참고

https://greeksharifa.github.io/computer%20vision/2022/03/01/EfficientNet/

모델의 성능을 높이는 방법

1. Network의 depth를 깊게 만드는 방법
2. Channel width를 늘리는 방법
3. 데이터의 해상도를 올리는 방법

efficientnet이란?

- 모델의 성능을 높이는 방법에 관여하는 depth, width, resolution을 compound coefficient를 통해 상관 관계를 가지는 식을 통해 효율적인 조합을 찾는다.
- 기존의 모델들보다 적은 FLOPS(계산량)으로 더 좋은 성능을 낸다.

Efficientnet B0 ~ B7의 차이

- B0에서 b7으로 갈수록 Compound scaling을 통한 데이터의 사이즈가 커진다.

$$\begin{aligned} \text{depth: } & d = \alpha^\phi \\ \text{width: } & w = \beta^\phi \\ \text{resolution: } & r = \gamma^\phi \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned} \tag{3}$$

-
- Efficientnet b0의 경우 $\alpha = 1, \beta = 1$로 설정되어있으며 매개 변수는 4.5M

Depth coeff를 1에서 1.06으로 늘린 이유

- Depth, width, resolution을 높일수록 정확도가 올라간다.
- 위의 식에서 보듯 width나 resolution을 올리면 coeff의 제곱만큼 매개변수가 많아지기 때문에 조금만 수정하여도 매개변수 5M이 넘었기 때문에 5M 이하의 최대의 매개변수를 가지게 하기 위해 상승폭이 적은 Depth coeff를 1.06으로 증가시켰다.

Dropout rate와 drop connect rate의 차이

- 두 매개변수 동일하게 과적합문제를 해결하기 위해 수정
- Dropout rate는 layer에 포함된 가중치를 확률적으로 포함되는 정도를 조정하며, drop connect layer는 학습에 사용되는 계층의 하위 집합을 무작위로 삭제함으로써 확률적으로 다른 깊이로 학습을 진행하게 한다.

Optimizer로 SGD를 선택한 이유

- Adam과 NAdam을 사용했을 때 train acc가 97~98%로 높았지만 val acc가 91~92%로 과적합 문제 발생
- SGD가 Adam보다 일반화의 성능이 더 좋다는 자료를 참고하였고 결과로 train acc는 96~97%로 조금 낮았지만, val acc 93~94%로 더 높은 결과를 얻을 수 있었다.

Lr 변경 구간 설정 이유

- 직접 모델은 훈련하면서 성능 향상이 안 이루어지는 구간을 관찰하고 multistep scheduler의 mile stone에 삽입함으로써 20, 30, 35, 40, 43, 45, 47 epoch를 진행할 때 lr수정이 이루어지도록 하였으며 훈련의 끝으로 갈수록 lr이 자주 감소되게 지정하였다.

데이터 전처리 및 증강

- 데이터 전처리는 이미지 사이즈는 수정하지 않았으며, 정규화는 가이드 코드의 계산값을 그대로 사용
- 증강으로는 초기에 데이터를 충분히 증강시키기 위하여 좌우반전, 상하반전, 45도 회전, zoom-in, 색 변환을 하였지만 낮은 성능을 내었다.
- 테스트셋에 있는 이미지를 보았을 때 상하반전된 이미지나 심하게 기울어진 이미지가 없었기에 증강 과정에서 상하반전, 색 변환, zoom-in을 제거하고, 회전 비율을 15도로 수정하였으며 더 좋은 성능을 내었다.