In The Name of God

Sharif University Of Technology

EE Department

Deep Learning

CA2 Reprot

Dr.Fatemizadh

Mahyar Jafari Nodeh

98101352

December 20, 2021

The Inception network was an important milestone in the development of CNN classifiers. Prior to its inception (pun intended), most popular CNNs just stacked convolution layers deeper and deeper, hoping to get better performance.
The Inception network on the other hand, was complex (heavily engineered). It used a lot of tricks to push performance; both in terms of speed and accuracy. Its constant evolution lead to the creation of several versions of the network. The popular versions are as follows:

- Inception v1.
- Inception v2 and Inception v3.
- Inception v4 and Inception-ResNet.

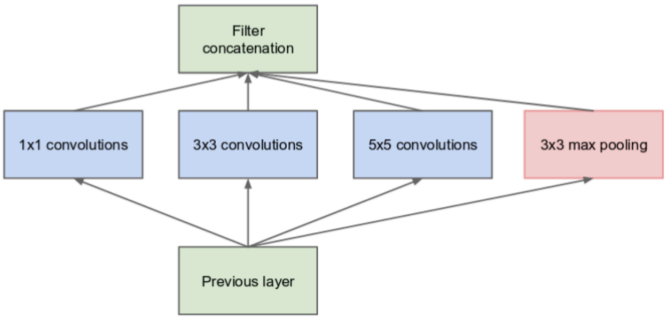
Each version is an iterative improvement over the previous one.

1 Architucture of InceptionV3

in this hw we are surveying Inception version 3.
This is where it all started. Let us analyze what problem it was purported to solve, and how it solved it. (Paper)
The Premise:

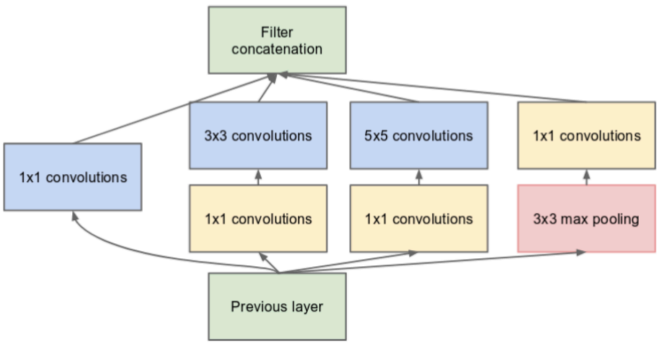
- Salient parts in the image can have extremely large variation in size. For instance, an image with a dog can be either of the following, as shown below. The area occupied by the dog is different in each image.
- Because of this huge variation in the location of the information, choosing the right kernel size for the convolution operation becomes tough. A larger kernel is preferred for information that is distributed more globally, and a smaller kernel is preferred for information that is distributed more locally.
- Very deep networks are prone to overfitting. It also hard to pass gradient updates through the entire network. Naively stacking large convolution operations is computationally expensive.

The Solution:
Why not have filters with multiple sizes operate on the same level? The network essentially would get a bit “wider” rather than “deeper”. The authors designed the inception module to reflect the same.
The below image is the “naive” inception module. It performs convolution on an input, with 3 different sizes of filters (1x1, 3x3, 5x5). Additionally, max pooling is also performed. The outputs are concatenated and sent to the next inception module.
As stated before, deep neural networks are computationally expensive. To make it cheaper, the authors limit the number of input channels by adding an extra 1x1 convolution before the 3x3 and 5x5 convolutions. Though adding an extra operation may seem counterintuitive, 1x1



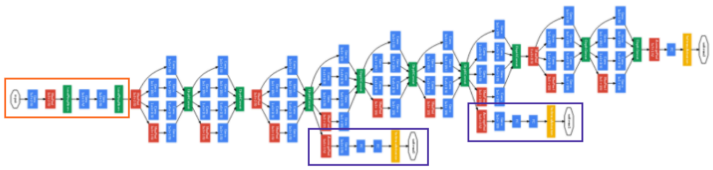
(a) Inception module, naïve version

convolutions are far more cheaper than 5x5 convolutions, and the reduced number of input channels also help. Do note that however, the 1x1 convolution is introduced after the max pooling layer, rather than before.



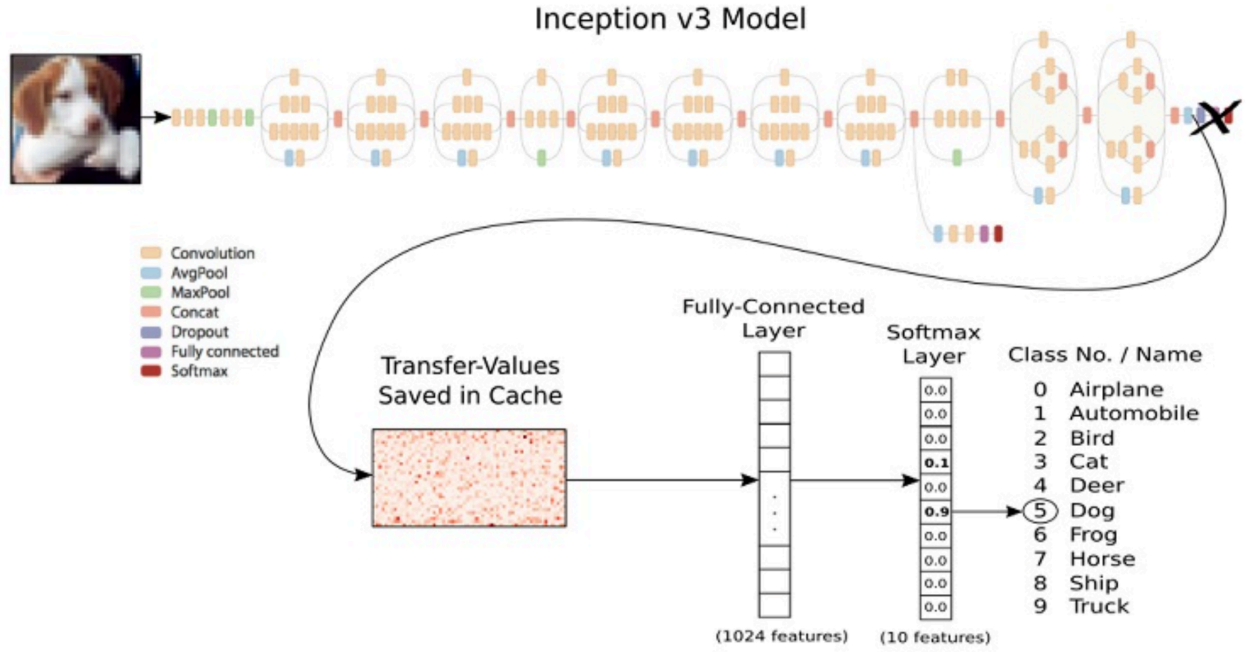
(b) Inception module with dimension reductions

Using the dimension reduced inception module, a neural network architecture was built. This was popularly known as GoogLeNet (Inception v1). The architecture is shown below:



GoogLeNet has 9 such inception modules stacked linearly. It is 22 layers deep (27, including the pooling layers). It uses global average pooling at the end of the last inception module.
Needless to say, it is a pretty deep classifier. As with any very deep network, it is subject to the vanishing gradient problem.
To prevent the middle part of the network from "dying out", the authors introduced two auxiliary classifiers (The purple boxes in the image). They essentially applied softmax to the outputs of two of the inception modules, and computed an auxiliary loss over the same labels. The total loss function is a weighted sum of the auxiliary loss and the real loss. Weight value used in the paper was 0.3 for each auxiliary loss.

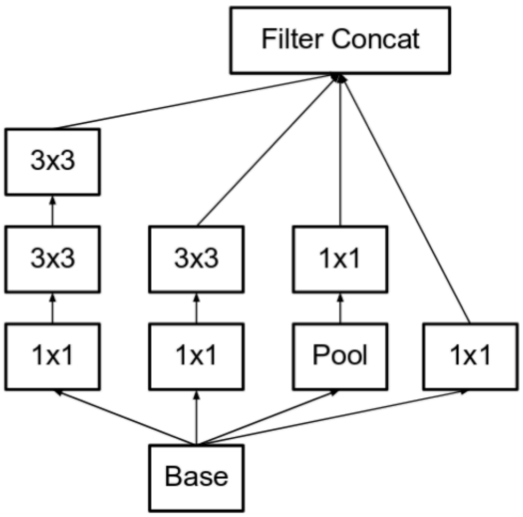
generality of Above architucture is litterally repeated in all the versions but modified in terms of depth, kernel sizes and auxiliary classifiers and ... , for example the depth pof network in version 3 is 48 while in version 1 it's 27.
and the general architucture of network which is modified in version 3 is like below :



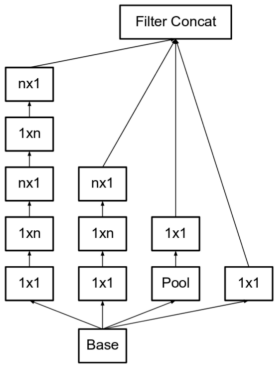
the above architucture may seem far different from firt figures which is true beacuase in version 2 and 3 kernel sizes got modified like below :

In version 2 below results were implemented on the network and modified the architucture of kernels :

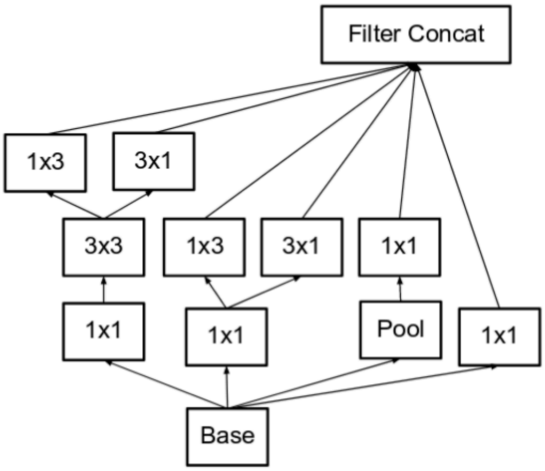
- Factorize 5x5 convolution to two 3x3 convolution operations to improve computational speed. Although this may seem counterintuitive, a 5x5 convolution is 2.78 times more expensive than a 3x3 convolution. So stacking two 3x3 convolutions infact leads to a boost in performance. This is illustrated in the below image.



- Moreover, they factorize convolutions of filter size nxn to a combination of 1xn and nx1 convolutions. For example, a 3x3 convolution is equivalent to first performing a 1x3 convolution, and then performing a 3x1 convolution on its output. They found this method to be 33 percent more cheaper than the single 3x3 convolution. This is illustrated in the below image.



- The filter banks in the module were expanded (made wider instead of deeper) to remove the representational bottleneck. If the module was made deeper instead, there would be excessive reduction in dimensions, and hence loss of information. This is illustrated in the below image.



The above three principles were used to build three different types of inception modules (Let's call them modules A,B and C in the order they were introduced. These names are introduced for clarity, and not the official names)
fianlly verssion 3 came out with below upgrades incorporating features of previous versions : RMSProp Optimizer.
Factorized 7x7 convolutions.
BatchNorm in the Auxillary Classifiers.
Label Smoothing (A type of regularizing component added to the loss formula that prevents the network from becoming too confident about a class. Prevents over fitting).
and for preventing from getting long version 4 of inception is not included in the explanation but generally it added resnet besides minor upgrades where it prevented gradient-vanshing and also used max -pool beside conv layers to prevent infos from dying in the first layers of network.

2 Size of inputs

Since mostly we are working with pretrained network and keeping its last layer , we should use a (299 * 299 * 3) Image cause keeping the last fc layer of nework requires a consatnd size for its input which is created during an adaptive-average-mean but if we skip the last fc and add our own fc we can give it inputs with wesired dimensions. but not every dimension due to kernel sizes of network.

3 Pre-processing

```
def preprocess_input(x):
   x = np.divide(x, 255.0)
   x = np.subtract(x, 1.0)
   x = np.multiply(x, 2.0)
   return x
```

above function is implementation of preprocess of network and you see the each pixel is normalized between -2 and 0.

4 Size of Output

working with imagenet , output of last conv-layer is (8, 8, 2048) which with global aveareg pooling becomes (2048) and gets ready for classification fc. and number of 2048 can represent number of final and important features of decision-making for last fc.

5 comparison

generally we cannot compare models and say this model is the best and different models have different performances in different taskes but inception and it's different versions are categorized as deep networks(specially version 4) and have high cost of computations but in most of tasks they are among top 5 classifiers in terms of either loss or accuracy and that's because of the innovations such as resnet and widening filter in them and also using auxilary classifiers which is a heavy advantage for them among deep networks because it prevents gradient vanishing which is common in deep networks and also among models of same depth it's lower-cost because of techniques such as dividing one kernel to 2 or 3 kernels which was stated as a save of cost in terms of number of model parameters which is also visible in charts where required space for each trained network is written.