

고급 컴퓨터 비전

주요 내용

- 합성곱 신경망의 주요 활용 분야(컴퓨터 비전)
 - 이미지 분류
 - 이미지 분할
 - 객체 탐지
- 합성곱 신경망 기본 아키텍처
 - 잔차 연결
 - 배치 정규화
 - 채널 분리 합성곱

컴퓨터 비전 주요 과제

Single-label multi-class classification



- Biking
- Running
- Swimming

Multi-label classification

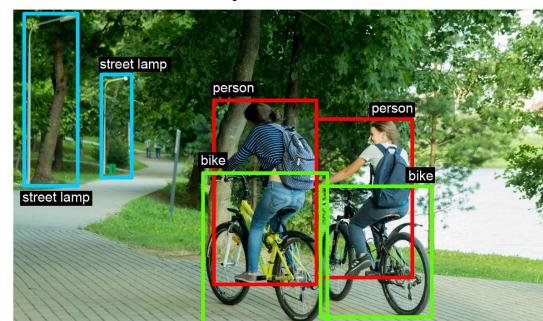


- | | |
|--|--|
| <input checked="" type="checkbox"/> Bike | <input checked="" type="checkbox"/> Tree |
| <input checked="" type="checkbox"/> Person | <input type="checkbox"/> Car |
| <input type="checkbox"/> Boat | <input type="checkbox"/> House |

Image segmentation

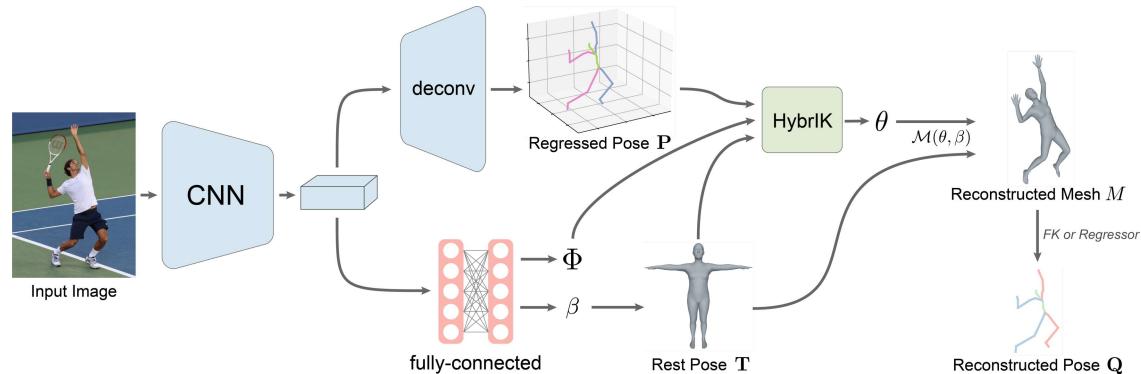


Object detection



기타 다양한 활용법

- 이미지 유사도 측정(image similarity scoring)
- 키포인트 탐지(keypoint detection)
- 자세 추정(pose estimation)
- 3D 메쉬 추정(3D mesh estimation)

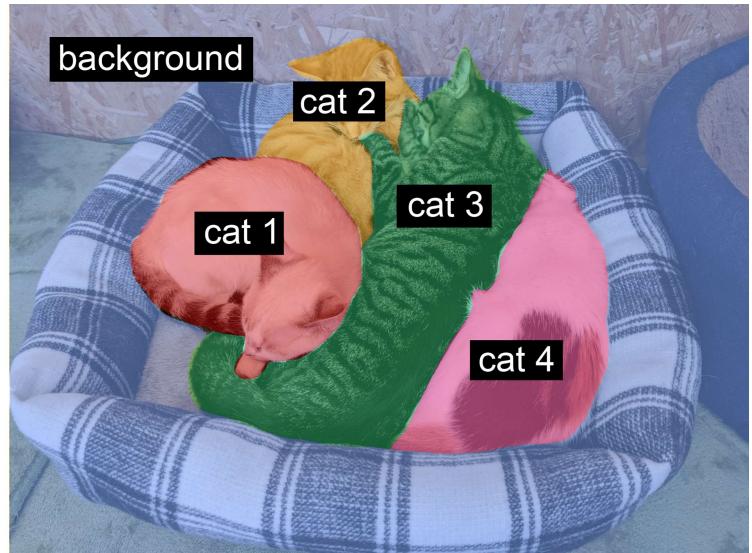


객체 탐지

- 작동원리 설명: 핸즈온 머신러닝 14장
- 기초 활용법: RetinaNet 활용 객체 탐지
- 주요 활용 예제: YOLOv5

이미지 분할

- 시맨틱 분할(semantic segmentation)
- 인스턴스 분할(instance segmentation)

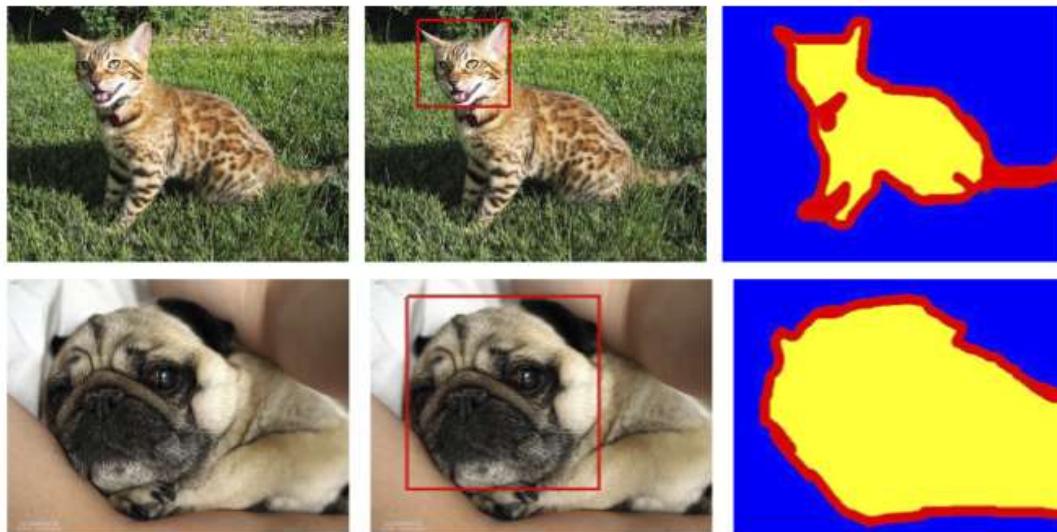


Oxford-IIIT 애완동물 데이터셋

- 데이터셋 크기: 7,390
- 클래스(범주) 개수: 37
- 클래스별 사진 수: 약 200 장
- 사진별 라벨: 종과 품종, 머리 표시 경계상자, 트라이맵 분할^{trimap segmentation} 마스크 등 4 종류로 구성

트라이맵 분할 마스크

- 원본 사진과 동일한 크기의 칼라 사진
- 1, 2, 3 셋 중에 하나의 값만 사용
 - 1: 동물의 몸에 해당하는 픽셀
 - 2: 배경에 해당하는 픽셀
 - 3: 동물과 배경을 구분하는 경계에 해당하는 픽셀



이미지 분할 모델 구성: 다운 샘플링

```
inputs = keras.Input(shape=img_size + (3,))
x = layers.Rescaling(1./255)(inputs)

x = layers.Conv2D(64, 3, strides=2, activation="relu", padding="same")(x)
x = layers.Conv2D(64, 3, activation="relu", padding="same")(x)
x = layers.Conv2D(128, 3, strides=2, activation="relu", padding="same")(x)
x = layers.Conv2D(128, 3, activation="relu", padding="same")(x)
x = layers.Conv2D(256, 3, strides=2, padding="same", activation="relu")(x)
x = layers.Conv2D(256, 3, activation="relu", padding="same")(x)
```

이미지 분할 모델 구성: 업 샘플링

```
x = layers.Conv2DTranspose(256, 3, activation="relu", padding="same")(x)
x = layers.Conv2DTranspose(256, 3, activation="relu", padding="same",
strides=2)(x)
x = layers.Conv2DTranspose(128, 3, activation="relu", padding="same")(x)
x = layers.Conv2DTranspose(128, 3, activation="relu", padding="same",
strides=2)(x)
x = layers.Conv2DTranspose(64, 3, activation="relu", padding="same")(x)
x = layers.Conv2DTranspose(64, 3, activation="relu", padding="same", strides=2)
(x)

outputs = layers.Conv2D(num_classes, 3, activation="softmax", padding="same")
(x)

model = keras.Model(inputs, outputs)
```

input_1 (InputLayer)	[None, 200, 200, 3]	0
rescaling (Rescaling)	(None, 200, 200, 3)	0
conv2d (Conv2D)	(None, 100, 100, 64)	1792
conv2d_1 (Conv2D)	(None, 100, 100, 64)	36928
conv2d_2 (Conv2D)	(None, 50, 50, 128)	73856
conv2d_3 (Conv2D)	(None, 50, 50, 128)	147584
conv2d_4 (Conv2D)	(None, 25, 25, 256)	295168
conv2d_5 (Conv2D)	(None, 25, 25, 256)	590080
conv2d_transpose (Conv2DTr)	(None, 25, 25, 256)	590080
conv2d_transpose_1 (Conv2DTr)	(None, 50, 50, 256)	590080
conv2d_transpose_2 (Conv2DTr)	(None, 50, 50, 128)	295040
conv2d_transpose_3 (Conv2DTr)	(None, 100, 100, 128)	147584
conv2d_transpose_4 (Conv2DTr)	(None, 100, 100, 64)	73792
conv2d_transpose_5 (Conv2DTr)	(None, 200, 200, 64)	36928
conv2d_6 (Conv2D)	(None, 200, 200, 3)	1731

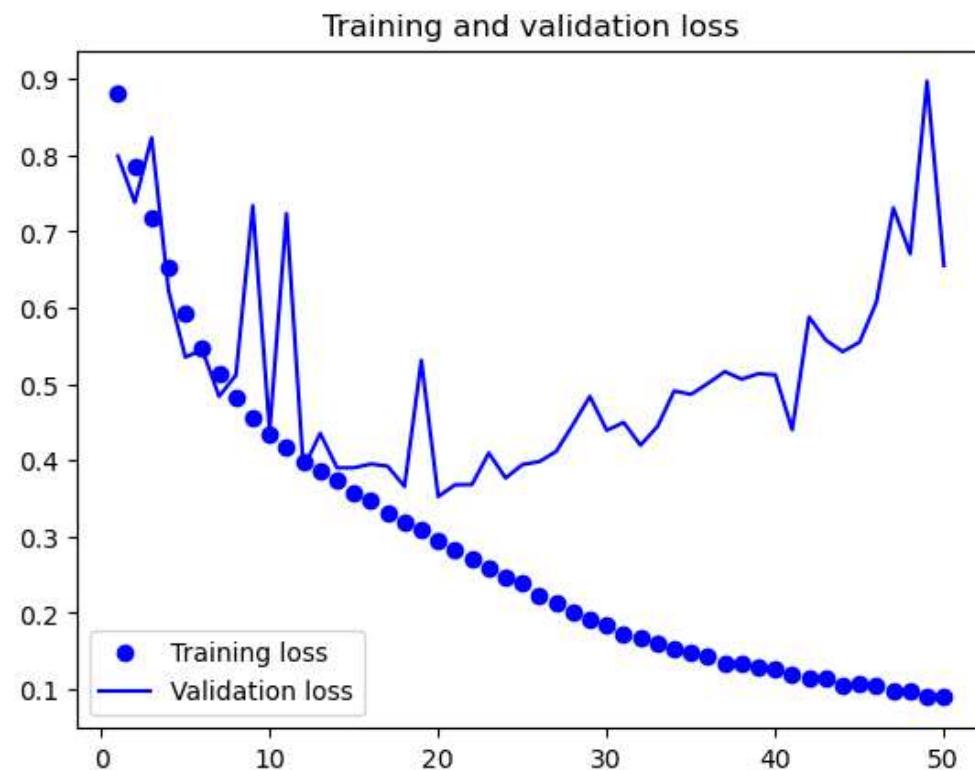
모델 컴파일과 훈련

```
model.compile(optimizer="rmsprop",
              loss="sparse_categorical_crossentropy")

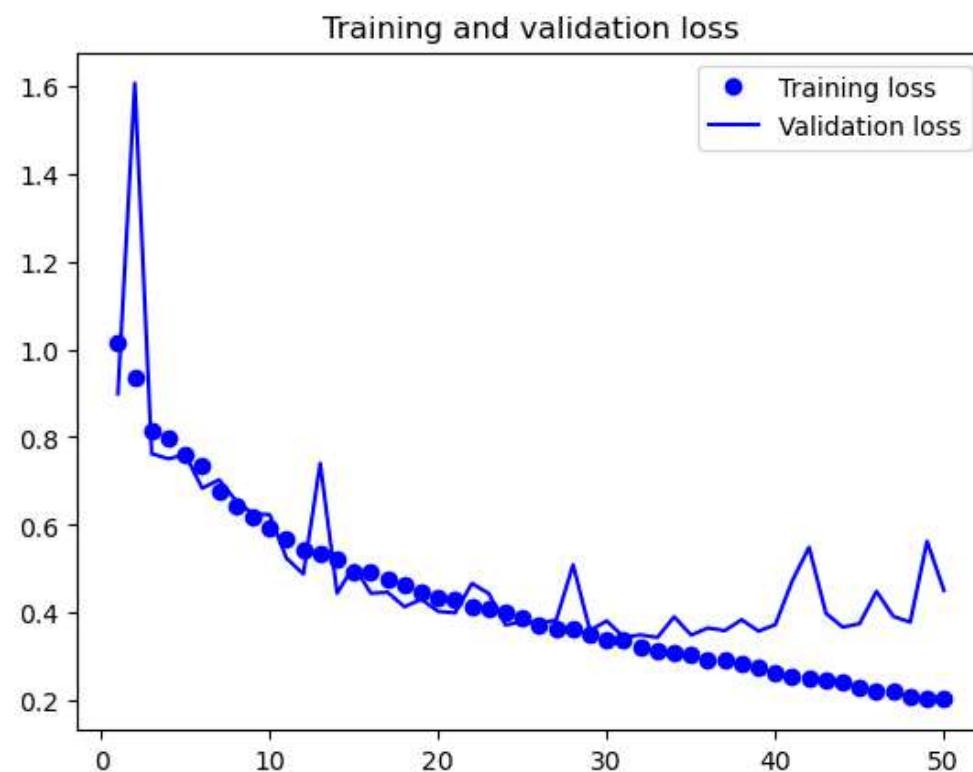
callbacks = [
    keras.callbacks.ModelCheckpoint("oxford_segmentation",
                                    save_best_only=True)
]

history = model.fit(train_input_imgs, train_targets,
                     epochs=50,
                     callbacks=callbacks,
#                         batch_size=64, # 고성능 GPU 활용
#                         batch_size=16, # 저성능 GPU 활용
                     validation_data=(val_input_imgs, val_targets))
```

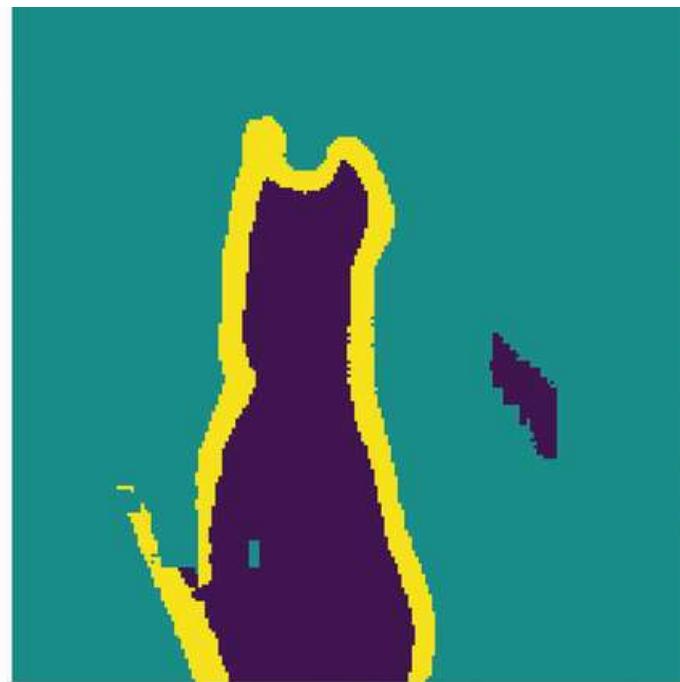
훈련 결과: batch_size = 16



훈련 결과: batch_size = 64



모델 예측



CNN 아키텍처 주요 유형

계속 이어짐...

