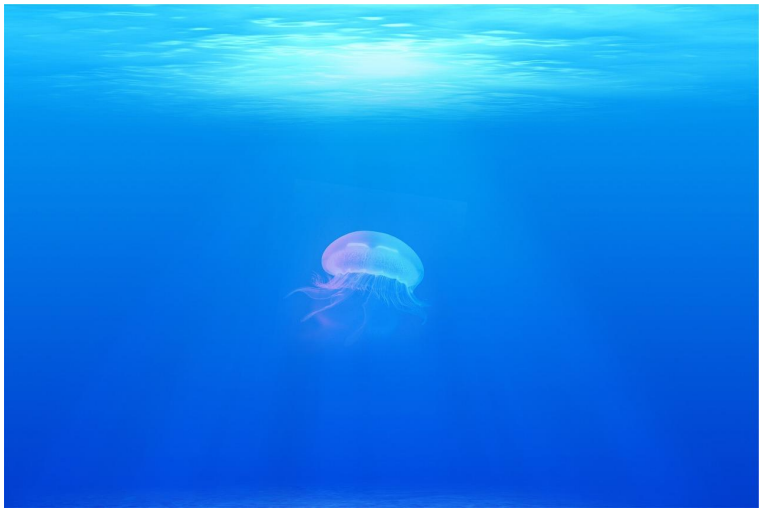
Hello!

Team WinWin(상생)



최현우, 김영진, 송민찬, 전다빈

Jellyfish

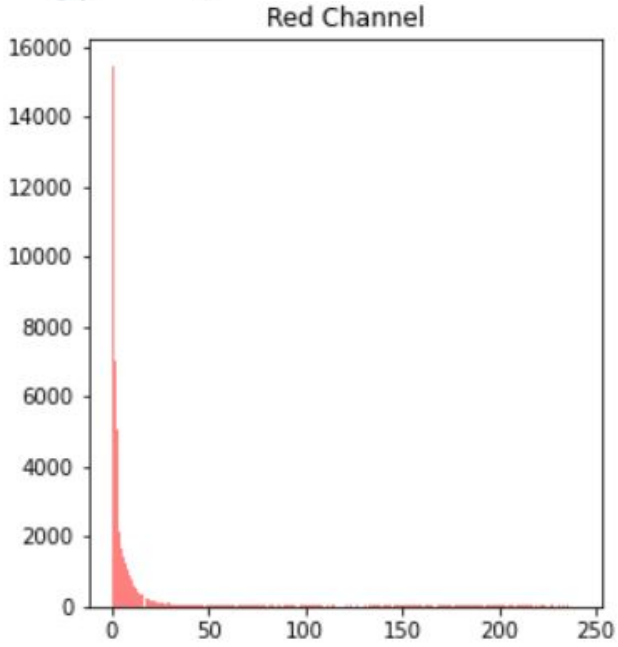
1. Data EDA & Preprocessing
2. Model Selection
3. W&B Analyze
4. Improve Model Performance
5. Result & Summary
6. Retrospect

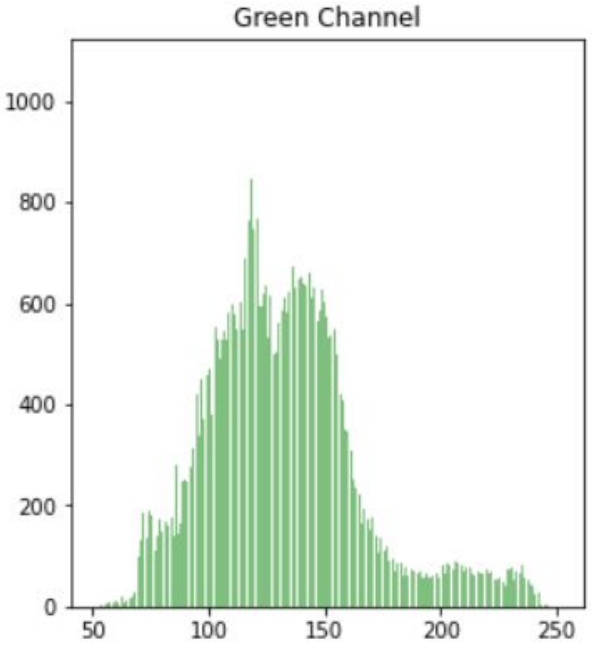
1. Data EDA &
Data Preprocessing

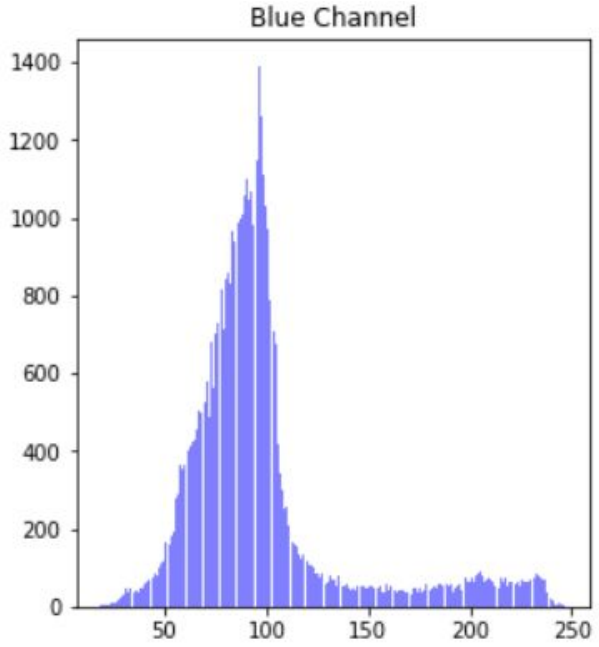
Data EDA

- 이미지 크기 분포 파악
- 라벨 분포 확인
- 픽셀 값 분포 분석
- 이상치 탐지(밝기)
- 라벨 시각화

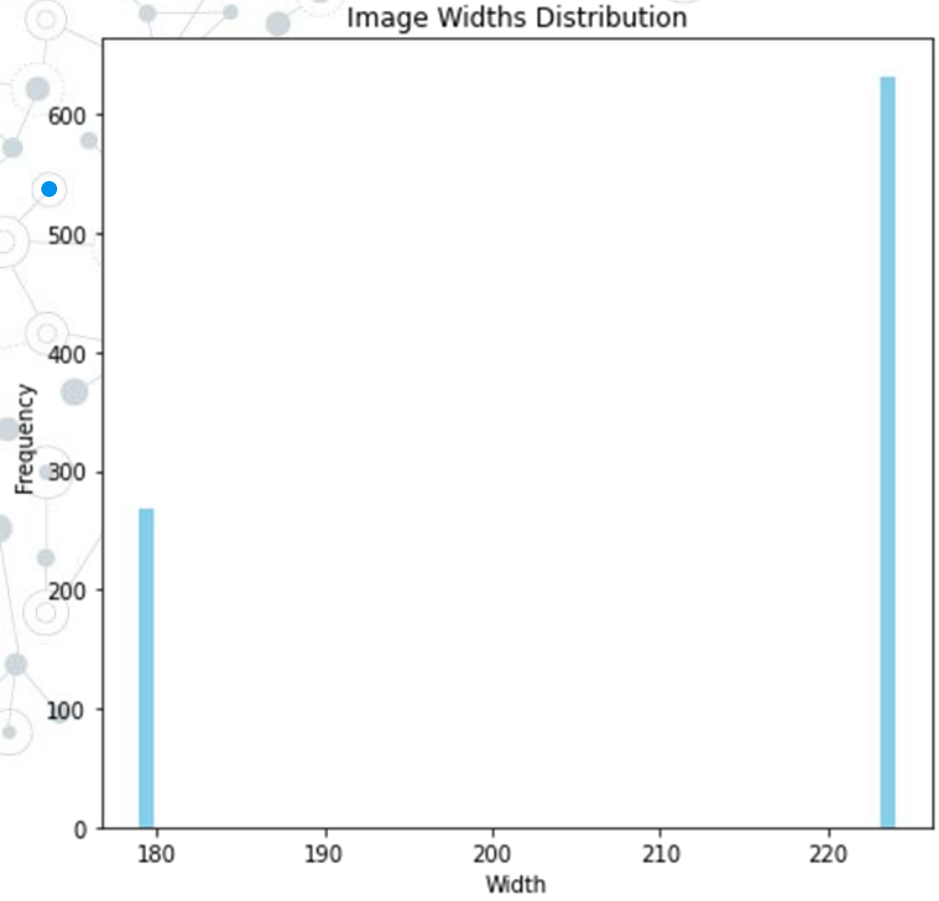
픽셀 값 분포 분석

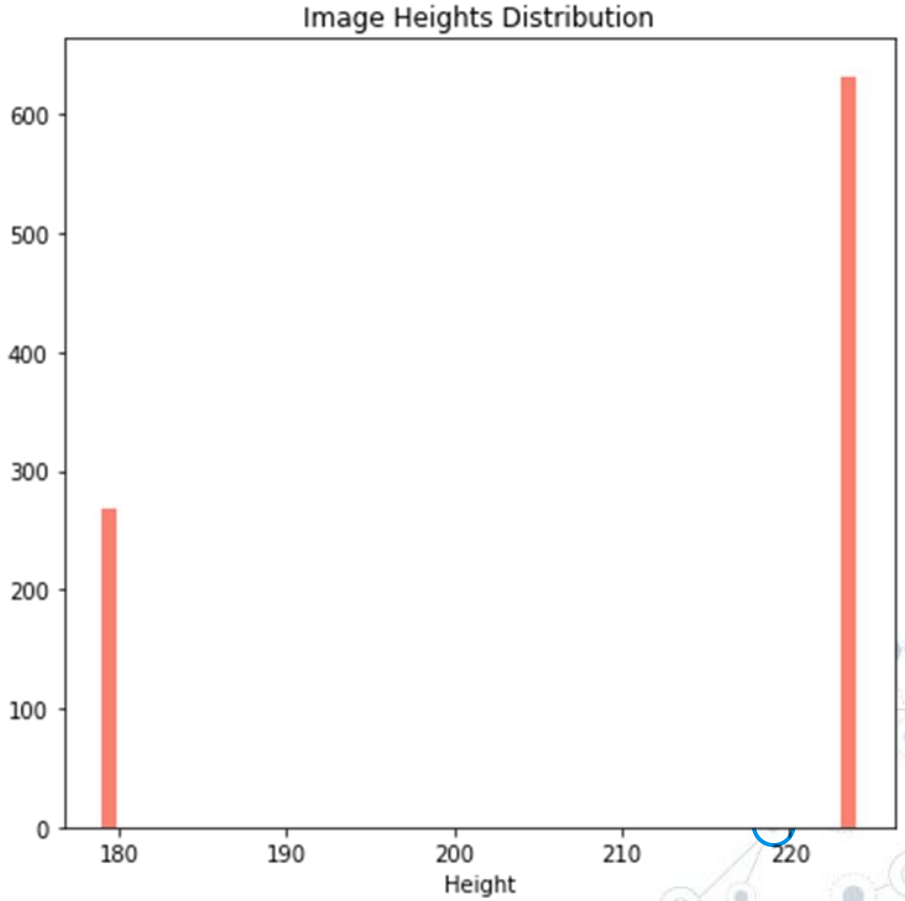

Red Channel
16000
14000
12000
10000
8000
6000
4000
2000
0
0
50
100
150
200
250


Green Channel
1000
800
600
400
200
0
50
100
150
200
250


Blue Channel
1400
1200
1000
800
600
400
200
0
50
100
150
200
250

이미지 크기 분포 파악





Size: (224, 224), Count: 632

Size: (179, 179), Count: 268

라벨 분포 확인

Label: compass_jellyfish, Count: 164
Label: Moon_jellyfish, Count: 162
Label: barrel_jellyfish, Count: 160
Label: blue_jellyfish, Count: 164
Label: lions_mane_jellyfish, Count: 165
Label: mauve_stinger_jellyfish, Count: 164

이상치 탐지 (밝기)

```
# 이상치로 판단
# 평균의 두 배를 넘는 밝기를 이상치로 설정
avg_threshold = np.mean([stat[1] for stat in intensity_stats])
 # 표준편차의 두 배를 넘는 변동을 이상치로 설정
std_threshold = np.mean([stat[2] for stat in intensity_stats])
```

```
Outlier detected: /aiffel/aiffel/jellyfish/Train_Test_Valid/Train/compass_jellyfish/aug-61-06.jpg
Average Intensity: 180.62722761461876
Intensity StdDev: 36.37232842824252

Outlier detected: /aiffel/aiffel/jellyfish/Train_Test_Valid/Train/mauve_stinger_jellyfish/52.jpg
Average Intensity: 181.17682424532313
Intensity StdDev: 59.181554417947204

Outlier detected: /aiffel/aiffel/jellyfish/Train_Test_Valid/Train/mauve_stinger_jellyfish/aug-27-35.JPG
Average Intensity: 181.03647410089155
Intensity StdDev: 49.07242897596128

Outlier detected: /aiffel/aiffel/jellyfish/Train_Test_Valid/Train/mauve_stinger_jellyfish/aug-55-68.jpg
Average Intensity: 187.9115446960034
Intensity StdDev: 44.15528983771726

Outlier detected: /aiffel/aiffel/jellyfish/Train_Test_Valid/Train/mauve_stinger_jellyfish/68.jpg
Average Intensity: 186.656037414966
Intensity StdDev: 47.56063746953782

Outlier detected: /aiffel/aiffel/jellyfish/Train_Test_Valid/Train/blue_jellyfish/aug-70-25.jpg
Average Intensity: 217.39431202168367
Intensity StdDev: 58.49630261889209
```

라벨 시각화


barrel_jellyfish
lions_mane_jellyfish
compass_jellyfish
mauve_stinger_jellyfish
lions_mane_jellyfish
Moon_jellyfish
compass_jellyfish
compass_jellyfish
lions_mane_jellyfish
Moon_jellyfish

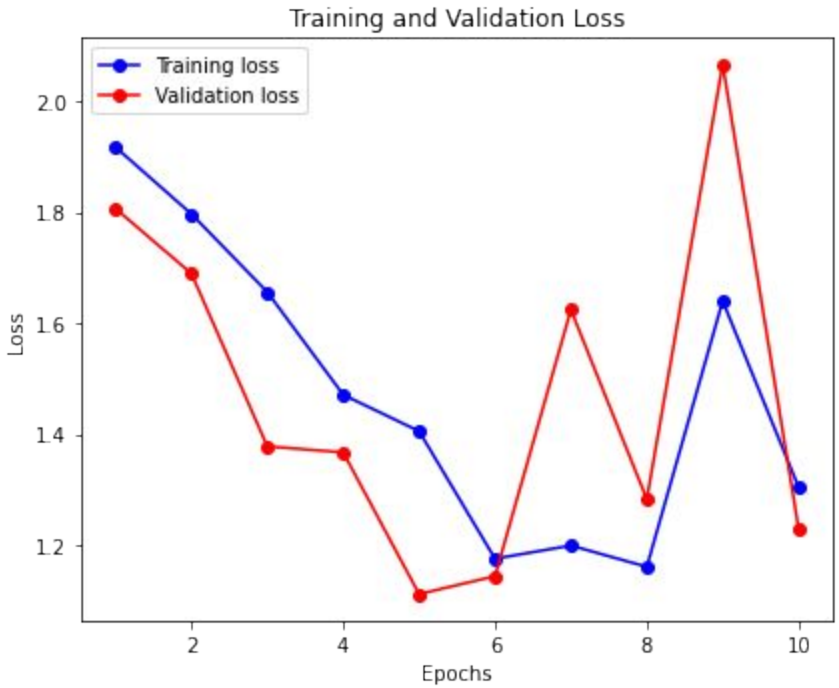
Data Preprocessing Function

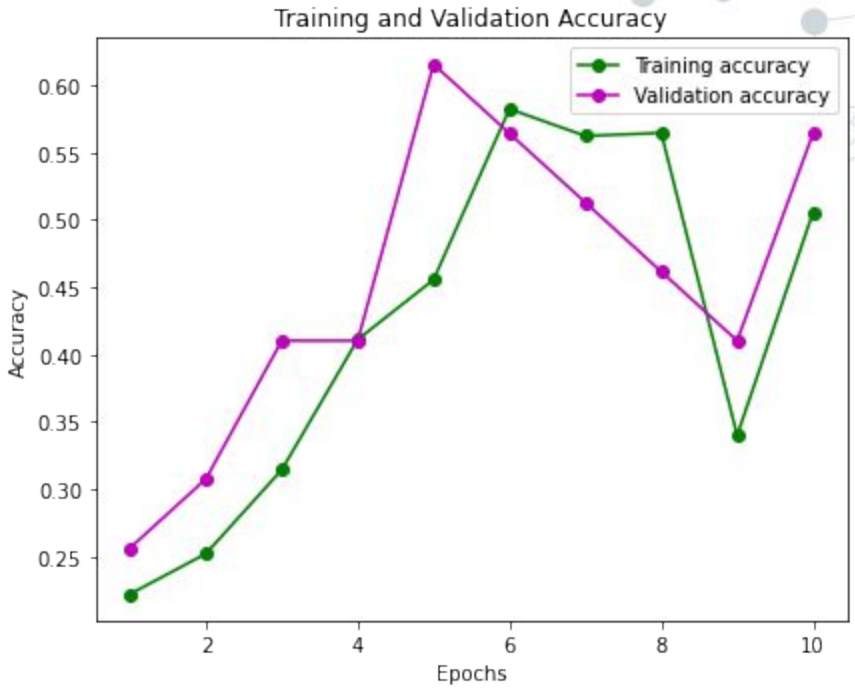
```python
def preprocess_image(image):
    # 이미지 데이터 타입을 uint8로 변환 (GoogleNet input shape 맞추기 위해)
    image = np.uint8(image)

    # PIL 이미지로 변환
    image = Image.fromarray(image)

    # 이상치 조정: 밝기
    enhancer = ImageEnhance.Brightness(image)
    target_brightness = 100  # 조정하려는 목표 밝기 (이상치 2배 설정 최소 177)
    # enhance(스케일 벡터, 인수로 1.0을 기준으로 밝기 조절, 목표밝기/기본 밝기 평균)
    image = enhancer.enhance(target_brightness / np.mean(image))

    # resize
    image = image.resize(output_size, Image.ANTIALIAS)

    # 색상 분포 균일화: 히스토그램 평활화, 우선 보류!했다가 투입!
    image = ImageOps.equalize(image)

    return np.array(image)
```

```
image = cv2.imread("jellyfish/Train_Test_Valid/Train/blue_jellyfish/31.jpg",
                   cv2.IMREAD_COLOR) # 흑백 이미지로 로드

kernel = np.array([[0, -1, 0],
                   [-1, 5,-1],
                   [0, -1, 0]]) # 커널을 만듭니다.

# 이미지를 선명하게 만듭니다.
image_sharp = cv2.filter2D(image, -1, kernel)

plt.imshow(image_sharp), plt.axis("off") # 이미지 출력
plt.show()
```

2. Model Selection

VGG16

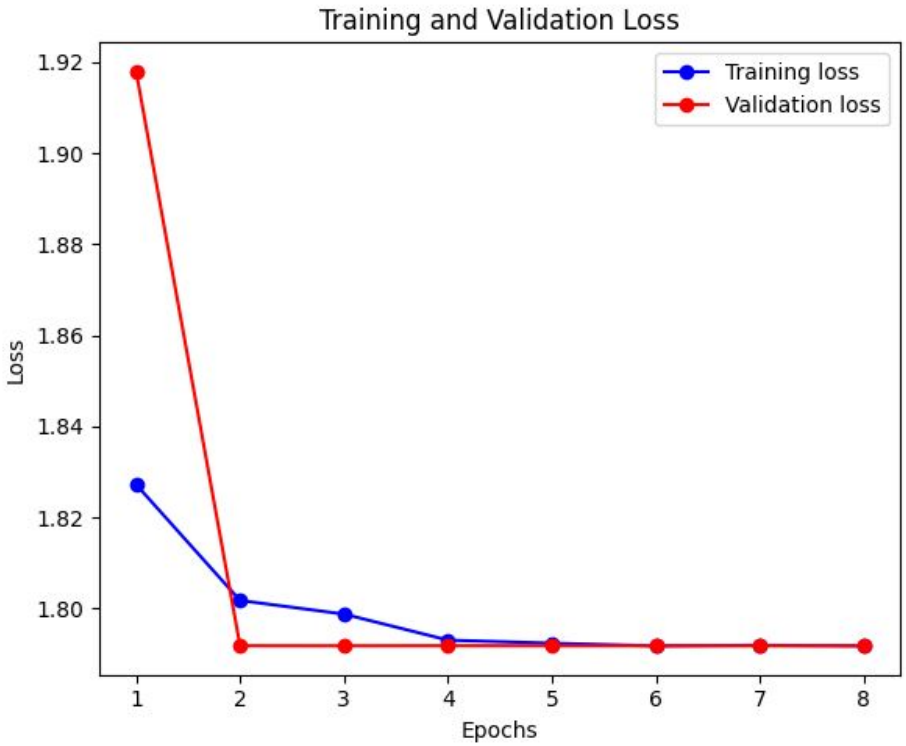


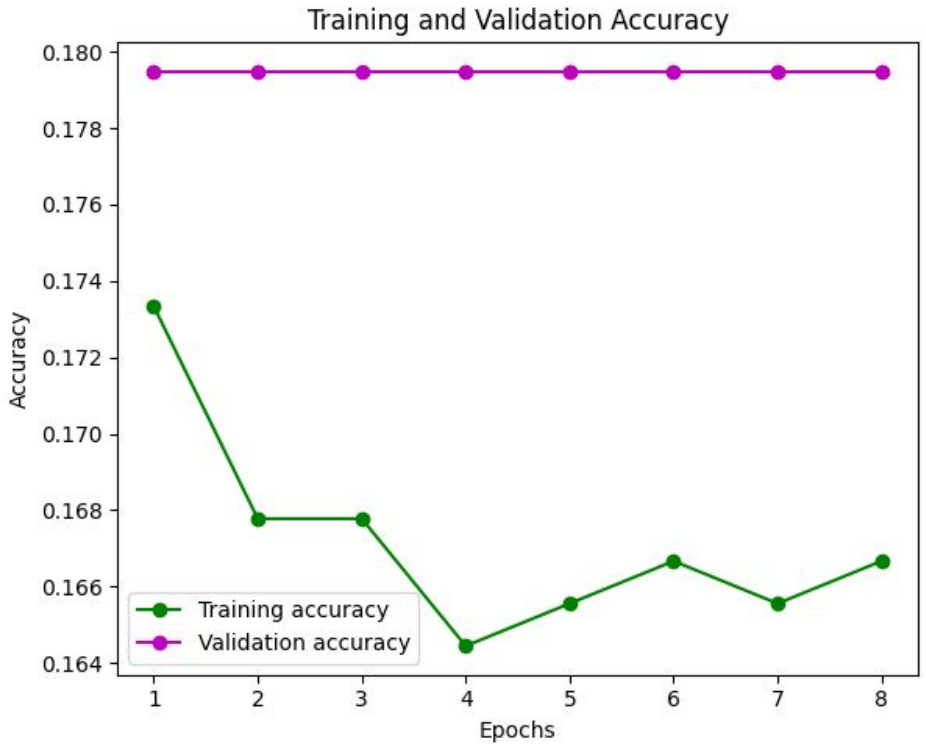


loss: 0.8749 - accuracy: 0.7750 - precision_2: 0.9524 - recall_2: 0.5000

Test : [0.8748747706413269, 0.7749999761581421, 0.9523809552192688, 0.5]

Resnet50V2

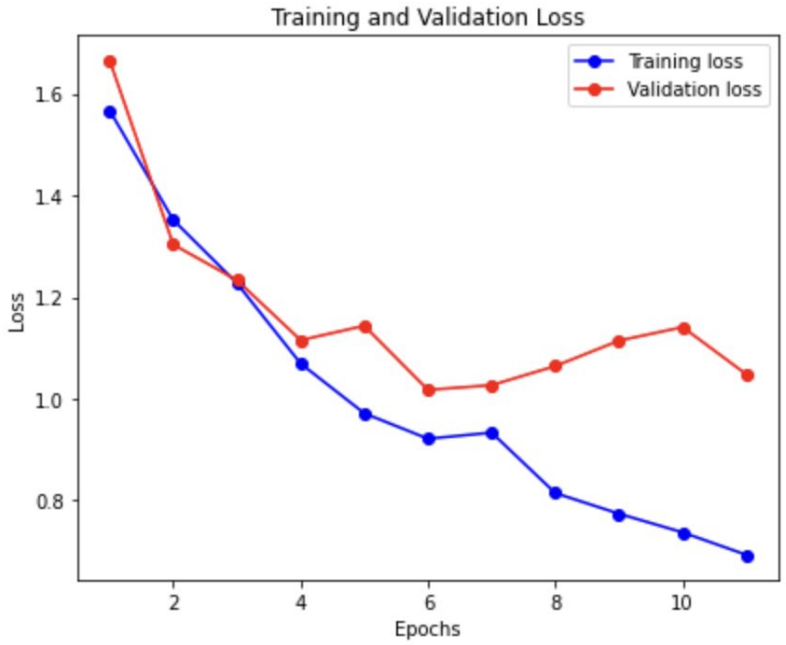




loss: 1.7919 - accuracy: 0.1750 - precision_11: 0.0000e+00 - recall_11: 0.0000e+00

Test : [1.7918907403945923, 0.17499999701976776, 0.0, 0.0]

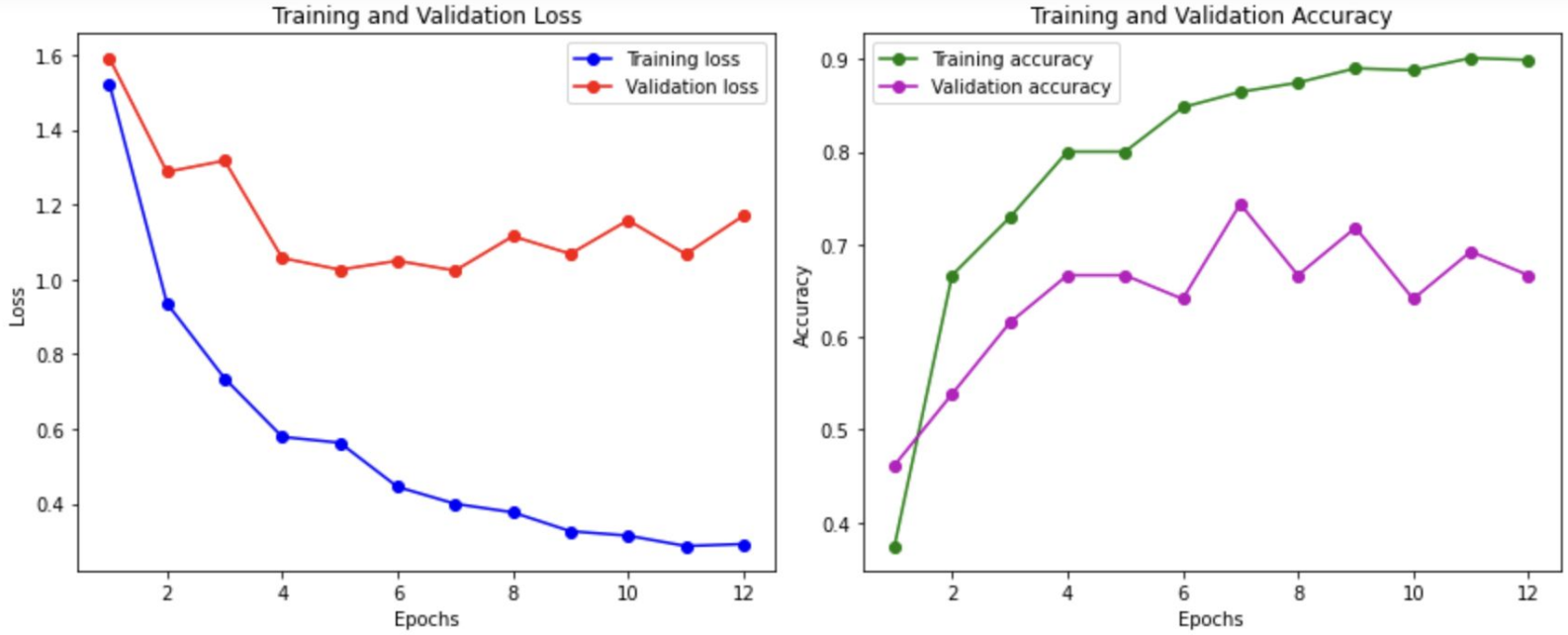
InceptionV3




```python
# 모델 평가
test_model = model.evaluate(test_generator)
print("Test :", test_model)
```

```
40/40 [==============================] - 1s 17ms/step - loss: 0.7373 - accuracy: 0.7500 - precision: 0.8438 - recal
l: 0.6750
Test : [0.7373492121696472, 0.75, 0.84375, 0.675000011920929]
```

MobileNetV3




```
# 모델 평가
test_model = model.evaluate(test_generator)
print("Test :", test_model)
```

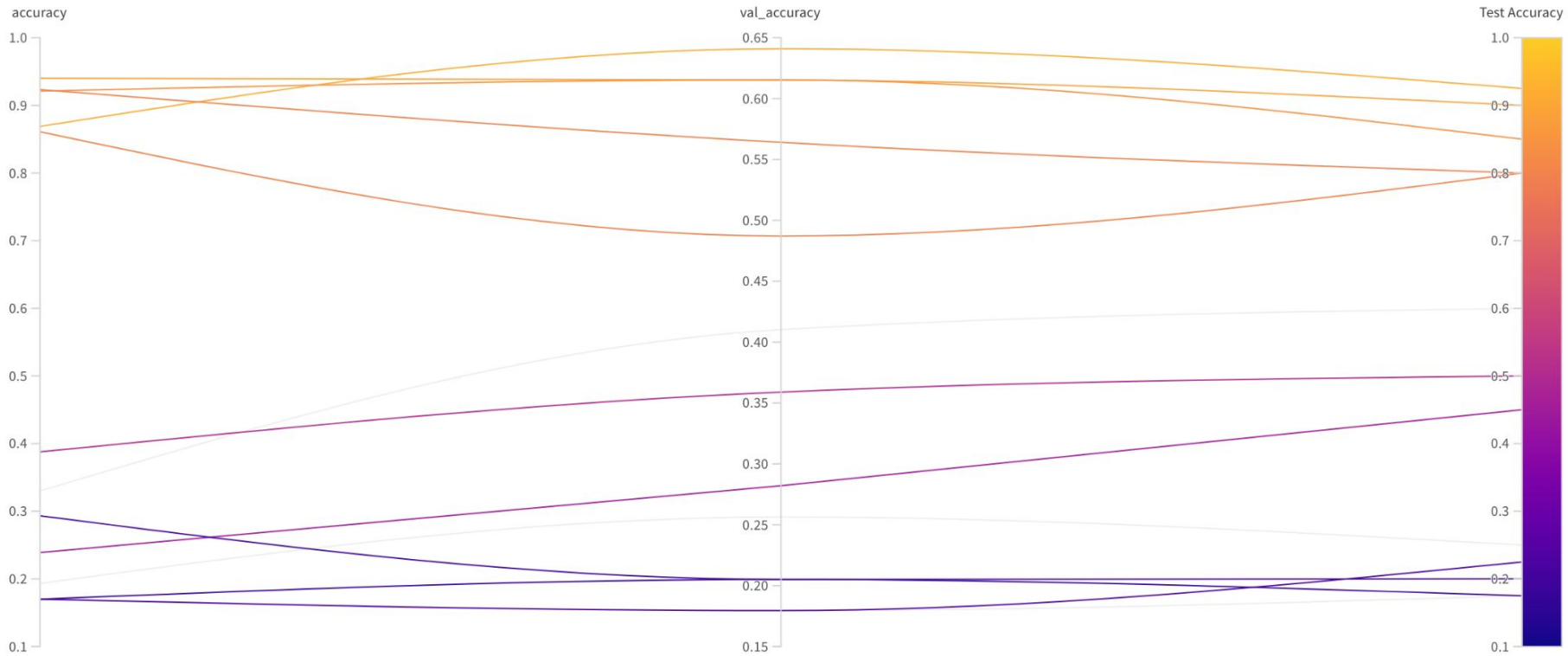
```
40/40 [==============================] - 0s 10ms/step - loss: 0.4086 - accuracy: 0.8500 - precision_1: 0.8947 - rec
all_1: 0.8500
Test : [0.4086295962333679, 0.8500000238418579, 0.8947368264198303, 0.8500000238418579]
```

MobileNetV3

성능

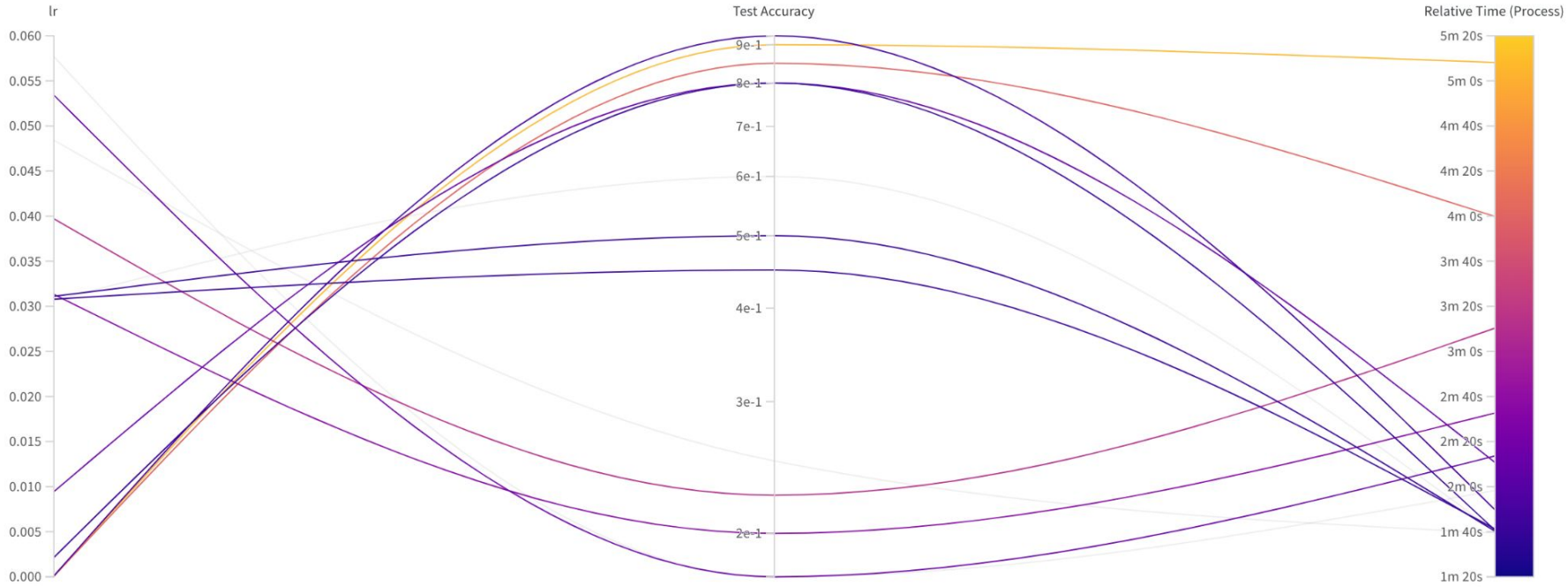
3. W&B Analyze

Test Accuracy



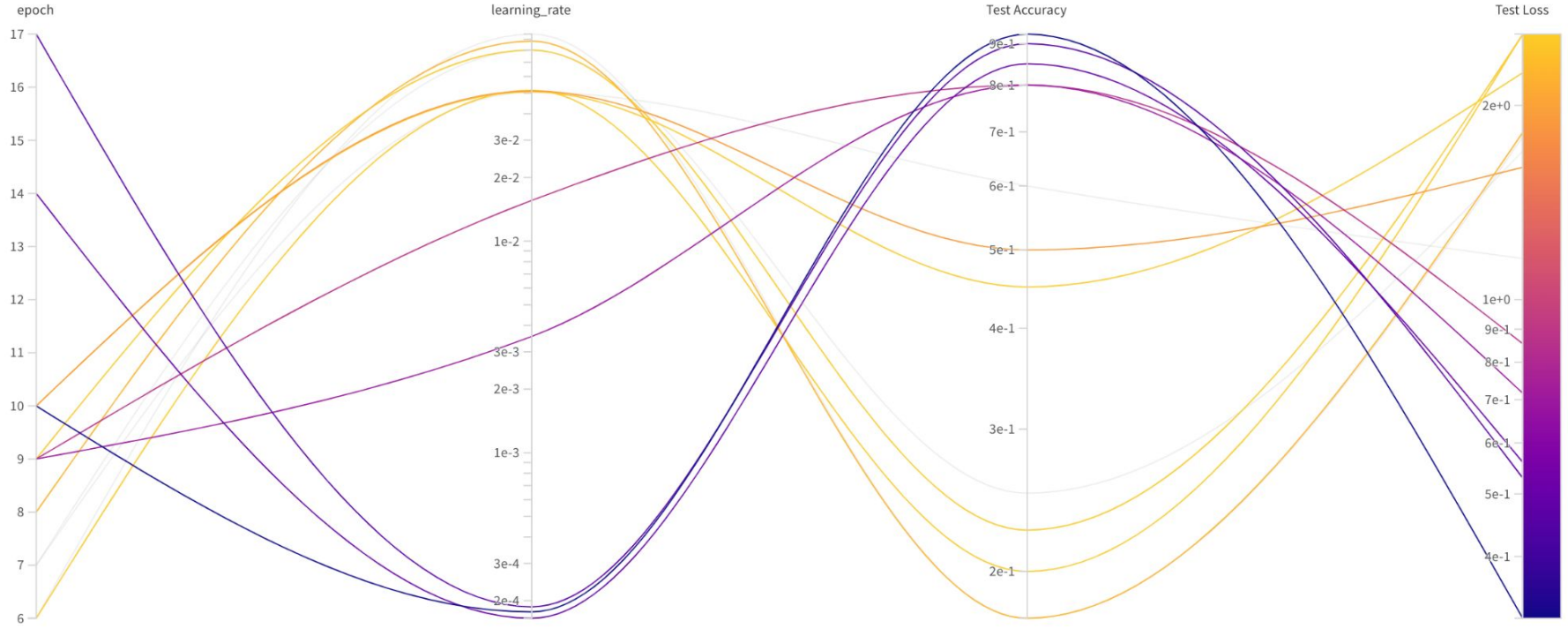
Accuracy → val_accuracy → Test_accuracy

Learning Rate



learning_rate→ Test_accuracy → Real Time (Process)

Learning Rate & Test Accuracy



epoch→ learning_rate→ Test_accuracy → Test_loss

4. Improve Model Performance

하이퍼 파라미터 조절

- Fine Tune (하위 15개 False-저소자 수준 pre-trained)
- Init learning_rate : 0.00018
- Lrs : 최고성능일 때 0.0001074
- earlystopping (patience = 5)
- epochs=20
- train data Augmentation
- Data preprocessing
- Metrics 4ea
- default lr = 0.0001

Data processing (성능 확인)

- 전처리 과정 추가 여부 (2.5% 차이)
- 선명도 (sharped filter 사용)

하이퍼 파라미터 조절 : Fine tunning

```
# Layers position Check, False가 pre-trained model weight load
for i, layer in enumerate(base_model.layers):
  print(i, layer.name, layer.trainable)
```

```
0 input_1 False
1 rescaling False
2 Conv False
3 Conv/BatchNorm False
4 tf.__operators__.add False
5 re_lu False
6 tf.math.multiply False
7 multiply False
8 expanded_conv/depthwise/pad False
9 expanded_conv/depthwise False
10 expanded_conv/depthwise/BatchNorm False
11 re_lu_1 False
12 expanded_conv/squeeze_excite/AvgPool False
13 expanded_conv/squeeze_excite/Conv False
14 expanded_conv/squeeze_excite/Relu False
15 expanded_conv/squeeze_excite/Conv_1 False
16 tf.__operators__.add_1 True
17 re_lu_2 True
18 tf.math.multiply_1 True
```

(하위 15개 False-저소자 수준 pre-trained)
16개를 기준으로 레이어 시각화 후 10~20개를 1개씩 해본 결과 92.5%
Extend_squeeze_excite 레이어 끝단까지 해서 저소자 수준 Tune,
다른 지표 최고 성능 향상

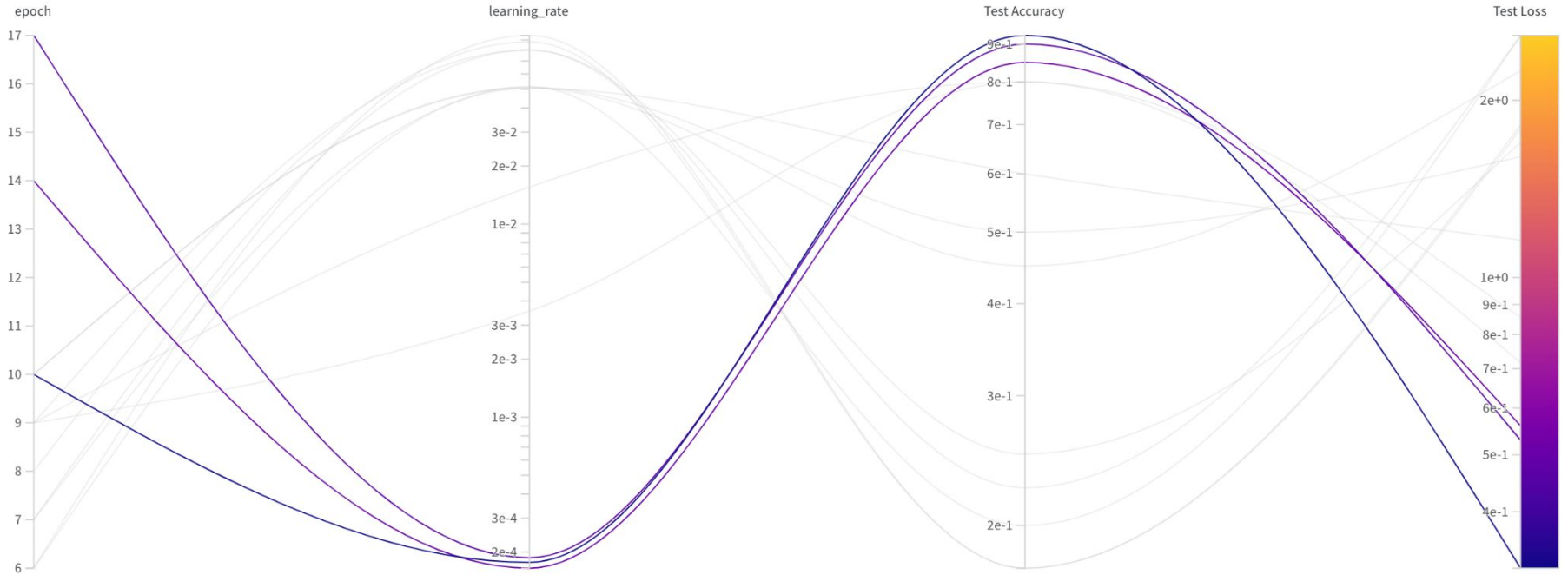
하이퍼 파라미터 조절 : earlyStopping

Earlystopping (patience = 5)

Patience를 늘려 epochs 증가

epochs=20

하이퍼 파라미터 조절 : learning_rate



Learning_rate_scheduler % 6, exp(-0.1)

Lr 0.000018 초기값부터 **(exp(-0.05)** 매우 미세하게 감소시키면서 찾기

W&B로 최적구간 **(0.0002~0.00005)** 확인

오버피팅 경향이 보여 일반화 작업 진행

W&B 분석 후 정리

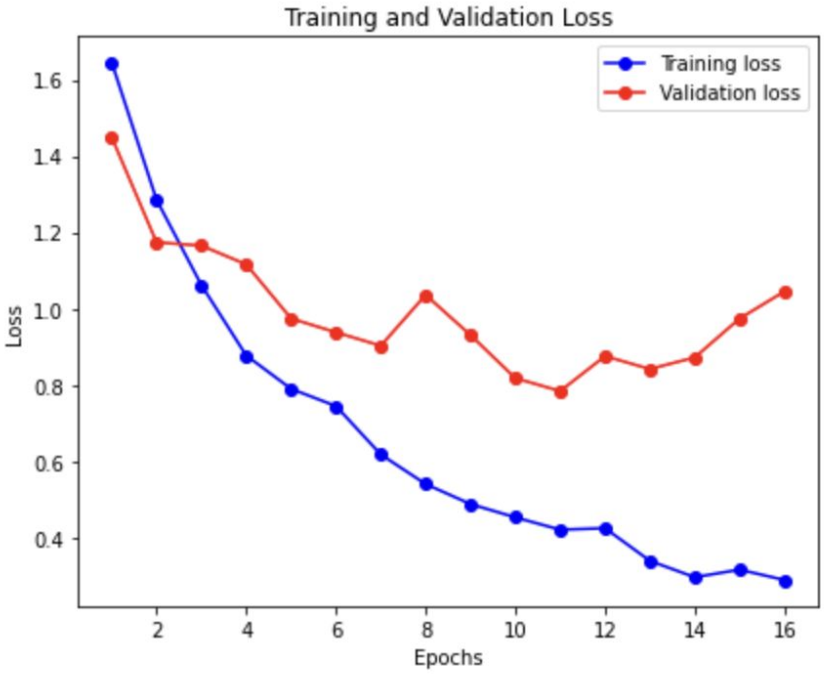
1. Epoch에 따른 val 성능 관계 (에폭, 정확도 비례, 로스 반비례)
2. Lr가 적절한 구간이 있다고 판단
3. Best param 성능 비교
4. Val_acc, test_acc와 밀접한 영향 : 거의 비슷하게 나옴
5. Epoch, test_acc, realtime

W&B 통해 각 하이퍼 파라미터 관계를 확인!

그 내용을 바탕으로 성능향상을 진행!

5. Result & Summary

Final_Result




```
# 모델 평가
test_model = model.evaluate(test_generator)
print("Test :", test_model)
```

```
40/40 [==============================] - 0s 11ms/step - loss: 0.2959 - accuracy: 0.9250 - precision: 0.9487 - recal
l: 0.9250
Test : [0.2958764433860779, 0.925000011920929, 0.9487179517745972, 0.925000011920929]
```

6. Retrospect

딥러닝 과정 이해 집중!

느낀점

1. 프로젝트 진행하면서 많은 중요한 에러 해결방법들을 알게 되어서 좋은 경험이 되었습니다!
2. 역할을 분담하여 협업을 진행하는 장점을 배울 수 있었고, W&B의 신세계를 알게 되었습니다!
3. 딥러닝 세부적인 과정의 전체적인 이해를 할 수 있었고, 팀원들과 프로젝트에 몰입할 수 있어서 좋았습니다!

Thanks!

Any questions?