**REPORT**

**Deeplearning Final Project**

|  |  |
| --- | --- |
| 과목 | 딥러닝 |
| 학과 | 빅데이터융합학과 |
| 학번 | 2020511019 |
| 이름 | 임 가 람 |

기존 강의를 토대로 MLP, ConvNet의 이해는 Low Level에서 직접 구현한 것으로 내부 동작에 대한 이해가 쌓였다. 하지만 실제 딥러닝 모델 연구/개발시 Hyper-parameter의 굉장히 많은 경우의 수에 의해 다양한 실험적 시도와 시간을 투자한다고 알고 있다. 따라서 본 프로젝트를 통해 실제 연구/개발시 큰 비중을 차지하는 실험적 모델 최적화 과정을 직접 경험하고, 여러 기법들을 모델에 적용해봄으로서 기법 간 성능에 미치는 영향력에 대한 직관을 키우고자 한다.

적용될 기법들은 교수님께서 강의 해주신 “최적화를 어떻게 할 것인지?”에 대한 고찰에서 강조된 기법들이다. 나아가 교수님께서 추가로 강조하신 Babysitting의 중요성을 상기시키며, 적은 양의 데이터에 대하여 특정 Hyper-parameter의 값을 자동적으로 순회하며 학습하도록 하고, 평가된 성능을 비교 분석 해보고자 한다. 따라서 본 프로젝트의 궁극적 목표는 실제 수업시간에 교수님께서 강의 해주신 여러가지 기법들의 이론(ex. Normalization, Dropout, Batch Normalization etc.)들과 여러 Hyper-parameter를 직접 모델에 실험적으로 적용해가는 과정을 반복하며, Accuracy, Training Loss Curve의 형태를 기준으로 모델 성능을 개선하는 기법에 대하여 비교 분석하는 것에 의의를 둔다.

기준 데이터 셋과 과제는 다음 <표-1>와 같다.

|  |  |
| --- | --- |
| **Dataset** | **Task** |
| CIFAR-10   * trainset : 50000 images * test : 10000 images | 10-label Clssification |

<표-1> 활용 데이터 셋 및 과제

비교의 용이함을 위하여 우선적으로 기준이 되는 네트워크 아키텍처와 일부 Hyper-parameter의 경우 지정하고, Accuracy와 Loss Curve를 결과로 정리하였다. 기준 네트워크 아키텍처와 Hyper-parameter는 다음 <표-2> 와 같다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Hyper-parameter** | | | | |
| **Batch size** | **Learning Rate** | **Epochs** | **Random Seed** | **optmizer** |
| 64 | 0.0002(2e-4) | 20 | 1004 | Adam |
| **Network(BasicCNN)** | | | | |
| <그림-1.1> BasicCNN | | | | |

<표-2> 기준 네트워크 및 Hyper-parameter

다음은 <표-2>를 기준으로 Weight Initialization, Batch Normalization, Normalization, Dropout 기법을 적용 후 Accuracy와 Loss Curve의 형태를 정리한 결과이다.

|  |  |  |
| --- | --- | --- |
| **Model** | **Technique** | **Accuracy [%]** |
| BasicCNN | - | 75.220 |
| **Network** | | |
| <그림-1.1> BasicCNN Network | | |
| **Loss Curve** | | |
| <그림-2.1> BasicCNN Loss Curve | | |

<표-3> BasicCNN 결과

|  |  |  |
| --- | --- | --- |
| **Model** | **Technique** | **Accuracy [%]** |
| DropoutCNN | Dropout | 78.035 |
| **Network** | | |
| <그림-1.2> DropoutCNN Network | | |
| **Loss Curve** | | |
| <그림-2.2> DropoutCNN Loss Curve | | |

<표-4> DropoutCNN 결과

|  |  |  |
| --- | --- | --- |
| **Model** | **Technique** | **Accuracy [%]** |
| BatchNormCNN | Batch-Normalization | 78.375 |
| **Network** | | |
| <그림-1.3> BatchNormCNN Network | | |
| **Loss Curve** | | |
| <그림-2.3> BatchNormCNN Curve | | |

<표-5> BatchNormCNN 결과

|  |  |  |
| --- | --- | --- |
| **Model** | **Technique** | **Accuracy [%]** |
| BatchNormDropoutCNN | Batch-Normalization + Dropout | 81.791 |
| **Network** | | |
| <그림-1.4> BatchNormDropoutCNN Network | | |
| **Loss Curve** | | |
| <그림-2.4> BatchNormDropoutCNN Curve | | |

<표-6> BatchNormDropoutCNN 결과

|  |  |  |
| --- | --- | --- |
| **Model** | **Technique** | **Accuracy [%]** |
| XavierBasicCNN | Xavier Weight Initialization | 74.770 |
| **Network** | | |
| <그림-1.5> XavierBasicCNN | | |
| **Loss Curve** | | |
| <그림-2.5> XavierBasicCNN Curve | | |

<표-7> XavierBasicCNN 결과

|  |  |  |
| --- | --- | --- |
| **Model** | **Technique** | **Accuracy [%]** |
| XavierBatchNormDropoutCNN | Xavier Initialization  Batch-Normalization  Dropout | 79.547 |
| **Network** | | |
| <그림-1.6> XavierBatchNormDropoutCNN | | |
| **Loss Curve** | | |
| <그림-2.6> XavierBatchNormDropoutCNN Curve | | |

<표-8> XavierBatchNormDropoutCNN 결과

전체 모델에 대한 종합 결과는 아래 표와 같다.

|  |  |
| --- | --- |
| **Model** | **Accuracy [%]** |
| BasicCNN | 75.220 |
| XavierBasicCNN | 75.731 |
| DropoutCNN | 78.035 |
| BatchNormCNN | 78.376 |
| XavierBatchNormDropoutCNN | 79.547 |
| BatchNormDropoutCNN | 81.791 |
| **Loss Curve** | |
| <그림-2.7> Models Loss Curve(Multi axes)    <그림-2.8> Models Loss Curve(Single axe) | |

<표-9> 모델 종합 결과

종합된 결과를 Accuracy를 기준으로 가장 많은 성능의 긍정적 기여를 한 기법은 BasicCNN 모델 대비 약 3%의 성능을 향상시킨 Dropout과 Batch Normalization으로 보인다. 나아가 Dropout과 Normalization 모두 사용한 BatchNormDropoutCNN 모델의 경우 BasicCNN 모델 대비 약 6%의 매우 높은 성능 향상을 가져다 주었다.

Dropout을 적용한 모델의 Loss Curve를 살펴보면, 타 모델의 Loss Curve 대비 완만히 떨어지는 형태에서 많은 반복 학습에서도 Overfitting을 피할 확률이 높을 것이라는 직관적인 해석이 가능하다고 본다. 나아가 20번의 Epoch에서의 Loss 값 또한 타 모델 대비 0으로의 수렴을 덜한 것으로 보인다. 따라서 더 높은 Epoch를 설정하여 loss를 0으로 더욱 수렴시킨다면, 더 높은 성능을 기대할 수 있을 것으로 보인다.

Weight Initialization의 경우 채택에 대한 항상 성능 향상을 보장해 주지는 못하는 경향을 보인다. 종합 결과에서 BasicCNN 모델 대비 XavierBasicCNN의 경우 0.5% 정도의 미세한 성능 향상이 있는 것으로 보인다. 하지만 XavierBatchNormDropoutCNN 모델과 BatchNormDropoutCNN 모델의 경우 오히려 Xaiver Weight Initialization을 적용한 결과가 성능을 약 2% 하락시키는 결과를 보여준다.

현재 Weight Initialization을 수행한 Weight는 Linear Layer이며, 현재 Activation Function으로 ReLU를 채택하였다. Xavier Weight Initialization의 경우 sigmoid, tanh와 같은 비선형함수에서 효과적인 결과를 보여주지만, ReLU 함수의 경우 0 보다 큰 지점에서는 선형함수이기 때문에 Weight Initialization 사용 시 출력 값이 0으로 수렴하게 되는 현상이 있다는 사실을 공개된 문서를 통해 확인하였다. 더불어 이를 일부 개선한Weight Initialization이 He Weight Initialization이라는 사실에서 He Weight Initialization을 통하여 추가 실험을 진행하였으며, 실험한 결과는 다음 표와 같다.

|  |  |  |
| --- | --- | --- |
| **Model** | **Technique** | **Accuracy [%]** |
| HeBasicCNN | He Initialization | 74.770 |
| **Network** | | |
| <그림-1.7> HeBasicCNN Network | | |
| **Loss Curve** | | |
| <그림-2.9> HeBasicCNN Curve | | |

<표-10> HeBasicCNN 결과

He Initialization을 적용한 결과가 어떠한 Weight Initialization을 적용하지 않은 BasicCNN보다 0.5% 더 낮은 성능을 보였다. 이를 통해 Weight Initialization은 드라마틱한 성능 개선에 기여하는 Hyper-parameter는 아닌 것으로 보이며, 성능 향상에 많은 기여를 하는 기법을 우선적으로 적용하여 일정 수준의 성능을 보장 받은 이후 추가적으로 여러 Weight Initialization을 적용하여 성능이 높은 기법을 채택하는 것이 Hyper-parameter 튜닝시 효율적이라는 것을 느끼는 바이다.

다음은 Loss Curve 기준으로 타 모델 대비 20 epoch에서 0으로의 수렴이 부족하며, 초기 epoch 구간에서 완만한 기울기를 지닌 XavierBatchNormDropoutCNN 모델을 활용하여 여러 Hyper-parameter 값의 다양한 조합을 넣어 자동적으로 학습하고 평가하는 형태로 구현하고, 각 값에 대한 결과를 정리하고 비교 분석하고자 한다. 다음 표는 활용할 Hyper-parameter와 값을 정의한다.

|  |  |  |
| --- | --- | --- |
| **Learning Rate** | **Batch Size** | **Epoch** |
| [2e-2, 2e-3, 2e-4] | [4, 16, 64] | [30, 40, 50] |

<표-11> 자동 학습/평가 Hyper-parameter 정의

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Learning Rate** | **Batch Size** | **Epoch** | **Accuracy** |
| 1 | 2e-2 | 4 | 30 | 10.000 |
| 2 | 40 | 10.000 |
| 3 | 50 | 10.080 |
| 4 | 16 | 30 | 10.000 |
| 5 | 40 | 68.490 |
| 6 | 50 | 67.530 |
| 7 | 64 | 30 | 68.239 |
| 8 | 40 | 9.996 |
| 9 | 50 | 10.016 |
| 10 | 2e-3 | 4 | 30 | 10.000 |
| 11 | 40 | 10.000 |
| 12 | 50 | 10.000 |
| 13 | 16 | 30 | 10.000 |
| 14 | 40 | 10.000 |
| 15 | 50 | 10.000 |
| 16 | 64 | 30 | 82.512 |
| 17 | 40 | 10.006 |
| 18 | 50 | 10.016 |
| 19 | 2e-4 | 4 | 30 | 83.680 |
| 20 | 40 | 83.590 |
| 21 | 50 | 83.870 |
| 22 | 16 | 30 | 82.690 |
| 23 | 40 | 84.080 |
| 24 | 50 | 84.630 |
| 25 | 64 | 30 | 80.228 |
| 26 | 40 | 81.350 |
| 27 | 50 | 82.312 |
| **Loss Curve** | | | | |
| <그림-2.10> Hyper-Parameter Tunning Loss Curve | | | | |
| **Loss Curve(Learning Rate = 0.02)** | | | | |
| <그림-2.11> Learning Rate 0.02 Loss Curve | | | | |
| **Loss Curve(Learning Rate = 0.002)** | | | | |
| <그림-2.12> Learning Rate 0.002 Loss Curve | | | | |
| **Loss Curve(Learning Rate = 0.0002)** | | | | |
| <그림-2.13> Learning Rate 0.0002 Loss Curve | | | | |

<표-12> Hyper-parameter Tuning 결과



<그림-2.14> Loss Curve with Learning Rate

이상적인 Loss Curve의 형태는 위 <그림-2.14>와 같이 초기에 급격히 떨어지다가 완만히 0에 가깝게 수렴하는 형태가 이상적이라고 하며, 이를 결정 짓는 가장 중요한 요소는 실험 종합 결과에서 알 수 있듯 Learning Rate이다. Learning Rate가 0.02, 0.002그래프에서는 대부분의 그래프가 초기에 급격히 Loss가 하강하다가 발산하기 시작하며, 너무 높은 Learning Rate에 의해 더 이상 학습이 진행되지 않고 일정한 Loss을 가지는 것을 볼 수 있다. 반면 Learning Rate가 0.0002에서는 기울기의 차이는 있으나, 대게 적절히 부드럽게 수렴하는 형태를 가지고 있다. 그 중 Batch Size가 16, 64인 경우에 더욱 자연스럽게 초기에 급격히 떨어지다가 완만히 수렴하는 형태를 가지고 있었다. <표-12>의 Accuracy 결과에서 알 수 있듯 적절한 Learning Rate 셋팅이 된다면, Epoch가 늘어남에 따라 Accuracy 또한 비례하여 좋아지는 경향을 보인다는 것을 알 수 있다. 종합적으로 가장 좋은 Accuracy를 보인 Hyper-parameter는 (Learning Rate=0.0002, Batch Size=16, Epoch=50) 인 경우 였으며, Cross Validation 기법으로 Validation Loss Curve로 체킹하며, Overfitting을 피하는 최대 Epoch 지점을 찾는다면, 더 높은 성능을 기대해볼 수 있을 것으로 보인다.

본 프로젝트를 진행하면서 느낀 점은 아래와 같이 크게 3가지를 느꼈다.

1. 모델 연구/개발 Making이 아닌 Search이다.

2. 높은 확률로 좋은 성능이 기대되는 기법을 우선적으로 적용해보지 않을 이유가 없다.

3. Hyper-parameter 튜닝시 거의 무한에 가까운 경우의 조합이 존재할 수 있으며, 중요한 것은 현재의 데이터 셋과 모델을 기준으로 무엇이 성능 개선에 큰 기여를 할 것인지를 직관적으로 알고 가설을 세운 후 우선적으로 적용하는 효율적인 실험 단계를 구성하는 것이 중요하다.

결과적으로 이번 프로젝트를 통하여 다양한 형태로 모델을 직접 구현할 수 있는 역량과 Loss Curve의 형태를 비교 분석하여 어떤 Hyper-parameter를 우선적으로 튜닝 해보는 것이 좋을지에 대한 일정 수준의 직관과 인사이트를 얻게 되었다는 것으로 본 프로젝트를 마무리한다.