



# [논문 리뷰]InceptionV3 [2015]

## Introduction

☀ **InceptionV3은 Scale up**을 사용하는 방법을 활용합니다.

2014년을 시작으로, CNN의 깊이와 넓이를 사용함으로써, Quality를 상당히 개선하였습니다.

VGGNet에 비해, InceptionV2 ~ InceptionV3는 Computational cost 가 낮으며, 높은 성과를 가져왔습니다.

해당 논문은 **few general principles and optimization idea**를 증명하는데 상당히 유용합니다.

**InceptionV3 module**은 **dimensional reduction and parallel structure**를 사용함으로써, 주변 구성 요소의 구조적 변화를 최대한 완화할 수 있습니다.

## General Design Principles

해당 논문은 Model Optimization을 위한, 4가지 원칙을 제시합니다. 아직 이론적인 부분이므로, 검증될 필요가 있습니다.

### Avoid representational bottlenecks

- “Representational”이라는 표현은, Feature Map의 사이즈를 의미하는데, Bottlenecks 구조에서, Pooling을 과도하게 하는 경우, **Feature Map의 Information**을 discard하는 **Problem**이 발생합니다.

### Higher dimensional representations easier to preprocess locally within network

- Increasing the activations per tile[모델 끝 부분]에서 정보를 쪼개는 경우 다양한 데이터로 표현하는 것이 가능합니다. 그렇게 하면, **학습 및 메모리 효율이 상승**합니다.

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

- Model이 Input을 받아드릴 때, Lower dimensional embeddings는 괜찮다는 것입니다.  
이러한 이유는 인접한 Pixel Information을 함축적으로 가지게 되고, feature map이 작아지는 구간에 진입하기 전까지는 적당히 사이즈를 줄여주는게 빠른 학습에 도움될 것입니다. (가설)

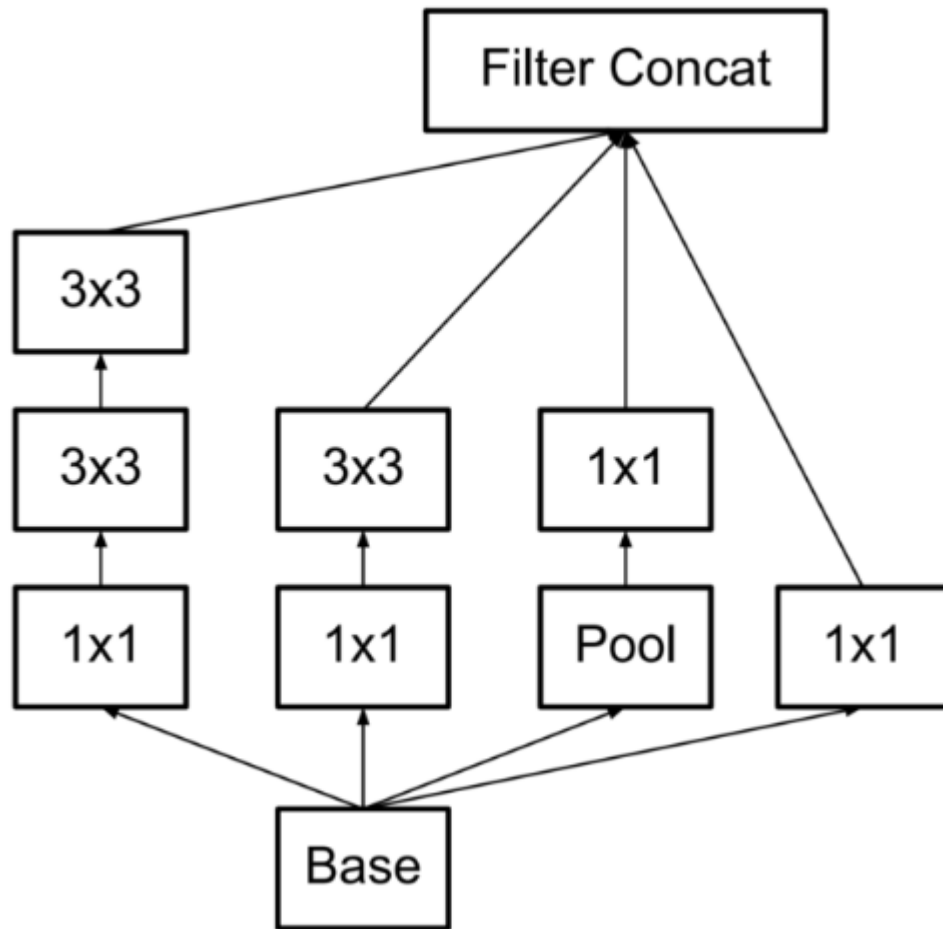
## Balance the width and depth of the Network

- 모델의 Width 와 depth를 적절히 조절해야 좋은 성능을 냅니다.

# Factorizing Conv with Large Filter Size

## Factorization into smaller convolutions

- 1x1 Conv Layer를 사용한 이후, 3x3 Conv Layer를 사용합니다. 이러한 방식은 Aggregation을 하기 전에 Activations를 reduced하고 **similarly expressive local representations**합니다.
- 또한, **5x5 Conv Layer를 3x3 Conv Layer 2개로 대체**함으로서, 큰 Filter를 작은 Filter 대체합니다.



## Spatial Factorization into Asymmetric Convolutions

- Inception은 **sequence of 3x3 Convolutions**를 2개 활용하여, 한 번 더 분해할 수 있습니다.

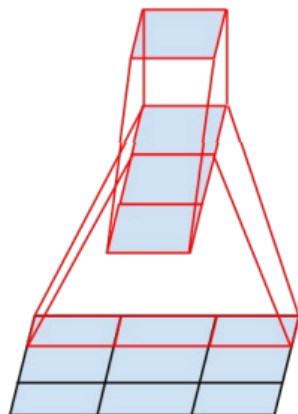
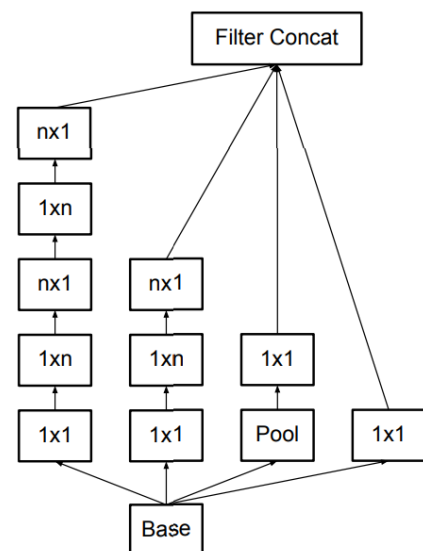


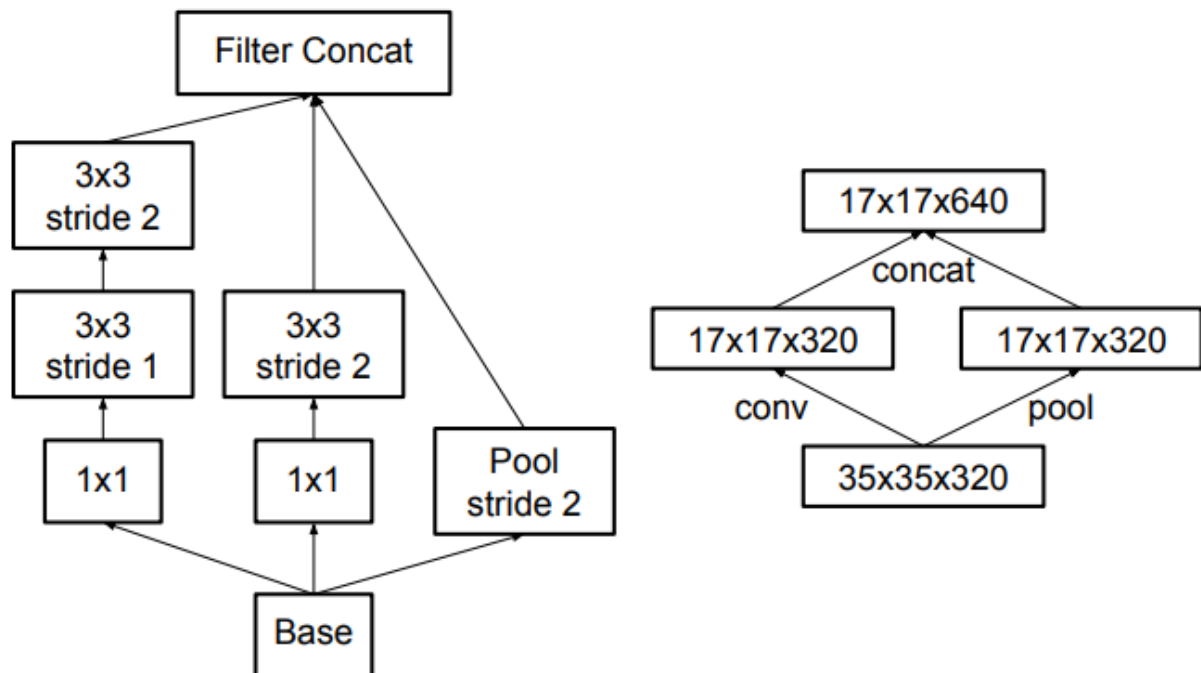
Figure 3. Mini-network replacing the  $3 \times 3$  convolutions. The lower layer of this network consists of a  $3 \times 1$  convolution with 3 output units.



## Efficient Grid Size Reduction

이전까지는 CNN의 Feature map의 크기를 축소시키기 위하여, Pooling기능을 활용하였습니다. 하지만 **Pooling만을 사용하여, 축소하는 경우, Computational이 복잡했습니다.**

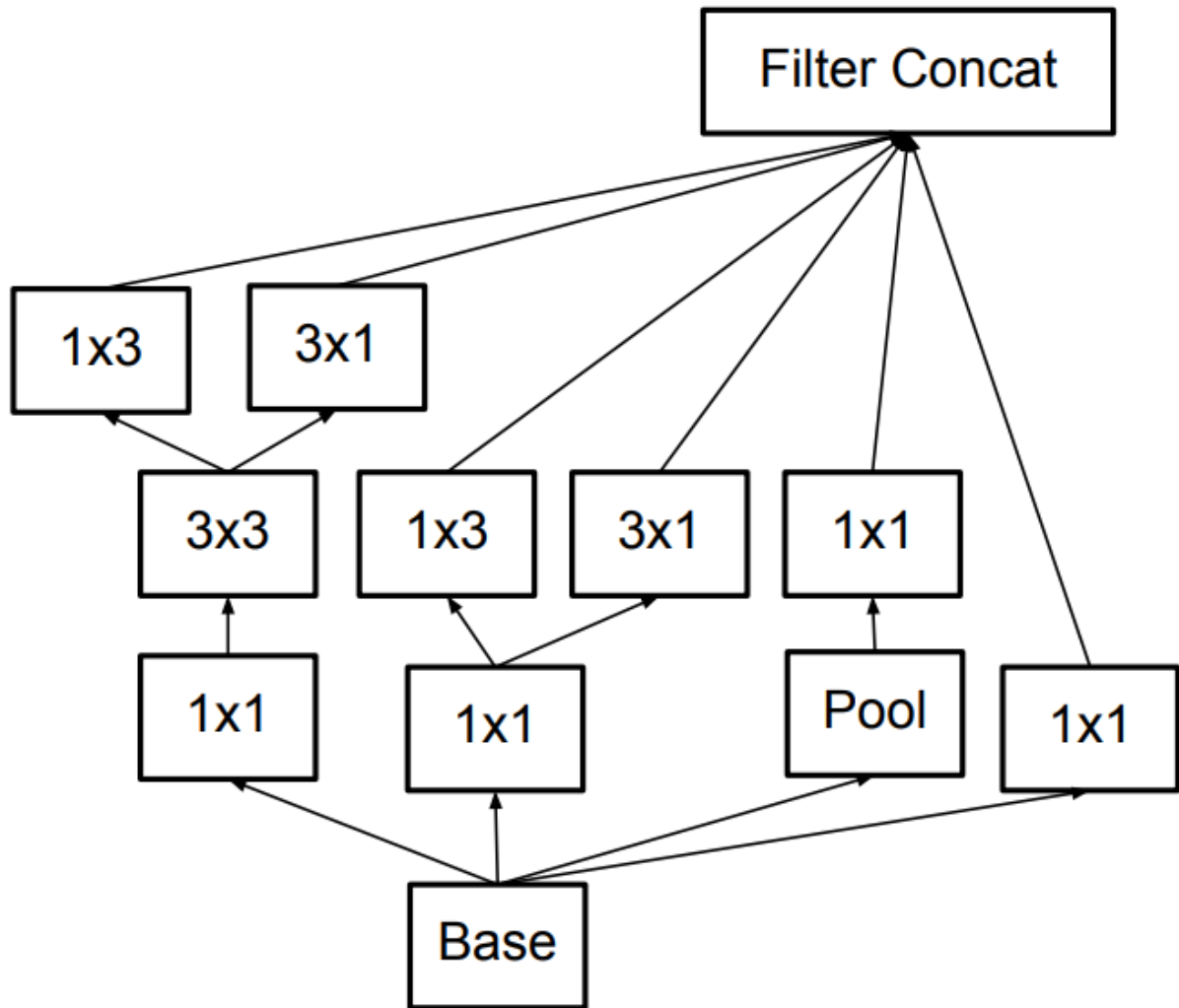
☀ Inception Block에서는 Stride2를 사용하여, Image Size를 축소하면서, Computational을 줄이는 방식을 활용합니다.



## Inception-V2

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

- Feature Map Size가 35x35일 때 부터는, 단락이 바뀔 때, Inception Block을 사용해서, 이미지 크기를 줄여 다음 단계로 넘깁니다.
- Pooling부분에서는 Feature map Size 8\*8을 사용하여, InceptionBlock을 두번 더 사용합니다.



## Model Regularization via Label Smoothing

이전의 정규화 방법인 Batch Normalization을 활용하기 보다는, 다른 정규화 방법인 Label Smoothing을 고안합니다.

쉽게 설명하자면, Label Smoothing은 문자 그래도, 정답인 Label과 아닌 Label에 너무 가혹하게 0과 1의 점수를 부여해서 모델이 과도하게 정답에 확신을 가하면 Overfitting으로 흐를 수가 있다. 그러니까, Overfitting을 피할 수 있도록 정답인 **Label**과 아닌 **Label**이 가지게 될 점수를 좀 더 **Smooth**하게 해주자! 라는 개념입니다. :sunglasses:

Class가 4개라고 했을 때, 정답이 [1, 0, 0, 0]의 점수로 되어 있다면, 좀더 Smooth하게 [0.75, 0.25, 0.25, 0.25] 정도 점수로 해줘서 너무 강하게 확신하지 않게 해주자~ 정도로 이해하면 될 것 같습니다.

수식으로 설명을 하자면, label  $y$ 를 가진  $x$  데이터에 대해서, 예측한 label인  $k$ 가 정답일 경우,  $k = y$ 이면, 1이고  $k \neq y$ 인 다른 모든 Label에 대해 0을 부여하는 방식으로 log-likelihood를 최대화 가져가는,  $q(k) = \delta_{k,y}$ 가 있다고 했을 때.

Label Smoothing은 아래와 같이 Smoothing parameter인 epsilon에 따라, 그리고 labels의 분포를 뜻하는  $u(k)$ 에 대해 아래와 같은 식을 가집니다.

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

논문의 저자는 전체 label 수인  $K$ 에 대한 uniform distribution으로  $u(k) = 1/K$ 를 사용했기 때문에, 위의 식은 아래와 같습니다.

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

그래서 Label Smoothing Regularization, LSR을 Cross-entropy에 적용하면, 아래와 같이 됩니다.  $u(k)$ 는 이미지넷이니깐  $1/1000$ ,  $\epsilon = 0.1$ 을 사용했다고 합니다. 결과적으로 top-1 error와 top-5 error 둘 다 이미지넷 2012 validation set에 대해 0.2%정도씩 성능향상이 있었다고 하네요!

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

## Experiment Results and Comparisons - InceptionV3

Network	Top-1 Error	Top-5 Error	Cost Bn Ops
GoogLeNet [20]	29%	9.2%	1.5
BN-GoogLeNet	26.8%	-	<b>1.5</b>
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 $7 \times 7$	21.6%	5.8	4.8
Inception-v2 BN-auxiliary	<b>21.2%</b>	<b>5.6%</b>	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	<b>18.77%</b>	<b>4.2%</b>

- InceptionV3는 InceptionV2 모델을 기본적으로 사용하면서, 위의 많은 기법들을 엮은 것에 불과합니다.
- 좌측 Table의 Inception-v2 BN-auxiliary는

InceptionV2 + RMSProp + Label Smoothing Factorized  $7 \times 7$  + BN-auxiliary입니다.

#### Rethinking the Inception Architecture for Computer Vision

Convolutional networks are at the core of most state-of-the-art computer vision solutions for a wide variety of tasks. Since 2014 very deep convolutional networks started to become

 <https://arxiv.org/abs/1512.00567>



#### Inception v2, v3 - 2015 | DataCrew

'1등을 하고도 받은 수모.. 내가 돌려주마..' 부들부들거리는 것 같은 Inception v2, v3 리뷰! 심플한 구조로 많은 사랑을 받은 VGG 때문에, 2014 이미지넷 Classification에서 훨씬 더 작은 학습량으로 1위를 했

 <http://datacrew.tech/inception-v2-v3-2015/>

