**AI Final\_Project**

12150981 박중규

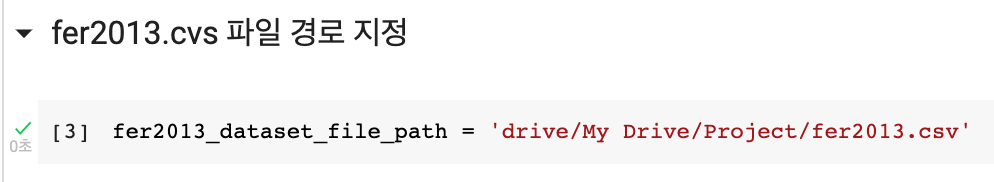
* **목차**
  1. 코드 과정 및 모델의 구조 설명
  2. 기존 모델과 개선한 모델 비교 및 결과 분석
  3. 결과에 대한 고찰 및 배운 점

1. **코드 과정 및 모델의 구조 설명**

텍스트이(가) 표시된 사진

자동 생성된 설명

* 프로젝트 진행을 위한 라이브러리들을 import 시킨다.  
  대표적으로 csv파일을 읽기 위한 csv 라이브러리,   
  배열 등의 자료구조를 손쉽게 다루기 위한 numpy 라이브러리,  
  이미지를 플로팅 시켜주는 matplotlib 라이브러리,  
  딥러닝 모델 제작 및 데이터 학습을 위한 tensorflow 라이브러리의 keras API가 있다.
* Google Colab 환경에서 코드를 수행하기 위하여 Google Drive를 마운트 시킨다.



텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

* 드라이브에 학습시킬 fer2013 데이터셋 파일을 업로드하고 경로를 지정한다 .
* 데이터 학습을 위한 변수, Validation을 위한 변수, Test를 위한 변수를 지정해준다.
* ‘fer2013.csv’ 파일을 반복문을 통해 각 행에 접근하여 필요한 데이터를 가져온다.  
  csv파일의 1번 째 열은 얼굴이 나타내는 감정의 이름(label), 2번 째 열은 이미지의 픽셀 데이터, 3번 째 열은 이미지의 용도(Training, Validation, Testing)를 나타낸다.  
  테이블이(가) 표시된 사진

  자동 생성된 설명  
    
  감정 label의 종류는 7가지로 구성되어있어 종류 별로 분류하여 label 배열에 번호를 부여하고,  
  Usage별로 분류하여 알맞는 변수에 추가한다.
* 데이터를 다 넣고 변수들을 numpy 배열로 변경시키고,  
  이미지 데이터는 48x48x1 크기로 reshape한다.  
  또 이미지 정규화를 위해서 이미지 픽셀의 스케일을 [0, 1] 범위로 조정해준다.

텍스트이(가) 표시된 사진

자동 생성된 설명

* 저장된 데이터 인스턴스 예시를 출력하여 확인해본다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

* 해당 프로젝트에서는 기존 모델과 개선한 모델, 2가지 모델을 만들고 테스트해보았다.
* CNN(Convolutional Neural Network, 합성곱 신경망) 모델은 인간의 시신경 구조를 모방한 기술로써 feature map을 사용하는 필터를 통해 학습하여 Computer vision 분야에서 성능이 우수하고, 특히 이미지를 인식하여 패턴을 찾는데 유용하다.  
    
  CNN은 크게 이미지의 1)특징을 추출하는 부분과 2)클래스를 분류하는 부분으로 나눌 수 있다.  
    
  1) 특징 추출 영역  
  특징 추출 영역은 Convolution Layer와 Pooling Layer를 여러 겹 쌓는 형태로 구성된다.  
    
  - Convolution(합성곱) Layer  
  필터를 통해 이미지의 특정 feature를 뽑아내는 layer이다. input data와 filter를 convolution시켜서 feature map을 얻는다. 즉 해당 층을 통해 이미지의 특징을 판단하게 된다.  
    
  - ReLU(Rectified Linear Unit)  
  활성화 함수로써 다른 활성화 함수들이 갖는 사라진 경사도 문제를 해결하기 위한 함수이다. 활성화 함수를 통해 선형적인 convolution의 결과에 비선형성을 추가할 수 있다.  
    
  - Batch Normalization  
  학습 과정에서 각 배치 별로 평균과 분산을 이용하여 정규화시키는 것이다.  
    
  - Pooling Layer  
  연산량을 줄이기 위하여 특정 feature을 강조하여 크기를 줄이는 layer이다.   
  크게 Max Pooling, Average Pooling, Min Pooling의 3가지 방법이 있는데 각각 선택 영역에서 가장 큰 값, 평균값, 가장 작은 값을 대표값으로 지정하게 된다.  
    
  2) 클래스 분류 영역  
  클래스를 분류하는 부분은 Fully Connected Layer(Dense Layer)이다.  
    
  - Flatten Layer  
  데이터 타입을 Fully Connected 네트워크 형태로 변경시켜준다. 입력 데이터의 shape을 변경하게 된다.  
    
  - Softmax Layer  
  Classification을 수행한다.  
    
  - Dropout Layer  
  복잡도를 줄여 overfitting을 방지시켜준다.

테이블이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

* 기존 모델의 학습결과이다.   
  Loss는 약 1.8216, Accuracy는 약 24.9373이다.

테이블이(가) 표시된 사진

자동 생성된 설명

* 기존 모델과 다른 점은 Average pooling 사용, Global Average Pooling사용이 있다. 자세한 차이점 비교는 뒷장에서 다루겠다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

* 개선한 모델의 학습결과이다.  
  Loss는 약 1.2560, Accuracy는 약 52.5733이다.

1. **기존 모델과 개선한 모델 비교 및 결과 분석**

|  |  |  |
| --- | --- | --- |
|  | **기존 모델** | **개선한 모델** |
| **Pooling** | Max Pooling | Average Pooling |
| **Classification** | Fully Connected Layer (Flatten Layer) | Global Average Pooling |
| **Loss** | 1.8216 | **1.2560** |
| **Accuracy** | 24.9373 | **52.5733** |

* + 기존 모델과 개선한 모델 비교
  + 결과분석  
      
    1. Pooling

Max Pooling을 사용하게 되면 영역의 최대값을 이용하기 때문에 이미지의 두드러지는 특징을 찾을 수 있다. 따라서 많은 이미지 인식에서 Max Pooling을 사용한다.  
Average Pooling을 사용하게 되면 영역의 평균값을 이용하기 때문에 덜 중요한 요소를 포함시킬 수 있는 반면, 분산을 이용할 수 있다. 분산을 사용하면 물체의 위치를 보다 쉽게 파악할 수 있어서 객체탐지분야에서 효과적으로 활용할 수 있다.

따라서 Average Pooling는 평균값을 이용하고 분산을 사용하기 때문에 이미지를 smoothing 시키는 효과가 있다.   
또 Max Pooling은 밝은 픽셀(값이 높음)에 잘 반응하는 반면 Average Pooling은 중간 밝기의 픽셀(값이 중간)에 잘 반응하게 된다.  
프로젝트에서는 fer2013의 이미지 데이터를 흑백으로 변환시켜 사용하게 되는데, 이미지 출력 시 하얀색이나 검은색의 극값에 가까운 색보다 회색의 중간값 색이 많음을 알 수 있다. 따라서 smoothing 효과와 중간값에 더 잘 반응하는 특징을 통하여 더 성능이 좋은 결과를 가질 수 있다고 예측해볼 수 있다.

2. Classification  
Fully Connected Layer는 위치 정보를 담고 있던 feature map 데이터를 dense한 형태로 flatten 시켜 1차원 데이터로 바꾸기 때문에 위치정보를 손실하게 된다.  
Global Average Pooling는 feature map에서 하나의 필터 내에 있는 모든 픽셀 값을 average pooling하여 1개의 값으로 수렴시킨다. 각 공간 차원이 1이 될 때 까지 공간 차원에 Average Pooling을 적용하고 다른 차원은 변경하지않고 그대로 둔다.  
CNN에서 Fully Connected Layer는 overfitting되는 경향이 있다. 이는 Dropout Layer등 을 통해 개선할 수 있는데, Global Average Pooling을 이용하여서도 overfitting을 낮출 수 있다. Global Average Pooling에서는 파라미터를 최적화 할 필요가 없기 때문에 overfitting을 피할 수 있다. 이러한 점이 성능을 향상시켰다고 예측해볼 수 있다.

1. **결과에 대한 고찰 및 배운 점**
   * 프로젝트를 진행하면서 CNN 모델 구조에 대해 보다 자세히 이해할 수 있었다.  
     프로젝트를 진행하기 위해서 CNN 모델과 내부의 각 Layer에 대한 보다 깊은 이해가 필요했고 자료를 찾고 직접 코드를 만들면서 모델에 대해 자세히 이해할 수 있었다. 파라미터값들이나 변수값들을 바꿔보며 모델을 학습시켜보는 경험을 할 수 있었고, 결과적으로 기존 practice 모델보다 2배 이상의 성능을 가지는 모델을 만들 수 있었다.  
     또 코드를 직접 타이핑하면서 import한 라이브러리들에 익숙해질 수 있었는데, 특히 tensorflow의 keras API의 함수들에 익숙해질 수 있었고 각 파라미터들이 의미하는 바를 공부할 수 있었다.
   * 기존 모델을 더 나은 성능을 가진 모델로 개선시키는 결과를 가져올 수 있었는데, 모델의 구조를 극적으로 변화시킨다거나 많은 Layer를 추가하는 등의 큰 변화를 주지는 않았다. 기본적인 모델의 구조는 같으나 Pooling의 방법, Classification 부분에서 작은 변화를 주었음에도 2배 이상으로 성능을 향상시킬 수 있었다.  
     이에 대해서 모델의 성능을 향상시킬 때 모델에 큰 변화를 가져오는 방법도 좋지만 input data에 대한 자세한 이해를 통해 적절한 함수를 사용하는 것 만으로도 큰 차이가 있음을 알 수 있었다. 또 overfitting은 성능을 크게 저하시키는 요인임을 확인할 수 있었고, 따라서 overfitting을 방지하기 위한 방법들을 적절히 사용해야함을 알 수 있었다.