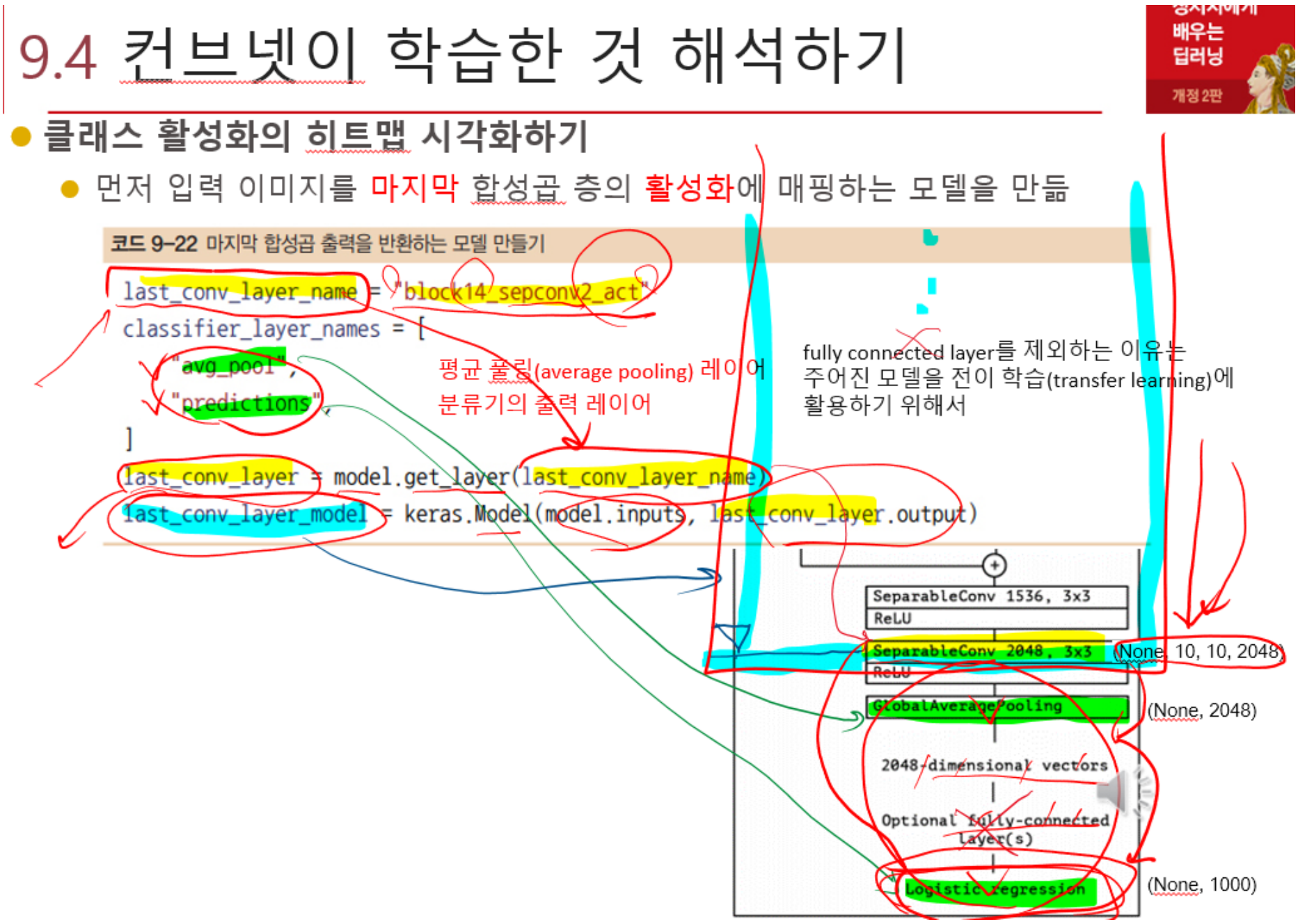
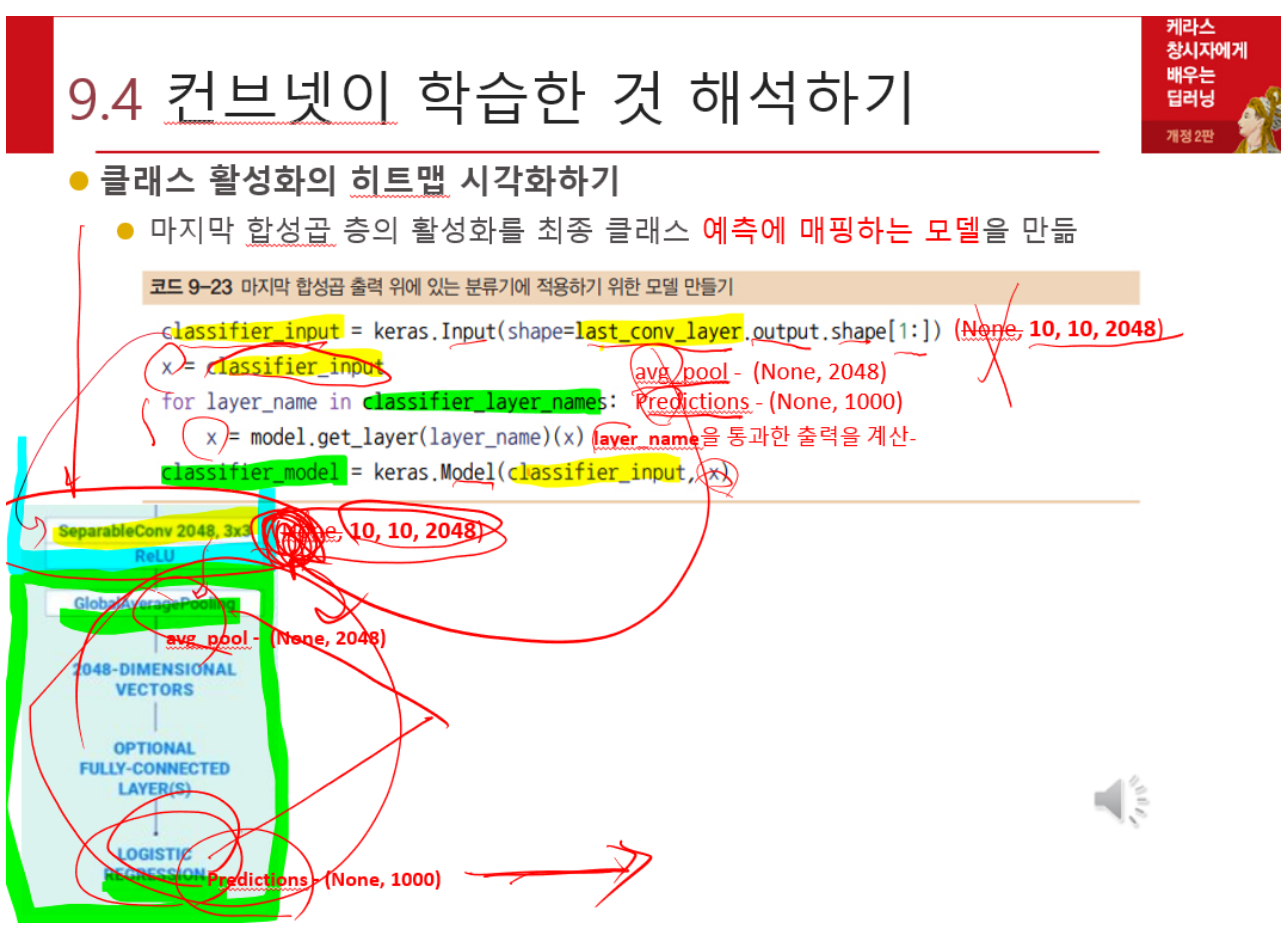
9 - e 클래스 활성화 히트맵 시각화

GRAD_CAM 처리 과정



“block14_sepcon2_act” 는 Xception 합성곱 기반에 있는 층 이름이다.

model.inputs를 input으로, block14_sepconv2_act.output을 output으로 사용하는 모델 **last_conv_layer_model**



classifier_model는 사용시 예측 확률을 얻을 수 있다.

이때 classifier는 평균 풀링레이어와 분류기의 출력 레이어를 사용한다.

9.4 컨브넷이 학습한 것 해석하기

- 클래스 활성화의 히트맵 시각화하기
  - 마지막 합성곱 층의 활성화에 대한 최상위 예측 클래스의 그레이디언트를 계산

코드 9-24 최상위 예측 클래스의 그레이디언트 계산하기

```
import tensorflow as tf

with tf.GradientTape() as tape:
    last_conv_layer_output = last_conv_layer_model(img_array)
    tape.watch(last_conv_layer_output)
    preds = classifier_model(last_conv_layer_output)
    top_pred_index = tf.argmax(preds[0])
    top_class_channel = preds[:, top_pred_index]

grads = tape.gradient(top_class_channel, last_conv_layer_output)
```

gradient를 계산하기 위한 컨텍스트 매니저

마지막 합성곱 층의 활성화를 계산하고 그레이디언트 테이프로 감시합니다.

텐서 추적

Img_array에 대한 1000개 클래스별 예측 확률

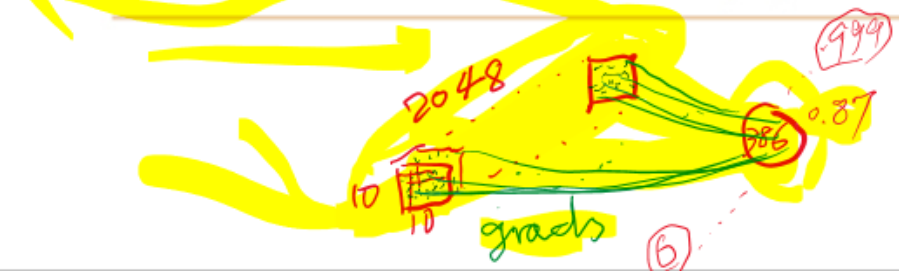
최상위 예측 클래스에 해당하는 활성화

top_pred_index(0, 386)에 해당되는 모든 정보

tf.Tensor ([0.8699399], shape=(1,), dtype=float32)

(1, 10, 10, 2048)

마지막 합성곱 층의 출력 특성 맵에 대한 최상위 예측 클래스의 그레이디언트를 계산합니다.



last_conv_layer_output은 img_array에 대한 마지막 합성곱 층 특성들이 나오고
이렇게 나온 출력(합성곱 층 특성들)을 classifier_model에 다시 입력으로 넣어주면 Img_array에 대한 1000개 클래스 별 예측 확률이 나온다.

이 예측 확률 중에서 가장 높은 값을 가지는 클래스 인덱스를 찾는다. tp.argmax(preds[0])
찾은 인덱스로 가장 높은 확률을 갖는 채널(클래스)를 찾는다. t_c_c = preds[:, top_pred_index]

모든 합성곱 층 출력 특성 맵에 대한 가장 높은 확률을 갖는 채널(클래스)의 미분값을 계산한다.

이는 (1,10,10,2048)의 shape을 가지며, grads라는 변수에 저장한다.

즉, grads에는 미분값이 들어있다.

9.4 컨브넷이 학습한 것 해석하기

- **클래스 활성화의 히트맵 시각화하기**
  - 그레이디언트 텐서를 평균하고 중요도 가중치를 적용하여 클래스 활성화 히트맵을 만듦

코드 9-25 그레이디언트를 평균하고 채널 중요도 가중치 적용하기

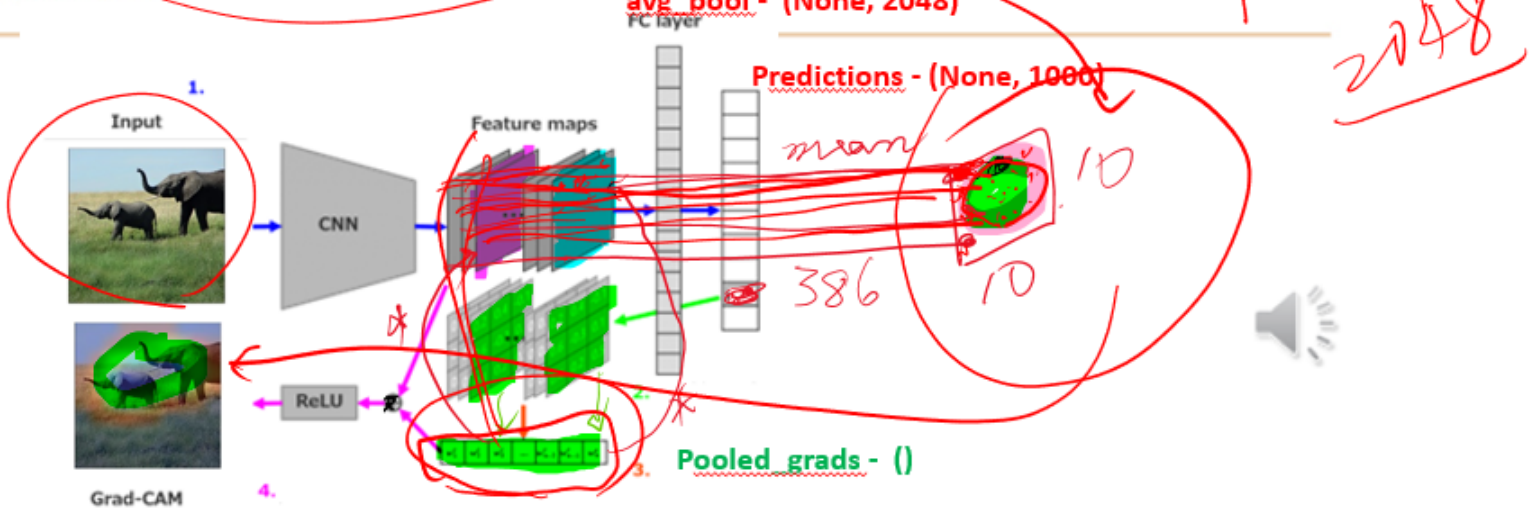
```
pooled_grads = tf.reduce_mean(grads, axis=(0, 1, 2)).numpy()
last_conv_layer_output = last_conv_layer_output.numpy()[0]
for i in range(pooled_grads.shape[-1]):
    last_conv_layer_output[:, :, i] *= pooled_grads[i]
heatmap = np.mean(last_conv_layer_output, axis=-1)
```

(1, 10, 10, 2048)

이 벡터의 각 원소는 어떤 채널에 대한 그레이디언트의 평균 강도입니다. 최상위 예측 클래스에 대한 각 채널의 중요도를 정량화한 것입니다.

마지막 합성곱 층의 출력에 있는 각 채널에 '채널의 중요도'를 곱합니다.

만들어진 특성 맵을 채널별로 평균하면 클래스 활성화 히트맵이 됩니다.



(1, 10 ,10, 2048) 크기의 미분값을 0,1,2축을 기준으로 평균을 구한다.
1은 그냥 무시하고 10 x 10 이미지 전체에 대한 평균을 구한다는 뜻으로 총 10x10 으로 이루어진 2048개 특성 맵의 평균을 구하는 것과 같다. → 구한 평균은 총 2048개가 된다.

2048개의 평균을 각각(2048개) 10 x 10 이미지에 다시 곱해준다.

이러면(1은 무시) 10 x 10 의 출력 특성 맵에 미분값(채널 중요도)이 곱해지게 된다는 뜻이다.

이를 다시 axis -1를 기준으로 평균을 구한다. → 각 채널별의 평균을 구한다.

결과로 클래스에 대해 각 위치 중요도 정보를 가진 클래스 활성화 비트맵이 나온다. (10x10 크기)

결과는 10x10의 클래스 활성화 비트맵이 나오게 된다.

클래스 활성화 히트맵(CAM) 시각화

시각화를 위해서 히트맵을 0과 1사이로 정규화 → 히트 맵과 원본 그림을 겹친다. 이때 히트 맵에 투명도를 40%로 설정한다. (100%로 합치면 원본이 안보이니까)