과제 #3

Building an Image Classifier Using CNN

Basic codes are given via LMS

과제 #3

Original

```
model = keras.models.Sequential([
    DefaultConv2D(filters=32, kernel_size=3, input_shape=[28, 28, 1]),
    keras.layers.MaxPooling2D(pool_size=2),
    DefaultConv2D(filters=64),
    keras.layers.MaxPooling2D(pool_size=2),
    DefaultConv2D(filters=128),
    keras.layers.Flatten(),
    keras.layers.Dense(units=128, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(units=10, activation='softmax'),
])
```

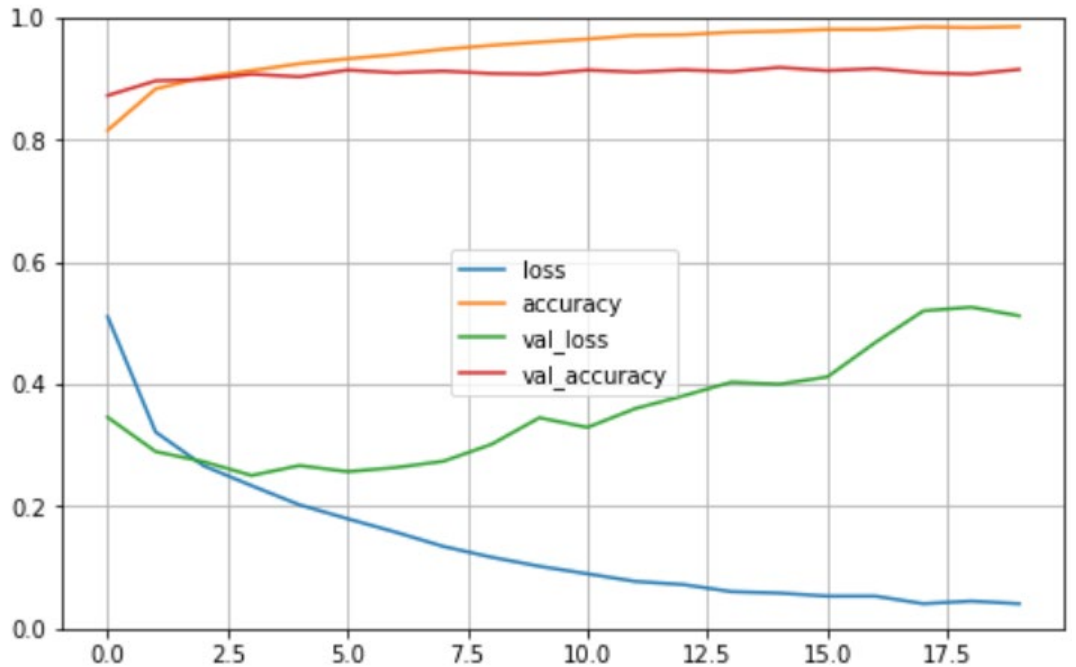
```
org_acc = model.evaluate(X_test, y_test)
print(org_acc[1]*100)
```

```
313/313 [==============================] - 1s
91.57000184059143
```

```
[10]  from keras import backend as K

      org_model_size = np.sum([K.count_params(w)
      print(org_model_size)
```

```
896906
```



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 32)	320
max_pooling2d (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_1 (Conv2D)	(None, 14, 14, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 7, 7, 64)	0
conv2d_2 (Conv2D)	(None, 7, 7, 128)	73856
flatten (Flatten)	(None, 6272)	0
dense (Dense)	(None, 128)	802944
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1290

Total params: 896,906
Trainable params: 896,906
Non-trainable params: 0

```
from keras import backend as K

org_model_size = np.sum([K.count_params(w) for w in model.trainable_weights])
print(org_model_size)
```

```
896906
```

과제 #3

- Submit a report that contains
  - Your model that has **less trainable parameters** than original model (baseline)
  - The result of evaluation: capture the result
  - Capture **model summary**
  - Capture the **exact parameters** used to train the model

과제 #3

```
1
2 print("[Acc] performance improvement: %.2f percent" % (yours[1]*100 - org_acc[1]*100))
3 print("[Size] size ratio: %.2f percent" % ((your_model_size / org_model_size)*100) )
4 if (yours[1]*100 - org_acc[1]*100) > 0:
5   print("Accuracy resolved")
6 if (your_model_size / org_model_size)*100 < 100:
7   print("Size resolved")
```

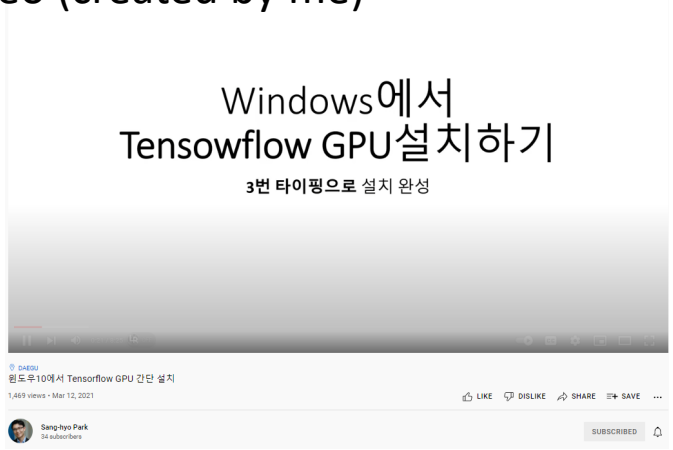
```
[Acc] performance improvement: 0.91 percent
[Size] size ratio: 57.75 percent
Accuracy resolved
Size resolved
```

MUST: More than 0 percent

MUST: Less than 70 percent

과제 #3

- NOTE:
  - Totally **two** models should be reported in one code (or notebook).
  - Submit the report in **PDF or MS word** format via **LMS with code (or notebook)!**
  - Deadline is three weeks later (See the exact timeline in LMS).
  - Interface:
    - Jupyter notebook is recommended to see the result but is not the mandatory.
    - Library: Keras is highly recommended for beginners. But when you use others such as PyTorch, then you must specify the library name as well as the version number.
    - If you want to install GPU-version Tensorflow, watch the 8 min video (created by me)
      - https://www.youtube.com/watch?v=M4urbN0fPyM&t=254s



과제 #3

- NOTE:
  - It may take more than minutes (for **CPU-version** TensorFlow).
    - So, take a rest for minutes.

```
history = model.fit(X_train, y_train, epochs=30,
                    batch_size=16,
                    validation_data=(X_valid, y_valid))
```

```
Epoch 1/30
3438/3438 [==============================] - 45s 13ms/step - loss: 0.4489 - accuracy: 0.8366 - val_loss: 0.3082 - val_accuracy: 0.8834
Epoch 2/30
3438/3438 [==============================] - 44s 13ms/step - loss: 0.2860 - accuracy: 0.8949 - val_loss: 0.2663 - val_accuracy: 0.9006
Epoch 3/30
3438/3438 [==============================] - 43s 12ms/step - loss: 0.2460 - accuracy: 0.9102 - val_loss: 0.2385 - val_accuracy: 0.9092
Epoch 4/30
3438/3438 [==============================] - 47s 14ms/step - loss: 0.2147 - accuracy: 0.9204 - val_loss: 0.2425 - val_accuracy: 0.9138
Epoch 5/30
3438/3438 [==============================] - 44s 13ms/step - loss: 0.1924 - accuracy: 0.9290 - val_loss: 0.2079 - val_accuracy: 0.9242
Epoch 6/30
```