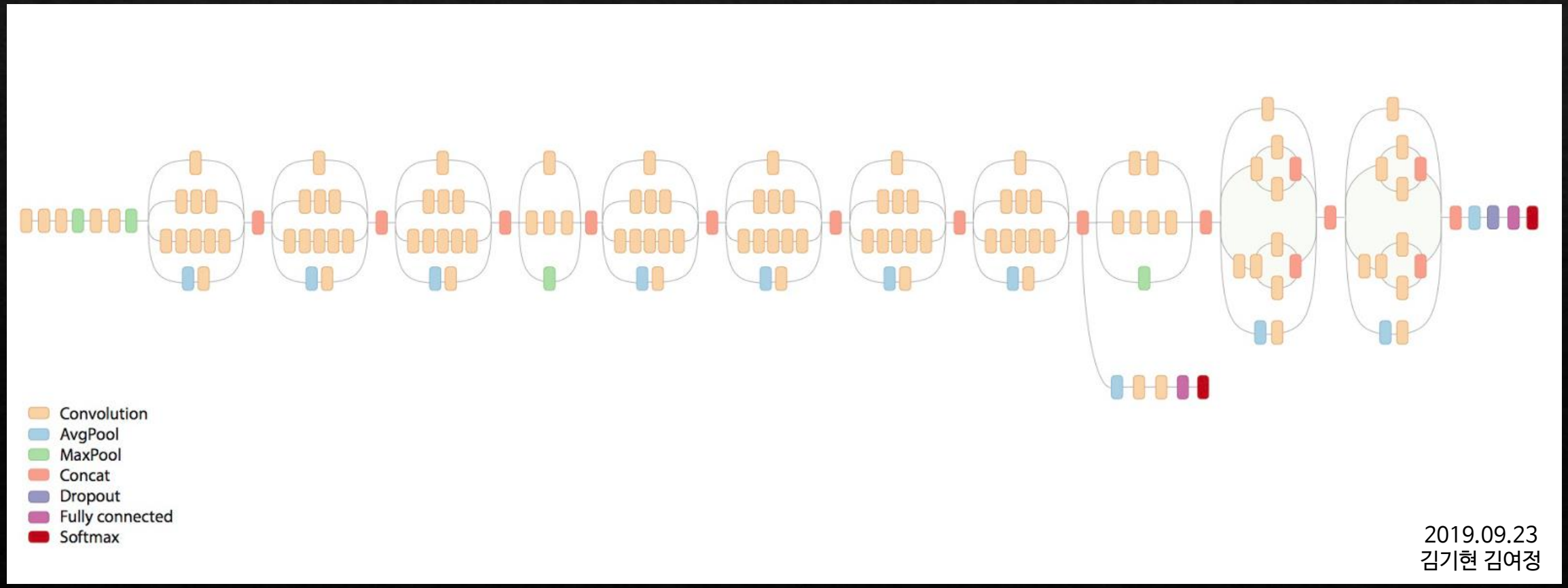
Inception-v4, Inception-ResNet

and the Impact of Residual Connections on Learning

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alexander A. Alemi
Google Inc. 1600 Amphitheatre Parkway Mountain View, CA



2019.09.23
김기현 김여정

목차

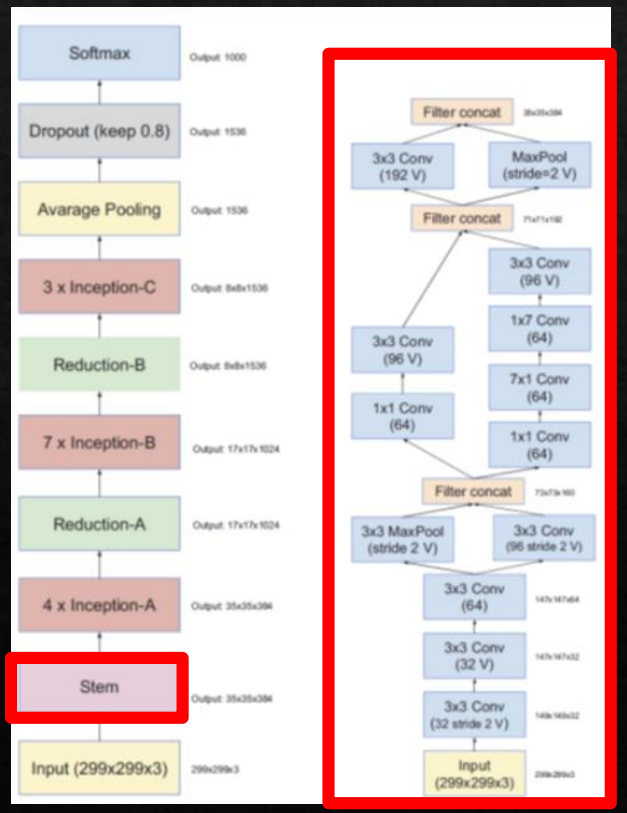
1. 인셉션 모델 소개
    논문 리뷰, 발전, 특징, 아키텍쳐

2. 코드
    전처리, 모델 추가, 실패요소들

3. 결과
    accuracy/loss 그래프, 하이퍼파라미터, 클래스별 오류율

Inception

왜 이름이 Inception일까? NIN!



Inception 모델 소개 - 발전

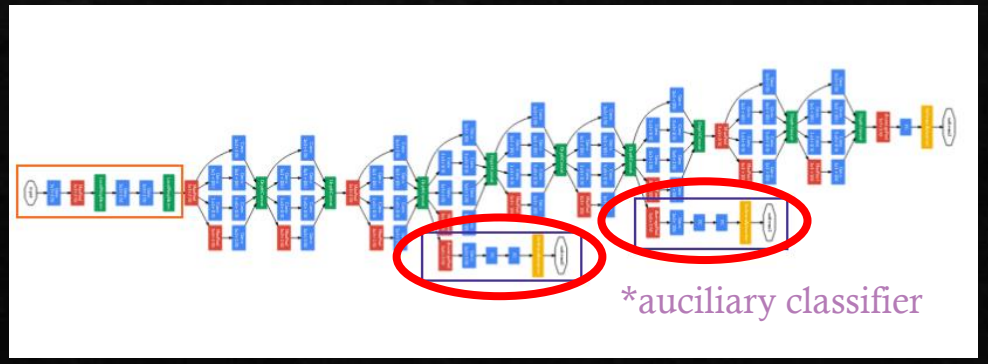
Inception v1 == GoogLeNet

Problem

깊은 네트워크의 과적합 문제, Gradient Vanishing문제, 비싼 계산 비용.

Solution

- 깊은 → 더 넓은 네트워크 만들기(같은 layer의 다른 크기의 filter 적용 -〉 여러 scale의 feature)
- Auxiliary classifiers로 gradient vanishing 문제를 해결.
- 1x1 conv를 이용해 차원을 줄여 연산량을 줄임.



total_loss =
real_loss + 0.3 * aux_loss_1 + 0.3 * aux_loss_2

Inception 모델 소개 - 발전

Inception-v2

Problem

계산 복잡성 문제

Solution

smart factorization method:

- 5x5 컨볼루션을 두 개의 3x3 컨볼루션 연산으로 분해 하여 계산 속도 향상.

  *5x5 컨볼루션은 3x3 컨볼루션보다 2.78 배 더 비쌈.

- nxn의 컨벌루션을 1xn 및 nx1 컨벌루션 조합으로 인수분해.

  예를 들어, 3x3 컨볼루션은 1x3 컨볼루션을 수행 한 다음 출력에서 3x1 컨볼루션을 수행하는 것과 같음.

  *단일 3x3 컨볼 루션보다 33 % 더 저렴한 계산비용.
  *n의 크기가 클 수록 절감효과가 상승.

Inception 모델 소개 - 발전

Inception-module

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

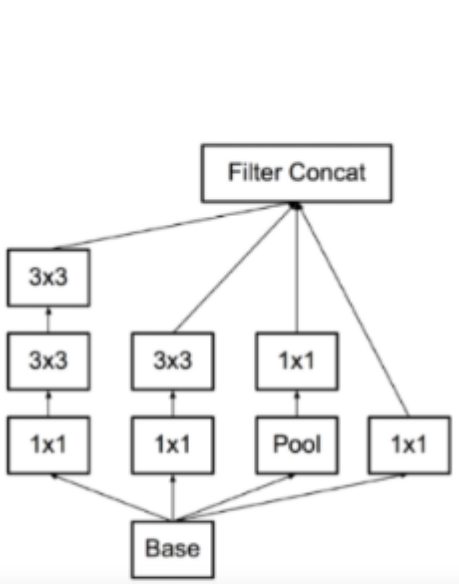


Fig. 5. 3 x Inception-A

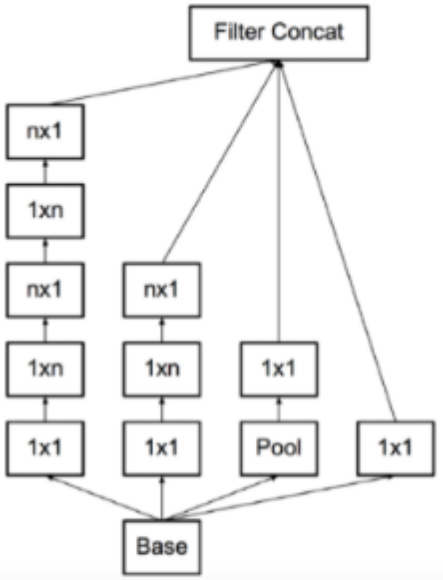


Fig. 6. 5 x Inception-B

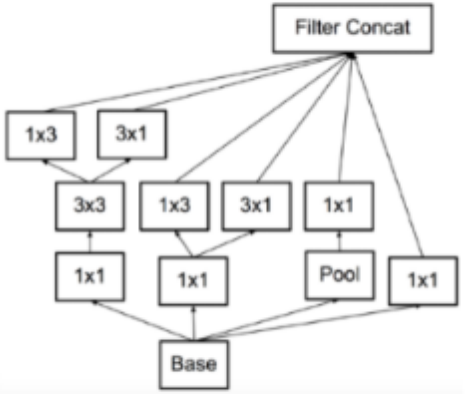


Fig. 7. 2 x Inception-C

Inception 모델 소개 - 발전

Inception-v3

Problem

정확도가 끝까지 다다를 때 쯤 auxiliary classifiers가 제 역할을 못함.

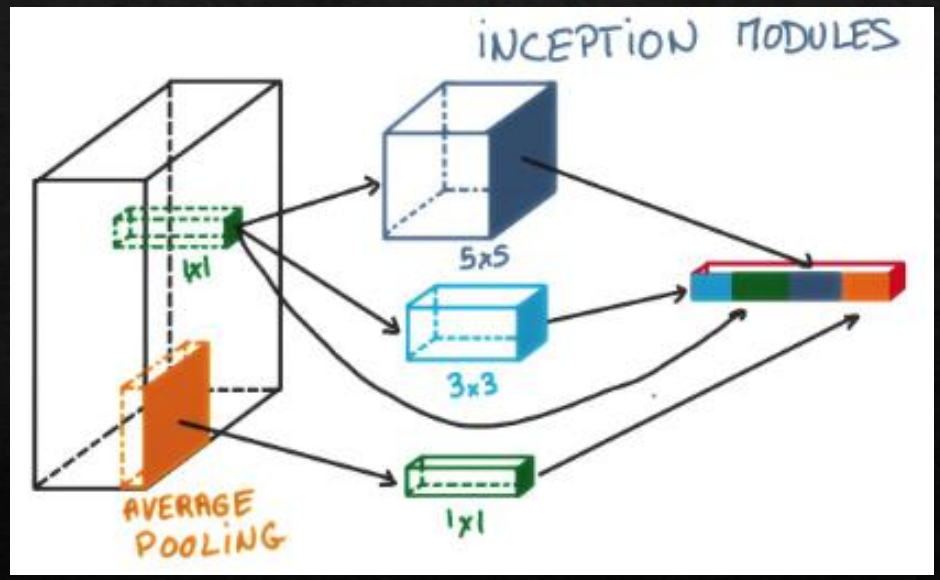
Solution

- RMSProp Optimizer 변경.
- 7x7 컨벌루션 인수 분해(3x3 연산을 세 번 하는 것으로 대체).
- 보조 분류기(auxiliary classifiers)의 BatchNorm.
- 라벨 스무딩 (Label Smoothing) : loss function에 추가 된 정규화 컴포넌트, 네트워크가 이 클래스가 맞다고 확신하는 것을 줄여줌(과적합 방지).

Inception 모델 소개 - 논문 간단 리뷰

Residual connections (He et al. 2015) 과 최신 Inception아키텍쳐(Szegedy et al. 2015b)를 결합

이 논문에서는 3가지 모델을 소개함.



Inception-v4

잔류 연결(residual connection)을 사용하지 않고

필터 연결을 사용하는 심층 회선 네트워크

InceptionResNet-v1, Inception-ResNet-v2

필터 연결 대신 residual connection을 사용하는 Inception 스타일 네트워크.

Inception-v3에 resnet을 접목한 것이 InceptionResNet-v1.

Inception-v3에 resnet을 접목한 것이 InceptionResNet-v2.

Inception 모델 소개 - 논문 간단 리뷰

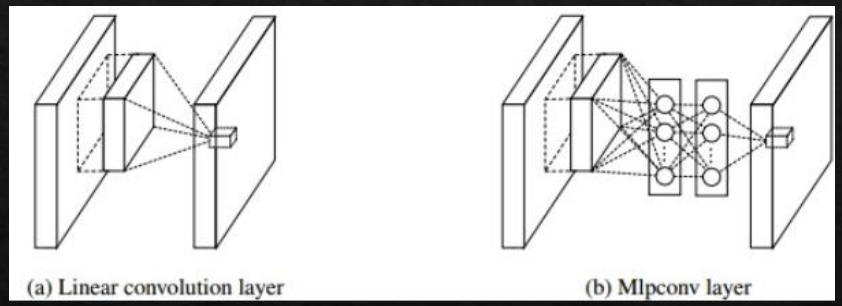
Residual connections은 깊은 CNN에 필수적이라고 여겨졌으나 해당 연구 결과 없어도 트레이닝이 가능하다고 판단. 그러나 트레이닝 속도를 크게 향상시키기 때문에 사용됨.

이미지넷 데이터로 검증했을 때, Inception-ResNet-v2가 제일 좋은 성능을 보임.

Inception 모델 소개 - 논문 간단 리뷰 전에 알고 갈 것.

MLPconv layer (Multi layer perceptron convolution layer)

한 층에 컨볼루션을 한 번 한 것 보다,
한 층에 컨볼루션 연산에 연결된 mlp layer를 넣어
비선형적 관계를 더 잘 표현함.



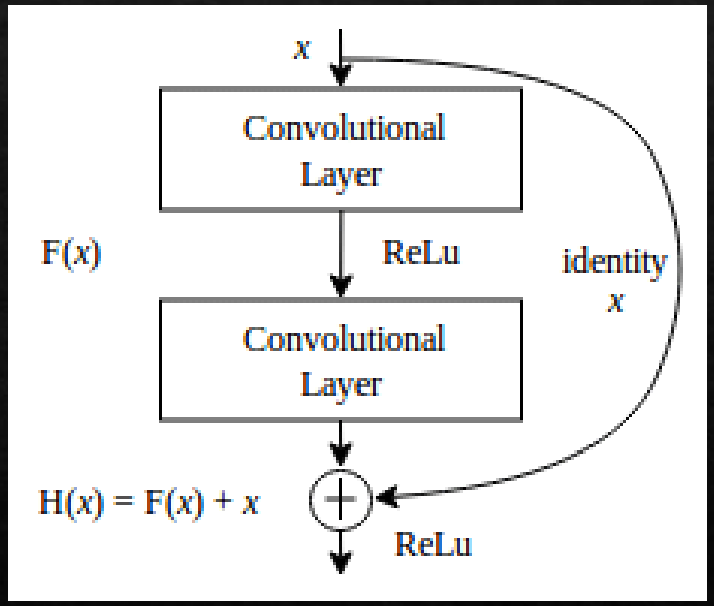
1x1 convolution

1x1 Conv는 채널(차원) 단위에서 Pooling하기 때문에 1x1 Conv 필터 개수를 입력채널보다 작게 하면 dimension reduction, 차원 축소가 가능.

Inception 모델 소개 - 논문 간단 리뷰 전에 알고 갈 것.

residual connections, skip conection

ResNet의 기본 개념. 기존에는 25개 이상의 레이어를 사용하면 gradient vanishing 문제로 정확도가 떨어지는데 이를 해결하기 위해 레이어1의 출력을 레이어2의 입력으로도 사용하면서 레이어 2의 출력과 연결함.



기존 y = f (x) 이지만 , residual network는 H(x) = f (x) + x

덧셈 연산의 추가 만으로 스킵 연결 구현 가능.

backpropgation시 미분값이 사라지는 현상을 줄이고

곱 형태보다 연산이 단순해짐.

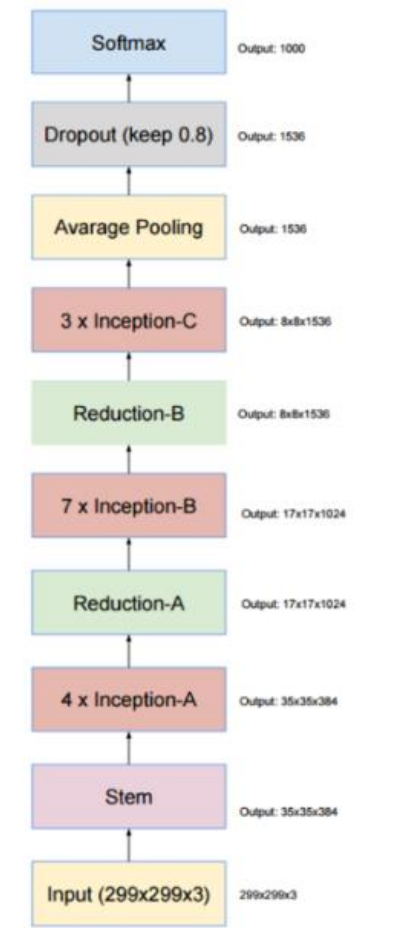
F(x)=H(x)-x로 나타낼 수 있고 F(x)를 0으로 만드는 방향

→H(x)를 입력이랑 같게 만드는 방향으로 학습함.

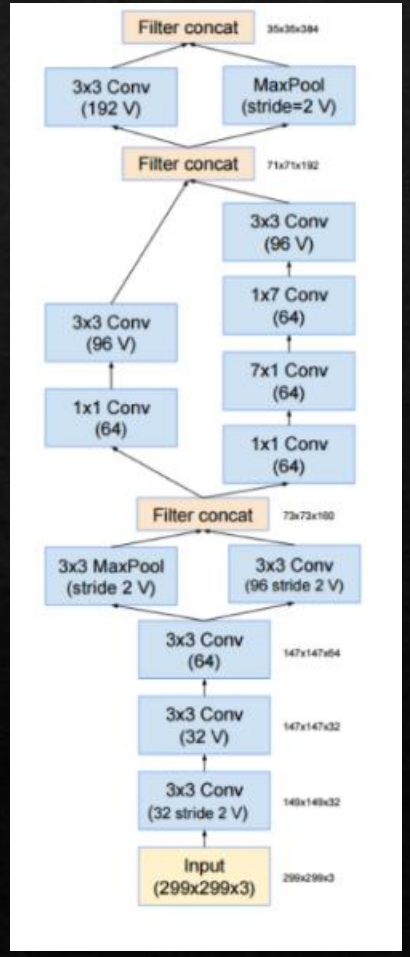
(다르게 말해보면 Identity가 short connection을 통해 출력값으로 넘어오기 때문에 Residual만 0이 되도록 하는 weight를 오류역전파에 의해 구하기가 쉬워짐.)

Inception-V4

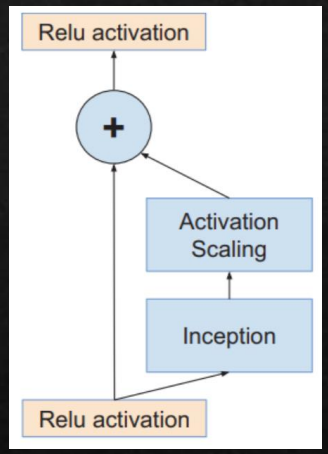
전체 구조



stem 구조

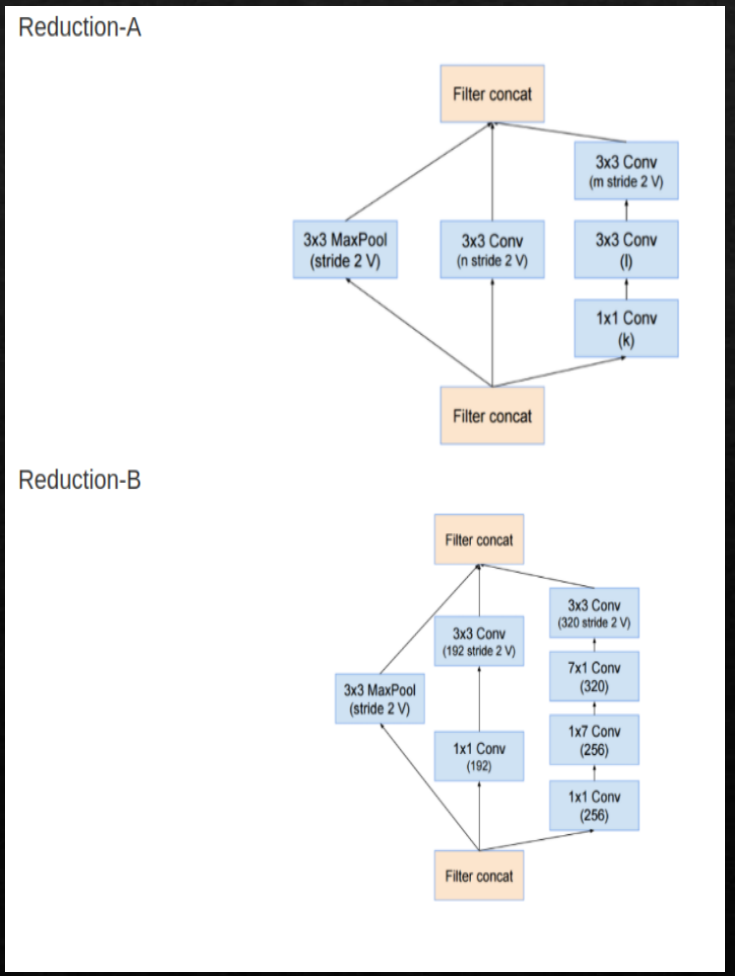


Residuals 스케일링



inception module :차원을 줄이면서 발생하는 정보 손실을 막기위한 방법으로 같은 입력값에 대하여 다양한 연산. 입출력 크기가 똑같음. 입출력 크기가 다른것은 **Reduction module**이라고 부름.

Reduction 모듈



Inception-resnet v1, v2

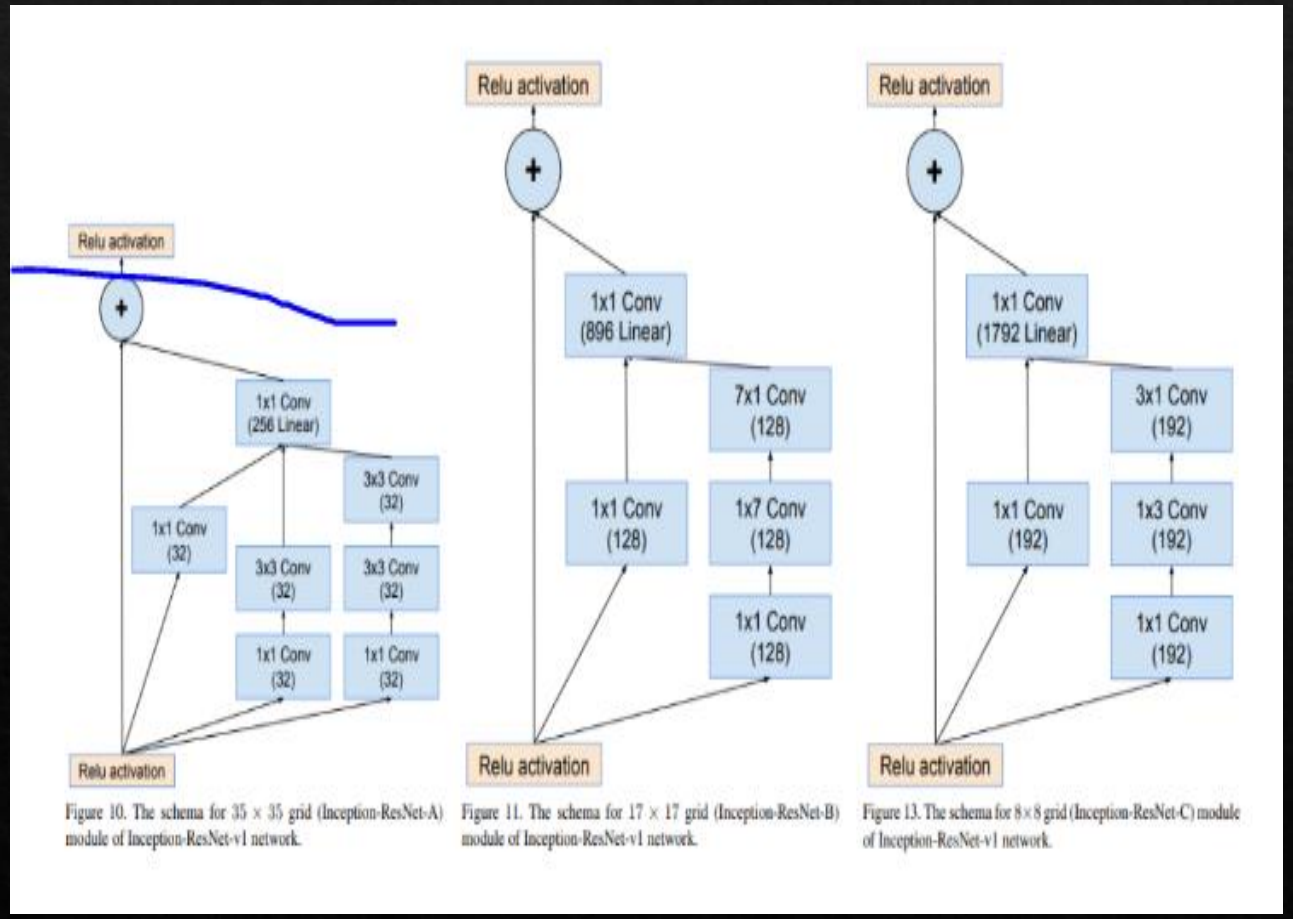
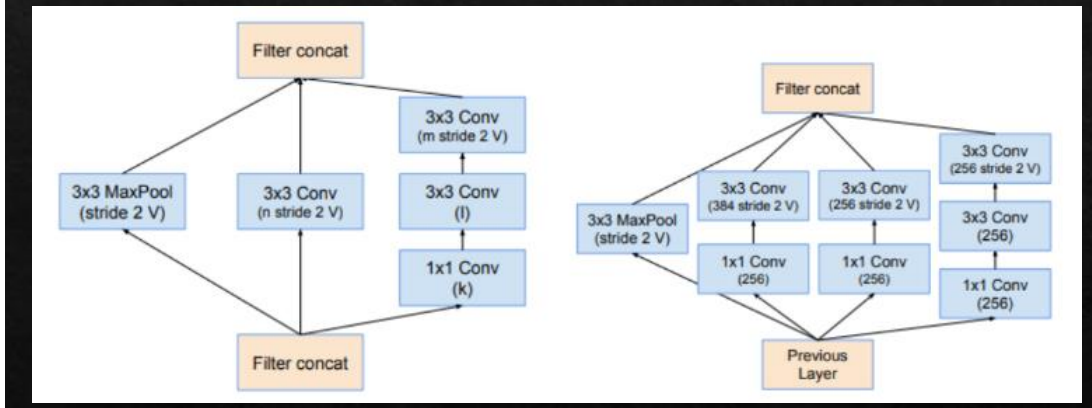


Figure 10. The schema for 35 × 35 grid (Inception-ResNet-A) module of Inception-ResNet-v1 network.

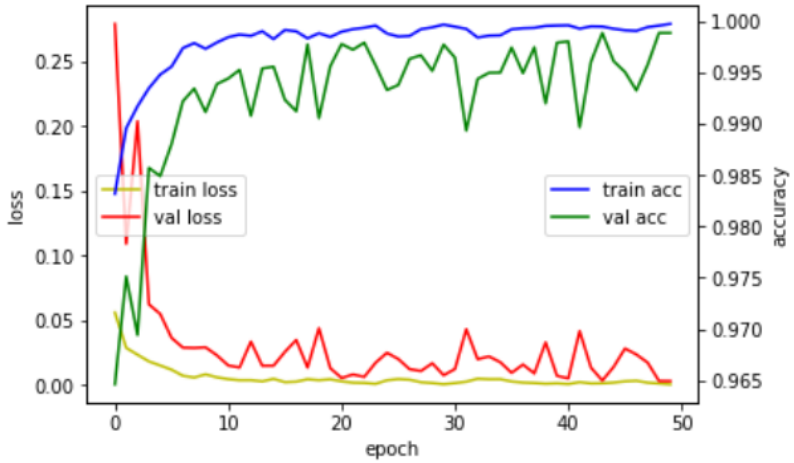
Figure 11. The schema for 17 × 17 grid (Inception-ResNet-B) module of Inception-ResNet-v1 network.

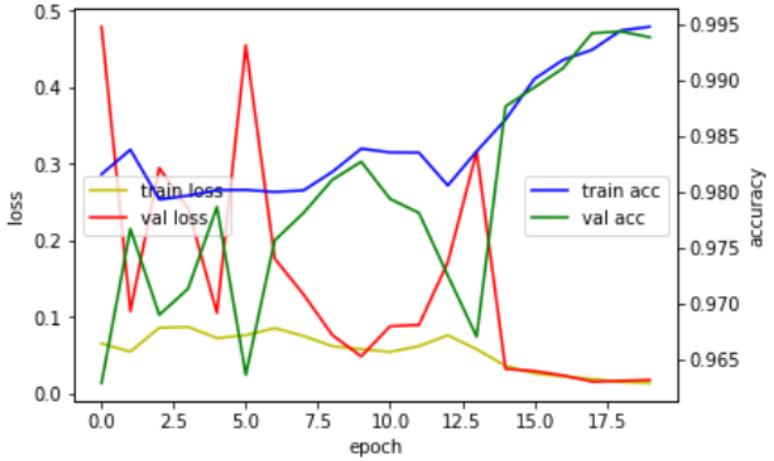
Figure 13. The schema for 8×8 grid (Inception-ResNet-C) module of Inception-ResNet-v1 network.



Inception-resnet-v2를 이용한 실제 모델 시행

- 사용 프레임 워크 : Keras
- 이미지는 원본에서 bounding-box부분을 crop한 것을 씀.
- Cv2와 os, scikit-learn 모듈을 이용한 이미지 불러오기 및 전처리.
  - 75x75로 input 이미지 resize.
  - Label은 임의로 폴더명을 사용.(one-hot encoding)
- Numpy배열에 위의 전처리한 데이터를 넣어 .npy로 저장한 후, np.load로 불러옴.
- Batch : 64, epochs : 50(위) 20(아래).
- 모델
  - pretrained Inception-resnet-v2(ImageNet) 사용.
  - Fully connected 이전 부분의 가중치만 사용, softmax를 이용해 51 classes로 분류.
  - Optimizer : Adam
  - Learning rate : 0.001
  - Loss : binary_crossentropy

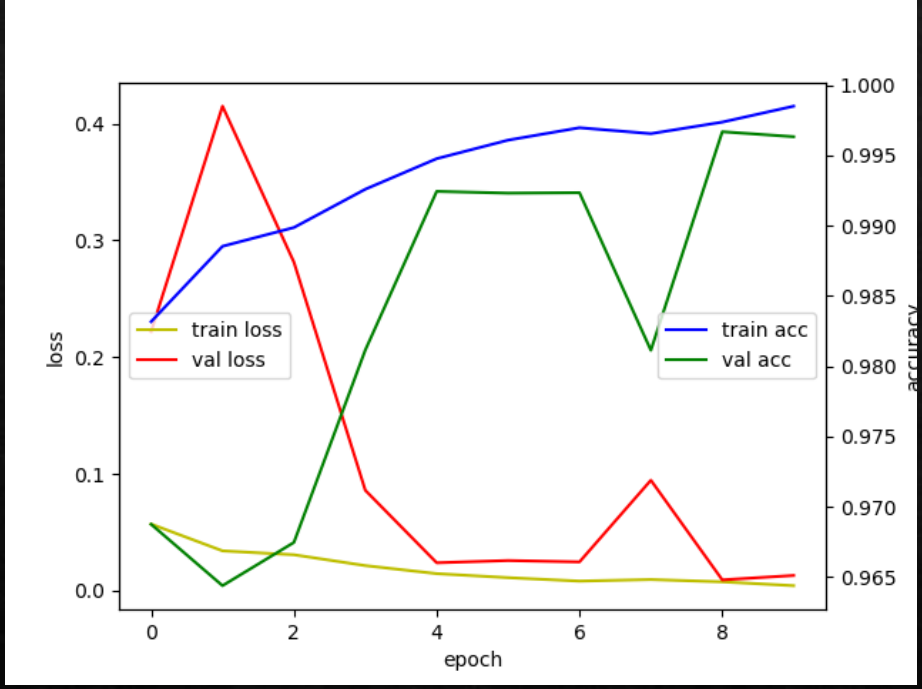




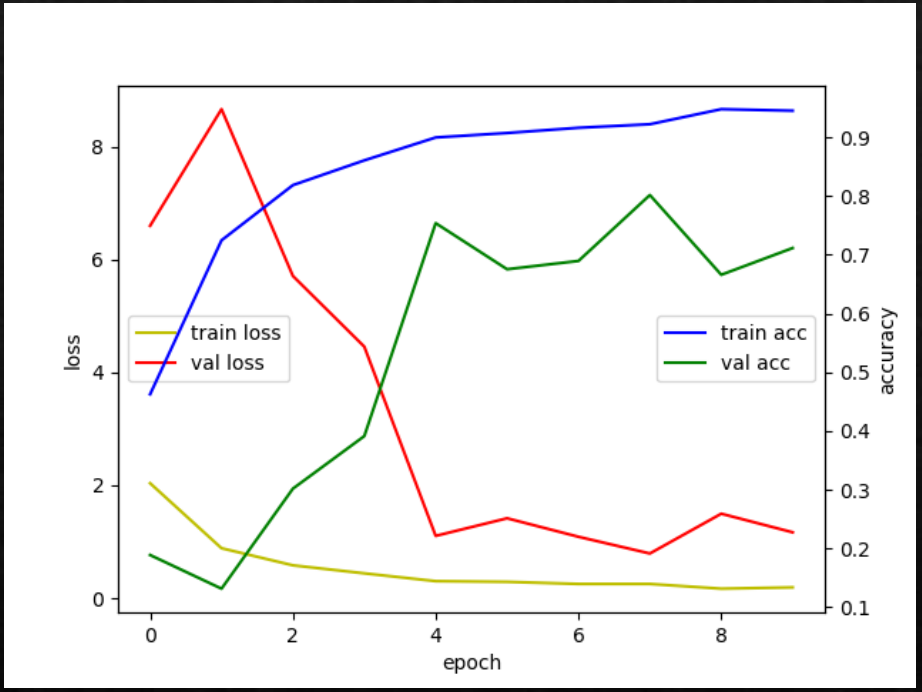
Inception-resnet-v2를 이용한 실제 모델 시행

- 특이점.
  - Resnet에 비해서 높은 정확도를 보여줌.
  - Image Augmentation한 것과 하지 않은 것에 대한 차이가 별로 없음.
  - Imagenet에서 weight를 가져오는 것과 그렇지 않은 것에 대한 차이가 별로 없다.
  - Loss function을 categorical_crossentropy로 하는 것보다 binary_crossentropy로 하는 것이 수렴 속도가 매우 빠름.

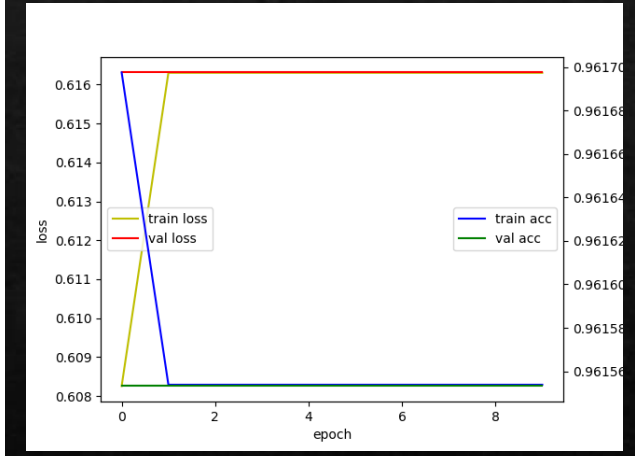
Inception-resnet-v2를 이용한 실제 모델 시행



<Inception with binary loss>



<Inception with categorical loss>



<Resnet50>