2조

오버워치 영웅 예측

박지훈 구성윤 김경환

목차

1. 박지훈

efficientnet_b0

2. 구성윤

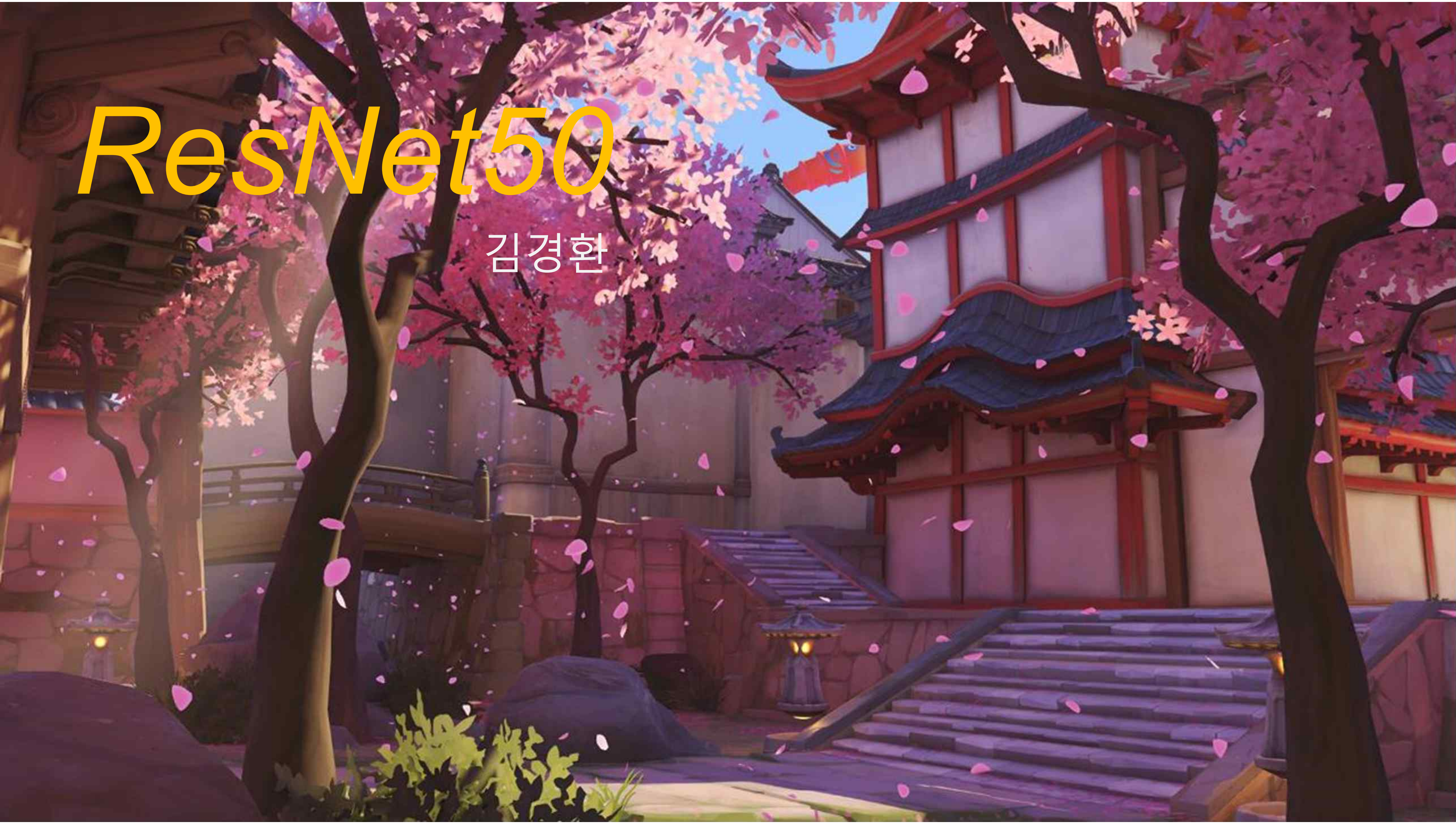
inception_v3

3. 김경환

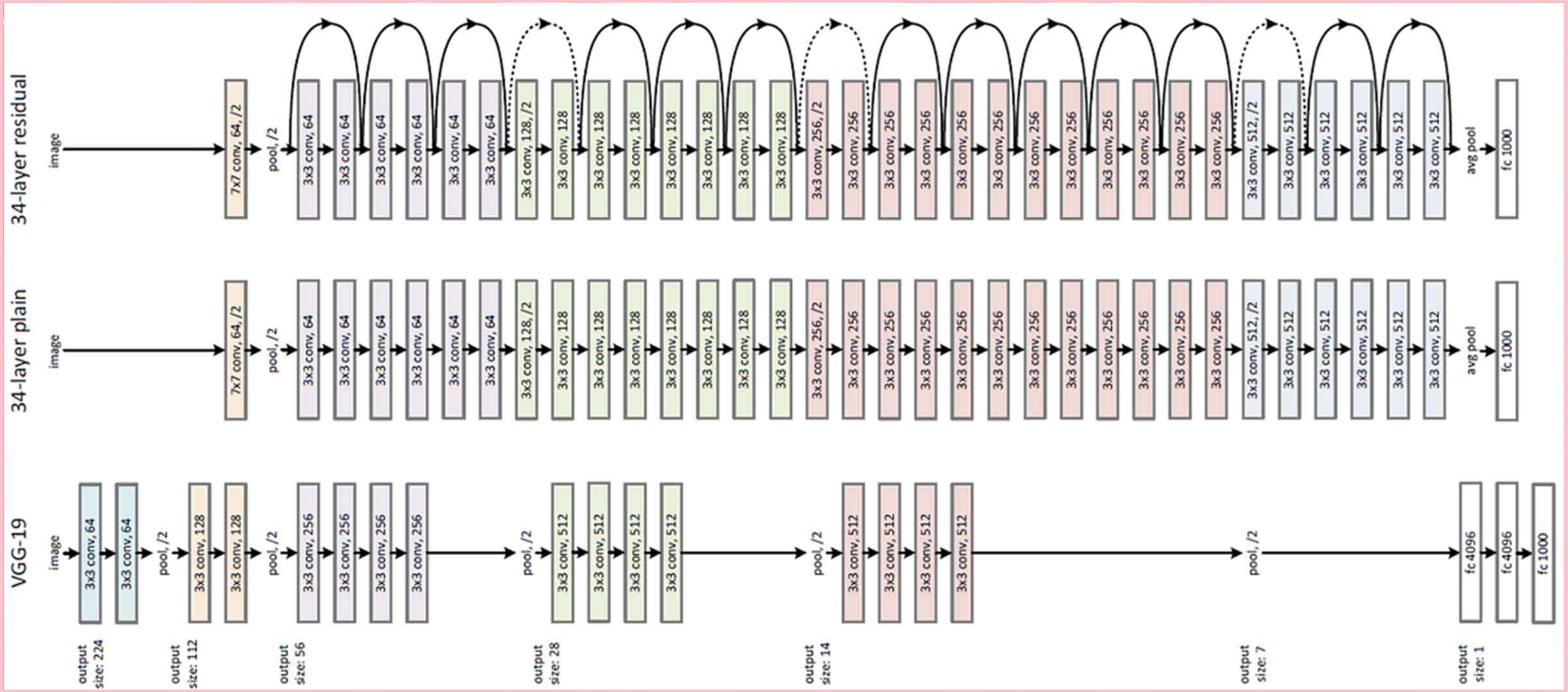
ResNet50

ResNet50

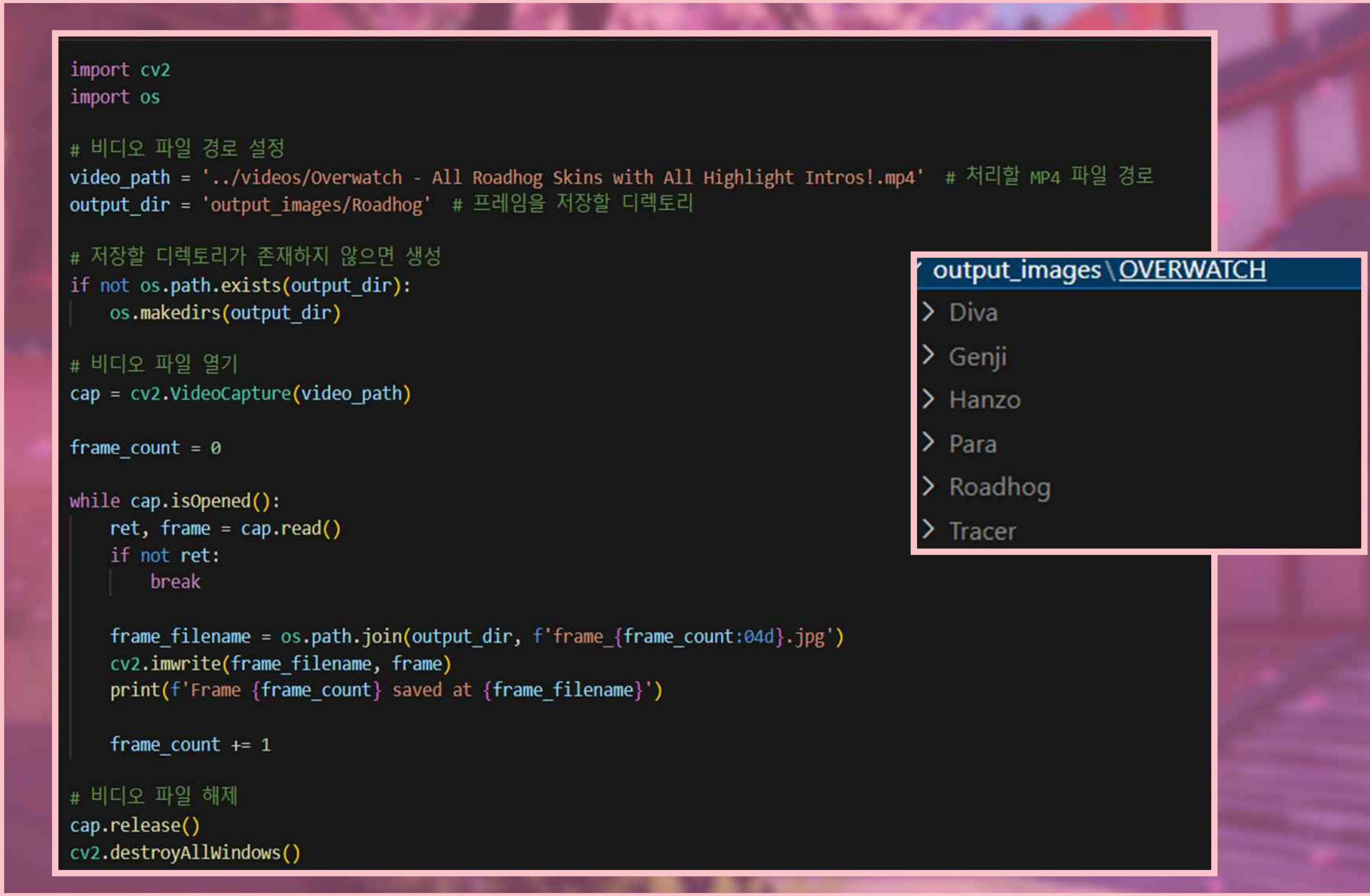
김경환



ResNet50

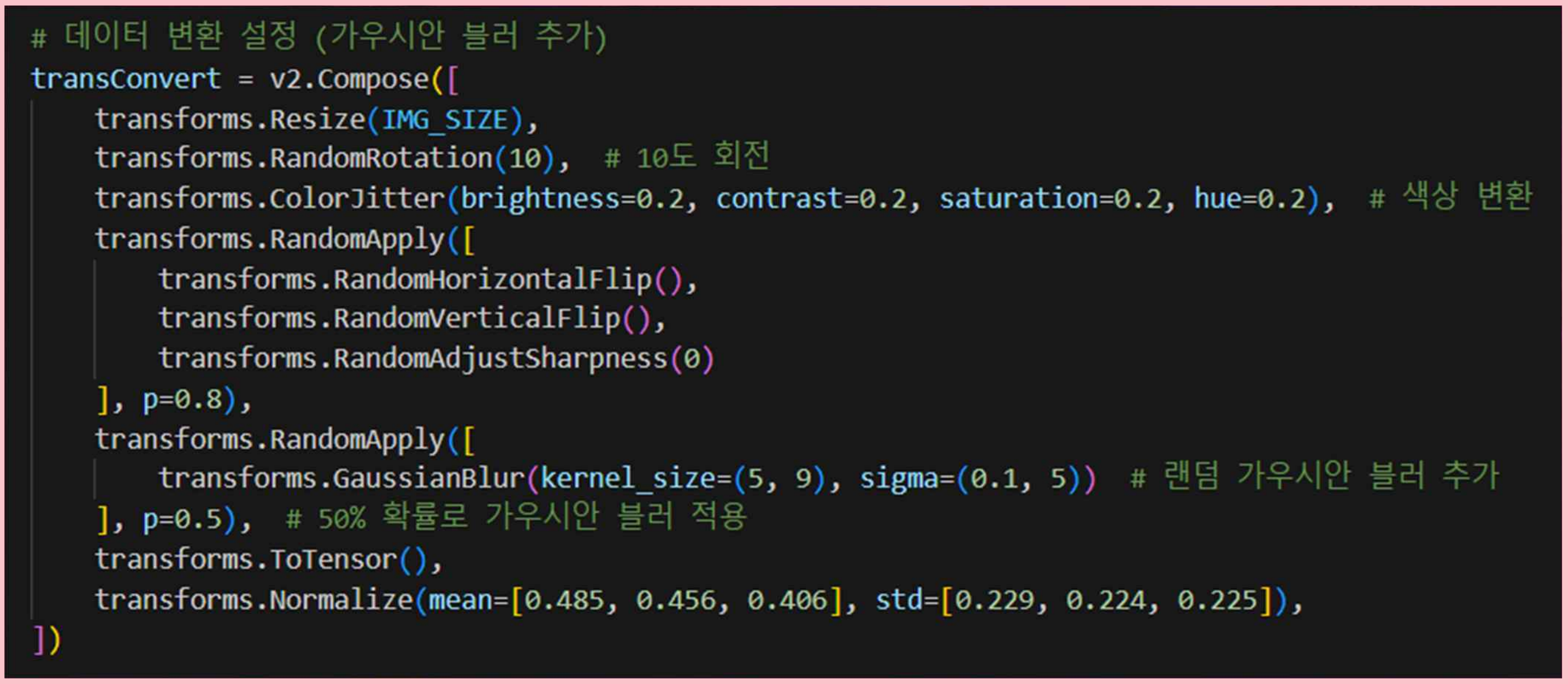

34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000
34-layer plain
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000
VGG-19
image
3x3 conv, 64
3x3 conv, 64
pool, /2
3x3 conv, 128
3x3 conv, 128
pool, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
pool, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
pool, /2
fc 4096
fc 4096
fc 1000
output size: 224
output size: 112
output size: 56
output size: 28
output size: 14
output size: 7
output size: 1

전처리 과정



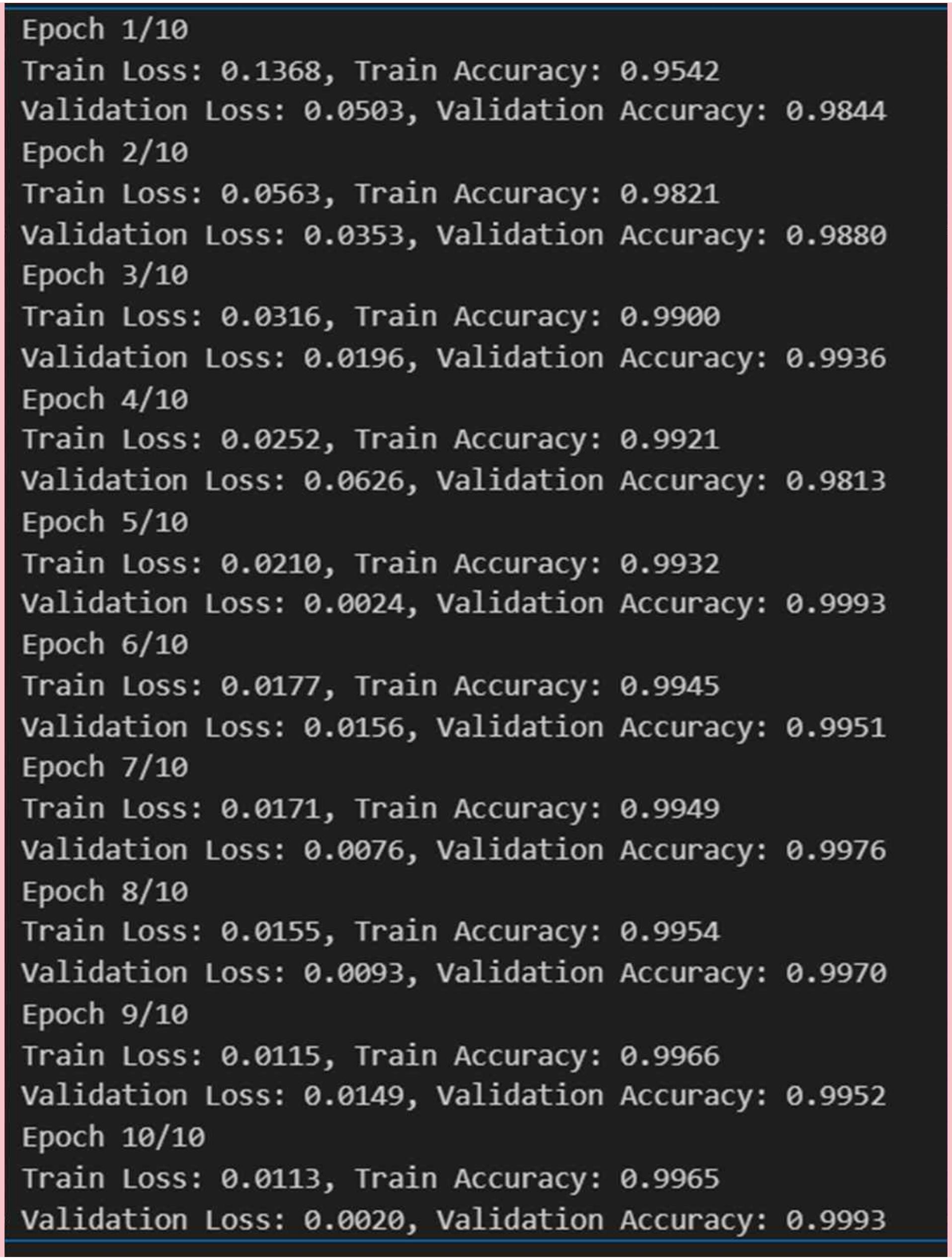
```
import cv2
import os

# 비디오 파일 경로 설정
video_path = '../videos/Overwatch - All Roadhog Skins with All Highlight Intros!.mp4'  # 처리할 MP4 파일 경로
output_dir = 'output_images/Roadhog'  # 프레임을 저장할 디렉토리

# 저장할 디렉토리가 존재하지 않으면 생성
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

# 비디오 파일 열기
cap = cv2.VideoCapture(video_path)

frame_count = 0

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    frame_filename = os.path.join(output_dir, f'frame_{frame_count:04d}.jpg')
    cv2.imwrite(frame_filename, frame)
    print(f'Frame {frame_count} saved at {frame_filename}')

    frame_count += 1

# 비디오 파일 해제
cap.release()
cv2.destroyAllWindows()
```

전처리 과정



```
# 데이터 변환 설정 (가우시안 블러 추가)
transConvert = v2.Compose([
    transforms.Resize(IMG_SIZE),
    transforms.RandomRotation(10),  # 10도 회전
    transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.2),  # 색상 변환
    transforms.RandomApply([
        transforms.RandomHorizontalFlip(),
        transforms.RandomVerticalFlip(),
        transforms.RandomAdjustSharpness(0)
    ], p=0.8),
    transforms.RandomApply([
        transforms.GaussianBlur(kernel_size=(5, 9), sigma=(0.1, 5))  # 랜덤 가우시안 블러 추가
    ], p=0.5),  # 50% 확률로 가우시안 블러 적용
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])
```

학습 후 스



```
Epoch 1/10
Train Loss: 0.1368, Train Accuracy: 0.9542
Validation Loss: 0.0503, Validation Accuracy: 0.9844
Epoch 2/10
Train Loss: 0.0563, Train Accuracy: 0.9821
Validation Loss: 0.0353, Validation Accuracy: 0.9880
Epoch 3/10
Train Loss: 0.0316, Train Accuracy: 0.9900
Validation Loss: 0.0196, Validation Accuracy: 0.9936
Epoch 4/10
Train Loss: 0.0252, Train Accuracy: 0.9921
Validation Loss: 0.0626, Validation Accuracy: 0.9813
Epoch 5/10
Train Loss: 0.0210, Train Accuracy: 0.9932
Validation Loss: 0.0024, Validation Accuracy: 0.9993
Epoch 6/10
Train Loss: 0.0177, Train Accuracy: 0.9945
Validation Loss: 0.0156, Validation Accuracy: 0.9951
Epoch 7/10
Train Loss: 0.0171, Train Accuracy: 0.9949
Validation Loss: 0.0076, Validation Accuracy: 0.9976
Epoch 8/10
Train Loss: 0.0155, Train Accuracy: 0.9954
Validation Loss: 0.0093, Validation Accuracy: 0.9970
Epoch 9/10
Train Loss: 0.0115, Train Accuracy: 0.9966
Validation Loss: 0.0149, Validation Accuracy: 0.9952
Epoch 10/10
Train Loss: 0.0113, Train Accuracy: 0.9965
Validation Loss: 0.0020, Validation Accuracy: 0.9993
```