project 2

담당 교수 : 이혁준 교수님

분반 : 월 6 수 5

학번 : 2018202060

전공 :컴퓨터정보공학부

이름 : 이준형

Contents

Introduction

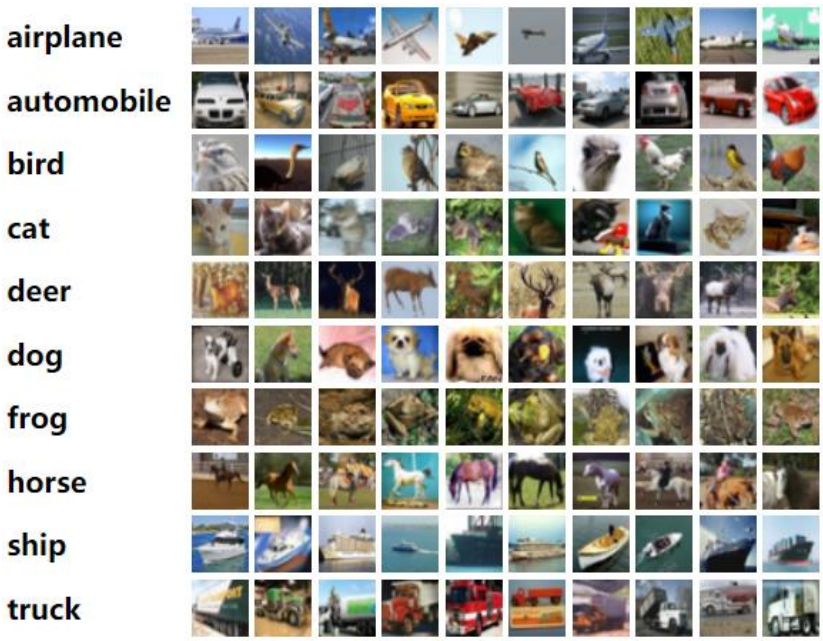
과제에 내용에 대한 설명

이번 인공지능 2 차 프로젝트는 tensorflow 안에 keras 라이브러리를 활용해서 cifar-10 dataset 을 CNN 모델을 구현해서 classification 하는것이 목표입니다.

Project 2-1 : 주어진 layer(5 개의 convolution layer 와 2 개의 pooling layer 그리고 fully connected layer)를 통해서 CNN 모델 구현

Project 2-2 : layer 의 순서, kernel size, stride, types of pooling, dropout, normalization, 등을 자유롭게 사용해서 test dataset 의 accuracy 가 최소 80%가 넘는 CNN 모델을 구현

구현에 앞서서 먼저 cifar-10 dataset 에 관해서 소개드리겠습니다. Cifar-10 은 기계 학습 및 컴퓨터 비전 알고리즘을 교육 하는 데 일반적으로 사용되는 dataset 입니다 . 60000 장, 32x32 컬러 이미지로 구성됩니다. 이미지에는 항공기, 자동차(트럭 또는 픽업 트럭 제외), 조류, 고양이, 사슴, 개, 개구리, 말, 배, 트럭(트럭 제외) 등 10 개의 상호 배타적 클래스 중 하나가 레이블로 표시됩니다. 클래스당 6000 개의 이미지가 있으며 클래스당 5000 개의 training 이미지 및 1000 개의 test 이미지가 있습니다.



이번 2 차 프로젝트의 핵심이 되는 tensorflow 중 keras 를 사용합니다.

keras 에서 layers, models, sequential 을 통해 모델을 설계하고, 설계한 모델을 models 중에서 load_model 로 model 을 불러올 수 있습니다.

Result

먼저 실습을 돌리기 앞서서 운영체제 Version 과 conda 안에 있는 라이브러리 버전들을 소개드리겠습니다. Keras 가 gpu 를 통해 학습하도록 코드를 설계했습니다.(제출코드에서는 주석처리 했습니다.)

OS : Window 10 Home

CPU : AMD Ryzen 5 3500X 6-Core Processor 3.59 GHz GPU : NVIDIA GeForce GTX 1660 SUPER

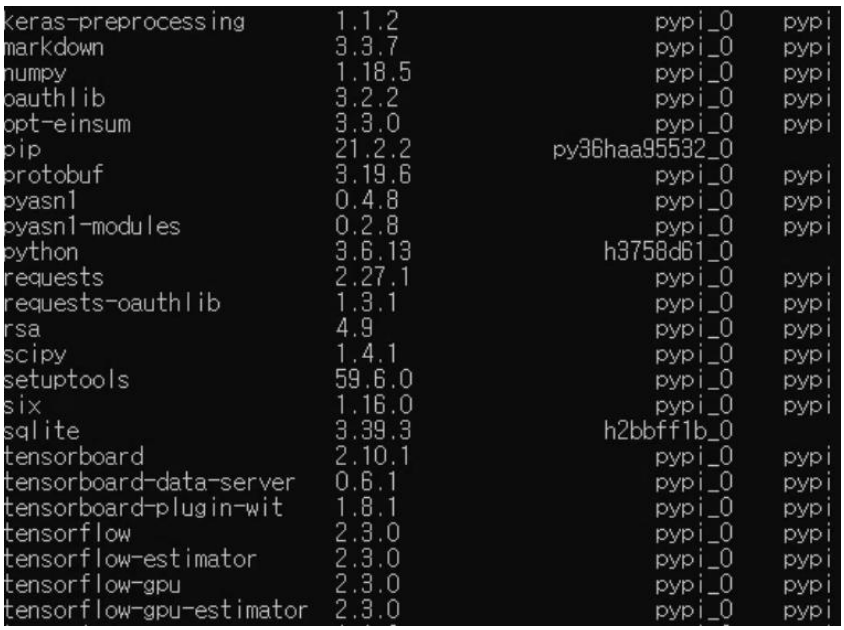
NVIDIA-SMI version : 516.94

CUDA version : 11.7

CuDnn version : 7.6.5

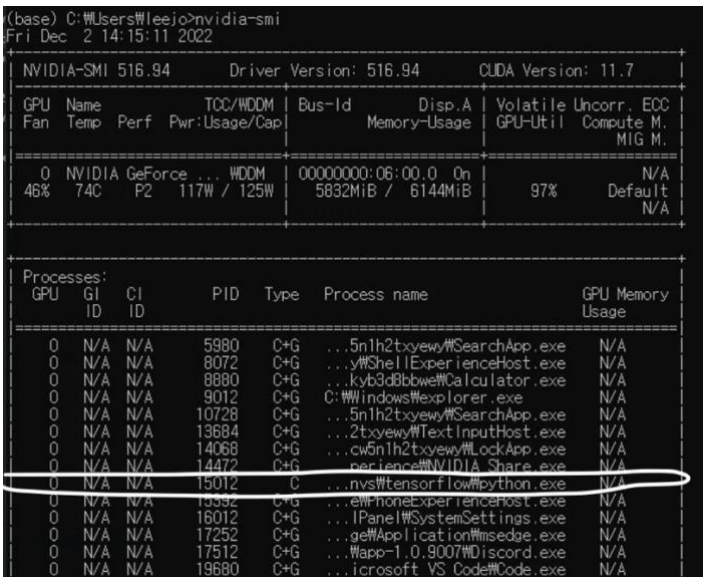
Python version : 3.6

Tensorflow version : 2.3.0



keras-preprocessing	1.1.2	pypi_0	pypi
markdown	3.3.7	pypi_0	pypi
numpy	1.18.5	pypi_0	pypi
oauthlib	3.2.2	pypi_0	pypi
opt-einsum	3.3.0	pypi_0	pypi
pip	21.2.2	py36haa95532_0	
protobuf	3.19.6	pypi_0	pypi
pyasn1	0.4.8	pypi_0	pypi
pyasn1-modules	0.2.8	pypi_0	pypi
python	3.6.13	h3758d61_0	
requests	2.27.1	pypi_0	pypi
requests-oauthlib	1.3.1	pypi_0	pypi
rsa	4.9	pypi_0	pypi
scipy	1.4.1	pypi_0	pypi
setuptools	59.6.0	pypi_0	pypi
six	1.16.0	pypi_0	pypi
sqlite	3.39.3	h2bbff1b_0	
tensorboard	2.10.1	pypi_0	pypi
tensorboard-data-server	0.6.1	pypi_0	pypi
tensorboard-plugin-wit	1.8.1	pypi_0	pypi
tensorflow	2.3.0	pypi_0	pypi
tensorflow-estimator	2.3.0	pypi_0	pypi
tensorflow-gpu	2.3.0	pypi_0	pypi
tensorflow-gpu-estimator	2.3.0	pypi_0	pypi

$conda list 를 입력했을때의 모습



```
(base) C:¥Users¥leejo>nvidia-smi
Fri Dec  2 14:15:11 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 516.94       Driver Version: 516.94       CUDA Version: 11.7     |
|-------------------------------+----------------------+----------------------+
| GPU  Name            TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ... WDDM  | 00000000:06:00.0  On |                  N/A |
| 46%   74C    P2   117W / 125W |   5832MiB /  6144MiB |     97%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      5980    C+G   ...5n1h2txyewy¥SearchApp.exe    N/A      |
|    0   N/A  N/A      8072    C+G   ...y¥ShellExperienceHost.exe    N/A      |
|    0   N/A  N/A      8880    C+G   ...kyb3d8bbwe¥Calculator.exe    N/A      |
|    0   N/A  N/A      9012    C+G   C:¥Windows¥explorer.exe         N/A      |
|    0   N/A  N/A     10728    C+G   ...5n1h2txyewy¥SearchApp.exe    N/A      |
|    0   N/A  N/A     13684    C+G   ...2txyewy¥TextInputHost.exe    N/A      |
|    0   N/A  N/A     14068    C+G   ...cw5n1h2txyewy¥LockApp.exe    N/A      |
|    0   N/A  N/A     14472    C+G   ...perience¥NVIDIA Share.exe    N/A      |
|    0   N/A  N/A     15012      C   ...nvs¥tensorflow¥python.exe    N/A      |
|    0   N/A  N/A     15332    C+G   ...e¥PhoneExperienceHost.exe    N/A      |
|    0   N/A  N/A     16012    C+G   ...IPanel¥SystemSettings.exe    N/A      |
|    0   N/A  N/A     17252    C+G   ...ge¥Application¥msedge.exe    N/A      |
|    0   N/A  N/A     17512    C+G   ...¥app-1.0.9007¥Discord.exe    N/A      |
|    0   N/A  N/A     19680    C+G   ...icrosoft VS Code¥Code.exe    N/A      |
```

$nvidia-smi 를 통해 tensorflow 가 gpu 로 학습되고 있는것을 확인

이번 프로젝트의 코드 순서입니다.

1. Train dataset 과 test dataset 불러오기
2. 0~1 사이값 조정하기 위해 255 나누기(one hot encoding)
3. Model sequential 을 통해 자기가 원하는 Model 설계
4. Model compile
5. Gpu device 가 있는지 확인
   있다면 GPU 로 학습시키기
   없다면 CPU 로 학습시키기
6. 모델 저장하기
7. 모델 불러오기
8. Model summary 출력하기
9. 모델을 test dataset 으로 평가하기

수행한 내용을 캡처 및 설명

Project 2-1

Project 2-1 에서 제시된 model 들을 sequential 안에 넣고, 엄태현 조교님의 텐서플로우 강의자료에서 사용된 model compile 을 이용했습니다.
Optimizer 는 'adam'을 사용했고, Loss 는 'sparse_categorical_crossentropy', 학습마다 accuracy 가 출력되게 했습니다. model.compile(optimizer = 'adam', loss = 'sparse_categorical_crossentropy', metrics=['accuracy'])
해당 Loss 를 사용하기 위해 수업시간에 배운 one hot encoding 을 했습니다.
X_train, x_test = x_train / 255.0, x_test / 255.0

학습시간은 한 epoch 당 10.7 초가 소요되었고, 총 학습시간은 약 50 초였습니다.
Test 는 1.2 초 정도 걸렸고 아래와 같은 결과가 나왔습니다.

```
313/313 [==============================] - 1s 2ms/step - loss: 0.9525 - accuracy: 0.6657
```

(추출된 model1.h5 파일을 test dataset 에 evaluate 한 결과)

Test accuracy 는 약 66.5%가 나온것을 확인할 수 있습니다.

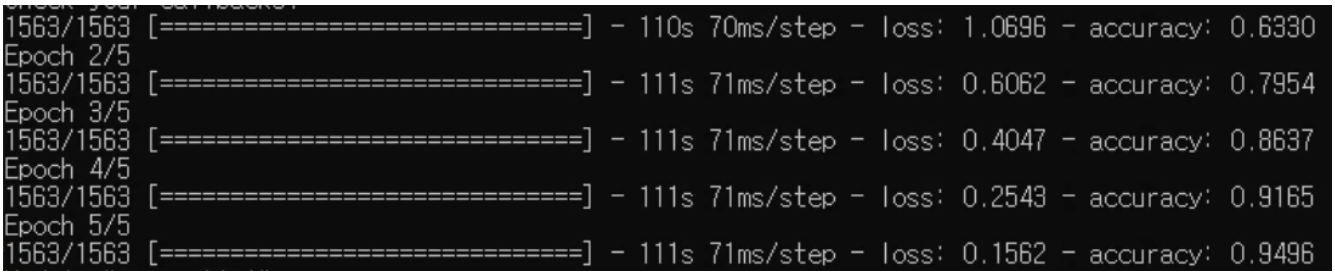
Project 2-2

Project 2-1 에서의 test 값이 생각보다 낮아서 많은 고민을 했습니다. 이것을 해결하기 위해서

1) 기존에 사용한 convolution 의 filter 갯수를 늘렸습니다. => filter 의 갯수를 늘림으로써 정확성을 높일 수 있음
2) 2-1 에서 Dense 에서 activation 을 relu 로 뒀었지만, layer 의 수를 낮추고, activation 함수를 'sigmoid'로 변경했습니다.
3) Dropout 의 수치를 0.001 로 낮췄습니다. => 기존에 많은양을 학습하지 않지만, dropout 의 수치가 생각보다 커서 학습이 제대로 되지않고 있다고 생각해서 dropout 의 수치를 많이 낮췄습니다.
4) 마지막 convolution, maxpooling 에서는 전에 사용했던것보다 더 많은 filter 와 Kernel_size 이기 때문에 padding 을 좀 더 의미있는 칸을 convolution 하고자 valid 를 주었습니다.
5) Batchnormalization 을 사용했습니다. => 학습 과정에서 각 배치 단위 별로 데이터가 다양한 분포를 가지더라도 각 배치별로 평균과 분산을 이용해 정규화함으로써 정확성을 조금 더 높일 수 있습니다.

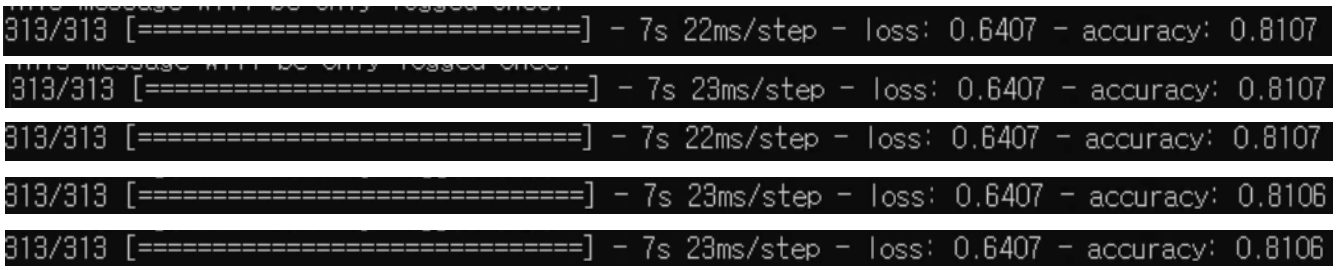
```
Model: "sequential_22"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_89 (Conv2D)           (None, 32, 32, 128)       3584
_________________________________________________________________
batch_normalization (BatchNo (None, 32, 32, 128)       512
_________________________________________________________________
conv2d_90 (Conv2D)           (None, 32, 32, 256)       295168
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 32, 256)       1024
_________________________________________________________________
max_pooling2d_44 (MaxPooling (None, 11, 11, 256)       0
_________________________________________________________________
batch_normalization_2 (Batch (None, 11, 11, 256)       1024
_________________________________________________________________
conv2d_91 (Conv2D)           (None, 11, 11, 512)       2097664
_________________________________________________________________
batch_normalization_3 (Batch (None, 11, 11, 512)       2048
_________________________________________________________________
conv2d_92 (Conv2D)           (None, 7, 7, 1024)        13108224
_________________________________________________________________
batch_normalization_4 (Batch (None, 7, 7, 1024)        4096
_________________________________________________________________
max_pooling2d_45 (MaxPooling (None, 1, 1, 1024)        0
_________________________________________________________________
batch_normalization_5 (Batch (None, 1, 1, 1024)        4096
_________________________________________________________________
flatten_22 (Flatten)         (None, 1024)              0
_________________________________________________________________
dense_46 (Dense)             (None, 60)                61500
_________________________________________________________________
dropout_22 (Dropout)         (None, 60)                0
_________________________________________________________________
dense_47 (Dense)             (None, 10)                610
=================================================================
Total params: 15,579,550
Trainable params: 15,573,150
Non-trainable params: 6,400
```

Model summary capture



Training process

학습시간은 한 epoch 당 약 111 초가 소요되었고, 약 9 분정도 소요가 되었습니다.



Capture of Top3 accuracy of My CNN model

추출된 model2.h5 를 바탕으로 5 번의 test 를 거쳤습니다. 81.07%가 공동 Top3 인것을 확인할 수 있었습니다.

Consideration

고찰 작성

이번 과제는 2 차프로젝트이지만 많은것을 느꼈습니다.

1. Improvement method

   1) filter 의 갯수를 늘림

   filter 의 갯수를 늘림으로써 좀 더 많은 연산을 통해 정확성을 높일것이라고 생각했습니다.

   2) dense 의 activation 을 relu 에서 sigmoid 로 교체 후, units 의 수를 줄임.

   sigmoid 보다 relu 를 더 많이 사용하는 추세라고 수업시간에 들었지만, relu 의 문제점인 입력값이 0 보다 작을 때, 함수 미분값이 0 이 되는 약점이 혹시 영향을 주었나 싶어서, sigmoid 함수를 적용을 했더니 더 좋은 결과가 나왔습니다.

   3) Batchnormalization 을 사용

   학습 과정에서 각 배치 단위 별로 데이터가 다양한 분포를 가지더라도 각 배치별로 평균과 분산을 이용해 정규화함으로써 정확성을 조금 더 높일 수 있습니다.

   4) Dropout 의 수치를 0.001 로 낮춤

   기존에 많은양을 학습하지 않지만, dropout 의 수치가 생각보다 커서 학습이 제대로 되지않고 있다고 생각해서 dropout 의 수치를 많이 낮췄습니다.

   5) 마지막 convolution, maxpooling 에서는 전에 사용했던것보다 더 많은 filter 와 Kernel_size 이기 때문에 padding 을 좀 더 의미있는 칸을 convolution 하고자 valid 를 주었습니다.

2. Problem solution

   이번 과제는 엄태현조교님께서 준비해주신 tensorflow 강의 자료와 tensorflow 공식홈페이지와 티스토리 글들을 이용해서 해결했습니다. 필터수, activation 함수를 바꿔가면서 정확도를 조금씩 올려가면서 기분이 좋았지만 처음에 80%는 넘지 못했습니다. 그래서 다시한번 프로젝트 제안서를 보면서 normalization 도 자유롭게 사용할 수 있다는 글을 읽은 후에, keras 에서 batchnormalization 이 있다는것을 알게되어 사용해보니 80%가 넘은것을 확인할 수 있었습니다. 결론을 짓자면, filter 의 수를 올리면 더 많은 연산을 그리고 normalization 을 통해 높은 정확성을 가졌지만, activation 함수는 적절한것을 찾아야 높아지기 때문에 이것저것 사용해가면서 알게 되었습니다.

3. 고찰

   평소에 게임할때 gpu 를 주로 사용했지만, 이러한 과제에서 직접 gpu 를 사용하면서 조금더 가치있게 사용한것 같아서 기분이 좋았습니다. keras 나 유명프레임워크들은 공식홈페이지가

있고, 그 공식홈페이지에서 사용법을 참고하면서 이게 좋은가 저게 좋은가를 생각하면서 써보면서 경험적인 측면도 중요시하면서 조금 더 공부할 수 있게 되는 계기가 되었습니다. 앞으로도 이러한 문제가 있다면 블로그글보다는 공식홈페이지에서 해결하는 습관을 갖게 되었습니다.

Reference

1) 인공지능 텐서플로우 실습자료(엄태현조교님)
2) Keras 공식 홈페이지
   https://keras.io/ko/

3) Cifar-10
   https://www.cs.toronto.edu/~kriz/cifar.html
4) Padding 관련자료
   https://wandb.ai/krishamehta/seo/reports/Difference-Between-SAME-and-VALID-Padding-in-TensorFlow--VmlldzoxODkwMzE
5) Dropout 관련자료
   https://heytech.tistory.com/127
6) Activation 함수
   https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=handuelly&logNo=221824080339
7) Batch normalization
   https://gaussian37.github.io/dl-concept-batchnorm/#batch-normalization-1