이미지 분류를 위한 딥러닝 모델 학습과 고찰

Analysis of Deep Learning Models
and Their Learning for Image Classification

김영민
소프트웨어학부
중앙대학교

2020

Abstract

이미지 분류는 컴퓨터 비전 분야에서 중요한 이슈이다. 사람이 고양이의 얼굴을 고양이라 구분하는 것은 쉬운 일이나, 컴퓨터가 고양이를 고양이로 인식하는 것은 어렵다. 이미지 분류는 처음 보는 이미지를 효율적으로 100%에 가까운 정확도로 구분해가는 것은 끊임없는 연구과제이다. 딥러닝에선 데이터 기반방법을 통하여 이미지 분류를 한다. 데이터 기반 방법이란, 데이터를 기반으로 네크워크 프레임워크를 만들어 문제를 해결하고자 하는 방법이다. 딥러닝 이미지 분류에서 어떤 네트워크 프레임워크가 가장 좋다고 정하여 말할 수 없는 이유는, 어떠한 조건의 데이터셋을 가지고 있느냐에 따라 다를 수 있기 때문이다.

본 논문은 Resnet모델을 이용하여 Binary 와 Cifar10 dataset에 네트워크 프레임워크를 다양하게 차이를 주며, 바꾸어가며 기존의 사실들에 대한 검증과 비교 분석을 통하여 고찰을 해보았다. 우선, 각각의 데이터 셋에 활성화 함수(Swish, Relu, Leaky relu, tanh), 옵티마이저(SGD, Adam, RMSprop)를 다양하게 변화시키며 어떤 특징이 있는지 비교해보았고, Regularization(Weight decay, Dropout)이 어느정도 정확도와 Loss에 영향을 주는지 분석해보았다. 그리고 Cifar10에 data의 갯수를 줄여가며, 데이터 양이 정확도와 Loss에 어떤 영향을 미치는지 분석해보았다. 마지막으로, 앞서 분석하였던 활성화 함수들의 특징을 고찰해보며, 단점을 보완할 활성화함수 SEU(Swish Exponential Unit)를 제안해보며 Binary와 Cifar10에 적용시켜보았다.

1. Introduction.

1-1. RESNET[1]

Resnet을 설명 하기 앞서, CNN이 무엇인지 알아야한다. CNN은 Convolutional Neural Network의 약자이다. CNN은 Convolution(합성곱)을 이용하여 일반 Deep Neural Network에서 이미지나 영상과 같은 데이터를 처리할 때 발생하는 문제점을 보완한다.

Resnet은 Network가 깊어질수록 학습하기 어렵다는 문제를 해결하기 위하여 만들어진 모델이다. 이전과 달리 residual(잔차)을 학습하는 방법으로 더 깊은 network를 쉽게 학습시키는게 가능해졌다.

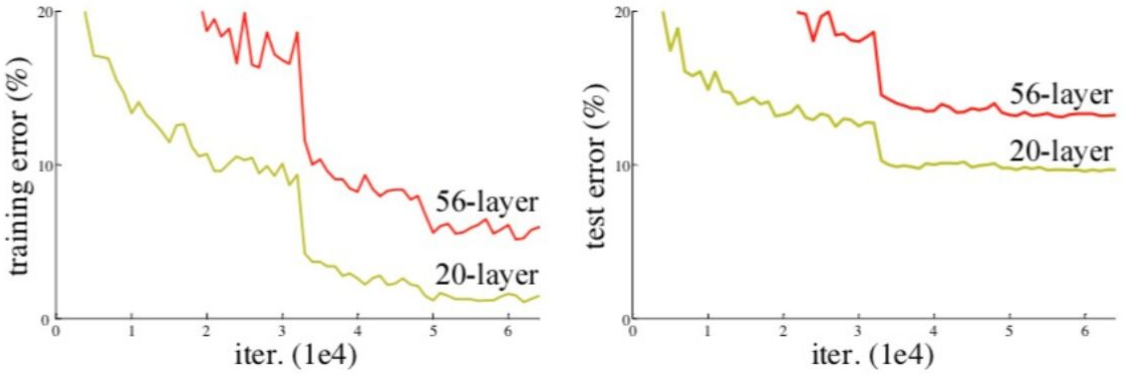


그림 1. Training error (왼쪽) and test error (오른쪽) on CIFAR-10

그림 1은 일반적인 Plain network에서 층을 더 깊게 만들면서 그에 따른 train loss와 test loss를 나타낸 그래프다. 네트워크를 깊게 만들다보면 한계를 지나면 오히려 성능이 떨어지는 모습을 보였다.

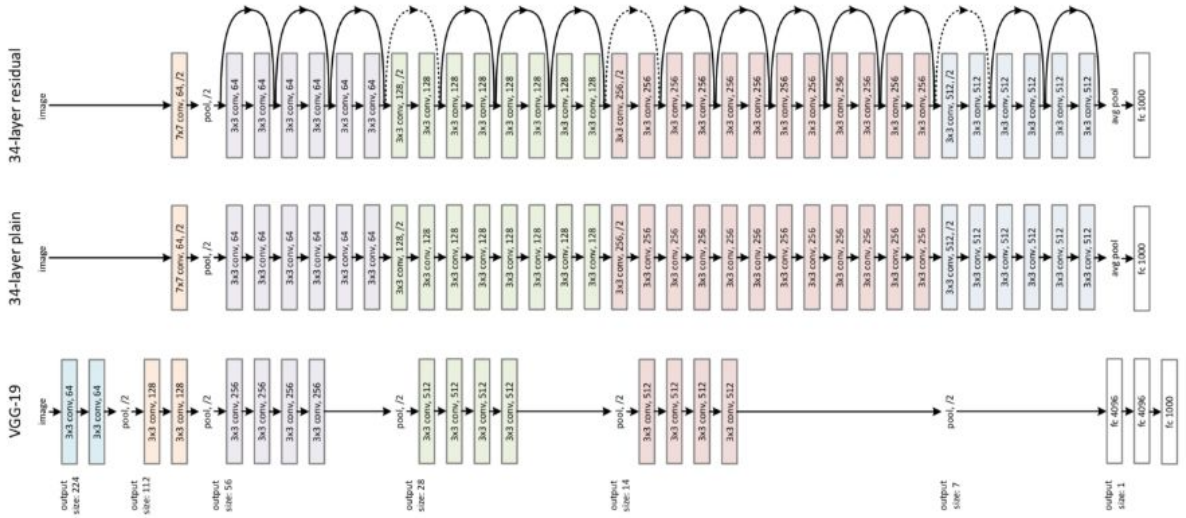


그림 2. Imagenet의 Network 구조의 예
(맨아래: VGG19 중간: Plain Imagenet 34 맨위: Resnet 34)

그림 2는 Resnet모델의 Network framework를 Plain network와 VGG network의 구조와 비교해본 그림이다. Plain 네트워크에선, 네트워크 층이 깊어지면 깊어 질수록 Degradation problem을 겪는다. Degradation problem이란, 망이 깊어지면 Vanishing gradient 문제 때문에 학습이 어려워지는 단점을 말한다. 다시 말하면, 네트워크가 깊어질수록, accuracy가 포화되어 degrade가 점점 빨라진다는 것이다. 이러한

Degradation문제는 Overfitting이 원인이 아니라, Model의 깊이가 깊어짐에 따라 training-error가 높아진다.

그림 2에서 Resnet모델을 참고하면, Short Connection을 하여 한 개 이상의 층을 생략하며, Residual Mapping을 한다. Resnet은 이전의 학습 방법들과 달리 Residual(잔차)를 이용하여 네트워크 층이 깊어질수록 성공적인 학습이 이루어질 수 있게 하는 Residual learning framework이다. 본 논문에서는 네트워크 프레임워크 Resnet18을 이용하여 이미지 분류를 테스트해보았다.

1-1. Dataset

Resnet18을 통하여 학습시키고 테스트한 Dataset은 Binary Dataset과 Cifar10이다. Cifar10이란, 일반적으로 머신러닝 및 컴퓨터 비전 알고리즘을 훈련시키는 데 사용되는 10개 이미지종류 모음이다. Binary Dataset은 흑백의 말과 사람 class가 있고, Cifar10은 컬러의 비행기, 자동차, 새, 고양이, 사슴, 개, 개구리, 말, 배, 트럭의 class로 이루어져있다. Cifar10은 5만개의 Train 이미지와 1만개의 Test 이미지로 이루어져 있으며, 효율적인 Classification을 위해선 깊은 네트워크 계층을 이용하여 학습시킨다. 반면, Binary Dataset은 Train 이미지가 1027개, Test 이미지가 256개로 Cifar10과 비교하여, 매우 적은량의 Dataset으로, 적은 층의 네트워크만으로도 그리 오랜시간을 요구하지 않고, Classification을 할 수 있다.

2. Binary Classification.

Binary dataset은 데이터의 양이 적고, Class 또한 2개로 간단히 이루어져있다. 비교적 간단한 데이터 셋에 대하여 Activation, Optimization, Regularization을 변화하였을 때 어떠한 특징을 보이는지 비교, 분석해보며, 결과에 대하여 고찰해았다.

학습에 이용한 데이터그룹은 말과 사람이다. 말과 사람을 구분하는 Binary Classification을 해보았다. 학습할 Trainset은 말 500개, 사람 527개로 총 1027개로 이루어져있고, 테스트용 Testset은 말 128개, 사람 128개로 총 256개이다. 각 이미지 한개당 크기는 100 x 100이고, 흑백이다.

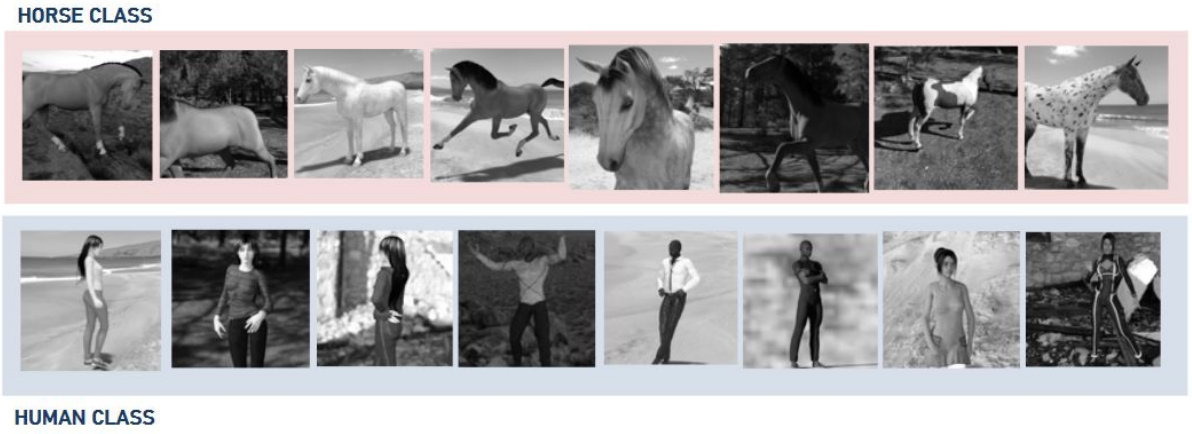


그림 3. Trainset에서 Class마다 무작위로 이미지 출력결과[2]

각각은 배경이 다르며, 말 이미지는 무늬, 밝기 ,위치가 다양하고, 일부분만 나온 사진도 있다. 사람 이미지도 마찬가지로 머리, 옷차림, 포즈, 밝기가 다양하다.

그림 4. Testset에서 Class마다 무작위로 이미지 출력결과[2]

Testset은 배경이 없는 이미지이고,, 각각의 말과 사람은 Trainset과 마찬가지로 다양하다.

2-1. 활성화 함수(Activation Function).

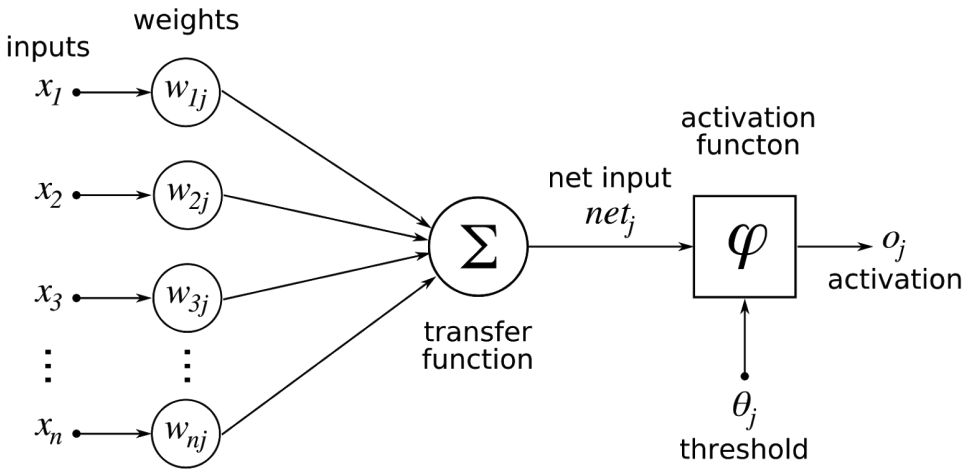


그림 5. 신경망의 구조

활성화 함수는 신경망의 출력을 결정하는 식이다. Binary 데이터셋에 대하여 Tanh[3], Relu[5], Leaky Relu[4], Swish[3] 활성화 함수를 적용시켜 비교해보았다. Optimizer는 공통적으로 Stochastic Gradient Descent(Mini batch = 8, Weight Decay = 1)를 이용하였으며, Loss Funtion으로는 Cross Entropy를 이용하였다. Learning Rate는 0.001로 시작하여 8,14 Epoch 에서 0.1씩 곱해 주었다.

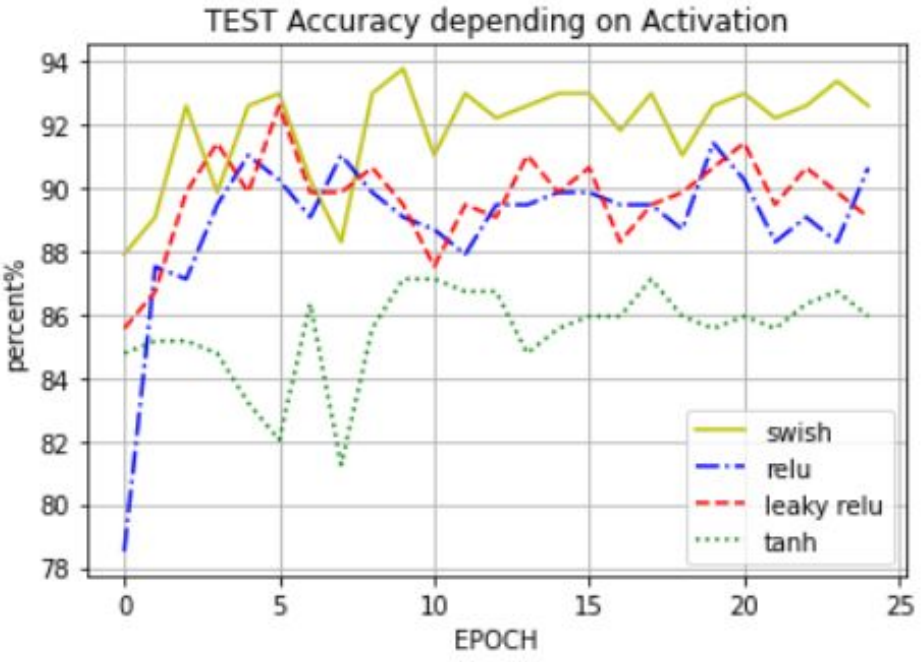


그림 6. TEST 정확도 그래프

Resnet18 모델에서 각각의 다른 활성화 함수에 대하여 매 Epoch마다 학습시켜 Testset에 대한 정확도를 그래프로 그려보았다. EPOCH 공통적으로 25회로 설정해주었다. Resnet18모델을 이용하여 비교적 적은 갯수의 Dataset에 학습시켰기 때문에 25회만으로도 Trainset은 약 99.7%정확도에 수렴하였다.

	SWISH	RELU	LEAKY RELU	TANH
평균 정확도	91.054%	89.764%	89.58%	85.94%
표준편차	0.77	1.03	1.26	0.98

그림 7. TEST 평균 정확도와 표준편차

그림 7는 그림 6와 같은 실험을 20번 반복하여 마지막 Epoch의 TEST 정확도들의 평균과 표준편차를 구해보았다. Relu와 Leaky Relu Activation Function을 사용하였을 때, Swish,와 Tanh보다 표준 편차가 크게 나타났다. Relu와 Leaky Relu는 선형함수의 Unit으로 이루어져있는 반면, Swish나 Tanh는 exponential함수로 이루어져있어 그래프 모양이 Relu나 Leaky Relu보다 Smooth하게 나타난다. 그렇기 때문에 파라미터들의 값이나, learning rate에 덜 민감하여 비교적 적은 Fluctuation을 보이게 된다.

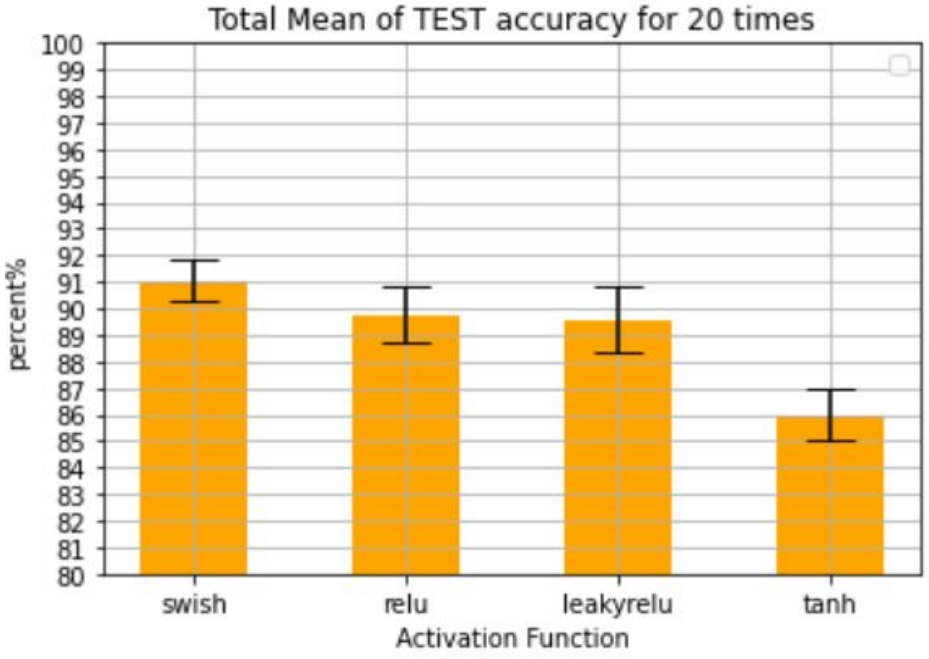


그림 8. 평균 TEST 정확도와 표준편차 막대 그래프

그림 8의 그림 7의 표 내용을 막대그래프로 표현에 보았다. Swish 활성화 함수를 적용 하였을 때, 평균적으로 가장 높은 정확도를 보였다. Leaky relu와 Relu 활성화 함수는 비슷한 정확도를 보였으며, Tanh 활성화 함수가 가장 낮은 정확도를 보였다.

Resnet18은 적지 않은 네트워크 층으로 이루어져 있으므로 tanh 활성화 함수를 이용하였을 때, Gradient vanishing 문제를 겪게 된다.

2-2. 최적화 (Optimization).

신경망 모델의 학습과 결과에 따른 손실의 값을 최소화하는 방향으로 weight의 값을 찾는 것을 최적화(Optimization)라 한다. Binary Dataset에 Optimization은 SGD, Adam, RMSprop를 이용하여 학습하였다. 공통적으로 Swish 활성화 함수를 이용하였다.

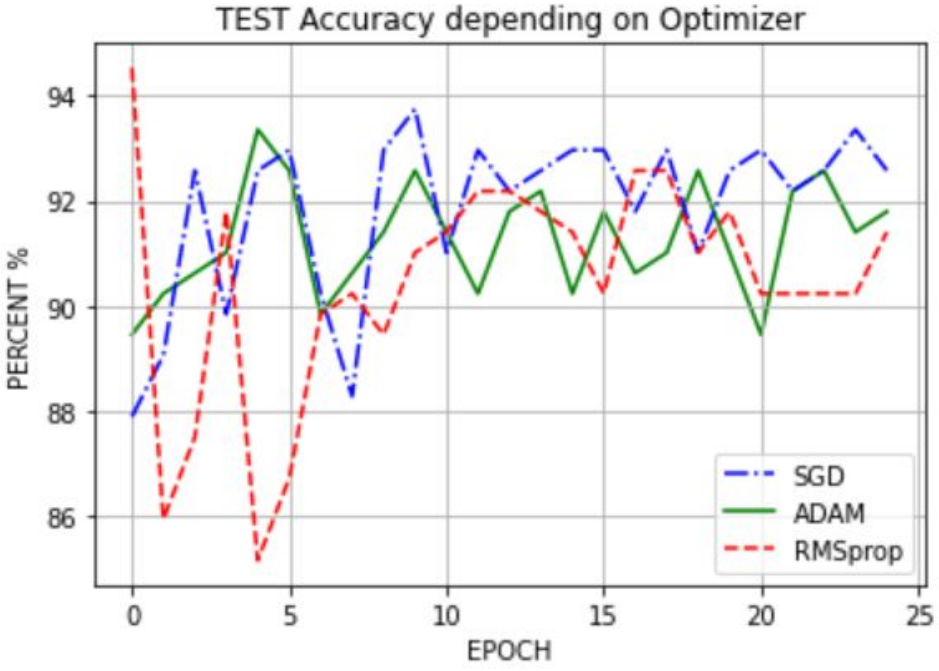


그림 9. Optimizer에 따른 Test 정확도 그래프

SGD, ADAM, RMSprop Optimizer를 Trainset에 학습시켰을 때, 그림 [9]의 그래프처럼 비슷한 Test 정확도를 보였다. SGD의 그래프는 0-10회에서 Test의 정확도가 88~93%를 보이고,15-25회에서는 91~93%의 Test 정확도를 보인다. 즉, epoch이 진행될수록, Fluctuation이 점점 안정화 되었으며, Test의 정확도가 점점 상승하는 추세를 보인다. RMSprop Optimizer는 첫 epoch 때부터, 높은 Test 정확도를 이뤄냈다. 그리곤 큰 진폭으로 Test 정확도가 진동하다 안정화된 모습을 보인다.

2-3. 정규화 (Regularization).

정규화(Regularization)란, 모델의 일반화 오류를 줄여 과적합을 방지하는 모든 기법을 총칭한다. Binary 데이터셋 학습에 Weight decay와 Dropout[6]을 이용하여 각각을 비교해보았다. 활성화 함수는 Swish를 이용하였고, Optimizer는 SGD를 이용하였다. 그림 [9]와 그림 [10]에서 이용된 Loss와 Accuracy는 같은 실험에서 동시에 학습된 결과이다.

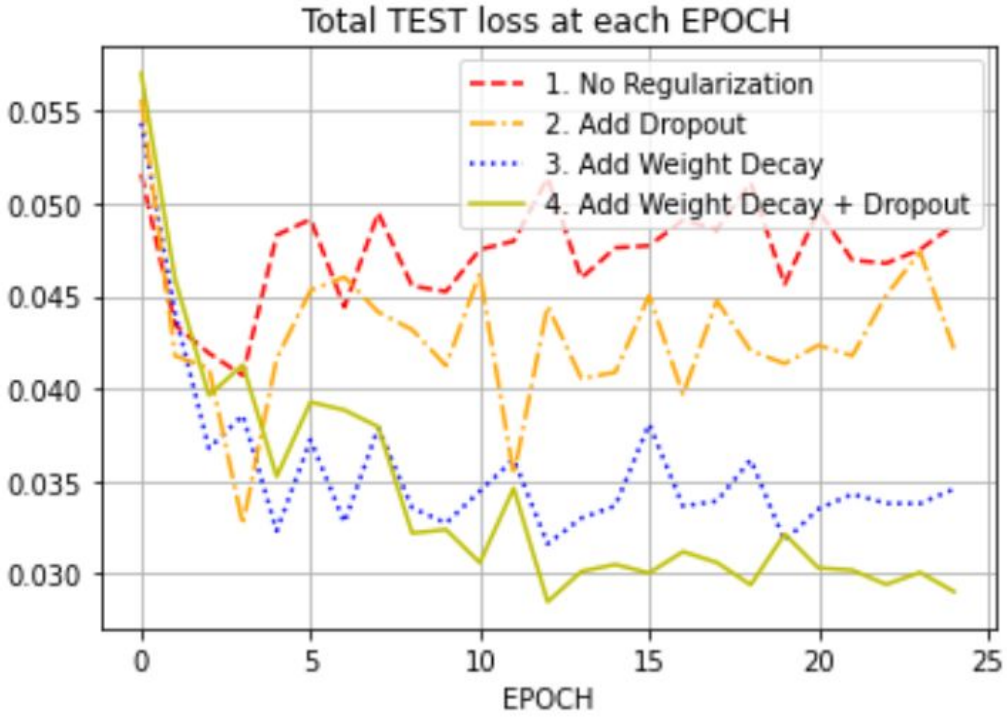


그림 10. Regularization에 대한 TEST Loss 그래프

그림 10에서 1번 그래프는 weight decay와 dropout을 설정하지 않았을 때, 2번 그래프는 신경망 유닛에서 dropout을 해주었을 때. 3번 그래프는 weight decay를 1로 해주었을 때, 4번 그래프는 weight decay와 dropout을 모두 설정해주었을 때, Test dataset에 대한 loss 그래프다.

1번은 loss가 0-5회 사이에서 loss가 0.035이하로 최저점을 찍고, epoch이 시행될수록 0.04-0.05 사이로 증가하는 추세를 보인다. 즉, 과적합(overfitting)이 발생하였다. 1번과 2번을 비교하였을 때, dropout을 해준 2번 그래프가 과적합이 조금 개선되었다. 하지만 epoch 0~5회 사이에선 0.03-0.045사이에 loss가 발생하는 반면, 20~25회 사이에선 0.04-0.05로 증가하여, 여전히 과적합이 발생하고 있다. 3번 그래프는 매 회 학습이 진행 될수록 Test loss는 감소하는 추세를 보인다. 1번과

비교하여, weight decay를 해주었을 때, 과적합 문제가 눈에 띄게 개선되었다. 결과적으로 weight decay와 dropout을 해준 학습이 가장 성능이 뛰어났다.

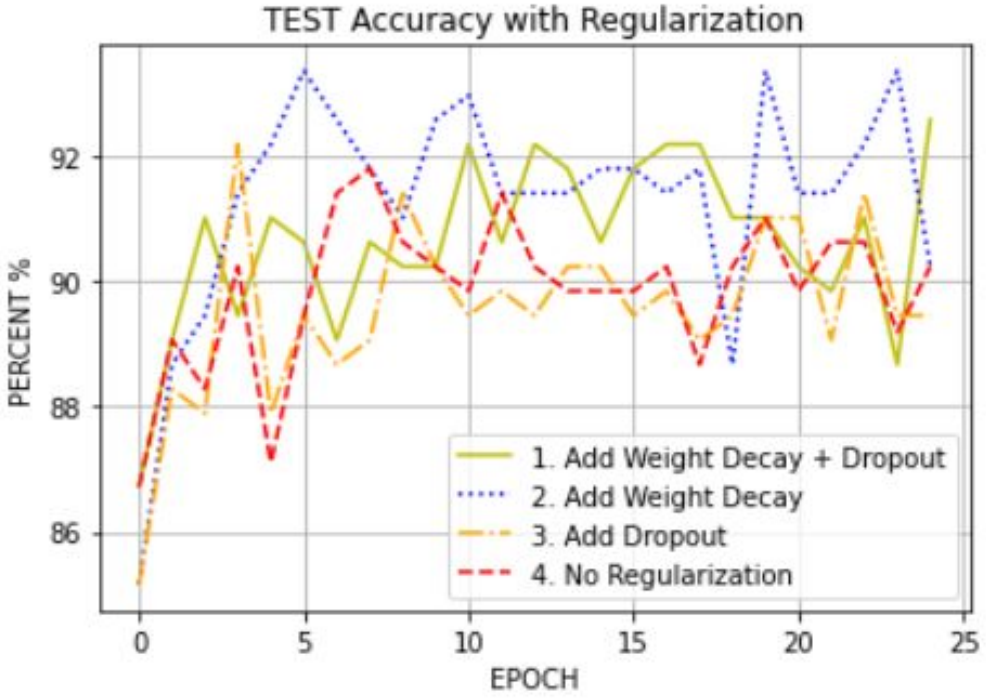


그림 11. Regularization에 대한 TEST 정확도 그래프

그림 11의 TEST 정확도 그래프는 그림[9]의 TEST loss 그래프만큼 눈에 띄게 차이가 크지 않았지만, 1,2번 그래프가 3,4번 그래프보다 전반적으로 높은 정확도를 보였다.

Test loss는 Regularization을 해준 쪽이 Test loss가 개선되었지만, 개선된 loss가 0.01~0.015 정도로 적은 량이기 때문에, Test 정확도는 큰 영향을 받지 않았다.

3. Cifar10 Classfication.

Resnet18을 이용하여 두번째로 학습에 이용한 데이터그룹은, 머신러닝이나 컴퓨터 비전 알고리즘을 훈련시키는데 흔히 사용되는 Cifar10이다. 이미지 사이즈 크기는 32x32이다. Cifar10은 컬러의 비행기, 자동차, 새, 고양이, 사슴, 개, 개구리, 말, 배, 트럭의 10개 class로 이루어져있다. 학습에 이용된 Trainset은 50000개로 각각 클래스당 5000개의 이미지, 테스트에 이용한 Testset은 10000개로 각각 클래스단 1000개의 이미지로, 총 60000개다.

Cifar10학습에는 공통적으로 60회 Epoch을 학습하였고, Train 정확도가 99.5%에 수렴하였다. 0.1 Learing rate를 설정하였고, 30회 45회 때, 0.1씩 곱해주어 변화하도록 설정하였다.

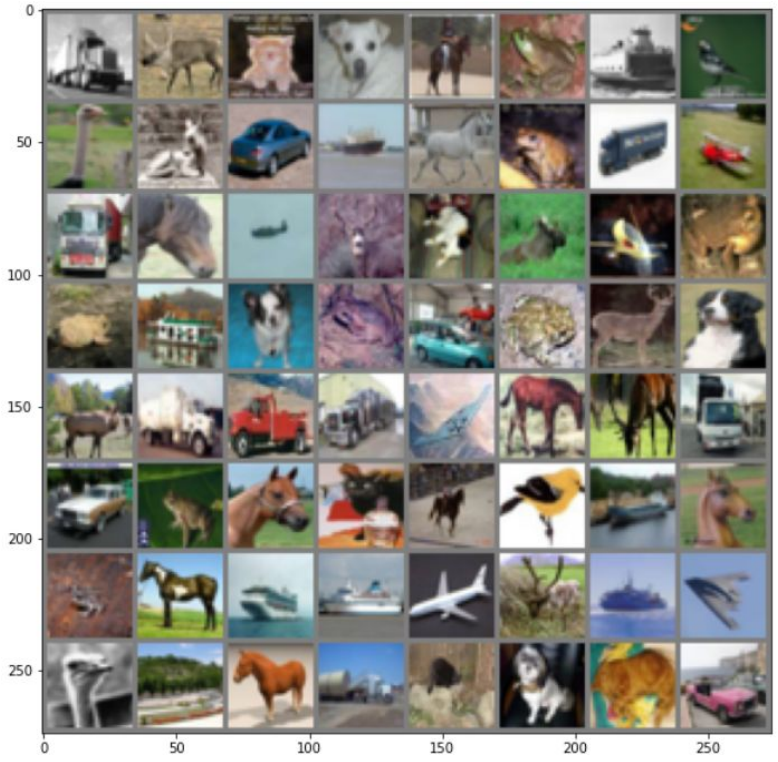


그림 12. Cifar10의 이미지 무작위 64개 출력결과

3-1. 활성화 함수(Activation Function).

Binary 데이터 셋과 마찬가지로, Cifar10 데이터 셋을 swish, relu, leaky relu, tanh 활성화 함수에 대하여 학습해보았다.

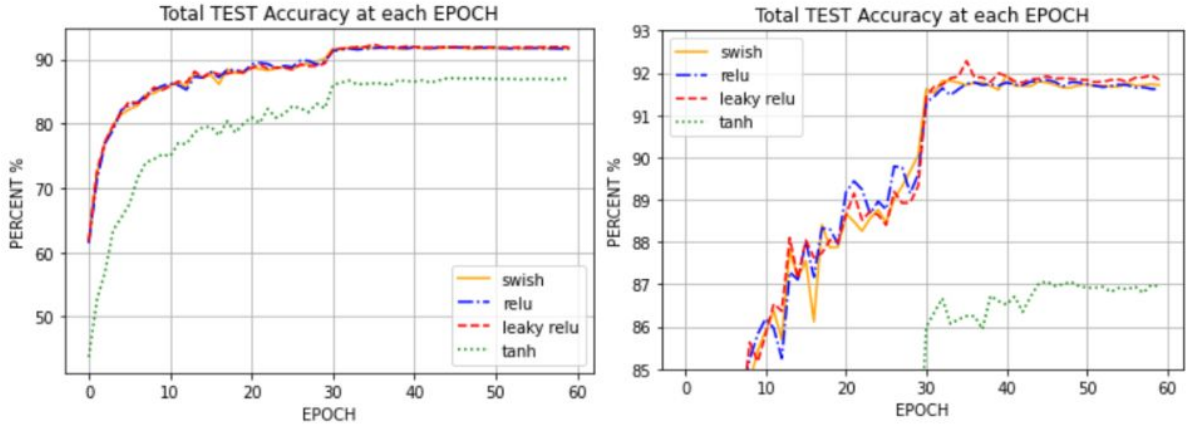
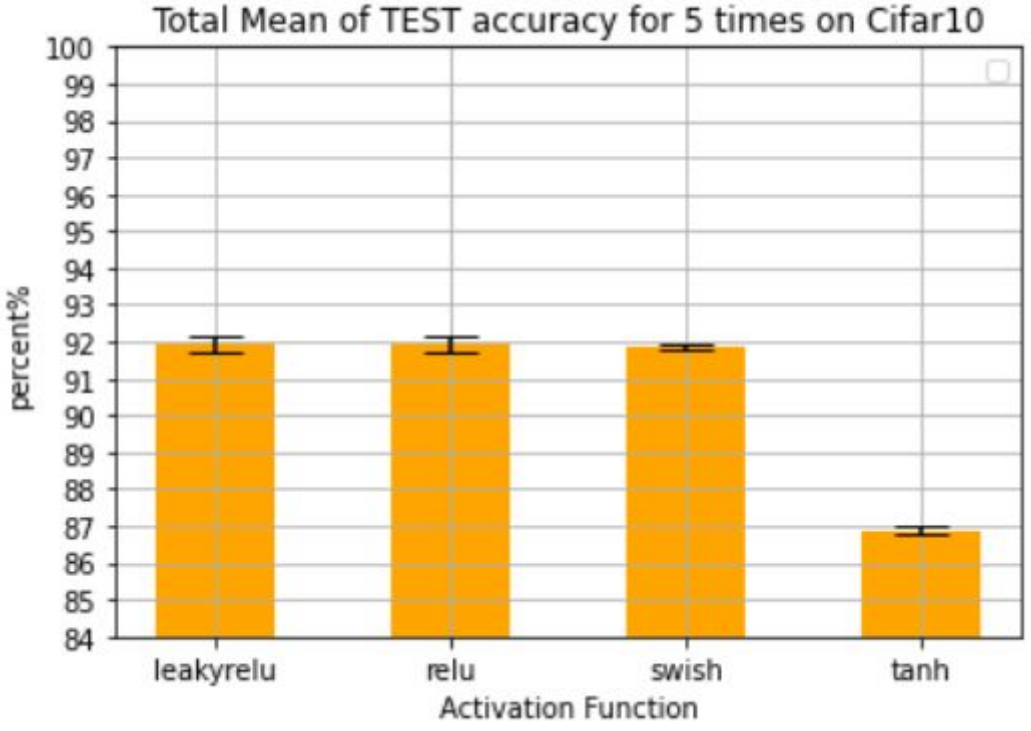


그림 13. 활성화 함수에 따른 TEST 정확도 그래프

그림 13은 활성화 함수에 따라, TRAIN Loss가 어느정도 수렴하고, 정확도가 99.5%일 때, TEST 정확도 그래프를 나타낸 것이다. 좌우 그래프 표는 동일한 그래프이나, 차이를 분명히 보기 위하여, 좌측 그래프를 확대한 것이 우측 그래프 모습이다. tanh 활성화 함수를 사용하여 학습 하였을 때, 가장 낮은 Test 정확도를 보였다. 아래 그래프 표를 보면 swish, relu, leaky relu 활성화 함수는 거의 비슷한 정확도를 보여준다.

Binary dataset을 활성화 함수를 변화하며 학습한 그림 5와 Cifar10을 학습한 그림 11을 비교하면, 데이터가 풍부한 Cifar10의 그림 11가 그래프 모양의 Fluctuation (변동)이 적음을 보인다.



	LEAKY RELU	RELU	SWISH	TANH
평균 정확도	91.96%	91.92%	91.86%	86.88%
오차	0.24	0.21	0.06	0.12

그림 14. 활성화 함수에 따른 평균 TEST 정확도와 표준편차 막대 그래프

그림 14는 그림 13와 같은 실험을 5번 반복하여, 마지막 Test 정확도 값들의 평균과 표준편차를 막대그래프로 표현하였다. TEST 정확도의 평균값 또한 swish, relu, leaky relu 활성화 함수들은 비슷한 정확도를 보인다.

그림[14]에서 tanh와 swish가, relu와 leaky relu보다 비교적 작은 표준편차를 보였다. tanh와 swish 활성화 함수의 그래프는 곡선로 이루어져 있기 때문에, relu나 leaky relu 활성화 함수보다 초기값과 Learning rate에 대하여 덜 민감하다. Binary Dataset에 각각 다른 활성화 함수를 적용하여 나타낸 평균 Test 정확도 막대그래프 그림 [8]에선 tanh와 swish 활성화 함수의 표준편차가 눈에 띄게 relu와 leaky relu보다 작지 않았는데, 이유는 Binary dataset의 양이 너무 적어 생기는 Fluctuation이 더 크게 작용하기 때문이다.

3-2. 최적화 (Optimization).

Optimizer로 SGD, Adam, RMSprop, Momentum(SGD에 추가)를 이용해 Cifar10을 swish와 relu 활성화 함수를 이용하여 학습해보았다.

■ SWISH 활성화 함수

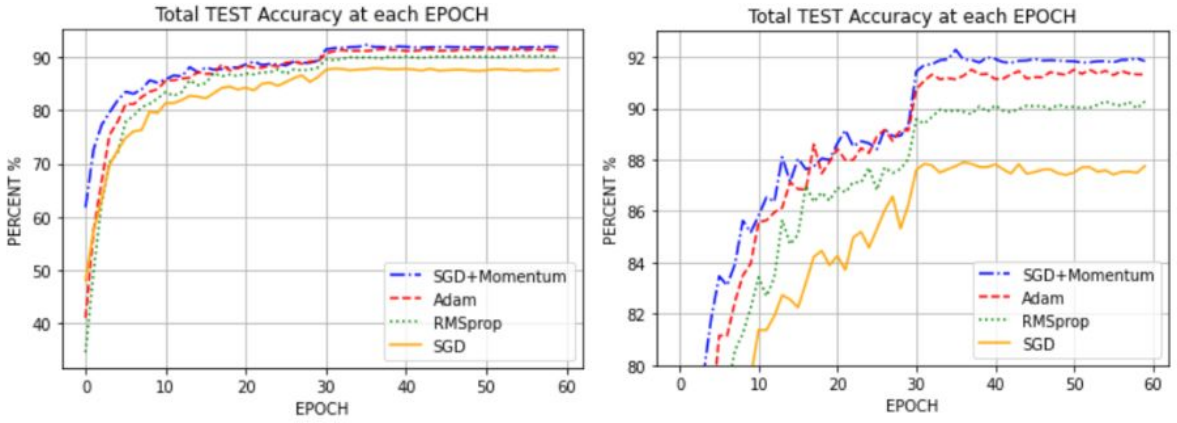


그림 15. Optimizer에 따른 TEST 정확도 (Swish)

그림 15은 Swish 활성화 함수를 이용한 Resnet18 모델에서 Optimizer에 따른 각각의 TEST 정확도 그래프 표다. 좌, 우 그래프 표는 같은 표이며, 우측 그래프는 좌측의 그래프 표를 확대한 것이다.

■ RELU 활성화 함수

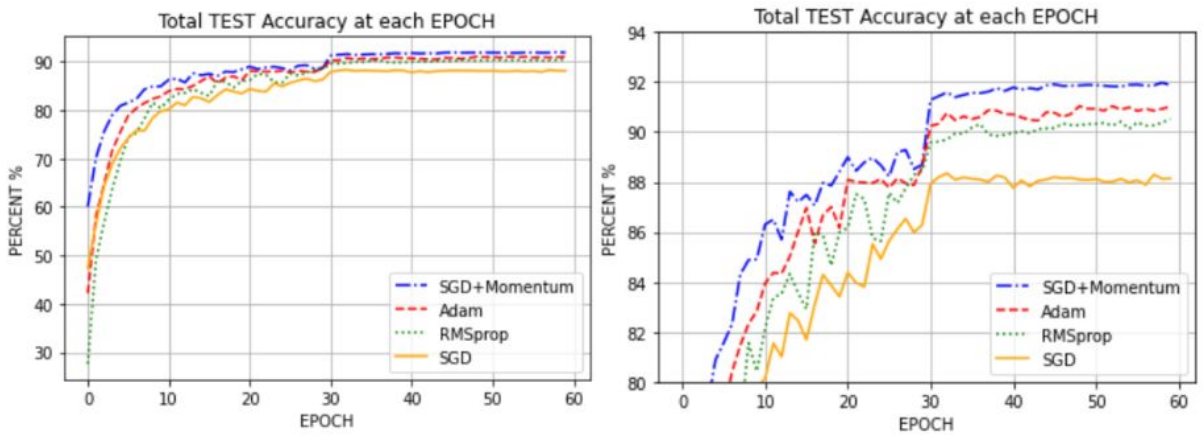


그림 16. Optimizer에 따른 TEST 정확도 (Relu)

그림 16은 RELU 활성화 함수를 이용한 Resnet18 모델에서 Optimizer에 따른 각각의 TEST 정확도 그래프 표이다. 좌, 우 그래프 표는 같은 표이며, 우측 그래프는 좌측의 그래프 표를 확대한 것이다. Swish 활성화 함수를 이용하였을 때의 Test 정확도 그림 15과 비슷한 결과를 보였다.

그림 15, 16 공통적으로, SGD+Momentum, Adam, RMSprop, SGD 순서대로 높은 TEST 정확도를 보인다. SGD(Stochastic Gradient Descent)에 Momentum을 추가하였더니 Adam보다 좋은 TEST 정확도를 보였다. 반면에 Momentum을 추가해주지 않은 SGD는 87~88%사이에 가장 낮은 정확도를 보였다.

RMSprop는 마지막 epoch때, 90~91% 정확도로 SGD보단 높았지만, 첫 회 학습 때 testset 정확도 30%으로 가장 낮게 시작을 하였다. 가장 좋은 정확도를 보인 SGD+Momentum은 첫 회 학습 때부터 60%이상의 testset 정확도로 시작을 하였다.

3-3. 정규화 (Regularization).

Cifar10은 Binary dataset과 마찬가지로 weight decay와 dropout을 이용하여 Regularization을 해보았다. Relu 활성화 함수와 Momentum을 설정한 SGD Optimization을 이용하였다.

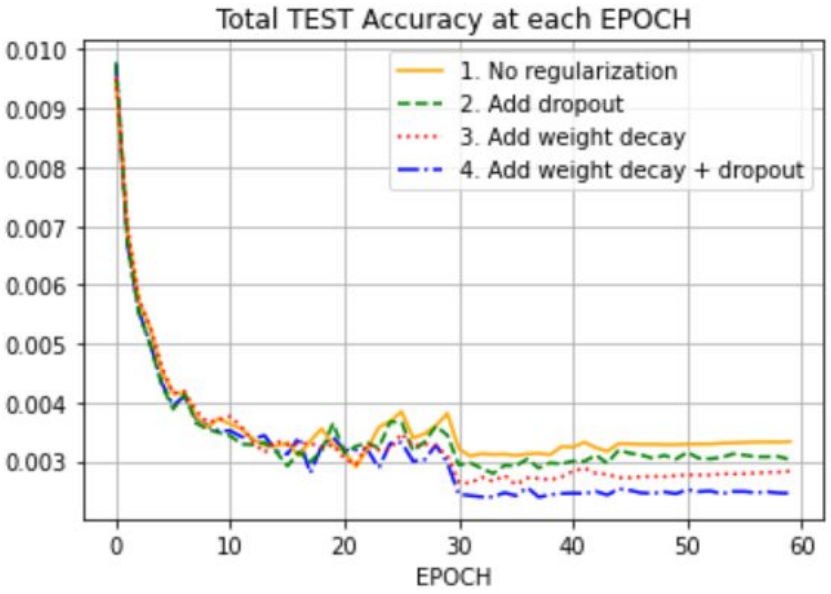


그림 17. Regularization에 따른 Test Loss 그래프

그림 17은 weight decay 와 dropout에 따른 Test Loss 그래프 표이다. 1번 그래프는 weight decay나 dropout을 설정하지 않았을 때 Test Loss를 측정한 것이다. 2번 그래프는 weight decay는 설정하지 않고, dropout만 설정하였을 때 그래프다. 3번 그래프는 dropout은 설정하지 않고 weight decay를 설정하였을 때 그래프이며, 4번 그래프는 weight decay와 dropout 모두 설정 하였을 때 그래프다.

epoch 초반에는 비슷한 Loss를 보이다가, 20 epoch 이후부터 1번, 2번, 3번, 4번 순서대로 Test Loss가 낮았다. 즉, weight decay나 dropout을 설정 해주었을 때, Test의 loss를 줄일 수 있었고, 둘다 설정 하였을 때 가장 낮은 test loss로 수렴하였다.

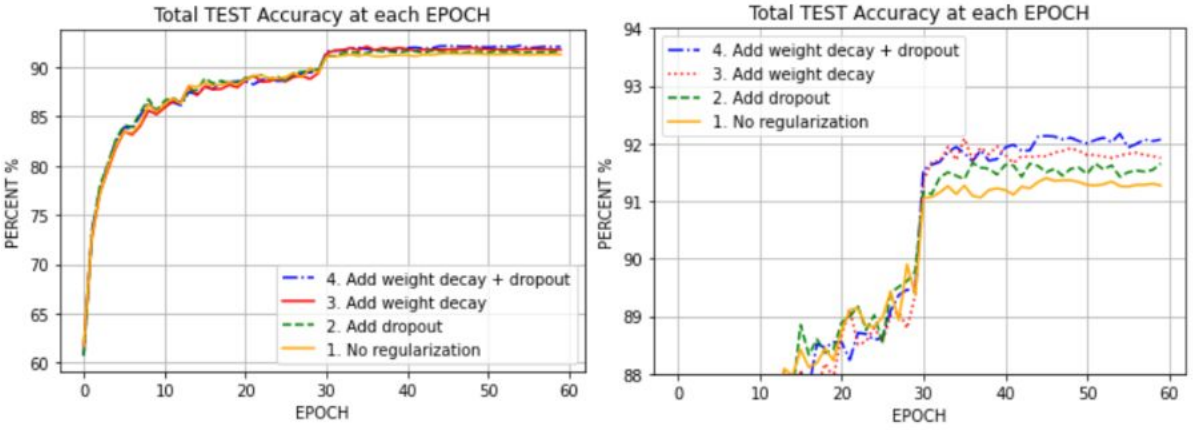


그림 18. Regularization에 따른 Test 정확도 그래프

그림 18은 그림 17의 실험에서 Test 정확도 그래프를 나타낸 것이며, 좌우 그래프표는 동일한 그래프이다. 그림 15에서 Test Loss가 낮은 순서대로 Test 정확도는 높았다.

3번 그래프 Weight decay은 정확도는 91.54%, 2번 그래프 Dropout은 정확도 91.75%를 보인다. 3번 그래프가 2번 그래프보다 Test 정확도와 Loss에 좋은 성능을 나타냈다. 4번 그래프는 마지막 epoch기준 정확도 92.17%을 보였으며, 1번 그래프는 91.27% 를 보였다. weight decay와 dropout 모두 해주었을 때, 결과적으로 두 가지 모두 해주지 않은 1번 그래프보다 Test 정확도를 약 1%정도 올릴 수 있었다.

3-4. Dataset Omission.

Cifar10의 데이터 셋은 <trainset 50000개, testset 10000개>로 이루어져있다. trainset, testset의 기존의 이미지 갯수를 <40000개, 8000개>, <30000개, 6000개>, <20000개, 4000개>, <10000개, 2000개>씩 랜덤으로 1/5씩 줄여가며 학습해보았다. 각각의 실험은 Dataset의 개수 빼곤 모두 동일한 조건에서 학습하였다. 공통적으로 Relu 활성화 함수와 SGD+Momentum Optimizer을 이용하였다.

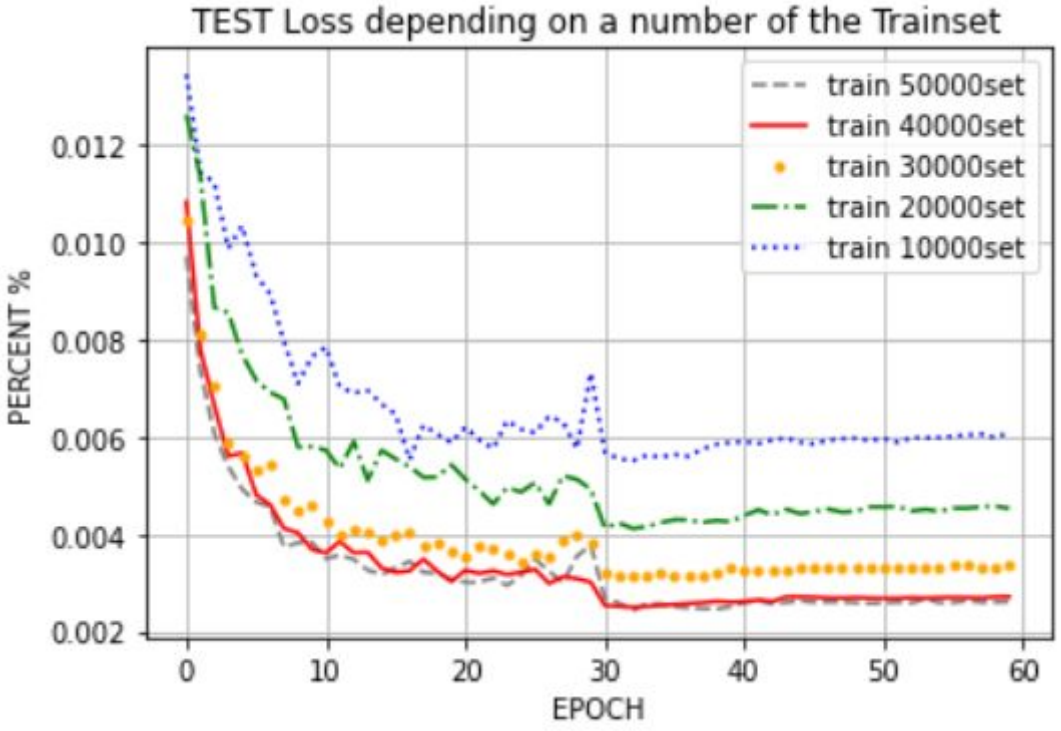


그림 19. trainset 이미지 갯수에 따른 test 이미지 정확도 그래프

trainset의 개수가 풍부해질수록, testset의 대한 Loss는 감소하였다. 그림 19 마지막 epoch 기준, trainset 40000개일 때 testloss는 0.0027147이며, trainset 50000개의 testloss는 0.0261567으로 그래프 상에선 미묘하지만 조금 감소하였다.
0000000000000

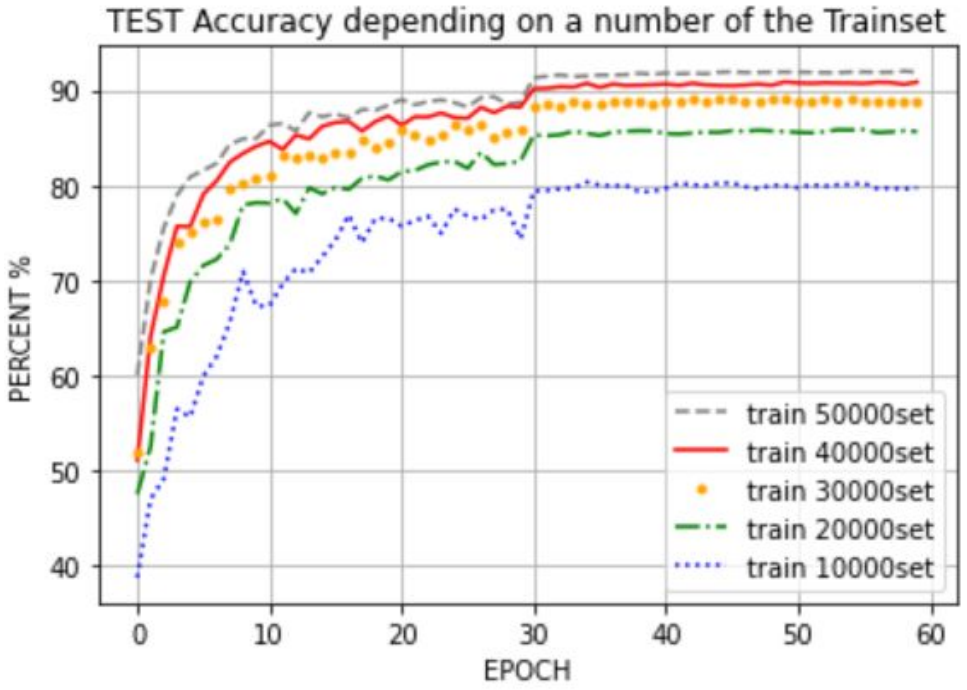


그림 20. 데이터 갯수에 따른 Test 정확도 그래프

trainset의 개수가 적어질수록, testset의 대한 정확도는 낮아진다. 그림 18에서 trainset의 이미지양이 10000개의 경우, 정확도는 80%으로 저조한 결과를 보인다. 그림 9에서 Binary dataset은 Trainset 1027개 만으로도 90%가 넘는 Test 정확도를 보였는데, 이는 Binary dataset은 class가 2가지이며, 흑백의 이미지이기 때문에 적은 데이터 만으로도 90%가 넘는 정확도를 나타냈다. Cifar10에서 testset의 이미지에 대하여 90%가 넘는 Classification 정확도를 보이기 위해선 적어도 trainset 40000개의 이미지가 필요하다.

마지막 epoch 기준, trainset 40000개일 때 test 정확도는 90.8%이며, trainset 50000개의 test 정확도는 92.0%이다. 그림 17의 Test Loss 그래프에선, trainset 이 40000개 일 때와 50000개 일때의 Loss 차이가 그래프에서 미묘하였지만, 그림 18의 정확도 그래프에선 확연하였다.

4. SEU(Swish Exponential Unit) 활성화 함수 제안

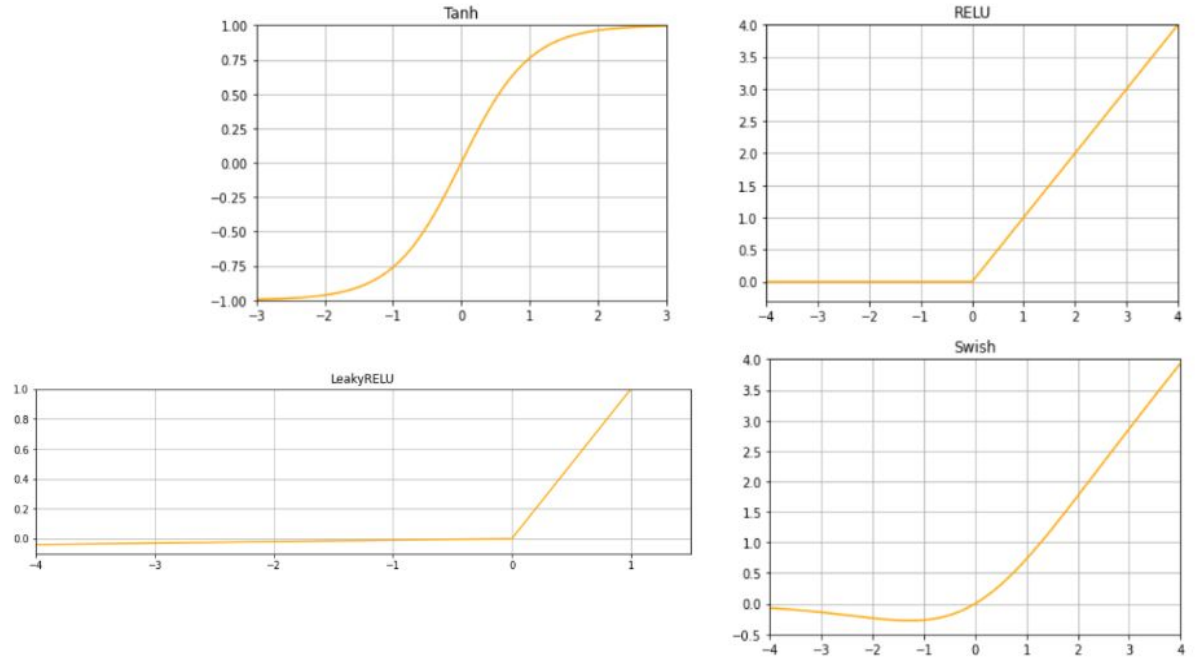


그림 21. Tanh, Relu, LeakyRelu, Swish 활성화 함수의 그래프

그림 19에서 Tanh 함수 그래프를 살펴보면 미분값(기울기)이 최대가 되는 지점이 x가 0일 때 1이다. x가 0이 아니라면 미분값은 1보다 작으며 0에 수렴한다. Weight들은 Loss를 최소화 시키는 방향으로 역전파(Backpropagation)을 수행한다. 예를 들어, 첫번째 Weight으로 미분을 계산하려면 체인 규칙을 이용하여 다음과 같이 Backpropagate한다.

$$\frac{\delta error}{\delta w1} = \frac{\delta error}{\delta output} \times \frac{\delta output}{\delta hidden_n} \times \frac{\delta hidden_n}{\delta hidden_{n-1}} \cdots \times \frac{\delta hidden_1}{\delta w1}$$

신경망이 깊으면 깊을 수록 활성화 함수를 미분한 값을 곱해지는 수가 많아지기 때문에 기울기 값이 사라지는 문제 (Gradient vanishing Problem)을 겪게 된다. 이러한 문제는 초기 Weight에서 각각의 Parameter들에 값에 대해 큰 변화가 일어나도 output에 대해서 큰 영향을 주지 못하게 됨을 의미한다. 따라서, tanh 활성화 함수를 층이 깊은 모델에 사용할 경우, Gradient vanishing 문제가 발생하여 네트워크를

효과적으로 학습시키지 못하고, Error rate이 미쳐 낮아지지 못한채 수렴해버리는 문제가 발생하는 것이다.

반면, 그림 9에서 Relu 그래프는 x가 양수라면 미분값은 1이며, 음수라면 미분값이 0이기 때문에, 기울기 값이 사라지는 문제 (Gradient vanishing Problem)에 대하여 tanh함수보다 개선시킬 수 있다. 하지만 입력 값이 0 또는 음수일 때, gradient가 0이 되므로 학습을 하지 못하는 문제(Dying relu Problem[7])를 겪게 될 수 있다.

Swish 활성화 함수는 그래프가 음수에서 기울기가 0이 아니기 때문에, 앞서 말한 Dying Relu Problem을 개선시킬수 있다. 또, 그림 7에서 swish는 relu보다 그래프가 smooth한 형태를 띈다. 이러한 형태의 그래프는 초기값이나 learning rate에 대하여 덜 민감한 반응을 보인다는 장점이 있다.. 하지만 Swish함수는 입력 값이 아주 작은 음수라면, 여전히 0으로 수렴하여 문제가 될 수 있다.

Swish 그래프가 작은 음수 쪽에선 0으로 수렴하는 그래프 모양을 개선 시키고자 Swish, Exponential 함수와 함께 Unit으로 나누어 Binary와 Cifar10 dataset에 Activation 함수로 이용하여 보았다.

$$SEU(x) = \begin{cases} \alpha(e^x - 1) & \text{if } x < 0 \\ \frac{x}{1+e^{-x}} & \text{if } 0 \leq x \end{cases}$$

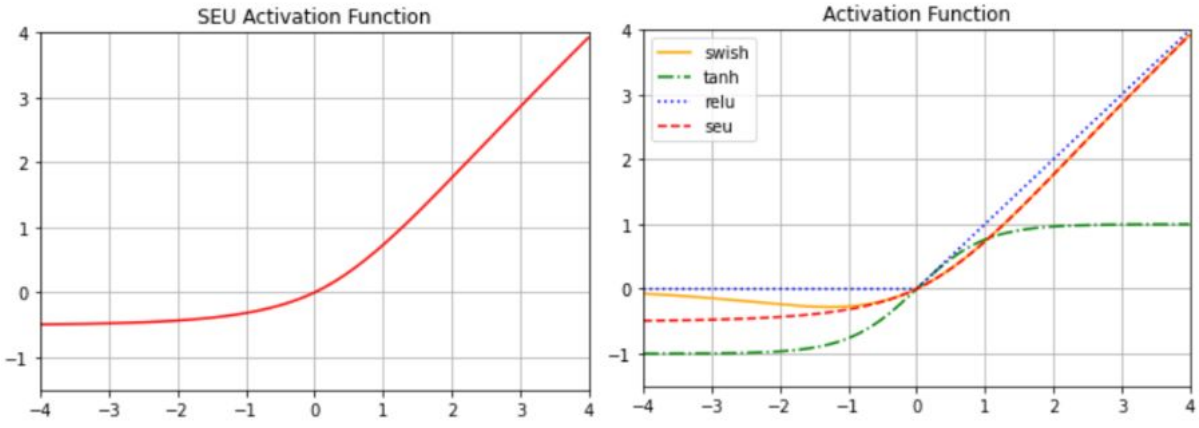
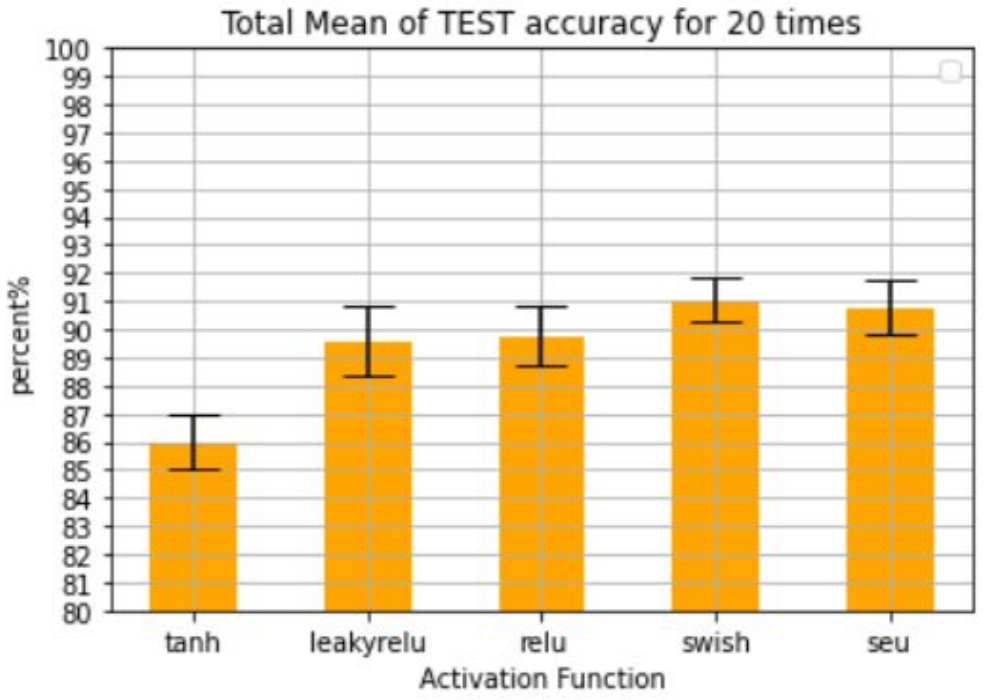


그림 22. SEU Activation 함수 그래프와 식

SEU 함수는 양수와 음수에서 상한 하한이 없고, ReLU 함수와 같이 음수에서 기울기가 0이 아니다. 따라서 Gradient Vanishing Problem에 강하다. 그리고 음수가 작아질수록 0에 수렴하지 않으며, Exponential을 이용하였기 때문에 Swish함수와

마찬가지로 Smooth한 형태를 띈다. 그렇기 때문에 초기값과 Learning rate에 대하여 덜 민감하다.

■ SEU on Binary Dataset

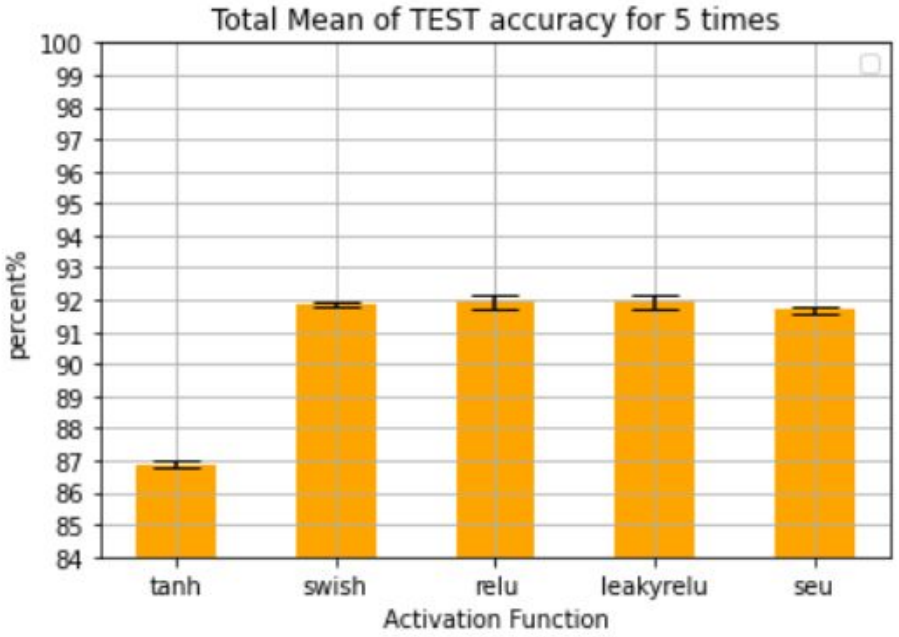


	TANH	LEAKY RELU	RELU	SWISH	SEU
평균 정확도	85.94%	89.75%	89.76%	91.17%	90.74%
오차	0.98	1.26	1.03	0.77	0.96

그림 23. TEST 정확도의 평균과 표준편차 비교 (Binary Dataset)

그림 23은 Binary Dataset에서 그림 7, 8과 동일한 조건의 실험을 SEU 활성화 함수를 적용하여 20번을 실험한 뒤, TEST 정확도의 평균과 표준편차를 구하여 그림 7,8와 합친 것이다. SEU 활성화 함수를 이용하여 Classification 했을 때, 평균 정확도 90.74%로 Swish 함수와 비슷한 성능을 보여주었다. Relu, Leaky Relu랑 비교하였을 땐, SEU 활성화 함수를 사용하였을 때 비교적 높은 TEST 정확도를 보여주었다.

■ SEU on Cifar10



	TANH	SWISH	RELU	LEAKY RELU	SEU
평균 정확도	86.88%	91.86%	91.92%	91.96%	91.69%
오차	0.12	0.06	0.21	0.24	0.11

그림 24. TEST 정확도의 평균과 표준편차 비교 (Cifar10)

그림 24는 Cifar10에 대하여 그림 14와 동일한 조건의 실험을 SEU 활성화 함수를 적용하여 5번을 실험한 뒤, TEST 정확도의 평균과 표준편차를 구하여 그림 14와 합친 것이다. SEU 활성화 함수를 이용하여 Classification 했을 때, 평균 정확도 91.69%로 Swish, RELU, Leaky RELU 함수와 비슷한 성능을 보여주었다. 표준 오차는 Rectified Exponential Linear Unit 함수들 보다 낮은 오차를 보냈으며, 이는 Swish와 마찬가지로 초기값이나 Learining rate에 덜 민감하여 Fluctuation이 적음을 보여준다.

Conclusion

이미지 분류를 위하여 Deep Learning Neural Network에서 Activation, Optimization, Regularization의 변화를 주어 학습해보았다. 활성화 함수는 Binary dataset에서 swish함수가 가장 좋은 학습을 보였고, Cifar10에선 Relu, Leaky relu, Swish가 큰 차이가 없었다. 그러므로 간단하고 적은 데이터셋 Binary dataset에는 Swish 활성화 함수를, Cifar10에서 Swish보다 6배 빠른 Relu 활성화 함수를 쓰는 것이 종합적으로

효율적인 학습이 된다. 이를 기반으로 활성화 함수의 특징을 연관하여 살펴보며 단점과 장점을 종합해본 뒤, 장점은 살리고 단점을 보완한 SEU함수를 제안하여 기존의 데이터셋에 학습하여 결과를 비교해 보았다.

Optimization은 Binary dataset에선 데이터가 적기 때문에 SGD, Adam, RMSprop가 큰차이를 보이지 않았지만, Cifar10에선 Momentum을 한 SGD가 가장 좋은 성능을 보였다. 정규화는 Weight decay와 dropout을 한 학습이 binary dataset과 cifar10 모두 test Loss와 정확도에서 효과적인 학습을 보여주었다. 또, 이미지 분류에서 Test의 정확도 90프로 이상의 어느정도 만족할만한 성능을 위해선 충분한 량의 data 양이 필요하였다.

Deap Learning Neural Network에서 이론적으로는 좋은 학습일 것이라 예측되는 framework는 실제로는 학습을 해보았을 때 아닌 경우도 있었다. 딥러닝에서 어떤 네트워크 프레임워크가 가장 좋다고 정하여 말할 수 없는 이유는, 어떠한 조건의 데이터셋을 가지고 있느냐에 따라 다를 수 있기 때문이다. 즉, 효율적인 성능의 학습을 위해선 Activation, Optimization, Regularization을 학습에 대상이 되는 Dataset의 특징에 따라 선택하는 것이 필요하다.

Reference

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. Computer Science, Mathematic ArXiv @2015

[2]horses-or-humans-dataset.
Available: http://www.laurencemoroney.com/horses-or-humans-dataset/

[3] Prajit Ramachandran* , Barret Zoph, Quoc V. Le .SEARCHING FOR ACTIVATION FUNCTIONS . Computer Science, Mathematic ArXiv @2017

[4] Vinod Nair, Geoffrey E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. Department of Computer Science, University of Toronto, Toronto, ON M5S 2G4, Canada @2010

[5] Abien Fred M. Agarap. Deep Learning using Rectified Linear Units (ReLU) Computer Science, Mathematic ArXiv @2018

[6] Nitish Srivastava ,Geoffrey Hinton, Alex Krizhevsky ,Ilya Sutskever, Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15 @2014 1929-1958

[7] Lu Lu, Yeonjong Shin, Yanhui Su, George Em Karniadakis. Dying ReLU and Initialization: Theory and Numerical Examples. Computer Science, Mathematics ArXiv @2019