Paper Notes

GOING DEEPER WITH CONVOLUTIONS

Authors of the Paper: Christian Szegedy et al.

COMPILED BY: Ashutosh Singh, Aniket Agarwal

Discussion held on 17th October 2018.

This paper introduced yet another milestone in the field of Image Classification by not only winning the ILSVRC 2014 image classification challenge but also altering the classic model architecture of Convolution -> Pooling -> FC layers; a trend being followed since the LeNet (Le Cunn et al., 1989). The authors propose a different unit called **Inception Module** which is used in the model pipeline so as to refine the classification results.

Motivation

1. Previous paper observations:
    a. **Krizhevsky et al., 2012:** Used deep convolutional networks for large-scale image recognition for the first time.
    b. **Min Lin et al., 2013:** Use of 1*1 convolutions so as to increase the non-linearity in the network. Also, global average pooling was used to reduce overfitting and make a more robust model for spatial transformations.
2. Inspired by the biological vision cortex which can change the receptive field on account of a range of object sizes.
3. To make the network deeper, 1*1 convolutions were used to keep the number of computations and parameters constant.
4. The network also took inspiration from the **Hebbian Principle**, which states that 'cells that fire together wire together', which can be used in the training and assigning weights for the various different convolutions.
5. ConvNets have traditionally used random and sparse connection tables in the feature dimension. This raised the question whether there is a possibility of extra

sparsity even at the filter level, which is rightly seen in the case of an Inception Module.

6. This paper tries to solve the **bottleneck** imposed on the depth of the network on account of the limited computational resources.

Work done in Parallel

1. **VGG (Simonyan et al., 2014):** Another deep ConvNet having a classical architecture and stood 2nd and 1st in image classification and localization challenges, respectively.
2. **Goodfellow et al., 2014:** Another deep ConvNet to the task of street number recognition, showing increased depth led to better performance.

Architecture

Inception Module

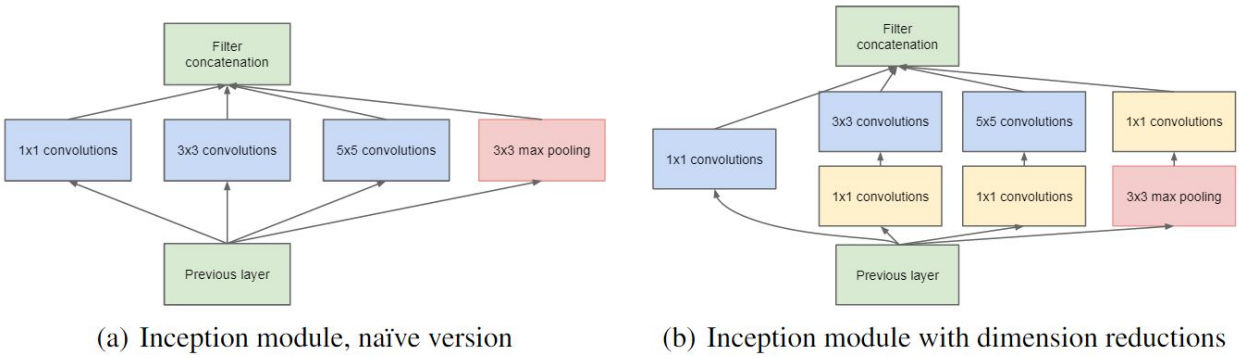

Figure 1: Inception Module (taken from paper)

1. The left diagram (Figure 1 (a)) shows the naive version of the Inception module, where the 4 different operations are carried out and the results of all the operations are stacked together.

2. The previous layers are appropriately padded before each operation so that the spatial dimension of the input and output layers are same.

3. So as to tackle the increased number of operations and params after every operation in the naive module, 1*1 convolution operations were used before all the computationally expensive operations (Figure 1(b)) and served two purposes:
    a. Increasing non-linearity in the network.
    b. Decreasing the number of operations and parameters in the network (nearly 10 times).
4. 1*1 convolutions are done **after** the pooling layer, rather than before like the other operations, so as to reduce the information loss.

GoogLeNet

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Table 1: GoogLeNet incarnation of the Inception architecture

1. **Preprocessing:** Subtracting every image by the mean RGB image computed on the training set.
2. All hidden layers are followed by ReLU non-linearity.

3. The inception module is used in the deeper parts of the network, which is due to the fact that there is much more important information that needs to be extracted in the deeper parts of the network.

4. The model has various extra networks (**auxiliary classifiers**) on the side which are connected to intermediate layers so as to encourage discrimination in lower stages of the network, increase the gradient signal that gets propagated back, and provide additional regularization.

5. During training, the auxiliary classifier's loss is weighted by 0.3 and gets added to the total loss of the network. At inference time, these classifiers are discarded.

Implementation Details

1. Usage of SGD with 0.9 momentum, fixed learning rate schedule (decreasing the learning rate by 4% every 8 epochs).
2. Polyak averaging (B.T. Polyak et al. 1992) was used to create the final model at inference time.
3. The sampling of various sized patches of image, with sizes distributed evenly between 8% and 100% of the image area and whose aspect ratio is chosen randomly between 3/4 and 4/3.

Observations

1. Usage of Inception Module in the Network led to decrease in the headache of finding an appropriate convolution size.
2. Usage of 1*1 convolution led to the decrease in parameters and computations and hence the network was able to be 22 layers deep, with fewer params and faster computation than VGG network.
3. The use of average pooling at the very end led to lower information loss, lesser overfitting and network being more robust to spatial translations of the input.

Take Away Message

Increasing the depth of the network mostly always increases the accuracy of the model. Also, 1*1 convolutions can be used to increase non-linearity in the network while at the same time making the network more efficient in terms of computation.