COSE474-2019F: Final Project Proposal
"CNN을 활용한 Image Classifier"

2016320266 컴퓨터학과 박준성 (role : 데이터 수집 및 모델 구현)

1. Motivation and problem definition

Convolution Neural Network을 사용하여 개, 고양이, 말, 나비를 분류하는 4-class image classifier를 구현하였다. 개발 환경은 Colab를 이용하여 구현하였고, image data는 Kaggle에서 다운로드하였다. VGGnet의 구조를 참고하여 CNN을 구성했고, 정규화를 위한 dropout층을 추가하였다. 또한 Image generator를 통하여 한 이미지당 2 3개의 이미지를 추가로 생성하여 데이터의 양을 증가시켜 정확도를 상향시켰다.

2. Data Preprocessing

데이터는 Kaggle에서 수집하였고, 각각의 이미지는 약 2000장씩 수집하였다. 하지만 데이터가 약간 부족하다는 생각이 들어 Imagedatagenerator를 사용하여 이미지를 늘리거나 회전시켜 데이터의 양을 증가시켰다.
주요 parameter는 다음과 같다.

- rotation range: 이미지 회전 범위 (degrees)
- width shift, height shift: 그림을 수평 또는 수직으로 랜덤하게 평행 이동시키는 범위 (원본 가로, 세로 길이에 대한 비율 값)
- rescale: 그래서 이를 1/255로 스케일링하여 0-1 범위로 변환시켜준다.
- shear range: 임의 전단 변환 (shearing transformation) 범위
- zoom range: 임의 확대/축소 범위
- horizontal flip: True로 설정할 경우, 50% 확률로 이미지를 수평으로 뒤집는다. 원본 이미지에 수평 비대칭성이 없을 때 효과적이다. 즉, 뒤집어도 자연스러울 때 사용하면 좋다.
- fill mode: 이미지를 회전, 이동하거나 축소할 때 생기는 공간을 채우는 방식

이런 방식으로 각 이미지 당 2 3장씩 추가하여 각 클래스당 4000장 정도의 이미지 파일을 생성하였다.

데이터 변환: 수집한 데이터들을 model의 input 값으로 변환하기 위해 one hot encoding을 사용하였다. one-hot인코딩이란 단 하나의 값만 True이고 나머지는 모두 False인 인코딩을 말한다. 즉, 1개만 Hot(True)이고 나머지는 Cold(False)이다.즉 현재 4가지의 클래스를 가지므로 해당 클래스가 말이라면 [1,0,0,0], 개라면 [0,1,0,0], 고양이라면 [0,0,1,0], 나비라면 [0,0,0,1]로 구성했다.

데이터 분류: training data와 test data를 5:1로 분류했고, training data 중 validation data로 20%를 사용하였다.

3. CNN 모델 구성

VGG16을 참고하여 모델을 구성하였다. 총 3단계의 convolution layer와 마지막 1단계의 fully

Figure 1.
UP: 원본이미지
DOWN: Imagedatagenerator를 통해 생성된 이미지

connected layer로 구성했고, 각 convolution layer는 2번의 convolution과 1번의 maxpooling, 마지막에는 과적합을 방지하기 위해 dropout을 적용하였다. 또한 vgg16은 1000개의 클래스를 분류하는 network이기 때문에 fully connected network의 layer가 많지 않은데 여기서는 4개의 클래스로 분류해야 하므로 vgg16의 layer를 그대로 사용한다면 feature 값을 잃어버릴 수 있다고 생각해서 fully connected layer의 hidden layer를 추가하였다. 또한 optimizer로는 Adam을 사용했고, 마지막 출력층을 제외한 나머지 layer의 activation function은 Relu함수를 적용하였다. 출력층의 activation function은 multiclass classification이므로 softmax함수를 사용했고, loss function은 categorical entropy를 적용했다.

Layer (type)	Output Shape	Param #
conv2d_7 (Conv2D)	(None, 64, 64, 64)	1792
conv2d_8 (Conv2D)	(None, 64, 64, 64)	36928
max_pooling2d_4 (MaxPooling2	(None, 32, 32, 64)	0
dropout_5 (Dropout)	(None, 32, 32, 64)	0
conv2d_9 (Conv2D)	(None, 32, 32, 128)	73856
conv2d_10 (Conv2D)	(None, 32, 32, 128)	147584
max_pooling2d_5 (MaxPooling2	(None, 16, 16, 128)	0
dropout_6 (Dropout)	(None, 16, 16, 128)	0
conv2d_11 (Conv2D)	(None, 16, 16, 256)	295168
conv2d_12 (Conv2D)	(None, 16, 16, 256)	590080
max_pooling2d_6 (MaxPooling2	(None, 8, 8, 256)	0
dropout_7 (Dropout)	(None, 8, 8, 256)	0
flatten_2 (Flatten)	(None, 16384)	0
dense_4 (Dense)	(None, 2048)	33556480
dense_5 (Dense)	(None, 1024)	2098176
dropout_8 (Dropout)	(None, 1024)	0
dense_6 (Dense)	(None, 4)	4100

Figure 2. CNN model

4. CNN training

minibatch gradient descent 방식을 사용하여 batch size는 32로 정했고, 총 epoch은 100으로 하고 training을 진행했다. 또한 콜백함수로 checkpoint와 early stopping이라는 함수를 사용했는데, checkpoint 함수는 epoch를 진행하면서 가장 좋은 성능을 보인 parameter를 가지는 모델을 저장하는 함수이고, early stopping은 epoch을 일정 개수만큼 더 반복해도 성능이 더 이상 향상되지 않을 때 더 epoch을 진행하지 않고 training을 종료시키는 함수이다. training data의 accuracy와 loss는 epoch이 진행될수록 향상되지만 validation data의 loss는 일정기간 동안은 감소하지만 epoch이 진행될수록 다시

증가할 수 있기 때문에 이러한 training data에 대한 overfitting을 방지하는 것이 중요하다. 실제로 early stopping 이후에 성능이 더 좋아질 수도 있다고 생각하여 종료된 이후 모델을 계속 학습을 시켜봤지만 성능이 더 좋아지지는 않았다. epoch을 100으로 설정했지만 earlystopping 함수로 인해 실제 진행된 epoch수는 50이 되었다.

5. Main Contributions

모델의 정확도를 높이려면 데이터가 충분해야 하는데 데이터가 충분하지 않아 가진 데이터를 data augmentation을 통해 양을 늘려 정확도를 향상시켰다. 또한 과적합을 방지하기 위해 vggnet에 dropout을 추가하였고, callback 함수로 checkpoint와 earlystopping 함수를 사용하였다.

6. Baseline

Image classification을 위해 Convolution Neural Network를 사용했고, 그 중 Vggnet 구조를 기초로 하여 가지고 있는 data feature에 맞게 수정하였다.

7.SOTA

현재 Image Classification의 SOTA는 EfficientNet, FixResNeXt-101 32x48d, PNASNet-5, NASNET-A, SENet-154 등이 있다.(figure 5)

8. Main Challenges

프로젝트를 진행하면서 어려웠던 점은 accuracy와 loss가 생각보다 향상이 잘 되지 않는다는 것이었다. Fully connected layer수를 조정하거나 optimizer를 바꿔도 성능 향상이 잘 되지 않았다. 처음에는 loss가 local minimum에 머물러 있어 성능이 잘 올라가지 않는 것 같아 learning rate을 올렸지만 그래도 성능이 크게 좋아지지 않아 반대로 learning rate을 작게 하니 오히려 성능이 더 좋아지는 것을 발견할 수 있었다. 또한 network layer의 수와 neuron의 수를 줄이자 성능이 더 좋아지기도 하였다. 그래서 여러 조합들을 실험해본 뒤 그 중 최적의 모델을 만들 수 있었다.

9. Model evaluation

Epoch의 수에 따른 모델의 loss와 accuracy는 다음과 같다. test data에 대해 시험해 본 결과

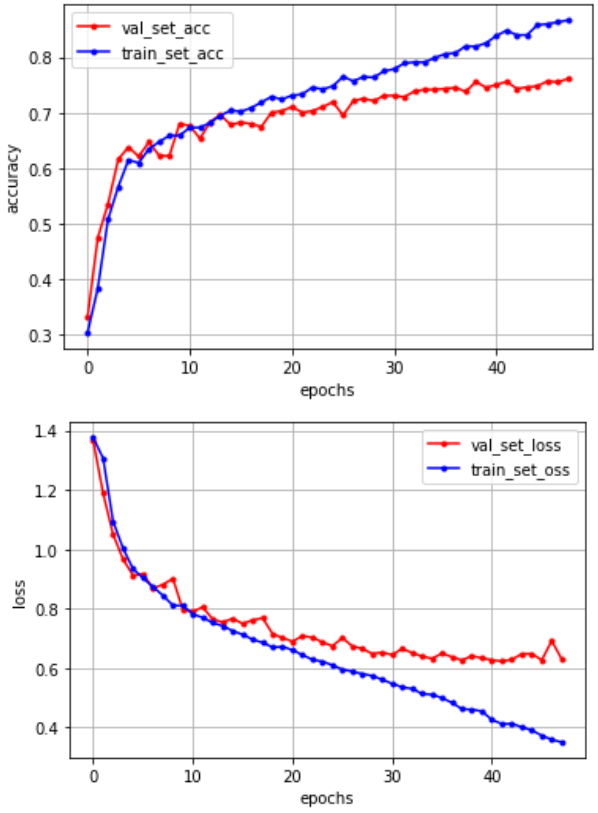

Figure 3. accuray, loss

accuracy는 76%가 나왔다.

```
1871/1871 [==============================] - 1s 758us/step
정확도 : 0.7600
```

Figure 4. test accuracy

10. Future direction

정해진 label을 가지고 training을 진행하는 supervised learning을 중심으로 구현했다면 앞으로는 Image Classification의 SOTA 중 하나인 Noisy Student가 사용하는 방식인 모델이 직접 data에 대한 psudo label을 만들어 학습을 진행하는 self supervised learning을 실제로 구현해 보고 싶다.

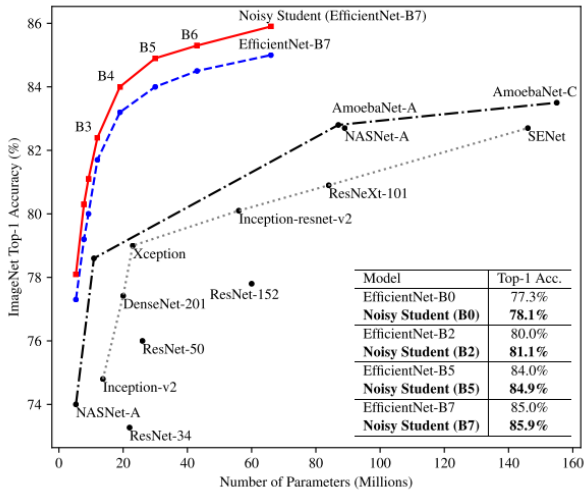

Figure 5. SOTA for Image Classification

11. Reference

1.https://paperswithcode.com/sota/image-classification-on-imagenet

2.https://www.kaggle.com/alessiocorrado99/animals10

3.https://hoya012.github.io/blog/Self-training-with-Noisy-Student-improves-ImageNet-classification-Review/