인공지능기초와활용

**Project #1 Image Classification Competition**

|  |  |
| --- | --- |
| 담당교수 | 이종환 |
| 학번 | 2018930012 |
| 학과 | 전자전기컴퓨터공학부 |
| 이름 | 박수영 |
| 제출일자 | 2022.05.30 |

이미지 분류 경진대회 보고서

|  |
| --- |
| 목차 |

1. 소개

2. 수업요약

3. 이미지 분류를 위한 사전 조사

4. 프로젝트 알고리즘 설계과정

5. 학습기록

6. 참고문헌

**1. 소개**

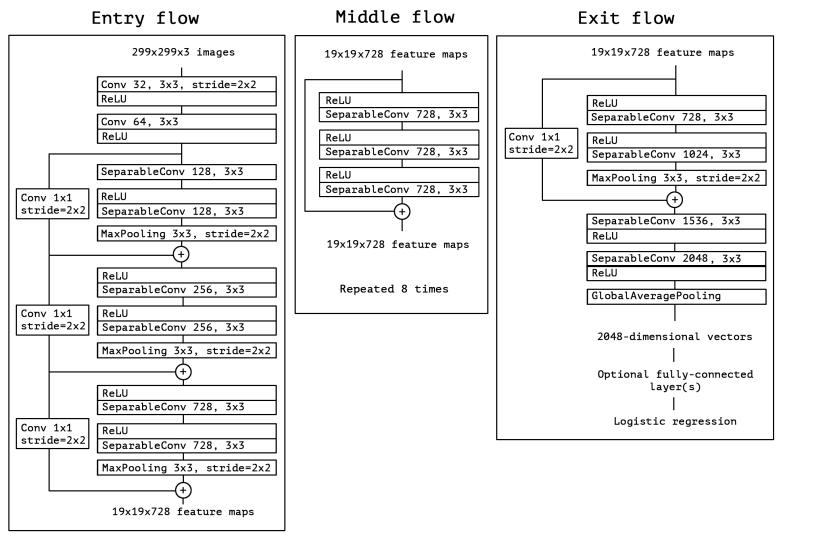
본 자료는 서울시립대학교 2022-1학기에 개설된 공학소양 교양교과목 ‘인공지능 기초와 활용’의 이미지 분류 경진대회 실험보고서이다. 경진대회는 2022.05.08~2022.05.30 동안 Kaggle 플랫폼에서 진행하였다.

**2. 수업요약**

1. **Deep learning**: 머신러닝의 한 분야로 컴퓨터가 방대한 데이터와 해답을 바탕으로 학습 모델을 생성해 예측을 하는 기법이다. 딥러닝은 연속된 층(Layer)을 사용하며 정형화된 데이터를 받지 않고 필요한 데이터를 수집 처리한다. 딥러닝은 구조적 데이터보다는 이미지와 자연어 등 지각에 관한 문제를 다루며 CNN, RNN등의 모델기법을 사용한다.
2. **Image data set**: 이미지 처리를 위해 사전에 분류된 데이터 셋으로 이미지 데이터와 레이블로 구성된다. 가로x세로의 크기가 1200x800인 컬러 고양이 사진의 경우, 데이터는 1200x800x3 (0~255)의 정수데이터로 나타내 진다. 레이블은 해당 이미지의 분류를 나타내는 지표로 [0:’dog’, 1: ‘cat’, 2:’truck’] 과 같이 사전에 지정해 놓는다. 만약 이미지 처리결과가 ‘0’이라면 알고리즘은 해당 이미지를 ‘dog’로 판단한다.
3. **Supervised learning:** 데이터에 대한 레이블(정답)이 주어진 상태에서 학습하는 방식이다. 학습한 모델로 테스트 데이터의 결과를 예측한다.
4. **Classification:** 여러 개의 결과값 중에 하나를 선택하는 것으로 Image classification 등이 있다.
5. **CNN:** Convolution Neural Networks. 합성곱 신경망은 인간의 신경 구조를 모방하여 만들어졌다. Convolution층. Pooling층, Fully\_connected 층 등으로 구성된다.
6. **Convolution층:** tensor와 tensor를 conv 연산 하는 층이다. 합성곱이 실행되는 단위 Filter가 tensor를 흩고 가며 conv 연산을 진행한다. 이미지의 전영역이 합성곱을 완료하면 Feature map을 출력한다.n개의 필터를 사용하면 Feature map은 n의 두께를 가진다.
7. **Padding:**  Input과 output의 size를 조정하는 것으로 Input data에 0으로 pad를 더해주는 것이다.
8. **Pooling 층:** 이미지의 크기를 유지한 상태에서 Fully connected layer로 이동한다면 연산 량이 크게 증가한다. 이미지의 크기를 줄이면서 특징을 유지하기 위해 pooling 층을 사용한다. CNN이 처리 해야 하는 이미지의 크기가 크게 줄기 때문에 학습시간을 절약하고 과적합을 방지할 수 있다.
9. **과적합해소: Data augmentation:** 과적합의 해소방안으로 적은 수의 학습 데이터를 보완하기 위해 데이터를 증강하는 방법이다. 이미지를 반전시키거나, 크기조정, 회전, 밝기 조정 등의 방법을 사용한다.
10. **과적합해소: Dropout:** 뉴런을 모두 연결하면 과적합이 발생해 모델이 진동할 수 있다. 이 문제를 해결하기 위해 Dropout을 사용하여 네트워크가 풍부한 표현을 가지도록 한다.
11. **과적합해소: Batch normalization:** 학습속도를 개선하고, 과적합을 해소 하기위해 사용한다. 중간 은닉층들을 정규화하여 학습효율을 개선할 수 있다.
12. **최적화:** RMSProp(Root Mean Square Propatation)의 약자로 기울기의 강하 속도를 증가시키는 알고리즘이다.
13. **Transfer learning:** 이미 훈련된 모델의 특성을 추출하고, 미세 조정을 가해 학습에 사용하는 것이다.

**3. 이미지 분류를 위한 사전 조사**

1. **Xception:** 이 모델은 CNN에서 각 채널간, 공간상의 상호 연결이 전체적으로 끊어질 수 있다는 가설에서 출발했다. Xception 아키텍처는 36개의 Convolution 층이 존재한다. 첫번쨰와 마지막 모듈을 제외하고, 모두 linear residual 모듈이 연결되어 있다. 즉, Xception 아키텍처는 depthwise seperable convolution 층의 linear stack이다. 30~40줄의 코드로 동작하기 때문에 다른 모델들 보다 쉽게 사용할 수 있다.



**Figure 1. Xception model architecture**

데이터는 먼저 entry flow로 진입한다. 그 다음 middle flow를 8번 반복한 뒤, 마지막으로 exit flow를 거친다. 모든 Convolution과 Separable Convolution 층은 Batch normalization층을 거친다.

테이블이(가) 표시된 사진

자동 생성된 설명

**Figure 2. Xception model accuracy**

Imagenet, JFT 데이터셋에서Xception은 기존의 Inception V3보다 더 좋은 분류 성능을 보인다.

1. **Freezing:** 일반적으로 모든 가중치는 학습 가능하다. layer. trainable=False이면, 모든 층의 가중치가 학습불가능으로 변한다. 이것을 ‘Freezing’이라 하다. Freezing된 층들은 학습시간 동안 수정되지 않는다.
2. **Fine tuning:** 한 번 모델이 새로운 데이터를 모으면 사용자는 모델의 처음부터 끝까지 모두 학습가능으로 만들 수 있다. 이 과정은 마지막 단계에서 성능향상을 기대할 수 있으며, 빠른 과적합을 유발할 수 있다. File-tuning은 매우 느린 학습속도를 가지고 있다. 그 이유는 이전과 달리 학습되어 있던 base\_model등을 훈련시켜야 하기 때문이다. 결과적으로 과적합이 빠르게 일어날 가능성을 막기 위해 적은 가중치를 사용해야 한다.

**4. 프로젝트 알고리즘 설계과정**

1. **데이터 분석:** 이미지 분류를 위한 자료로 test와 train 이미지로 구분되어 있다. Train 데이터는 cat 700, deer 600, dog900, horse 600개의 4종류 레이블로 총 2800개의 이미지로 구성된다. Test 데이터는 729개의 이미지가 레이블 없이 구성된다. 이미지의 사이즈는 제각각이며 (256,256,3)으로 통일하였다.
2. **데이터 불러오기:** 먼저, Kaggle/input/에 저장되어있는 raw 데이터들을 불러와 trainp와 testp에 저장하였다. Sample submission 파일을 불러와 sub에 저장하였다. ImageDataGenerator를 사용해 데이터를 증강하였다. Test 데이터용 tgen은 rescale만 하였다.
3. **데이터 변경을 위한 변수설정:** 데이터를 편리하게 수정하기 위해 변수 bs(batch\_size), target\_size1, isize, ishape를 설정헀다.
4. **훈련용, 검증용 데이터 설정:**  flow\_from\_directory를 사용해 trainp를 불러온다. Label이 정수이므로 sparse를 사용한다. 훈련용 데이터는 shuffle하여 학습시킨다.
5. **테스트용 데이터 설정:** flow\_from\_dataframe을 사용해 test파일을 불러온다. x축은 ‘file’로, class\_mode: None으로 설정한다.
6. **CNN라이브러리 불러오기:** CNN라이브러리를 불러온다. Conv2D를 비롯한 메소드를 가져온다.
7. **전이학습 모델 불러오기:** 사전에 훈련된 Xception모델을 불러온다. 가중치는 Imagenet으로 사전에 훈련된 가중치를 적용한다. Input\_shape: (256,256,3)
8. **전이학습 모델+ CNN모델 설정**: 전이학습 모델은 학습불가능으로 한다. (256,256,3)차원의 데이터가 입력으로 한다. 입력된 데이터는 base\_model을 먼저 통과한다. 그 다음 globalAveragePooling을 사용해 출력을 4개의 변수에서 2개의 변수로 pooling한다. (None, 8, 8, 2048) =>(None, 2048). Dense층과 BatchNormalization층을 함께 배치하여 노이즈를 줄이고 과적합을 해소한다. Dropout를 사용하여 과적합을 해소한다. 출력은 4개의 매개변수를 softmax로 출력한다.
9. **모델 컴파일:** 최적화를 RMSProp을 사용하였다. 보폭을 줄일 때, 이전 맥락 상황을 반영하는 기법이다. Learning\_rate=0.001를 사용하였다.
10. **모델 fit:** traind 데이터를 epochs=30에서 fitting한다.
11. **시각화:** matplotlib를 사용해 loss, al\_loss, acc, val\_acc를 시각화한다.
12. **내보내기:**  model에 test 데이터를 predict한 값을 정수로 label에 저장해 내보낸다.

**5. 학습기록**

|  |
| --- |
| 1. CNN모델: MaxPooling2D와 BatchNormalization, Dropout, Flatten을 활용, bs=64, epochs=25  >>> loss: 0.7347 - acc: 0.6821 - val\_loss: 1.5665 - val\_acc: 0.3500  2. CNN모델: Data Augmentation: width\_shift\_range=0.2, height\_shift\_range=0.2, zoom\_range=0.2 추가  >>> loss: 1.3758 - acc: 0.3206 - val\_loss: 1.3740 - val\_acc: 0.3214에서 정지  3. CNN모델: Data Augmentation: width\_shift\_range=0.3, height\_shift\_range=0.3, zoom\_range=0.2  horizontal\_flip=True,vertical\_flip=True,shear\_range=0.1,brightness\_range=(0.6,1.2),rotation\_ range=20  >>> loss: 1.3812 - acc: 0.3214 - val\_loss: 1.3799 - val\_acc: 0.3214 에서 정지  4. CNN모델: target\_size=400  >>> loss: 1.3719 - acc: 0.3214 - val\_loss: 1.3716 - val\_acc: 0.3214 에서 정지  -----데이터 증강시 정확도 개선이 정지 되었다-----  5. CNN모델: Layer 추가  >>> loss: 1.3506 - acc: 0.3524 - val\_loss: 1.4158 - val\_acc: 0.3179. 진동하는 구간이 길다.  6. 전이학습:  >>> loss: 1.3697 - acc: 0.3214 - val\_loss: 1.3714 - val\_acc: 0.3214 알고리즘을 잘못설계  7. 전이학습 수정: Xception을 사용하기 위해 input을 -1~1로 스케일링 하였다.  >>>ValueError: Negative dimension size caused by subtracting 3 from 2 for '{{node conv2d\_136/Conv2D}} = Conv2D[T=DT\_FLOAT, data\_format="NHWC", dilations=[1, 1, 1, 1], explicit\_paddings=[], padding="VALID", strides=[1, 1, 1, 1], use\_cudnn\_on\_gpu=true](Placeholder, conv2d\_136/Conv2D/ReadVariableOp)' with input shapes: [?,2,2,16], [3,3,16,32].  8. 전이학습 수정:  >>> loss: 1.3490 - acc: 0.3337 - val\_loss: 1.3414 - val\_acc: 0.3250 진동  9. learning\_step: 0.0001=>0.01  >>> loss: 1.2937 - acc: 0.3857 - val\_loss: 1.3608 - val\_acc: 0.3964  10. epochs: 10=>20  >>>loss: 1.2847 - acc: 0.3933 - val\_loss: 1.2290 - val\_acc: 0.4857  11. Layer 2개 층 추가: BatchNorm, Conv2D  >>> loss: 1.2362 - acc: 0.3987 - val\_loss: 1.3506 - val\_acc: 0.2893 보폭이 크다  12. learning\_step: 0.01=>0.003, Layer 3개층 추가 BatchNorm, conv2D  >>> loss: 0.8012 - acc: 0.6808 - val\_loss: 1.9876 - val\_acc: 0.4268: acc는 증가하는데 Val\_acc는 진동  13. Layer 추가: Dropout  >>> loss: 0.9312 - acc: 0.6089 - val\_loss: 1.2439 - val\_acc: 0.5214  14. Layer추가: Dense층 4개  >>> loss: 1.2116 - acc: 0.4210 - val\_loss: 1.2660 - val\_acc: 0.4125: val 값이 진동을 많이 한다.  16. Layer 감소 Dense, Conv2D, Layer 추가: BatchNorm  >>> loss: 1.1965 - acc: 0.4254 - val\_loss: 1.2509 - val\_acc: 0.4143  17. 코드 재작성: Xception 모델의 input를 scale 된 것에서 scale 되지 않은 값으로 수정  >>> loss: 0.5168 - acc: 0.8393 - val\_loss: 0.1296 - val\_acc: 0.9679  18. Layer 추가 Dense층  >>> val\_acc: 0.9764  19. epchs 10=>30, Dropout값 축소  >>> loss: 0.1669 - acc: 0.9272 - val\_loss: 0.2489 - val\_acc: 0.9661 0.96에서 진동  20.Layer 수정: Dense층 매개변수 증가  >>> loss: 0.0282 - acc: 0.9906 - val\_loss: 0.2959 - val\_acc: 0.9696  21. Layer 감소: Conv2D층 삭제  >>> loss: 0.0276 - acc: 0.9933 - val\_loss: 0.2876 - val\_acc: 0.9696 :  제출: loss: 0.0272 - acc: 0.9902 - val\_loss: 0.3135 - val\_acc: 0.9625  0.97119  22. 데이터 증강  >>> loss: 0.3186 - acc: 0.8933 - val\_loss: 0.3430 - val\_acc: 0.8732  23. 데이터 증강: bias 축소, epochs 30=> 70  >>> loss: 0.1087 - acc: 0.9683 - val\_loss: 0.2870 - val\_acc: 0.9304  24. 데이터 증강 축소  >>> loss: 0.0229 - acc: 0.9911 - val\_loss: 0.2626 - val\_acc: 0.9714  25. fine tuning: base\_model도 훈련  >>>loss: 1.0088 - acc: 0.6121 - val\_loss: 9.0067 - val\_acc: 0.6339  26. FineTuning삭제  >>>loss: 0.3665 - acc: 0.8987 - val\_loss: 0.2757 - val\_acc: 0.9250  27. Dense층 개수 축소  >>>loss: 0.3462 - acc: 0.8929 - val\_loss: 0.7213 - val\_acc: 0.8429  28. batch\_size 50=>64, learning\_rate 0.06으로 변경  >>>loss: 0.2043 - acc: 0.9487 - val\_loss: 0.4383 - val\_acc: 0.8893  29. Dropout, Dense층 축소, learning\_rate=0.002  >>>loss: 0.0293 - acc: 0.9906 - val\_loss: 0.3045 - val\_acc: 0.9304  30. 모델 수정  >>>loss: 0.0378 - acc: 0.9884 - val\_loss: 0.3880 - val\_acc: 0.9179  31. 층 수정  >>> loss: 0.0400 - acc: 0.9897 - val\_loss: 0.4079 - val\_acc: 0.9054  32. **21번**으로 복귀  >>> 해당 항목을 제출 |

**6. 참고문헌**

1. Franc¸ois Chollet , “Xception: Deep Learning with Depthwise Separable Convolutions”
2. 인공지능 기초와 활용 수업자료
3. 사이토 고키, “밑바닥부터 시작하는 딥러닝”