**Inception  
Going Deeper with Convolutions**

한현수

**Abstract**

이 논문의 network 아키텍처는 네트워크 안에서 컴퓨팅 리소스의 활용을 향상시켰다.  
그래서 computing resource는 유지하면서 network의 width, depth를 증가할 수 있도록 설계되었다.

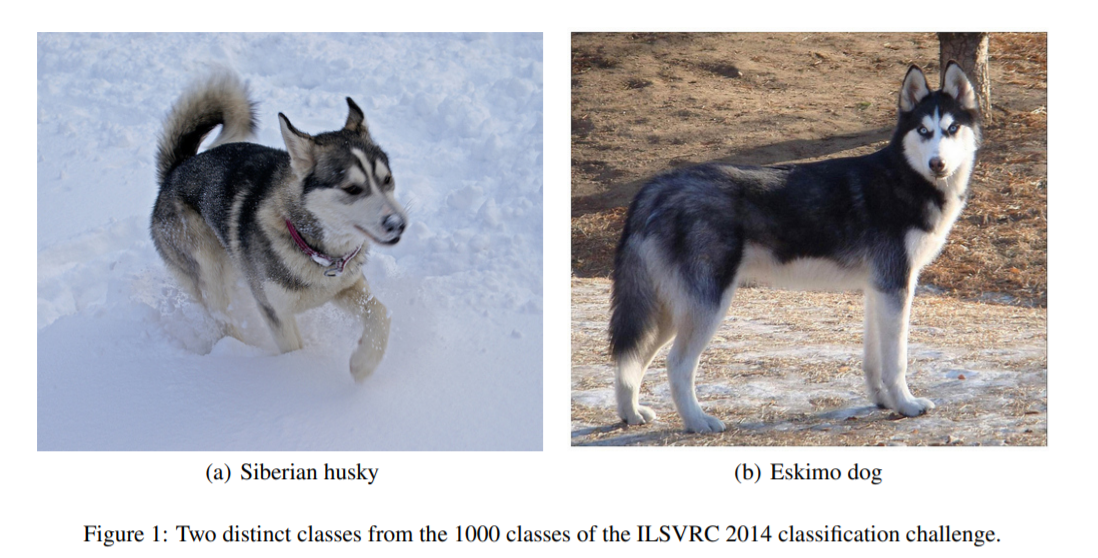
**1. Introduction**

지난 deep learning의 발전은 CNN, 하드웨어, 데이터, 더 커진 모델만이 아니라 모바일, 임베디드 컴퓨팅의 관심으로 인한 알고리즘의 효율성, 새로운 아이디어, 향상된 network 아키텍처의 결과이다. 그래서, 대부분의 실험은 1.5billion 연산이 넘지 않도록 설계하였다. 그 이유는 실제로 사용하기 위함이라고 한다.   
본 논문에서의 “deep”은 “Inception module”의 형태로 새로운 조직을 도입한다는 것과, 네트워크의 깊이를 늘린다는 의미로 사용된다.

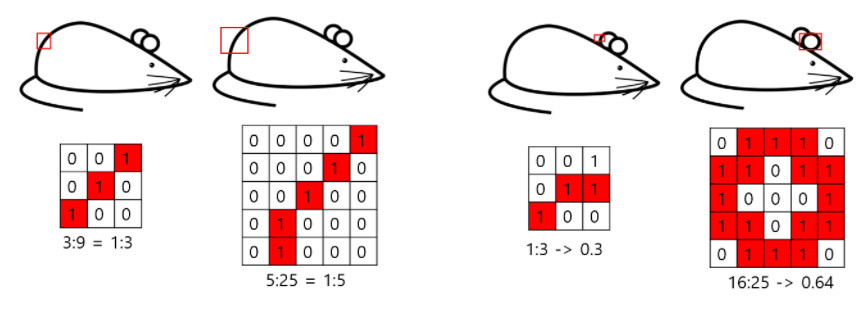
**2. Related work**

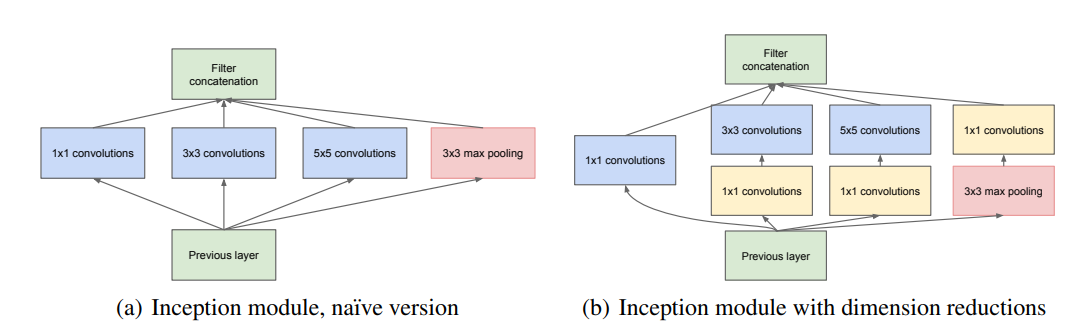
CNN은 일반적으로 convolutional layer(선택적으로 normalization, max pooling이 따름)에 fully-connected layers가 따라오는 구조이다.  
최근 트렌드는 dropout을 사용하여 overfitting을 막으면서, layer와 layer size를 늘리는 것이라고 한다.  
Network-in-Network는 neural network의 표현력을 높이기 위해 사용되는데, 이것이 convolutional layer에 적용될 때, 1X1 convolutional layer를 사용하고, ReLU를 뒤에 붙인다고 한다. 이는 CNN pipelines의 통합을 쉽게 가능하게 해준다고 한다.  
1X1 convolutions는 computational bottlenecks를 제거하기 위한 차원축소 모듈로 사용되고, network의 크기를 제한하는 목적을 가진다. 이는 성능저하 없이 network의 depth, width를 증가시킬 수 있다고 한다.

**3. Motivation and High Level Considerations**

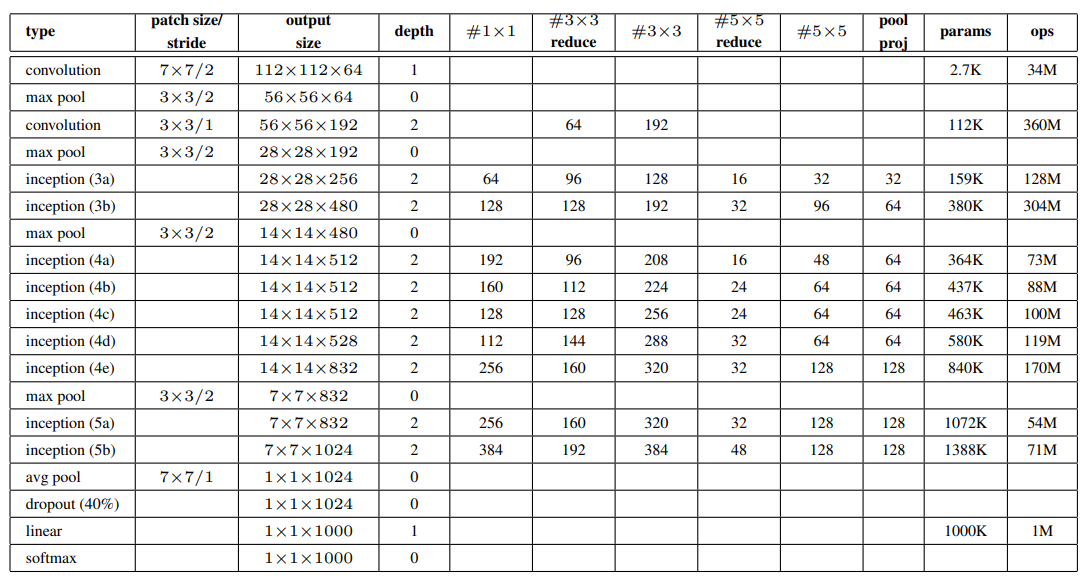
Network의 크기를 점점 증가시키는 것은 depth(layer 개수), width(layer당 unit 개수)를 키우는 것이다. 이는 모델의 성능을 향상시키는 가장 쉽고 간단한 방법이다. 하지만 이에는 단점이 있다.  
1. Network의 크기를 키우는 것은 parameter 개수를 증가시키는 것을 의미하는데, 이는 network가 좀더 쉽게 overfitting된다는 것이다. 아래 그림처럼 ImageNet과 같이 세밀한 범주를 분류해야 하는 경우 심각한 병목현상이 될 수 있다는 것이다.  
2. Network의 크기가 증가함에 따라 computational resources는 급증하게 된다는 것이다. 예를 들어 filter를 2배 키우는 것은 computation을 4배 키우게 된다.  
그래서 이 2가지 단점을 해결하는 가장 근본적인 방법은 fully-connected architectures를 sparsely connected architectures로 변환하는 것이다. 하지만 현재 computing infrastructures는 sparse matrix를 계산하기에 비효율 적이므로 이를 효율적으로 계산하기 위해 sparse matrix를 클러스터링하여 밀도 높은, dense한 submatrix를 만드는 것을 제안하였다.

**4. Architectural Details**

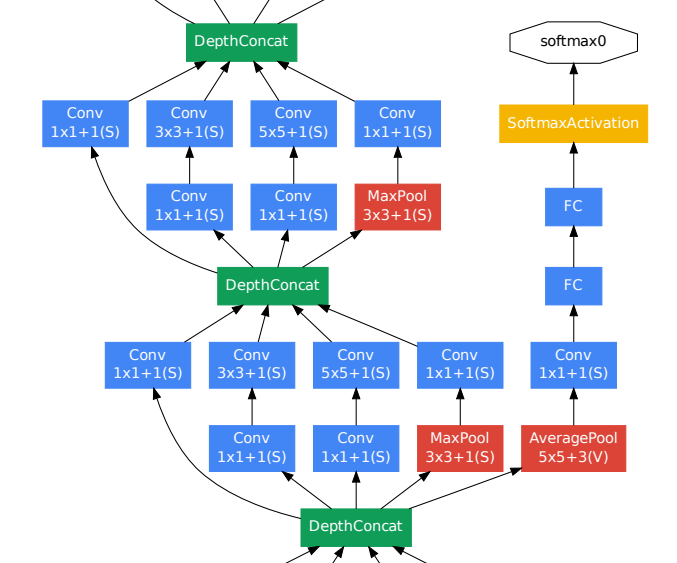
Inception 구조의 아이디어는 sparse matrix를 묶어 dense한 submatrix를 만드는 것이다. 이때 낮은 layer에서는 특정 지역에서 correlated unit들이 집중되어 있다. 그래서 이 부분은 1x1 convolution으로 처리할 수 있다.

하지만 위의 그림처럼 더 넓은 영역의 convolutional filter가 필요한 상황이 나타날 수도 있다. 그래서 1x1, 3x3, 5x5 convolution을 병렬적으로 수행한다.  
높은 추상적인 feature는 높은 layer에서 발견되는데, 이런 추상적인 feature는 공간 집중도가 감소한다는 것을 의미한다. 그래서 layer가 높을수록 3x3, 5x5 convolutional filter의 수가 늘어나야 한다고 한다. 그런데 이런 3x3, 5x5 convolutional filter를 사용하면 연산량이 많아지는 문제가 발생한다.  
그래서 이 문제를 해결하기 위해 위의 그림(Inception module)과 같이 1x1 convolutional filter를 이용하여 차원을 축소시킨다. 이를 통하여 3x3, 5x5 convolution을 이용한다고 하더라도 연산량을 낮출 수 있다. 하지만 이러한 Inception module은 효율적인 메모리 사용을 위해 낮은 layer에서는 기본적인 CNN을 사용하고 높은 layer에서 Inception module을 사용한다고 한다.  
이러한 Inception module을 사용함으로써 얻는 두가지 효과는 아래와 같다.  
1. 차원 축소를 통해 연산량 증가 문제없이 각 stage에서 unit의 수를 증가시킬 수 있다.  
2. 1x1, 3x3, 5x5 convolution 연산을 통해 다양한 scale로 특징을 추출하기 때문에 동시에 서로 다른 layer에서 특징을 추출할 수 있다.

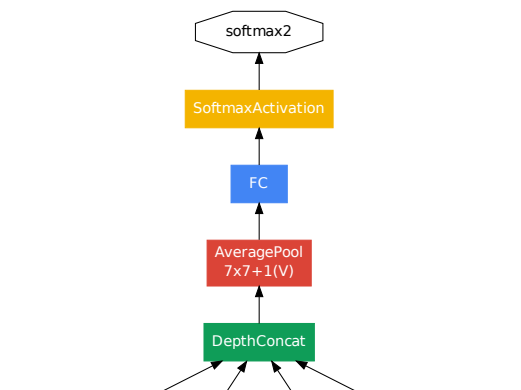
**5. GoogLe Net**

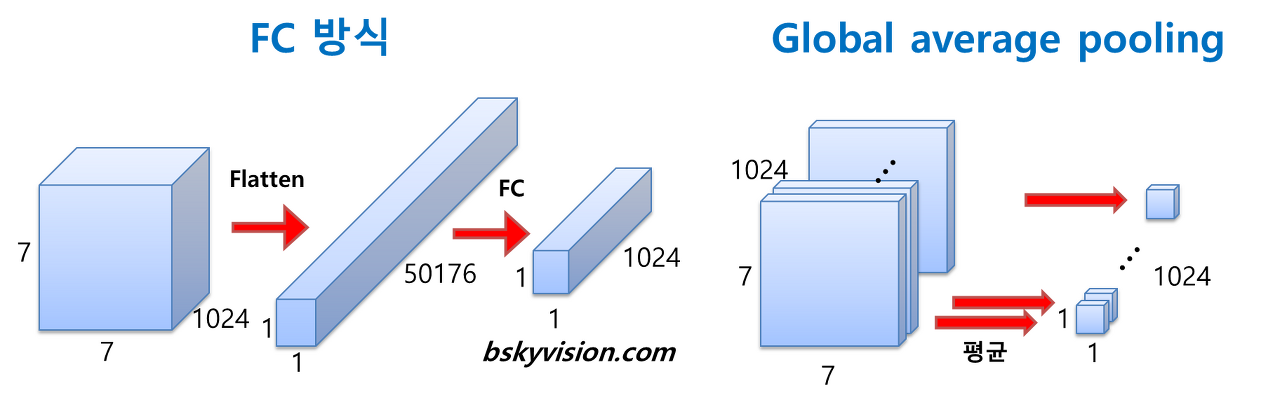
이러한 Inception module이 적용된 GoogLe Net은 LeNet으로부터 유래하였다고 하며 GoogLe Net의 구조는 아래와 같다.

위와 같은 구조에서 Inception module 내부, 모든 convolution layer엔 ReLU가 적용되어 있다. 또한 Receptive field의 크기는 224x224, RGB color channel을 가지며 mean subtraction을 적용한다. 위의 표에서 #3x3 reduce, #5x5 reduce, pool proj는 각 3x3, 5x5 convolutional layer, max pooling layer 앞, 뒤에 사용되는 1x1 convolutional layer의 채널 수를 의미한다.



모델의 깊이가 깊어지는 경우 gradient가 0으로 수렴하는 gradient vanishing 문제가 발생할 수 있다. 이를 해결하기 위해 위에 그림처럼 중간 layer에 auxiliary classifier를 추가하여 중간에 결과를 출력해 추가적인 back propagation을 일으켜 gradient가 깊은 모델에서도 전달될 수 있게 한다. 이러한 auxiliary classifier는 전체 모델에 많은 영향을 주는 것을 방지하기 위해 loss에 0.3을 곱하고, test에서는 이를 제거해서 사용하였다고 한다.



위의 그림은 모델의 마지막 부분으로 최종 classifier 이전에 average pooling layer를 사용하는데, 이는 Global Average Pooling이 적용된 것으로 이전의 feature map을 평균 내어 1d vector로 만든다.

이와 같이 바로 flatten 하는 경우보다 GAP(Global Average Pooling)를 사용하여 1d vector로 만들면 가중치 수가 상당히 많이 줄어드는 것을 볼 수 있다. 이를 사용하면 classifer의 가중치 수가 줄기 때문에 fine tuning을 하기 쉬워진다.

**6. Training Methodology**

Model training을 할 때 Google 팀에서는 Stochastic gradient descent, 0.9 momentum을 사용하였고, learning rate는 8epoch마다 4%씩 감소시킨다.  
또, 이미지의 영역을 3:4, 4:3비율료 고정시켜 원래 사이즈의 8% ~100%가 되도록 다양한 patch와 photometric distortions(광도 왜곡)을 사용하여 data를 늘렸다고 한다.

**7. Conclusions**

Inception module의 구조는 sparse한 구조를 dense한 구조로 근사하여 성능을 개선하였다. 이는 기존의 cnn모델과 같은 depth와 width에서 개선된 성능, 연산량을 보여주었다.

참고자료

Going deeper with convolutions (https://arxiv.org/pdf/1409.4842.pdf)

https://phil-baek.tistory.com/entry/3-GoogLeNet-Going-deeper-with-convolutions-%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0