Inception

Abstract

깊은 convolution network는 주요 작업이 되어졌습니다. 하지만 모델 사이즈를 키우는 것은 성능을 높이는데 일조할 수 있지만 적은 파라미터와 컴퓨팅 효율성은 모바일 기기에서 중요하기 때문에 간과해서는 안됩니다. 그래서 우리는 적절한 컨볼루션 재구성과 정규화를 통해서 네트워크를 키우며 연산량을 줄이는 방식을 제안합니다.

Introduction

2012년 ImageNet대회에서 AlexNet이 성공을 이뤘습니다. 그리고 더 깊고 넓은 네트워크를 사용한 VGGNet과 GoogleNet은 2014년 ILSVRC대회에서 높은 성적을 거두었습니다.

VGGNet의 경우 간단한 모형이지만 많은 자원을 필요로 합니다. 반면 GoogleNet은 성능을 보장함과 동시에 더 작은 자원을 필요로 하고, 이는 AlexNet보다 12배 이상으로 자원 소모를 감소시켰습니다.

GoogleNet 5M, AlexNet 60M, VGGNet 180M, 그리고 Inception은 VGGNet보다 컴퓨팅 자원을 상당히 줄였으며 더 높은 성능을 보여줍니다. 이를 통해 많은 데이터를 학습 시킬 수 있으며 모바일 기기에서도 사용이 가능해졌습니다. 다른 방안으로도 컴퓨팅 자원 소모를 줄일 수 있지만, 이러한 경우 복잡하며 게다가 Inception module을 사용함과 동시에도 사용이 가능합니다.

다만, 우리의 모델이 복잡해서 응용하여 사용하는 것은 어려운 점이 있습니다. 이 논문에서는 우리의 인셉션 모델만을 다루는 것이 아닌 다양한 기법에 대해서 탐구해 보고자 합니다.

General design principle

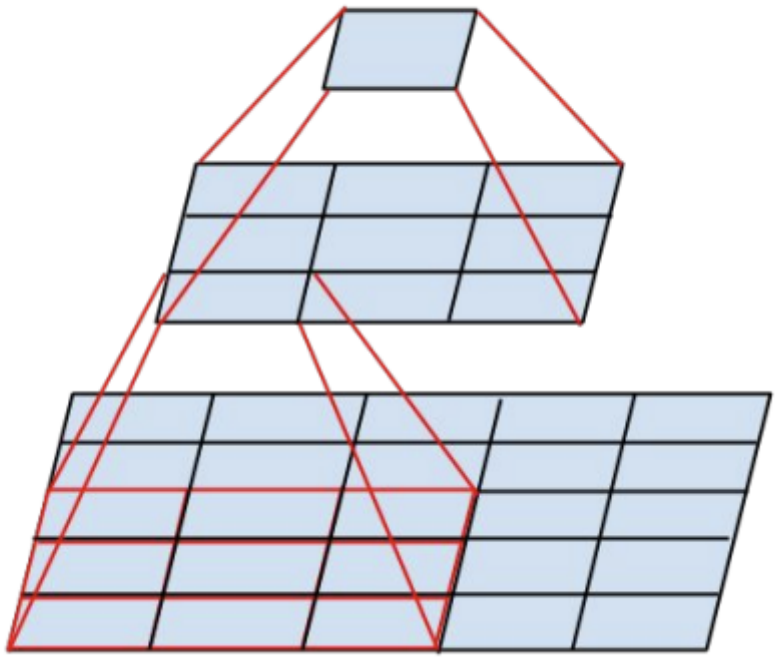
1. Feed-Forward network를 네트워크에 초기에 배치하는 것을 피하라. FFN은 어떠한 정보가 중요한지를 나타내며 특정 정보를 분리하게 되어 몇몇 정보에 대해 무시를 할 수 있다? , 병목 현상을 피하기 위해서 컨볼루션 네트워크의 크기를 서서히 줄여나가야 한다.
2. 네트워크에서 풀링을 통해 더 빠른 학습을 진행할 수 있습니다.
3. 차원을 줄인 다음 3*3 컨볼루션을 진행합니다. 예를 들어, 1*1 컨볼루션을 진행하여 차원을 줄인 다음에 3*3 컨볼루션을 진행합니다. 차원 축소는 더 빠른 학습을 진행한다.
4. 네트워크의 깊이와 너비에 대해 균형을 지킨다.

Factorizing Convolutions with Large Filter Size

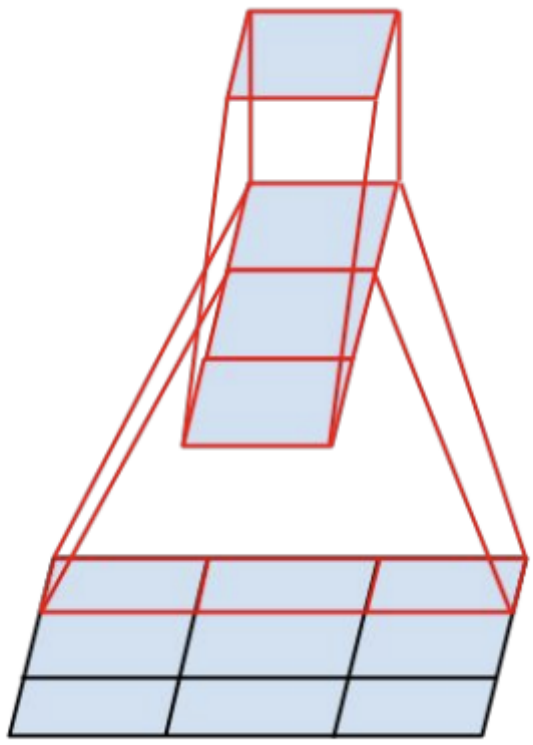
GoogleLeNet에서는 많은 다양한 차원 감소 기법이 있었다. 이는 컨볼루션 재구축을 통해 효율성을 얻었다. 예를 들어 1*1conv진행 후 3*3conv를 진행하는 것이다. 이 논문에서는 더욱 더 다양한 효율성을 얻는 방식을 제안한다.

1. Factorization into smaller convolutions

5*5 , 7*7 필터는 많은 컴퓨팅 자원을 요구한다. 그러나 5*5 컨볼루션의 경우 multi-layer 네트워크를 통해서 더 작은 파라미터 수와 같은 아웃풋의 차원이 나오도록 변형시킬 수 있다. 5*5의 경우 3*3의 두개의 컨볼루션 레이어로 재구축 할 수 있다. 이럴 경우 5*5 = 25 → 3*3 + 3*3 = 18 로 25 → 18로 컴퓨팅 자원이 감소하여 효율성을 추구 할 수 있다. 그리고 layer를 두개로 나누게 되어 activation 함수를 두 번 적용해야 했는데 이 때 Relu함수를 두 번 적용하는 것이 좋은 것으로 나타났다.



2. Spatial Factorization into Asymmetric Convolutions

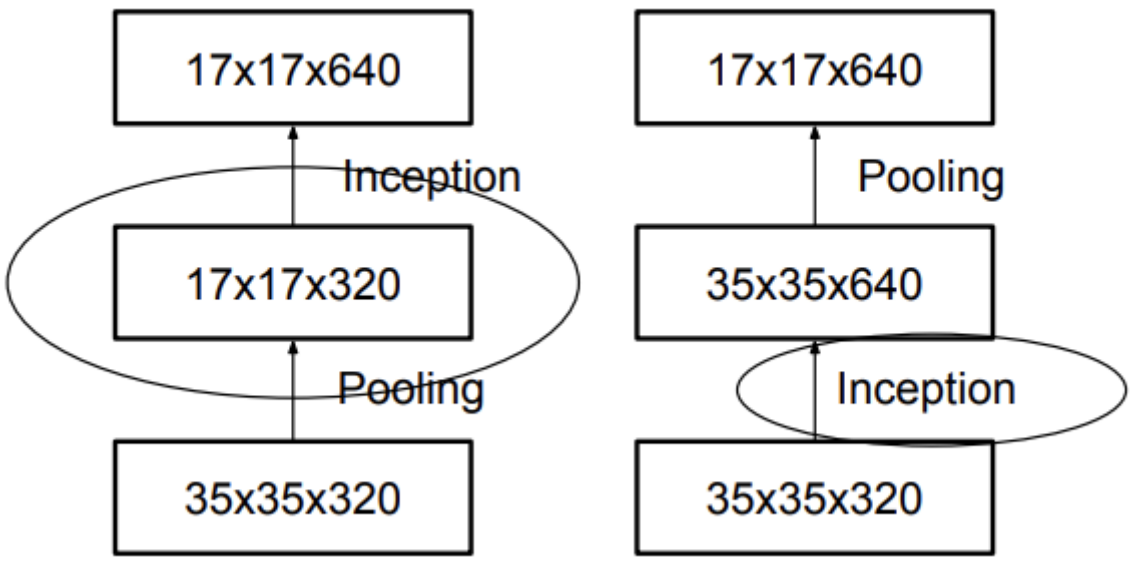


N*N 필터의 경우는 1*N, N*1 필터로 바꾸는게 효율적이다. 예를 들어, 5*5 필터의 경우 5*1, 1*5 필터로 변경이 가능하다. 이때, 25 → 5 + 5 로 연산량을 많이 줄일 수 있게 된다. 그래서 많은 필터를 3의 크기로 하면 좋다. 여기서 왜 2로 바꾸지 않은가를 생각해보면 3*3은 3*1, 1*3으로 전환이 가능하다. 그리고 2*2, 2*2로 전환이 가능한데 이는 연산량이 6대8로 3*1, 1*3이 더 적은 것을 확인할 수 있다. 실용적으로 우리는 이러한 구조는 초기 레이어에서 잘 작동하지 않는 것을 확인하였다. 그러나 중간 크기의 필터 사이즈들에서는 좋은 효과를 보였다. 12~20 사이에서 그리고 7*1, 1*7 컨볼루션에서 좋은 결과를 얻어냈다.

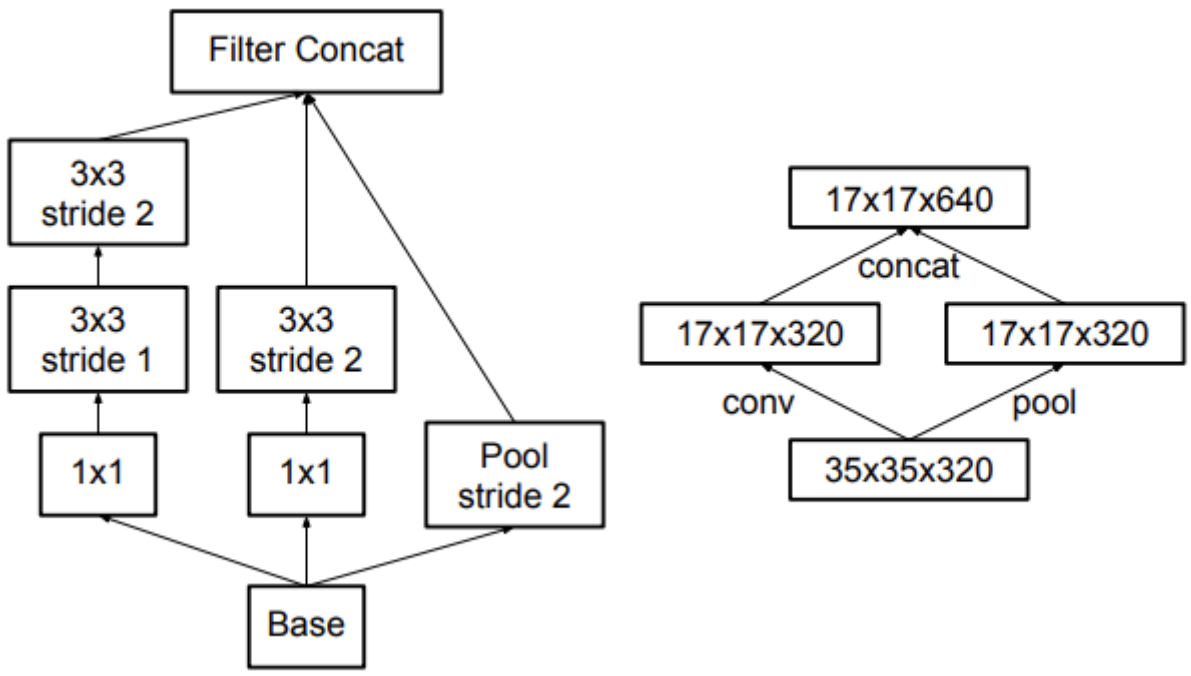
Utility of Auxiliary Classifiers

보조 분류기는 깊은 네트워크에서 성능을 높여주는 것으로 알려져있다. 원래의 목적은 기울기 소실을 방지하기 위하여 유용한 기울기 값을 전달해주기 위해 사용되어졌다. 흥미롭게도 우리는 보조 분류기가 초기 학습에서 그러한 역할을 하지 못하는 것을 알아냈다. 보조 분류기가 있든 없든 높은 성능을 내는 것으로 보여졌다. 보조 분류기는 메인 분류기에 정규화를 진행해주는 것처럼 보여졌다.

Efficient Grid Size Reduction



전통적을 convolution network는 pooling을 사용했다. 병목현상(정보 훼손)을 피하기 위해서 convolution 진행시 필터의 수를 줄이면서 채널을 늘린다음, pooling을 진행한다. 이러할 경우 연산량을 줄일 수 있지만 결국 풀링으로 인하여 정보가 훼손이 된다. 좌측의 경우에는 언급한 병목현상을 야기하고, 우측의 경우에는 연산량이 3배정도 더 크다.



따라서, 우리는 위와 같은 방식을 제시한다. 위 그림은 연산량을 줄이고 병목현상을 피할 수 있는 네트워크 구조를 제안한다. 한 쪽은 convolution을 다른 한쪽은 pooling연산을 진행하여 이후 합친 결과물을 출력한다.

→ 이러한 구조는 내 생각에는 결국 컨볼루션과 pooling을 순차적으로 진행한 것이 아닌 한번에 진행함으로써 모든 장점을 합친다음에 절반의 효과를, 모든 단점을 합친다음에 절반의 효과를 나타내는 것이 아닌가란 생각이 든다.

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

이때까지 논문에서 언급하였던 기법들을 다 합쳐서 새로운 아키텍쳐를 제시한다.

우리의 이러한 구조는 GoogleNet이나 VGG보다 훨씬 더 효율적임을 보이고 있다.

Model Regularization via Label Smoothing

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

기존의 분류에서 예측을 할 경우 정답은 1로 정답이 아닌 것은 0으로의 강한 확신을 유도하는 학습이 진행되어졌다. 하지만, 이러한 강한 학습은 과적합이 유도되어지게 된다. 따라서 위의 식을 통해 1로 학습하는 것은 보다 낮은 값을 갖도록 0으로 학습 되는 것은 약간 더 높게 학습할 수 있도록 유도되어진다.

Training Methodology

우리는 barch size를 32로 하고 100epoch으로 학습을 진행하였다. 그리고 stochastic gradient를 사용하였고, 0.9의 decay값을 가지는 momentum을 사용한다. 그리고 RMSProp을 0.9 decay, 앱실론 = 1로 설정하였을 때 가장 좋은 성능을 얻었다. 우리는 learning rate를 0.04로 사용하고 2에폭을 학습할 때마다 0.94를 learning rate에 곱하여 사용했다.

Performance on Lower Resolution Input

전형적인 비전 네트워크 문제는 object detection이다. 이때의 문제는 object가 작거나 상대적으로 저해상도인 경우이다. 이때 의문점은 낮은 해상도의 input이 들어왔을 때 어떻게 다루어야하는가이다. 보통 높은 해상도가 들어오게 될 경우 높은 성능을 보인다. 그러나 해상도가 증가했을 때와 모델이 커짐에 따름에 대한 성능 향상인지 구분하는 것은 중요하다.

만약 모델의 변경 없이 해상도만 변경할 경우 모델은 더욱 어려운 문제를 해결하는 꼴이 된다. 물론 이러한 경우 적은 계산으로 성능은 당연히 안좋을 것이다. 다만, 이러한 현상을 정확히 평가하기 위해서 모델이 주는 애매한 힌트를 살펴볼 필요가 있다. 가장 주된 의문은 계산량이 같을 경우 높은 해상도 input은 얼마나 정확도에 영향을 끼칠까이다.

1. 299×299 receptive field with stride 2 and maximum pooling after the first layer.
2. 151×151 receptive field with stride 1 and maximum pooling after the first layer.
3. 79×79 receptive field with stride 1 and **without** pooling after the first layer.

이를 위해 우리는 높은 해상도의 경우 모델 자체의 계산량을 줄여 같은 연산량을 가질 수 있도록 설정한다. 세번째의 경우 연산량이 1% 미만 정도 더 작지만 이는 무시한다.

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

위에 네트워크 실험에 대한 결과표이다. 해상도가 낮은 네트워크가 학습하는데 더 많은 시간이 걸리긴 하지만 비슷한 성능을 나타내며, 가장 높은 해상도에서 가장 높은 성능을 나타낸다. 이는 작은 물체 감지에 대해서 높은 해상도가 필요함을 암시한다.

Experimental Results and Comparisions

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%***

ILSVRC 2012 대회에서 우리 모델이 가장 높은 성능을 달성한 것을 확인할 수 있다.

Conclusion

우리는 convolution networks 에서 더 깊은 모델을 만들 수 있는 몇가지 방식을 제시하였다. 이러한 방식은 더 높은 성능과 낮은 연산량을 가진다. 또한, 보조 분류기, label-smoothing 방식은 상대적으로 적은 데이터셋에서 좋은 성능을 야기한다.