컴퓨터 비전 프로젝트 최종발표

ResNet

Deep Residual Learning for Image Recognition

(CVPR 2016)

2023. 6. 8.

20194147 김동현

20192568 이나현

목차

ResNet

1 문제 제기 / 필요성

CNN 모델 소개 및 문제점

[CNN 모델 소개]

- CNN 모델은 image recognition, image classification 등 컴퓨터 비전 관련 task에서 사용되는 neural network architecture
- convolution, pooling 등의 작업을 통해 이미지의 다양한 feature level을 추출하고 학습

[기존 CNN 모델의 문제점]

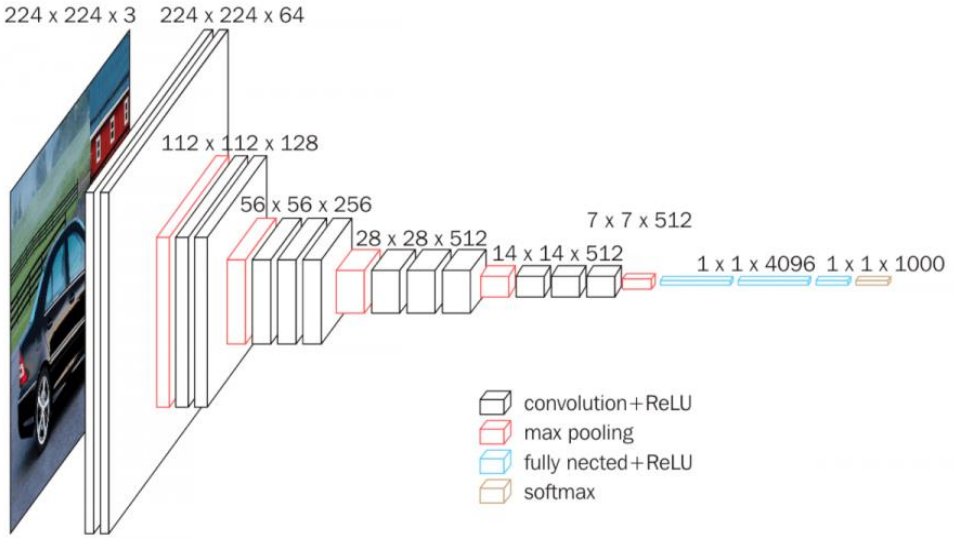
- 기존 CNN은 레이어가 깊어지면, 역전파 과정에서 Activation function 미분 값에 의해 그 값이 점점 희미해지는(0으로 수렴하는) Gradient Vanishing 현상이 발생
- CNN은 이론상 레이어가 깊어질수록 더 정확한 결과를 도출할 확률이 높아짐. 하지만 Gradient Vanishing 현상 때문에 깊은 레이어 구현이 불가능.
- 더 깊은 레이어를 구성하기 위해 Gradient Vanishing 현상을 억제해야 함

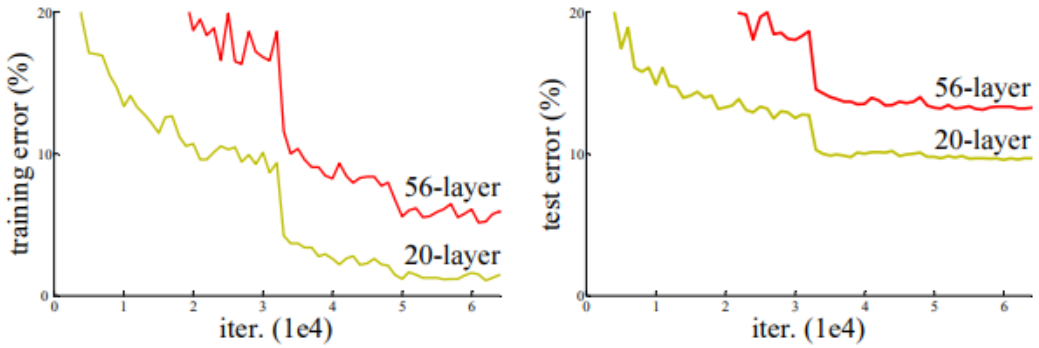
내용

기존 연구 소개 및 기존 연구들의 문제점

- VGGNet
  - 3x3 크기의 필터를 사용해 깊은 네트워크를 구성하여 이미지에서 특징 추출
- GoogleNet
  - Inception 모듈이라는 새로운 구조를 통해 깊은 네트워크 구성
- 기존 연구들의 문제점
  - Degradation Problem : Network가 깊어질수록 accuracy가 낮아지는 문제
  - 딥러닝 모델의 깊이 증가 -> gradient vanishing/exploding 문제 발생

  ⇒ hard optimization & low accuracy
- BUT
  - 이미지의 다양한 feature levels을 추출하고 학습하기 위해서 network의 depth는 중요





ResNet 이란?

- Residual Connection을 이용
  - 입력 값이 일정 층들을 건너뛰어 출력에 더해짐
- 모델이 깊어져도 gradient 소실 문제를 해결할 수 있음

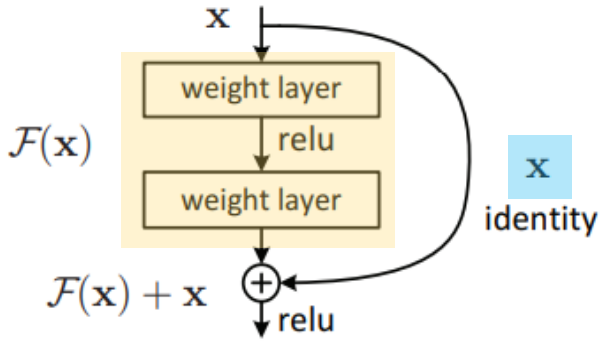
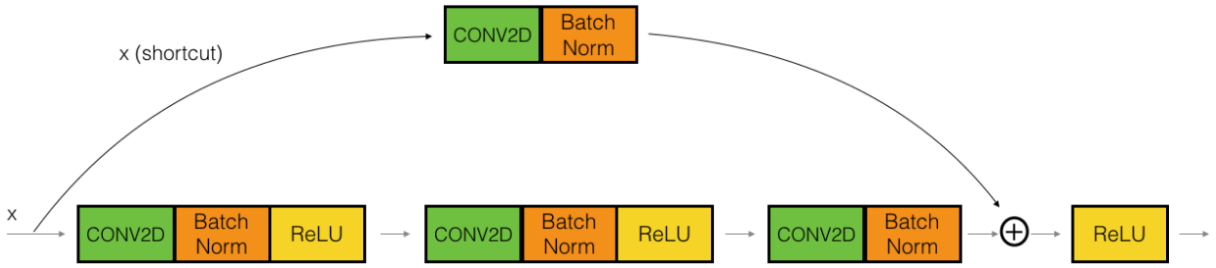


Figure 2. Residual learning: a building block.

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}.$$

layer name	output size	18-layer
conv1	112×112	
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$
	1×1	
FLOPs		1.8×10^9

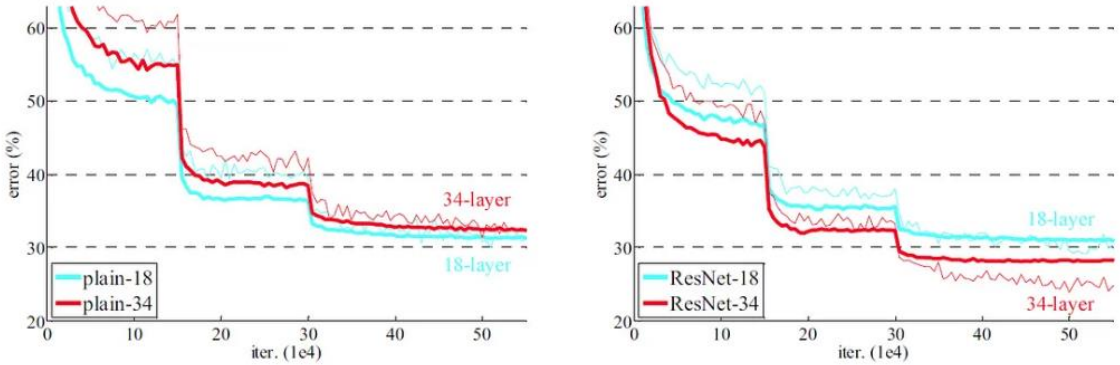


ResNet18 모델 구현

내용

기존 연구와의 차이점 (ResNet vs VGGNet)

- Gradient Vanishing 문제 해결
  - Residual Connection을 도입하여 gradient vanishing 문제 해결
- 더 깊은 네트워크 구조
  - 깊은 레이어를 쌓지 못하는 문제 해결
- 효율적인 연산
  - VGGNet에 비해 더 적은 파라미터 수를 갖기 때문에 연산 속도가 더 빠름



Validation Error: 18-Layer and 34-Layer Plain Network (Left), 18-Layer and 34-Layer ResNet (right)

	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	**25.03**

실험

실험 환경 및 데이터셋

실험 환경

- CPU i7-12700
- RAM 64GB
- GPU 3070ti
- 정확한 실험 결과 분석을 위해 random.seed 고정

데이터셋

- CIFAR10
- Why CIFAR10?
  - 원래 MNIST로 할 계획이었지만 실험 결과 첫번째 에폭의 정확도가 97.96% 나옴
  - 5번째 에폭에서는 99.44% 달성
  - 이것저것 바꿔가며 실험을 해보기에는 부적절하다고 판단

실험

Base Model

- 논문에 ResNet18로 CIFAR10을 실험한 결과 존재 X
- 논문에 나와있는 세팅 그대로 실험 진행 -> 해당 결과를 베이스 결과를 만들고 이것저것 바꿔가며 추가 실험 진행
- 논문 기본 세팅
  - Learning rate: 0.1 : 0.01 : 0.001 = 2 : 1 : 1
  - Activation Function: ReLU
  - Data Augmentation: No
    - 논문에는 몇가지 Data Augmentation을 했지만 실험을 위해 어떠한 augmentation 기법을 적용하지 않음
- Accuracy : 94.42%

실험

실험 소개

[Learning rate]

2 : 1 : 1
(Base Model)

5 : 3 : 2

5 : 4 : 1

[Activation Function]

Sigmoid

tanh

ReLU
(Base Model)

LeakyReLU

[Data Augmentation]

X
(Base Model)

기존 데이터 +
크롭 & 뒤집기 +
블러 & 색 변환
=> 15만개

기존 데이터 +
크롭 & 뒤집기
=> 10만개

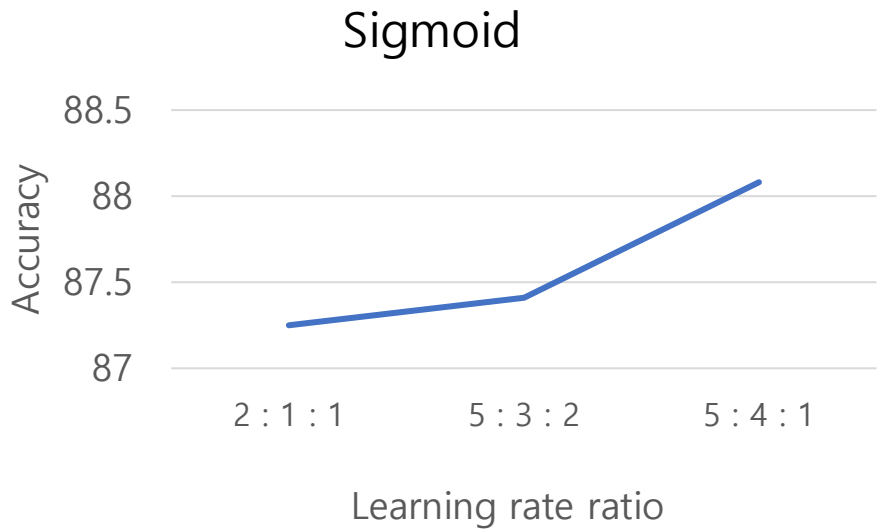
실험

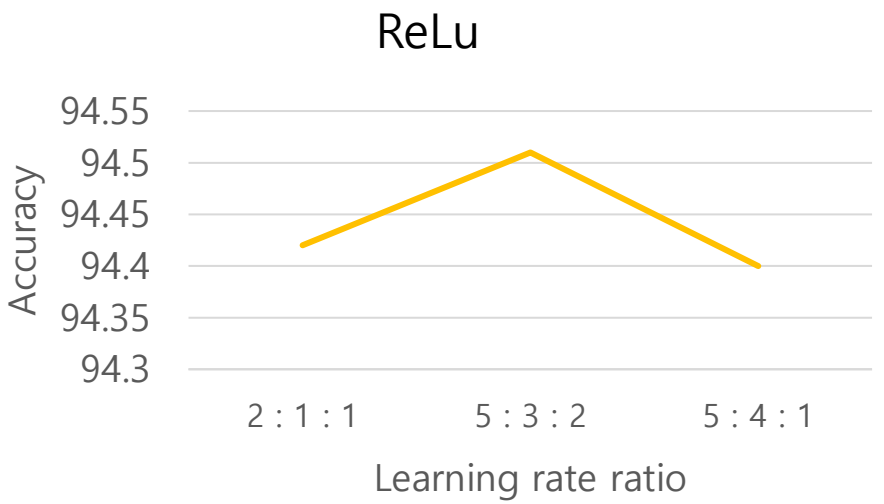
실험 결과 분석 - 표

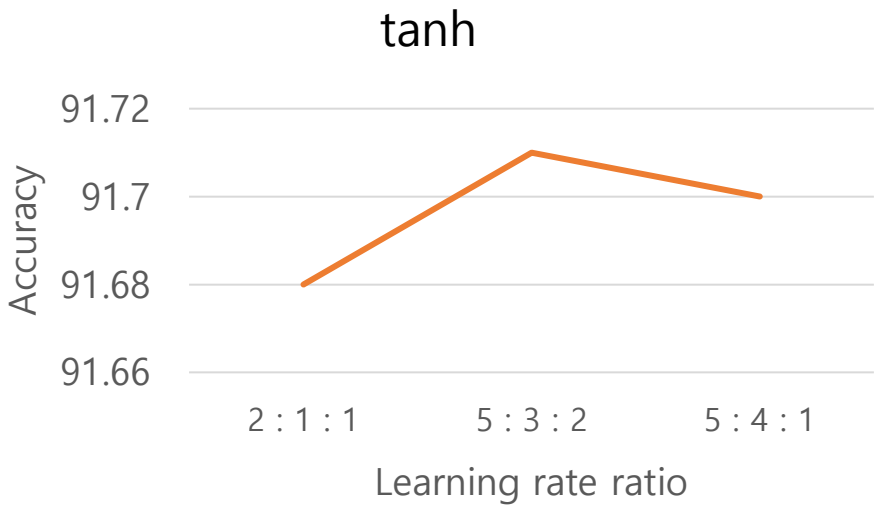
		2 : 1 : 1	5 : 3 : 2	5 : 4 : 1
Data Augmentation X	Sigmoid	87.25	87.41	88.08
	tanh	91.68	91.71	91.70
	ReLu	94.42	94.51	94.40
	LeakyReLU	94.41	94.54	94.28
Data Augmentation (crop&flip + color&blur)	Sigmoid	86.94	88.07	87.16
	tanh	91.35	90.97	91.28
	ReLu	93.12	93.51	93.15
	LeakyReLU	93.22	93.51	93.45
Data Augmentation (crop + flip)	Sigmoid	88.92	89.00	89.12
	tanh	92.12	92.54	92.84
	ReLu	94.98	94.99	95.14
	LeakyReLU	94.68	94.92	94.77

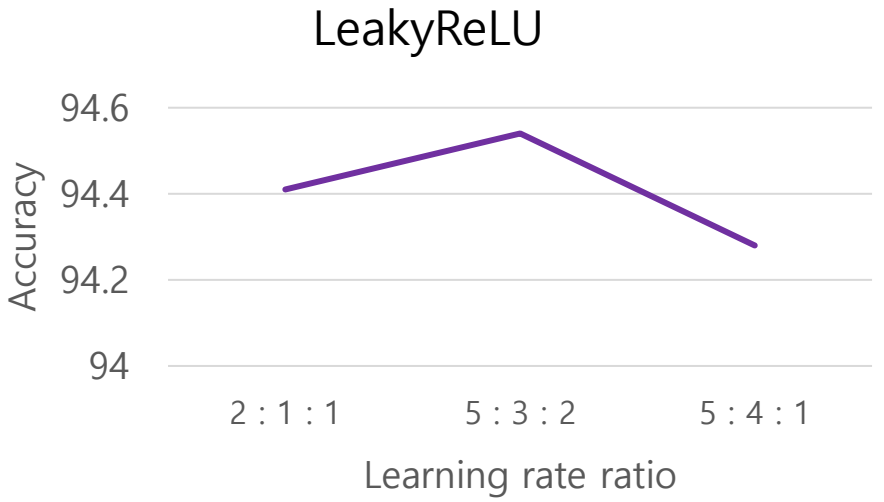
실험

실험 결과 분석 – Learning rate



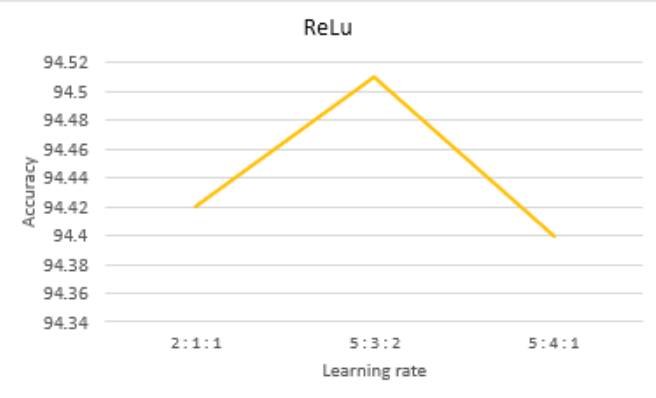


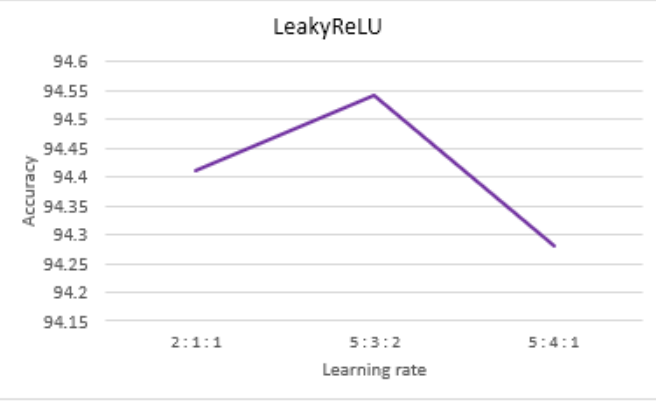




실험

실험 결과 분석 – Learning rate





[실험 결과]

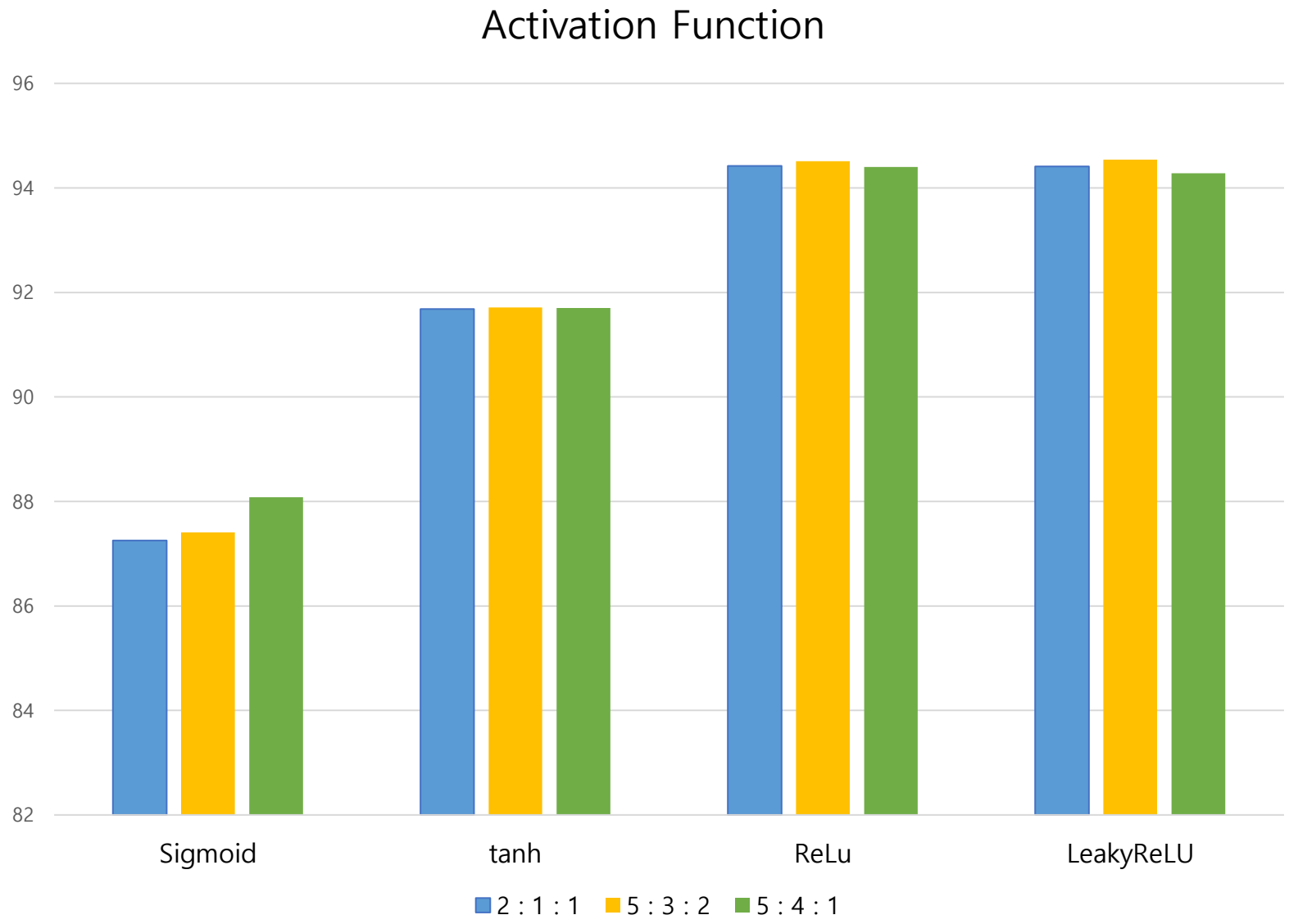
- Sigmoid를 제외한 나머지 활성화 함수에서 5:3:2 비율이 성능이 제일 높음

[결과 분석]

- 모델 초기에는 학습률이 크게 감소하고 점점 학습률이 작아지며 미세 조정을 함
- 즉 초기 학습 단계에서는 수렴을 빠르게 시키고, 이후에는 세밀한 조정을 함

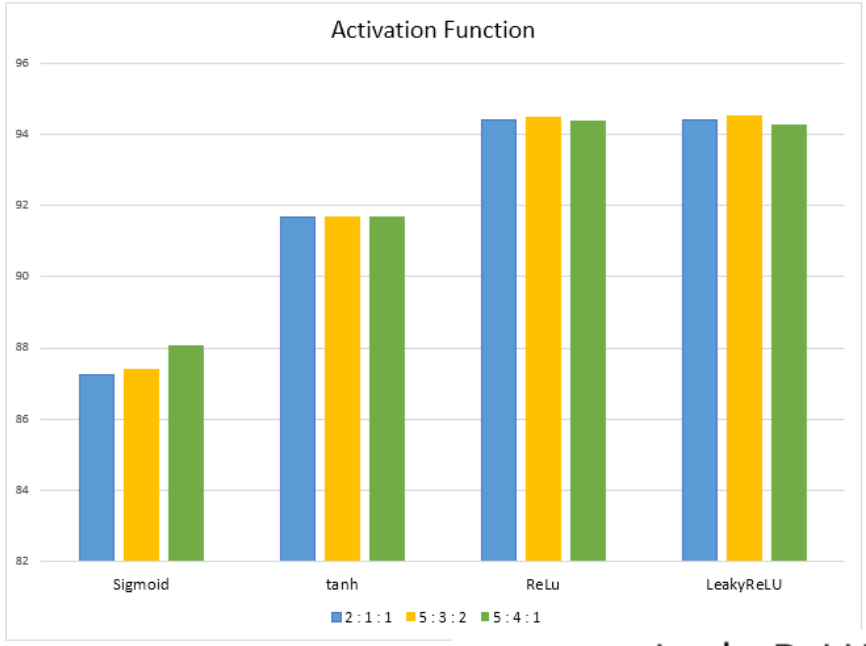
실험

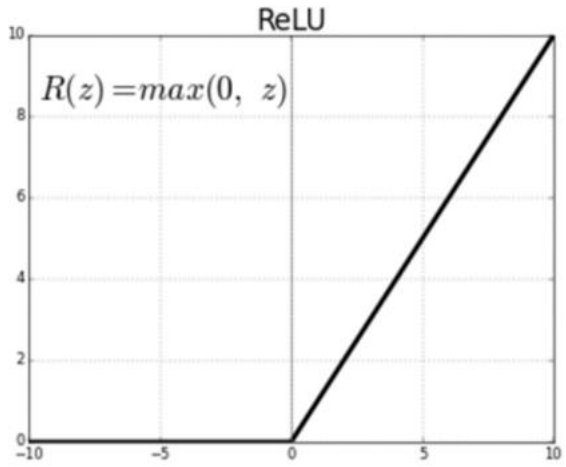
실험 결과 분석 – Activation Function

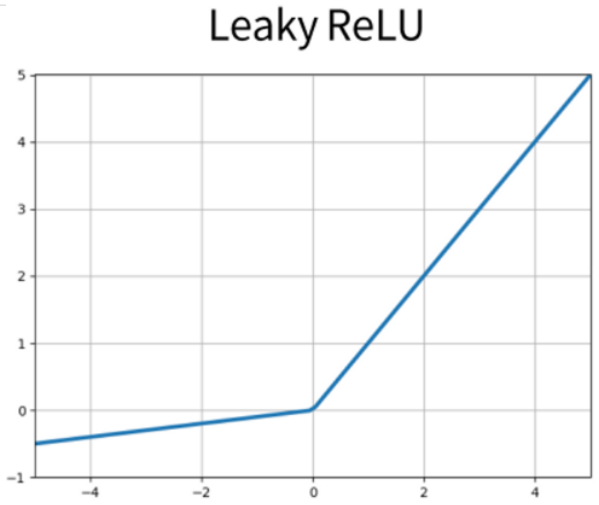


실험

실험 결과 분석 – Activation Function







[가정]

- Sigmoid < tanh < ReLU < LeakyReLU 순으로 성능이 좋아질 것이라고 예상

[실험 결과]

- ReLU와 LeakyReLU의 성능은 거의 비슷

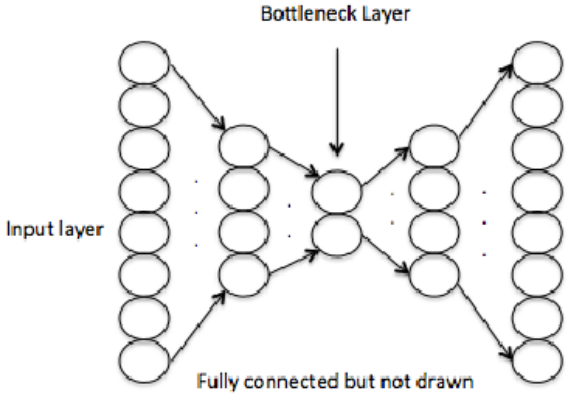
[결과 분석]

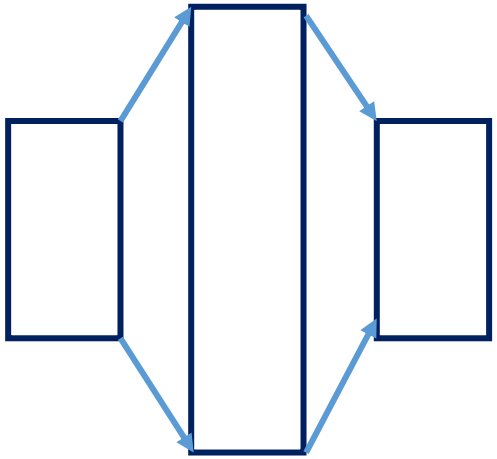
- dying ReLU 문제가 실제로 자주 발생하지 않음

실험

실패한 실험 소개

거꾸로 Bottle neck





ReLU(F(x)) + x

- 입력값이 더 강하게 유지되지 않을까?
- 성능 비슷

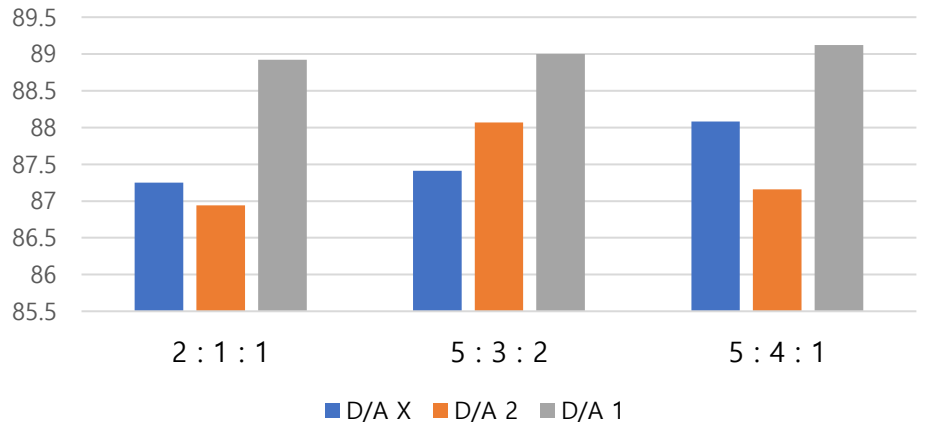
ReLU(F(x) + x^2)

- 입력값이 더 강하게 유지되지 않을까?
- 성능 Bad

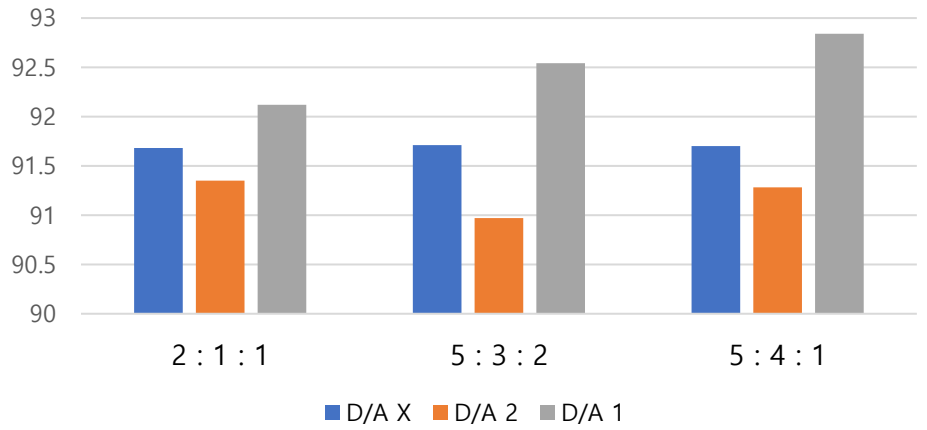
실험

실험 결과 분석 – Data Augmentation

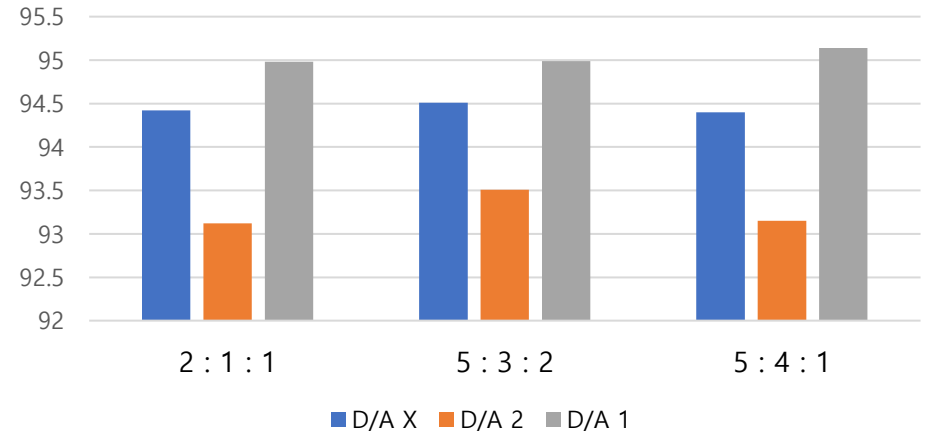
Sigmoid



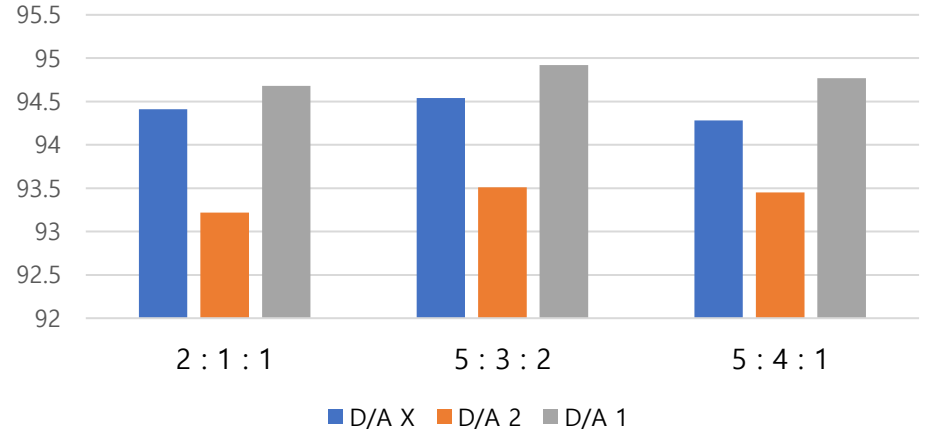
tanh



ReLU

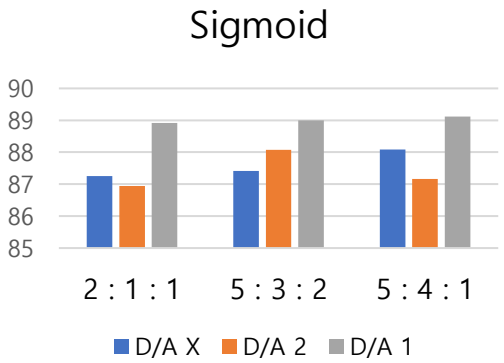


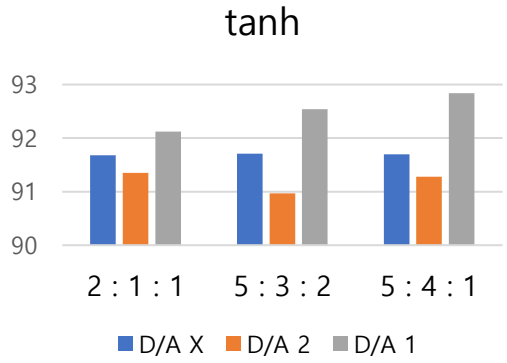
LeakyReLU

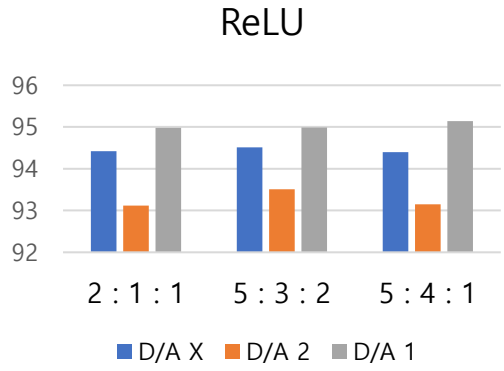


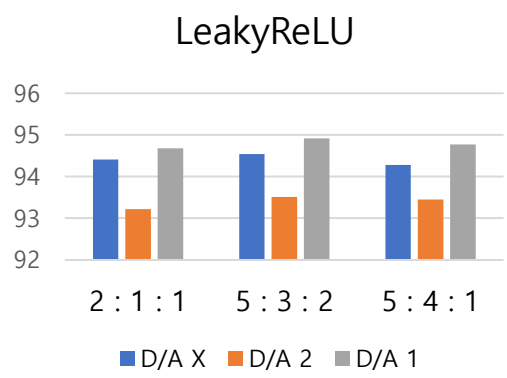
실험

실험 결과 분석 – Data Augmentation









[가정]

- Data Augmentation X < Crop/Flip < Crop/Flip + Color/Blur 순으로 성능이 좋아질 것이라 예상

[실험 결과]

- Color/Blur 을 적용하면 성능이 떨어짐
- Crop/Flip만 적용한 게 성능이 제일 좋음

[결과 분석]

- CIFAR10 데이터 특성 상 Blur 적용하는 것은 적합하지 않음
- 이미지가 작고 상대적으로 저해상도이기 때문에 블러링을 적용하면 세부 정보를 잃을 수 있음

4 결론

실험 결과 정리

- Learning rate 5:4:1
- Activation Function : ReLU
- Data Augmentation : Crop + Flip

  을 적용한 것의 성능이 가장 좋음

참고문헌

- He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
- 핸즈온 머신러닝, 오렐리앙 제롱, 한빛미디어, 2020
- 혼자 공부하는 딥러닝, 박해선, 한빛미디어 2020
- https://velog.io/@uonmf97/%EB%AA%A8%EB%91%90%EB%A5%BC-%EC%9C%84%ED%95%9C-%EB%94%A5%EB%9F%AC%EB%8B%9D-%EC%8B%9C%EC%A6%8C-2-ResNet-CIFAR10
- https://github.com/ndb796/Deep-Learning-Paper-Review-and-Practice/blob/master/code_practices/ResNet18_CIFAR10_Train.ipynb
- https://www.youtube.com/watch?v=671BsKl8d0E