8-b 정리

```
import os, shutil, pathlib

original_dir = pathlib.Path("train")
new_base_dir = pathlib.Path("cats_vs_dogs_small")

def make_subset(subset_name, start_index, end_index):
    for category in("cat", "dog"):
        dir = new_base_dir / subset_name / category
        os.makedirs(dir)
        fnames = [f"{category}.{i}.jpg"
                  for i in range(start_index, end_index)]
        for fname in fnames :
            shutil.copyfile(ser=original_dir / fname,
                            dis=dir / fname)

make_subset("train", start_index=0,end_index=1000)
make_subset("validation", start_index=1000,end_index=1500)
make_subset("test", start_index=1500,end_index=2500)
# 강아지 2500, 고양이 2500
```

모델 만들기

맵 채널수는 점진적으로 증가한다 (32 ~ 256)

맵의 크기는 감소한다. (180x180 ~ 7×7)

```
from tensorflow import keras
from tensorflow.keras import layers

inputs = keras.Input(shape=(180,180,3))

x = layers.Rescaling(1./255)(inputs)
x = layers.Conv2D(filters = 32, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters = 64, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters = 128, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Conv2D(filters = 256, kernel_size=3, activation="relu")(x)
x = layers.MaxPooling2D(pool_size=2)(x)
x = layers.Flatten()(x)
outpus = layers.Dense(1, activation="sigmoid")(x)
model = keras.Model(inputs=inputs, outputs = outputs)
```

1. 이미지가 3원색이기 때문에 Conv2D 과정에서 R, G, B 각각 따로 분리한 뒤 filters를 32번 적용 하고 그 값을 다시 merge해준다.(이과정은 책에도 생략되어있다)
2. Flatten을 통해 입력 값은 3D텐서이므로 이를 Densly Connected Network에 주입하기 위해서 1D텐서로 변환한다
3. 강아지 or 고양이를 구분하는 이진 분류이므로 마지막 층은 크기가 1이고 sigmoid사용

모델 훈련 설정하기

```
model.compile(loss="binary_crossentropy",
              optimizer="rmsprop",
              metrics = ["accuracy"])
```

모델의 마지막이 하나의 시그모이드 유닛이므로 이진 크로스 엔트로피를 사용한다.

```
from tensorflow.keras.utils import image_dataset_from_directory

train_dataset = image_dataset_from_directory(
    new_base_dir / "train"
    image_size = (180,180)
    batch_size=32
)

# ... validation, test도 동일
```

image_dataset_from_directory 유틸리티 이용하기

image_dataset_from_directory를 이용해서 데이터를 편하게 전처리한다.

1. 각기 다른 사이즈의 이미지를 읽어서 내가 지정한(180,180) 사이즈에 맞게 늘리거나 줄여준다.
2. train, validation, test 이미지들을 카테고리별로 알맞게 0과 1의 레이블을 할당해준다.

Dataset을 사용하여 모델 훈련하기 - Callback 사용하기

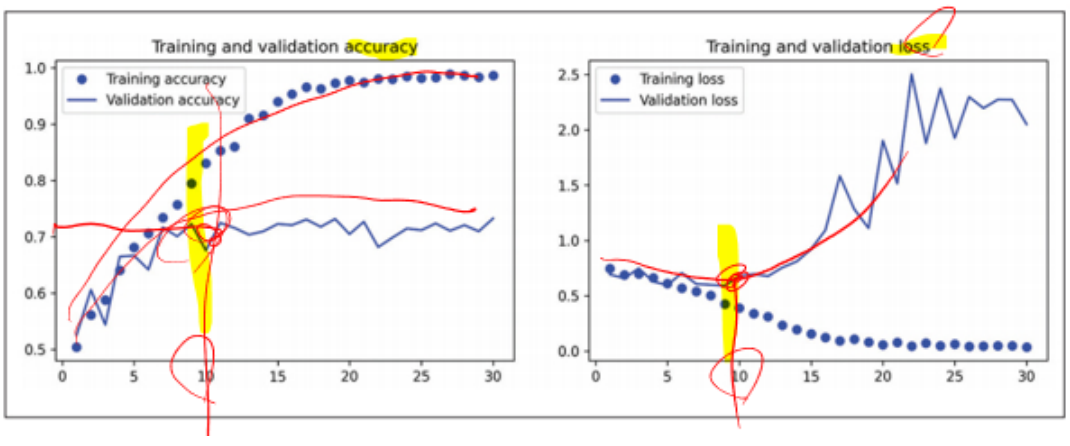
```
callbacks = [
    keras.callbacks.ModelCheckpoint(
        filepath = "convnet_from_scratch.keras". # 저장할 파일 명
        save_best_only=True, # val_loss가 가장 좋을 때(값이 최소일 때)만 저장한다.
        monitor="val_loss"
        # validation_loss를 모니터링해서 가장 좋을 때(최소일떄)를 저장한다.
    )
]

history = model.fit(
    train_dataset,
    epochs = 30,
    validation_data = validation_dataset,
    callbacks=callbacks)
```

- save_best_only = False최고 상태만 저장하는게 아니라, 에포크1, 2, 3 ... 에포크 n 모든 상태를 다 저장한다.

환련 과정의 정확도와 손실 그래프 그리기

코드는 생략

▼ 그림 8-9 간단한 컨브넷의 훈련과 검증 지표



훈련 정확도는 선형적으로 100%에 달하지만 검증 정확도는 75가 최고(epochs = 10)
검증 손실 또한 열 번의 에포크에서 최솟값이만 훈련 손실에서는 점점 감소한다.

이는 비교적 적은 데이터로 인하여 생긴 과대적합이 원인이다.

```
# 가장 좋은 val_loss를 저장해둔 모델
test_model = keras.models.load_model("convnet_from_scratch.keras")
test_loss, test_acc = test_model.evaluate(test_dataset)
print(f"테스트 정확도: {test_acc.3f}")
```

테스트 정확도에서는 69.5 %가 나온다