Article presentation / Lab1 experience

Mehdi Berrada & Selim Zighed

19/02/2025

Overview

Introduction

- ***Title*** : EfficientNet : Rethinking Model Scaling for Convolutional Neural Networks
- ***Author*** : Mingxing Tan & Quoc V. Le

Context

- Convolutional Neural Networks (CNNs) are widely used in image recognition.
- Higher accuracy is often achieved by scaling up CNNs. Scaling means increasing depth (layers), width (channels), or resolution (input image size).
- ***Issue*** : Scaling inefficiently increases computational cost and memory usage without proportional accuracy gains.
- ***Problematic*** : How can we scale CNNs more efficiently to improve accuracy while reducing computational cost ?

State of The Art vs Suggested Solution

- Existing methods scale only one dimension at a time, leading to suboptimal efficiency.
- ***EfficientNet*** : Introduces Compound Model Scaling, a method that balances depth, width, and resolution, using a single scaling coefficient ϕ, and without changing layer operators in the baseline model.

Notations

- ***FLOPs*** : Floating Point Operations per Second to measure the computational cost of a deep learning model.
- $\boldsymbol{d}$: depth, $\boldsymbol{w}$: width, $\boldsymbol{r}$: resolution
- α, β, γ : Constants for scaling depth, width, and resolution.
- ***Compound Scaling Hypothesis*** : $d = \alpha^{\phi}, w = \beta^{\phi}, r = \gamma^{\phi}$

Problem formulation

Objective :

$$\max_{\phi} \quad \text{Accuracy}\big(N(\alpha^\phi, \beta^\phi, \gamma^\phi)\big)$$

Subject to :

$$\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

$$\text{Memory}(N(\alpha^\phi, \beta^\phi, \gamma^\phi)) \leq \text{target memory}$$

$$\text{FLOPs}(N(\alpha^\phi, \beta^\phi, \gamma^\phi)) \leq \text{target FLOPs}$$

Intuition

- Increasing **depth** (more layers) improves accuracy.
- Increasing **width** (more channels) enhances feature learning.
- Increasing **resolution** (input size) captures details.

EfficientNet Architecture

- ***Baseline Model EfficientNet-B0*** : Built using *SE blocks*, *SiLU activation function* ($SiLU(x) = x\sigma(x)$) and *Depthwise Separable Convolutions* (75% less computational costs) ; all suggested by **NAS**, Neural Architecture Search.
- Also uses normal convolution layers as well as *MBConv* layers with more attention and residuals, which lowers the convolution costs.

EfficientNet Architecture

Stage	Operator	Resolution	Channels
1	Conv3x3	224×224	32
2	MBConv1, k3x3	112×112	16
3	MBConv6, k3x3	112×112	24
4	MBConv6, k5x5	56×56	40
5	MBConv6, k3x3	28×28	80
6	MBConv6, k5x5	14×14	112
7	MBConv6, k5x5	14×14	192
8	MBConv6, k3x3	7×7	320
9	Conv1x1, Pooling, FC	7×7	1280

Table – EfficientNet-B0 Network Architecture

Scaling EfficientNet-B0 to EfficientNet-B7

Model	**Input Res**	**Depth (α^ϕ)**	**Width (β^ϕ)**	**Params (M)**
EffNet-B0	224^2	1.0	1.0	5.3
EffNet-B3	300^2	1.4	1.3	12
EffNet-B7	600^2	3.1	2.0	66

Table – Compound Scaling examples on EfficientNet-B0

Experimental Results

1. ***ImageNet Benchmark Results*** : 84.3% for EffNet-B7 (top1 amongst the state of the art with 8.4x fewer params and 6.1x faster inference that the closest model).
2. ***Transfer Learning Performance*** :
   - CIFAR-100 : 91.7%
   - Oxford Flowers : 98.8%
   - Stanford Cars : 93.6%

Conclusion

- Lower Computational Cost → Reduces AI training expenses.
- Scalable for Future Research → Inspired EfficientDet (object detection).

Model Used : ResNet18

Stage	Type	Layers	Output Size
Input	Image	-	$3 \times 224 \times 224$
Conv1	Conv2D 3×3	1	$64 \times 112 \times 112$
MaxPool	3×3	1	$64 \times 56 \times 56$
Stage 1	BasicBlock	2	$64 \times 56 \times 56$
Stage 2	BasicBlock	2	$128 \times 28 \times 28$
Stage 3	BasicBlock	2	$256 \times 14 \times 14$
Stage 4	BasicBlock	2	$512 \times 7 \times 7$
Global Pooling	AdaptiveAvgPool2D	1	$512 \times 1 \times 1$
FC Layer	Linear	1	10 (CIFAR-10)

Table – ResNet-18 Layers Breakdown

Feature Map Sizes in ResNet-18

Layer	Feature Map Size	Filters (Width)
Input	$3 \times 224 \times 224$	-
Conv1	$64 \times 112 \times 112$	64
MaxPool	$64 \times 56 \times 56$	64
Stage 1	$64 \times 56 \times 56$	64
Stage 2	$128 \times 28 \times 28$	128
Stage 3	$256 \times 14 \times 14$	256
Stage 4	$512 \times 7 \times 7$	512
Global Pooling	$512 \times 1 \times 1$	512
FC Layer	10 (CIFAR-10)	-

Table – Feature Map Sizes in ResNet-18

Total Parameters & Memory Usage

- Total Parameters : 11.18M
- Memory Usage : 106 MB
- Lightweight model $\rightarrow$ Great for CIFAR-10
- Good accuracy/FLOPs trade-off compared to deeper ResNets (84.87% accuracy)

Fixed Training Params

- ***Epochs*** : 25
- ***Loss Function*** : Cross Entropy

Learning Rate Influence

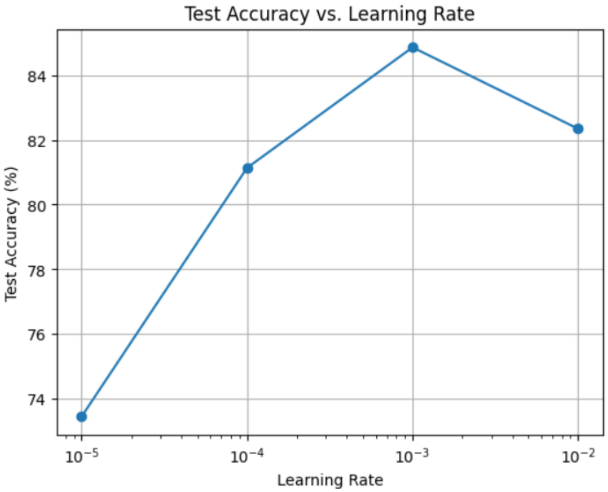

Figure – Test accuracy values for different learning rates

Momentum Influence when Using SGD

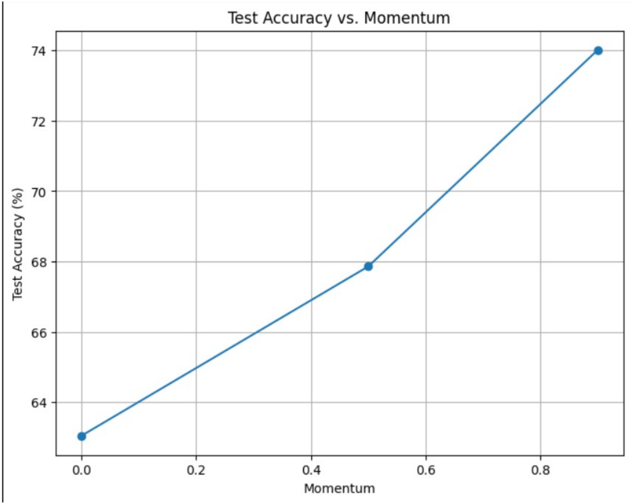

Figure – Test accuracy values for different momentum values

Optimiser Influence

Optimiser	**Accuracy**
Adam	84.87%
SGD ($momentum = 0.9$)	74.01%

Table – Accuracies per optimiser

Trade Off Accuracy/Size

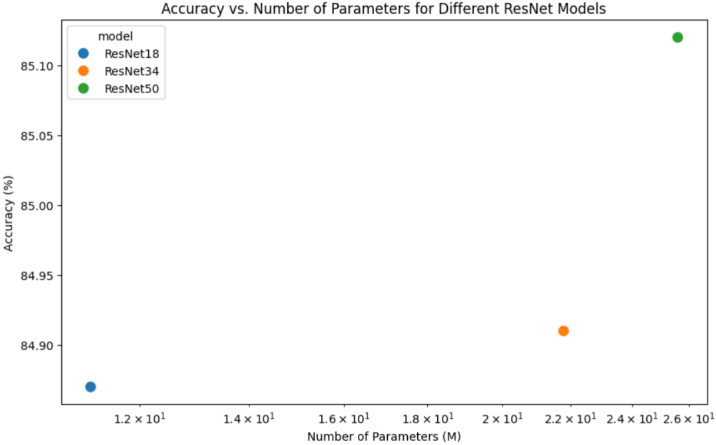

Figure – Accuracy and Number of Parameters for ResNets

Other Idea : Compound Scaling Implementation

Scaling Factors Used :

- $\alpha = 1.2$, $\beta = 1.1$, $\gamma = 1.15$
- Different values of ϕ determine scaled versions of ResNet-18.

Model	**Input Res**	**Depth** (α^ϕ)	**Width** (β^ϕ)	**FLOPs (M)**
ResNet-18	32^2	1.0	1.0	1.8B
($\phi = 1$)	36^2	1.2	1.1	2.4B
($\phi = 2$)	48^2	1.4	1.2	3.1B

Table – Scaling ResNet-18 on CIFAR-10