Inception-v2, v3

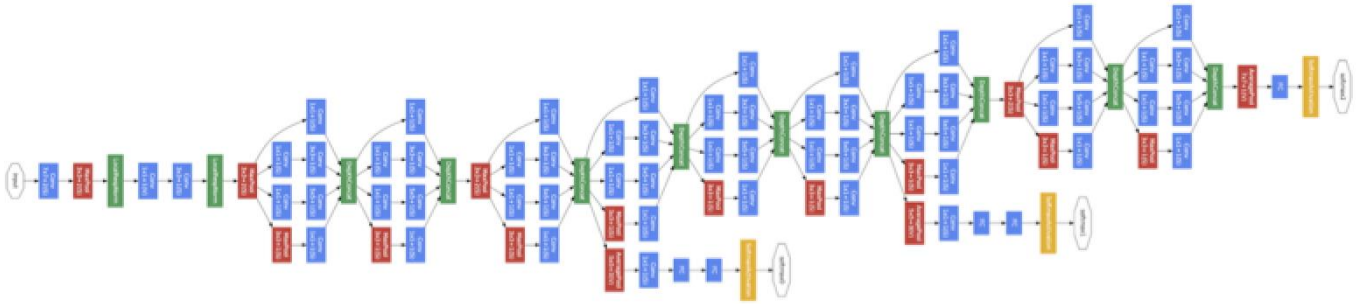
Rethinking the Inception Architecture for Computer Vision

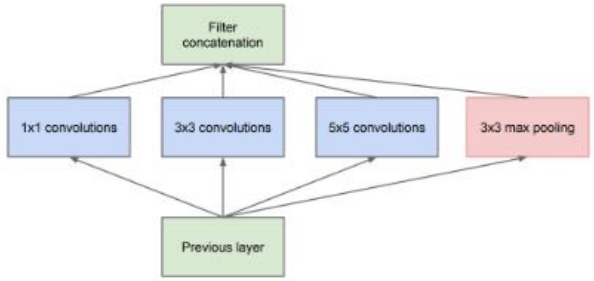
230221 / 7기 최명헌

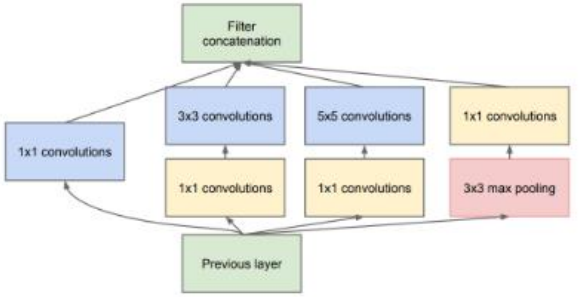
0. 목차

1. Introduction

- **Rethinking : 이전 CNN 연구**
  - VGGNet, GoogLeNet 은 비슷하게 좋은 성능을 보임.
  - Classification 성능에서의 향상이 다른 task 에서의 성능 향상으로 전환되었음
    - Architectural improvements 가 점점 더 visual feature 를 잘 학습할 수 있게 된 것.

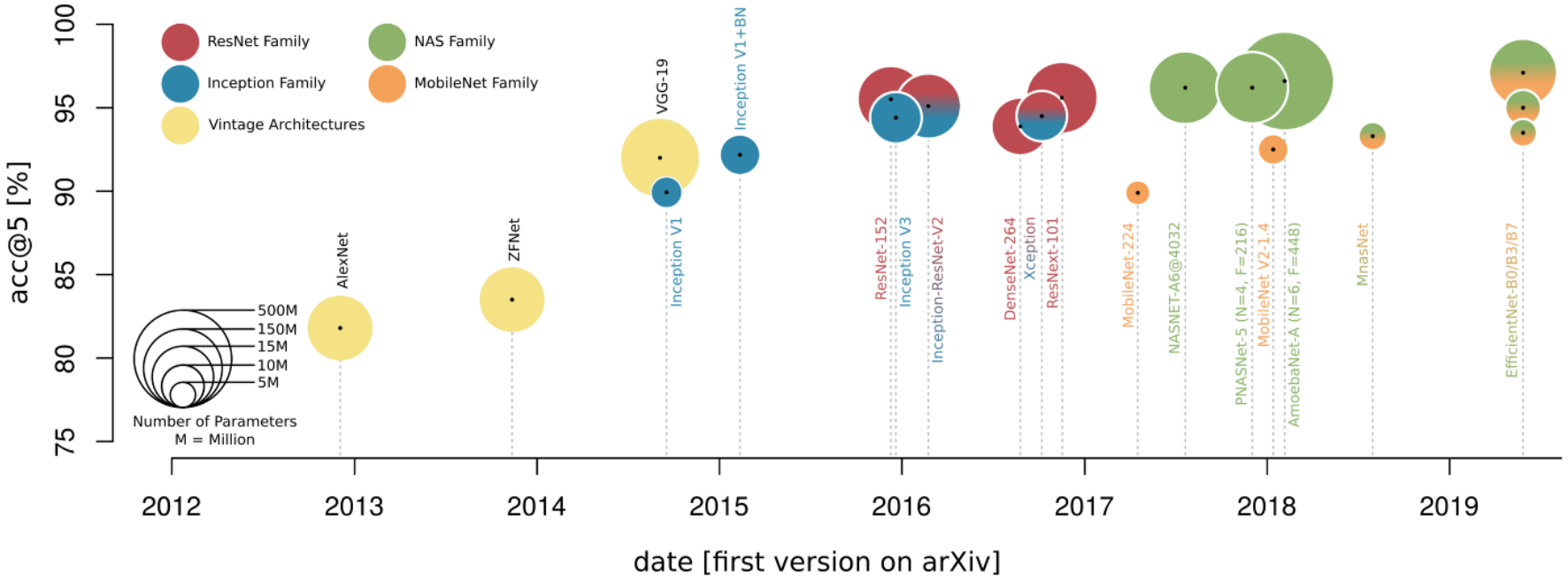






1. Introduction

- **Rethinking : 이전 CNN 연구**
  - VGGNet 은 구조를 단순화하려 했지만 이는 high-cost computation problem 으로 이어짐
  - GoogLeNet 은 Inception 구조를 통해 parameter 를 줄이면서 performance 를 올림
  - 이러한 parameter 를 줄이려는 노력은 mobile setting 등 다양한 환경에서 사용할 수 있게 함



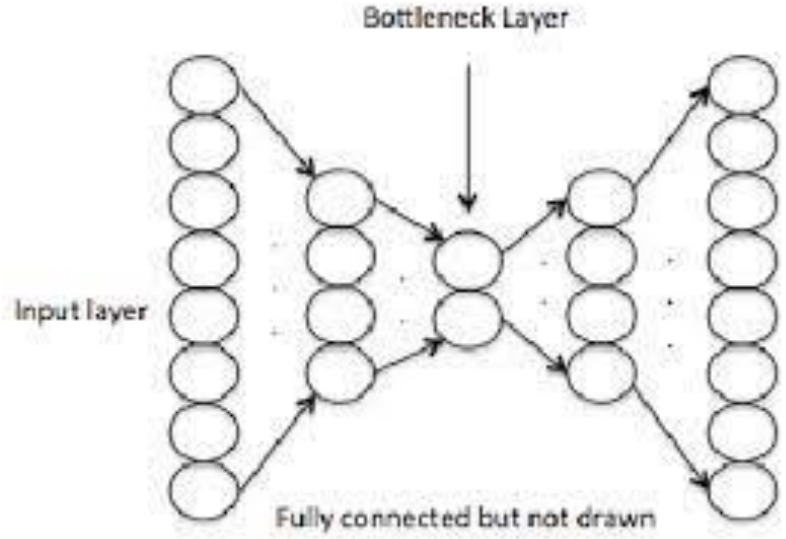
1. Introduction

- **Rethinking : 이전 CNN 연구**
  - 하지만 Inception 은 complexity가 높아서 network 구조를 변경하는 것이 힘듦
    - 만약 architecture 가 확장되면 Inception 으로 인해 얻는 computational gain 은 확 줄어듦
    - 논문에서도 GoogLeNet 에서 다양하게 디자인을 바꾸기 위해 알아야 할 요소에 대한 언급이 없었음
    - Example : filter bank size 를 2배 늘리면 parameter 는 4배 늚

- **In this paper**
  - Conv network 를 효과적으로 scaling up 하기 위한 general principle, optimization idea 소개
    - Inception 모듈 말고 다른 network 에도 적용 가능함
    - 하지만 Inception 모듈이 유연하여 잘 적용되기 때문에 Inception 모듈에서 이러한 방법이 잘 쓰임

2. General Design Principles

- 1. Network 초기의 bottleneck 을 피해라.
  - FFN 은 input layer 에서 classifier 또는 regressor 로 연결되는 비순환 그래프로 표현될 수 있음
  - FFN 은 input 에서 output 까지 갈 때에 representation size 가 서서히 감소해야 하는데,
    초기의 bottleneck 은 input 정보를 많이 손실시키기 때문에 이는 안 좋은 영향을 끼친다.



- 2. 고차원의 representation 은 network 안에서 지역적으로 얻기 쉽다.
  - Activations per tile 을 늘리면 disentangled feature 를 얻기 쉬워진다

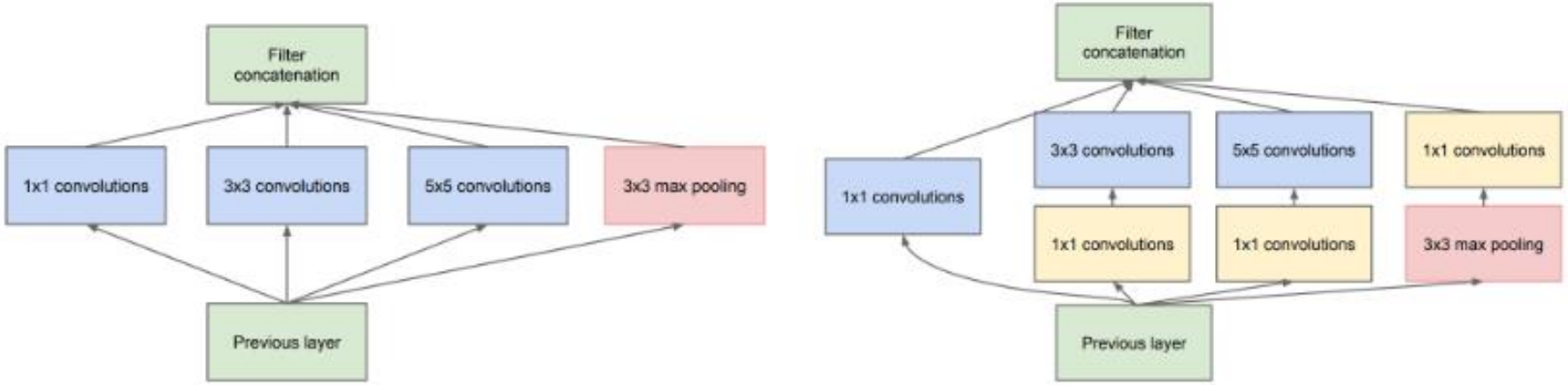
    = filter map 을 많이 사용하면 다양한 high dimensional representation feature 를 얻기 쉽다.
  - 이는 학습 속도를 빠르게 해준다.

- 3. Spatial aggregation 은 표현력 감소 없이 저차원에서 이루어질 수 있다.
  - 3x3 convolution 같은 연산을 하기 전에 심각한 부작용 없이 입력 표현의 차원을 줄일 수 있음
  - 적절한 차원 축소는 학습 속도에 좋은 영향을 끼친다.

- 4. 네트워크의 깊이와 너비의 균형을 잘 조절해야 한다.
  - 네트워크 각 층의 필터의 개수와 깊이를 잘 조절함으로써 네트워크의 성능이 최적화 될 수 있음
  - 네트워크의 너비와 깊이를 동시에 늘리는 것은 성능 향상에 도움이 될 수 있음
  - 둘을 병렬적으로 같이 늘려야 계산량에서도 optimal improvement 을 도달할 수 있음

3. Factorizing Convolutions with Large Filter Size

- GoogLeNet 에서는 dimension reduction 을 통해 많은 이득을 보았다.
- 이는 factorizing convolution 을 계산상 효과적인 방법으로 수행한 것으로 볼 수 있다.
- 본 논문에서는 GoogLeNet 에서의 방법 외에 computation efficiency 를 늘릴 수 있는 방법을 고안한다.



3. Factorizing Convolutions with Large Filter Size

1. Factorization into smaller convolutions

큰 사이즈의 filter 로 구성된 convolution 은 계산상에 비효율적임.

그니까 5x5 convolution 쓸 바에는 3x3 convolution 2개 쓰는 게 더 좋음.

- Q1. 이러한 replacement 는 표현력 손실을 일으키는가?
- Q2. 계산의 linear part 에 대한 factorizing 이 목적이라면 first layer 에서 linear activation 을 유지해야 하는가?

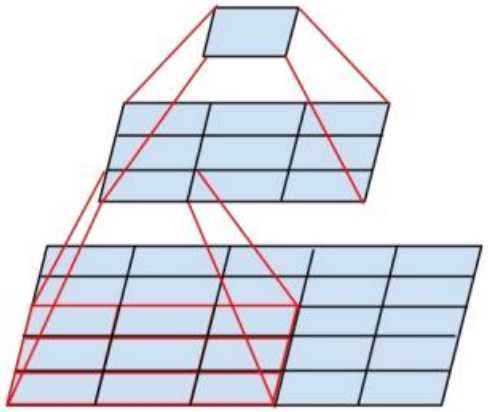
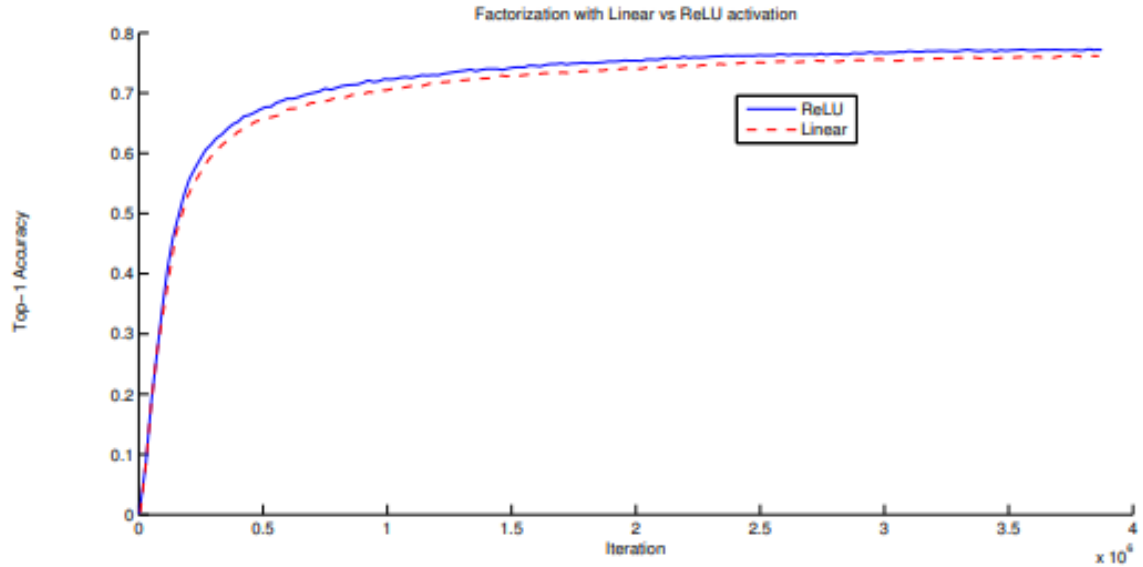


Figure 1. Mini-network replacing the 5×5 convolutions.



3. Factorizing Convolutions with Large Filter Size

1. Factorization into smaller convolutions

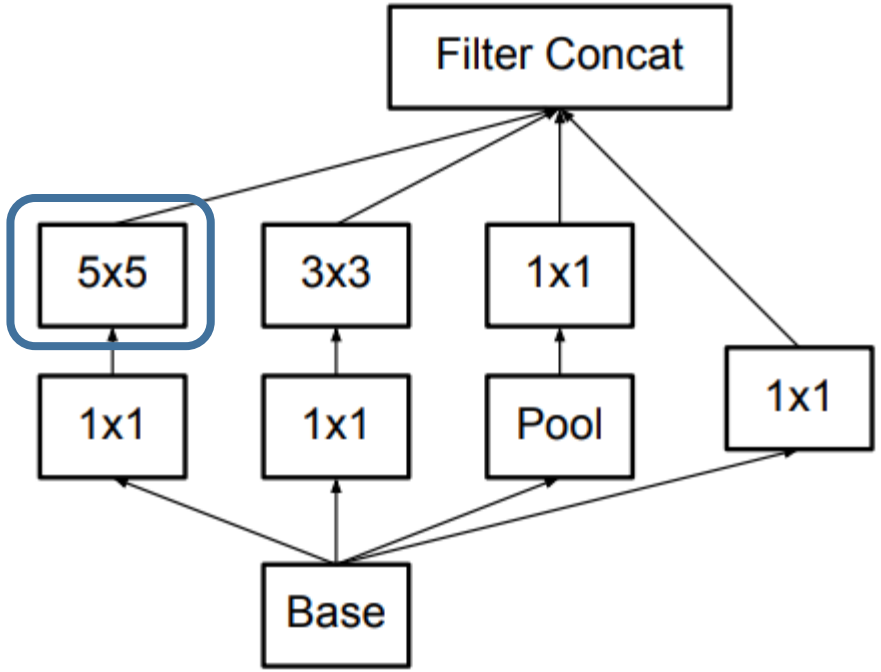


Figure 4. Original Inception module as described in [20].

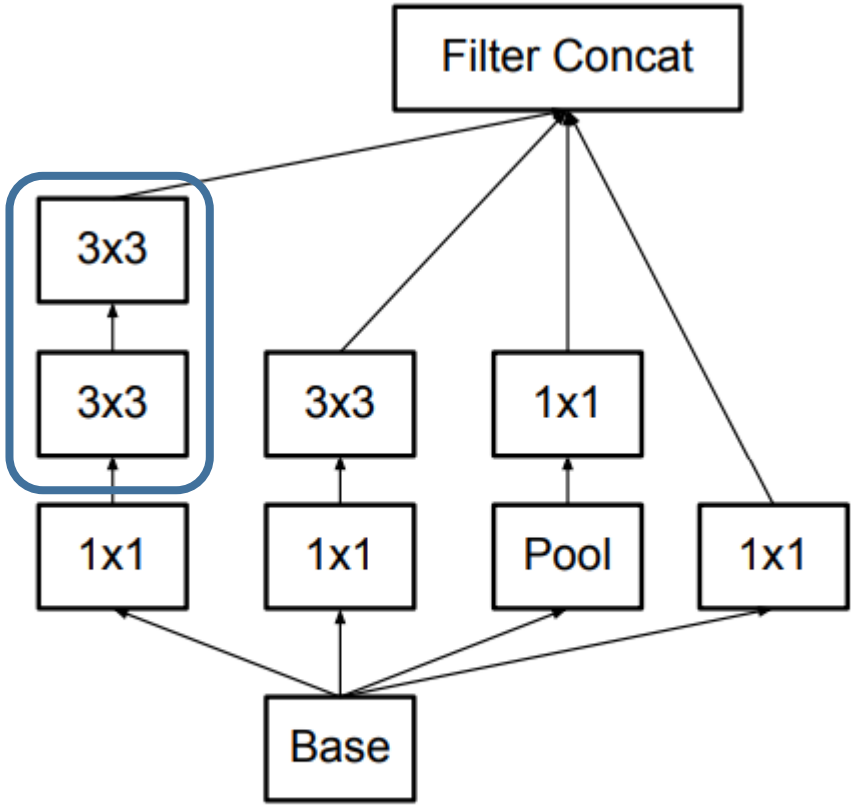


Figure 5. Inception modules where each 5×5 convolution is replaced by two 3×3 convolution, as suggested by principle 3 of Section 2.

3. Factorizing Convolutions with Large Filter Size

2. Spatial Factorization into Asymmetric Convolutions

앞에서 큰 사이즈의 convolution filter 를 작은 거 여러 개 쓰면 좋다 했는데, 3x3 도 잘게 나누어서 사용하면 안됨?

2x2 보다 nx1 짜리 asymmetric convolution 사용하는 것이 더 좋음!

즉, 3x3 filter 대신 3x1 filter, 1x3 filter 를 사용하자

하지만 실험을 통한 결과로써, 초반부의 layer 에서는 잘 작동하지 않지만, 중 후반부의 layer 에서는 잘 작동함.

→ Feature map 사이즈가 (12~20 일 때)

이때는 1x7, 7x1 filter 를 사용하는 것이 더 효과적이었음

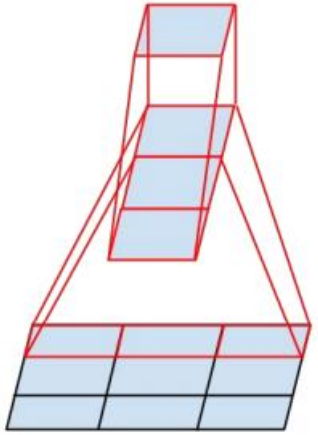


Figure 3. Mini-network replacing the 3×3 convolutions. The lower layer of this network consists of a 3×1 convolution with 3 output units.

3. Factorizing Convolutions with Large Filter Size

2. Spatial Factorization into Asymmetric Convolutions

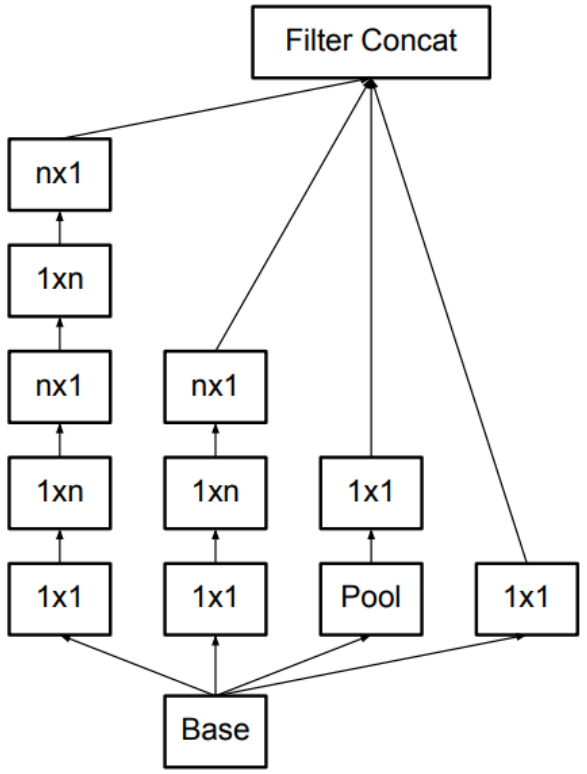


Figure 6. Inception modules after the factorization of the $n \times n$ convolutions. In our proposed architecture, we chose $n = 7$ for the 17×17 grid. (The filter sizes are picked using principle 3)

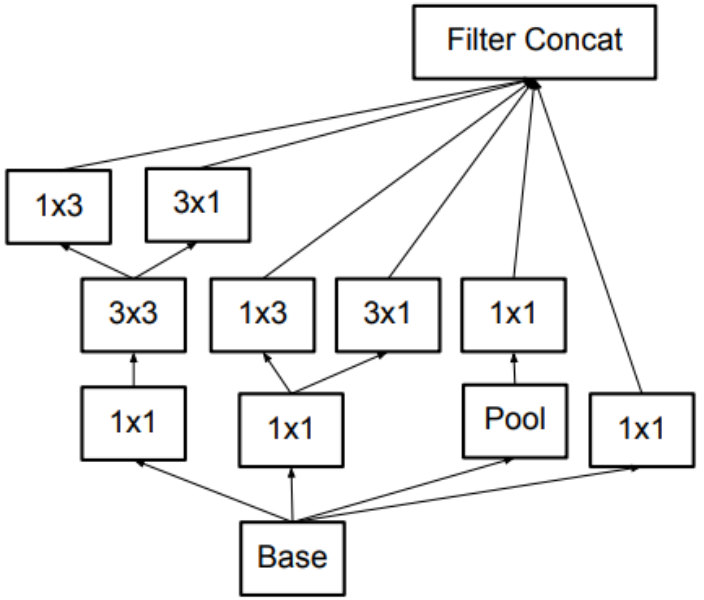


Figure 7. Inception modules with expanded the filter bank outputs. This architecture is used on the coarsest (8×8) grids to promote high dimensional representations, as suggested by principle 2 of Section 2. We are using this solution only on the coarsest grid, since that is the place where producing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

4. Utility of Auxiliary Classifiers

GoogLeNet 에서는 Auxiliary Classifier 를 사용하여 깊은 DNN 의 수렴을 도왔다.

→ 낮은 층의 layer 에서 제때 유용한 gradient 를 받고, 수렴 속도를 향상시키기 위함. 결국 안정된 학습과 더 나은 수렴을 보여줌

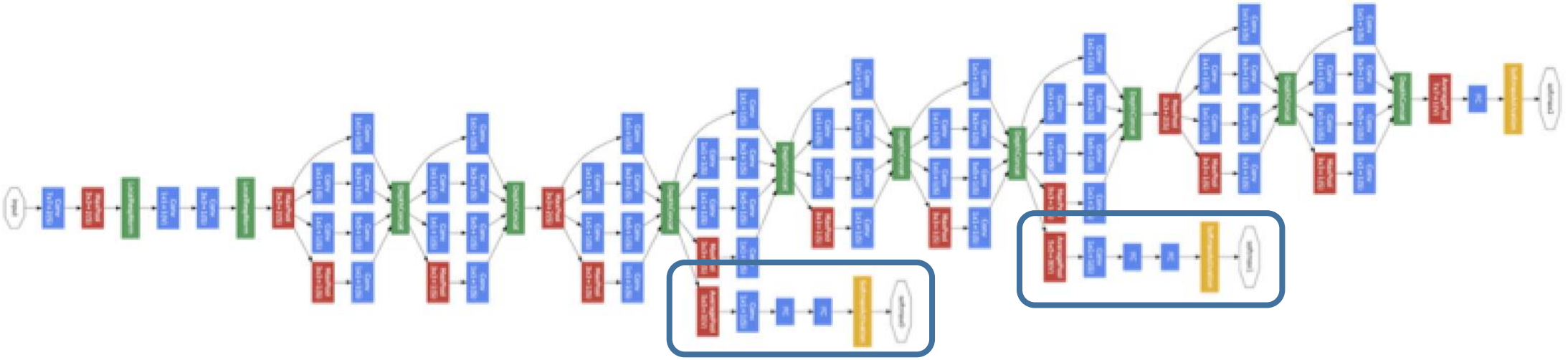
하지만 이런 Auxiliary Classifier 는 학습 초기에는 도움이 되지 않음.

→ 정확도가 높아지기 전에는 있는 것과 없는 것에 성능 차이 X, 학습이 끝날 무렵 auxiliary classifier 가 있을 때 조금 더 높은 성능

GoogLeNet 에서는 2개의 auxiliary classifier 를 사용했지만 초기 layer의 하나 없어도 괜찮았음.

→ 이는 GoogLeNet 에서 주장했던 auxiliary classifier 가 low-level feature 를 찾는데 도움이 된다는 가설과 반대됨

→ 위 논문의 저자들은 auxiliary classifier 가 regularizer 로써의 역할을 한다고 주장함



4. Utility of Auxiliary Classifiers

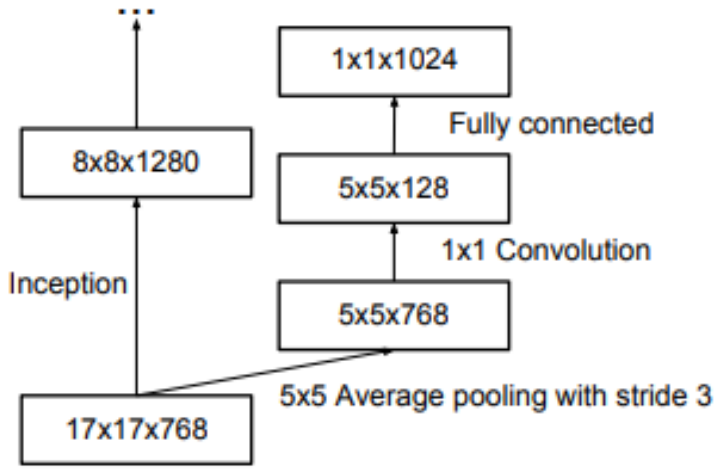


Figure 8. Auxiliary classifier on top of the last 17×17 layer. Batch normalization[7] of the layers in the side head results in a 0.4% absolute gain in top-1 accuracy. The lower axis shows the number of itertions performed, each with batch size 32.

5. Efficient Grid Size Reduction

전통적으로 conv net 은 feature map 의 크기를 줄이기 위해 pooling 을 사용했음.

Bottleneck 현상을 막기 위해 pooling layer 이전에 dimension (=channel) 을 늘렸음.

Ex. 만약 d x d 크기의 k 개의 필터가 있을 때 pooling layer 이전에 2k 개의 filter 로 늘려야 하므로 $2d^2k^2$ 의 연산량이 필요함.

다른 방법으로는 pooling layer 와 convolution layer 의 순서를 바꾸는 방법이 있음.

Ex. 이는 $2\frac{d^2}{2}k^2$ 의 연산량만을 필요로 하기 때문에 연산량의 이득을 얻지만, 병목현상으로 인해 덜 효과적인 네트워크 구조임.

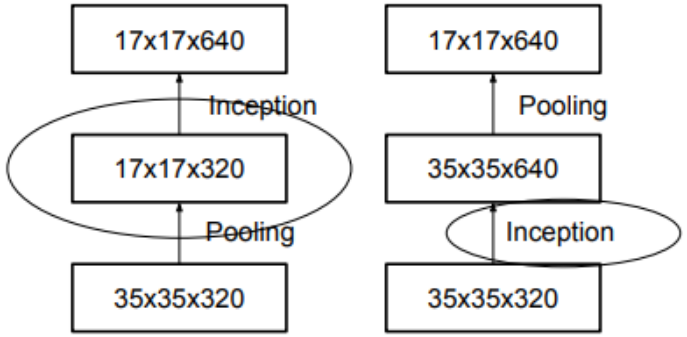


Figure 9. Two alternative ways of reducing the grid size. The solution on the left violates the principle 1 of not introducing an representational bottleneck from Section 2. The version on the right is 3 times more expensive computationally.

5. Efficient Grid Size Reduction

그래서 본 논문에서는 병목 현상을 없애면서 연산량을 줄일 수 있는 새로운 방법을 제안함.

Stride 가 2인 Pooling layer, Convolution layer 을 병렬로 계산하여 concatenate 함.

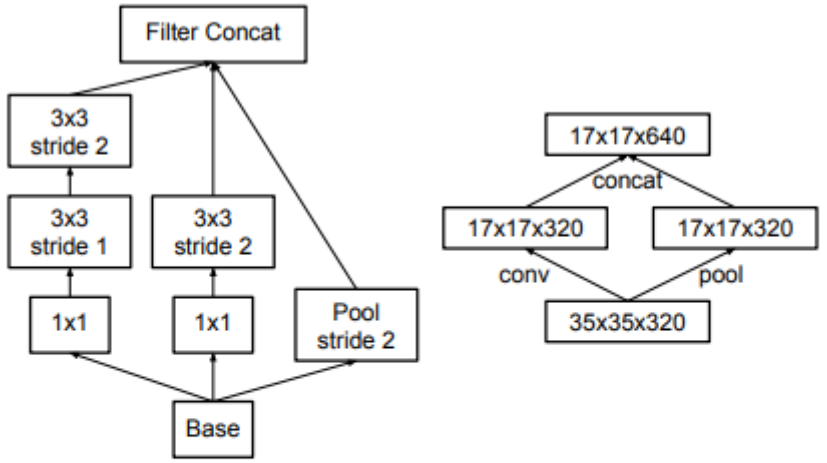


Figure 10. Inception module that reduces the grid-size while expands the filter banks. It is both cheap and avoids the representational bottleneck as is suggested by principle 1. The diagram on the right represents the same solution but from the perspective of grid sizes rather than the operations.

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

7x7 convolution → three 3x3 convolutions

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

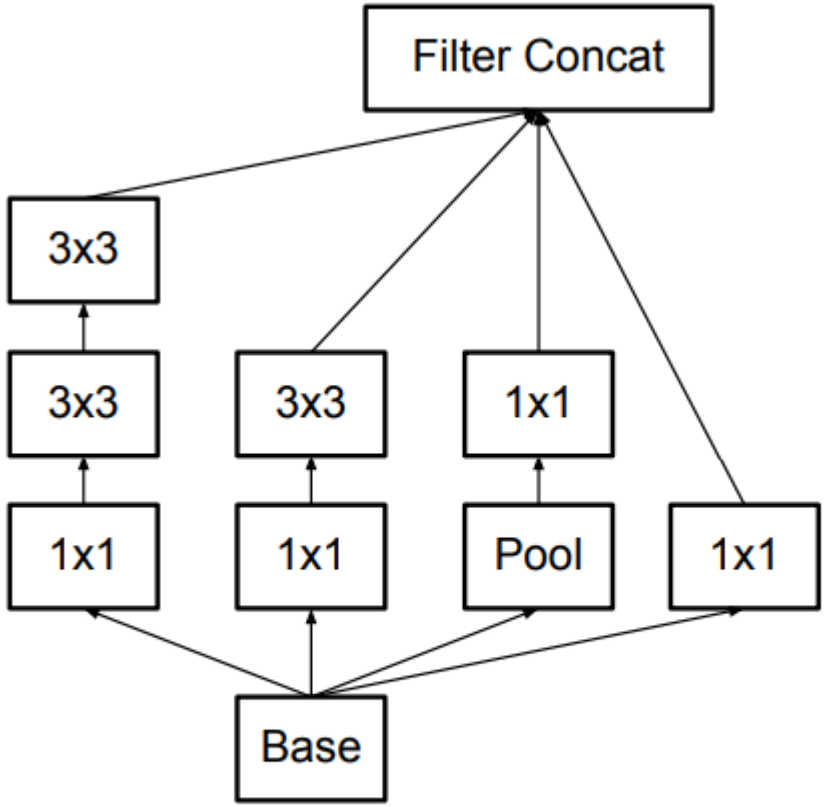


Figure 5. Inception modules where each 5×5 convolution is replaced by two 3×3 convolution, as suggested by principle 3 of Section 2.

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

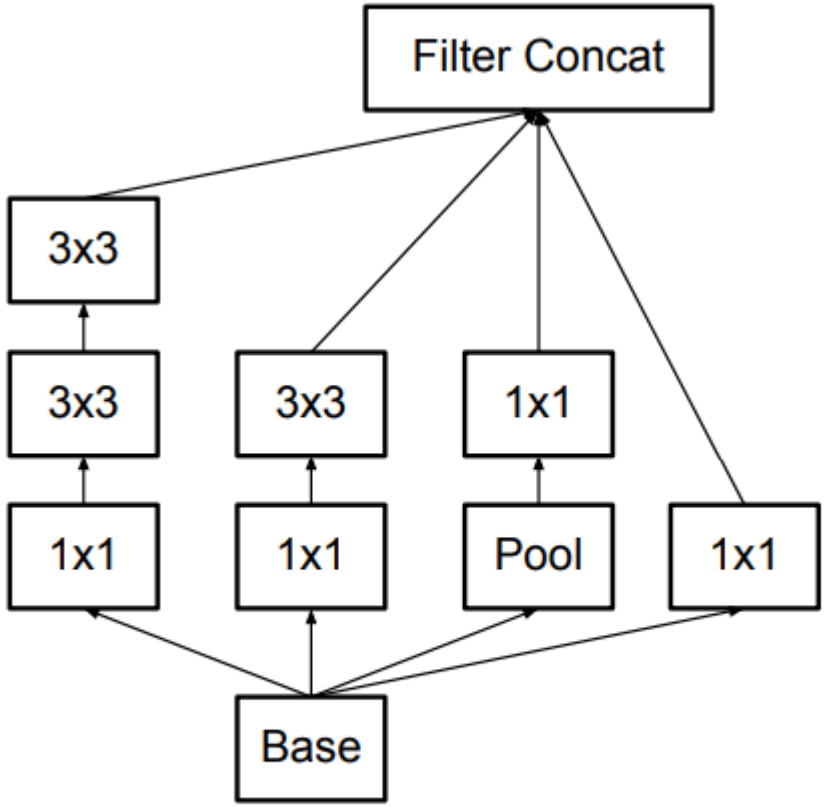


Figure 5. Inception modules where each 5×5 convolution is replaced by two 3×3 convolution, as suggested by principle 3 of Section 2.

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

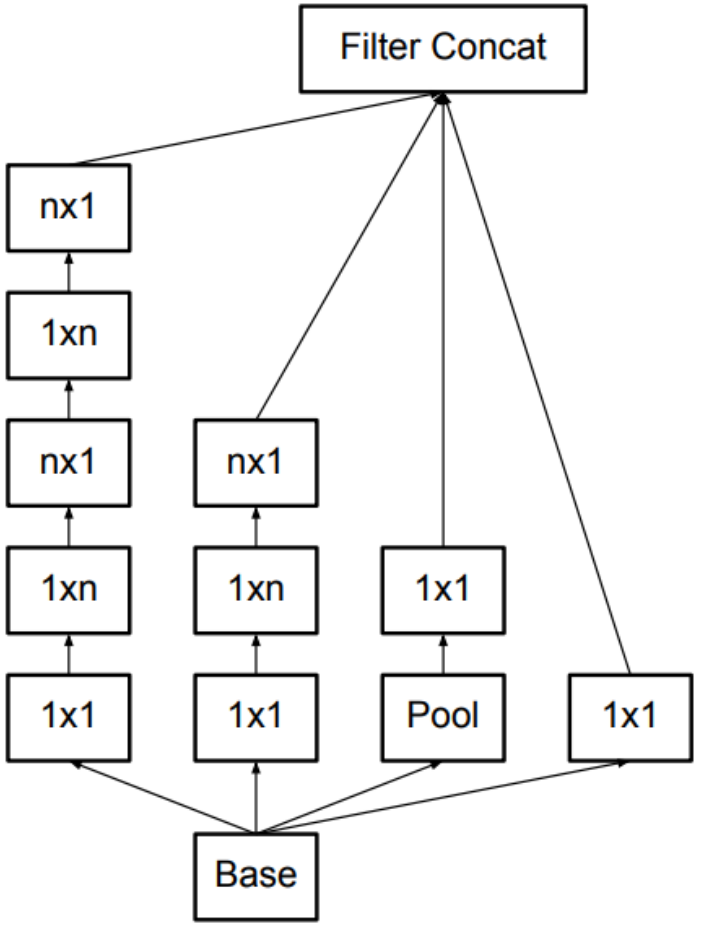


Figure 6. Inception modules after the factorization of the $n \times n$ convolutions. In our proposed architecture, we chose $n = 7$ for the 17×17 grid. (The filter sizes are picked using principle 3)

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

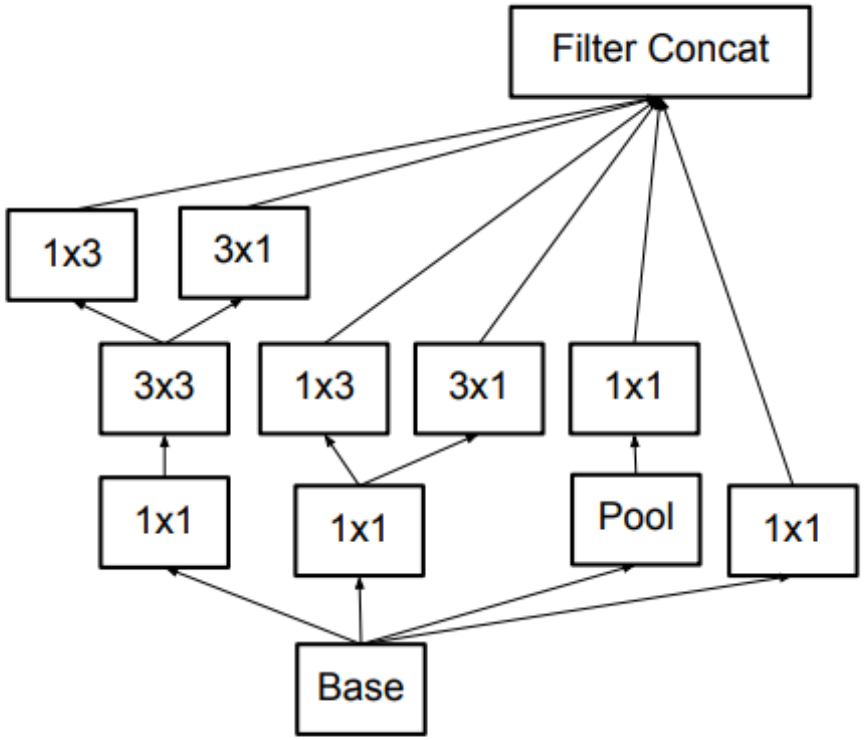
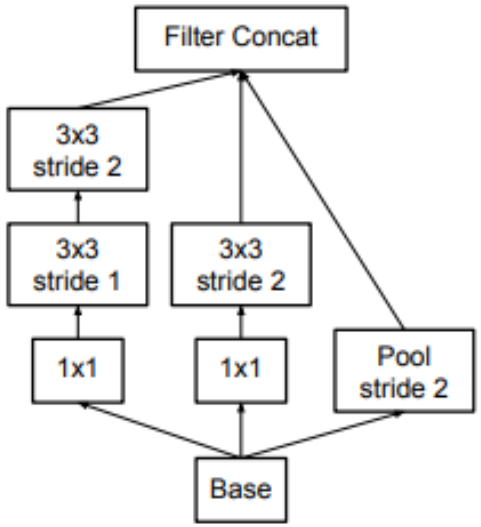


Figure 7. Inception modules with expanded the filter bank outputs. This architecture is used on the coarsest (8 × 8) grids to promote high dimensional representations, as suggested by principle 2 of Section 2. We are using this solution only on the coarsest grid, since that is the place where producing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

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



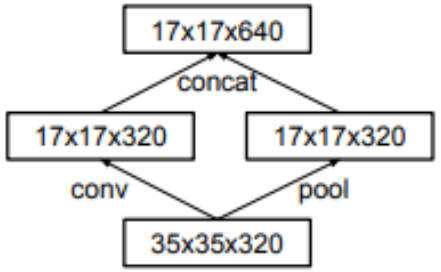


Figure 10. Inception module that reduces the grid-size while expands the filter banks. It is both cheap and avoids the representational bottleneck as is suggested by principle 1. The diagram on the right represents the same solution but from the perspective of grid sizes rather than the operations.

7. Model Regularization via Label Smoothing

기존 softmax function 의 문제!

1. over-fitting 의 결과를 얻을 수 있다.
2. 가장 큰 logit 과 나머지 logit 들 간의 차이가 크다. 이것은 모델이 적응하는 능력을 줄인다.

➔ 모델이 over-confident 해진다.

본 논문에서는 over-confident 해지는 것을 방지하는 mechanism(= label smoothing)을 소개한다.

Log-likelihood 를 최대화 하려는 것이 아닌, 모델을 좀 더 적응력있고 정규화하려는 시도이다.

$$q'(k|x) = (1-\epsilon)\delta_{k,y} + \epsilon u(k)$$

$$H(q', p) = -\sum_{k=1}^{K} \log p(k) q'(k) = (1-\epsilon) H(q, p) + \epsilon H(u, p)$$

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

8. Training Methodology

Batch size 32 for 100 epochs

Momentum with a decay 0.9

RMSProp with decay 0.9, epsilon 1.0 and learning rate 0.045

Gradient clipping with threshold 2.0 (Gradient exploding 을 방지하려는 목적)

Evaluation 은 계산 된 parameter 의 running average 를 사용

9. Performance on Lower Resolution Input

Detection → low resolution 에서의 classification 문제를 다뤄야 함

➔ High resolution image 를 input 으로 받자!

➔ 모델을 그대로 유지한 채 이미지의 사이즈만 늘린다면 이는 모델의 성능을 안 좋게 할 수 있다.

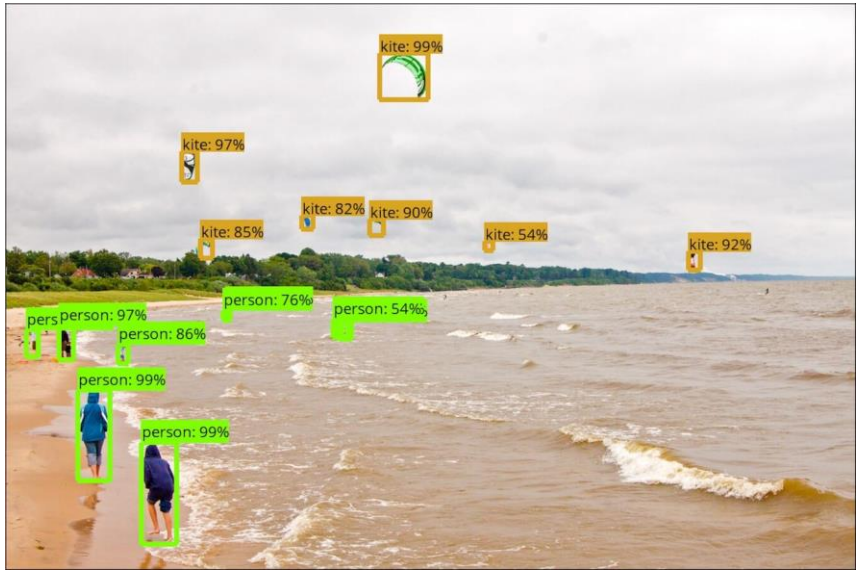
왜냐하면 모델의 capacity 는 일정하지만 받아들이는 정보의 양이 많아지기 때문이다.

➔ 첫 layer 의 receptive field 의 resolution 을 증가시키자!

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

→ Stride 1, without pooling layer
→ Stride 1, with pooling layer
→ Stride 2, with pooling layer

Table 2. Comparison of recognition performance when the size of the receptive field varies, but the computational cost is constant.



10. Experimental Results and Comparions

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.

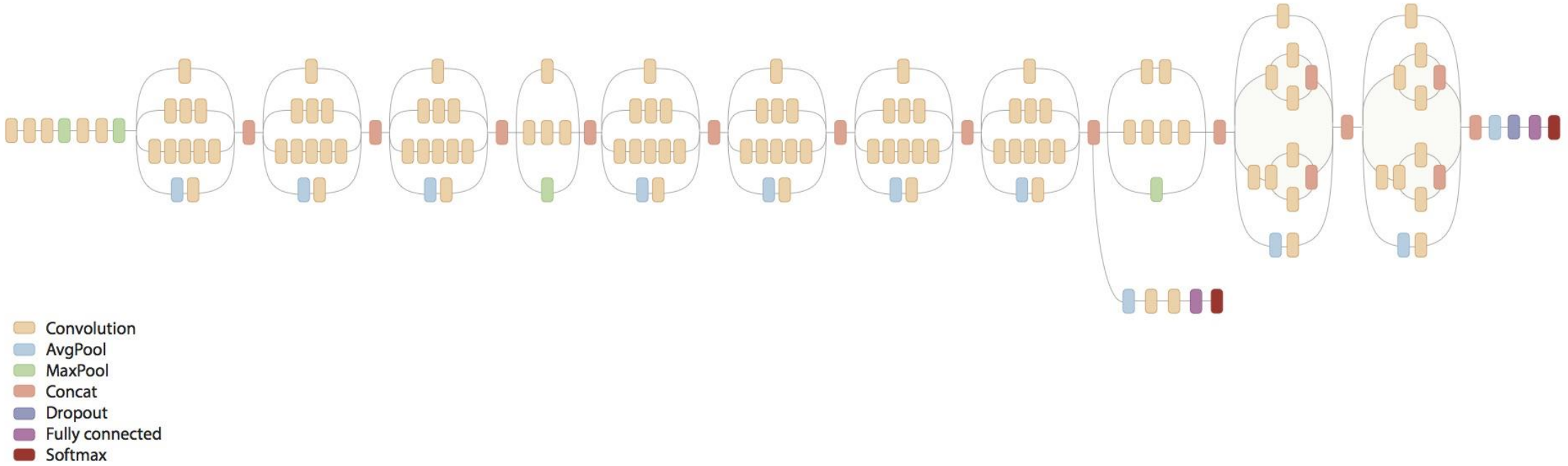
Network	Crops Evaluated	Top-5 Error	Top-1 Error
GoogLeNet [20]	10	-	9.15%
GoogLeNet [20]	144	-	7.89%
VGG [18]	-	24.4%	6.8%
BN-Inception [7]	144	22%	5.82%
PReLU [6]	10	24.27%	7.38%
PReLU [6]	-	21.59%	5.71%
Inception-v3	12	19.47%	4.48%
Inception-v3	144	**18.77%**	**4.2%**

Table 4. Single-model, multi-crop experimental results comparing the cumulative effects on the various contributing factors. We

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.

11. Conclusion



Reference

Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

https://towardsdatascience.com/a-reading-guide-about-deep-learning-with-cnns-3a0e0fc99b78

https://wikidocs.net/137251