Introduction

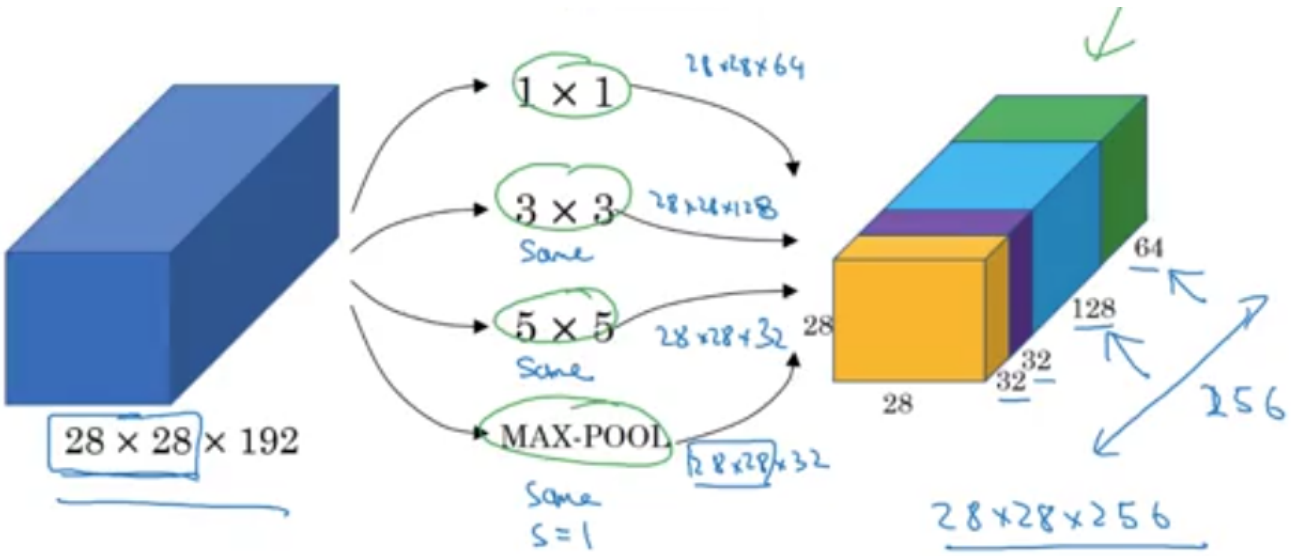
- Inception net achieved a milestone in CNN classifiers when previous models were just going deeper to improve the performance and accuracy but compromising the computational cost. The Inception network, on the other hand, is heavily engineered.
- It uses a lot of tricks to push performance, both in terms of speed and accuracy. It is the winner of the ImageNet Large Scale Visual Recognition Competition in 2014, an image classification competition, which has a significant improvement over ZFNet (The winner in 2013), AlexNet (The winner in 2012) and has relatively lower error rate compared with the VGGNet (1st runner-up in 2014).

The major issues faced by deeper CNN models such as VGGNet were:

1. Although, previous networks such as VGG achieved a remarkable accuracy on the ImageNet dataset, deploying these kinds of models is highly computationally expensive because of the deep architecture.
2. Very deep networks are susceptible to overfitting. It is also hard to pass gradient updates through the entire network.

Motivation for Inception Network

Instead of deciding whether to use a 1x1 convolution, or a 3x3 or a 5x5 Convolution, or whether to use a Pooling layer - Why not use all of them?

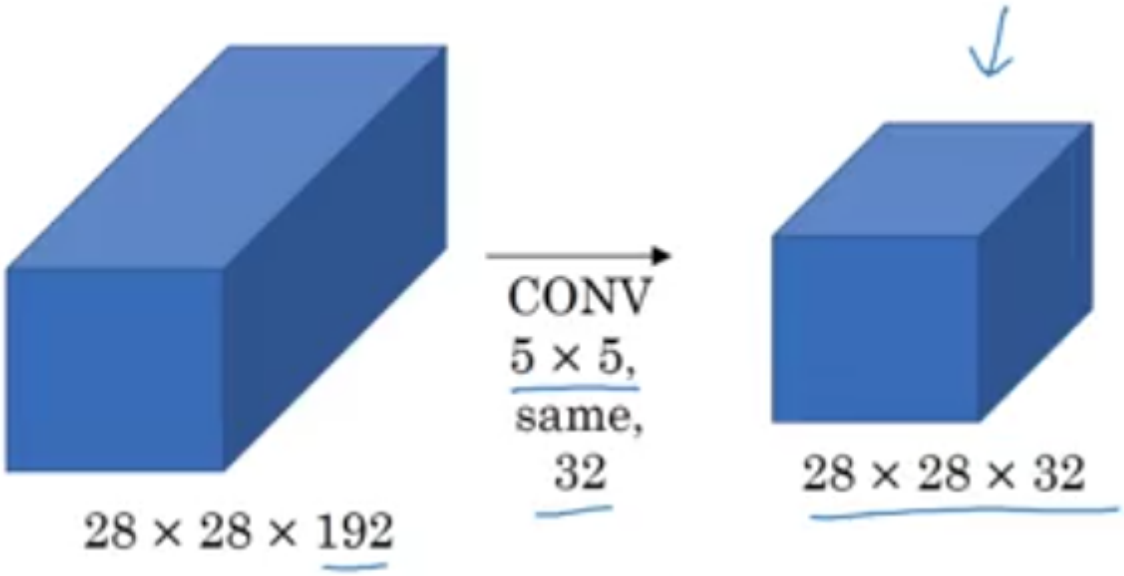


In the example above, all the filters are applied to the input to generate a stacked outpu, whioch contains the output of each filter stacked on top of each other. The Padding is kept at ‘same’ to ensure that the output from all the filters are of the same size.

Disadvantage: Huge memory cost

Solving the problem of memory cost

For example, the computational cost of th 5x5 filer in the above diagram:



Input: 28x28x192

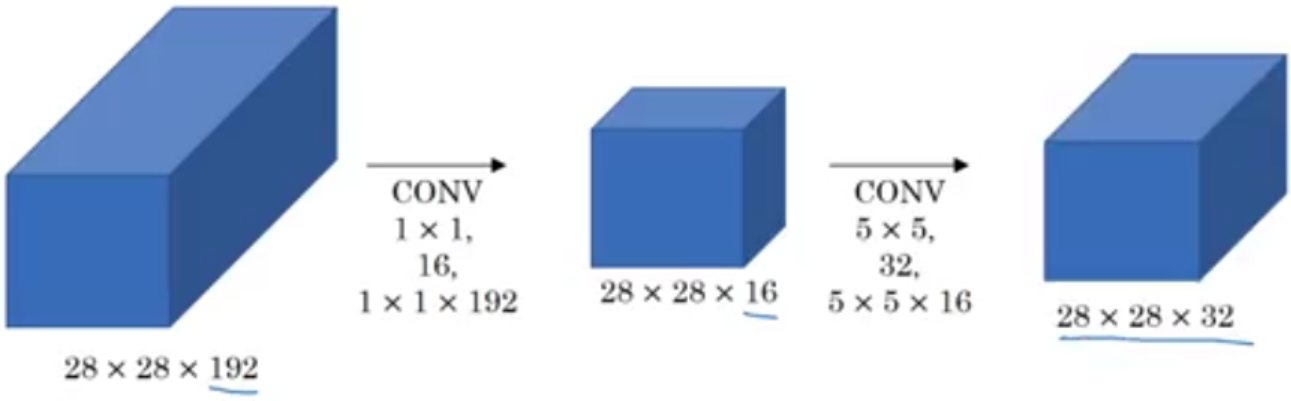
Filter: Conv 5x5x192, same, 32

Output: 28x28x32

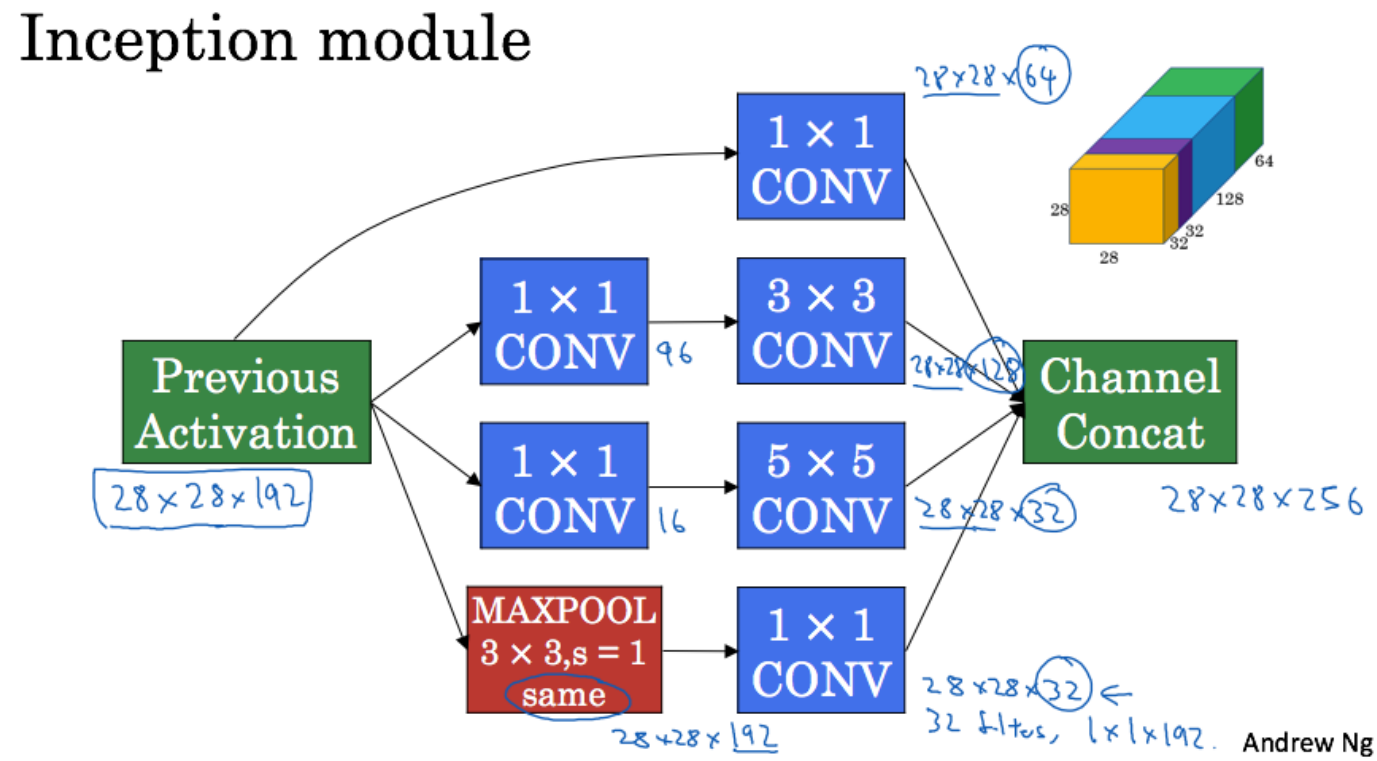
Total number of calculations = (28 *28* 32) *(5* 5 * 192) = 120 Million !!

Using 1x1 Convolution to reduce computation cost

A 1x1 convolution is added before the 5x5 cvonvolution -= Also called a bottleneck layer



Total number of calculations = [(28 *28* 16) *(1* 1 *192)]* + *[(28* 28 *32)* (5 *5* 16)] = 12.4 Million !! (earlier the cost was 120 Million)



The popular versions are as follows:

Inception v1.

Inception v2 and Inception v3.

Inception v4 and Inception-ResNet.

Inception V1

- This architecture has 22 layers in total! Using the dimension-reduced inception module, a neural network architecture is constructed. This is popularly known as GoogLeNet (Inception v1).
- GoogLeNet has 9 such inception modules fitted linearly. It is 22 layers deep (27, including the pooling layers). At the end of the architecture, fully connected layers were replaced by a global average pooling which calculates the average of every feature map. This indeed dramatically declines the total number of parameters.
- The above are the explainof V1

Problems of Inception V1 architecture:

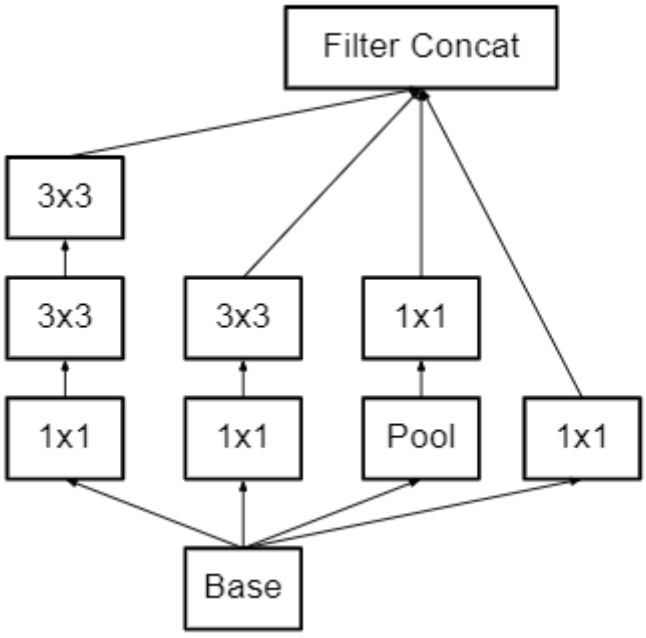
- Inception V1 have sometimes use convolutions such as 5*5 that causes the input dimensions to decrease by a large margin. This causes the neural network some accuracy decrease. The reason behind that the neural network is susceptible to information loss if the input dimension decreases too drastically.
- Furthermore, there is also complexity decrease when we use bigger convolutions like 5×5 as compared to 3×3.We can go further in terms of factorization i.e. that we can divide a 3×3 convolution into an asymmetric convolution of 1×3 then followed by 3×1 convolution. This is equivalent to sliding a two-layer network with the same receptive field as in a 3×3 convolution but 33% more cheaper than 3×3. This factorization does not work well for early layers when input dimensions are big but only when the input size mxm (m is between 12 and 20). According to the Inception V1 architecture, the auxiliary classifier improves the convergence of the network. They argue that it can help reduce the effect of the vanishing gradient problem in deep network by pushing the useful gradient to earlier layers (to reduce the loss). But, the authors of this paper found that this classifier didn't improve the convergence very much early in the training.

Inception V2 And Inception V3

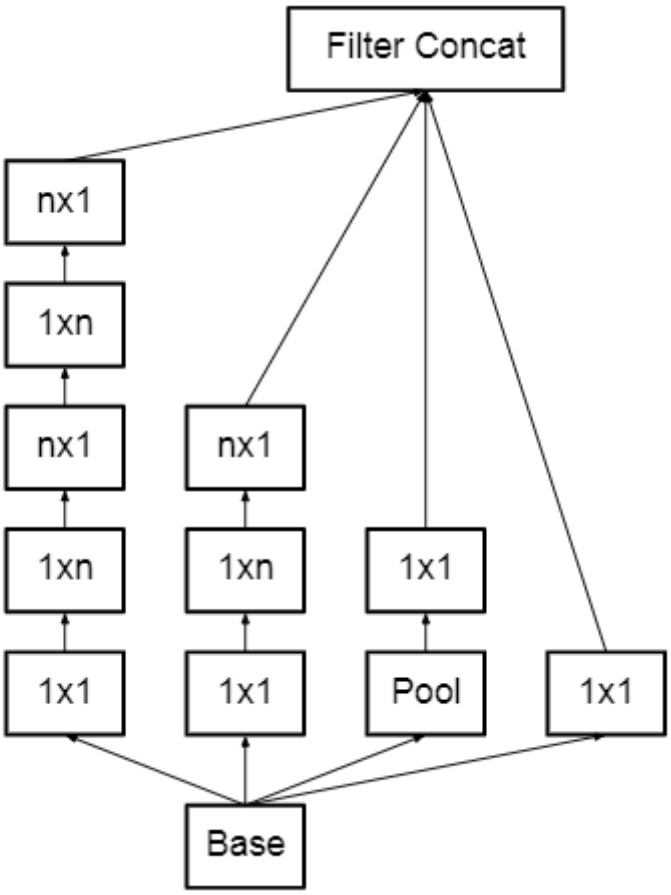
Inception-v2(2015)

Architectural Changes in Inception V2:

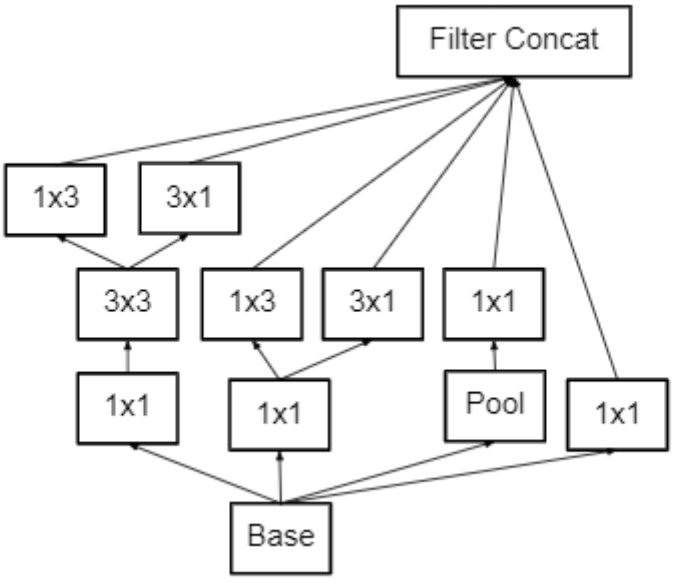
In the Inception V2 architecture. The 5×5 convolution is replaced by the two 3×3 convolutions. This also decreases computational time and thus increase computational speed because a 5×5 convolution is 2.78 more expensive than 3×3 convolution. So, Using two 3×3 layers instead of 5×5 increases the performance of architecture.



This architecture also converts nXn factorization into 1xn and nx1 factorization. As we discuss above that a 3×3 convolution can be converted into 1×3 then followed by 3×1 convolution which is 33% cheaper in terms of computational complexity as compared to 3×3.



To deal with the problem of the representational bottleneck, the feature banks of the module were expanded instead of making it deeper. This would prevent the loss of information that causes when we make it deeper.



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). The architecture is as follows:

type	patch size/stride or remarks	input size
conv	$3\times3/2$	$299\times299\times3$
conv	$3\times3/1$	$149\times149\times32$
conv padded	$3\times3/1$	$147\times147\times32$
pool	$3\times3/2$	$147\times147\times64$
conv	$3\times3/1$	$73\times73\times64$
conv	$3\times3/2$	$71\times71\times80$
conv	$3\times3/1$	$35\times35\times192$
$3\times$Inception	As in figure 5	$35\times35\times288$
$5\times$Inception	As in figure 6	$17\times17\times768$
$2\times$Inception	As in figure 7	$8\times8\times1280$
pool	8×8	$8\times8\times2048$
linear	logits	$1\times1\times2048$
softmax	classifier	$1\times1\times1000$

Algorithm advantages:

1. **Improved learning rate** : In the BN model, a higher learning rate is used to accelerate training convergence, but it will not cause other effects. Because if the scale of each layer is different, then the learning rate required by each layer is different. The scale of the same layer dimension often also needs different learning rates. Usually, the minimum learning is required to ensure the loss function to decrease, but The BN layer keeps the scale of each layer and dimension consistent, so you can directly use a higher learning rate for optimization.

1. **Remove the dropout layer** : The BN layer makes full use of the goals of the dropout layer. Remove the dropout layer from the BN-Inception model, but no overfitting will occur.

1. **Decrease the attenuation coefficient of L2 weight** : Although the L2 loss controls the overfitting of the Inception model, the loss of weight has been reduced by five times in the BN-Inception model.

1. **Accelerate the decay of the learning rate** : When training the Inception model, we let the learning rate decrease exponentially. Because our network is faster than Inception, we will increase the speed of reducing the learning rate by 6 times.

1. **Remove the local response layer** : Although this layer has a certain role, but after the BN layer is added, this layer is not necessary.

1. **Scramble training samples more thoroughly** : We scramble training samples, which can prevent the same samples from appearing in a mini-batch. This can improve the accuracy of the validation set by 1%, which is the advantage of the BN layer as a regular term. In our method, random selection is more effective when the model sees different samples each time.

1. **To reduce image distortion**: Because BN network training is faster and observes each training sample less often, we want the model to see a more realistic image instead of a distorted image.

Inception-v3-2015

This architecture focuses, how to use the convolution kernel two or more smaller size of the convolution kernel to replace, but also the introduction of **asymmetrical layers i.e. a convolution dimensional convolution** has also been proposed for pooling layer Some remedies that can cause loss of spatial information; there are ideas such as **label-smoothing , BN-ahxiliary** .

Experiments were performed on inputs with different resolutions . The results show that although low-resolution inputs require more time to train, the accuracy and high-resolution achieved are not much different.

The computational cost is reduced while improving the accuracy of the network.

General Design Principles

We will describe some design principles that have been proposed through extensive experiments with different architectural designs for convolutional networks. At this point, full use of the following principles can be guessed, and some additional experiments in the future will be necessary to estimate their accuracy and effectiveness.

1. **Prevent bottlenecks in characterization** . The so-called bottleneck of feature description is that a large proportion of features are compressed in the middle layer (such as using a pooling operation). This operation will cause the loss of feature space information and the loss of features. Although the operation of pooling in CNN is important, there are some methods that can be used to avoid this loss as much as possible (I note: later hole convolution operations).
2. **The higher the dimensionality of the feature, the faster the training converges** . That is, the independence of features has a great relationship with the speed of model convergence. The more independent features, the more thoroughly the input feature information is decomposed. It is easier to converge if the correlation is strong. Hebbin principle : fire together, wire together.
3. **Reduce the amount of calculation through dimensionality reduction** . In v1, the feature is first reduced by 1x1 convolutional dimensionality reduction. There is a certain correlation between different dimensions. Dimension reduction can be understood as a lossless or low-loss compression. Even if the dimensions are reduced, the correlation can still be used to restore its original information.
4. **Balance the depth and width of the network** . Only by increasing the depth and width of the network in the same proportion can the performance of the model be maximized.

Inception V3 is similar to and contains all the features of Inception V2 with following changes/additions:

```
    1. Use of RMSprop optimizer.
    2. Batch Normalization in the fully connected layer of Auxiliary classifier.
    3. Use of 7×7 factorized Convolution
    4. Label Smoothing Regularization: It is a method to regularize the classifier
by estimating the effect of label-dropout during training. It prevents the classif
ier to predict too confidently a class. The addition of label smoothing gives 0.2%
improvement from the error rate.
```

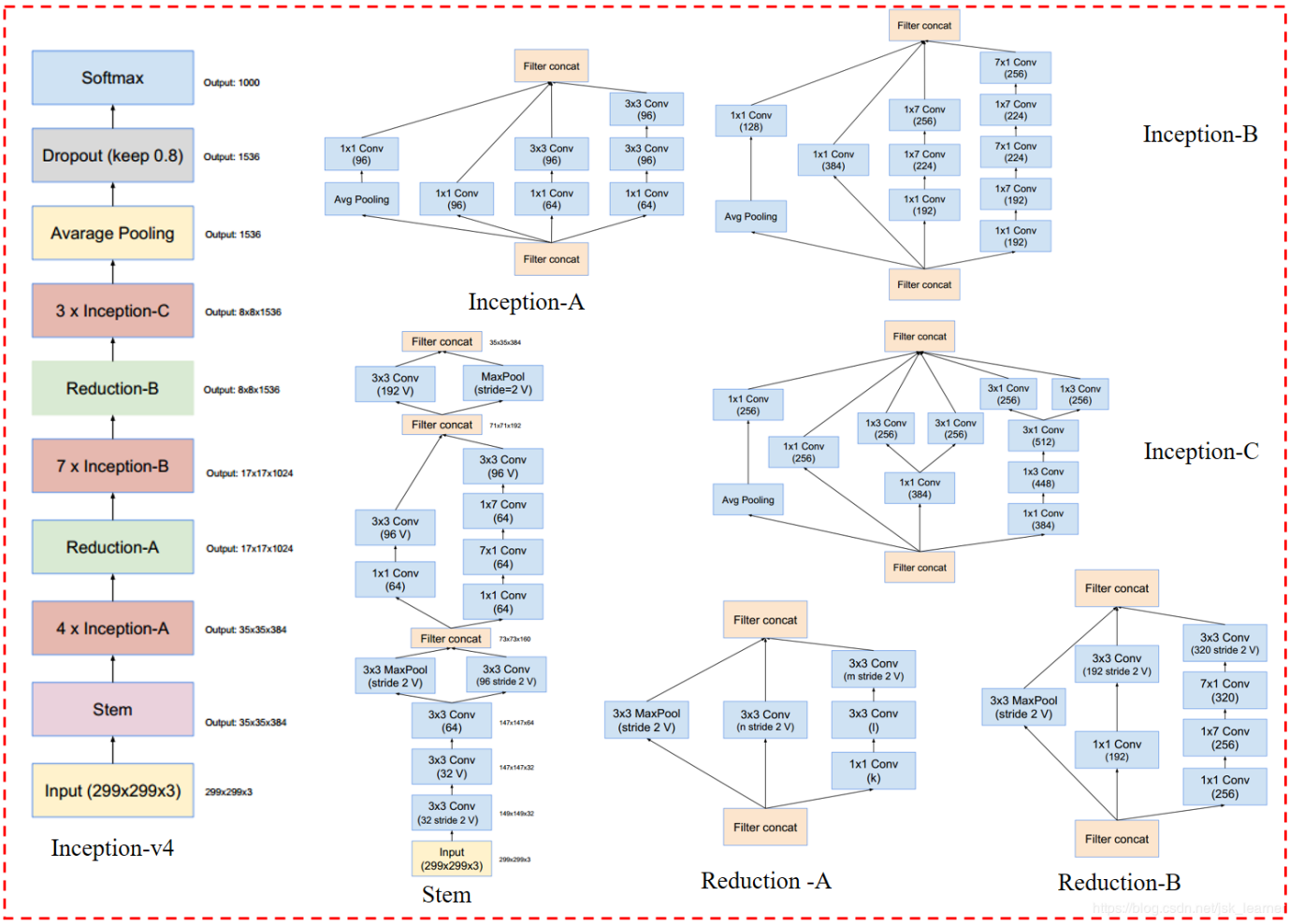
Inception-v4-2016

- Inception V4 was introduced in combination with Inception-ResNet by thee researchers a Google in 2016. The main aim of the paper was to reduce the complexity of Inception V3 model which give the state-of-the-art accuracy on ILSVRC 2015 challenge. This paper also explores the possibility of using residual networks on Inception model.

Introduction

Residual conn works well when training very deep networks. Because the Inception network architecture can be very deep, it is reasonable to use residual conn instead of concat.

Compared with v3, Inception-v4 has more unified simplified structure and more inception modules.



The big picture of Inception-v4:

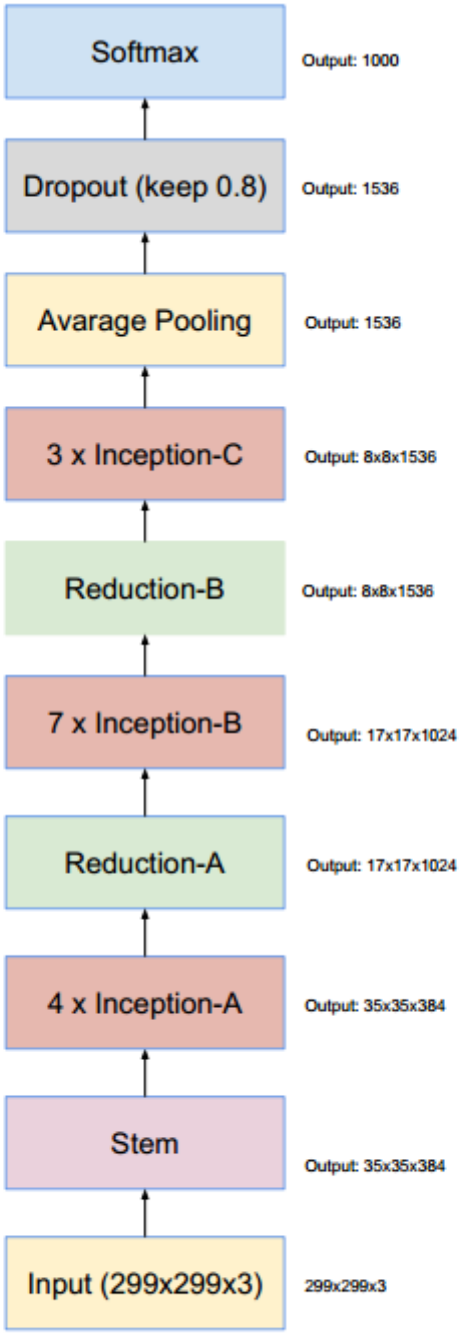


Figure 9. The overall schema of the Inception-v4 network. For the detailed modules, please refer to Figures 3, 4, 5, 6, 7 and 8 for the detailed structure of the various components.

Fig9 is an overall picture, and Fig3,4,5,6,7,8 are all local structures. For the specific structure of each module, see the end of the article.

Residual Inception Blocks

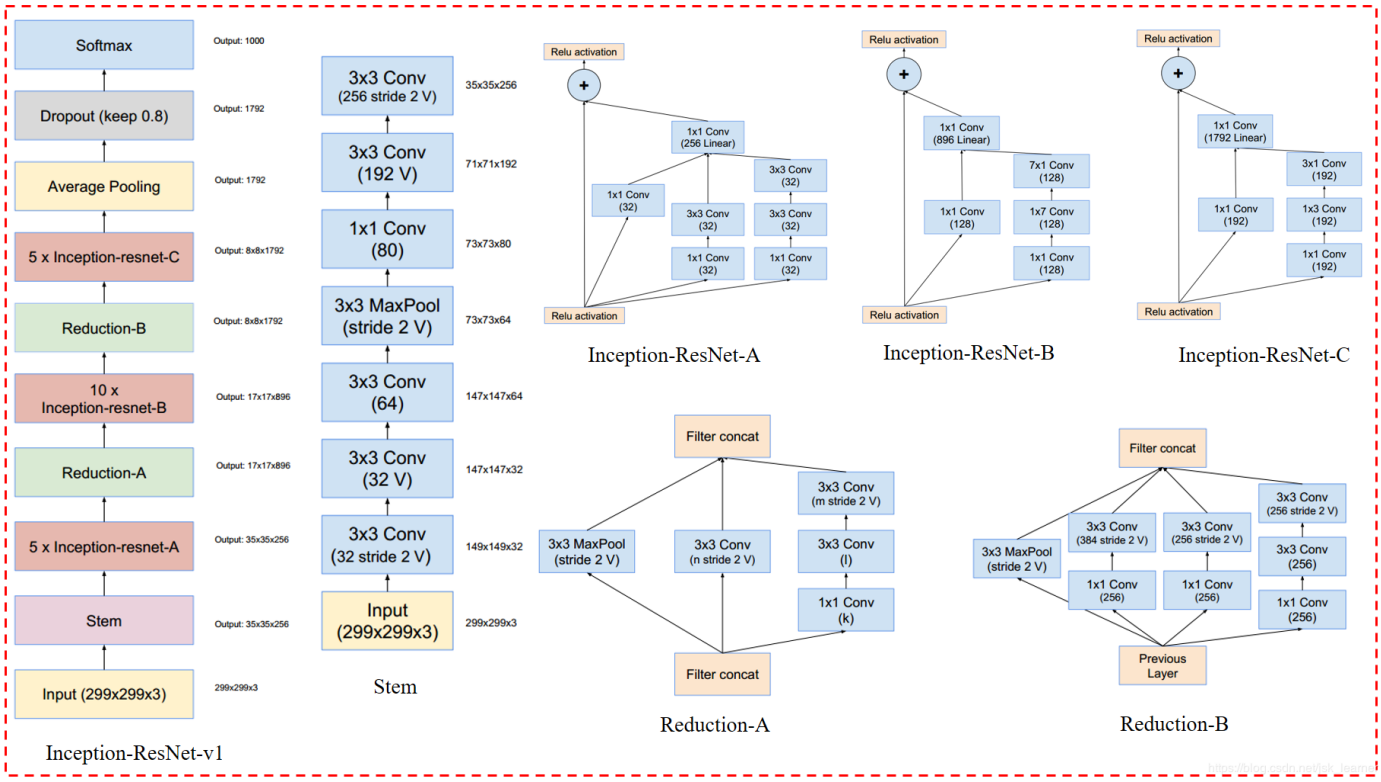
For the residual version in the Inception network, we use an Inception module that consumes less than the original Inception. The convolution kernel (followed by 1x1) of each Inception module is used to modify the dimension, which can compensate the reduction of the Inception dimension to some extent.

One is named **Inception-ResNet-v1**, which is consistent with the calculation cost of Inception-v3. One is named **Inception-ResNet-v2**, which is consistent with the calculation cost of Inception-v4.

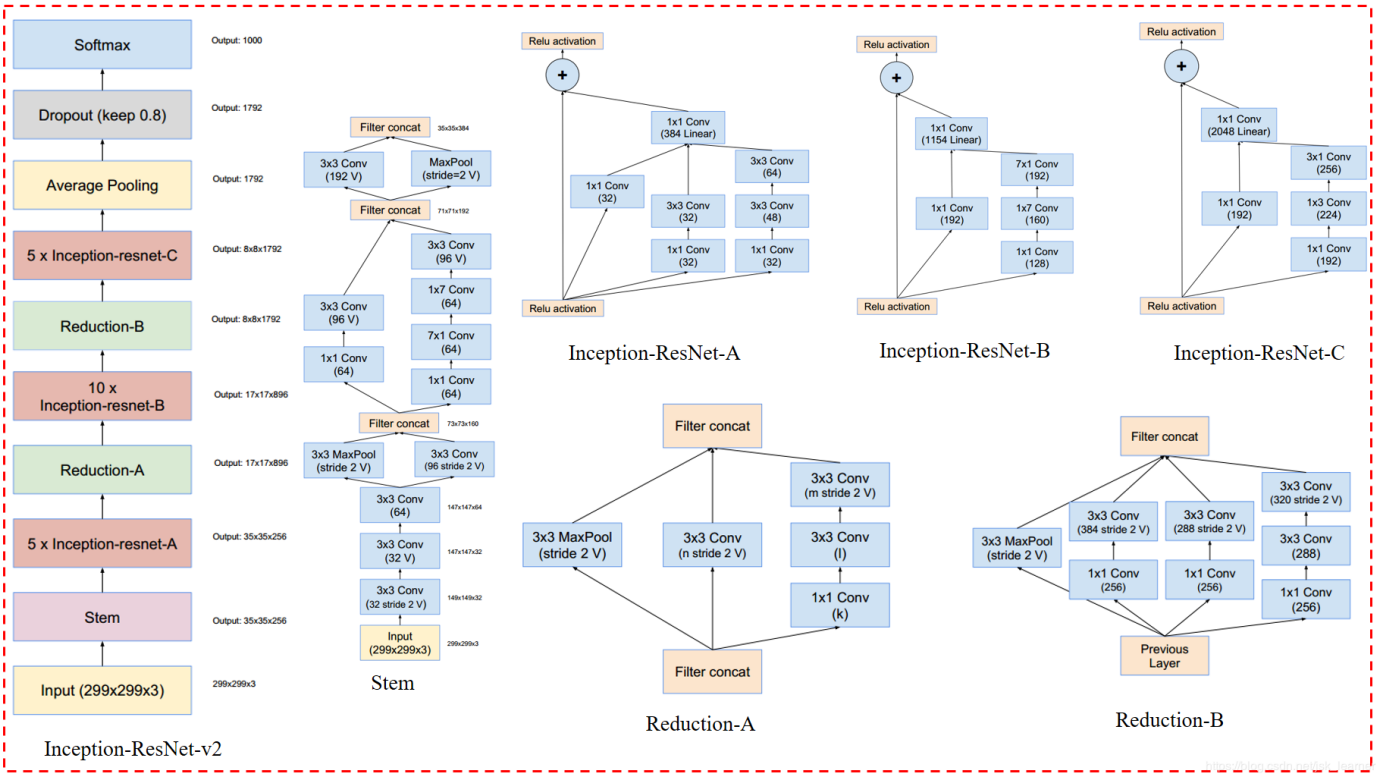
Figure 15 shows the structure of both. However, Inception-v4 is actually slower in practice, probably because it has more layers.

Another small technique is that we use the BN layer in the header of the traditional layer in the Inception-ResNet module, but not in the header of the summations. ** There is reason to believe that the BN layer is effective. But in order to add more Inception modules, we made a compromise between the two.

Inception-ResNet-v1



Inception-ResNet-v2



Scaling of the Residuals

This paper finds that when the number of convolution kernels exceeds 1,000 , the residual variants will start to show instability , and the network will die in the early stages of training, which means that the last layer before the average pooling layer is in the Very few iterations start with just a zero value . This situation cannot be

prevented by reducing the learning rate or by adding a BN layer . Hekaiming's ResNet article also mentions this phenomenon.

This article finds that scale can stabilize the training process before adding the residual module to the activation layer . This article sets the scale coefficient between 0.1 and 0.3.

In order to prevent the occurrence of unstable training of deep residual networks, He suggested in the article that it is divided into two stages of training. The first stage is called warm-up (preheating) , that is, training the model with a very low learning first. In the second stage, a higher learning rate is used. And this article finds that if the convolution sum is very high, even a learning rate of 0.00001 cannot solve this training instability problem, and the high learning rate will also destroy the effect. But this article considers scale residuals to be more reliable than warm-up.

Even if scal is not strictly necessary, it has no effect on the final accuracy, but it can stabilize the training process.

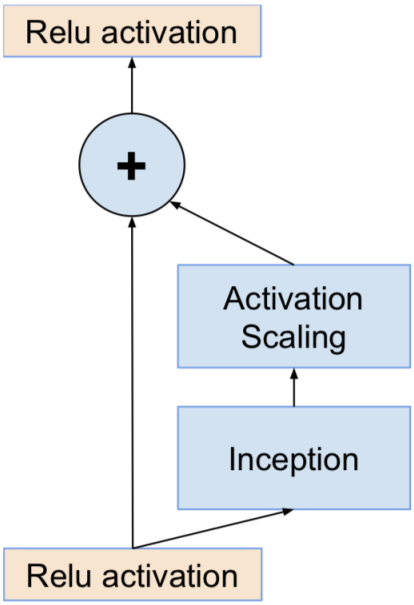


Figure 20. The general schema for scaling combined Inception-resnet moduels. We expect that the same idea is useful in the general resnet case, where instead of the Inception block an arbitrary subnetwork is used. The scaling block just scales the last linear activations by a suitable constant, typically around 0.1.

Conclusion

Inception-ResNet-v1 : a network architecture combining inception module and resnet module with similar calculation cost to Inception-v3;

Inception-ResNet-v2 : A more expensive but better performing network architecture.

Inception-v4 : A pure inception module, without residual connections, but with performance similar to Inception-ResNet-v2.