Rethinking the Inception Architecture for Computer Vision

0. abstract

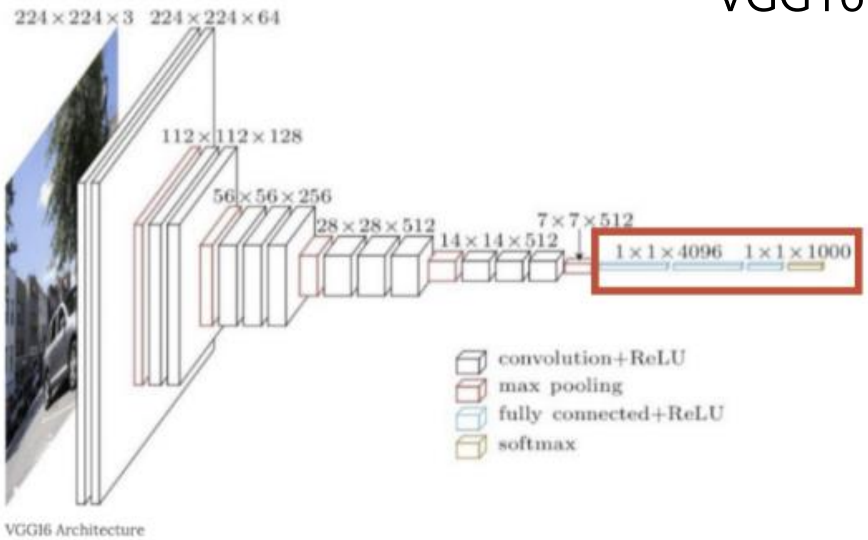
(1) CV 기술은 많은 발전을 이뤄왔고 이때 model의 큰 size와 높은computational cost가 model의 성능 향상으로 이어지는 경향을 보인다.

(2) 따라서 해당 논문에서는 아래 방식으로 모델의 크기를 효율적으로 증가시켰다고 한다.

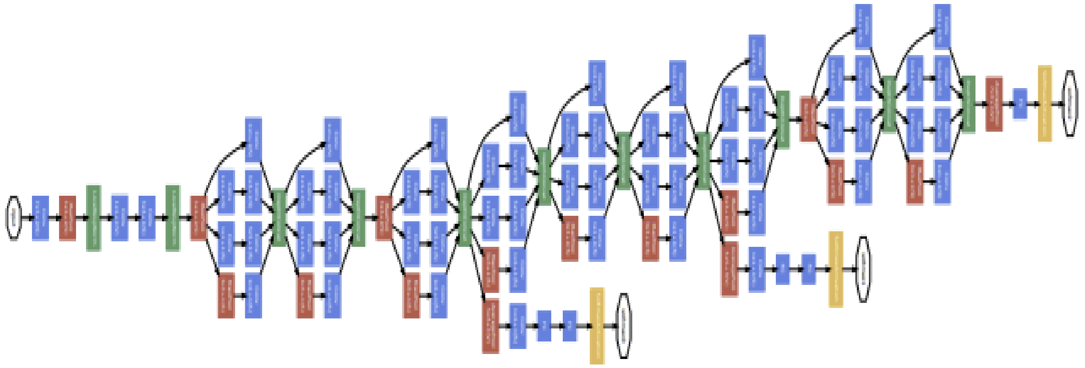
- factorized convolutions
- aggressive regularization

1. Introduction

VGG16



GoogLeNet



VGGNet: (장점) 단순하다 / (단점) 비용이 많이 든다

GoogLeNet: AlexNet보다 약 12배 적은 파라미터를 가지면 더 높은 성능을 보인다.

2. General Design Principles

1. Avoid representational bottleneck, especially early in the network

-> Representational Bottleneck이 발생하지 않도록 representation은 서서히 감소해야 한다.

2. Higher dimensional representations are easier to process locally with in a network

-> Conv Layer의 activation map 개수를 늘리면 disentangled feature를 많이 얻을 수 있으며, 네트워크가 더 빨리 학습할 수 있다.

3. Spatial aggregation can be done over lower dimensional embeddings without much or any loss in representational power

-> Conv 연산을 다수 수행할 때 적절한 dimension reduction을 해주는 것이 빠른 학습에 도움이 된다.

4. Balance the width and depth of the network

-> network의 width와 depth를 늘리면 성능 향상에 도움이 된다. 단, depth와 width를 균형있게 구성해야 한다.

3. Factorizing Convolutions

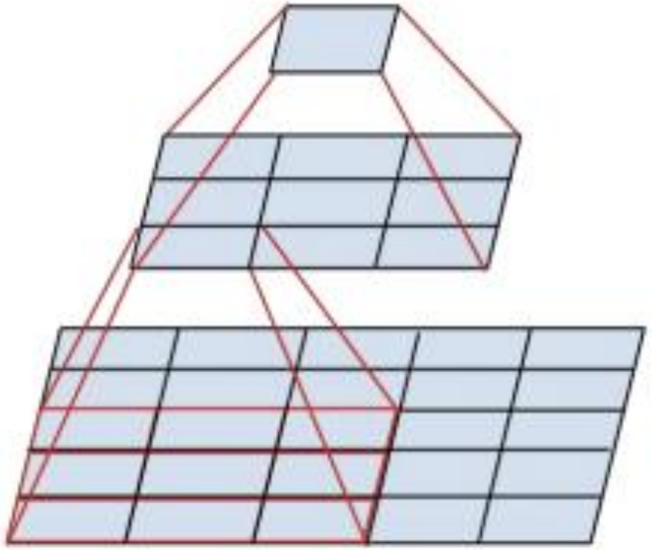
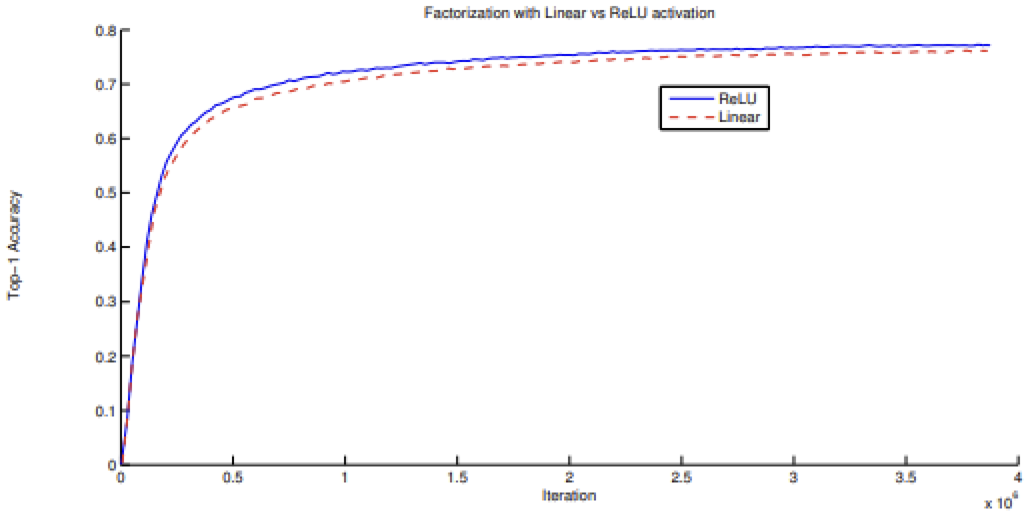


Figure 1. Mini-network replacing the 5 × 5 convolutions.



3. Factorizing Convolutions

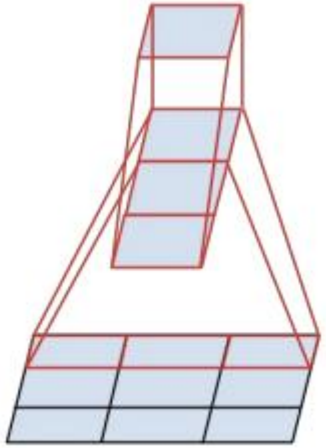


Figure 3. Mini-network replacing the 3 × 3 convolutions. The lower layer of this network consists of a 3 × 1 convolution with 3 output units.

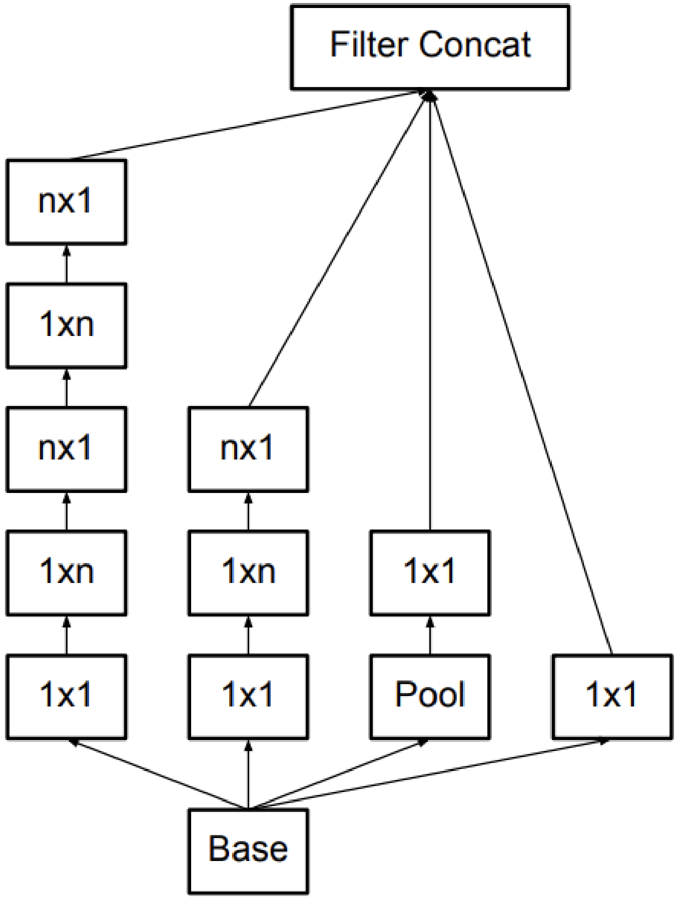
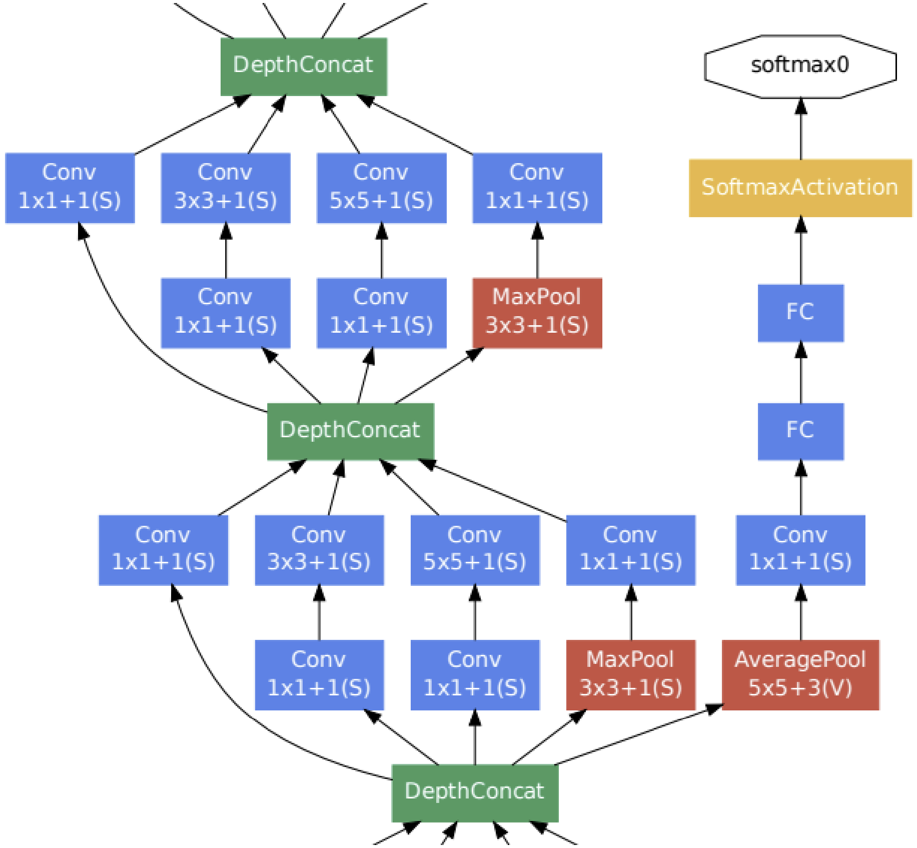


Figure 6. Inception modules after the factorization of the $n \times n$ convolutions. In our proposed architecture, we chose $n = 7$ for the 17 × 17 grid. (The filter sizes are picked using principle 3)

4. Utility of Auxiliary Classifiers



4. Utility of Auxiliary Classifiers

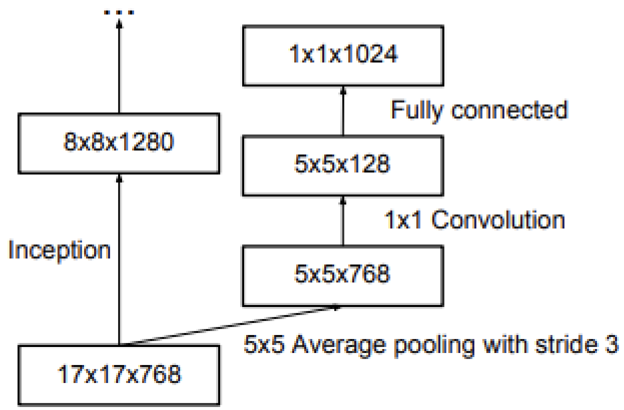


Figure 8. Auxiliary classifier on top of the last 17×17 layer. Batch normalization[7] of the layers in the side head results in a 0.4% absolute gain in top-1 accuracy. The lower axis shows the number of itertions performed, each with batch size 32.

5. Efficient Grid Size Reduction

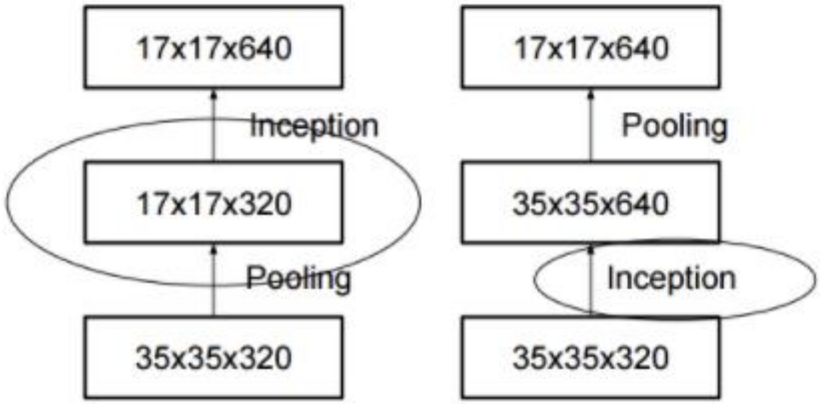
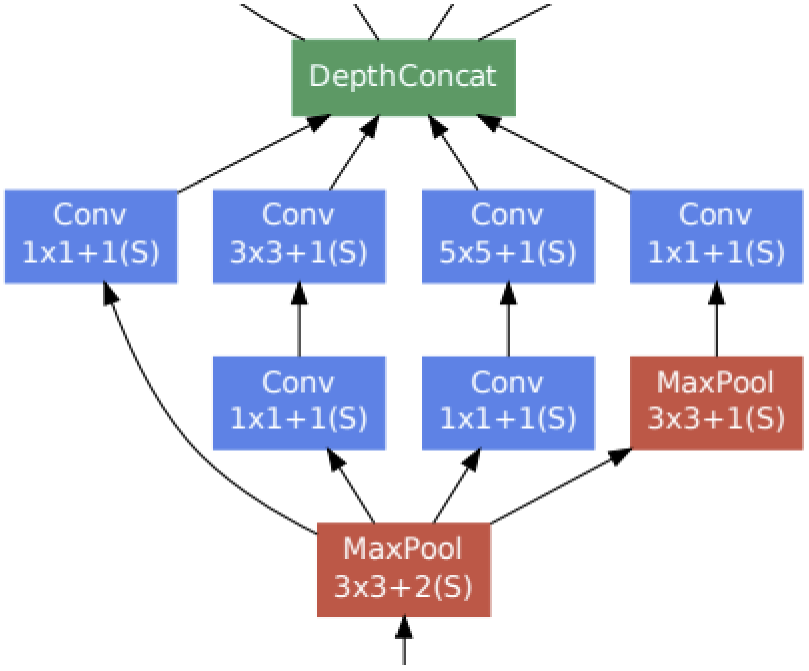


Figure 9. Two alternative ways of reducing the grid size. The solution on the left violates the principle 1 of not introducing an representational bottleneck from Section 2. The version on the right is 3 times more expensive computationally.

5. Efficient Grid Size Reduction



Inception module

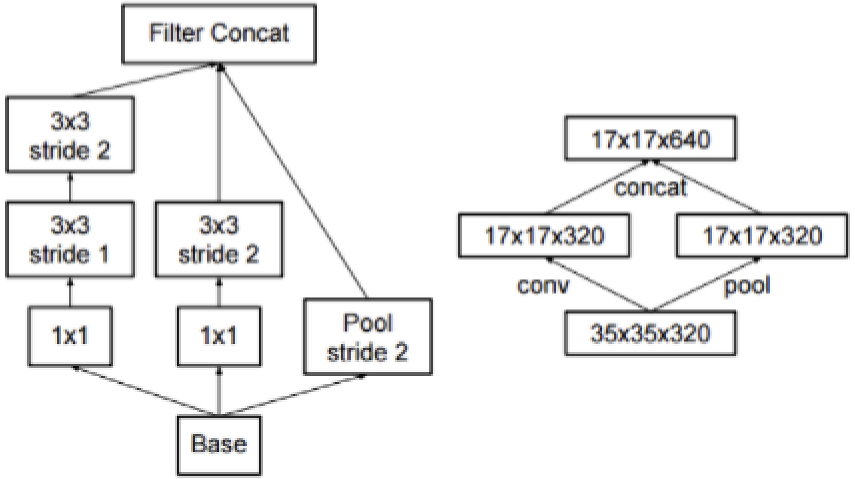
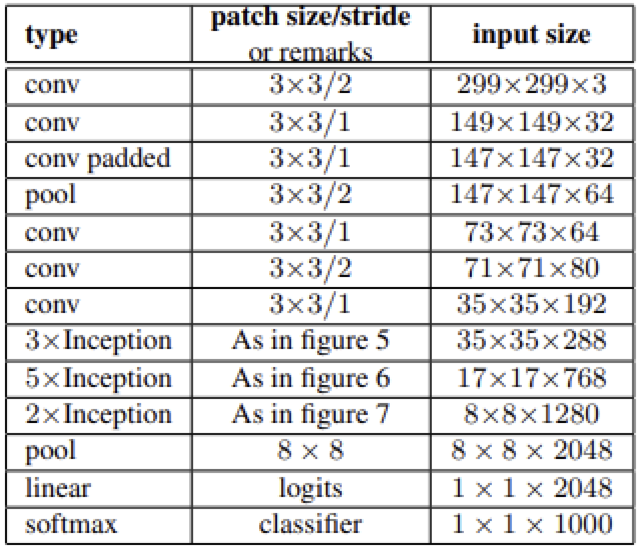


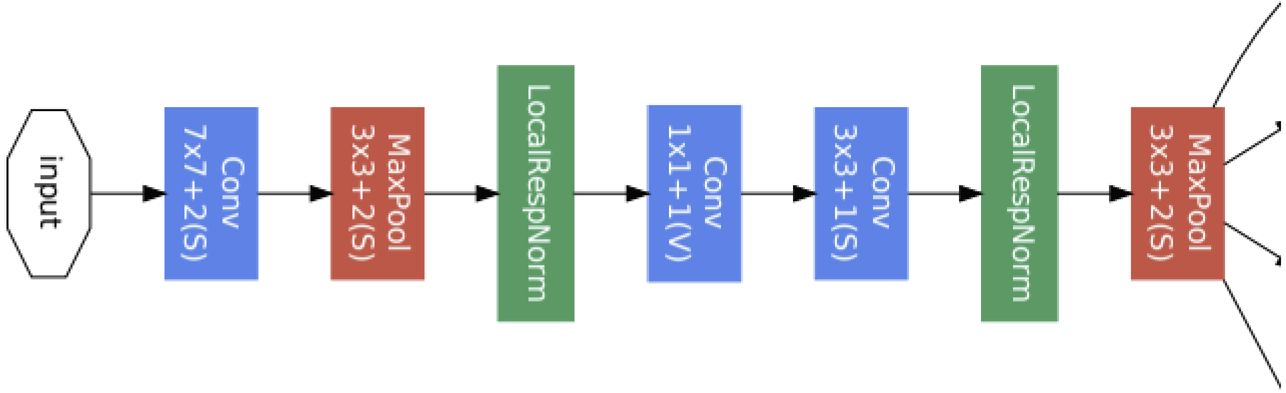
Figure 10. Inception module that reduces the grid-size while expands the filter banks. It is both cheap and avoids the representational bottleneck as is suggested by principle 1. The diagram on the right represents the same solution but from the perspective of grid sizes rather than the operations.

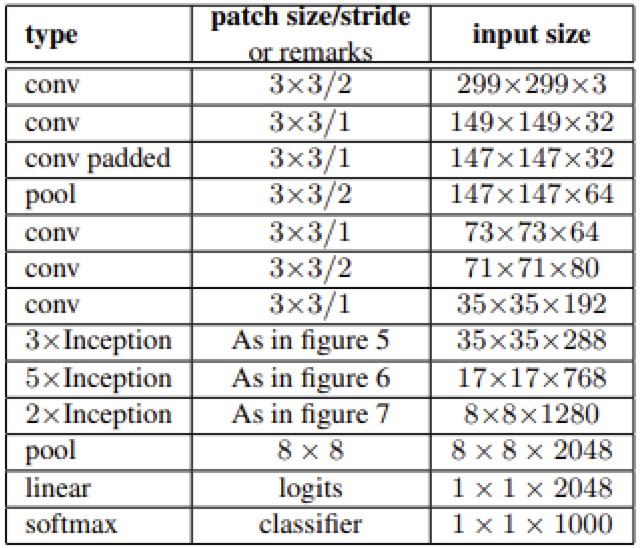
6. Inception-v2



type	**patch size/stride or remarks**	**input size**
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

6. Inception-v2





type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

6. Inception-v2

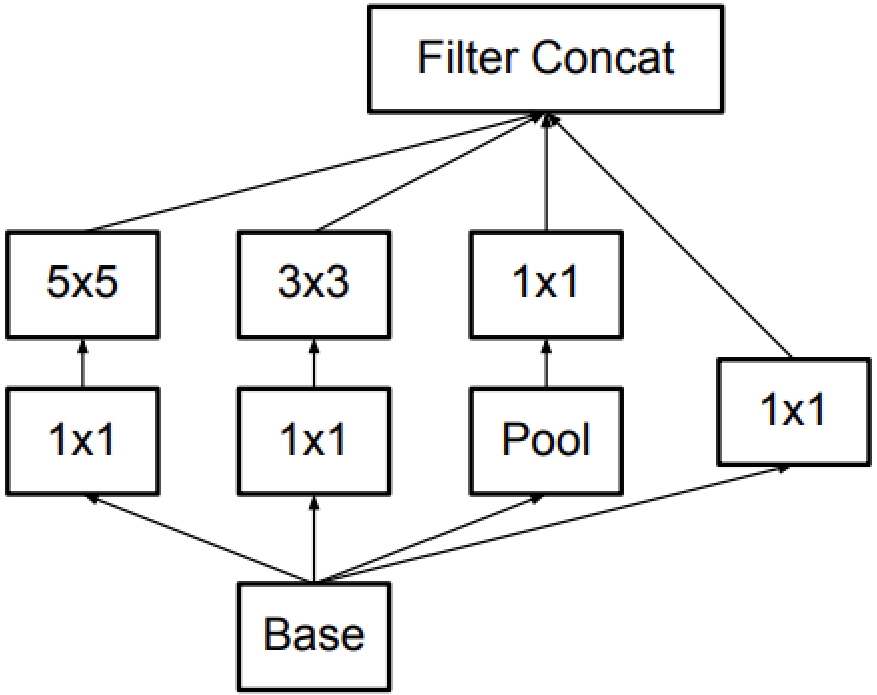
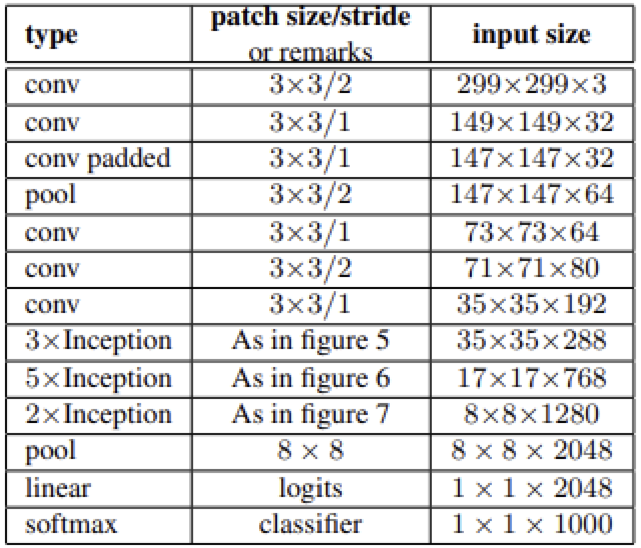


Figure 4. Original Inception module as described in [20].



type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

6. Inception-v2

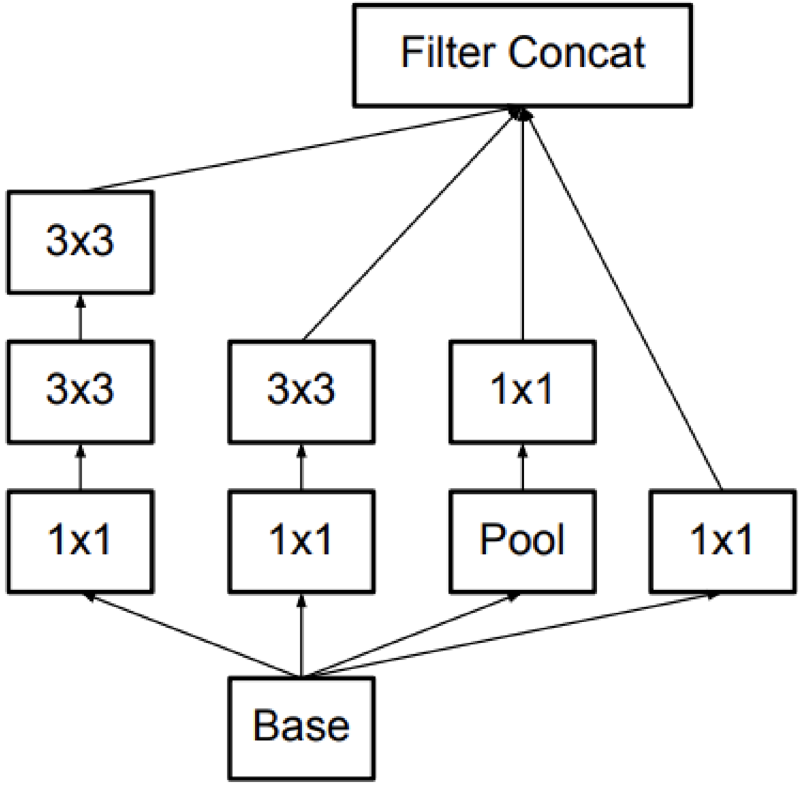
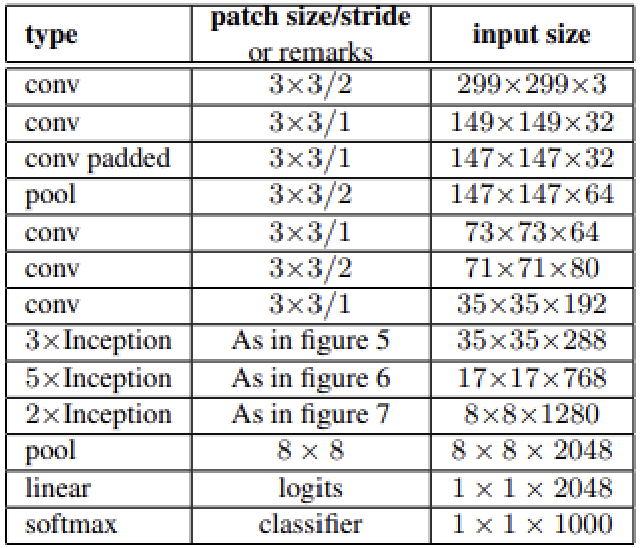


Figure 5. Inception modules where each 5 × 5 convolution is replaced by two 3 × 3 convolution, as suggested by principle 3 of Section 2.



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×Inception	As in figure 5	$35\times35\times288$
5×Inception	As in figure 6	$17\times17\times768$
2×Inception	As in figure 7	$8\times8\times1280$
pool	8×8	$8\times8\times2048$
linear	logits	$1\times1\times2048$
softmax	classifier	$1\times1\times1000$

6. Inception-v2

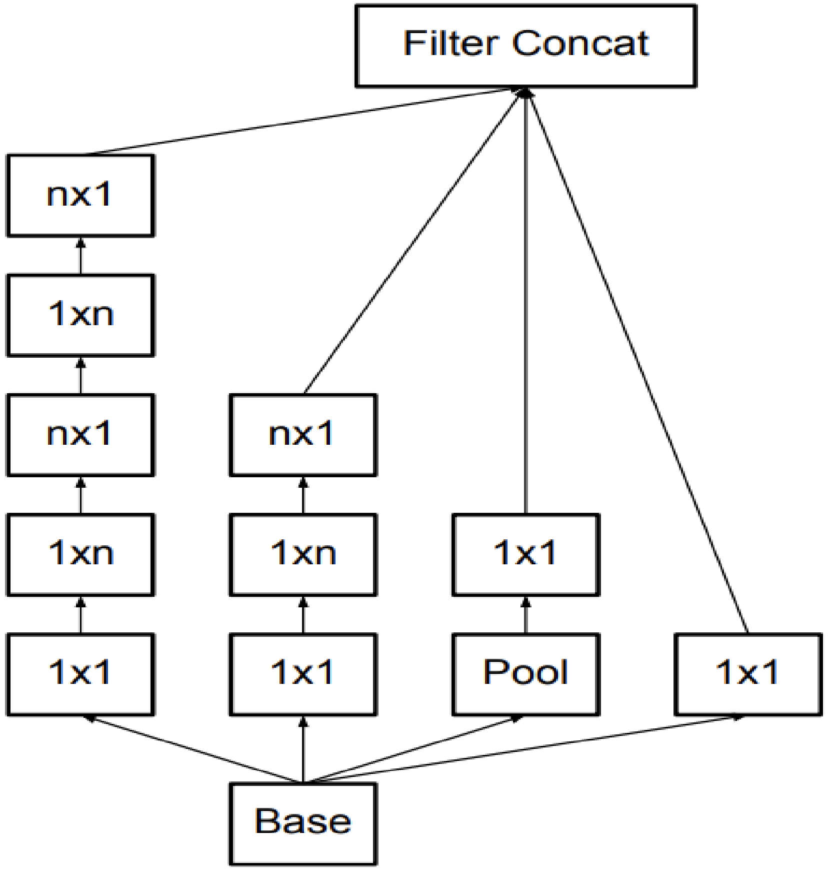
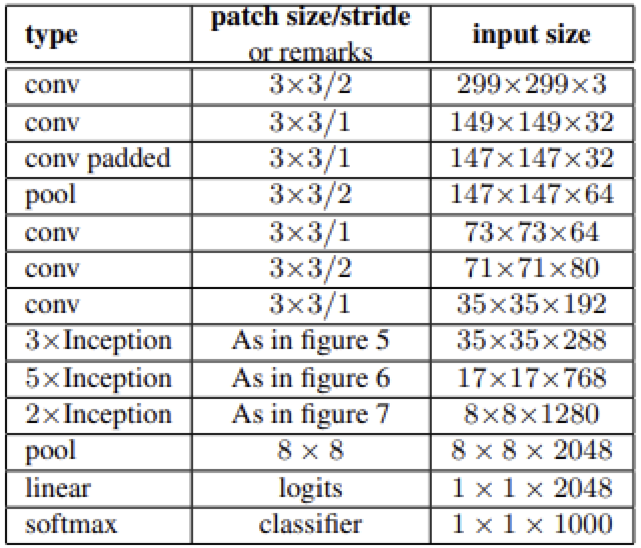


Figure 6. Inception modules after the factorization of the $n \times n$ convolutions. In our proposed architecture, we chose $n = 7$ for the 17×17 grid. (The filter sizes are picked using principle 3)



type	**patch size/stride or remarks**	**input size**
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

6. Inception-v2

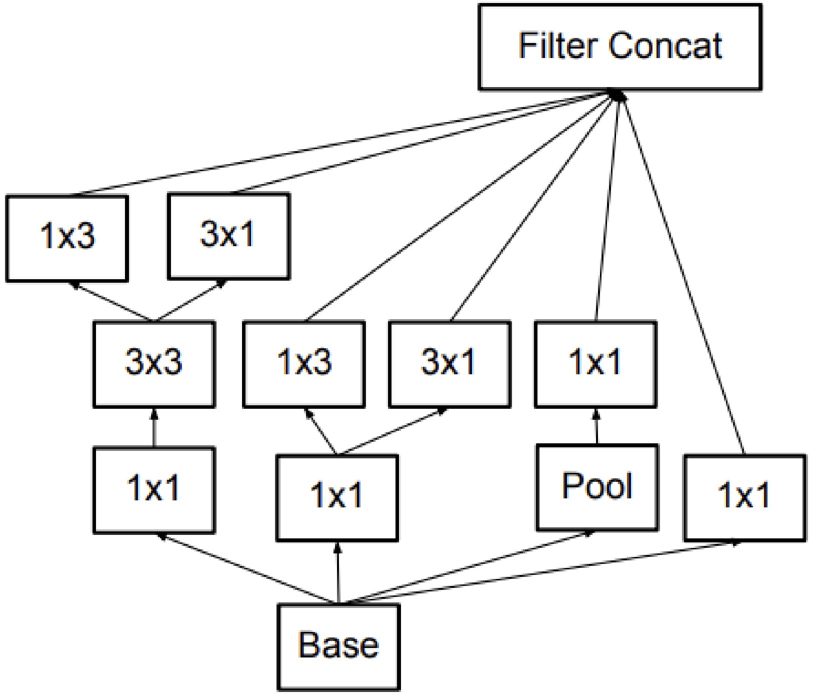
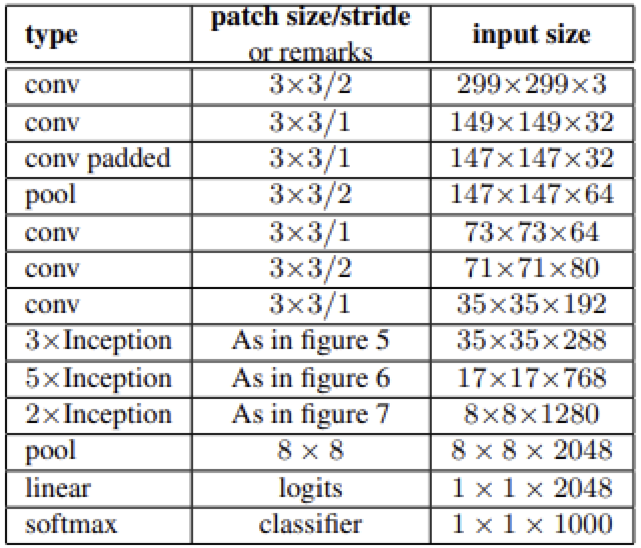


Figure 7. Inception modules with expanded the filter bank outputs. This architecture is used on the coarsest (8 × 8) grids to promote high dimensional representations, as suggested by principle 2 of Section 2. We are using this solution only on the coarsest grid, since that is the place where producing 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	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

6. Inception-v2

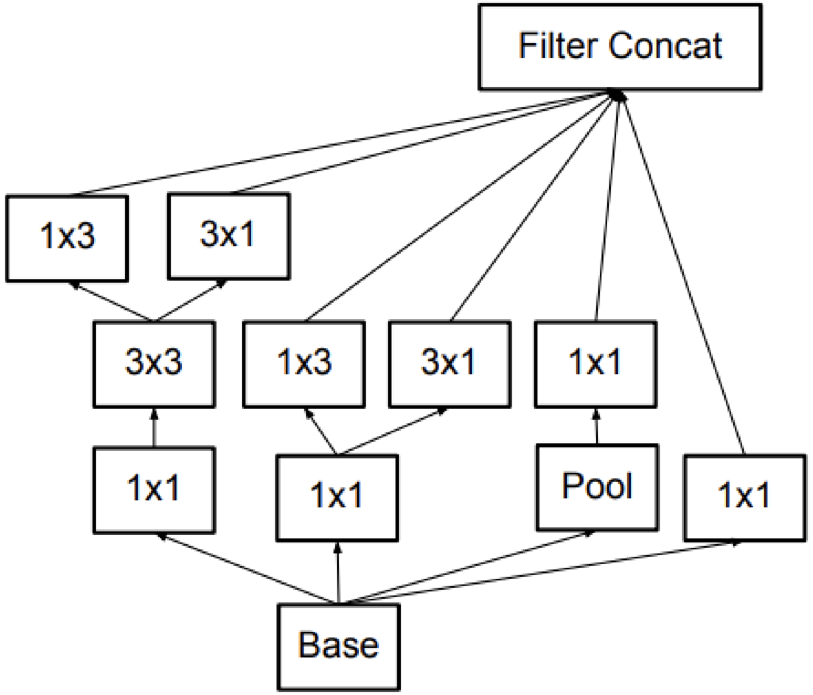
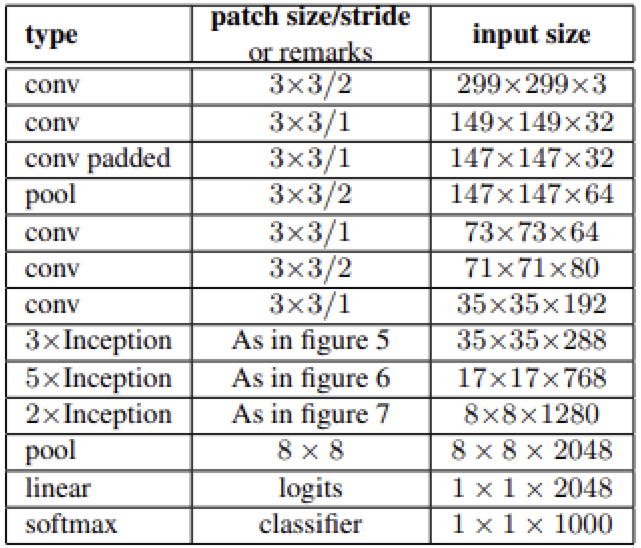


Figure 7. Inception modules with expanded the filter bank outputs. This architecture is used on the coarsest (8 × 8) grids to promote high dimensional representations, as suggested by principle 2 of Section 2. We are using this solution only on the coarsest grid, since that is the place where producing 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.

- Batch size: 32
- Epoch: 100
- Optimizer: (ealier) momentum, decay=0.9 / (best) RMSProp, decay=0.9, ϵ=1.0
- Learning rate: (initial) 0.045, decayed every two epoch using an exponential rate of 0.94



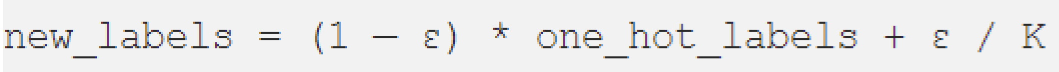
type	patch size/stride or remarks	input size
conv	$3\times3/2$	$299\times299\times3$
conv	$3\times3/1$	$149\times149\times32$
conv padded	$3\times3/1$	$147\times147\times32$
pool	$3\times3/2$	$147\times147\times64$
conv	$3\times3/1$	$73\times73\times64$
conv	$3\times3/2$	$71\times71\times80$
conv	$3\times3/1$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

7. Model Regularization via Label Smoothing

(1) 논문에서 소개한 Label Smoothing 방식은 모델의 일반화 성능을 높였다.

(2) [0, 1, 0, 0] -> [0.025, 0.925, 0.025, 0.025]

(3) 정답에 대한 확신을 감소시켜 일반화된 성능을 나타낼 수 있다.

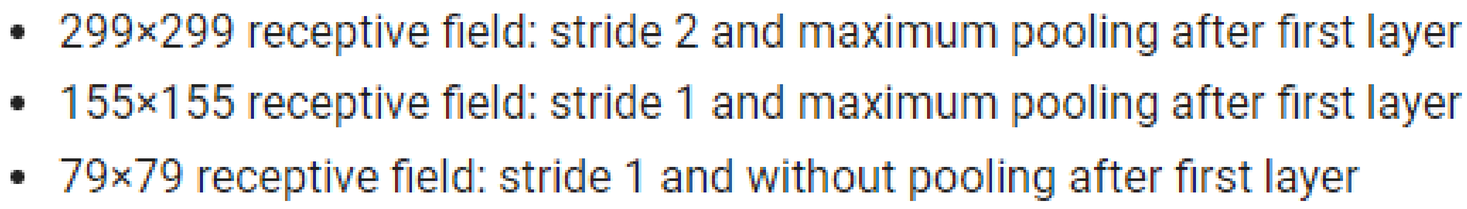


```
new_labels = (1 — ε) * one_hot_labels + ε / K
```

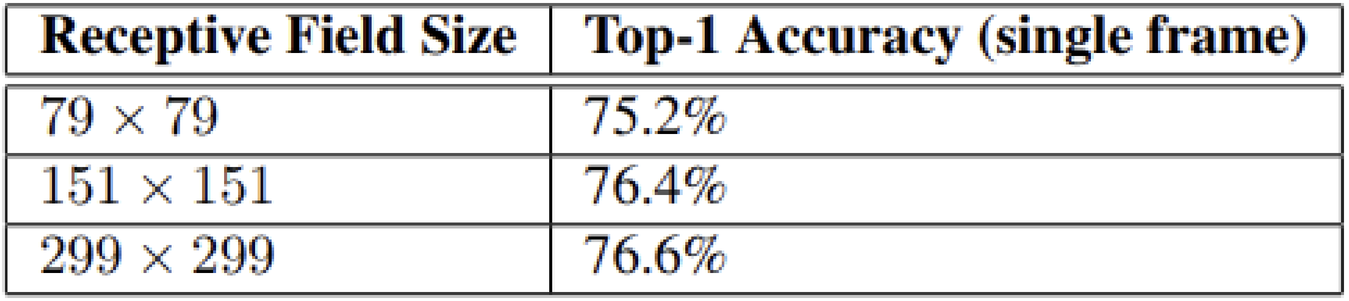
Performence on Lower Resolution Input

(1) post-classification – ex) Multibox, R-CNN

(2) 일반적으로 higher resolution receptive field를 사용하면 성능 향상이 가능하다.

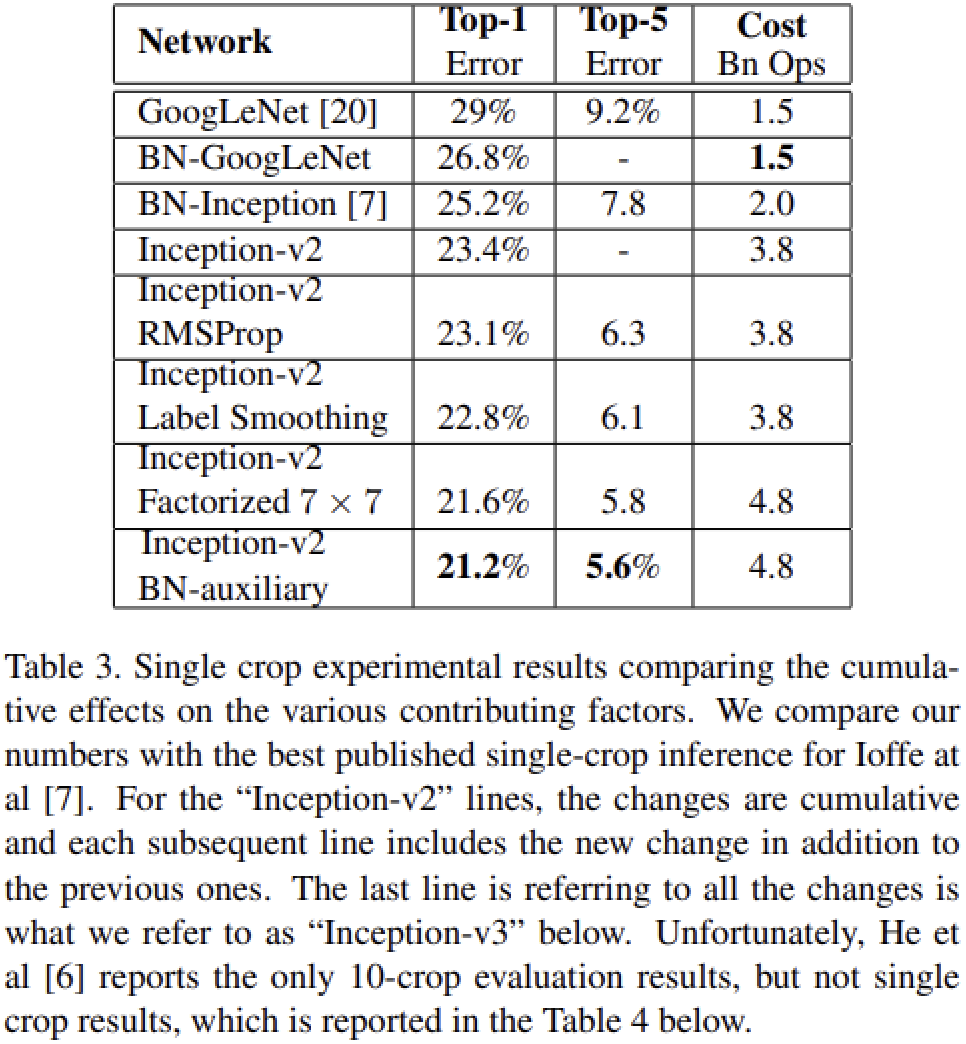


- 299×299 receptive field: stride 2 and maximum pooling after first layer
- 155×155 receptive field: stride 1 and maximum pooling after first layer
- 79×79 receptive field: stride 1 and without pooling after first layer



Receptive Field Size	**Top-1 Accuracy (single frame)**
79 × 79	75.2%
151 × 151	76.4%
299 × 299	76.6%

Experimental Results and Comparisons

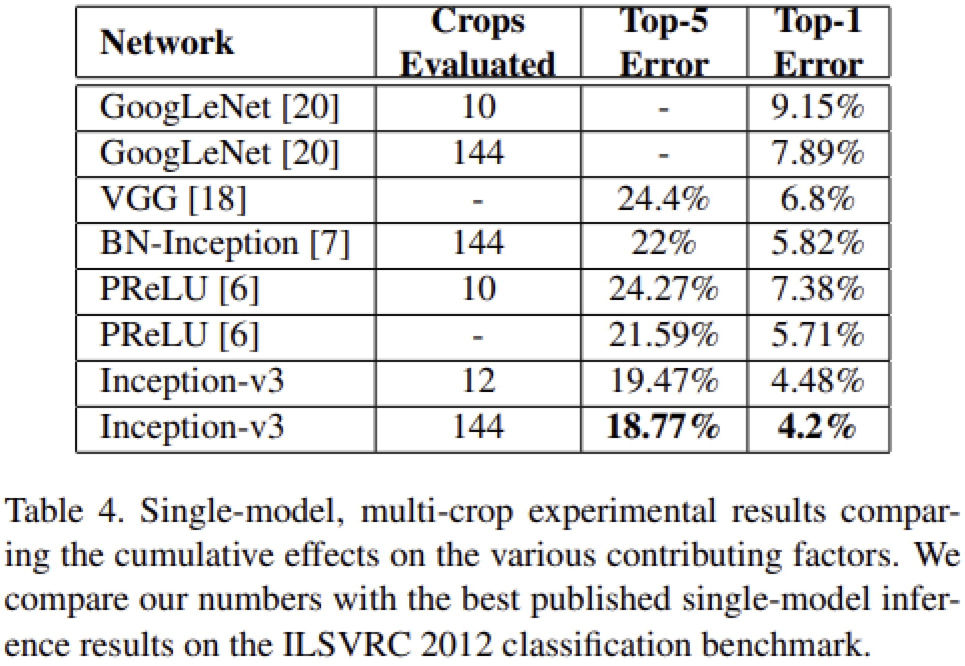


Network	**Top-1** Error	**Top-5** Error	**Cost** Bn Ops
GoogLeNet [20]	29%	9.2%	1.5
BN-GoogLeNet	26.8%	-	**1.5**
BN-Inception [7]	25.2%	7.8	2.0
Inception-v2	23.4%	-	3.8
Inception-v2 RMSProp	23.1%	6.3	3.8
Inception-v2 Label Smoothing	22.8%	6.1	3.8
Inception-v2 Factorized 7 × 7	21.6%	5.8	4.8
Inception-v2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

Table 3. Single crop experimental results comparing the cumulative effects on the various contributing factors. We compare our numbers with the best published single-crop inference for Ioffe at al [7]. For the "Inception-v2" lines, the changes are cumulative and each subsequent line includes the new change in addition to the previous ones. The last line is referring to all the changes is what we refer to as "Inception-v3" below. Unfortunately, He et al [6] reports the only 10-crop evaluation results, but not single crop results, which is reported in the Table 4 below.

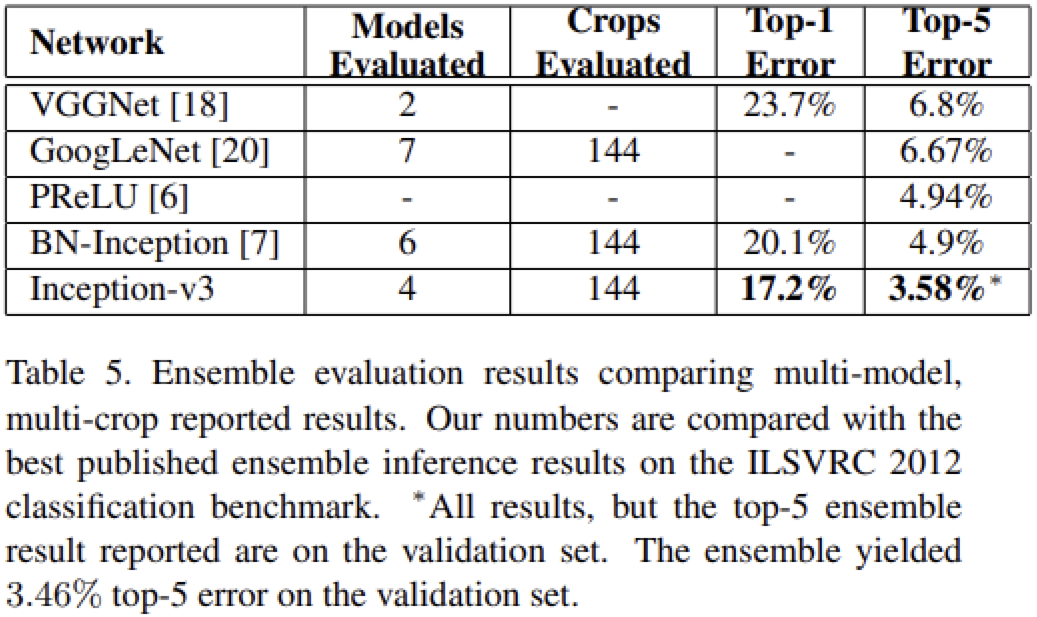
- Inception-v2에 위에서 설명한 각 방식을 적용한 single-crop 성능이다.
- BN-auxiliary는 Conv layer뿐만 아니라 Fc layer에도 BN을 적용한다.
- Inception-v3는 아래 기법들을 모두 적용한 Inception-v2이다.

Experimental Results and Comparisons



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%**

Table 4. Single-model, multi-crop experimental results comparing the cumulative effects on the various contributing factors. We compare our numbers with the best published single-model inference results on the ILSVRC 2012 classification benchmark.



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%***

Table 5. Ensemble evaluation results comparing multi-model, multi-crop reported results. Our numbers are compared with the best published ensemble inference results on the ILSVRC 2012 classification benchmark. *All results, but the top-5 ensemble result reported are on the validation set. The ensemble yielded 3.46% top-5 error on the validation set.