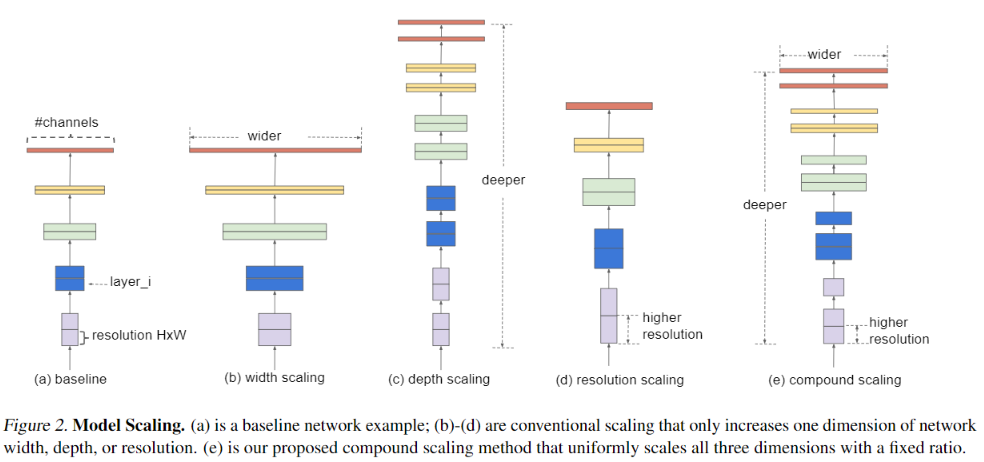
**Downscaling Deep ConvNet**

2014-17952 정재철

2015-19099 황보준호

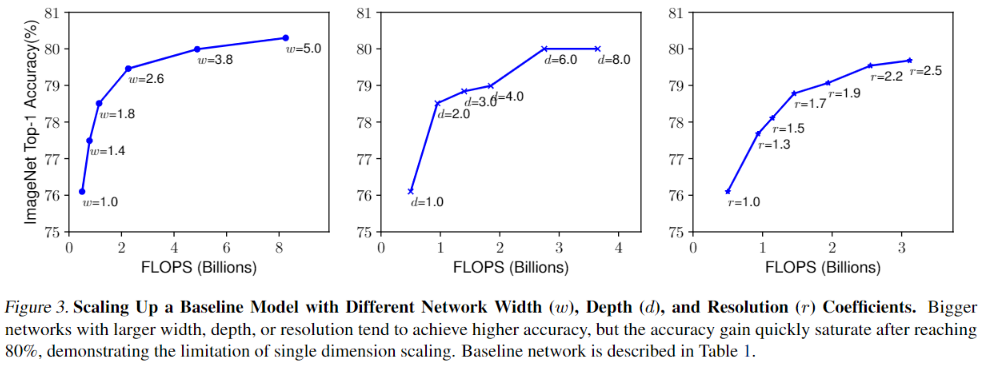
**1. Introduction**

ConvNet의 스케일은 세 가지 요인(width, depth, resolution, 이하 W, D, R)을 조절함으로서 키울 수 있다. Baseline model을 선택한 후 스케일을 키우면서 성능이 얼마나 변하는지 측정한 연구가 여럿 존재한다. 선행 연구에서는 W, D, R 중 하나만 키우면서 퍼포먼스를 측정하였다. 한편, 이번 프로젝트의 참고 논문에서는 세 가지 요인을 전부 고려하면서 어떻게 scaling up 해야 리소스(파라미터 수, FLOP 수) 측면에서 최적의 모델을 찾을 수 있는지에 대한 이론을 제시하고 실험 결과를 보고했다. 이를 위해 EfficientNet이라는 baseline model을 제작했고, 이들이 사용한 scaling up optimization 이론에 관해서는 3. compound model scaling에 자세히 서술한다.



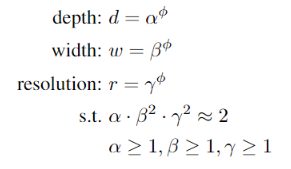
Baseline model을 scaling down하는 작업에 대한 선행 연구는 W, D, R의 세 가지 요인에 대해서 각각도 거의 이루어지지 않았고 참고 논문의 저자들도 관심이 없는 것으로 보인다. 따라서 이번 프로젝트에서 compound model scaling의 결과를 활용하여 최적의 scaling down 방식을 찾는 실험을 진행한다. Baseline model로는 동일하게 EfficientNet-B0을 사용하고 compound scaling down 방식으로 EfficientNet-A0, A1를 제작하여, MobileNet과의 비교를 통해 기존의 Deep ConvNet을 scaling down 하는 것이 어떤 의미를 가지는지에 대해 고찰할 것이다.

**2. Related work**



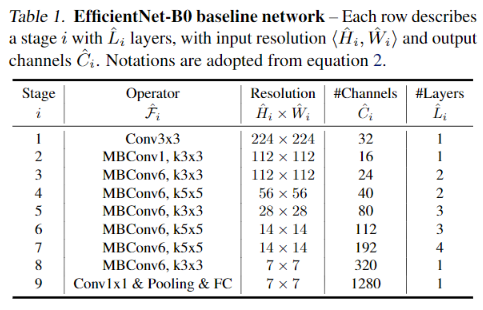
EfficientNet-B0을 요인 하나만 scaling up 했을 때 이미지넷 데이터의 Top-1 accuracy 그래프가 위와 같다. 따라서 일반적인 scaling down은 위 그래프의 역방향이라고 생각된다. 즉 W, D, R 중 하나만 바꾸면 성능이 크게 감소할 것으로 예측할 수 있다.

**3. Compound model scaling**

만약 FLOPS를 2배로 하려 할 때, Baseline W, D, R에 곱할 상수 w, d, r의 최적값을 결정한다고 하자. FLOPS는 에 비례하므로 이다. small grid search를 통해(참고 논문의 저자들은 구글의 TPU 전문가들이다.) 최적의 값을 , , 로 찾아냈다. 이를 이용하여 FLOPS를 만큼 늘리려면 , , 가 되는 것을 알 수 있다.

이번 프로젝트에서는 scaling down, 즉 인 경우에 대한 실험을 진행할 것이다.

**4. EfficientNet Architecture**



EfficientNet-B0의 구조는 다음과 같다. 메인 블록으로 mobile inverted bottleneck MBConv를 사용하고, 참고 논문의 저자들이 개발한 squeeze-and-excitation optimization이 적용되었다. 이번 프로젝트에서는 이를 이용하여 , 인 모델인 EfficientNet-A0, A1을 제작하였다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Width | Depth | Resolution | Dropout | #params |
| EfficientNet-B0 | 1 | 1 | 224x224 | 0.2 | 4.1M |
| EfficientNet-A0 | 0.9 | 0.8 | 195x195 | 0.2 | 3.5M |
| EfficientNet-A1 | 0.8 | 0.7 | 180x180 | 0.2 | 2.3M |

**5. Experiments**

이번 프로젝트에서 이미지넷 데이터셋을 학습하는 것은 하드웨어 리소스의 한계로 불가능하였다. 따라서 같은 Image classification이면서 크기가 작은 Oxford Flower 102 dataset을 채택했다.

데이터셋 설명: <https://www.robots.ox.ac.uk/~vgg/data/flowers/102/>을 통해 다운로드 받을 수 있다. 압축을 풀면 train, valid, test 폴더에 6552, 818, 819개의 꽃 사진이 존재한다. image preprocessing은 다음과 같다.

* train: 256x256으로 resize, 224x224로 center crop, random horizontal flip, target resolution으로 다시 resize한다.
* test, valid: 256x256으로 resize, 224x224로 center crop, target resolution으로 다시 resize 한다.

모델 설명: Not pre-trained. (Weight initialization은 github를 참고.)

학습 하이퍼파라미터:

* Learning rate는 0.01, 0.001로 실험하고 더 높은 test accuracy 값을 기록한다.
* Momentum=0.9
* Loss: Cross entropy를 사용.
* Optimizer: SGD를 사용.
* Batch size=4, **Training epoch=20**

학습 결과는 다음과 같다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 실험 | Model | #params | Top1 acc. | Top5 acc. | Resolution |
| 1 | EfficientNet-B0 | 4.1M | 79.73% | 94.75% | 224x224 |
| 2 | EfficientNet-A0 | 3.5M | 76.56% | 92.80% | 195x195 |
| 3 | MobileNetV2 (w=1.2) | 3.5M | 75.09% | 92.19% | 195x195 |
| 4 | EfficientNet-A1 | 2.3M | 74.60% | 92.67% | 180x180 |
| 5 | MobileNetV2 (w=1.0) | 2.3M | 73.87% | 90.84% | 180x180 |

**6. Discussion & Conclusion**

1) 왜 이렇게 accuracy가 낮은가?

- 이미지넷 데이터셋을 미리 학습한 pre-trained model에 비하여 훨씬 모자란 feature extraction 성능을 보여준다. 만약 하드웨어 리소스와 시간이 더 주어졌다면 보다 의미있는 결과가 나올 것으로 확신한다.

- Training epoch이 너무 적다. (실험당 약 30분 소요) 한 마디로 학습이 되다가 말았다.

2) Compound scaling down은 효과가 있는가?

- 동일한 Baseline model의 W, D, R 중 하나씩만 scaling down한 모델과의 정량적인 비교를 위해 다양한 실험을 해보아야 하므로, 일반적인 Scaling down method와의 비교는 할 수 없다. 한편 이번 프로젝트에서는 리소스(파라미터 수, FLOPS)를 줄이기 위하여 MobileNet과 같이 더 light한 모델을 채택하는 기존의 방식 보다는 효율이 좋다는 결론을 얻었다.

딥러닝 아키텍처의 발전은 앞으로 정확도 향상보다는 모바일 디바이스에도 적용할 만큼 하드웨어 리소스를 적게 이용하는 경량화에 주안점을 두고 있다고 생각한다. 따라서 이번 연구를 통해 기존에 잘 개발된 모델들을 어떻게 scaling down 할 지에 대해 고민해볼 수 있었고 최신 연구 주제인 compound model scaling과 이미지 분류에 아주 강력한 딥러닝 네트워크인 EfficientNet을 사용해 보는 것으로 이번 프로젝트의 의의를 둔다.

**7. Reference.**

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, Mingxing Tan, Quoc V. Le, 2019

https://arxiv.org/abs/1905.11946에서 확인할 수 있습니다. 본문의 모든 그림의 원본이 수록되어있습니다.

**Appendix. 코드 구현 과정**

Google Colaboratory(https://colab.research.google.com/)에서 GPU(Tesla K80)를 사용하여 모델 학습에 사용하였습니다.

https://github.com/narumiruna/efficientnet-pytorch 에는 EfficientNet B0~B7이 pytorch로 구현되어 있습니다. efficientnet.py를 변경하여 EfficientNet A0, A1을 만드는데 사용하였습니다.

MobileNetV2 코드의 출처는 https://github.com/d-li14/mobilenetv2.pytorch 입니다.

https://github.com/Muhammad-MujtabaSaeed/102-Flowers-Classification 의 ipynb 파일을 기본 스켈레톤으로 필요한 부분을 추가하여 사용하였습니다. 이 깃허브는 기본적으로 pre-trained resnet18로 Flower 102 dataset을 학습하고 테스트하는 모든 과정을 포함하고 있습니다.

102\_Flowers\_classification.ipynb 파일을 실험을 위해 전체적으로 수정했습니다.

* Google drive, EfficientNet과 연동하기 위해 몇몇 코드를 추가하였습니다.
* Image preprocessing 부분을 resolution을 변화시키기 위해 수정하였습니다.
* resnet18 대신 MobileNet, EfficientNet을 사용하기 위해 model 지정에 관련된 코드를 수정하였습니다.

직접 재현해보시려면 위해서는 첨부된 README 파일을 참고해주세요.