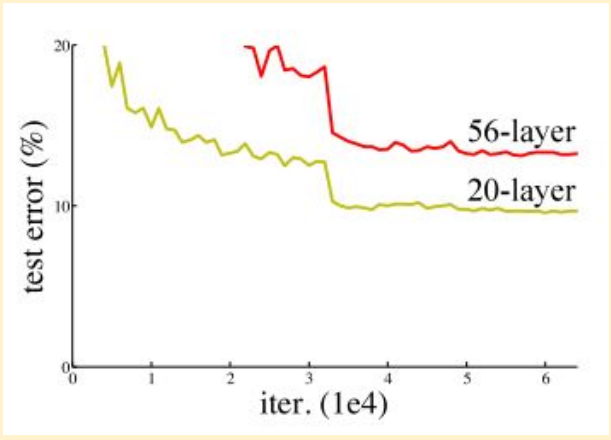
ResNet

What happens if you increase the depth of the network?

1. Let us consider the basic architecture of the VGG-19 network when compared to another 34 layer network architecture

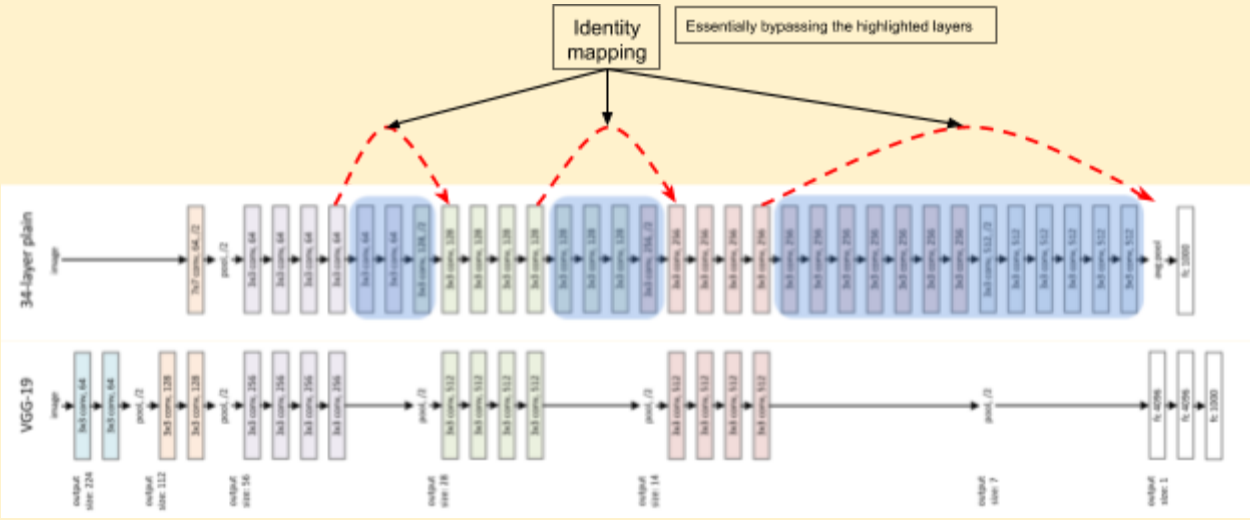


2. The train/test error curves were plotted for some other 20-layer and 56-layer networks

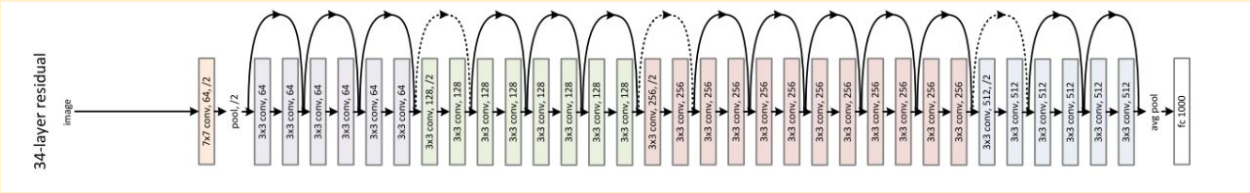
Training curves	Test Curves
	
Unexpectedly, the 56-layer had a higher train error than the 20-layer. It was expected to have overfit the training data, thereby having a lower training error. It was hypothesised that the gradients were not able to flow well through this deeper network.	Here, as predicted, the 56-layer performed worse than the 20-layer due to overfitting.

3. Now, when comparing the 19 and 34-layer networks, we see that at the very least, the 34-layer network should be able to match the performance of the 19-layer network.

4. Matching the performance could be done by bypassing the additional layers in the 34-layer network by using Identity Mapping.
5. Identity mapping refers to learning filter values such that the output is preserved identical across a finite number of layers, till it reaches the target layer. Basically cloning the layer output till required.



6. However, it wasn't able to match the 19-layer's error. This implies that the information from the input is getting highly morphed and by the time we reach the output, it is highly transformed.
7. A simple solution would be to keep passing the input information repeatedly in stages.
8. To attempt this, they tried the Residual Network or the ResNet



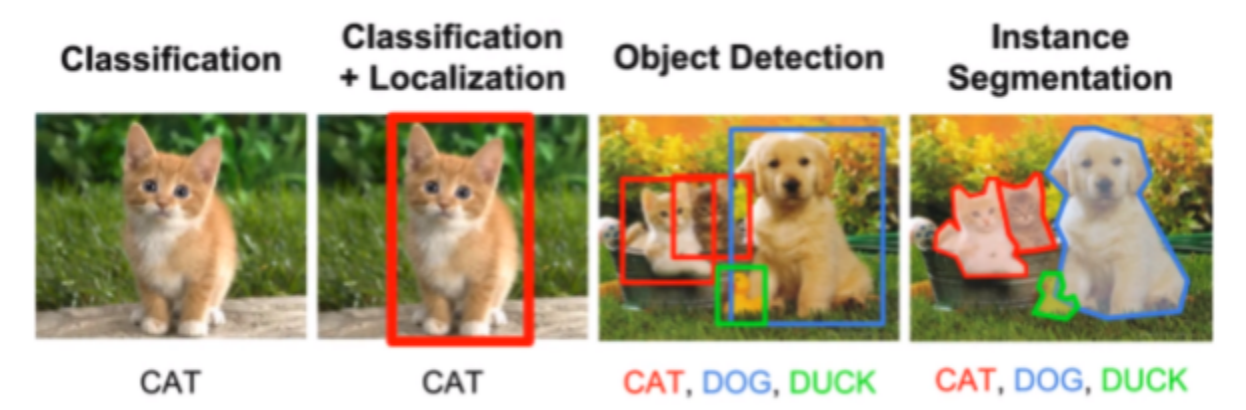
9. In the ResNet, **every two layers, we pass the input given to the first layer along with the output obtained at the second layer.**
   a. Input: x_1
   b. Output: $x_2 = f(x_1) + x_1$
   c. Output after two layers: $x_3 = f(x_2) + x_2$
10. This helped the gradients to flow back better and the training to improve
11. It is called a Residual Network because at every stage, there is a residue of the input which is passed once again with the output.
12. Using this technology, they were able to train very deep Neural networks of up to 151 layers.

PadhAI: CNN Architectures

One Fourth Labs

---

13. The ResNet showed remarkable performance among the various tasks



14. It was the winner among the 4 main tasks across the following datasets
    a. ImageNet Classification **(ResNet-151)**
    b. ImageNet Localization **(ResNet-101)**
    c. ImageNet Detection **(ResNet-101)**
    d. Coco Detection **(ResNet-101)**
    e. Coco Segmentation **(ResNet-101)**
15. Some of the popular ResNets are **(ResNet-51, ResNet-101 and ResNet-151)**