인공지능개론 과제 2: Image Classification

2021312063 조재희

CNN모델의 정확도를 개선하기 위해

1. 첫째, 제공된 baseline 코드가 어떻게 구성되어 있는지 파악하고 개선할 점을 찾은 후
2. 둘째, 하이퍼파라미터와 filter 크기를 조절해 개선사항의 영향력을 파악해 보았으며,
3. 기존에 발표되었던 CNN이 응용된 모델을 적용해보고,
4. 최종 CNN 모델을 구현하였다.
5. **baseline 코드**

**hyperparameter**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**Data Set**

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

**model**

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

**model. symmary()**

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

**결과**: accuracy- 0.7800

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

라인, 그래프, 도표, 경사이(가) 표시된 사진

자동 생성된 설명라인, 그래프, 도표, 경사이(가) 표시된 사진

자동 생성된 설명

**<분석>**

데이터 증강을 위해 resize와 rotation을 기존 이미지에 적용하였으며, CNN모델을 사용하였다. baseline의 모델은 특징추출을 위한 Feature Extractor와 classifier 부분으로 이루어졌으며, Feature Extractor는 convolution layer와 pooling layeer를 반복적으로 구성하여 특징표현을 학습한 후에 classifier 레이어에서 input값을 분류한다. 이 모델의 Feature Extractor 부분은 conv2d, 활성화함수, maxpooling2d를 사용하여 두개의 층을 쌓았고, classifier 층에서는 Dense layer로 구성하였다. Dense layer에서의 학습을 위해 1차원 데이터로 바꾸기 위해 Desnse 앞에는 flatten layer를 두었다. 그 결과, 0.7800의 성능을 보였으며, baseline 코드에서 아직 적용해 보지 않은 방법을 생각해 볼 때 batch size를 줄이고 epoch를 늘려 모델이 학습을 더 세세하고 많이 진행할 수 있도록 할 수 있으며, layer를 더 깊게 쌓고 특징을 추출하는 filter의 적절한 수를 찾거나, depth가 깊어질수록 filter를 조절해 나가는 방향을 생각해 볼 수 있다. 또한 batch normalization이나 dropout등 overfitting을 피하기 위한 조치를 취하여 모델의 accuracy를 향상시킬 수 있을 것이라고 판단하였다.

개선1) 기존 baseline에서 epochs를 10에서 30으로 늘리고 batch\_size를 줄였다. 또한 filter의 수를 늘려 나가며 기존 코드에서 레이어를 더 깊게 쌓았다.

**hyperparameter**

텍스트, 스크린샷, 폰트, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

**model**

텍스트, 전자제품, 스크린샷, 소프트웨어이(가) 표시된 사진

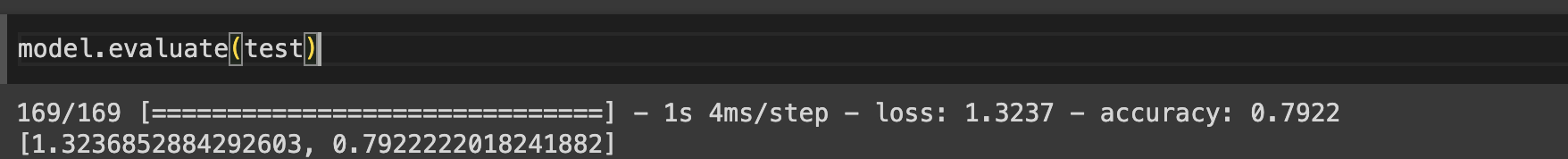
자동 생성된 설명

**model.summary()**

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

**결과:** accuracy - 0.7922



텍스트, 그래프, 라인, 도표이(가) 표시된 사진

자동 생성된 설명텍스트, 라인, 도표, 그래프이(가) 표시된 사진

자동 생성된 설명

**<분석>**

여러 번의 Parameter 수정과 filter 수 조절을 시행해 보았을 때 accuracy가 0.79~ 0.81 사이에서 정해졌으며 이를 통해 filter 수와 망의 깊이를 조절하는 것이 모델에 영향을 준다는 점을 알 수 있었다. 다만 개선1의 실행 결과에서는 기존 baselinecode 대비 뚜렷한 결과를 얻지는 못하였다. 또한 accuracy그래프를 보면, 8 epoch 부근에서부터 성능이 안나오면서 오히려 accuracy가 줄기 시작하는 모습을 보이는데 처음 hyper parameter로 epoch = 30 을 주었지만 overfitting으로 Early stop된 것을 보아, 이후 모델을 개선하기 위해서는 overfitting에 초점을 두는 것이 필요하다고 판단하였다.

**개선2) VGGNet 응용**

**model**

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**model.summary()**

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

**결과:** accuracy-8.704

스크린샷, 텍스트, 멀티미디어 소프트웨어, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 그래프, 도표, 라인이(가) 표시된 사진

자동 생성된 설명텍스트, 그래프, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

**<분석>**

VGGNet을 응용한 방식의 모델로 기존의 모델을 조금 수정하였다. VGGNet은 망의 depth가 성능에 주는 영향을 확실하게 파악할 수 있는 모델로, Convolution layer 뒤에 max-pooling을 해 나가며 이미지의 해상도를 줄여가는 방식이다. depth를 깊게 하기 위해 layer를 더 깊이 쌓을 수록 activation function, RELU가 더 많이 사용되면서 비선형성이 커져 유용한 특징을 파악할 수 있다는 점이 장점이다.

따라서 개선1)의 방식에서 depth를 더 깊게 하는 것이 모델의 성능 향상에 영향을 미친다는 점을 파악하고, VGGNet을 활용하고자 하였다.

모델은 3 블록으로 구성되어 앞쪽 두 블록은 Feature Extractor로 마지막 Block은 classifier로 두었다.

Feature Extractor는 한 block당 두번의 convolution과 활성화 함수를 거쳤다.

또한 앞선 개선1)의 방식에서 확인하였듯 overfitting이 학습에 미치는 영향을 고려하여 이를 피하기 위해 BatchNormalization을 각 활성화 함수 뒤에 배치하였으며, MaxPooling뒤에 0.5만큼의 Dropout을 두었다.

BatchNormalization는 신경망의 Input 즉, 각 batch 값을 평균 0, 분산 1로 정규화하고 Dropout은 해당 layer의 일부 노드의 activation을 지워 overfitting을 방지하는 효과를 기대할 수 있다.

마지막으로 Classifier 층에서는 모델의 복잡도를 줄이기위해 뉴런 수를 줄여가며 Dense층을 두 번 두었다.

그러나 모델 summary에서 볼 수 있듯, parameter의 수가 22,675,914로 Epoch 30에 10시간의 실행이 걸려 학습시간이 지나치게 오래 걸리는 문제가 있었고, 결국 resource의 부족으로 학습을 중단할 수밖에 없었다.

**개선 3)**

(1)filter 수를 유지하면서 feature classifier의 layer를 깊게 하는 방식을 이용하였다.

(2)가중치 초기화, 배치 정규화를 적극 사용하였다.

**model**

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

**model.summary()**

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**결과**: accuracy - 0.9152

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**<분석>**

개선2)에서 VGGNet은 파라미터의 수가 많아 연산의 크기가 크다는 점을 개선하기 위해서 망을 깊게 쌓는 대신, filter수를 일정하게 유지하면서 레이어를 더 쌓으면 전체 파라미터 수를 줄이는 방식을 이용하였다.

이 모델에서는 filter 수를 64로 유지하면서 Conv2D,BatchNormalization,Activation조합의 층을 두번 쌓아 Maxpooling하고 이를 3번 반복하는 식으로 Feature Extraction 부분을 구성하였고, Classifier는 모델의 복잡도를 줄이기 위해 Dense의 뉴런 수를 두 번 줄이고 dropout을 이용하여 과적합을 방지하도록 하였다.

model.summary()에서 parameter의 수를 확인해보면 parameter의 수가 393,328으로 개선2의 parameter가 22,675,914 에 비하면 확연히 줄어듦을 확인 할 수 있다. 또한 이렇게 한정된 데이터셋에서 파라미터 수를 줄이면 overfitting의 방지 효과를 기대해 볼 수 있다. 또한 CNN의 특성 상 낮은 레이어에서는 간단한 특징을 학습하고, 높은 레이어에서는 이러한 낮은 레이어의 특징을 조합하여 더 복잡한 특징을 학습하므로, 필터 수를 일정하게 유지하면서 레이어를 깊게 쌓으면 이러한 추상화 및 특징 재사용이 강화되었을 것이라고 생각한다. 그러나, 성능이 더욱 개선되지 않는 것은 Depth가 깊어져 Vanishing Gradient 의 발생을 의심해볼 수 있다.

앞선 모델에서와는 달리 가중치 초기화를 사용하였는데, 이는 Keras에서 기본적으로 제공하는 가중치 초기화 방식은 ‘random\_uniform’으로 layer를 깊게 쌓을수록 Back propagation과정에서 Gradient가 제대로 조절되지 않는 문제가 발생할 수 있기 때문에 가장 자주 쓰이는 가중치 초기화 방식인 “he\_normal”을 사용해보았다.

**개선 4) dataset의 부족을 의심하고 trainset 과 validation set을 분리하지 않았다.**

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

[나머지 코드는 개선3)의 코드와 같으므로 생략.]

**결과 :** accuracy- 0.9330

필터가 50 일 때 결과: accuracy - 0.9057

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

필터가 64일 때 결과: accuracy -0.9330

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**<분석>**

최종 코드에서는 처음에 Data를 Load할 때에 trainset 과 validation set을 분리하지 않고 오직 trainset과 testset만을 8:2 비율로 구성하였다. 개선 3)에서 데이터 셋이 부족하다고 판단하여 최대한 Data를 학습에 활용하기 위한 과정이었다.

추가적으로 여러가지 필터(50->64)와 배치사이즈(15 vs 30)를 실험해 보았을 때, 배치사이즈가 15일때 가장 최적의 성능을 보여줌을 확인하였다.

**<최종 분석>**

결론적으로, 처음 Baseline Code의 성능 **accuracy - 0.7922** 에서 최종적으로 **accuracy- 0.9330**로 유의미한 결론을 도출해 낼 수 있었다.

수행한 과정을 요약하면 다음과 같다.

1. Epoch와 Batch size등의 Hyperparameter의 조정,
2. 적절한 filter수와 layer의 수 조정
3. layer를 쌓는 방식 바꿔보기
4. BatchNormalization, Dropout, 가중치초기화 추가
5. trainset을 최대한 활용하기

모든 과정을 종합해 볼 때, 도출할 수 있었던 사실은 첫째, 모델의 depth를 깊게 한다고 해서 무조적으로 accuracy를 높일 수 있는 것이 아니며 DataSet에 augment를 적용하거나 validation set을 없애 데이터의 수를 최대한 많이 사용하려고 하더라도 절대적으로 그 양이 부족하기 때문에 overfitting을 피하기 위한 방법을 찾는 것이 모델의 성능을 향상시키기 위한 핵심적인 부분임을 확인 할 수 있었다.

최종 코드 첨부

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 멀티미디어 소프트웨어, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명