

환경 설정하는 데에 시간을 많이 썼다. 7 장까지는 cpu 만 있어도 프로그램이 잘 돌아갔다. 프로그램 8-7 을 수행하는 것부터 쉽지 않았다. Epoch 1에서만 무한 로딩되는 느낌이었다. CPU와 메모리 활용률은 거의 100%였고, 외장 그래픽카드가 달린 노트북에서도 GPU 활용률이 10% 아래였다. 그래서 외장 GPU가 달린 노트북에서 gpu를 활성화시키려 했다. 하지만, 이 부분에서 계속 오류가 발생했다. 버전 호환성 문제 때문이었던 것 같았지만, 프로그램(anaconda, cuda, cudnn)을 몇 번을 재설치하고 tensorflow-gpu에 대해서 찾아보고 시도해도 해결되지 않았다.

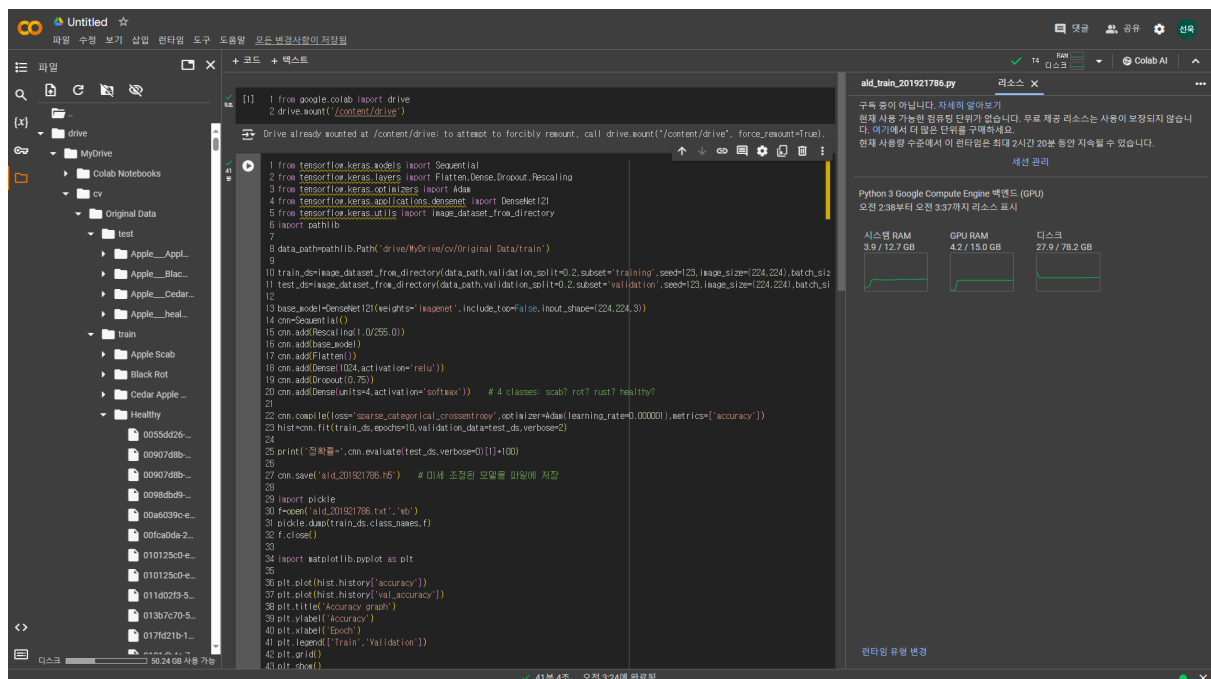
Colab을 사용하면 편리하다고 해서 결국 colab을 썼다. 프로그램 8-7을 돌려보고 싶었지만, dataset을 구글 드라이브에 동기화하는 게 오래 걸려서 그냥 바로 과제를 수행했다. colab에서 새 노트를 만들고 구글 드라이브와 연동했다. 코드는 8-7과 다른 부분은 별로 없다. 데이터 경로 부분, CNN의 마지막 layer를 추가하는 부분에서 4개(Apple leaf 상태가 총 4개 부류(apple scab, black rot, cedar apple rust, healthy)로 분류되기 때문이다) 노드로 출력되게 하는 부분, 그리고 마지막으로 학습을 할 때 epoch을 200이 아닌 10개만 반복하는 부분이다.

프로그램 8-7 설명과 다르게 epoch #1에서만 시간이 엄청 오래 걸리고 epoch #2부터는 1분 정도밖에 걸리지 않았다. 백본모델을 DenseNet121이 아닌 다른 걸(ex. VGG16) 쓰거나 CNN layer의 구조를 바꾸는 게 맞는 건지, 이렇게 첫 epoch에서만 시간이 오래 걸려도 괜찮은지는 더 알아봐야 할 것 같다. 정상적인 현상은 아닌 것 같다.

비전 에이전트를 만드는 건 실패했다. 모델 학습도 잘 했고, 모델 저장도 잘 한 것 같은데, 오류가 많이 발생했다. 위 내용처럼 백본모델이나 신경망의 층 구조를 바꿔야 하는 건지 아니면 이와 별개의 문제인지 감이 잘 안 잡혔다. 오류 초반부에 “conv1d/conv”가 적힌 걸 보면 conv2d여야 하는데 conv1라서 오류가 발생한 것 같기도 했다. 오류가 하나가 아니어서 모델을 학습하는 부분부터 천천히 살펴봐야 오류를 해결할 수 있을 것 같다. 따라서 GUI는 이름만 조금 수정했을 뿐 개선사항은 없다.

마지막으로 외장 GPU가 달린 노트북에서 gpu를 활성화시키는 게 편할 것 같다. Colab은 세션 만료 때문에 불편했고, 사실 시행착오를 많이 겪으면서 구글 계정을 3개를 돌려썼다.

아래는 colab에서 과제를 수행한 결과이다.



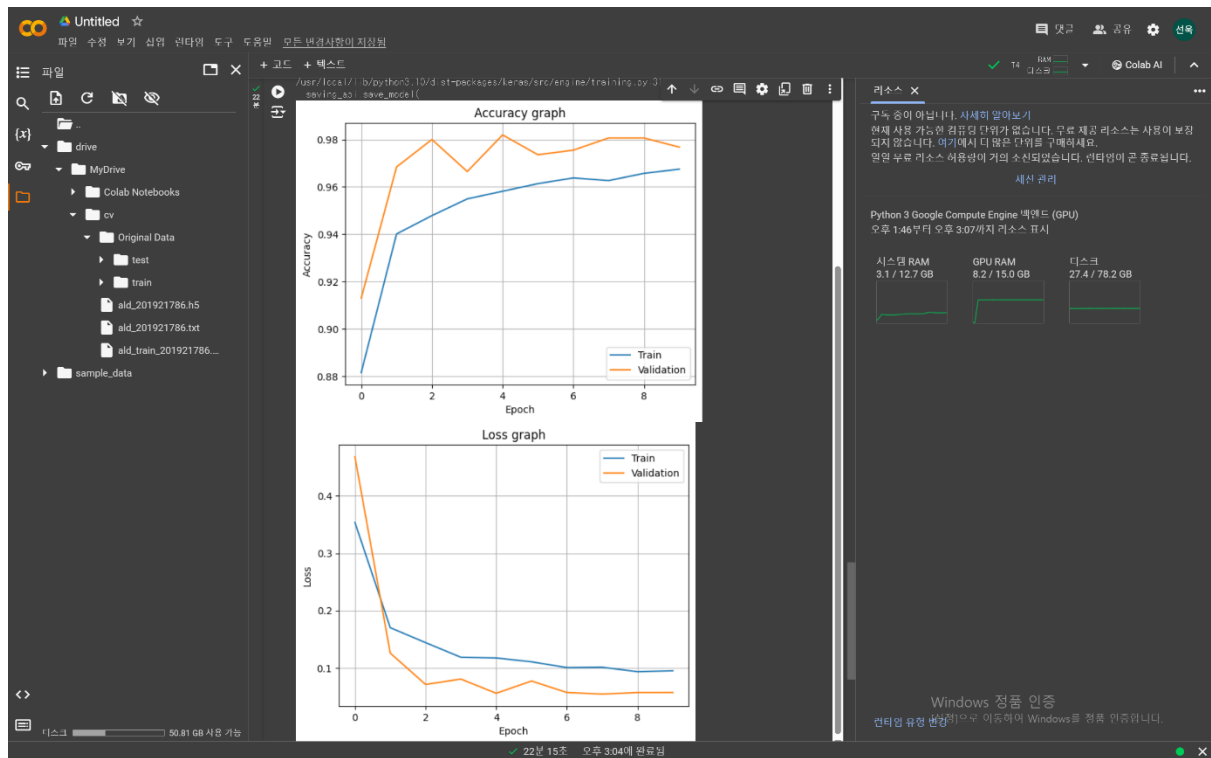
The screenshot displays a Google Colab notebook interface. The left sidebar shows the file explorer with a folder named 'train' containing several files. The main area shows a Python script titled 'old_train_201921786.py'. The script imports various libraries and defines a data path. It then loads a dataset and trains a DenseNet121 model. The output shows the training progress and accuracy.

```
1 from google.colab import drive
2 drive.mount('/content/drive')

3
4
5
6
7
8 data_path=pathlib.Path('/content/drive/MyDrive/cv/Original Data/train')
9
10 train_data_loader=train_data_loader_validation_split(0.2, subset='training', seed=23, image_size=(224, 224), batch_size=12)
11 test_data_loader=test_data_loader_validation_split(0.2, subset='validation', seed=23, image_size=(224, 224), batch_size=12)
12
13 base_model=DenseNet121(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
14 cnn=Sequential()
15 cnn.add(base_model)
16 cnn.add(Dense(1024, activation='relu'))
17 cnn.add(Dense(1024, activation='relu'))
18 cnn.add(Dense(1024, activation='relu'))
19 cnn.add(Dense(1024, activation='relu'))
20 cnn.compile(loss='sparse_categorical_crossentropy', optimizer='adam', learning_rate=0.00001, metrics=['accuracy'])
21
22 hist=train_data_loader.fit(train_data_loader, validation_data_loader, verbose=2)
23
24 print('정확률:', cnn.evaluate(test_data_loader, verbose=0)[1]-100)
25
26
27 cnn.save('aid_201921786.h5') # 이세 조영의 모델을 파일에 저장
28
29 import pickle
30 f=open('aid_201921786.txt', 'w')
31 pickle.dump(train_data_loader.class_names, f)
32 f.close()
33
34 import matplotlib.pyplot as plt
35
36 plt.plot(hist.history['accuracy'])
37 plt.plot(hist.history['val_accuracy'])
38 plt.title('Accuracy graph')
39 plt.xlabel('Epoch')
40 plt.ylabel('Accuracy')
41 plt.legend(['train', 'validation'])
42 plt.grid()
43 plt.show()
```

The right sidebar shows the 'Runtime' tab with a table of system resources:

시스템 RAM	GPU RAM	디스크
3.9 / 12.7 GB	4.2 / 15.0 GB	27.9 / 78.2 GB



```

1 import cv2 as cv
2 import numpy as np
3 import tensorflow as tf
4 import winsound
5 import pickle
6 import sys
7 from PyQt5.QtWidgets import *
8
9 cnn=tf.keras.models.load_model('old_201921786.h5') # 보일 읽기
10 dog_species=pickle.load(open('old_201921786.txt','rb')) # 글자 읽기
11
12 class AppleLeafDiseaseRecognition(QMainWindow):
13     def __init__(self):
14         super().__init__()
15         self.setWindowTitle("사과 나뭇잎 질병 인식")
16         self.setGeometry(200,200,700,100)
17
18         fileButton=QPushButton("나뭇잎 사진 열기",self)
19         recognitionButton=QPushButton("인식",self)
20         quitButton=QPushButton("나가기",self)
21
22         fileButton.setGeometry(10,10,100,30)
23         recognitionButton.setGeometry(110,10,100,30)
24         quitButton.setGeometry(510,10,100,30)
25
26         fileButton.clicked.connect(self.pictureOpenFunction)
27         recognitionButton.clicked.connect(self.recognitionFunction)
28         quitButton.clicked.connect(self.quitFunction)
29
30     def pictureOpenFunction(self):
31         fileName=QFileDialog.getOpenFileName(self,"나뭇잎 사진 열기",".",")
32         self.img=cv.imread(fileName)
33         if self.img is None: sys.exit("파일을 찾을 수 없습니다.")
34
35         cv.imshow('Apple Leaf image',self.img)
36
37     def recognitionFunction(self):
38         x=np.reshape(self.img,(224,224),(1,224,224,3))
39         res=cnn.predict(x)[0] # 결과
40         top5=np.argsort(-res)[:5]
41         top5_dog_species_names=[dog_species[i] for i in top5]
42         for i in range(5):
43             probe = "%s(%s[top5[%d]])" % (
44                 namestr(top5_dog_species_names[i]).split('.')[1],
45                 cv.putText(self.img,probe+name,(10,100+1*30),cv.FONT_HERSHEY_SIMPLEX,0.7,(255,255,255),2)
46             cv.imshow('Apple Leaf image',self.img)
47             winsound.Beep(1000,500)
48
49     def quitFunction(self):
50         cv.destroyAllWindows()
51         self.close()
52
53 app=QApplication(sys.argv)
54 win=AppleLeafDiseaseRecognition()
55 win.show()
56 app.exec_()

```

File: ~\anaconda3\envs\cv\Lib\site-packages\keras\src\layers\reshaping\flatten.py:72 in compute_output_spec

```

output_shape = self.compute_output_shape(inputs.shape)
AttributeError: Exception encountered when calling Flatten.call().
'list' object has no attribute 'shape'
Arguments received by Flatten.call():
  * args=(['KerasTensor shape=(None, 512), dtype=float32, sparse=False, name=KerasTensor_63'],)
  * kwargs={'class': 'inspect_empty'}

```

Windows 정품 인증
(선택)으로 이동하여 Windows를 정품 인증합니다.