컴퓨터 그래픽스 과제 #11

학과 : 컴퓨터공학과
학번 : 201802161
이름 : 조은빈

1. 구현 코드

```
[1]  import tensorflow as tf
     import cv2
     import numpy as np
     from skimage import transform
```

- Fully Connected Layer 하나만 학습

```
# Fully Connected Layer 하나만 학습
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar100.load_data()
resized_xtrain = np.zeros((1350,224,224,3))
for i in range (1350):
    resized_xtrain[i] = transform.resize(x_train[i], (224, 224, 3), order=1, mode='reflect')
resized_xtest = np.zeros((150,224,224,3))
for i in range (150):
    resized_xtest[i] = transform.resize(x_test[i], (224, 224, 3), order=1, mode='reflect')
resized_ytrain = np.zeros((1350,1))
for i in range (1350):
    resized_ytrain[i] = transform.resize(y_train[i], (1, ), order=1, mode='reflect')
resized_ytest = np.zeros((150,1))
for i in range (150):
    resized_ytest[i] = transform.resize(y_test[i], (1, ), order=1, mode='reflect')


base_model = tf.keras.applications.ResNet50(weights='imagenet', input_shape=(224,224,3))
# base_model.summary()
base_model = tf.keras.models.Model(base_model.inputs, base_model.layers[-2].output)

base_model.trainable = False # 이미 학습 되어있으므로 추가 학습 시키지 않는다
x = base_model.output
pred = tf.keras.layers.Dense(100, activation='softmax')(x)
model = tf.keras.models.Model(inputs=base_model.input, outputs=pred)
```

```
opt = tf.keras.optimizers.Adam(learning_rate=0.0001)

model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['acc'])

from sklearn.model_selection import train_test_split

resized_xtrain, x_valid, resized_ytrain, y_valid = train_test_split(resized_xtrain, resized_ytrain, test_size=0.2, random_state=123)

resized_ytrain = tf.squeeze(tf.one_hot(resized_ytrain, 100), axis=1)
y_valid = tf.squeeze(tf.one_hot(y_valid, 100), axis=1)
resized_ytest = tf.squeeze(tf.one_hot(resized_ytest, 100), axis=1)

history = model.fit(x=resized_xtrain, y=resized_ytrain, batch_size=32, epochs=20, validation_data=(x_valid, y_valid))

import matplotlib.pyplot as plt

plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('201802161 model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'valid'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('201802161 model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'valid'], loc='upper left')
plt.show()

results = model.evaluate(resized_xtest, resized_ytest, batch_size=32)

print('test accuracy')
print(results[1])
```

- 일부분을 다시 학습

```
# 일부분을 다시 학습

(x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar100.load_data()
resized_xtrain = np.zeros((1350,224,224,3))
for i in range (1350):
    resized_xtrain[i] = transform.resize(x_train[i], (224, 224, 3), order=1, mode='reflect')
resized_xtest = np.zeros((150,224,224,3))
for i in range (150):
    resized_xtest[i] = transform.resize(x_test[i], (224, 224, 3), order=1, mode='reflect')
resized_ytrain = np.zeros((1350,1))
for i in range (1350):
    resized_ytrain[i] = transform.resize(y_train[i], (1, ), order=1, mode='reflect')
resized_ytest = np.zeros((150,1))
for i in range (150):
    resized_ytest[i] = transform.resize(y_test[i], (1, ), order=1, mode='reflect')


base_model = tf.keras.applications.ResNet50(weights='imagenet', input_shape=(224,224,3))
# base_model.summary()
base_model = tf.keras.models.Model(base_model.inputs, base_model.layers[-2].output)

x = base_model.output
pred = tf.keras.layers.Dense(100, activation='softmax')(x)
model = tf.keras.models.Model(inputs=base_model.input, outputs=pred)
```

```
for layer in model.layers:
  layer.trainable = True

for idx, layer in enumerate(model.layers):
  print(idx, layer)
  if idx < 100:
    layer.trainable = False

opt = tf.keras.optimizers.Adam(learning_rate=0.0001)

model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['acc'])

from sklearn.model_selection import train_test_split

resized_xtrain, x_valid, resized_ytrain, y_valid = train_test_split(resized_xtrain, resized_ytrain, test_size=0.2, random_state=123)

resized_ytrain = tf.squeeze(tf.one_hot(resized_ytrain, 100), axis=1)
y_valid = tf.squeeze(tf.one_hot(y_valid, 100), axis=1)
resized_ytest = tf.squeeze(tf.one_hot(resized_ytest, 100), axis=1)

results = model.evaluate(resized_xtest, resized_ytest, batch_size=32)

print('test accuracy')
print(results[1])

history = model.fit(x=resized_xtrain, y=resized_ytrain, batch_size=32, epochs=20, validation_data=(x_valid, y_valid))
```

- 전체를 다시 학습

```
# 전체를 다시 학습
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar100.load_data()
resized_xtrain = np.zeros((1350,224,224,3))
for i in range (1350):
    resized_xtrain[i] = transform.resize(x_train[i], (224, 224, 3), order=1, mode='reflect')
resized_xtest = np.zeros((150,224,224,3))
for i in range (150):
    resized_xtest[i] = transform.resize(x_test[i], (224, 224, 3), order=1, mode='reflect')
resized_ytrain = np.zeros((1350,1))
for i in range (1350):
    resized_ytrain[i] = transform.resize(y_train[i], (1, ), order=1, mode='reflect')
resized_ytest = np.zeros((150,1))
for i in range (150):
    resized_ytest[i] = transform.resize(y_test[i], (1, ), order=1, mode='reflect')


base_model = tf.keras.applications.ResNet50(weights='imagenet', input_shape=(224,224,3))
# base_model.summary()
base_model = tf.keras.models.Model(base_model.inputs, base_model.layers[-2].output)

x = base_model.output
pred = tf.keras.layers.Dense(100, activation='softmax')(x)
model = tf.keras.models.Model(inputs=base_model.input, outputs=pred)

opt = tf.keras.optimizers.Adam(learning_rate=0.0001)

model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['acc'])
```

```
from sklearn.model_selection import train_test_split

resized_xtrain, x_valid, resized_ytrain, y_valid = train_test_split(resized_xtrain, resized_ytrain, test_size=0.2, random_state=123)

resized_ytrain = tf.squeeze(tf.one_hot(resized_ytrain, 100), axis=1)
y_valid = tf.squeeze(tf.one_hot(y_valid, 100), axis=1)
resized_ytest = tf.squeeze(tf.one_hot(resized_ytest, 100), axis=1)

results = model.evaluate(resized_xtest, resized_ytest, batch_size=32)

print('test accuracy')
print(results[1])

history = model.fit(x=resized_xtrain, y=resized_ytrain, batch_size=32, epochs=20, validation_data=(x_valid, y_valid))
```

2. 코드 설명

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar100.load_data()
resized_xtrain = np.zeros((1350,224,224,3))
for i in range (1350):
    resized_xtrain[i] = transform.resize(x_train[i], (224, 224, 3), order=1, mode='reflect')
resized_xtest = np.zeros((150,224,224,3))
for i in range (150):
    resized_xtest[i] = transform.resize(x_test[i], (224, 224, 3), order=1, mode='reflect')
resized_ytrain = np.zeros((1350,1))
for i in range (1350):
    resized_ytrain[i] = transform.resize(y_train[i], (1, ), order=1, mode='reflect')
resized_ytest = np.zeros((150,1))
for i in range (150):
    resized_ytest[i] = transform.resize(y_test[i], (1, ), order=1, mode='reflect')
```

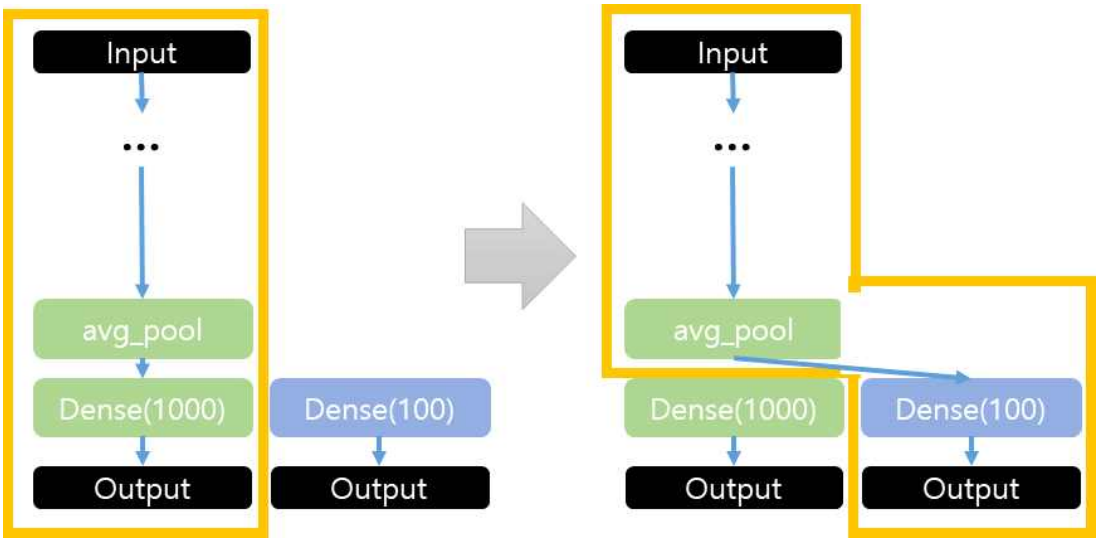
이번 과제에선 ResNet50 모델을 imageNet 데이터셋으로 미리 학습 시킨 다음 cifar100 데이터셋으로 추가적인 학습을 진행하였다. imageNet 데이터셋의 size는 224x224x3이므로 imageNet으로 학습시킨 모델의 입력 이미지 size는 224x224x3여야 한다. cifar100의 이미지 size는 32x32x3이므로 224x224x3로 바꿔준다. 이때, cifar100은 총 60,000개의 이미지가 있는데 이를 전부 224x224x3으로 바꿔주면 메모리 사용량이 너무 많아 colab이 멈춘다. train 이미지를 1만 개만 224x224x3으로 바꿔도 colab이 멈춰버렸기 때문에 어쩔 수 없이 colab이 작동하는 선까지 줄여서 train 데이터 1350개, test 데이터 150개로 데이터셋을 구성했다.

```
base_model = tf.keras.applications.ResNet50(weights='imagenet', input_shape=(224,224,3))
```

현재 base_model은 imageNet으로 학습되어 1000개의 클래스를 구분할 수 있다. cifar100은 100개의 클래스를 구분해야한다.

```
base_model = tf.keras.models.Model(base_model.inputs, base_model.layers[-2].output)

base_model.trainable = False # 이미 학습 되어있으므로 추가 학습 시키지 않는다
x = base_model.output
pred = tf.keras.layers.Dense(100, activation='softmax')(x)
model = tf.keras.models.Model(inputs=base_model.input, outputs=pred)
```

따라서 위 코드를 통해 base_model에서 마지막 레이어인 Fully Connected Layer를 떼어낸다. base_model은 이미 학습된 상태이므로 trainable을 false로 설정해 더 학습시키지 않는다. 여기에 Dense가 100인 Fully Connected Layer를 접합한다. base_model은 Fully Connected Layer만 학습할 수 있는 상태가 된다.



왼쪽 이미지가 imageNet으로 학습시킨 base_model 모형이다. 오른쪽 이미지가 기존 Fully Connected Layer를 떼어내고 다른 Fully Connected Layer를 접합시킨 base_model 모형이다. base_model을 학습시키고 결과를 낸다.

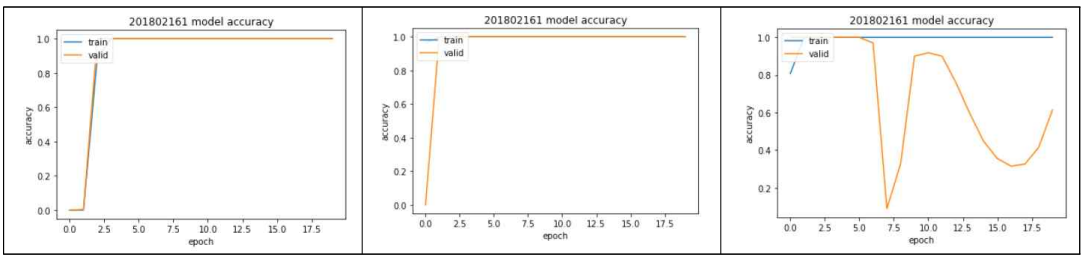
```
for layer in model.layers:
  layer.trainable = True

for idx, layer in enumerate(model.layers):
  print(idx, layer)
  if idx < 100:
    layer.trainable = False
```

일부분을 학습하는 코드에는 위 코드가 추가되었다. 위 코드는 model에서 100개의 레이어를 제외한 나머지 레이어는 학습하도록 설정하는 코드이다.

```
# 전체를 다시 학습
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar100.load_data()
resized_xtrain = np.zeros((1350,224,224,3))
for i in range (1350):
    resized_xtrain[i] = transform.resize(x_train[i], (224, 224, 3), order=1, mode='reflect')
resized_xtest = np.zeros((150,224,224,3))
for i in range (150):
    resized_xtest[i] = transform.resize(x_test[i], (224, 224, 3), order=1, mode='reflect')
resized_ytrain = np.zeros((1350,1))
for i in range (1350):
    resized_ytrain[i] = transform.resize(y_train[i], (1, ), order=1, mode='reflect')
resized_ytest = np.zeros((150,1))
for i in range (150):
    resized_ytest[i] = transform.resize(y_test[i], (1, ), order=1, mode='reflect')


base_model = tf.keras.applications.ResNet50(weights='imagenet', input_shape=(224,224,3))
# base_model.summary()
base_model = tf.keras.models.Model(base_model.inputs, base_model.layers[-2].output)

x = base_model.output
pred = tf.keras.layers.Dense(100, activation='softmax')(x)
model = tf.keras.models.Model(inputs=base_model.input, outputs=pred)

opt = tf.keras.optimizers.Adam(learning_rate=0.0001)

model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['acc'])
```

전체를 다시 학습하는 코드에는 trainable을 설정하는 부분이 없다. 모델의 모든 레이어를 다시 학습하기 때문이다.

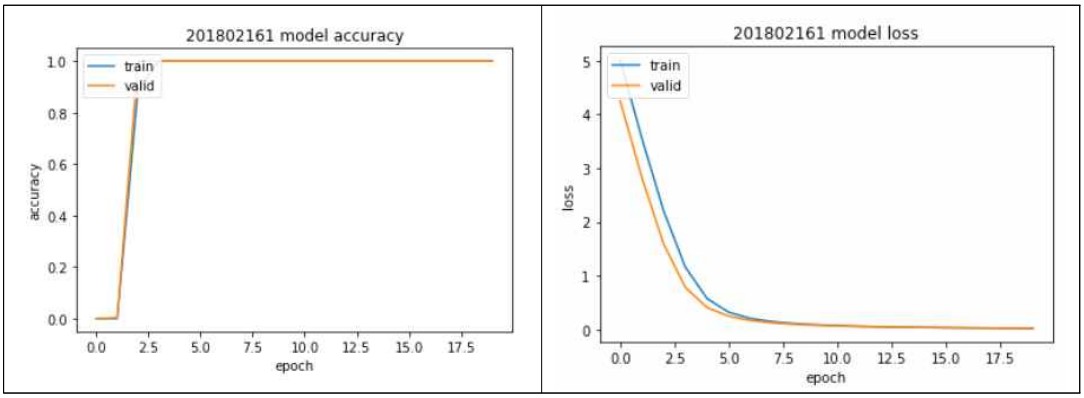
결과를 비교해보겠다. 왼쪽부터 Fully Connected Layer 하나만 학습한 경우, 일부만 다시 학습한 경우, 전체를 다시 학습한 경우이다. Fully Connected Layer 하나만 학습한 경우와 일부만 다시 학습한 경우는 둘 다 1.0으로 매우 높은 정확도가 나와서 어느 것이 더 좋다고 판별하기 어려웠다. 반면 전체를 다시 학습한 경우는 약 0.6으로 상대적으로 낮은 경향을 보였다.



3. 이미지

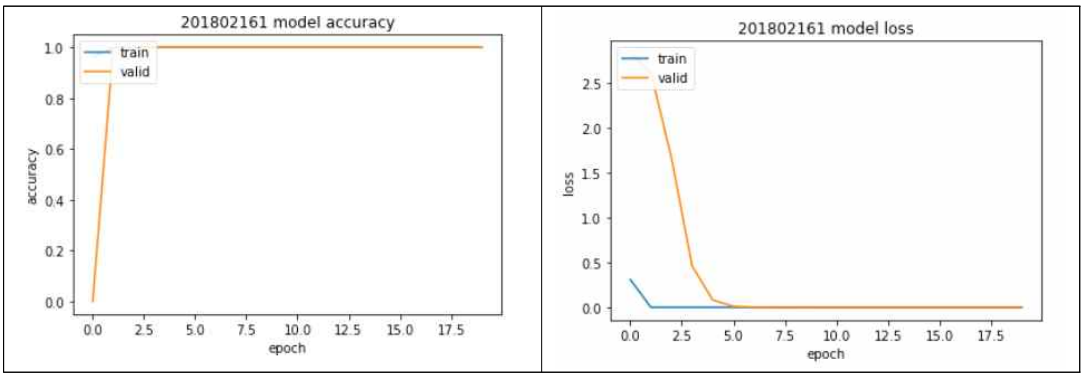
- Fully Connected Layer 하나만 학습

```
5/5 [==============================] - 1s 132ms/step - loss: 0.0213 - acc: 1.0000
test accuracy
1.0
```



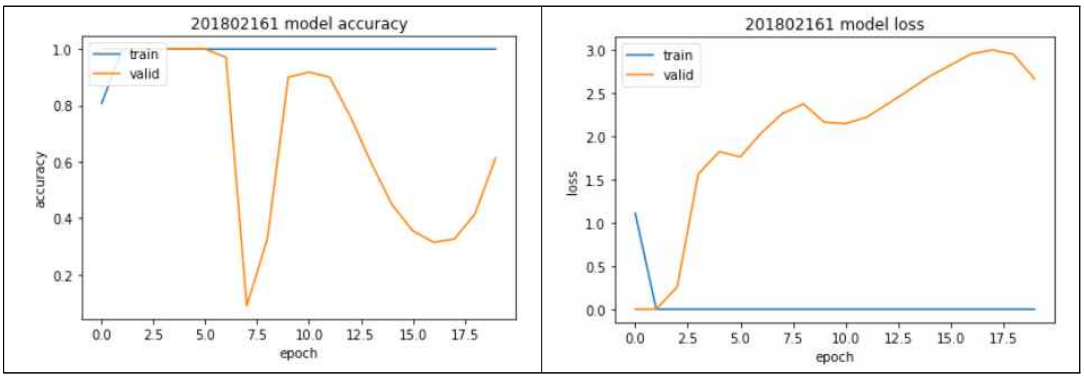
- 일부분을 다시 학습

```
5/5 [==============================] - 0s 74ms/step - loss: 7.3981e-06 - acc: 1.0000
test accuracy
1.0
```



- 전체를 다시 학습

```
5/5 [==============================] - 0s 72ms/step - loss: 2.4801 - acc: 0.6533
test accuracy
0.653333306312561
```



4. 느낀점

처음에 train 이미지 개수를 1000개, test 데이터를 400개로 주었는데 학습이 전혀 되지 않아 많이 헤맸다. 튜터님께 여쭤보니 보통 train:test 비율은 9:1 정도가 적당하다는 답변

을 받았다. train 데이터와 test 데이터의 비율도 학습에 큰 영향을 끼친다는 것을 알았다.
5. 과제 난이도
cifar100 데이터 사이즈를 줄이는 부분이 조금 어려웠다.