CS 182 Deep Neural Networks **Spring 2023**

EfficientNet: Understanding the Compound Scaling Method

Siqi Zhu, Ethan Qiu, Yuexi Shen, Charlie Cheng

1 Introduction

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks [1] introduces a novel model architecture scaling approach, optimizing for FLOPs and parameter efficiency. As a reminder, FLOPs stand for floating point operations per second, and parameters are a general measure for the complexity of a model. This architecture aims to capture the intuition that architecture dimensions (depth, width, resolution) should not be independent of each other, and indeed, should be balanced with each other. To this end, Efficient introduces compound model scaling, leveraging training-aware neural architecture search to develop a strong baseline model, which can then be scaled up to a variety of models.

2 Intuition of the Compound Scaling Method

In this problem, we will quickly walk through the Compound Scaling Method, and attempt to communicate the intuition behind the approach. For ConvNet layers, generally, there are three essential parameters that could be scaled: depth, width, and resolution:

Depth: Generally, deeper networks are able to capture more complex features. However, there is also a trade off considering the deeper networks take more resources (time, compute) to train. Notably, there appears to be diminishing returns with regards to continually increasing the depth of a network, such as ResNet-1000 having similar performance to ResNet-101.

Width: Wider networks are generally more capable of capturing complex features, they also tend to be easier to train. Unfortunately, wide and shallow networks tend to have difficulties in capturing high level features.

Resolution: Higher resolution images allow the capture of more complex features. But, again, there are resource constraints (time, compute), and diminishing returns.

Problem 1.1. Intuitively, should the different dimensions (depth, width, resolution) be independent? Let's say that you have some extra compute to improve a model, what dimension(s) will you tune?

Solution to 1.1: These dimensions should not be independent. Noting from above, given that there are diminishing returns for the dimensions, if we have some extra compute to spend, we would not throw it just increase network depth, or width, or resolution. We would increase all of them. □

Let us pose the construction of a ConvNet in the form of an optimization problem. Rather obviously, we are trying to maximize the performance of our model, thus accuracy. For sake of simplicity, we will be assuming that every ConvNet layer has the same architecture.

Problem 1.2. Let us formulate an ConvNet stage as a function f. Please fill in the blanks regarding the domain and codomain of the function. Let us have input tensor (to the layer) be X with shape H, W, C

[1]Mingxing Tan, Quoc V. Le *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. `https://arxiv.org/abs/1905.11946`

representing height, width, channels. L being the layer repetition on this stage (We ignore batching for simplicity).

$$\mathcal{F}^{\_\_} : \text{Input Tensor}_{\langle \_\_, \_\_, \_\_ \rangle} \rightarrow \text{Output Tensor}$$

Solution 1.2:

$$\mathcal{F}^{L} : \text{Input Tensor}_{\langle H,W,C \rangle} \rightarrow \text{Output Tensor}$$

□

Problem 1.3. Given our ConvNet stage, please derive an expression for an entire ConvNet $\mathcal{N}$ composed of n stages. (Remember H, W, C and L could vary from stage to stage!) We use $\bigodot$ to denote composition of stages.

$$\mathcal{N} = \_\_$$

Solution 1.3:

$$\mathcal{N} = \bigodot_{i=1}^{n} \mathcal{F}^{L_i}(X_{\langle H_i, W_i, C_i \rangle})$$

□

Problem 1.4. Suppose that we have defined parameters $\hat{H}_i, \hat{W}_i, e\hat{C}_i, \hat{L}_i$, and tune able parameters d, w, r representing depth, width, and resolution respectively. And we are given functions *Accuracy*, *Memory* and $FLOPS : Network \rightarrow \mathbb{Z}$ that takes in a network, and output respective values of the network, as well as hard constraints *targetmemory* and *targetFLOPS*, please fill in the blanks for the following optimization problem:

(Reusing your work is suggested).

$$\begin{aligned} max_{d,w,r} \; & \text{Accuracy}(\_\_) \\ s.t. \; & \mathcal{N}(d,w,r) = \_\_\_\_\_\_\_\_\_\_\_\_ \\ & \_\_\_\_\_\_\_\_ \leq targetmemory \\ & \_\_\_\_\_\_\_\_ \leq targetFLOPS \end{aligned}$$

Solution to 1.4:

$$\begin{aligned} max_{d,w,r} \; & \text{Accuracy}(\mathcal{N}(d,w,r)) \\ s.t. \; & \mathcal{N}(d,w,r) = \bigodot_{i=1}^{n} \mathcal{F}^{d \cdot \hat{L}_i}(X_{\langle r\hat{H}_i, r\hat{W}_i, w\hat{C}_i \rangle}) \\ & Memory(\mathcal{N}) \leq targetmemory \\ & FLOPS(\mathcal{N}) \leq targetFLOPS \end{aligned}$$

□

As you have hopefully observed, changing d, w, r is a complex problem. And hence, the main contribution of the paper:

3 The Compound Scaling Method

Tan, Quoc and Le introduces the Compound Scaling Method, which they have formulated as an optimization problem below.

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

Looking at the formulation, we note that α, β, γ are parameters obtained through a grid search, thus in application, the only variable to be varied is ϕ, the compound coefficient. Thus, the compound scaling method produces a way for one to determine the dimensions of a network by only varying one paramter, the compound coefficient.

Problem 2.1. Looking at the optimization problem, what will happen to the depth, width, and resolution as you increase or decrease ϕ? What advantages could this approach bring compared to arbitrarily scaling these dimensions?

Solution to 2.1: Increasing ϕ will increase depth, width and resolution. Given that we observe that these parameters should not be independent, this systematic approach allows us to skip some manual tuning, and approach the problem in a principaled manner. □

Problem 2.2. For an arbitrary ConvNet (thinking about our optimization problem formulation above, and the basic understanding of what a ConvNet is), fixing all other variables:

a. What will doubling the depth do to the total FLOPS?

b. What will doubling the width do to the total FLOPS?

c. What will doubling the resolution do to the total FLOPS?

You may think of FLOPS just as operations that need to be performed. Thinking in quasi-big-o concepts might be useful.

Solution to 2.2: Looking back to the strcture of a ConvNet:

a. Doubling depth will double FLOPS

b. Doubling width will result in a $2^2 = 4$ times increase in FLOPS.

c. Doubling resolution will result in a $2^2 = 4$ times increase in FLOPS

□

Problem 2.3. Considering the constraint $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$, how much will the total FLOPS be for some ϕ? Again, thinking in quasi-big-o concepts might be useful.

Solution to 2.3: Total FLOPS will increase by approximately 2^{ϕ}. □

4 Implmenting EfficientNet

The rest of this problem will be in jupyter notebook form. We also have the conceptual questions on latex. The coding questions are in the notebook!

Problem 3.1. Play around with this function, at what point do you start to see new depth layers emerging? (You might also want to use this opportunity to check your intuition about the previous conceptual questions).

Solution: Answer may vary, but generally you should see new depths emerging around 5. □

Problem 3.2. What's the shape of data in train loader for a single batch? (in terms of $[N, C, H, W]$)

Solution: 60000, 3, 32, 32.

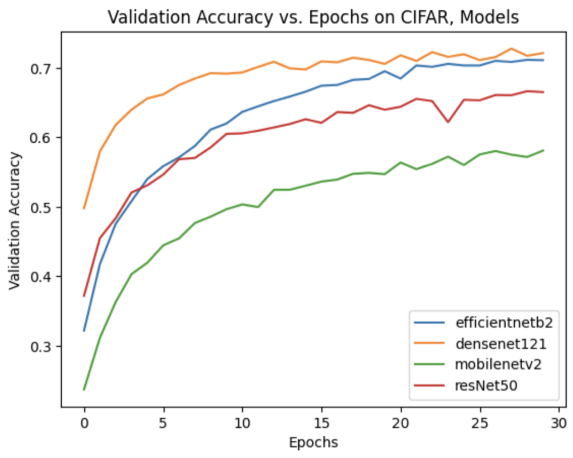
This is just a sanity check. □

Problem 3.3. What is the best accuracy you can get? What is the best accuracy you can get with the same number of parameters as the EfficientNet-B2 model? Feel free to use different models and find the one with the best validation accuracy.

Solution: Solution after 30 epochs for B2 is around 0.70. There might be slight variations owing to randomness. Generally, the higher numbered models (b7 vs. b0) will perform better. □

Problem 3.4. Looking at the accuracy of EfficientNet compared to other state-of-the-art model architectures, how does the validation accuracy compare? Anything else interesting you've noticed about these plots?

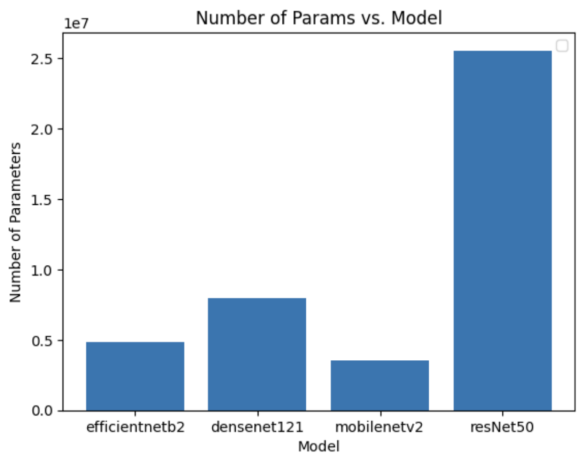
Solution: Answers may vary. This is the validation accuracy vs. epochs graph we might expect to see. In general, efficientnetb2 performs very well, only slightly less than densenet121.



□

Problem 3.4. Examine the number of parameters of the different models, does anything stand out to you? If we take this as context, is there anything you want to comment about the validation performance of the models?

Solution: Answers may vary. We notice that efficientnetb2, is well, efficient. It is the second lowest number of parameters amongst the cohort of models that we tested against. In context, perhaps the student might say that relative to the expense of the number of parameters, efficientnetb2 achieves very respectable results. We also do just observe that densenet121 is the next highest model in terms of parameters, whilst also being the model that beat efficientnetb2 in terms of accuracy. But we also note that the parameter number difference is non-trival.


Number of Params vs. Model
1e7
2.5
2.0
1.5
1.0
0.5
0.0
Number of Parameters
efficientnetb2
densenet121
mobilenetv2
resNet50
Model

□