

이종대, 윤상석, 하지민 | 깃허브 링크 : <https://github.com/amamov/vgg16-vs-resnet>

## 1. Introduction

VGG16과 ResNet를 리뷰하고 코드 설계, 프레임워크 관점에서 성능을 비교 분석한다. 수업 시간에 배운 전통적인 CNN 모델과 유사한 VGG16 모델과 후에 나온 ResNet 모델의 성능 차이의 원인에 대한 궁금증을 바탕으로 주제를 선정하게 되었다. OOP 원칙에 의거한 재사용이 가능하고 확장성 있는 pytorch 기반의 머신 클래스를 설계하여 비교 대조 실험이 용이하게 하고자 하였다. 또한 pytorch 뿐만 아니라 keras, tensorflow 등 다양한 프레임워크를 활용하여 코드를 작성하며 각 프레임워크의 이점을 알아보고자 하였다.

## 2. Implementation Details

CNN은 사진, 동영상처럼 비주얼 데이터를 전처리할 때 기존의 FC보다 성능이 좋아 각광받았다. CNN은 FC와 달리 filter를 이용함으로써 인간의 시신경과 비슷한 receptive field를 갖게 되었다. Receptive field란 시각 세포에서 신호가 활성화되는 영역을 의미하며 Fully-connected layer와 달리 비주얼 데이터에 존재하는 공간정보와 채널정보를 모두 학습할 수 있다.

VGG 이전의 모델들은 다양한 크기의 비주얼 데이터에 receptive field의 크기, 즉 filter의 크기를  $7 \times 7$ ,  $5 \times 5$ ,  $3 \times 3$  등 다양하게 적용해왔다. 사실 Receptive field는 filter가 쌓이면 쌓일 수록 커진다. 그래서 크기가 큰 filter 하나를 쌓는 것과 작은 filter 여러 개를 쌓은 것의 receptive field는 동일하다. 예를 들어  $5 \times 5$  filter를 통과하여 생성된 kernel의 하나의 pixel은 25 pixel에 대한 공간정보를 가지고 있고,  $3 \times 3$  filter를 두 번 통과하여 생성된 kernel의 하나의 pixel은 직전 통과된 filter를 통해 9 pixel를, 그리고 해당 9 pixel은 맨 처음 통과한 filter를 통해 25 pixel에 대한 공간정보를 갖게 된다. 즉,  $3 \times 3$  filter 두 개를 연달아 사용하는 것과  $5 \times 5$  pixel을 하나 쌓는 것은 동일한 크기의 receptive field를 갖는 것이다. 이를 반복 적용하면  $3 \times 3$  filter를 3개 쌓는 것은  $7 \times 7$  filter 하나를 쌓는 것과 같은 receptive field를 갖는다.

이로 인한 이점은 두 가지이다. 하나는 파라미터의 수를 줄일 수 있다는 것이다.  $7 \times 7$  filter 하나는 49개의 파라미터를 갖지만  $3 \times 3$  filter 3개는 27개의 parameter를 갖는다. 이는 약 55% 감소시키는 효과가 있고 줄어든 파라미터만큼 더 층을 쌓음으로써 성능을 더욱 향상시킬 수 있었다. 나머지 하나는  $3 \times 3$  filter를 여러 개 사용함으로써 활성화 함수를 더 많이 통과시킬 수 있고 결과적으로 비선형성을 더 많이 부여할 수 있다는 것이다. 이런 연구 결과에 따라 filter의 사이즈는  $3 \times 3$  또는  $1 \times 1$ 만 사용한다.

하지만  $3 \times 3$  filter를 무작정 더 쌓는다고 계속해서 성능이 오르지는 않았다. 층을 더 쌓는 경우 chain rule에 따른 작은 gradient updating으로 gradient vanishing 현상이 발생했고, 그 결과 성능이 더 낮아지는 degradation현상이 생겼다. 따라서 이를 해결하기 위한 방안으로 skip connection을 통한 residual, 즉 학습이 잘 이루어지지 않는 잔여 feature에 대해서만 학습을 시키는 것이다. skip connection은 이전 layer까지 학습된 결과를 그대로 다음 층으로 넘김으로써 성능 향상에 대한 학습의 어려움을 덜어주는 역할을 한다.

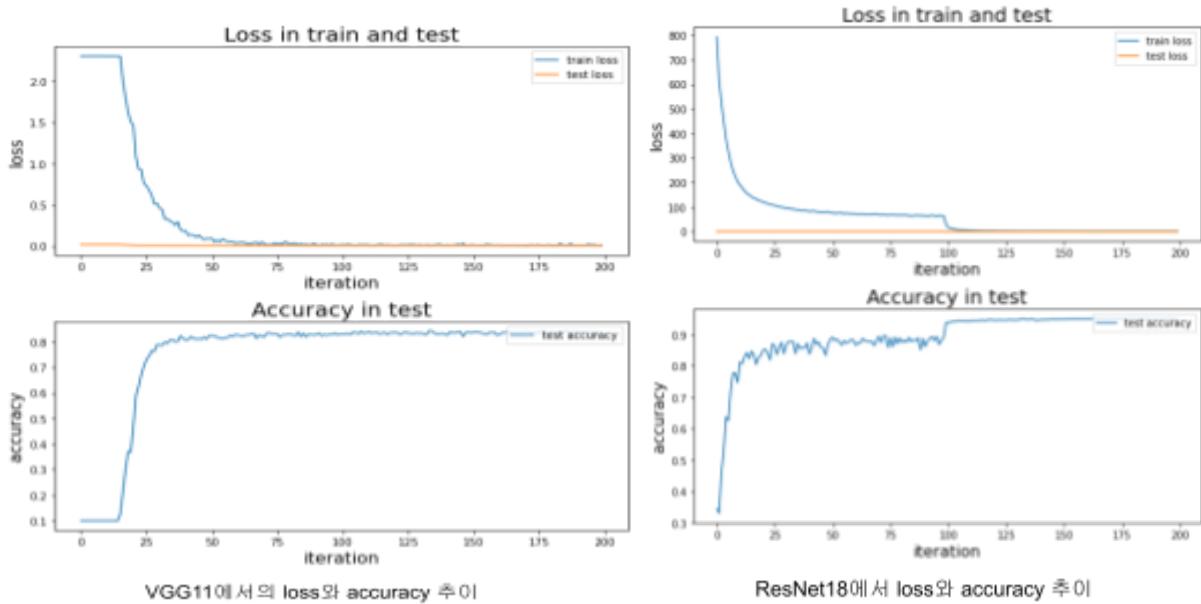
실험 방법은 팀원 각각 각기 다른 관점에서 주제에 접근하였다. 먼저 이론적으로 ResNet과 VGG16 성능 차이를 분석하고 실제 코드로 학습을 시키면서 결과를 확인하면서 실제 ResNet 모델과 VGG16 모델의 성능 차이가 나는 이유를 이해하였다. 그리고 캡슐화, 추상화를 사용하여 pytorch 기반의 VGG16 모델 머신 디자인 클래스를 설계하고 ResNet 모델 머신에서 상속받아 batch\_size, epoch\_size, learning\_rate, momentum 파라미터를 조정할 수 있도록 클래스 설계를 하면서 대조 실험을 용이하고 효율적이게 하고 ResNet 뿐만 아니라 다양한 모델 실험이 쉽도록 구현했다. 또한 pytorch 대신에 keras와 tensorflow를 사용하여 각 프레임워크에 대한 장단점을 분석하면서 각 프레임워크를 적절하게 사용하는 방법을 익혔다. 공통적으로 CIFAR 10 데이터 셋을 사용하였다.

## 3. Experiment Result

[ ResNet vs VGG 실험 - 1 ] (이론적으로 ResNet과 VGG 성능 차이를 분석하고 실제 코드로 학습을 시키면서 결과를 확인)

vgg : <https://colab.research.google.com/drive/1A31cF0qSkkKzU33Ie4meR0fQquKQKtrH?usp=sharing>  
resnet : <https://colab.research.google.com/drive/19vcW0WKBE1YAIGVEi3MlpicqTTKYLUPD?usp=sharing>

VGG에 대해서 11층까지  $3 \times 3$  filter와  $1 \times 1$  filter만을 사용했을 때 11층까지 쌓을 수 있었고 그 결과 test data에 대해 accuracy가 약 83% 나다. 크기가 32\*32인 CIFAR-10 데이터를 이용하여 층을 더 쌓아도 accuracy가 더 눈에 띠게 올라가지는 않았다. 여기에 identity mapping을 적용한 skip connection을 층마다 연결해준 결과, 층을 더 쌓을 수록 성능 향상이 있었으며 18층까지 쌓은 결과 test data에 대해 accuracy가 12% 가량 증가하여 약 95%가 나왔다.



### [ ResNet vs VGG 실험 - 2 ] (OOP 원칙에 따른 확장성있고 재사용 가능한 실험 코드 설계)

<https://colab.research.google.com/drive/1vkRvQHx9Xu4sOfqWBV87OnLrVTiReUNi?usp=sharing>

위 코드는 기본적인 CIFAR10 데이터셋을 기반으로한 재사용 가능한 VGG16 모델 머신 클래스를 설계하고 ResNet 모델 머신에서 상속받아 batch\_size, epoch\_size, learning\_rate, momentum 파라미터를 조정할 수 있도록 클래스 설계를 했다. 슈퍼 클래스인 VGG16 모델 머신 클래스를 기반으로 ResNet 뿐만 아니라 다양한 모델 기반의 머신 클래스를 추가적으로 상속을 통해 설계할 수 있다. 이로써 다양한 모델을 쉽게 실험할 수 있다. 또한 슈퍼 클래스를 다형성있게 사용할 수 있도록 했다. 예를 들어 커스텀 데이터셋을 사용하고 싶을 경우 해당 메서드를 오버라이딩해서 편하게 사용할 수 있도록 했다. 하나의 모델에서 4개의 하이퍼 파라미터를 조정하면서 비교 분석도 가능하다. 위 코드로 학습시킨 결과 **ResNet vs VGG 실험 - 1** 와 비슷한 결과로 동일한 데이터셋과 환경에서 VGG 모델과 ResNet 모델의 성능 차이는 약 8%만큼 차이 났다.

### [ ResNet vs VGG 실험 - 3 ] (Keras 사용)

vgg: <https://colab.research.google.com/drive/1m9yBd2TOc3BUWX64DBpoj0CHgig-mRRn?usp=sharing>

resnet: [https://colab.research.google.com/drive/19qkST37e\\_xMcVxiXEKJYSB8guVMIbb4-?usp=sharing](https://colab.research.google.com/drive/19qkST37e_xMcVxiXEKJYSB8guVMIbb4-?usp=sharing)

vgg와 resnet에 대해 자료 조사를 하면서 cnn을 구현하는 코드에서 keras 프레임워크가 빈번하게 사용되는 것을 볼 수 있었다. 그리고 그들은 비교적 간결하게 작성되어있는 것을 알고 keras를 활용하여 코드를 작성해보고 pytorch와 비교했을 때 keras의 이점이 무엇인지 파악해보고자 하였다.

keras에는 다양한 dataset이 내장되어있어 우리가 사용할 cifar10 dataset도 손쉽게 불러올 수 있었고, 데이터 전처리 과정을 돋기 위한 다양한 class를 제공하여 cnn의 학습 효과를 증대시켰다. 대표적으로 ImageDataGenerator는 학습 도중에 이미지에 임의 변형 및 정규화를 적용하고 변형된 이미지를 배치 단위로 불러올 수 있는 generator를 생성하여 모델이 학습 데이터에만 맞춰지는 것을 방지하고, 새로운 이미지도 잘 분류할 수 있게 만들었다. 이렇게 캐라스는 일반 사용 사례에 최적화된 간단하고 일관적인 인터페이스를 제공하는 반면, pytorch는 수학적으로 연관된 더 많은 사용자들을 위해 더 유연하고 저수준의 접근성을 제공하기 때문에 keras는 업계, pytorch는 학계에서 활발히 활용된다고 한다.

## 4. Conclusion

해당 코드를 확장하여 추후에 여러가지 다른 모델들을 비교 분석을 할 계획이다. 또한 하나의 모델에서 4개의 하이퍼 파라미터를 조정하면서 비교 분석도 할 계획이다. 이번 프로젝트를 하면서 아쉬운 점은 실험을 하는데에 학습 시간이 오래 걸려서 어려움을 겪었다. colab 무료 서버 사용 특성상 러닝중 쉽게 끊기는 경우도 빈번했다. 이를 보완하기 위해 다음 프로젝트에서는 러닝 중간에 결과를 캐싱하는 코드도 구현해야 할 계획이다. 또한 기본적으로 criterion와 optimizer를 각각 CrossEntropyLoss와 SGD로 구현했는데 이를 유연하게 바꿀 수 있도록 설계할 계획이다.

## 5. Reference

- VGG16 논문:<https://arxiv.org/pdf/1409.1556.pdf>
- ResNet 논문:<Deep Residual Learning for Image Recognition>