ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
j.inst.Korean.electr.electron.eng.Vol.24,No.3,766~776,September 2020
논문번호 20-03-12
http://dx.doi.org/10.7471/ikeee.2020.24.3.766
90

기계학습 옵티마이저 성능 평가

Performance Evaluation of Machine Learning Optimizers

주 기 훈*, 박 치 현*,**, 임 현 승*,**★

Gihun Joo*, Chihyun Park*,**, Hyeonseung Im*,**★

Abstract

Recently, as interest in machine learning (ML) has increased and research using ML has become active, it is becoming more important to find an optimal hyperparameter combination for various ML models. In this paper, among various hyperparameters, we focused on ML optimizers, and measured and compared the performance of major optimizers using various datasets. In particular, we compared the performance of nine optimizers ranging from SGD, which is the most basic, to Momentum, NAG, AdaGrad, RMSProp, AdaDelta, Adam, AdaMax, and Nadam, using the MNIST, CIFAR-10, IRIS, TITANIC, and Boston Housing Price datasets. Experimental results showed that when Adam or Nadam was used, the loss of various ML models decreased most rapidly and their F1 score was also increased. Meanwhile, AdaMax showed a lot of instability during training and AdaDelta showed slower convergence speed and lower performance than other optimizers.

요 약

최근 기계학습에 대한 관심이 높아지고 연구가 활성화됨에 따라 다양한 기계학습 모델에서 최적의 하이퍼 파라미터 조합을 찾는 것이 중요해지고 있다. 본 논문에서는 다양한 하이퍼 파라미터 중에서 옵티마이저에 중점을 두고, 다양한 데이터에서 주요 옵티마이저들의 성능을 측정하고 비교하였다. 특히, 가장 기본이 되는 SGD부터 Momentum, NAG, AdaGrad, RMSProp, AdaDelta, Adam, AdaMax, Nadam까지 총 9개의 옵티마이저의 성능을 MNIST, CIFAR-10, IRIS, TITANIC, Boston Housing Price 데이터를 이용하여 비교하였다. 실험 결과, 전체적으로 Adam과 Nadam을 사용하였을 때 기계학습 모델의 손실 함숫값이 가장 빠르게 감소하는 것을 확인할 수 있었으며, F1 score 또한 높아짐을 확인할 수 있었다. 한편, AdaMax는 학습 중에 불안정한 모습을 많이 보여주었으며, AdaDelta는 다른 옵티마이저들에 비하여 수렴 속도가 느리며 성능이 낮은 것을 확인할 수 있었다.

Key words : Machine Learning, Deep Learning, Optimizer, Performance Evaluation, Adam, Nadam

* Dept. of Medical Bigdata Convergence, Kangwon National University
** Dept. of Computer Science and Engineering, Kangwon National University
★ Corresponding author
E-mail : hsim@kangwon.ac.kr, Tel : +82-33-250-8447

※ Acknowledgment
This study was supported by 2017 Research Grant from Kangwon National University (No. 520170088).
This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1F1A1063272).
Manuscript received Aug. 31, 2020; revised Sep. 13, 2020; accepted Sep. 21, 2020.

Ⅰ. 서론

최근 기계학습을 활용하는 연구가 폭발적으로 증가하고 있으며, 이에 따라 기계학습에 대한 사람들의 관심도 증가하고 있다. 기계학습을 편리하게 수행할 수 있게 해주는 주요 프레임워크로는 TensorFlow,[1] Keras,[2] Pytorch[3] 등이 있으며, 이러한 프레임워크가 많이 발전함에 따라 사용자는 원하는 기계학습 모델을 점점 더 쉽게 만들 수 있게 되었다. 하지만 기계학습 모델을 구성할 때, 모델의 하이퍼 파라미터(hyperparameter)들을 적절히 최적화하지 못하면, 높은 성능은 보장할 수 없다. 이러한 기계학습 하이퍼 파라미터에는 학습을 얼마나 진행할지 결정하는 에포크(epoch), 모델 내부의 뉴런의 수, 뉴런을 얼마나 활성화시킬지 결정하는 드롭아웃(dropout), 모델 학습 시 손실 함숫값을 최소화하는 방향으로 모델의 가중치 값을 갱신하기 위한 전략을 결정하는 옵티마이저(optimizer) 등이 있다. 이러한 하이퍼 파라미터들의 모든 가능한 조합을 고려하여 최적의 조합을 찾는 것은 많은 계산량이 요구되며 매우 어려운 문제이다. 따라서 보통은 연구자의 경험이나 직관에 의존하여 몇 가지 조합을 시도해보는 매뉴얼 검색(manual search)이나 보다 정량적인 방식인 그리드 검색(grid search) 또는 베이지안 최적화(Bayesian optimization)와 같은 기법이 사용된다.

여러 하이퍼 파라미터 중에서 옵티마이저는 학습과 가장 직접적으로 관련된 파라미터로, 모델의 손실 값을 최소화할 수 있도록 가중치를 조정하는 데 가장 중요한 역할을 담당한다. 동일한 구조와 동일한 하이퍼 파라미터를 갖는 기계학습 모델도 사용된 옵티마이저에 따라 학습이 잘 되거나 잘 안 될 수도 있어서 성능 차이가 크게 발생할 수 있다. 따라서 데이터와 사용하는 기계학습 모델에 적합한 옵티마이저를 찾는 것은 높은 성능을 얻기 위해서 매우 중요하다.

본 논문에서는 기계학습의 대표적인 프레임워크인 Keras를 이용하여 다양한 옵티마이저에 대해 블랙박스 검사(Black-box testing)를 수행하고, 각 옵티마이저가 어떤 모델에서 얼마나 좋은 성능을 보여주는지 실험하였다. 블랙박스 검사는 입력값과 출력값 기반 테스팅 기법으로 기능 테스트라고도 부르며, 소프트웨어의 내부 로직을 알지 못하더라도 소프트웨어의 동작을 검사할 수 있는 방법이다. 실험은 기본적인 심층 신경망(Deep Neural Network, DNN) 모델을 대상으로 하였으며, 온라인에 공개되어 자주 활용되는 MNIST, CIFAR-10, IRIS, TITANIC, Boston Housing Price 데이터를 이용하여 각 옵티마이저의 성능을 측정하였다.

본 논문에서는 가장 기초적인 최적화 알고리즘인 Gradient Descent(경사 하강법)의 학습이 오래 걸리는 단점을 개선한 Stochastic Gradient Descent (확률적 경사 하강법, SGD), SGD의 지그재그로 움직이며 최적화하는 문제를 개선한 Momentum [1], Momentum 방식을 기초로 하지만 기울기를 계산하는 방식을 개선한 Nesterov Accelerated Gradient (NAG) [2], 학습률을 감소시키는 기법을 적용한 AdaGrad [3], AdaGrad의 단점을 개선한 RMSProp [4], RMSProp과 비슷하게 보완한 AdaDelta [5], RMSProp과 Momentum 방식을 조합한 Adam [6], Adam의 extension으로 제안된 AdaMax [6], NAG와 Adam 옵티마이저의 개념을 합친 Nadam [7]의 성능을 비교 분석하였다. 실험 결과, 전체적으로 Adam과 Nadam을 사용하였을 때 기계학습 모델의 손실 함숫값이 가장 빠르게 감소하는 것을 확인할 수 있었으며, F1 score 또한 높아짐을 확인할 수 있었다. 한편, AdaMax는 학습 중에 불안정한 모습을 많이 보여주었으며, AdaDelta는 다른 옵티마이저들에 비하여 수렴 속도가 느리며 성능이 낮은 것을 확인할 수 있었다.

Ⅱ. 본론

1. 옵티마이저

본 절에서는 옵티마이저의 개념과 각 옵티마이저의 특징에 대해 설명한다. 기계학습 모델인 신경망을 이용하여 학습을 진행하는 과정은 손실 함수의 값을 가능한 낮출 수 있는 매개변수 값을 찾는 과정이다. 이러한 과정을 매개변수 최적화(Optimization)라고 한다. 심층 신경망이 널리 활용되면서 뉴런 내의 매개변수가 매우 많아짐에 따라 한 번에 최적의 솔루션을 구하는 것은 사실상 불가능해졌다. 따라서 일반적으로 매개변수의 최적값을 찾기 위해,

1) https://www.tensorflow.org/
2) https://keras.io/
3) https://pytorch.org/

매개변수의 기울기인 미분을 이용하여 기울기의 반대 방향으로 매개변수 값을 업데이트하는 과정을 반복하여 최적값과 근사한 값을 구하게 된다. 가장 기초적인 옵티마이저인 Gradient Descent(경사 하강법, GD)는 네트워크의 출력값과 실제값의 차이인 손실을 최소화하기 위해 기울기를 이용하지만, 최적값을 찾기 위해 업데이트 과정을 한 번 반복할 때마다 모든 데이터를 사용하기 때문에 비효율적이다. 이러한 단점을 개선하기 위해 Stochastic Gradient Descent(확률적 경사 하강법, SGD) 기법이 제안되었다.

가. Stochastic Gradient Descent (SGD)

SGD는 앞서 설명한 GD의 전체 데이터를 이용하는 단점을 개선한 대표적인 경사 하강법으로 다음과 같은 공식을 이용한다.

$$W \leftarrow W - \eta \frac{\partial L}{\partial W}$$

여기에서 W는 갱신할 가중치 매개변수이며 η는 학습률, 학습률 옆의 식은 W에 대한 손실 함수의 기울기를 의미한다. GD와는 다르게 미니 배치를 이용하여 데이터 세트를 조금씩 훑어보고 가중치를 업데이트한다. 앞으로 살펴볼 여러 옵티마이저들 중에서 가장 단순하고 구현하기도 쉽지만, 비효율적인 경우가 많다. 대표적인 단점으로는 비등방성(anisotropy) 함수와 같이 방향에 따라 기울기가 달라지는 함수에 대해서는 탐색 경로가 비효율적이라는 점이 있다. 또한, 학습률이 낮으면 최솟값으로 곧장 최적화하지 못하고 지그재그로 이동하게 되면서 지역 최솟값(local minimum)에 빠질 수 있으며, 학습률이 높으면 최적화 자체를 실패할 수 있다는 단점이 존재한다.

나. Momentum

Momentum은 [1] SGD가 지그재그로 움직이며 최적화되는 문제를 해결하기 위해 제안된 방법으로, 경사 하강법을 통해 이동하는 과정 중에 일종의 관성을 이용하는 방법이다. 비유컨대, 공이 그릇의 곡면을 따라 구르듯이 움직이는 것을 목표로 하되, 이동 방향과는 별개로 과거에 이동했던 방향을 기억하여 그 방향으로 일정량을 추가로 이동하는 기법으로 다음과 같은 공식을 이용한다.

$$v \leftarrow \alpha v - \eta \frac{\partial L}{\partial W},\ W \leftarrow W + v$$

여기에서 v는 속도이고, a는 모멘텀 계수로 이전 업데이트 값을 어느 정도의 비율로 적용할 것인지에 해당하며, 보통 0.9의 값을 가진다. 자주 이동하는 방향에 대해 관성이 생겨 지그재그로 이동하는 정도가 SGD에 비해 덜하다는 장점이 있다. 관성에 의해 상대적으로 SGD에 비해 지역 최솟값에서 빠져나올 가능성은 높지만 기존의 네트워크 파라미터 이외에 과거에 이동했던 방식을 저장해야 하므로 메모리가 더 많이 필요하다는 단점이 있다.

다. Nesterov Accelerated Gradient (NAG)

NAG [2] 방식은 Momentum 방식을 기초로 하고 있지만, 실제 기울기를 계산하는 방식이 다른 방법이다. 간단히 말해, NAG는 미리 경사를 확인할 수 있게 하고 경사에 맞춰 속도를 조절하는 아이디어에 기반한다. Momentum 방식에서는 이동 벡터를 계산할 때, 현재 위치에서의 기울기와 모멘텀 스텝을 독립적으로 계산하고 합치는데, NAG 방식에서는 모멘텀 스텝을 먼저 이동했다고 가정한 후 그 위치에서의 기울기를 구해 기울기 스텝을 이동하게 된다. NAG 역시 Momentum처럼 모멘텀 계수가 0인 경우 SGD와 동일한 알고리즘이 된다. 멈추어야 할 시점에서도 관성에 의해 더 멀리 나아갈 수 있는 Momentum과 달리 NAG는 관성을 이용해 이동하되 학습이 어느 정도 진행된 상황에서는 관성에 의해 최적값을 지나치는 문제를 방지할 수 있다는 장점이 있다.

라. Adaptive Gradient (AdaGrad)

실제 신경망 학습에서는 학습률 값이 학습에 큰 영향을 준다는 문제가 존재한다. 학습률 값이 너무 작은 경우 학습 시간이 오래 걸리며, 너무 큰 경우 학습이 제대로 이루어지지 않는다. 이러한 문제는 학습률 감소 기법(learning rate decay)으로 해결할 수 있으며, 가장 간단한 방법은 매개변수 전체의 학습률 값을 일괄적으로 낮추는 것이다. 이를 발전시킨 AdaGrad [3] 방식은 다음과 같이 각 매개변수에 맞춤형 값들을 만들어 준다.

$$h \leftarrow h + \frac{\partial L}{\partial W} \odot \frac{\partial L}{\partial W},\ W \leftarrow W - \eta \frac{1}{\sqrt{h}} \frac{\partial L}{\partial W}$$

여기에서 h는 기존 기울기 값을 제곱하여 계속 더해주는 역할이다. 매개변수의 원소 중에서 크게 갱신된 원소는 학습률이 낮아진다는 의미이며, 매개변수 원소마다 다르게 적용된다. 하지만 학습을 반복하다 보면 기울기 제곱의 값이 점점 줄어들게 되고 갱신 강도가 약해지다가 결국 어느 순간 0이 되어 학습이 더 이상 진행되지 않을 수도 있다는 단점이 존재한다.

마. RMSProp

RMSProp은 [4] 앞서 설명한 AdaGrad의 단점을 해결하기 위해 G. Hinton이 제안한 옵티마이저이다. AdaGrad의 식에서 기울기의 제곱 값을 더해나가면서 구한 h를 합이 아닌 지수 평균으로 바꾸었다. AdaGrad처럼 h가 무한정 커지지는 않으며 변수 간 최근 변화량의 상대적 크기 차이는 유지가 가능하다는 장점이 존재한다.

$$G = \gamma G + (1-\gamma)(\nabla_\theta J(\theta_t))^2$$
$$\theta = \theta - \frac{\eta}{\sqrt{G+\epsilon}} \cdot \nabla_\theta J(\theta_t)$$

AdaGrad에서 G는 현재 시간까지의 변화량의 합으로 정의되기 때문에 시간이 지날수록 증가하게 되고 학습률은 감소하지만, RMSProp에서는 이전의 변화량과 현재의 변화량의 지수 평균으로 정의되기 때문에 학습률이 급격하게 감소하는 현상을 방지할 수 있다.

바. AdaDelta

AdaDelta [5] 역시 RMSProp처럼 AdaGrad의 학습이 진행됨에 따라 학습률이 매우 작아져 학습이 진행되지 않는 단점을 보완하기 위해 제안된 옵티마이저이다. G를 구할 때 합을 이용하지 않고 지수 평균을 이용한다. 다음과 같이 스텝 크기를 단순히 η로 사용하는 대신 스텝 크기의 변화 값의 제곱을 가지고 지수 평균값을 계산한다는 특징이 있다.

$$G = \gamma G + (1-\gamma)(\nabla_\theta J(\theta_t))^2$$
$$\triangle_\theta = \frac{\sqrt{s+\epsilon}}{\sqrt{G+\epsilon}} \cdot \nabla_\theta J(\theta_t)$$
$$\theta = \theta - \triangle_\theta$$
$$s = \gamma s + (1-\gamma)\triangle_\theta^2$$

사. Adaptive Moment Estimation(Adam)

Adam은 [6] 학습률을 변경하는 RMSProp과 최적화에 의한 갱신 경로를 변경하는 Momentum을 조합하여 만들어진 옵티마이저로 Momentum처럼 지금까지 계산해온 기울기의 지수 평균을 저장한다. 또한, 다음과 같이 RMSProp처럼 기울기의 제곱 값의 지수 평균을 저장한다.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta J(\theta)$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla_\theta J(\theta))^2$$

하지만 Adam에서는 m과 v가 처음에 0으로 초기화되어 있어, 학습 초반부에 m_t, v_t가 0에 가깝게 편향되어 있을 것으로 판단하여 다음과 같이 이를 보정해준다.

$$\hat{m} = \frac{m_t}{1-\beta_1^t},\ \hat{v_t} = \frac{v_t}{1-\beta_2^t},\ \theta = \theta - \frac{\eta}{\sqrt{\hat{v_t}+\epsilon}}$$

여기서 β_1은 0.9, β_2는 0.999, ϵ은 10^{-8} 정도로 사용한다.

아. AdaMax

AdaMax는 [6] 앞서 살펴본 Adam의 확장으로 제안된 알고리즘으로 Adam의 경우 L_2-norm을 기반으로 학습률을 조절하는 데 반해 AdaMax의 경우 L_p-norm을 이용한다. 단 p가 매우 클 경우 L_p-norm은 극단적인 값을 갖는 등의 매우 불안정한 모습을 보여준다. Adam 논문의 저자는 p가 무한대로 갈 때 매우 간단하고 안정적인 알고리즘이 만들어진다고 언급한다[6].

$$v_t = \beta_2^p v_{t-1} + (1-\beta_2^p)(\nabla_\theta J(\theta))^p$$

위 식은 Adam의 학습 속도를 조절하는 v_t를 L_2-norm에서 L_p-norm으로 확장한 것이다. 위 식을 정리하면 다음과 같다.

$$v_t = \beta_2^p v_{ij}^{t-1} + (1-\beta_2^p)\left|\frac{\partial L}{\partial w_{ij}^{(t)}}\right|^p$$
$$= (1-\beta_2^p)\sum_{i=1}^{t}\beta_2^{p(t-i)}\left|\frac{\partial L}{\partial w_{ij}^{(t)}}\right|^p$$

위 식에서 p가 무한대로 갈 때의 식 $(v_t)^{1/p}$는 아래와 같이 정의된다.

$$v_t = \max\left(\beta_2 v_{t-1}, \left|\frac{\partial L}{\partial w_{ij}^{(t)}}\right|\right)\}$$

자. Nadam

Nadam은 [7] 앞서 설명한 NAG와 Adam의 개념을 합친 옵티마이저로 NAG에서 사용했던 방식대로 현재 위치에서 다음 위치로 이동할 기울기와 모멘텀 값을 구하는 것이 아닌 모멘텀 값으로 이동한 뒤에 기울기를 구하는 방식이다. Adam에서는 m(첫 번째 모멘텀)과 v(두 번째 모멘텀)를 사용하는데, m은 기울기 값을 좀 더 빠르게 계산할 수 있도록 돕는 역할이고, v는 데이터의 분포가 희소한 곳에서 영향력을 극대화함으로써 희소한 영역을 벗어날 수 있게 도움을 주는 역할을 한다. Nadam은 여기서 NAG의 철학을 이용하여, 현재 위치가 아닌 모멘텀 방향으로 이동한 뒤의 위치에서 기울기 값을 구하며, 이 때문에 1차 편미분(야코비안, Jacobian)을 계산할 때, Adam과는 다르게 $\theta - \gamma m_{t-1}$ 위치 상에서 계산한다. Nadam은 Adam과 NAG의 장점을 합쳤기 때문에 Adam보다 좀 더 빠르고 정확하게 전역 최솟값(global minimum)을 찾아낼 수 있다는 장점이 있다.

2. 관련 연구

기계학습의 활용 가능성이 매우 높아지면서 다양한 옵티마이저에 대한 연구도 활발히 진행되고 있다[1]-[7]. 또한, 다양한 옵티마이저의 성능을 비교 평가하는 연구도 많이 진행되고 있다[8]-[10]. [8]에서는 본 연구와 유사하게 대표적 공개 데이터 세트인 Fashion MNIST, CIFAR-10, ImageNet, LM1B을 이용하여, SGD, Momentum, NAG, RMSProp, Adam, Nadam 총 6개의 옵티마이저를 비교 평가하였다. [8]에 따르면 각 옵티마이저는 서로 관련성을 갖고 있으며, 결론적으로 Adam이 Momentum이나 SGD에 비해 더 빠르고 대체로 Nadam이 좋은 성능과 속도를 보여주었다.

[9]에서는 남강댐 유역의 강우 관측 지점의 일강우량을 바탕으로 남강댐으로 들어오는 일 유입량을 인공신경망을 이용하여 예측하는 알고리즘을 개발하였는데, Adam과 Nadam이 가장 적절한 옵티마이저이며 SGD는 성능상 한계가 존재함을 확인하였다. 특히, Adam은 Nadam에 비해 느리게 수렴하나 성능은 조금 더 높은 것을 확인할 수 있었다.

[10]에서는 사람의 중기 염색체로 이루어진 디지털 이미지에서 염색체 객체를 검출할 때 필요한 경사 하강 최적화 기법의 성능을 비교하였다. 구체적으로 Faster R-CNN 네트워크를 VGG16 [11]과 ResNet50 [12]으로 구성하고 AdaGrad, RMSProp, AdaDelta, AdaMax에 이용하여 성능 비교를 진행하였다. VGG16에서는 AdaMax가 제일 좋은 성능을 보여주었으며, ResNet50에서는 AdaDelta가 가장 좋은 성능을 보여주었다.

본 논문에서는 Keras 라이브러리에서 제공하는 대표적인 9개의 옵티마이저에 대해, 각 옵티마이저가 5개의 주요 공개 데이터에서 어느 정도의 성능을 보여주는지, 어느 옵티마이저가 가장 빠르게 학습하는지 등을 비교 평가하였다.

3. 데이터 전처리 및 모델

본 절에서는 실험에 사용한 5개의 공개 데이터인 MNIST, CIFAR-10, IRIS, TITANIC, Boston Housing Price 및 전처리 과정을 소개한다. 각 데이터의 특징은 표 1에 정리되어 있다.

Table 1. Characteristics of the datasets.
표 1. 데이터 세트의 특성

Data	Type	# of Features	# of Training Data
MNIST	Image	28*28	60,000
CIFAR-10	Image	32*32	50,000
IRIS	Numerical vector	4	120
TITANIC	Numerical vector	8	712
Boston Housing Price	Numerical vector	13	404

가. MNIST

MNIST는 기계학습 공부를 시작하면 가장 처음 만나볼 수 있는 데이터 세트이다. Convolutional Neural Network(CNN)를 공부할 때 가장 쉽게 접할 수 있는 컴퓨터 비전 데이터로 NIST의 원래 데이터 세트의 샘플을 재혼합하여 만들어진 데이터이다. 해당 데이터는 학습에 적합하도록 이미 전처리가 이루어져 있다. 데이터는 0부터 9까지의 손글씨로 적힌 데이터로 gray scale의 28*28 크기의 픽셀 이미지와 각 이미지에 해당되는 라벨을 포함하고 있다.

55,000개의 학습 데이터, 5,000개의 검증 데이터, 10,000개의 테스트 데이터로 구성되어 있다. 학습 데이터는 각 숫자별 평균 6,000개의 데이터를 포함하고 있으며 테스트 데이터는 평균 1,000개의 데이터를 포함하고 있다. 본 실험을 위해 추가로 전처리를 진행하였다. 픽셀 데이터의 경우 0부터 255 범위의 픽셀값을 가지고 있으므로 정규화(normalization)를 진행하여 0.0에서 1.0의 범위를 갖도록 변환하였다. 해당 데이터에 CNN과 DNN 모델을 모두 적용하기 위해 두 개의 데이터 세트를 생성하였다. CNN을 위한 데이터는 28*28의 크기를 채널을 포함하여 28*28*1의 크기로 변경해주었으며, DNN을 위한 데이터는 28*28의 크기를 728 길이의 벡터로 변경하여 구성하였다.

최종적으로 변환된 데이터를 이용하여 CNN과 DNN에 대한 실험을 진행하였는데, CNN의 경우 2개의 Convolution 계층과 2개의 Max Pooling 계층을 가지며 마지막에 두 개의 Dense 계층을 갖는 기본적인 CNN 모델을 구성하였다. DNN의 경우 입력층 - Dense 계층 - 출력층으로 간단히 구성하였다. 각 실험은 50 에포크를 반복하였으며 손실 값의 변화를 이용하여 비교를 진행하였다.

나. CIFAR-10

CIFAR-10은 기계학습에서 MNIST 다음으로 유명한 컴퓨터 비전 데이터 세트이다. 기계학습에서 흔히 사용되는 벤치마크 데이터이기도 하다. CIFAR-10 데이터 세트는 픽셀 이미지를 비행기, 자동차, 새, 고양이, 사슴, 개, 개구리, 말, 배, 트럭의 총 10개의 라벨로 분류하는 것이 목표이다. 각 이미지는 RGB 값으로 구성된 32*32 크기의 픽셀 이미지이며, 10개의 클래스로 라벨링이 이루어져 있다. 각 카테고리별로 6,000개의 이미지가 존재하며 50,000개의 학습 데이터, 10,000개의 테스트 데이터로 구성되어 있다. 앞서 설명한 MNIST의 경우 정규화를 적용하였으나 CIFAR-10의 경우 원 데이터에서 평균을 빼고 표준편차로 나누어주는 표준화(standardization)를 적용하였다. 실험은 Convolution 계층 3개와 Max Pooling 계층 3개, 2개의 Dense 계층으로 구성된 CNN을 이용하여 50 에포크씩 실험을 진행하였다.

다. IRIS

IRIS는 통계학자 R. Fisher가 수집한 붓꽃에 관한 데이터로 꽃받침(Sepal)과 꽃잎(Petal)의 길이와 폭을 가지고 세 개의 붓꽃 종류인 Versicolor, Virginica, Setosa를 분류하는 것이 목표이다. 각 종별로 50개의 샘플로 구성되어 있어 총 150개의 데이터가 존재한다. IRIS는 그림 1과 같은 분포를 보이는데 Setosa의 경우 다른 데이터와 뚜렷하게 구분 가능하나 Versicolor와 Verginica는 구분하기 어렵다. 데이터 분석 결과 크게 전처리를 할 부분이 없어 전처리 없이 실험을 진행하였다. 실험에는 입력층 - Dense 계층 - 출력층으로 구성된 간단한 DNN 모델을 이용하였으며, 이전 실험과 동일하게 50 에포크를 반복하며 손실 값의 변화에 대한 비교를 진행하였다.

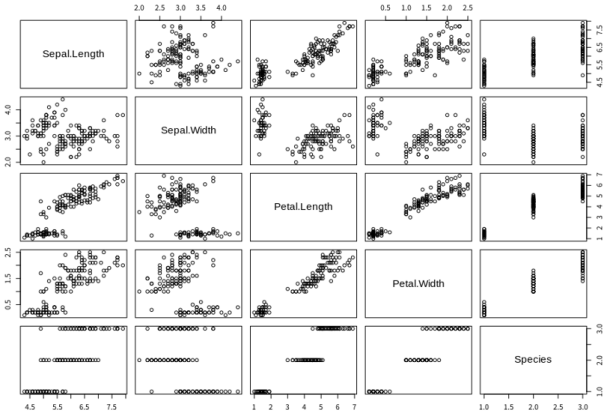

Fig. 1. IRIS Data Distribution.
그림 1. IRIS 데이터 분포도

라. TITANIC

TITANIC은 타이타닉호에 탑승했던 사람들의 정보를 바탕으로 생존자를 예측하기 위한 데이터로 기계학습 경진대회인 캐글(Kaggle)에 공개되어 있다. 예측해야 하는 Survived를 포함한 총 12개의 자질(feature)이 존재하며 전처리를 통해 각 개체별로 고윳값을 갖는 Name과 Passengerid를 제외하였다. 891개의 학습 데이터와 418개의 테스트 데이터가 존재하나 테스트 데이터에 대한 survived는 공개되지 않았기 때문에 본 실험에서는 학습 데이터를 다시 학습 데이터와 테스트 데이터로 나누어서 사용하였다. 데이터 분석 과정에서 결측치가 많이 검출됐는데, 대표적으로 Age에서 177개, Cabin에서 687개, Embarked에서 2개의 결측치가 나왔다. Age의 경우 Name 정보에서 Mr, Miss, Mrs, Other 정보를 추출하여 이를 바탕으로 각 그룹별

평균값으로 결측치를 채웠으며, 연령대가 0세부터 80세까지 다양하게 존재하므로 5개의 카테고리로 나누어 진행하였다. Embarked의 경우 1등급이 S 도시에서 탑승한 사람들이 대부분이므로 S로 가정하여 결측치를 채워 넣었으며, Cabin의 경우 알파벳과 숫자가 합쳐진 형태의 데이터이기 때문에 알파벳만을 이용하여 밀접한 관계가 있는 클래스별로 Cabin의 중간값으로 결측값을 해결하여 실험을 진행하였다. 실험에는 입력층 - Dense 계층 - 드롭아웃 - Dense 계층 - 드롭아웃 - 출력층으로 이전에 사용한 모델보다는 좀 더 깊은 DNN 모델을 사용하였다.

마. Boston Housing Price

본 데이터는 파이썬의 유명한 라이브러리 중 하나인 Scikit learn[4]에서 제공하는 회귀 분석용 예제 데이터이다. 1987년에 발표된 미국 보스턴 지역의 주택 가격을 포함하고 있으며 주택의 여러 가지 요건들과 주택의 가격정보가 포함되어 있으며 주택의 가격에 영향을 미치는 요인들을 분석하는 것이 주목적인 데이터 세트이다. 전체 506개의 레코드가 13개의 입력변수와 1개의 종속 변수로 구성되어 있다. 해당 데이터는 이미 전처리가 되어있는 데이터이기 때문에 따로 전처리를 진행하지 않았으며, 상관계수 분석을 진행하였으나 서로 밀접한 관계를 가지는 자질들이 없었기 때문에 원본 데이터를 가지고 실험을 진행하였다. 앞선 데이터들과 다르게 본 데이터는 가격을 예측하는 회귀 문제이므로 손실 함수로는 평균 제곱 오차(Mean Squared Error, MSE)를, 성능 척도로는 평균 절대 오차(Mean Absolute Error, MAE)를 사용하였다. 실험에는 입력층 - Dense 계층 - Dense 계층 - 출력층으로 구성된 간단한 DNN 모델을 이용하였다.

4. 실험

가. 실험 환경

실험은 Intel I9-9900KF 3.6GHz CPU, 64GB RAM, RTX 2080Ti VGA 사양의 Ubuntu 18.24 워크스테이션에서 진행하였다. 각 데이터별로 동일한 실험 환경에서 손실 값이 얼마나 빨리 수렴하는지에 대하여 비교를 진행하였으며, 각 옵티마이저의 F1 score를 계산하여 비교하였다. 각 옵티마이저의 파라미터는 모두 기본값을 이용하였다. 본 실험의 목표는 최적화된 예측 모델을 찾는 것이 아니라 주어진 데이터에 대해 각 옵티마이저 별로 얼마나 학습이 잘 진행되는지를 비교하는 것이기 때문에, 벤치마크에 사용되는 각 데이터 세트를 학습 : 테스트 용도로 8 : 2의 비율로 나누어 실험을 진행하였다.

나. 실험 결과

(1) MNIST

MNIST의 경우 DNN과 CNN을 이용하여 각각 실험을 진행하였다. 각 옵티마이저 별로 50 에포크를 진행하여 손실 값의 상태변화를 분석하였다.

그림 2는 MNIST 데이터에 대한 DNN 실험 결과이다. Nadam, Adam, RMSProp, AdaMax, NAG, Momentum이 좋은 성능을 보여주고 있으며, 그 중 Adam, Nadam, RMSProp이 가장 빠르게 손실 값을 수렴시키는 것을 확인할 수 있다. 반면, AdaDelta가 성능이 가장 좋지 않은 것을 확인할 수 있다. F1 score는 AdaDelta의 경우 0.91로 가장 낮은 성능을 보여주었으며, 앞서 언급한 우수한 성능을 보인 6개의 옵티마이저는 모두 약 0.98로 좋은 성능을 보여주었다.

그림 3은 MNIST 데이터에 대한 CNN 실험 결과이다. 앞선 DNN 실험과 유사하게 Nadam과 Adam이 손실 값을 빠르게 수렴시키는 것을 확인할 수 있었다. 이 실험에서도 AdaDelta가 가장 느리게 수렴하는 것을 확인할 수 있다. F1 score를 비교해보았을 때 가장 느리게 수렴하는 AdaDelta의 경우 0.95로 가장 낮았으며, 그 외의 옵티마이저들은 대부분 0.99로 주어진 데이터를 잘 분류하는 것을 확인할 수 있었다. 즉, MNIST와 같이 간단한 데이터 세트에 대해서는 대부분의 옵티마이저들이 좋은 성능을 보여주었으며, 특히 Nadam과 Adam이 가장 빠르게 수렴하는 것을 확인할 수 있었다.

(2) CIFAR-10

CIFAR-10 데이터의 경우 앞서 설명하였듯이 간단한 CNN 모델을 이용하여 실험을 진행하였다. 그림 4와 같이 모든 옵티마이저가 전체적으로 거의 비슷하게 손실 값을 수렴시키는 모습을 보여주었으나, 그래도 Nadam이 가장 좋은 성능을 보여주었다. 여러 번의 실험 결과 AdaGrad와 AdaMax의

4) https://scikit-learn.org/stable/index.html

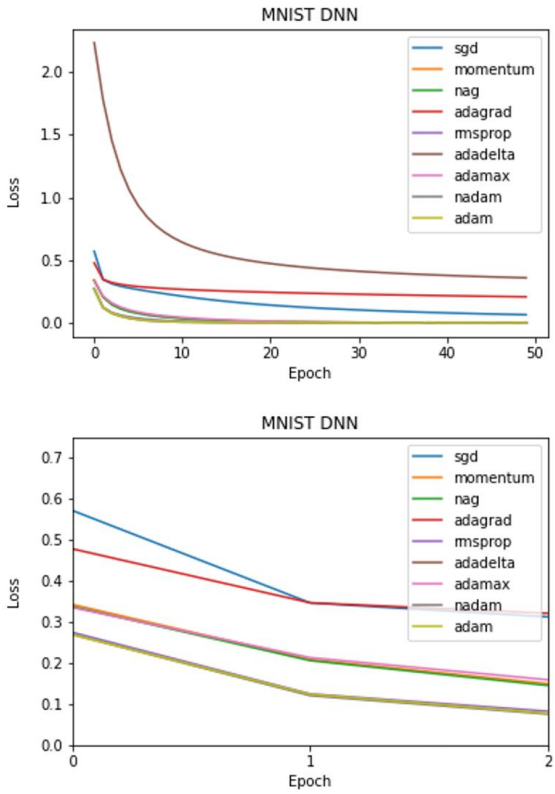

Fig. 2. MNIST DNN experiment result.
그림 2. MNIST DNN 실험 결과

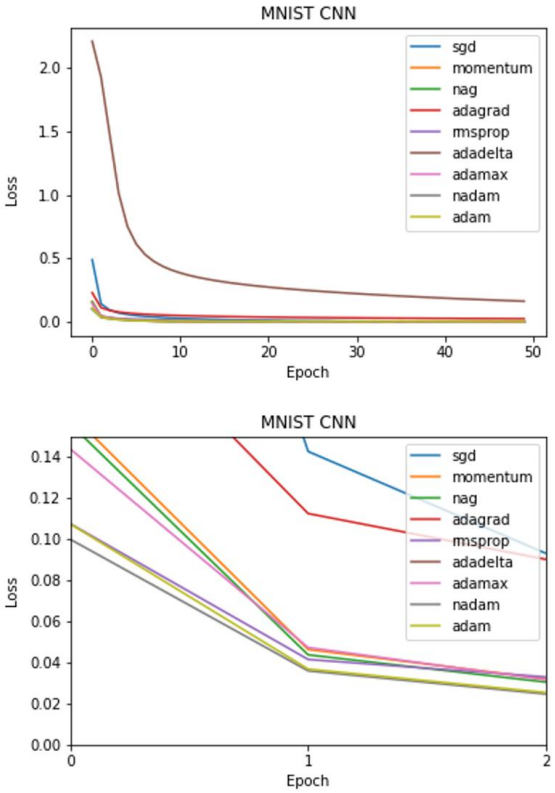

Fig. 3. MNIST CNN experiment result.
그림 3. MNIST CNN 실험 결과

경우 빠르게 수렴하는 경우도 있었으며, Momentum의 경우 다른 옵티마이저보다 수렴하는 속도가 느린 것을 확인할 수 있었다. 모든 모델이 비슷하게 수렴하였듯이 F1 score도 평균 0.77로 비슷한 성능을 보여주었다.

(3) IRIS

IRIS 데이터는 데이터 수가 적고 분류가 다른 모델에 비해 쉽다. 본 실험에서는 기본적인 DNN 모델을 사용하였는데, 그림 5에서 확인할 수 있듯이, 이전 실험에서는 다른 모델에 비해 성능이 좋았던 Nadam과 Adam보다 Momentum과 NAG가 더 좋은 성능을 보여주고 있다. AdaDelta의 경우 수렴 및 발산하지 못하며 학습이 진행되지 않았고, AdaMax의 경우 초기화에 따라 발산하는 경우도 있는 것을 실험을 통해 알 수 있었다. 하지만 F1 score를 확인해보았을 때 수렴하는 속도가 빠른 Momentum의 경우 평균적으로 0.87이었으며 NAG, Nadam, Adam의 경우 1로 정확하게 분류하는 것을 확인할 수 있었다. 즉, 수렴 속도는 Momentum도 빠르지만, 성능은 기대에 못 미치는 것을 확인할 수 있었다.

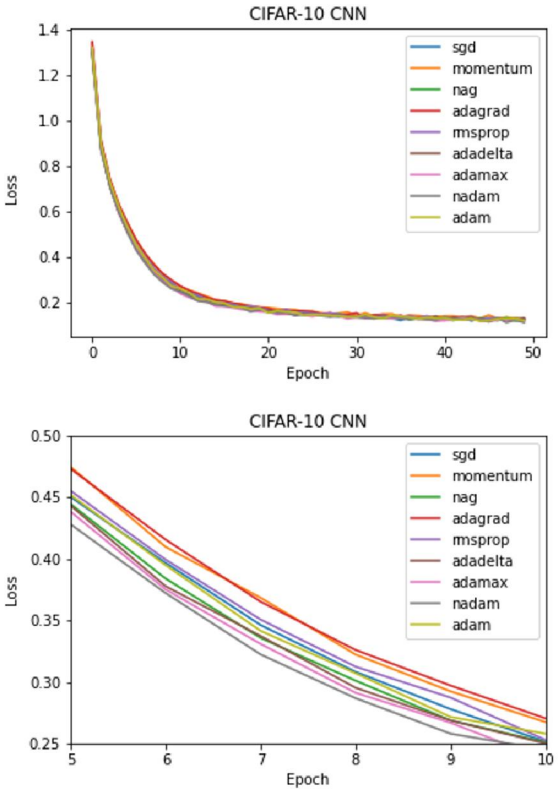

Fig. 4. CIFAR-10 CNN experiment result.
그림 4. CIFAR-10 CNN 실험 결과

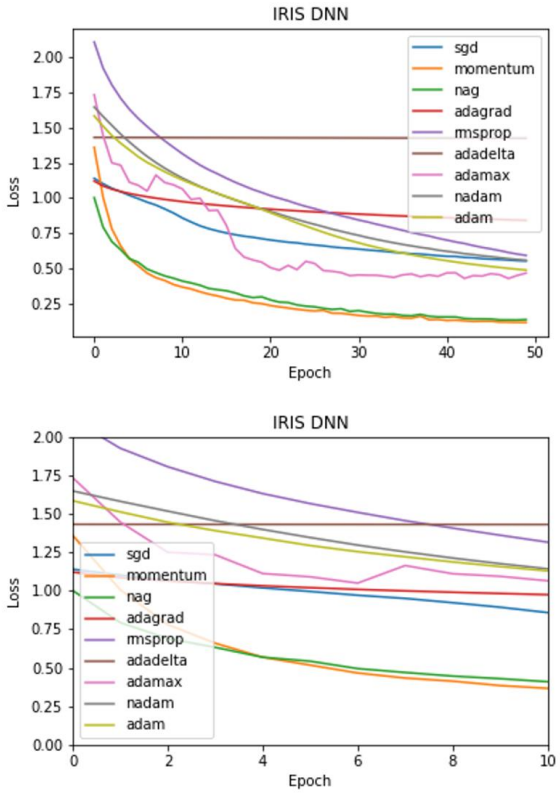

Fig. 5. IRIS DNN experiment result.
그림 5. IRIS DNN 실험 결과

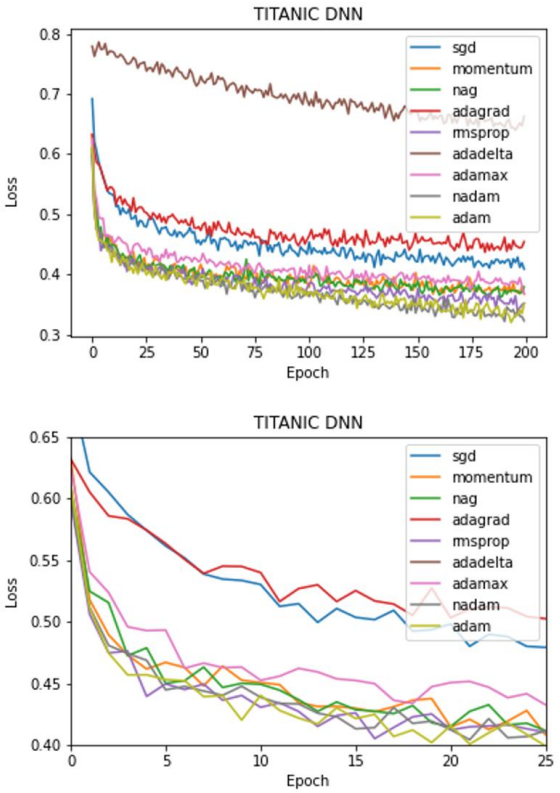

Fig. 6. TITANIC DNN experiment result.
그림 6. TITANIC DNN 실험 결과

AdaMax의 경우 수렴하는 것은 불안정하였으나 F1 score는 0.97로 높은 모습을 보여주었다.

(4) TITANIC

Titanic 데이터의 경우 결측치도 많고 기계학습에서 사용할 수 없는 속성들이 많아 탐색적 자료분석(Exploratory Data Analysis, EDA)이 중요하다. 본 실험에서는 앞서 설명하였듯이 매우 기본적인 EDA만 수행하였기 때문에 각 모델에 대한 성능이 그렇게 높지는 않았다. 또한, AdaDelta의 경우 학습이 느리게 진행되어, 에포크를 200으로 늘려 실험을 진행하였다. 그림 6에서 볼 수 있듯이 AdaDelta는 200 에포크를 진행하여도 다른 모델과 비슷한 정도까지 수렴하지 못하고 매우 천천히 수렴하는 것을 확인할 수 있었다. AdaMax의 경우 여러 번의 실험 과정에서 가끔 발산하는 문제가 발생하였다. 나머지 옵티마이저들은 비슷하게 수렴하였으나 Nadam과 Adam, RMSProp이 가장 빠르게 수렴하였다. F1 score의 경우 가장 느리게 수렴하는 AdaDelta가 0.38로 가장 낮았으며, Nadam과 Adam이 0.84로 가장 높았다.

(5) Boston Housing Price

Boston Housing Price 데이터를 이용한 실험에서는 주택의 가격을 예측하는 문제를 해결하였으며, 손실 함수로 MSE를 사용하고 성능 척도로 MAE를 사용한 것이 앞선 실험과 다른 점이다. MSE와 MAE는 다음과 같이 정의된다.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - x|$$

본 실험에서는 그림 7에서 볼 수 있듯이 SGD와 NAG가 가장 빠르게 수렴하였으며, 에포크가 지남에 따라 모두 비슷하게 손실 값이 줄어들었다. 하지만 AdaDelta의 경우 수렴하지 못하였으며 AdaGrad의 경우 조금씩 수렴하여 나중에는 비슷하게 수렴하였다. 한편, 여러 번의 실험 과정에서 Momentum과 NAG도 수렴하지 못하고 발산하는 경우가 있었다. 각 모델의 MAE 값을 비교한 결과 SGD가 약 2.4로 가장 작았으며 Nadam과 Adam도 약 2.6 정도로 좋은 성능을 보여주었다.

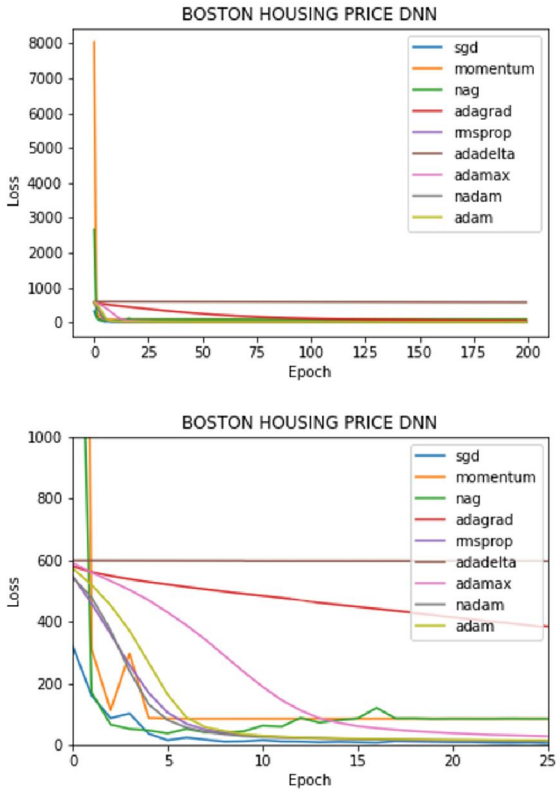

Fig. 7. Boston Housing Price DNN experiment result.
그림 7. Boston Housing Price DNN 실험 결과

III. 결론

본 논문에서는 온라인에 공개된 다양한 데이터들에 대해 간단한 기계학습 모델을 만들고, 9가지 옵티마이저에 대해 각각 얼마나 빠르게 손실 값이 수렴하는지 확인해보고 성능을 비교해보는 실험을 진행하였다. 실험 결과 이미지 기반 예측 모델에서는 모든 옵티마이저가 좋은 성능을 보였으나 Nadam이 가장 빠르게 수렴하는 것을 확인할 수 있었다. 또한, 테이블 데이터 기반 예측 모델에서도 Nadam이 좋은 결과를 보여주었으며, Adam과 RMSProp이 그 뒤를 이었다. 하지만 IRIS 데이터와 같이 자질 개수가 작고, 정확한 구분이 쉬운 데이터의 경우 Momentum과 NAG가 빠른 속도를 보여주었으며, 결과적으로는 NAG가 좋은 성능을 보여주었다. 마지막으로 가격을 예측하는 문제에서는 SGD가 가장 좋은 성능 및 수렴 속도를 보여주었으며, Nadam과 Adam의 경우 성능은 조금 낮았지만 빠르게 수렴하는 것을 확인할 수 있었다.

전체적인 실험 결과를 분석해볼 때, 최근 가장 많이 사용되고 있는 Adam 옵티마이저가 좋은 성능을 보여주며 빠르게 수렴하는 것을 확인할 수 있었다. 하지만 NAG와 Adam의 개념을 합친 방식인 Nadam이 조금 더 빠르게 수렴하며 이따금 더 좋은 성능을 보여주는 경우도 있었다. 한편, AdaMax, Momentum, NAG의 경우 데이터 종류 및 초기화 방식에 따라 발산하는 경우를 보여 사용 시 주의해야 할 것으로 생각된다. 마지막으로 AdaDelta의 경우 다양한 데이터에서 좋지 못한 성능을 보였다. AdaDelta는 안장점(saddle point)에서 문제가 발생하지 않는다는 장점이 있지만 2차 미분까지 하기 때문에 다른 옵티마이저에 비해 수렴 속도가 매우 느렸다.

본 연구에서는 옵티마이저의 파라미터 값으로 Keras에 정의된 기본값을 이용하였기 때문에 최적화에 따라 성능이 변할 수 있다는 한계가 존재한다. 후속 연구로 시퀀스 데이터에서 유용한 RNN과 같은 기계학습 모델에서는 어떠한 옵티마이저가 좋은 성능을 보여주는지 분석해볼 계획이며, 블랙박스 검사뿐만 아니라 화이트박스 검사(White-box testing)도 활용하여 좀 더 정확한 성능 평가를 수행해볼 계획이다.

References

[1] N. Qian, “On the momentum term in gradient descent learning algorithms,” *Neural networks,* vol.12, no.1, pp.145-151, 1999.
DOI: 10.1016/S0893-6080(98)00116-6

[2] Y. E. Nesterov, “A method for unconstrained convex minimization problem with the rate of convergence O(1/k^2),” *Dokl AN SSSR*, vol.269, no.3, pp. 543-547, 1983.

[3] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of machine learning research,* vol.12, pp.2121-2159, 2011.
DOI: 10.5555/1953048.2021068

[4] S. Ruder, “An overview of gradient descent optimization algorithms,” arXiv preprint arXiv: 1609.04747, 2016.

[5] M. D. Zeiler, “Adadelta: An adaptive learning rate method,” arXiv preprint arXiv:1212.5701, 2012.

[6] D. P. Kingma and J. Ba, “Adam: A method

for stochastic optimization," arXiv preprint arXiv: 1412.6980, 2014.
[7] T. Dozat, "Incorporating nesterov momentum into adam," in the 4th International Conference on Learning Representations (ICLR 2016) Workshop track, 2016.
[8] D. Choi et al., "On empirical comparisons of optimizers for deep learning," arXiv preprint arXiv: 1910.05446, 2019.
[9] M. Mahsa and T. Lee, "Comparison of Optimization Algorithms in Deep Learning-Based Neural Networks for Hydrological Forecasting: Case Study of Nam River Daily Runoff," *J. Korean Soc. Hazard Mitig.*, vol.18, no.6, pp.377-384, 2018. DOI: 10.9798/KOSHAM.2018.18.6.377
[10] W. Jung, B.-S. Lee, and J. Seo, "Performance Comparison of the Optimizers in a Faster R-CNN Model for Object Detection of Metaphase Chromosomes," *J. Korea Inst. Inf. Commun. Eng.*, vol.23, no.11, pp.1357-1363, 2019.
[11] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *in the 3rd International Conference on Learning Representations (ICLR 2015),* 2015.
[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition", *in 2016 IEEE Conference on Computer Vision and Pattern Recognition* (CVPR 2016), pp.770-778, 2016.

BIOGRAPHY

Gihun Joo (Member)

2018 : BS degree in Computer Science, Kangwon National University.
2020 : MS degree in Computer Science, Kangwon National University.
2020~ : PhD student in Department of Medical Bigdata Convergence, Kangwon National University

Chihyun Park (Member)

2007 : BS degree in Computer Engineering, Hongik University.
2009 : MS degree in Computer Science, Yonsei University.
2013 : PhD degree in Computer Science, Yonsei University.
2013~2015 : Research Staff Member, Samsung Advanced Institute of Technology, Samsung Electronics.
2015~2017 : Senior Researcher, Korea Institute of Science and Technology.
2018~2019 : Postdoctoral Research Fellow, Cleveland Clinic, USA.
2019~2019 : Postdoctoral Researcher, Yonsei University.
2020~ : Assistant Professor in Department of Computer Science and Engineering, Kangwon National University.

Hyeonseung Im (Member)

2006 : BS degree in Computer Science, Yonsei University.
2012 : PhD degree in Computer Science and Engineering, Pohang University of Science and Technology (POSTECH).
2012~2014 : Postdoc, Université Paris-Sud, France.
2014~2015 : Postdoc, Inria, France.
2015~ : Associate Professor in Department of Computer Science and Engineering, Kangwon National University.