**인공지능 MNIST 성능향상 과제**

컴퓨터과학부 2014920044 이홍석

1. **서론**

MNIST는 숫자를 손으로 쓴 그림파일들의 모음입니다. 이 MNIST를 CNN으로 학습시켜서 구분하는 것이 이번 수업시간에 배운 내용입니다. 수업시간에 사용한 예제를 활용하여 MNIST dataset을 학습할 경우(20000번) Accuracy가 최대 0.9924 정도 나왔습니다. 따라서 이번 실험에서는 다양한 파라미터들을 조절하여 최종적으로 Accuracy 0.994 이상을 목표로 진행해보겠습니다. 그리고 어떤 파라미터를 조절할 때 Accuracy에 어떤 영향을 끼치는지 다양한 실험을 통해 알아보겠습니다.

1. **실험 환경**
   1. Python 3.6.7, Python Virtual Environment
   2. Tensorflow-gpu 1.12.0
   3. Jupyter notebook
   4. Google Colab
   5. 사용 컴퓨터 사양: I7 8700K CPU, DDR4 16GB RAM, GTX1060 6GB
2. **예제코드 변경점**
   1. 예제코드 실험도중 Nan이 자주 뜨는 것을 발견하여 모든 변수를 tf.float64형으로 변경하였습니다.
   2. 그래픽 메모리가 자주 초과하는 현상이 발생하여 메모리 할당을 동적으로 추가할 수 있도록 Session을 수정하였고 평가를 4번으로 나누어 진행하였으며, 메모리를 더욱 많이 사용하는 경우는 batch 사이즈를 50에서 10으로 줄였습니다.
   3. Seed는 저의 학번인 ‘2014920044’로 설정하였습니다.
3. **실험한 파라미터 및 본문 목차**
   1. Hidden Layer의 수 조절
   2. Node 수 조절(전결합층)
   3. Filter 수 조절(Convolution Layer)
   4. Filter 크기 조절(Convolution Layer)
   5. Activation 함수
   6. 종합실험
4. **실험결과**
   1. Hidden Layer의 수 조절
      1. 3개

3개의 Hidden Layer(Convolution Layer + Pooling Layer)를 사용할 때는 2개일때 보다 학습속도가 느리고 Accuracy도 빨리 올라가지 않았습니다. 그래서 학습횟수를 4만회로 늘렸습니다. 세번째 Pooling Layer에서 7x7크기를 반으로 줄여야 하는데 문제가 생기므로 커널을 3칸씩 움직여서 padding까지 포함하여 3등분하였습니다. 이 실험의 결과 학습을 진행할수록 Loss도 늘어났습니다. 기존 코드를 사용했을 때보다 Accuracy가 0.001정도 올라간 것을 알 수 있었습니다. 최대 Accuracy는 0.9939 입니다.

* + 1. 4개

4개의 Hidden Layer를 사용할 땐 4번재 Pooling Layer에서 3x3크기를 반으로 줄이기 힘드므로 strides=[1,1,1,1]을 적용하여 3x3크기 그대로 사용하였습니다. 실험 결과 Accuracy가 오히려 3개일 때보다 0.001정도 낮았습니다. 게다가 Loss도 매우 올라가는 것을 알 수 있었습니다. 최대 Accuracy는 0.9923입니다.

* 1. Node 수 조절
     1. 512개

기본코드의 1024개를 사용했을 때보다 Accuracy가 오히려 조금 올랐습니다. 최대 Accuracy는 0.9931입니다.

* + 1. 2048개

512, 1024개때와 Accuracy가 거의 비슷한 것을 볼 수 있었습니다. 그러나 512개때보다 Accuracy가 조금 올랐습니다. 최대 Accuracy는 0.9933입니다.

* + 1. 4096개

이번에는 위의 실험들보다 Accuracy가 좀 더 높았습니다. 하지만 Loss는 위의 실험들 때보다 훨씬 올라간 것을 볼 수 있었습니다. 최대 Accuracy는 0.9940입니다.

* 1. Filter 수 조절
     1. 64, 128(Conv1, Conv2)

Filter의 수가 많으므로 전결합층의 노드 개수를 2048개로 설정하였습니다. 기존 코드를 사용했을 때보다 Accuracy가 좀 더 올라간 것을 볼 수 있었습니다. 그러나 Loss는 많이 올랐습니다. 최대 Accuracy는 0.9940입니다.

* + 1. 128, 256(Conv1, Conv2)

이번에는 Loss가 더욱 많이 늘어났으며 Accuracy는 위의 실험보다 더 떨어졌습니다. 최대 Accuracy는 0.9930입니다.

* 1. Filter 크기 조절
     1. 6x6(Conv1, Conv2)

기본 코드에서 Convolution Layer들의 Filter크기를 6x6으로 변경하였습니다. 기본 코드보다 Accuracy는 조금 올라갔지만 Loss도 올라갔습니다. 최대 Accuracy는 0.9935입니다.

* + 1. 10x10(Conv1, Conv2)

이번에는 10x10으로 변경하였습니다. 그 결과, Loss는 올라가고 Accuracy는 위의 실험보다 떨어졌습니다. 최대 Accuracy는 0.9922입니다.

* + 1. 3x3(Conv1, Conv2)

이번에는 반대로 3x3으로 변경하였습니다. 그 결과, Loss는 기본코드와 비슷하고 Accuracy는 기본코드보다 조금 올라갔습니다. 최대 Accuracy는 0.9935입니다.

* 1. Activation 함수
     1. tanh

기본코드에서 Activation function으로 relu를 tanh로 변경하였습니다. 학습속도가 느려 40000번 학습을 진행하였습니다. 그 결과, Loss는 기본코드와 비슷했고 Accuracy는 매우 떨어진 것을 볼 수 있었습니다. 최대 Accuracy는 0.9905입니다.

* + 1. sigmoid

이번에는 Activation function으로 sigmoid를 사용했습니다. Loss는 위의 실험보다 더 낮았습니다. 그리고 Accuracy는 기본코드와 비슷했습니다. 하지만 학습속도가 느려서 위 실험과 같이 40000번 학습시켰습니다. 최대 Accuracy는 0.9928입니다.

* 1. 종합실험

이번에는 위의 실험결과들을 이용하고 다양한 파라미터들을 마음대로 섞어서 실험 목표였던 Accuracy 0.994를 넘겨보도록 하였습니다. 최대 Accuracy는 0.9945입니다. 이 실험에 사용된 코드의 Layer 구성과 정보는 다음과 같습니다.

* + 1. Convolution Layer 1: [5,5,1,64], relu
    2. Convolution Layer 2: [5,5,64,128], sigmoid
    3. Pooling Layer 1: [28,28]->[14,14]
    4. Convolution Layer 3: [5,5,128,256]
    5. Convolution Layer 4: [3,3,256,256], relu
    6. Pooling Layer 2: [14,14]->[7,7]
    7. Convolution Layer 5: [3,3,256,512], relu
    8. Pooling Layer 3: [7,7]->[3,3]
    9. 전결합층: 2048개, dropout(0.5), relu
    10. Softmax
    11. Train 횟수: 60000번
    12. 학습률: 0.0001

1. **결론**

다양한 파라미터들을 변화시킨 결과 처음 생각했던 것만큼 Accuracy가 올라가지 않았습니다. 좀더 복잡하게 파라미터들이 얽혀야 더 좋은 결과가 나올 수 있는 것 같습니다. 파라미터별로 Accuracy와의 관계를 분석해보겠습니다. 먼저 Hidden Layer의 수입니다. 3개일때 Accuracy가 올라갔지만 4개일때는 올라가지 않았습니다. 4개일때는 다른 파라미터를 조절해야 더 올라가는 것 같습니다. 전결합층의 노드 수를 늘리거나 Filter의 수를 같이 늘려야 할 것 같습니다. 다음으로 전결합층의 노드 수는 높을수록 Accuracy가 높았습니다. 하지만 Loss도 많이 높아졌습니다. Loss가 내려갔다가 다시 올라가는 이유는 Training Set에 대한 Overfitting이 발생하였기 때문입니다. 따라서 중간 Layer에 Dropout을 넣는 등 다른 파라미터를 같이 변화시켜야 합니다. Filter 수에 대한 결과는 5x5와 비슷하게 움직이는 것이 좋은 것 같습니다. 오히려 너무 크면 Accuracy가 떨어지는 결과를 보였습니다. Activation 함수는 ReLU가 제일 결과가 좋지만 계산과정에서 Nan이 나올 가능성이 높기 때문에 Layer가 Deep해질경우 특정 Layer에 sigmoid나 tanh를 넣어주는 것이 좋은 것 같습니다. 이렇게 다양한 파라미터들과 그들이 어떤 영향을 끼치는지에 대해 실험해보았습니다. 보고서와는 별개로 다양한 실험을 해보았고 그중 가장 좋았던 결과를 종합실험 칸에 적었습니다. Colab과 개인 GPU를 사용하여 실험을 진행하였으나 생각보다 그래픽 메모리를 많이 사용하여 원하는 모든 실험을 해볼 수는 없었기 때문에 좀 아쉬웠던 실험이었습니다. 그래도 이번 실험을 통해서 파라미터들에 대한 감각이 생긴 것 같습니다. 만약 다른 그림 문제에 대해 CNN을 구현해야 한다면 이번 실험에서 생긴 감각을 이용하여 Accuracy를 많이 올릴 수 있을 것 같습니다.