인공지능 과제

aws활용 인공지능 실험

21828752 이건희

기존코드

In [10]:
```
# 모델 생성
model.compile(
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# 모델 학습
history = model.fit(X_train, Y_train, validation_data=(X_dev, Y_dev), epochs=epochs)
```

```
2023-06-02 11:37:51.607243: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 99878400 exceeds 10% of free system memory.

Epoch 1/5
19/19 [==============================] - ETA: 0s - loss: 27.0147 - accuracy: 0.4133
2023-06-02 11:37:53.071892: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 349574400 exceeds 10% of free system memor
y.
19/19 [==============================] - 2s 62ms/step - loss: 27.0147 - accuracy: 0.4133 - val_loss: 9.4247 - val_accuracy: 0.5343
Epoch 2/5
19/19 [==============================] - 0s 26ms/step - loss: 13.9993 - accuracy: 0.4733 - val_loss: 11.8438 - val_accuracy: 0.5019
Epoch 3/5
19/19 [==============================] - 1s 28ms/step - loss: 8.7962 - accuracy: 0.6083 - val_loss: 18.7785 - val_accuracy: 0.5852
Epoch 4/5
19/19 [==============================] - 0s 25ms/step - loss: 10.8831 - accuracy: 0.5350 - val_loss: 6.7494 - val_accuracy: 0.6367
Epoch 5/5
19/19 [==============================] - 0s 27ms/step - loss: 8.8431 - accuracy: 0.6517 - val_loss: 1.8967 - val_accuracy: 0.7957
```

In [12]:
```
# 정확도 및 손실률을 기준으로 모델 성능 평가
score = model.evaluate(X_test, Y_test)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
10/10 [==============================] - 0s 4ms/step - loss: 2.0999 - accuracy: 0.7533
Test loss: 2.09993839263916
Test accuracy: 0.753333330154419

2023-06-02 11:37:56.191017: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 49939200 exceeds 10% of free system memory.
```

leaning_late 추가 0.001

In [11]:
```
# 모델 생성
model.compile(loss='categorical_crossentropy',
              optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
              metrics=['accuracy'])
|
# 모델 학습
history = model.fit(X_train, Y_train, validation_data=(X_dev, Y_dev), epochs=epochs)
```

```
Epoch 1/5
2023-06-02 11:45:56.869430: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 99878400 exceeds 10% of free system memory.
13/19 [===================>.....] - ETA: 0s - loss: 19.9001 - accuracy: 0.3245
2023-06-02 11:45:58.063950: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 349574400 exceeds 10% of free system memor
y.
19/19 [==============================] - 2s 75ms/step - loss: 15.3804 - accuracy: 0.3667 - val_loss: 5.4147 - val_accuracy: 0.6329
Epoch 2/5
19/19 [==============================] - 0s 26ms/step - loss: 4.9668 - accuracy: 0.5917 - val_loss: 3.4049 - val_accuracy: 0.6495
Epoch 3/5
19/19 [==============================] - 0s 26ms/step - loss: 2.6876 - accuracy: 0.7133 - val_loss: 2.9594 - val_accuracy: 0.7038
Epoch 4/5
19/19 [==============================] - 0s 27ms/step - loss: 1.6678 - accuracy: 0.7583 - val_loss: 2.0738 - val_accuracy: 0.7405
Epoch 5/5
19/19 [==============================] - 0s 27ms/step - loss: 1.5183 - accuracy: 0.7633 - val_loss: 1.9114 - val_accuracy: 0.7024
```

In [13]:
```
# 정확도 및 손실률을 기준으로 모델 성능 평가
score = model.evaluate(X_test, Y_test)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
10/10 [==============================] - 0s 4ms/step - loss: 1.6998 - accuracy: 0.7233
Test loss: 1.6997570991516113
Test accuracy: 0.7233333587646484
```

leaning_late 추가 0.001 결과

추가전: 모델 학습률이 0.6517 / 성능 평가 : 0.7533
추가후: 모델 학습률이 0.7633 / 성능 평가 : 0.7523
결론 : 모델 학습률이 초기코드에서 0.6517 > 0.7633 으로 0.1116으로 증가하는반면
성능평가는 0.7533 > 0.7523으로 0.001 감소함

leaning_late 추가 0.003

In [18]:
```
# 모델 생성
model.compile(loss='categorical_crossentropy',
              optimizer=tf.keras.optimizers.Adam(learning_rate=0.003),
              metrics=['accuracy'])

# 모델 학습
history = model.fit(X_train, Y_train, validation_data=(X_dev, Y_dev), epochs=epochs)
```

```
Epoch 1/5
2023-06-02 11:48:03.396681: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 99878400 exceeds 10% of free system memory.
13/19 [===================>.....] - ETA: 0s - loss: 26.5361 - accuracy: 0.5529
2023-06-02 11:48:04.677782: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 349574400 exceeds 10% of free system memor
y.
19/19 [==============================] - 2s 59ms/step - loss: 19.5741 - accuracy: 0.5950 - val_loss: 2.0503 - val_accuracy: 0.7895
Epoch 2/5
19/19 [==============================] - 0s 26ms/step - loss: 1.7750 - accuracy: 0.8317 - val_loss: 2.7479 - val_accuracy: 0.7800
Epoch 3/5
19/19 [==============================] - 0s 26ms/step - loss: 2.1448 - accuracy: 0.7900 - val_loss: 1.3467 - val_accuracy: 0.8424
Epoch 4/5
19/19 [==============================] - 0s 26ms/step - loss: 1.7921 - accuracy: 0.8250 - val_loss: 1.8814 - val_accuracy: 0.8186
Epoch 5/5
19/19 [==============================] - 0s 26ms/step - loss: 1.5341 - accuracy: 0.8267 - val_loss: 0.9750 - val_accuracy: 0.8776
```

In [20]:
```
# 정확도 및 손실률을 기준으로 모델 성능 평가
score = model.evaluate(X_test, Y_test)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
10/10 [==============================] - 0s 10ms/step - loss: 1.2579 - accuracy: 0.8667
Test loss: 1.2579450144042969
Test accuracy: 0.8666666746139526
```

leaning_late 추가 0.003 결과

추가전: 모델 학습률이 0.6517 / 성능 평가 : 0.7533
추가후: 모델 학습률이 0.8267 / 성능 평가 : 0.8666
결론 : 모델 학습률이 초기코드에서 0.6517 > 0.8267 으로 0.175으로 증가하고
성능평가는 0.7533 > 0.8666 으로 0.1133으로 증가함

leaning_late 추가 0.005

In [25]:
```
# 모델 생성
model.compile(loss='categorical_crossentropy',
              optimizer=tf.keras.optimizers.Adam(learning_rate=0.005),
              metrics=['accuracy'])

# 모델 학습
history = model.fit(X_train, Y_train, validation_data=(X_dev, Y_dev), epochs=epochs)
```

```
Epoch 1/5
2023-06-02 11:49:55.744883: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] Allocation of 99878400 exceeds 10% of free system memory.
19/19 [==============================] - 2s 78ms/step - loss: 36.5073 - accuracy: 0.5583 - val_loss: 12.5004 - val_accuracy: 0.6614
Epoch 2/5
19/19 [==============================] - 0s 27ms/step - loss: 7.3771 - accuracy: 0.7633 - val_loss: 6.2105 - val_accuracy: 0.7557
Epoch 3/5
19/19 [==============================] - 0s 26ms/step - loss: 3.1344 - accuracy: 0.8200 - val_loss: 2.6671 - val_accuracy: 0.8424
Epoch 4/5
19/19 [==============================] - 0s 24ms/step - loss: 2.7893 - accuracy: 0.8400 - val_loss: 3.6379 - val_accuracy: 0.8014
Epoch 5/5
19/19 [==============================] - 0s 25ms/step - loss: 2.9604 - accuracy: 0.8500 - val_loss: 6.2085 - val_accuracy: 0.7505
```

In [27]:
```
# 정확도 및 손실률을 기준으로 모델 성능 평가
score = model.evaluate(X_test, Y_test)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
10/10 [==============================] - 0s 9ms/step - loss: 5.3906 - accuracy: 0.7733
Test loss: 5.390617370605469
Test accuracy: 0.7733333110809326
```

leaning_late 0.003 에 배치사이즈 32추가

In [33]:
```
model.compile(loss='categorical_crossentropy',
              optimizer=tf.keras.optimizers.Adam(learning_rate=0.003),
              metrics=['accuracy'])

# 모델 학습
history = model.fit(X_train, Y_train, validation_data=(X_dev, Y_dev), epochs=epochs, batch_size=32)
```

```
Epoch 1/5
19/19 [==============================] - 6s 266ms/step - loss: 61.3160 - accuracy: 0.3917 - val_loss: 6.0636 - val_accuracy: 0.6143
Epoch 2/5
19/19 [==============================] - 4s 236ms/step - loss: 12.8298 - accuracy: 0.4450 - val_loss: 0.7628 - val_accuracy: 0.7443
Epoch 3/5
19/19 [==============================] - 4s 235ms/step - loss: 2.3387 - accuracy: 0.4017 - val_loss: 1.0407 - val_accuracy: 0.4295
Epoch 4/5
19/19 [==============================] - 4s 237ms/step - loss: 1.0563 - accuracy: 0.3983 - val_loss: 0.8800 - val_accuracy: 0.5733
Epoch 5/5
19/19 [==============================] - 4s 236ms/step - loss: 1.0565 - accuracy: 0.4083 - val_loss: 1.0590 - val_accuracy: 0.3924
```

```
              optimizer=tf.keras.optimizers.Adam(learning_rate=0.003),
              metrics=['accuracy'])

# 모델 학습
history = model.fit(X_train, Y_train, validation_data=(X_dev, Y_dev), epochs=epochs)
```

배치사이즈를 추가하여 비교과정에서 유의미한 차이를 확인하지 못하여 없는상태로 진행

```
Epoch 1/5
19/19 [==============================] - 6s 266ms/step - loss: 66.7656 - accuracy: 0.3817 - val_loss
Epoch 2/5
19/19 [==============================] - 4s 231ms/step - loss: 13.6219 - accuracy: 0.4033 - val_loss: 0.8307 - val_accuracy: 0.6557
Epoch 3/5
19/19 [==============================] - 4s 238ms/step - loss: 1.0456 - accuracy: 0.4733 - val_loss: 0.8734 - val_accuracy: 0.6290
Epoch 4/5
19/19 [==============================] - 4s 230ms/step - loss: 1.0123 - accuracy: 0.4533 - val_loss: 0.8522 - val_accuracy: 0.6000
Epoch 5/5
19/19 [==============================] - 4s 232ms/step - loss: 1.0002 - accuracy: 0.4417 - val_loss: 0.8103 - val_accuracy: 0.6214
```

leaning_late 추가 0.005 결과

추가전: 모델 학습률이 0.6517 / 성능 평가 : 0.7533
추가후: 모델 학습률이 0.8500 / 성능 평가 : 0.7733
결론 : 모델 학습률이 초기코드에서 0.6517 > 0.8500 으로 0.1983으로 증가하고
성능평가는 0.7533 > 0.7733으로 0.2 증가함

leaning_late 추가 결론

In [12]:
```
# 정확도 및 손실률을 기준으로 모델 성능 평가
score = model.evaluate(X_test, Y_test)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
10/10 [==============================] - 0s 4ms/step - loss: 2.0999 - accuracy: 0.7533
Test loss: 2.09993839263916
Test accuracy: 0.7533333301544189
```

In [13]:
```
# 정확도 및 손실률을 기준으로 모델 성능 평가
score = model.evaluate(X_test, Y_test)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
10/10 [==============================] - 0s 4ms/step - loss: 1.6998 - accuracy: 0.7233
Test loss: 1.6997570991516113
Test accuracy: 0.7233333587646484
```

In [20]:
```
# 정확도 및 손실률을 기준으로 모델 성능 평가
score = model.evaluate(X_test, Y_test)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
10/10 [==============================] - 0s 10ms/step - loss: 1.2579 - accuracy: 0.8667
Test loss: 1.2579460144042969
Test accuracy: 0.8666666746139526
```

In [27]:
```
# 정확도 및 손실률을 기준으로 모델 성능 평가
score = model.evaluate(X_test, Y_test)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
10/10 [==============================] - 0s 9ms/step - loss: 5.3906 - accuracy: 0.7733
Test loss: 5.3906171370605469
Test accuracy: 0.77333331108093262
```

가장 성능이 좋은경우는 leanning_late 를 0.003로 한 경우로 판단되어 이후로 은닉계층과 드롭아웃을 수행할때 leanning_late를 0.003로 사용했음

은닉층 및 드롭아웃(기본코드)

In [9]:
```
# 신경망 모델 생성
model = tf.keras.Sequential(name = 'FirstModel')

model = tf.keras.Sequential(name='SingleLayerPerceptron')
model.add(tf.keras.layers.Flatten(input_shape=(n_H, n_W, 3), name='Flatten')) # 입력층
model.add(tf.keras.layers.Dense(units=3, activation='softmax', name='Output')) # 출력층

# 생성될 신경망 모델 구조 출력
model.summary()
```

```
2023-06-02 11:44:07.622002: E tensorflow/compiler/xla/stream_executor/cuda/cuda_driver.cc:266] failed call to cuInit: CUDA_ERROR_NO_DEVIC
E: no CUDA-capable device is detected
```

```
Model: "SingleLayerPerceptron"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Flatten (Flatten)           (None, 41616)             0

 Output (Dense)              (None, 3)                 124851

=================================================================
Total params: 124,851
Trainable params: 124,851
Non-trainable params: 0
_________________________________________________________________
```

은닉층 및 드롭아웃(은닉층+1드롭아웃+1)

In [48]:
```
# 모델 생성
model = tf.keras.Sequential(name='SingleLayerPerceptron')
model.add(tf.keras.layers.Flatten(input_shape=(n_H, n_W, 3), name='Flatten')) # 입력층
model.add(tf.keras.layers.Dense(units=256, activation='relu', name='Hidden1')) # 은닉층
model.add(tf.keras.layers.Dropout(0.5)) # 드롭아웃을 통한 정규화
model.add(tf.keras.layers.Dense(units=3, activation='softmax', name='Output')) # 출력층

# 생성될 신경망 모델 구조 출력
model.summary()
```

```
Model: "SingleLayerPerceptron"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Flatten (Flatten)            (None, 41616)             0

Hidden1 (Dense)              (None, 256)               10653952

dropout_4 (Dropout)          (None, 256)               0

Output (Dense)               (None, 3)                 771

=================================================================
Total params: 10,654,723
Trainable params: 10,654,723
Non-trainable params: 0
_________________________________________________________________
```

은닉층을 1게 추가하였고 드롭아웃을 0.5를 해주었다

결과 : 성능이 기존 0.7733 > 0.6133으로 0.16 감소함

```
score = model.evaluate(X_test, Y_test)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
10/10 [==============================] - 0s 24ms/step - loss: 0.7845 - accuracy: 0.6133
Test loss: 0.7844974994659424
Test accuracy: 0.6133333444595337
```

은닉층 및 드롭아웃(은닉층+2드롭아웃없엠)

In [64]:
```
# 모델 생성
model = tf.keras.Sequential(name='SingleLayerPerceptron')
model.add(tf.keras.layers.Flatten(input_shape=(n_H, n_W, 3), name='Flatten')) # 입력층
model.add(tf.keras.layers.Dense(units=256, activation='relu', name='Hidden1')) # 은닉층

model.add(tf.keras.layers.Dense(units=128, activation='relu', name='Hidden2')) # 은닉층

model.add(tf.keras.layers.Dense(units=3, activation='softmax', name='Output')) # 출력층

# 생성될 신경망 모델 구조 출력
model.summary()
```

```
Model: "SingleLayerPerceptron"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Flatten (Flatten)            (None, 41616)             0

Hidden1 (Dense)              (None, 256)               10653952

Hidden2 (Dense)              (None, 128)               32896

Output (Dense)               (None, 3)                 387

=================================================================
Total params: 10,687,235
Trainable params: 10,687,235
Non-trainable params: 0
_________________________________________________________________
```

은닉층을 2게 추가하였고 드롭아웃을 재거하였다

결과 : 성능이 기존 0.7733 > 0.8433으로 0.7 증가함

In [67]:
```
# 정확도 및 손실률을 기준으로 모델 성능 평가
score = model.evaluate(X_test, Y_test)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
10/10 [==============================] - 0s 17ms/step - loss: 0.6658 - accuracy: 0.8433
Test loss: 0.6657990217208862
Test accuracy: 0.8433333039283752
```

은닉층 및 드롭아웃(은닉층+2드롭아웃+2)

In [32]:
```
# 모델 생성
model = tf.keras.Sequential(name='SingleLayerPerceptron')
model.add(tf.keras.layers.Flatten(input_shape=(n_H, n_W, 3), name='Flatten')) # 입력층
model.add(tf.keras.layers.Dense(units=256, activation='relu', name='Hidden1')) # 은닉층
model.add(tf.keras.layers.Dropout(0.5)) # 드롭아웃을 통한 정규화
model.add(tf.keras.layers.Dense(units=128, activation='relu', name='Hidden2')) # 은닉층
model.add(tf.keras.layers.Dropout(0.5)) # 드롭아웃을 통한 정규화
model.add(tf.keras.layers.Dense(units=3, activation='softmax', name='Output')) # 출력층

# 생성될 신경망 모델 구조 출력
model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
Flatten (Flatten)            (None, 41616)             0

Hidden1 (Dense)              (None, 256)               10653952

dropout (Dropout)            (None, 256)               0

Hidden2 (Dense)              (None, 128)               32896

dropout_1 (Dropout)          (None, 128)               0

Output (Dense)               (None, 3)                 387

=================================================================
Total params: 10,687,235
Trainable params: 10,687,235
Non-trainable params: 0
_________________________________________________________________
```

은닉층을 2게 추가하였고 드롭아웃을 0.5로 계층마다 추가하였다
결과 : 커널이 터지고 사진을 못찍었지만 드롭아웃이 없는 경우가 성능이 더 좋아서 사진을 못찍음

은닉층 및 드롭아웃(은닉층+2드롭아웃+2)

In [72]:
```
# 모델 생성
model = tf.keras.Sequential(name='SingleLayerPerceptron')
model.add(tf.keras.layers.Flatten(input_shape=(n_H, n_W, 3), name='Flatten')) # 입력층
model.add(tf.keras.layers.Dense(units=256, activation='relu', name='Hidden1')) # 은닉층

model.add(tf.keras.layers.Dense(units=128, activation='relu', name='Hidden2')) # 은닉층
model.add(tf.keras.layers.Dense(units=64, activation='relu', name='Hidden3')) # 은닉층

model.add(tf.keras.layers.Dense(units=3, activation='softmax', name='Output')) # 출력층

# 생성될 신경망 모델 구조 출력
model.summary()
```

Model: "SingleLayerPerceptron"

Layer (type)	Output Shape	Param #
Flatten (Flatten)	(None, 41616)	0
Hidden1 (Dense)	(None, 256)	10653952
Hidden2 (Dense)	(None, 128)	32896
Hidden3 (Dense)	(None, 64)	8256
Output (Dense)	(None, 3)	195

Total params: 10,695,299
Trainable params: 10,695,299
Non-trainable params: 0

은닉층을 3게 추가하였고
드롭아웃을 재거하였다

결과 : 성능이 기존 0.7733 > 0.8299으로 0.566 증가함

In [75]:
```
# 정확도 및 손실률을 기준으로 모델 성능 평가
score = model.evaluate(X_test, Y_test)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
10/10 [==============================] - 0s 30ms/step - loss: 0.5479 - accuracy: 0.8300
Test loss: 0.5478652119636536
Test accuracy: 0.8299999833106995
```

은닉층 및 드롭아웃 결과

성능 평가 기준으로
은닉층이 1 > 2 > 3 으로 실험한결과 2게인경우가 가장 성능평가가 좋음
드롭아웃의 경우 은닉층 사이사이 넣는것보다는 마지막에 한번 넣거나 없는경우가
성능평가가 가장 좋았음

CNN으로 변경 및 드롭아웃 수치및 위치조절 (CNN변경코드)

```
In [9]: # 모델 생성
model = tf.keras.Sequential(name='ConvolutionalNeuralNetwork')
model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(n_H, n_W, 3), name='Conv1'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='MaxPool1'))
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu', name='Conv2'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='MaxPool2'))
model.add(tf.keras.layers.Flatten(name='Flatten'))
model.add(tf.keras.layers.Dense(units=256, activation='relu', name='Hidden1'))
model.add(tf.keras.layers.Dense(units=128, activation='relu', name='Hidden2'))
model.add(tf.keras.layers.Dense(units=3, activation='softmax', name='Output'))

# 생성된 신경망 모델 구조 출력
model.summary()
```

```
2023-06-02 12:22:42.956557: E tensorflow/compiler/xla/stream_executor/cuda/cuda_driver.cc:266] failed call to cuInit: CUDA_ERROR_NO_DEVIC
E: no CUDA-capable device is detected
2023-06-02 12:22:43.094676: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] All
2023-06-02 12:22:43.180754: W tensorflow/tsl/framework/cpu_allocator_impl.cc:83] All
```

Model: "ConvolutionalNeuralNetwork"

Layer (type)	Output Shape	Param #
Conv1 (Conv2D)	(None, 100, 134, 32)	896
MaxPool1 (MaxPooling2D)	(None, 50, 67, 32)	0
Conv2 (Conv2D)	(None, 48, 65, 64)	18496
MaxPool2 (MaxPooling2D)	(None, 24, 32, 64)	0
Flatten (Flatten)	(None, 49152)	0
Hidden1 (Dense)	(None, 256)	[illegible]
Hidden2 (Dense)	(None, 128)	[illegible]
Output (Dense)	(None, 3)	[illegible]

Total params: 12,635,843
Trainable params: 12,635,843
Non-trainable params: 0

CNN으로 변경하고 은닉층을 2게 사용하여 드롭아웃을 재거했다

결과 : 평가성능이 0.8433 > 0.8899로 0.466 증가함

```
In [12]: # 정확도 및 손실률을 기준으로 모델 성능 평가
score = model.evaluate(X_test, Y_test)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
10/10 [==============================] - 1s 145ms/step - loss: 0.4592 - accuracy: 0.8900
Test loss: 0.45921334624290466
Test accuracy: 0.8899999856948853
```

CNN으로 변경 및 드롭아웃 수치및 위치조절 (드롭아웃조절)

In [9]:
```
# 모델 생성
model = tf.keras.Sequential(name='ConvolutionalNeuralNetwork')
model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(n_H, n_W, 3), name='Conv1'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='MaxPool1'))
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu', name='Conv2'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='MaxPool2'))
model.add(tf.keras.layers.Flatten(name='Flatten'))
model.add(tf.keras.layers.Dense(units=256, activation='relu', name='Hidden1'))
model.add(tf.keras.layers.Dropout(0.1)) # 드롭아웃을 통한 정규화

model.add(tf.keras.layers.Dense(units=128, activation='relu', name='Hidden2'))
model.add(tf.keras.layers.Dropout(0.1)) # 드롭아웃을 통한 정규화

model.add(tf.keras.layers.Dense(units=3, activation='softmax', name='Output'))

# 생성될 신경망 모델 구조 출력
model.summary()
```

```
2023-06-02 12:38:17.112444: E tensorflow/compiler/xla/stream_executor/cuda/cuda_driver.cc:266] failed call to cuInit:
E: no CUDA-capable device is detected
```

```
Model: "ConvolutionalNeuralNetwork"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Conv1 (Conv2D)              (None, 100, 134, 32)      896

 MaxPool1 (MaxPooling2D)     (None, 50, 67, 32)        0

 Conv2 (Conv2D)              (None, 48, 65, 64)        18496

 MaxPool2 (MaxPooling2D)     (None, 24, 32, 64)        0

 Flatten (Flatten)           (None, 49152)             0

 Hidden1 (Dense)             (None, 256)               12583168

 dropout (Dropout)           (None, 256)               0

 Hidden2 (Dense)             (None, 128)               32896

 dropout_1 (Dropout)         (None, 128)               0

 Output (Dense)              (None, 3)                 387

=================================================================
Total params: 12,635,843
Trainable params: 12,635,843
Non-trainable params: 0
_________________________________________________________________
```

드롭아웃을 0.1으로 은닉계 층마다 추가해줌
결과 : 추가전 성능평가 0.8899 > 0.8733 으로 0.166 감소됐다

In [12]:
```
# 정확도 및 손실률을 기준으로 모델 성능 평가
score = model.evaluate(X_test, Y_test)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
10/10 [==============================] - 1s 136ms/step - loss: 0.4690 - accuracy: 0.8733
Test loss: 0.46897140453018 2
Test accuracy: 0.8733333349227905
```

CNN으로 변경 및 드롭아웃 수치및 위치조절 (드롭아웃조절)

In [25]:
```
# 모델 생성
model = tf.keras.Sequential(name='ConvolutionalNeuralNetwork')
model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(n_H, n_W, 3), name='Conv1'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='MaxPool1'))
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu', name='Conv2'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='MaxPool2'))
model.add(tf.keras.layers.Flatten(name='Flatten'))
model.add(tf.keras.layers.Dense(units=256, activation='relu', name='Hidden1'))
model.add(tf.keras.layers.Dropout(0.3)) # 드롭아웃을 통한 정규화

model.add(tf.keras.layers.Dense(units=128, activation='relu', name='Hidden2'))
model.add(tf.keras.layers.Dropout(0.3)) # 드롭아웃을 통한 정규화

model.add(tf.keras.layers.Dense(units=3, activation='softmax', name='Output'))

# 생성될 신경망 모델 구조 출력
model.summary()
```

```
Model: "ConvolutionalNeuralNetwork"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Conv1 (Conv2D)               (None, 100, 134, 32)      896

MaxPool1 (MaxPooling2D)      (None, 50, 67, 32)        0

Conv2 (Conv2D)               (None, 48, 65, 64)        18496

MaxPool2 (MaxPooling2D)      (None, 24, 32, 64)        0

Flatten (Flatten)            (None, 49152)             0

Hidden1 (Dense)              (None, 256)               12583168

dropout_2 (Dropout)          (None, 256)               0

Hidden2 (Dense)              (None, 128)               32896

dropout_3 (Dropout)          (None, 128)               0

Output (Dense)               (None, 3)                 387

=================================================================
Total params: 12,635,843
Trainable params: 12,635,843
Non-trainable params: 0
_________________________________________________________________
```

드롭아웃을 0.3으로 은닉계 층마다 추가해줌
결과 : 추가전 성능평가 0.8899 > 0.8766 으로 0.133 감소됐다

In [28]:
```
# 정확도 및 손실률을 기준으로 모델 성능 평가
score = model.evaluate(X_test, Y_test)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
10/10 [==============================] - 1s 137ms/step - loss: 0.3273 - accuracy: 0.8767
Test loss: 0.32732757925987244
Test accuracy: 0.8766666650772095
```

CNN으로 변경 및 드롭아웃 수치및 위치조절 (드롭아웃조절)

In [17]:
```
# 모델 생성
model = tf.keras.Sequential(name='ConvolutionalNeuralNetwork')
model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(n_H, n_W, 3), name='Conv1'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='MaxPool1'))
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu', name='Conv2'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='MaxPool2'))
model.add(tf.keras.layers.Flatten(name='Flatten'))
model.add(tf.keras.layers.Dense(units=256, activation='relu', name='Hidden1'))
model.add(tf.keras.layers.Dropout(0.5)) # 드롭아웃을 통한 정규화

model.add(tf.keras.layers.Dense(units=128, activation='relu', name='Hidden2'))
model.add(tf.keras.layers.Dropout(0.5)) # 드롭아웃을 통한 정규화

model.add(tf.keras.layers.Dense(units=3, activation='softmax', name='Output'))

# 생성될 신경망 모델 구조 출력
model.summary()
```

```
Model: "ConvolutionalNeuralNetwork"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Conv1 (Conv2D)               (None, 100, 134, 32)      896

MaxPool1 (MaxPooling2D)      (None, 50, 67, 32)        0

Conv2 (Conv2D)               (None, 48, 65, 64)        18496

MaxPool2 (MaxPooling2D)      (None, 24, 32, 64)        0

Flatten (Flatten)            (None, 49152)             0

Hidden1 (Dense)              (None, 256)               12583168

dropout (Dropout)            (None, 256)               0

Hidden2 (Dense)

dropout_1 (Dropout)

Output (Dense)

=================================================================
Total params: 12,635,843
Trainable params: 12,635,843
Non-trainable params: 0
_________________________________________________________________
```

드롭아웃을 0.5으로 은닉계 층마다 추가해줌
결과 : 추가전 성능평가 0.8899 > 0.9100 으로 0.201 증가됐다

In [20]:
```
# 정확도 및 손실률을 기준으로 모델 성능 평가
score = model.evaluate(X_test, Y_test)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
10/10 [==============================] - 1s 136ms/step - loss: 0.2599 - accuracy: 0.9100
Test loss: 0.25986477732658386
Test accuracy: 0.9100000262260437
```

CNN으로 변경 및 드롭아웃 수치및 위치조절 (CNN계층조절)

```
warnings.warn(
```

In [9]:
```
# 모델 생성
model = tf.keras.Sequential(name='ConvolutionalNeuralNetwork')
model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(n_H, n_W, 3), name='Conv1'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='MaxPool1'))
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu', name='Conv2'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='MaxPool2'))
model.add(tf.keras.layers.Conv2D(filters=128, kernel_size=(3, 3), activation='relu', name='Conv3'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='MaxPool3'))
model.add(tf.keras.layers.Dropout(0.2))  # 드롭아웃 비율 조정
model.add(tf.keras.layers.Flatten(name='Flatten'))
model.add(tf.keras.layers.Dense(units=256, activation='relu', name='Hidden1'))

model.add(tf.keras.layers.Dense(units=128, activation='relu', name='Hidden2'))

model.add(tf.keras.layers.Dense(units=3, activation='softmax', name='Output'))

# 생성된 신경망 모델 구조 출력
model.summary()
```

```
2023-06-02 12:55:15.509122: E tensorflow/compiler/xla/stream_executor/cuda/cuda_driver.cc:266] failed call to cuInit: 
RROR_NO_DEVICE: no CUDA-capable device is detected

Model: "ConvolutionalNeuralNetwork"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Conv1 (Conv2D)              (None, 100, 134, 32)      896

 MaxPool1 (MaxPooling2D)     (None, 50, 67, 32)        0

 Conv2 (Conv2D)              (None, 48, 65, 64)        18496

 MaxPool2 (MaxPooling2D)     (None, 24, 32, 64)        0

 Conv3 (Conv2D)              (None, 22, 30, 128)       73856

 MaxPool3 (MaxPooling2D)     (None, 11, 15, 128)       0

 dropout (Dropout)           (None, 11, 15, 128)       0

 Flatten (Flatten)           (None, 21120)             0

 Hidden1 (Dense)             (None, 256)               5406976

 Hidden2 (Dense)             (None, 128)               32896

 Output (Dense)              (None, 3)                 387

=================================================================
Total params: 5,533,507
Trainable params: 5,533,507
Non-trainable params: 0
_________________________________________________________________
```

CNN계층을 3게로 추가해주고 드롭아웃을 0.2로 CNN계층의 마지막에 추가해줬다, 은닉계층은 2게로 설정함
결과 : CNN계층 1게 은닉계층 2게 드롭아웃 없는경우의 기점으로 0.8433 > 0.9133 으로 0.7 증가됐다

In [12]:
```
# 정확도 및 손실률을 기준으로 모델 성능 평가
score = model.evaluate(X_test, Y_test)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
10/10 [==============================] - 2s 159ms/step - loss: 0.2367 - accuracy: 0.9133
Test loss: 0.23669537739658356
Test accuracy: 0.9133333563804626
```

CNN으로 변경 및 드롭아웃 수치및 위치조절 (CNN계층조절)

```
In [9]: # 모델 생성
model = tf.keras.Sequential(name='ConvolutionalNeuralNetwork')
model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(n_H, n_W, 3), name='Conv1'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='MaxPool1'))
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu', name='Conv2'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='MaxPool2'))
model.add(tf.keras.layers.Conv2D(filters=128, kernel_size=(3, 3), activation='relu', name='Conv3'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='MaxPool3'))
model.add(tf.keras.layers.Conv2D(filters=256, kernel_size=(3, 3), activation='relu', name='Conv4'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='MaxPool4'))
model.add(tf.keras.layers.Dropout(0.2)) # 드롭아웃 비율 조절
model.add(tf.keras.layers.Flatten(name='Flatten'))
model.add(tf.keras.layers.Dense(units=256, activation='relu', name='Hidden1'))

model.add(tf.keras.layers.Dense(units=128, activation='relu', name='Hidden2'))

model.add(tf.keras.layers.Dense(units=3, activation='softmax', name='Output'))

# 생성된 신경망 모델 구조 출력
model.summary()
```

```
2023-06-02 13:02:43.782203: E tensorflow/compiler/xla/stream_executor/cuda/cuda_driver.cc:266] failed call to cuInit: CUDA_ERROR_NO_DEVICE: no CUDA-capable device is detected
```

```
Model: "ConvolutionalNeuralNetwork"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Conv1 (Conv2D)               (None, 100, 134, 32)      896

MaxPool1 (MaxPooling2D)      (None, 50, 67, 32)        0

Conv2 (Conv2D)               (None, 48, 65, 64)        18496

MaxPool2 (MaxPooling2D)      (None, 24, 32, 64)        0

Conv3 (Conv2D)               (None, 22, 30, 128)       73856

MaxPool3 (MaxPooling2D)      (None, 11, 15, 128)       0

Conv4 (Conv2D)               (None, 9, 13, 256)        295168

MaxPool4 (MaxPooling2D)      (None, 4, 6, 256)         0

dropout (Dropout)            (None, 4, 6, 256)         0

Flatten (Flatten)            (None, 6144)              0

Hidden1 (Dense)              (None, 256)               1573120

Hidden2 (Dense)              (None, 128)               32896

Output (Dense)               (None, 3)                 387

=================================================================
Total params: 1,994,819
Trainable params: 1,994,819
Non-trainable params: 0
_________________________________________________________________
```

CNN계층을 하나더 추가해줘보았다
결과 : CNN계층 1게 은닉계층 2게 드롭아웃 없는경우의 기점으로 0.8433 > 0.8500 으로 0.067 증가됐다
이전실험보다는 0.9133 > 0.8500으로 0.633 감소되었다

```
In [12]: # 정확도 및 손실률을 기준으로 모델 성능 평가
score = model.evaluate(X_test, Y_test)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
10/10 [==============================] - 2s 174ms/step - loss: 0.3901 - accuracy: 0.8500
Test loss: 0.39005571603775024
Test accuracy: 0.8500000238418579
```

CNN으로 변경 및 드롭아웃 수치및 위치조절 (드롭아웃 위치조절)

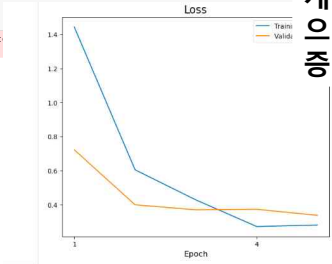
```
In [9]: # 모델 생성
model = tf.keras.Sequential(name='ConvolutionalNeuralNetwork')
model.add(tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(n_H, n_W, 3), name='Conv1'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='MaxPool1'))
model.add(tf.keras.layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu', name='Conv2'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='MaxPool2'))
model.add(tf.keras.layers.Conv2D(filters=128, kernel_size=(3, 3), activation='relu', name='Conv3'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2), name='MaxPool3'))

model.add(tf.keras.layers.Flatten(name='Flatten'))
model.add(tf.keras.layers.Dense(units=256, activation='relu', name='Hidden1'))

model.add(tf.keras.layers.Dense(units=128, activation='relu', name='Hidden2'))
model.add(tf.keras.layers.Dropout(0.2))  # 드롭아웃 비율 조정
model.add(tf.keras.layers.Dense(units=3, activation='softmax', name='Output'))

# 생성될 신경망 모델 구조 출력
model.summary()
```

```
2023-06-02 13:10:10.864395: E tensorflow/compiler/xla/stream_executo
E: no CUDA-capable device is detected
```

Model: "ConvolutionalNeuralNetwork"

Layer (type)	Output Shape	Param #
Conv1 (Conv2D)	(None, 100, 134, 32)	896
MaxPool1 (MaxPooling2D)	(None, 50, 67, 32)	0
Conv2 (Conv2D)	(None, 48, 65, 64)	18496
MaxPool2 (MaxPooling2D)	(None, 24, 32, 64)	0
Conv3 (Conv2D)	(None, 22, 30, 128)	73856
MaxPool3 (MaxPooling2D)	(None, 11, 15, 128)	0
Flatten (Flatten)	(None, 21120)	0
Hidden1 (Dense)	(None, 256)	5406976
Hidden2 (Dense)	(None, 128)	32896
dropout (Dropout)	(None, 128)	0
Output (Dense)	(None, 3)	387

Total params: 5,533,507
Trainable params: 5,533,507
Non-trainable params: 0

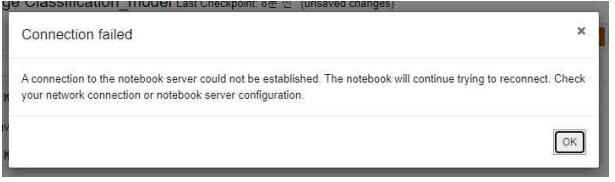



```
n [12]: # 정확도 및 손실률을 기준으로 모델 성능 평가
score = model.evaluate(X_test, Y_test)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
10/10 [==============================] - 2s 160ms/step - loss: 0.2118 - accuracy: 0.9200
Test loss: 0.21181893348693848
Test accuracy: 0.9200000166893005
```

CNN계층을 3게 은닉층을 2게 드롭아웃을 은닉층의 마지막에 두고 실험해봤다
결과 : CNN계층 1게 은닉계층 2게 드롭아웃 없는경우의 기점으로 0.8433 > 0.9200 으로 0.767 증가됐다

그리고



추가적으로 고치던도중 서버가 멈춰버렸다..
과제가 나오고 8~10시 간에 자주 일정이있어서 시간이 부족하여 실험을 더많이 못한부분이좀 아쉬웠다.

결론

CNN 계층 1개, 은닉 계층 2개, 드롭아웃 없는 경우의 성능 평가 0.8433과 비교하여
CNN 계층 3개, 은닉 계층 2개, 은닉 계층의 마지막에 드롭아웃을 추가한 경우의 성능 평가가 0.9200으로 크게 증가하였다.

실험과정에서의 성능평가중 가장 높은 확율을 선택하면
최종 모델은 CNN 계층 3개, 은닉 계층 2개, 은닉 계층의 마지막에 드롭아웃을 추가한 설정으로 0.9200의 성능을 가지는 모델을 선택하게 됐다.

느낀점

하이퍼파라미터 조정의 중요성: 실험을 통해 하이퍼파라미터인 학습률과 leaning_late의 조정이 모델의 성능에 큰 영향을 미친 것을 알 수 있었습니다. 올바른 하이퍼파라미터 값을 찾는 것이 모델 성능 향상에 중요한 요소라는 것을 깨달았습니다.

드롭아웃의 효과: 드롭아웃은 과적합을 방지하고 일반화 성능을 향상시키는데 효과적인 방법임을 확인했습니다. 하지만 드롭아웃을 너무 많이 적용하면 성능이 저하될 수 있다는 점도 발견했습니다.

모델 복잡도와 성능: CNN 계층과 은닉 계층의 개수를 조정하면 모델의 복잡도와 성능 간의 관계를 확인할 수 있었습니다. 모델이 복잡해질수록 성능이 향상되지만, 일정 수준 이상으로 복잡해지면 오히려 성능이 감소할 수 있다는 점을 알게 되었습니다.

실험 결과의 일관성: 실험 결과가 항상 일관되지 않는다는 점을 인지했습니다. 동일한 조건에서 실험을 반복해도 성능이 조금씩 변할 수 있으며, 이는 랜덤 요소와 데이터의 특성에 따른 영향이 있을 수 있음을 이해했습니다.