Google Developer Student Clubs

Week8 Presentation

Brain tumors 256x256
Performance improvements

GDSC Hanyang ML/DL Basic : 이재승, 서지현

Index

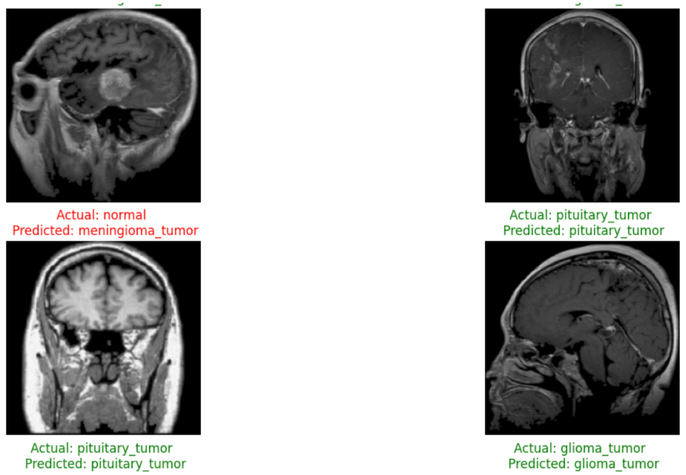
Performance improvements

- Remind
- Trial
  - Simple Changes
  - Data augmentation
  - Hyperparameter tuning
  - Model change
    - ResNet50V2, VGG, ResNet101, …

Remind

Performance Improvements

- Brain tumors 256x256
  - 뇌의 3가지 종양 (glioma, meningioma, pituitary) 찾기
  - 문제의 특성상, Accuracy보다 Recall 점수가 더 중요
    - 암인데 암이지 않다고 판단하는 경우 => 사고 발생 !
  - Base code
    - ResNet50 + Average pooling => Flatten => Dense Layer
    - Base code result : Accuracy - 89%, Recall - 90%



Predicted Results

Trial - Simple changes

Performance Improvements

Basecode :

Acc : 89%, Recall : 90%

- Simple changes
  - Dense Layer 갯수 증가 시키기
    - Hidden layer 증가 => 학습 결과 증진 예측
  - Average Pooling => Max pooling
    - 어떤 것이 더 나은지 궁금해서 해보았음
  - **Batch normalization**
    - 찾아봤던 Code에 적용되어 있었음 => 좋은 결과 !

```
x=tf.keras.layers.Flatten()(x)
x=tf.keras.layers.Dense(512, activation="relu")(x)
x=tf.keras.layers.Dropout(0.5)(x)
x=tf.keras.layers.Dense(512, activation="relu")(x)
x=tf.keras.layers.Dropout(0.5)(x)
x=tf.keras.layers.Dense(512, activation="relu")(x)
x=tf.keras.layers.Dropout(0.5)(x)

# Create the output activation layer
outputs=tf.keras.layers.Dense(4, activation="softmax",name="output_layer")(x)
```

Dense layer 추가 : Accuracy 68%, Recall 68%

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_layer (InputLayer)    [(None, 224, 224, 3)]     0

 resnet50 (Functional)       (None, None, None, 2048   23587712
                             )

 global_max_pooling_layer (  (None, 2048)              0
 GlobalMaxPooling2D)

 flatten_3 (Flatten)         (None, 2048)              0
```

Average -> Max pooling
Accuracy : 83%, Recall 83%

Trial - Data Augmentation

Performance Improvements

Basecode :

Acc : 89%, Recall : 90%

- Data Augmentation by Keras API
  - 사용한 것
    - Random(Flip, Rotation, Contrast)
    - Rescaling(1/255)
  - 사용 결과
    - Accuracy : 89%, Recall : 90%
    - 결과는 비슷하나 Loss가 굉장히 낮아지는 결과

```python
data_augmentation = tf.keras.Sequential(
    [
        tf.keras.layers.experimental.preprocessing.RandomFlip("horizontal"),
        tf.keras.layers.experimental.preprocessing.RandomRotation(0.1),
        tf.keras.layers.experimental.preprocessing.RandomContrast(0.1),
    ]
)

# Create inputs into the base model
inputs = tf.keras.layers.Input(shape=(224, 224,3), name="input_layer")
x = data_augmentation(inputs)
x = tf.keras.layers.experimental.preprocessing.Rescaling(1./255)(x)
x = base_model(inputs)
```

```
F1-Score:[0.8927986806989086]
Precision:[0.8961247755788683]
Recall:[0.889773881213025]
Classification_Report
----------------------
              precision    recall  f1-score   support

           0       0.88      0.88      0.88       184
           1       0.84      0.83      0.83       179
           2       0.92      0.96      0.94        81
           3       0.92      0.91      0.92       175

    accuracy                           0.89       619
   macro avg       0.89      0.90      0.89       619
weighted avg       0.89      0.89      0.89       619
```

Trial - Batch normalization + aug

Performance Improvements

Basecode :

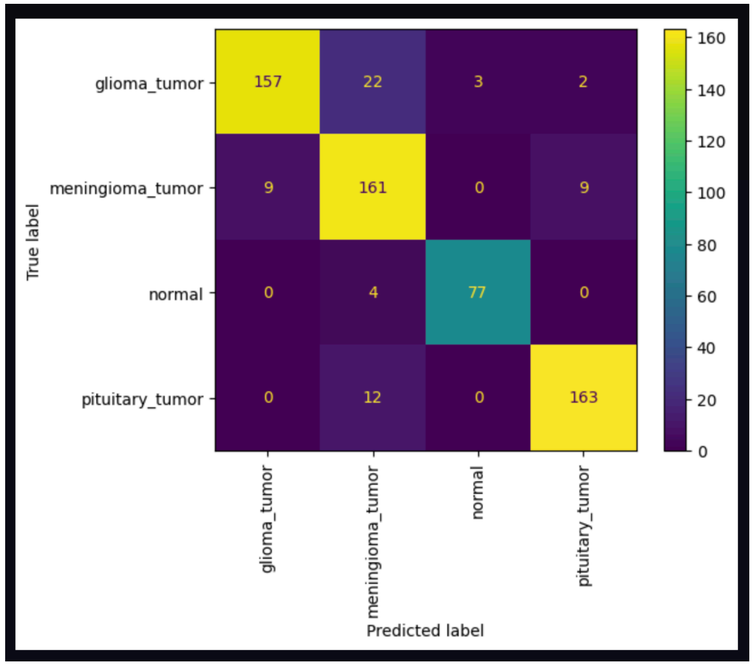
Acc : 89%, Recall : 90%

- Batch normalization
  - Batch normalization layer를 넣어 모델을 학습 : **Acc - 91%, Recall - 91%** 개선 !

```
F1-Score:[0.9099034673321454]
Precision:[0.9086870164316323]
Recall:[0.913527491964044]
Classification_Report
----------------------
              precision    recall  f1-score   support

           0       0.95      0.85      0.90       184
           1       0.81      0.90      0.85       179
           2       0.96      0.95      0.96        81
           3       0.94      0.93      0.93       175

    accuracy                           0.90       619
   macro avg       0.91      0.91      0.91       619
weighted avg       0.91      0.90      0.90       619
```



Trial - Hyperparameter tuning

Performance Improvements

Basecode :

Acc : 89%, Recall : 90%

- Hyperparameter tuning (with batch norm, aug)
  - 더 정밀한 결과를 위해 Batch size : 32 -> 16으로 줄임 : Acc - 90%, Recall - 91%

```
F1-Score:[0.899373586440259]
Precision:[0.9054023376638953]
Recall:[0.8942214642615909]
Classification_Report
----------------------
              precision    recall  f1-score   support

           0       0.87      0.86      0.87       184
           1       0.85      0.83      0.84       179
           2       0.91      0.99      0.95        81
           3       0.95      0.94      0.95       175

    accuracy                           0.89       619
   macro avg       0.89      0.91      0.90       619
weighted avg       0.89      0.89      0.89       619
```

Trial - ResNet50V2

Performance Improvements

Basecode :

Acc : 91%, Recall : 92%

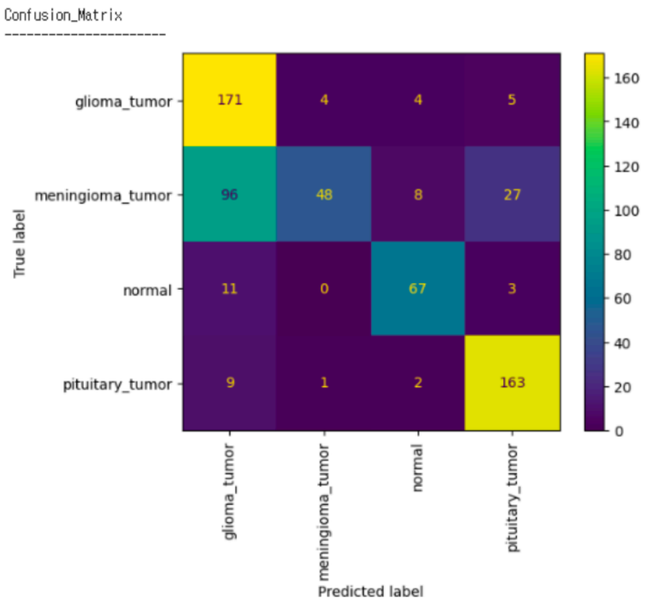
(지현님)

- ResNet50V2 사용 : acc와 recall 둘 다 상당히 떨어지고, meningioma 를 glioma로 오분류 하는 케이스가 많아졌다.

- Recall(marco avg): 0.74

	precision	recall	f1-score	support
0	0.60	0.93	0.73	184
1	0.91	0.27	0.41	179
2	0.83	0.83	0.83	81
3	0.82	0.93	0.87	175
accuracy			0.73	619
macro avg	0.79	0.74	0.71	619
weighted avg	0.78	0.73	0.69	619

Confusion Matrix



Trial - VGG

Performance Improvements

Basecode :

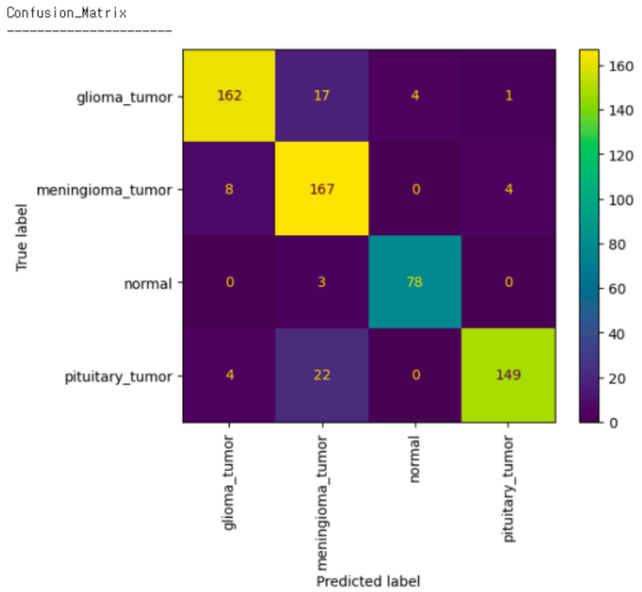
Acc : 91%, Recall : 92%

(지현님)

- VGG : Acc, Recall 둘 다 ResNet50 보다 조금 낮게 나옴.

- Recall(marco avg): 0.91

	precision	recall	f1-score	support
0	0.93	0.88	0.91	184
1	0.80	0.93	0.86	179
2	0.95	0.96	0.96	81
3	0.97	0.85	0.91	175
accuracy			0.90	619
macro avg	0.91	0.91	0.91	619
weighted avg	0.91	0.90	0.90	619



Trial - ResNet101

Performance Improvements

Basecode :

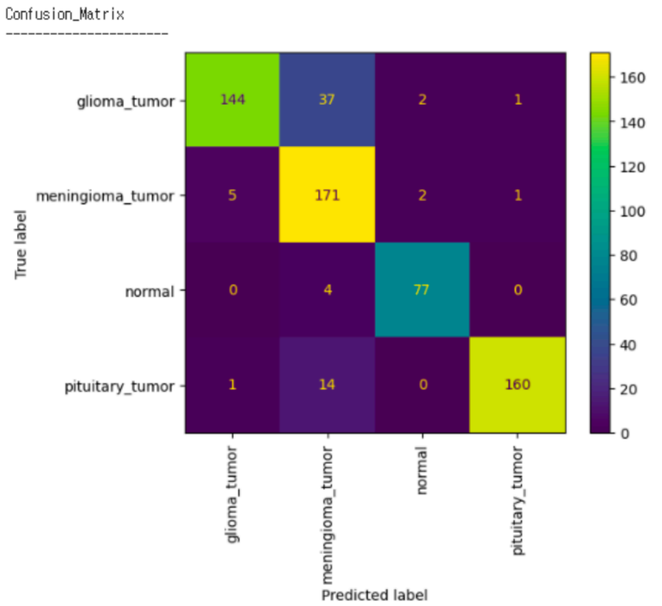
Acc : 91%, Recall : 92%

(지현님)

- ResNet101 : Matrix를 보았을 때 ResNet50보다 더 잘 분류한 클래스도 있지만, Acc, Recall 둘 다 ResNet50보다 적게 나왔음.

- Recall(marco avg): 0.90

	precision	recall	f1-score	support
0	0.96	0.78	0.86	184
1	0.76	0.96	0.84	179
2	0.95	0.95	0.95	81
3	0.99	0.91	0.95	175
accuracy			0.89	619
macro avg	0.91	0.90	0.90	619
weighted avg	0.91	0.89	0.89	619



Summary

Performance Improvements

- 의미 있었던 시도
  - Batch normalization
  - Data augmentation
- 앞으로 해 볼 시도
  - GridSearch, K-Fold cross validation 등의 시도

Google Developer Student Clubs

Thank you for listening

Brain tumors 256x256
Performance improvements

GDSC Hanyang ML/DL Basic : 이재승, 서지현