Rethinking the Inception Architecture for Computer Vision

Christian Szegedy
Google Inc.
szegedy@google.com

Vincent Vanhoucke
vanhoucke@google.com

Sergey Ioffe
sioffe@google.com

Jonathon Shlens
shlens@google.com

Zbigniew Wojna
University College London
zbigniewwojna@gmail.com

Presented by Lee Jongdae

Abstract & 1. Introduction

- VGG: simple but expensive

  much parameter than AlexNet(3x)

- GoogLeNet: good performance in restrict environment

  less parameter than AlexNet(1/12x)

Also, the computational cost of Inception is much cheaper

Thus, Inception is suitable to the scenarios where memory or computational capacity is limited(e.g. mobile vision).

Describe a few principle and optimization ideas that proved to be useful for scaling up convolution networks in efficient ways.

2.General Design Principles

The utility of the principles below are speculative and additional future experimental evidence will be necessary to assess.

1. Avoid representational bottlenecks, especially early in the network.
2. Higher dimensional representations are easier to process locally within a network
3. Spatial aggregation can be done over lower dimensional embeddings without much or any loss in representational power(just hypothesis)
4. Balance the width and depth of the network

The idea is to use them judiciously in ambiguous situations only.

3. Factorizing Convolutions

3.1. Factorization into small convolutions

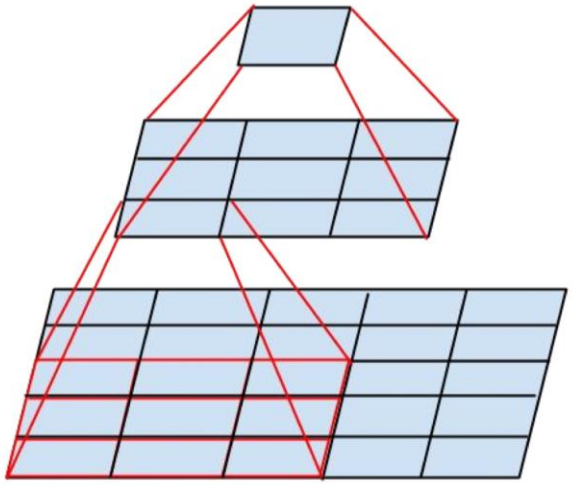

Figure 1. Mini-network replacing the 5×5 convolutions.

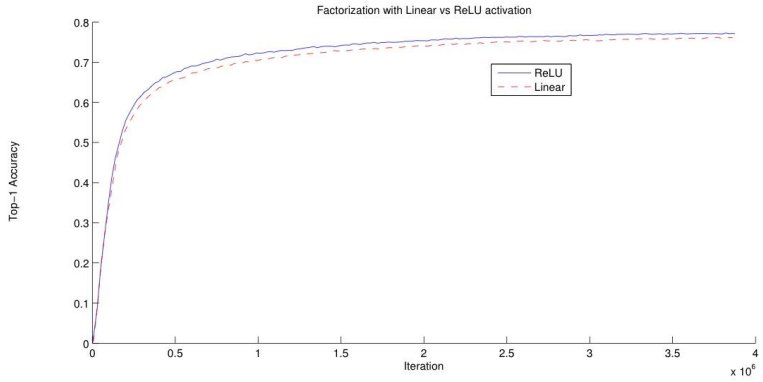

Figure 2. One of several control experiments between two Inception models, one of them uses factorization into linear + ReLU layers, the other uses two ReLU layers. After 3.86 million operations, the former settles at 76.2%, while the latter reaches 77.2% top-1 Accuracy on the validation set.

3. Factorizing Convolutions

3.1. Factorization into small convolutions

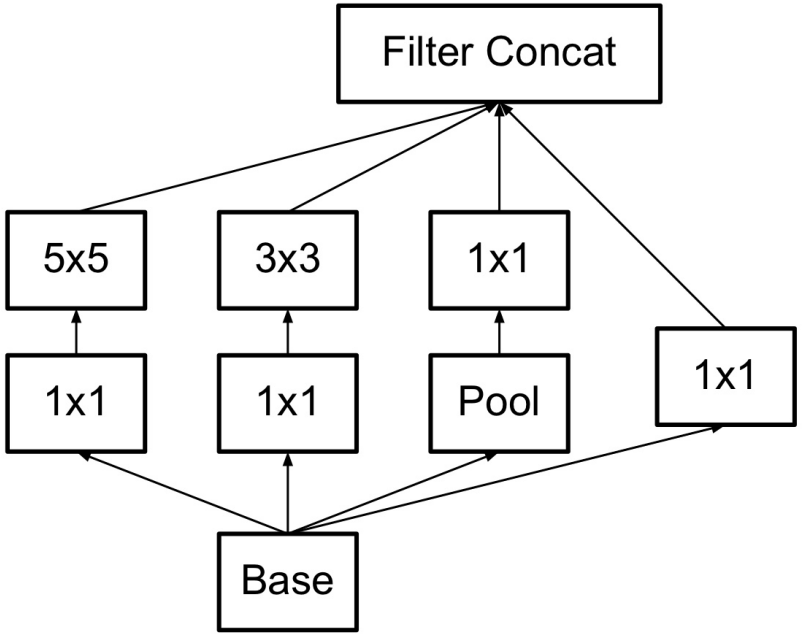

Figure 4. Original Inception module as described in [20].

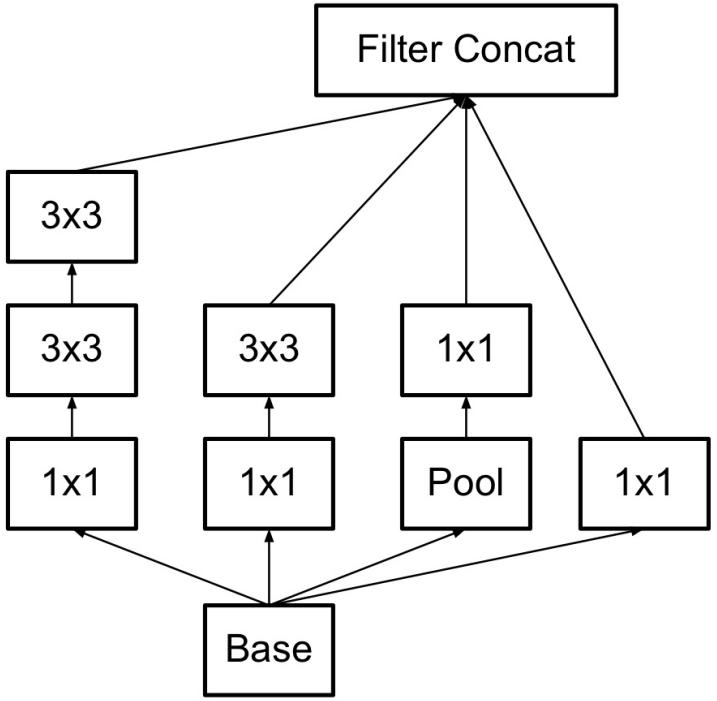

Figure 5. Inception modules where each 5×5 convolution is replaced by two 3×3 convolution, as suggested by principle 3 of Section 2.

3. Factorizing Convolutions

- 3.2. Spatial Factorization into Asymmetric Convolutions

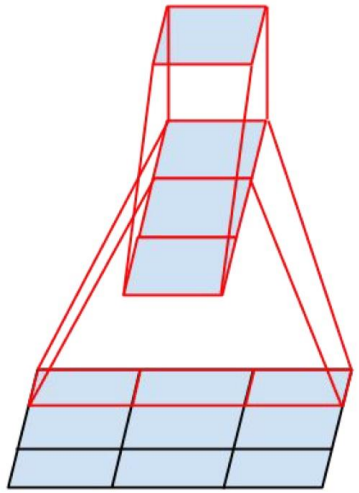

Figure 3. Mini-network replacing the 3×3 convolutions. The lower layer of this network consists of a 3×1 convolution with 3 output units.

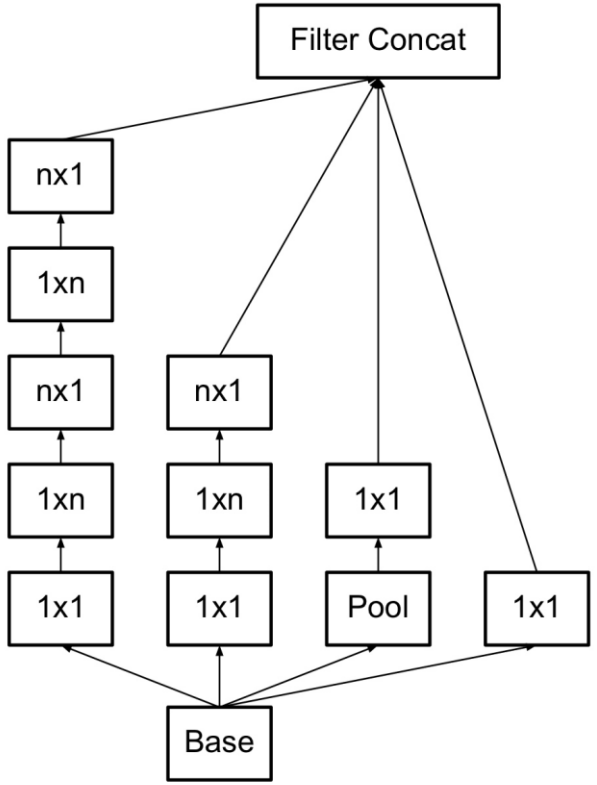

Figure 6. Inception modules after the factorization of the $n \times n$ convolutions. In our proposed architecture, we chose $n = 7$ for the 17×17 grid. (The filter sizes are picked using principle 3)

4. Utility of Auxiliary Classifiers

- Original motivation: to push useful gradients to the lower layers to make them immediately and improve the convergence during training
- Indeed, it did not result in improved convergence early in the training
- The removal of the lower auxiliary branch did not have any effect

- Instead, we argue that the auxiliary classifiers act as regularizer.
- (This is supported by the fact that the main classifier of the network performs better if the side branch is batch-normalized or has a dropout layer. This also gives a weak supporting evidence for the conjecture that batch normalization acts as a regularizer)

5. Efficient Grid Size Reduction

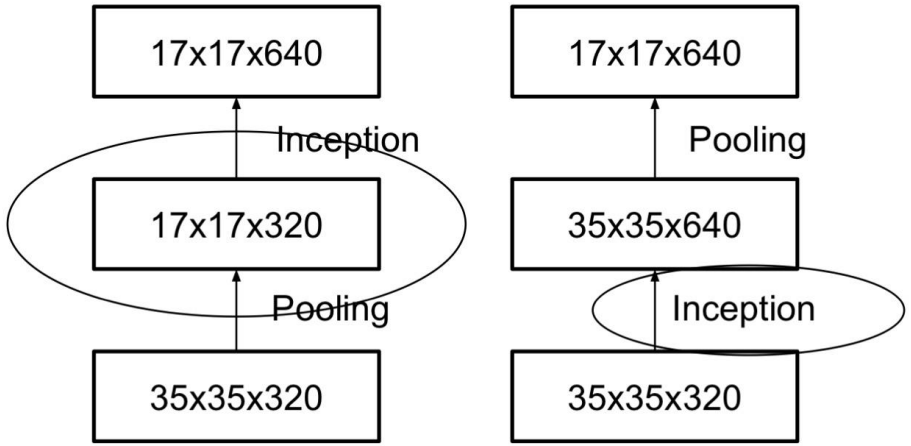

Figure 9. Two alternative ways of reducing the grid size. The solution on the left violates the principle 1 of not introducing an representational bottleneck from Section 2. The version on the right is 3 times more expensive computationally.

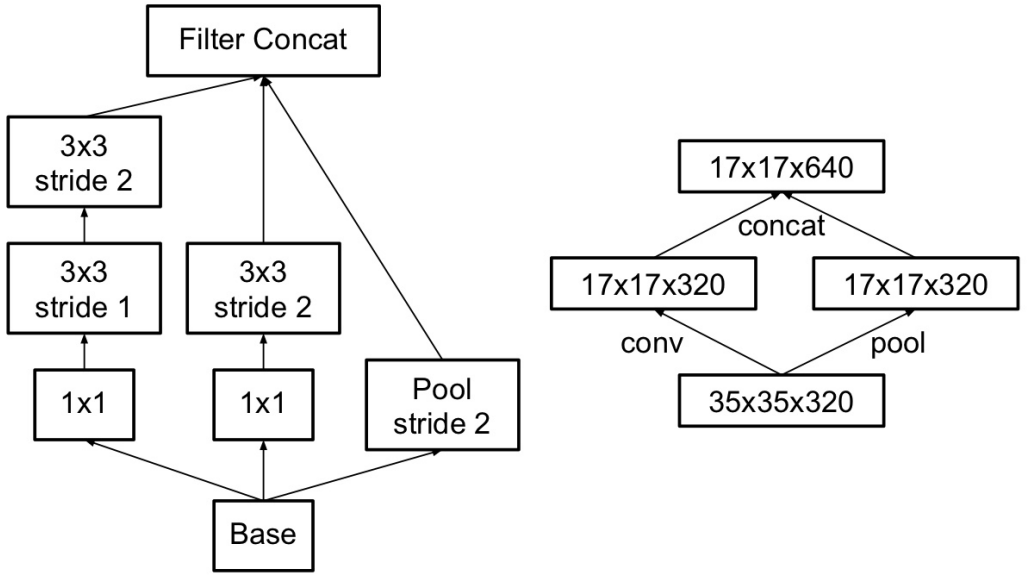

Figure 10. Inception module that reduces the grid-size while expands the filter banks. It is both cheap and avoids the representational bottleneck as is suggested by principle 1. The diagram on the right represents the same solution but from the perspective of grid sizes rather than the operations.

5. Efficient Grid Size Reduction

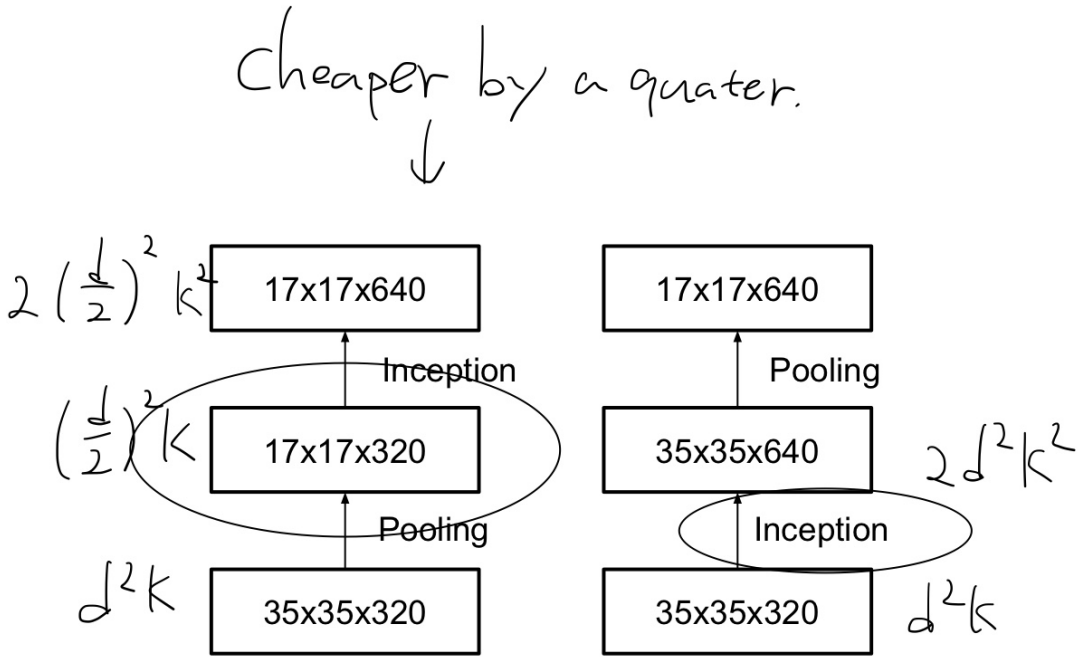

Figure 9. Two alternative ways of reducing the grid size. The solution on the left violates the principle 1 of not introducing an representational bottleneck from Section 2. The version on the right is 3 times more expensive computationally.

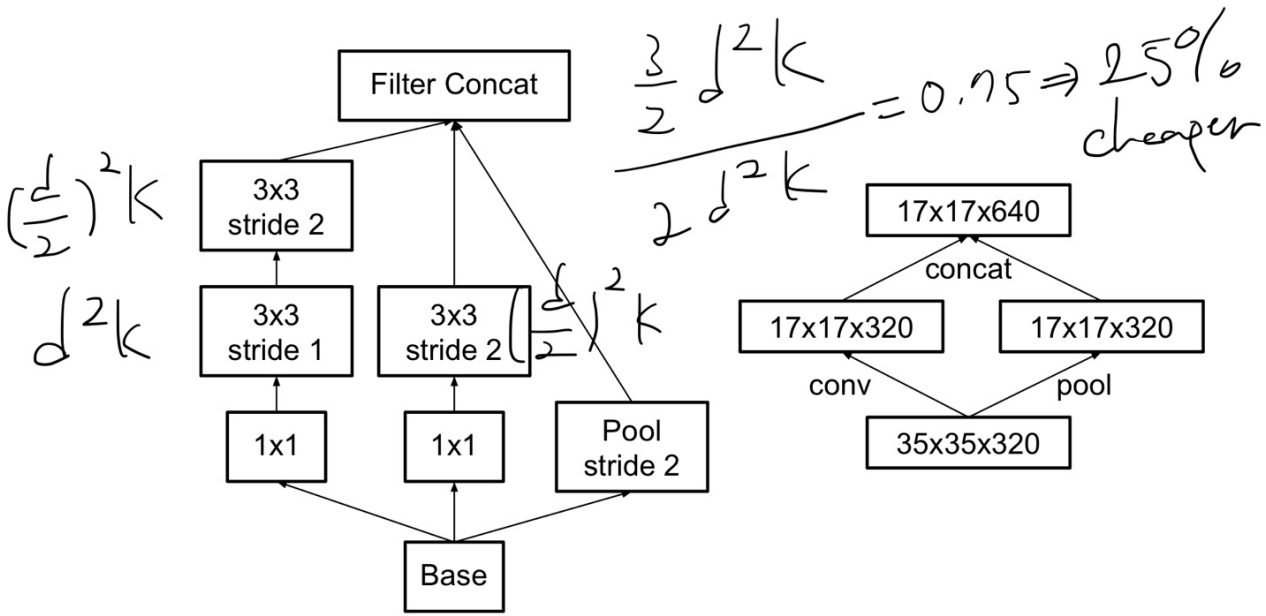

Figure 10. Inception module that reduces the grid-size while expands the filter banks. It is both cheap and avoids the representational bottleneck as is suggested by principle 1. The diagram on the right represents the same solution but from the perspective of grid sizes rather than the operations.

6. Inception-v2

Although our network is 42 layers deep, our computation cost is only about 2.5 higher than that of GoogLeNet(22 layers) and it is still much more efficient than VGGNet.

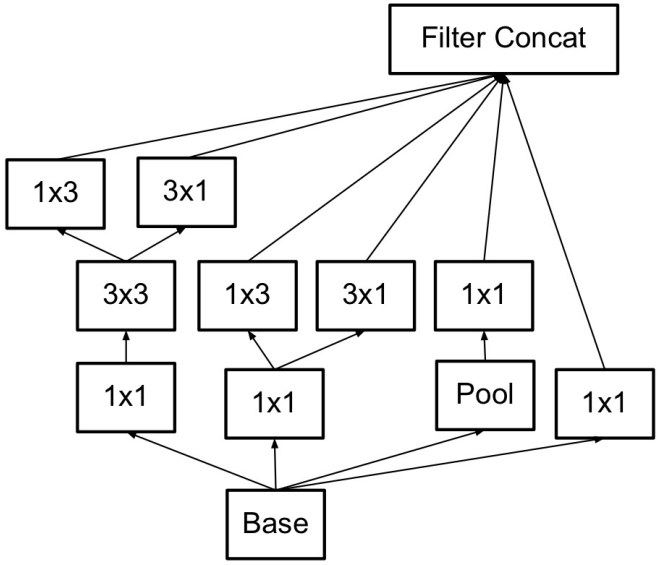

Figure 7. Inception modules with expanded the filter bank outputs. This architecture is used on the coarsest (8×8) grids to promote high dimensional representations, as suggested by principle 2 of Section 2. We are using this solution only on the coarsest grid, since that is the place where producing high dimensional sparse representation is the most critical as the ratio of local processing (by 1×1 convolutions) is increased compared to the spatial aggregation.

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	$8 \times 8 \times 2048$
linear	logits	$1 \times 1 \times 2048$
softmax	classifier	$1 \times 1 \times 1000$

Table 1. The outline of the proposed network architecture. The output size of each module is the input size of the next one. We are using variations of reduction technique depicted Figure 10 to reduce the grid sizes between the Inception blocks whenever applicable. We have marked the convolution with 0-padding, which is used to maintain the grid size. 0-padding is also used inside those Inception modules that do not reduce the grid size. All other layers do not use padding. The various filter bank sizes are chosen to observe principle 4 from Section 2.

7. Model Regularization via Label Smoothing

- $k \in 1 \ldots K.$
- $p(k \mid x) = \frac{e^{z_k}}{\sum_{i=1}^{K} e^{z_i}}.$
- z_i는 *logit* 혹은 *unnormalized log-probability*
- $\ell = -\sum_{k=1}^{K} \log(p(k))q(k).$
- $\frac{\partial \ell}{\partial z_k} = p(k) - q(k).$
- Bounded in $[-1, 1]$

7. Model Regularization via Label Smoothing

Two problems of the existing maximum likelihood estimation:

1. Over-fitting
2. Large differences between the largest logit and all others

Proposition: a mechanism for encouraging the model to be less confident

This regularize the model and makes it more adaptable.

7. Model Regularization via Label Smoothing

u(k): distribution of the labels independent of the training data

LSR: label-smoothing regularization

Replace the label distribution q($q(k|x) = \delta_{k,y}$

$$q'(k) = (1-\epsilon)\delta_{k,y} + \frac{\epsilon}{K}$$

It prevents the largest logit from becoming much larger than all others.

10. Experimental Results and Comparisons

Network	Top-1 Error	Top-5 Error	Cost Bn Ops
GoogLeNet [20]	29%	9.2%	1.5
BN-GoogLeNet	26.8%	-	**1.5**
BN-Inception [7]	25.2%	7.8	2.0
Inception-v2	23.4%	-	3.8
Inception-v2 RMSProp	23.1%	6.3	3.8
Inception-v2 Label Smoothing	22.8%	6.1	3.8
Inception-v2 Factorized 7×7	21.6%	5.8	4.8
Inception-v2 BN-auxiliary	**21.2%**	**5.6%**	4.8

Network	Crops Evaluated	Top-5 Error	Top-1 Error
GoogLeNet [20]	10	-	9.15%
GoogLeNet [20]	144	-	7.89%
VGG [18]	-	24.4%	6.8%
BN-Inception [7]	144	22%	5.82%
PReLU [6]	10	24.27%	7.38%
PReLU [6]	-	21.59%	5.71%
Inception-v3	12	19.47%	4.48%
Inception-v3	144	**18.77%**	**4.2%**

Network	Models Evaluated	Crops Evaluated	Top-1 Error	Top-5 Error
VGGNet [18]	2	-	23.7%	6.8%
GoogLeNet [20]	7	144	-	6.67%
PReLU [6]	-	-	-	4.94%
BN-Inception [7]	6	144	20.1%	4.9%
Inception-v3	4	144	**17.2%**	**3.58%***