**[Project 1] CIFAR-10 분류**



|  |  |
| --- | --- |
| 교과목명 | 딥러닝 및 실습 |
| 담당교수님 | 홍두화 교수님 |
| 학과 | 컴퓨터‧메카트로닉스공학부  소프트웨어전공 |
| 학번 | 2019101412 |
| 이름 | 이예찬 |
| 제출일 | 2022. 11. 20. 일요일 |

1. 실험 목표

: 수업 시간에 MLP와 CNN을 학습하면서 매주 실습을 진행했지만 확실하게 구조를 이해하기에는 부족했다. 따라서 이번 실험을 진행하면서 MLP와 CNN의 구조를 이해하고 각각의 층에서 어떤 역할을 하며, 하이퍼 파라미터가 성능을 높이기 위해 각각 어떤, 그리고 얼만큼의 역할을 하는지 알고자 한다.

1. MLP 실험 분석 및 결과
2. model\_mlp = tf.keras.models.Sequential()
3. model\_mlp.compile(loss = "sparse\_categorical\_crossentropy",
4. optimizer = "adam",
5. metrics = ["accuracy"])
6. history = model\_mlp.fit(X\_train, Y\_train,
7. epochs=30,
8. validation\_data=(X\_valid, Y\_valid),
9. batch\_size=200,
10. callbacks=[early\_stopping\_callback,tensorboard\_callback])
11. model\_mlp.add(Flatten(input\_shape=[32,32,3]))
12. model\_mlp.add(Dense(750, activation='relu'))
13. model\_mlp.add(Dense(300, activation='relu'))
14. model\_mlp.add(Dense(128, activation='relu'))
15. model\_mlp.add(Dense(64, activation='relu'))
16. model\_mlp.add(Dense(32, activation='relu'))
17. model\_mlp.add(Dense(10, activation='softmax'))

* 가장 간단한 구조인 Sequential API로 초기 모델을 구성한다.
* 컬러 이미지이기 때문에 3개의 채널으로 구성되어 있고 앞에서 데이터의 shape으로 확인해봤을 때 32\*32픽셀을 가진 이미지였기 때문에 입력층을 위와 같이 넣어준다.
* 처음엔 흑백과 컬러로 간단하게 구성된 뉴런 300-100개인, 활성화 함수는 ReLU인 은닉층을 넣어주고 마지막으로 뉴런 10개를 가진 Dense 출력층을 주었음. 이 출력층은 소프트맥스 활성화 함수를 사용. 이 결과 각각 41.73%, 49.46%로 높지 않은 결과를 얻을 수 있었다.
* 앞에서는 optimizer를 sgd를 썼지만 adam으로 바꾸고 층의 개수도 늘리고 batch size를 추가하여 다시 시도를 해 봤다. 또한 층의 뉴런 개수를 늘리고 줄이면서 여러 시도를 도전했다.
* 최종으로 가장 성능이 좋았던 모델은 뉴런 700-258-128-64-32-16인 모델로 batch size가 250였고 최종 성능은 53.19%을 얻을 수 있었다.
* 중간에 콜백함수를 써서 성능이 더 좋아지지 않으면 일찍 끝낼 수 있도록 했다.

1. CNN 실험 분석 및 결과
2. model = tf.keras.models.Sequential()
3. model.add(Conv2D(32,(3, 3), padding='same', strides=1, activation='relu', input\_shape=(32,32,3)))
4. model.add(Conv2D(32,(3, 3), padding='same', strides=1, activation='relu'))
5. model.add(MaxPooling2D(2,2))
6. model.add(Dropout(0.25))
7. model.add(Conv2D(64,(3, 3), padding='same', strides=1, activation='relu'))
8. model.add(Conv2D(64,(3, 3), padding='same', strides=1, activation='relu'))
9. model.add(MaxPooling2D(2,2))
10. model.add(Dropout(0.25))
11. model.add(Conv2D(128,(3, 3), padding='same', strides=1, activation='relu'))
12. model.add(Conv2D(128,(3, 3), padding='same', strides=1, activation='relu'))
13. model.add(MaxPooling2D(2,2))
14. model.add(Dropout(0.25))
15. model.add(Conv2D(256,(3, 3), padding='same', strides=1, activation='relu'))
16. model.add(Conv2D(256,(3, 3), padding='same', strides=1, activation='relu'))
17. model.add(MaxPooling2D(2, 2))
18. model.add(Dropout(0.25))
19. model.add(Flatten())
20. model.add(Dropout(0.5))
21. model.add(Dense(512, activation='relu'))
22. model.add(Dense(258, activation='relu'))
23. model.add(Dense(128, activation='relu'))
24. model.add(Dense(64, activation='relu'))
25. model.add(Dense(32, activation='relu'))
26. model.add(Dense(10, activation='softmax'))
27. model.summary()
    * 초기 모델은 MLP와 같이 Sequential 모델을 썼고, 처음에는 각 Conv층을 하나씩만 구성하고 마지막 은닉층도 하나씩만 구성해서 시도해봤다. 처음에는 MLP와 성능이 비슷하게 51.16%으로 얻을 수 있었다.
    * 두번째 시도 때는 conv층을 두 개로 늘려 비선형성을 포함하여 더 특성을 잘 표현하고, 더 적은 파라미터로 강력한 특성을 학습하려고 했고, 은닉층은 그대로 두고 학습해 보았다. 정확도는 비슷했다.
    * 세번째는 은닉층을 더 추가해 보았다. MLP 학습 시 은닉층을 추가했을 때 정확도가 향상되는 것을 보고 앞에서 정확도가 가장 높았던 은닉층 구조를 가져와서 더해보았다. 그 결과 정확도가 확실히 향상되는 것을 볼 수 있었다.
    * 네번째는 MLP에서 batch size 조절 시에 변화가 있었기 때문에 batch size를 조절해 보았다. 하지만 큰 변화는 없었기 때문에 CNN의 경우 batch size보다는 층의 구성에 따라 정확도가 많이 달라짐을 알 수 있었다.
    * 다섯 번째 시도에서는 필터를 256개로 설정한 Conv층을 두 개 더 추가해서 시도해 보았다. 조금 정확도가 나아져 80%가 넘는 정확도를 처음으로 얻을 수 있었지만 큰 차이가 있던 건 아니였기 때문에 성능을 높일 수 있는 다른 방법이 없을지 강구하려고 했다.
    * 여섯 번째 시도에서는 원래 CNN의 구조에는 Conv층에는 dropout이 적용되는 방법을 많이 사용하는 예를 많이 볼 수 있었고, 이 방법이 성능을 높인다는 논문을 찾았기 때문에 은닉층에 적용되는 dropout은 신경 쓰지 않았다. 그래서 이제까지 실험에는 은닉층에는 적용하지 않았는데 이 이유 때문에 정확도가 높아지지 않았나 하는 의문이 들었다. 그렇기 때문에 이에 대한 정보를 찾아보고 은닉층에 dropout을 0.5 적용하면서 실험해 보았다. 조금 높아진 80.77%의 정확도를 얻게 되었다.
    * 일곱 번째 시도에서는 이 구조에 성능을 높이기 위해 이미 훈련된 모델을 참고해보고자 했다. Vgg6라는 모델을 한 예제를 보고 층 구성을 살펴본 뒤 비슷하게 구성해 보고자 했다. 3~5번째 conv층에 한 층 씩 더 추가하고 은닉층은 줄이는 방식으로 구성했다.
    * 여덟 번째 시도에서는
28. 시사점

* MLP
* optimizer는 adam을 써야지 효과적이었다.
* batch size를 넣어야지 정확도가 상승했다.
* 같은 모델로 batch size만 바꿔서 하면 학습율이 더 좋아지는 듯 했다.
* 처음부터 다시 런타임을 돌려야 정확한 결과값이 나오는 듯 하다.
* layer는 너무 많이 넣거나 너무 적게 넣어도 좋지 않다. 입력층과 출력층을 포함해서 7개의 층을 넣으면 적당했다.
* CNN
* 일단 이렇게 실행시키는데 실행시간이 오래 걸리는지 몰랐다. 컴퓨터 자원을 많이 쓰기도 하고 런타임이 돌다가 멈추기도 했기 때문에 모델을 만들 때 이런 부분도 고려를 해야 한다는 것을 배웠다.
* MLP의 구조를 바탕으로 은닉층을 적용했을 때도 성능이 올라가는 것을 알 수 있었다.
* 생각보다 의외의 부분에서 성능을 높일 수 있다는 것을 배웠다. 예를 들어, Dropout은 단지 과적합을 막기 위한 수단이라고 생각했는데, 성능을 높이기 위해 dropout을 적용하는 논문도 찾을 수 있었다.