VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANA SANGAMA, BELAGAVI - 590018

Technical Seminar Report

On

"EfficientNet: Rethinking Model Scaling for CNNs"

Submitted in partial fulfillment of the requirements for the 8th semester of

Bachelor of Engineering in Computer Science and Engineering

of Visvesvaraya Technological University, Belagavi

Submitted by:

Yash Vora **1RN16CS123**

Under the guidance of:

Mr. Devraju B.M.
Assistant Professor
Dept. of CSE

Department of Computer Science and Engineering

(NBA Accredited for academic years 2018-19, 2019-20, 2020-21)

RNS Institute of Technology

Channasandra, Dr. Vishnuvardhan Road, Bengaluru-560 098

2019-2020

RNS Institute of Technology

Channasandra, Dr. Vishnuvaradana Road,

Bengaluru-560 098

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(NBA Accredited for academic years 2018-19, 2019-20, 2020-21)

CERTIFICATE

Certified that the Technical Seminar work entitled ***"EfficientNet: Rethinking Model Scaling for CNNs"*** has been successfully carried out by **Yash Vora** bearing USN **1RN16CS123**; bonafide student(s) of **RNS Institute of Technology** in partial fulfillment of the requirements for the **8th semester, Bachelor of Engineering** in **Computer Science and Engineering** of **Visvesvaraya Technological University**, Belagavi, during the academic year 2018-2019. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated. The project report has been approved as it satisfies the project requirements of 8th semester BE in CSE.

Mr. Devraju B.M.	**Dr. G. T. Raju**	**Dr. M. K. Venkatesha**
Assistant Professor	**Vice Principal**	**Principal**
Dept. of CSE	**Professor and HoD**	

External Viva:

Name of the Examiners **Signature with date**

1. Name

2. Name

ACKNOWLEDGEMENTS

The joy and satisfaction that accompany the successful completion of any task would be incomplete without thanking those who made it possible. I consider myself proud to be a part of RNS Institute of Technology, the institution which moulded me in all my endeavours. I express my gratitude to beloved Chairman **Dr. R N Shetty**, for providing state of art facilities. I express my deep gratitude to **Dr. H N Shivashankar**, Director, who has always been a great source of inspiration.

I would like to express my sincere thanks to **Dr. M K Venkatesha**, Principal and **Dr. G.T. Raju**, Vice Principal, Professor and HOD, Department of CSE, for their valuable guidance and encouragement throughout. I extend my sincere thanks and heartfelt gratitude to my Technical seminar reviewer **Mr. Devraju B.M.**, Asst. Prof., Department of CSE.

Finally, I take this opportunity to extend my earnest gratitude and respect to my parents, teaching and non-teaching staff of the department, the library staff and all my friends who have directly or indirectly supported me.

Date:

Place: Yash Vora 1RN16CS123

ABSTRACT

Convolutional neural networks (CNNs) are commonly developed at a fixed resource cost, and then scaled up in order to achieve better accuracy when more resources are made available. For example, ResNet can be scaled up from ResNet-18 to ResNet-200 by increasing the number of layers, and recently, GPipe achieved 84.3

The conventional practice for model scaling is to arbitrarily increase the CNN depth or width, or to use larger input image resolution for training and evaluation. While these methods do improve accuracy, they usually require tedious manual tuning, and still often yield suboptimal performance.

What if, instead, we could find a more principled method to scale up a CNN to obtain better accuracy and efficiency?

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Convolutional Neural Networks are widely used to achieve better accuracy in various deep learning and computer vision tasks. Since AlexNet won the 2012 ImageNet competition, CNNs (short for Convolutional Neural Networks) have become the de facto algorithms for a wide variety of tasks in deep learning, especially for computer vision. From 2012 to date, researchers have been experimenting and trying to come up with better and better architectures to improve models accuracy on different tasks. They are primarily developed on a fixed budget and scaled up to meet requirement later. The seminar details creation of efficient models as the competition is to beat the top accuracy; Scaling, if done correctly, can also help in improving the efficiency of a model.

1.2 Definitions

Here the topics are described and defined which are later required in the technical seminar.

1.2.1 Convolutional Neural Networks (CNN)

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. They have applications in image and video recognition, recommender systems, image classification, medical image analysis, natural language processing, and financial time series.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The ”fully-connectedness” of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

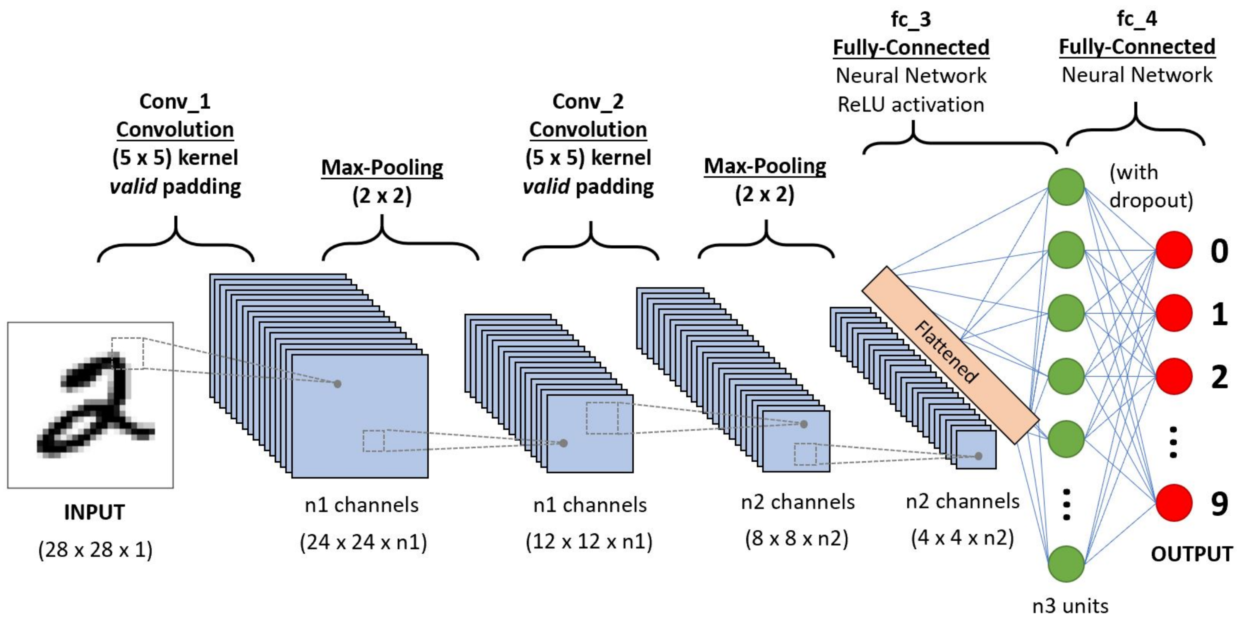


Figure 1.1: ConvNet.jpg

1.2.2 Hyperparameters

In machine learning, a hyperparameter is a parameter whose value is set before the learning process begins. By contrast, the values of other parameters are derived via training.

Hyperparameters can be classified as model hyperparameters, that cannot be inferred while fitting the machine to the training set because they refer to the model selection task, or algorithm hyperparameters, that in principle have no influence on the performance of the model but affect the speed and quality of the learning process. An example of a model hyperparameter is the topology and size of a neural network. Examples of algorithm hyperparameters are learning rate and mini-batch size.

1.2.3 Grid Search

The traditional way of performing hyperparameter optimization has been grid search, or a parameter sweep, which is simply an exhaustive searching through a manually specified subset of the hyperparameter space of a learning algorithm. A grid search algorithm must be guided by some performance metric, typically measured by cross-validation on the training set or evaluation on a held-out validation set.

Since the parameter space of a machine learner may include real-valued or unbounded value spaces for certain parameters, manually set bounds and discretization may be necessary before applying grid search.

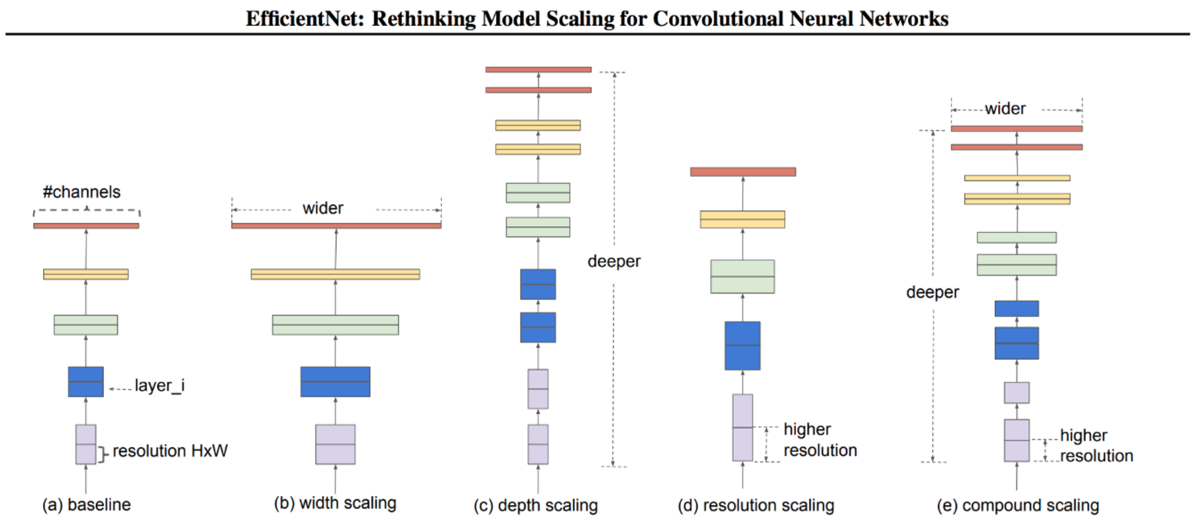


Figure 1.2: Scaling parameters of CNN.png

1.2.4 Scaling

There are three scaling dimensions of a CNN: depth, width, and resolution. Depth simply means how deep the networks is which is equivalent to the number of layers in it. Width simply means how wide the network is. One measure of width, for example, is the number of channels in a Conv layer whereas Resolution is simply the image resolution that is being passed to a CNN. Intuitively, the compound scaling method makes sense because if the input image is bigger, then the network needs more layers to increase the receptive field and more channels to capture more fine-grained patterns on the bigger image.

Chapter 2

Literature Survey

2.1 Deep Residual Learning for Image Recognition (2015)

Deeper neural networks are more difficult to train. The authors present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. They explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. Then the authors provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. Hence the authors provide significant evidence showing increasing depth of the network results in higher accuracy.

2.2 Wide Residual Networks (2017)

Deeper residual networks were shown to be able to scale up to thousands of layers and still have improving performance. However, each fraction of a percent of improved accuracy costs nearly doubling the number of layers, and so training very deep residual networks has a problem of diminishing feature reuse, which makes these networks very slow to train. To tackle these problems, in this paper the authors conduct a detailed experimental study on the architecture of ResNet blocks, based on which they proposed a novel architecture where they decreased the depth and increased width of residual networks. The authors call the resulting network structures wide residual networks (WRNs) and show that these are far superior over their commonly used thin and very deep counterparts. The authors successfully demonstrate that making a network wider increases accuracy.

2.3 GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism (2016)

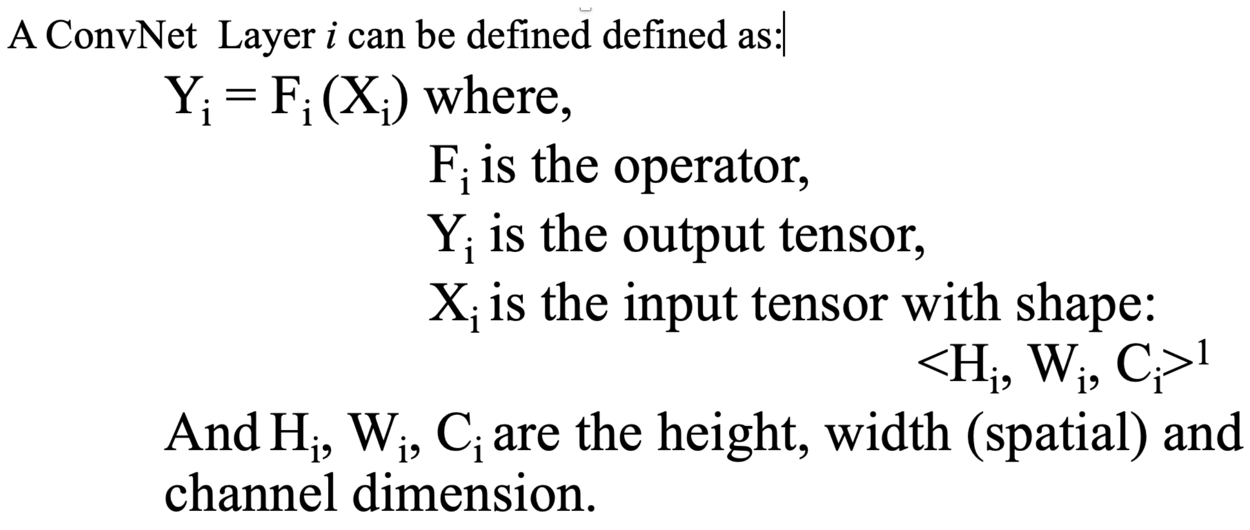
Scaling up deep neural network capacity has been known as an effective approach to improving model quality for several different machine learning tasks. In many cases, increasing model capacity beyond the memory limit of a single accelerator has required developing special algorithms or infrastructure. These solutions are often architecture-specific and do not transfer to other tasks. To address the need for efficient and task-independent model parallelism, the authors introduced GPipe, a pipeline parallelism library that allows scaling any network that can be expressed as a sequence of layers. By pipelining different sub-sequences of layers on separate accelerators, GPipe provides the flexibility of scaling a variety of different networks to gigantic sizes efficiently. This paper demonstrated that the number of channels of a network also affect the accuracy of the model.

Chapter 3

Problem Statement

3.1 Statement

The problem statement can be reduced to the following. The ConvNet can also be demonstrated



A ConvNet Layer *i* can be defined defined as:

$Y_i = F_i(X_i)$ where,

F_i is the operator,

Y_i is the output tensor,

X_i is the input tensor with shape:

$<H_i, W_i, C_i>^1$

And H_i, W_i, C_i are the height, width (spatial) and channel dimension.

Figure 3.1: Problem Definition.png

as a series of layers, with the following form. In practice, ConvNet layers are often partitioned into multiple stages and all layers in each stage share the same architecture: for example, ResNet has five stages, and all layers in each stage has the same convolutional type except the first layer performs down-sampling. Therefore, we can define a ConvNet as:

Unlike regular model scaling, this method tries expand network length(Li), width(Ci), and/or resolution(Hi, Wi) without changing Fi. By fixing Fi, task of model scaling is simplified but there still remains a large design space to explore Li, Ci, Wi, Hi, for each layer. In order to further

$$\mathcal{N} = \mathcal{F}_k \odot \ldots \odot \mathcal{F}_2 \odot \mathcal{F}_1(X_1) = \bigodot_{j=1\ldots k} \mathcal{F}_j(X_1)$$

$$\mathcal{N} = \bigodot_{i=1\ldots s} \mathcal{F}_i^{L_i}\left(X_{\langle H_i, W_i, C_i\rangle}\right)$$

where $\mathcal{F}_i^{L_i}$ denotes layer *F_i* is repeated *L_i* times in stage *i*.

Figure 3.2: Layer-wise representation of CNN.png

reduce design space, the authors restrict that all layers must be scaled uniformly with constant ratio. Hence the target is to maximize model accuracy for any given resource constraint, which can be formulated as the following optimization problem.

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

Figure 3.3: Optimization Problem.png

Chapter 4

Architecture

4.1 Compound Scaling

The authors proposed a simple yet very effective scaling technique which uses a compound coefficient Phi to uniformly scale network width, depth, and resolution in a principled way: Phi is a

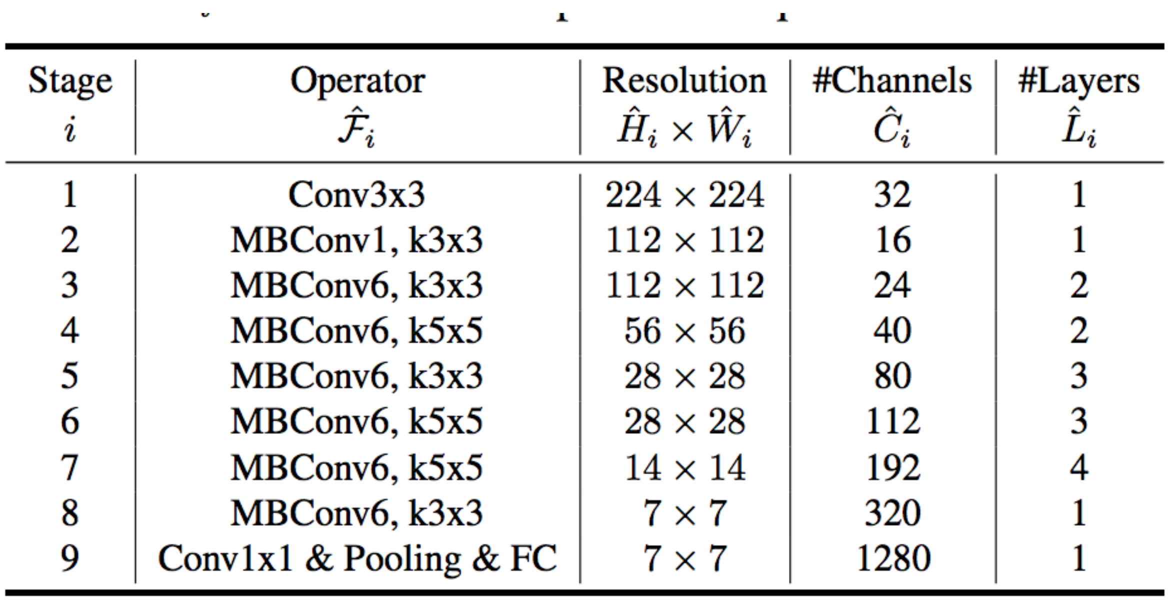
$$\begin{aligned} \text{depth: } & d = \alpha^{\phi} \\ \text{width: } & w = \beta^{\phi} \\ \text{resolution: } & r = \gamma^{\phi} \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned}$$

Figure 4.1: Compound Scaling.png

user-specified coefficient that controls how many more resources are available for model scaling. In a CNN, Conv layers are the most compute expensive part of the network. Also, FLOPS of a regular convolution op is almost proportional to d, w squared, r squared, i.e. doubling the depth will double the FLOPS while doubling width or resolution increases FLOPS almost by four times. Hence, in order to make sure that the total FLOPS don't exceed Phi squared. The constraint applied is that the product of alpha, beta and gamma squared should be nearly equal to 2.

4.2 EfficientNet Architecture

Scaling doesn't change the layer operations, hence it is better to first have a good baseline network and then scale it along different dimensions using the proposed compound scaling. The authors obtained their base network by doing a Neural Architecture Search (NAS) that optimizes for both accuracy and FLOPS. The architecture is similar to M-NASNet as it has been found using the similar search space. The network layers/blocks are as shown below: Now we have the base



Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	28×28	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Figure 4.2: EfficientNet baseline network.png

network, we can search for optimal values for our scaling parameters. If you revisit the equation, you will quickly realize that we have a total of four parameters to search for: alpha, beta, gamma and Phi. In order to make the search space smaller and making the search operation less costly, the search for these parameters can be completed in two steps.

- Fix Phi as 1, assuming that twice more resources are available, and do a small grid search for alpha, beta and gamma. For baseline network B0, it turned out the optimal values are alpha as 1.2, beta as 1.1, and gamma as 1.15 such that product of alpha, beta and gamma squared is almost equal to 2.

- Now fix alpha, beta and gamma as constants (with values found in above step) and experiment with different values of Phi. The different values of Phi produce EfficientNets B1-B7.

Chapter 5

Experiments and Results

5.1 Scaling

Scaling Up a Baseline Model with Different Network Width (w), Depth (d), and Resolution (r) Coefficients. Bigger networks with larger width, depth, or resolution tend to achieve higher accuracy, but the accuracy gain quickly saturate after reaching 80

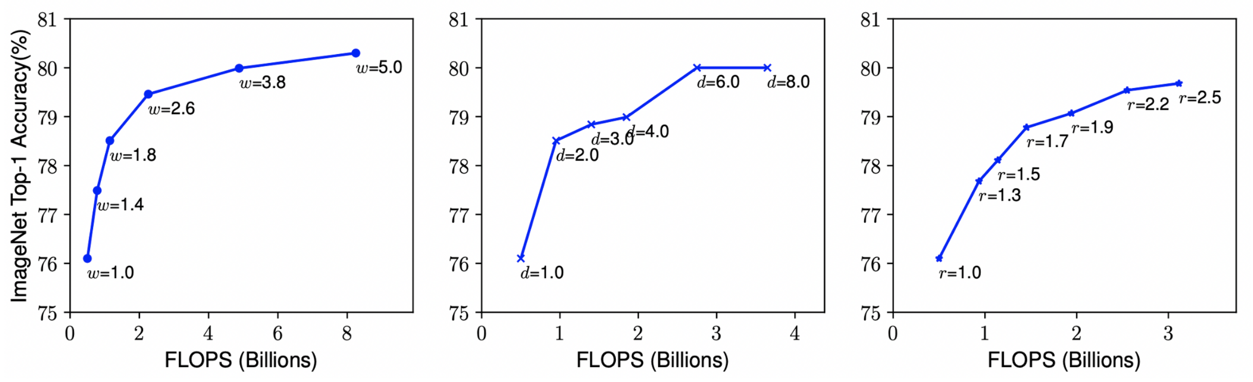


Figure 5.1: Depth Scaling.png

Scaling Network Width for Different Baseline Networks. Each dot in a line denotes a model with different width coefficient (w). All baseline networks are from the previous table. The first baseline network (d=1.0, r=1.0) has 18 convolutional layers with resolution 224x224, while the last baseline (d=2.0, r=1.3) has 36 layers with resolution 299x299.

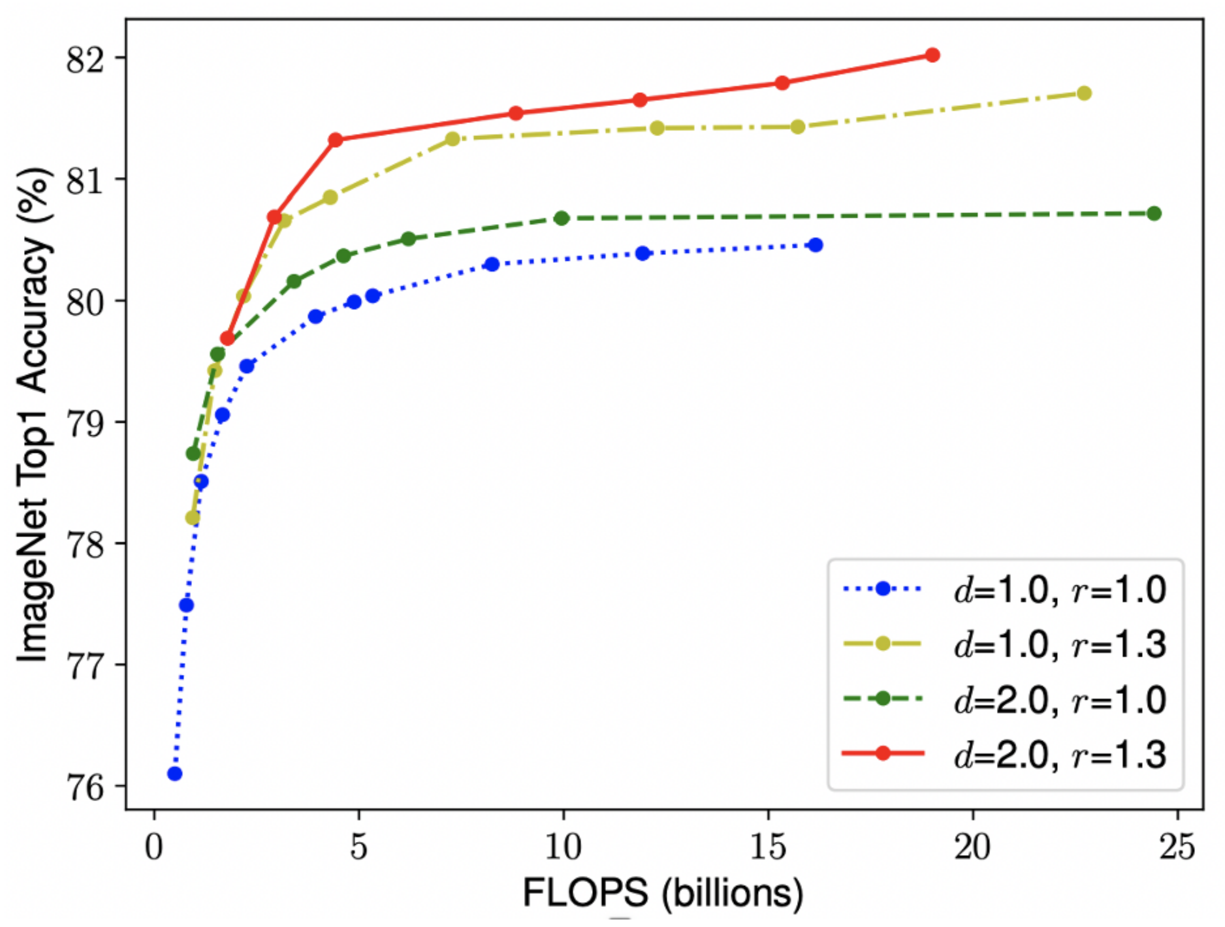
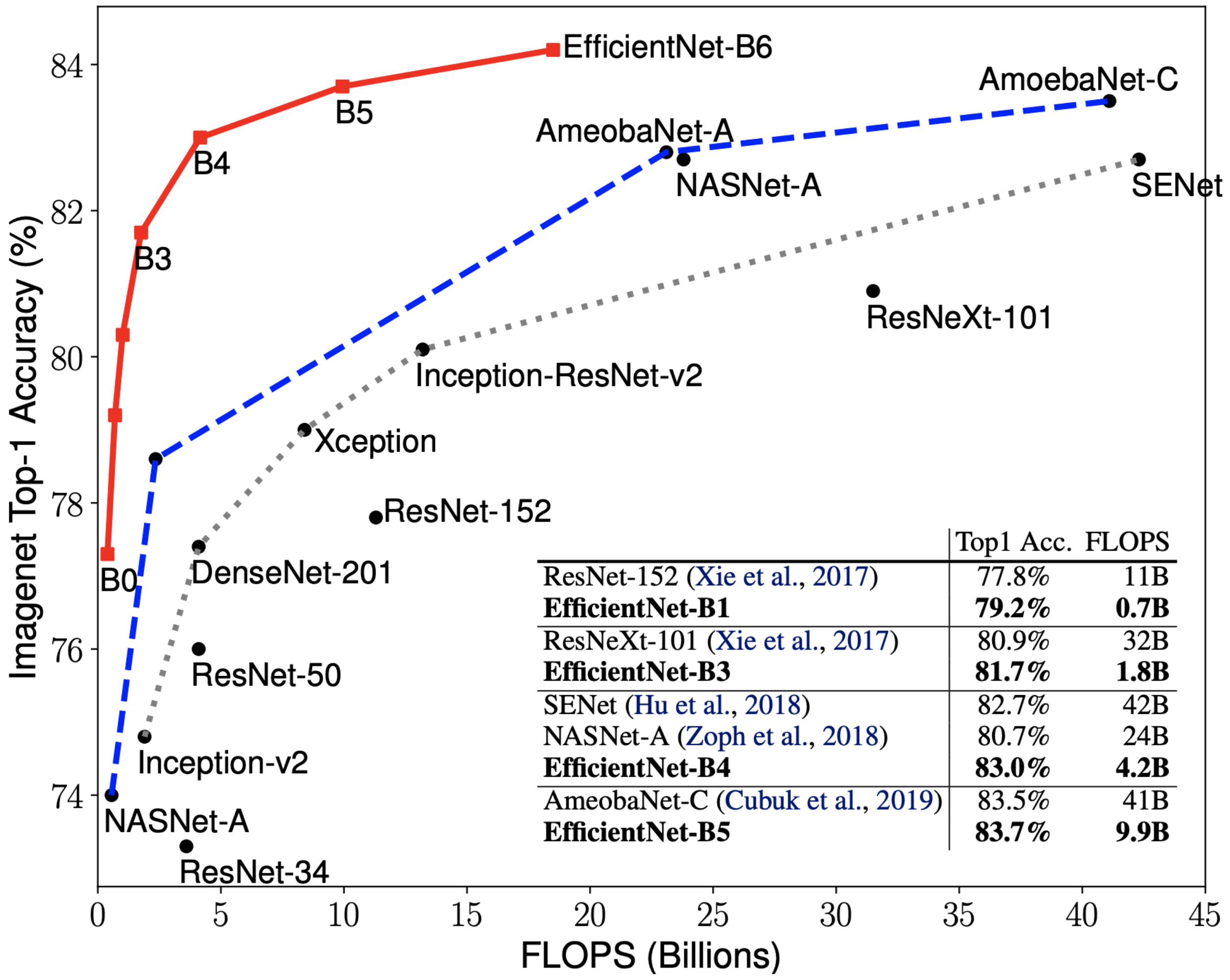


Figure 5.2: Scaling Depth and Resolution.png

5.2 Comparison of EffecientNet

As seen in the figure, the scaled models use upto 8.4x less parameters and a total reduction of upto 16x FLOPS is observed. The models were compared head-to-head against state of the art networks in their respective tasks.



	Top1 Acc.	FLOPS
ResNet-152 (Xie et al., 2017)	77.8%	11B
EfficientNet-B1	**79.2%**	**0.7B**
ResNeXt-101 (Xie et al., 2017)	80.9%	32B
EfficientNet-B3	**81.7%**	**1.8B**
SENet (Hu et al., 2018)	82.7%	42B
NASNet-A (Zoph et al., 2018)	80.7%	24B
EfficientNet-B4	**83.0%**	**4.2B**
AmeobaNet-C (Cubuk et al., 2019)	83.5%	41B
EfficientNet-B5	**83.7%**	**9.9B**

Figure 5.3: Results.png

It is also very evident that the compound scaling method also improves CNN dependability as we can see that the CNN is now focusing on the required objects for the task in a more accurate manner. This is done by freezing the training at an intermediate step and then observing one of the layers of the CNN. This demonstrates the ability of this algorithm to also apply to transfer learning tasks with good performance. Hence we can use these modified networks in mobile devices with less computing power but with the same efficiency of the full size model.

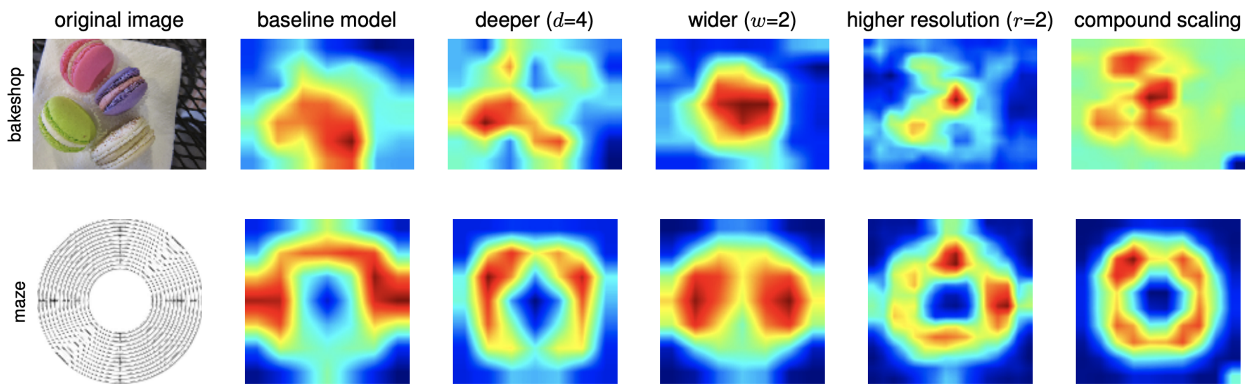


Figure 5.4: What the network can see.png

5.3 Limitations

It is possible to achieve even better performance by searching for alpha, beta and gamma directly around a large model, but the search cost becomes prohibitively more expensive on larger model. Hence we would require much higher computation power for doing such a task.

Bibliography

[1] **EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks**: `http://https://arxiv.org/abs/1905.11946`

[2] **Deep Residual Learning for Image Recognition**: `http://https://arxiv.org/abs/1512.03385`

[3] **Wide Residual Networks**: `http://https://arxiv.org/abs/1605.07146`

[4] **GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism**: `http://https://arxiv.org/abs/1811.06965`

[5] **On the Expressive Power of Deep Neural Networks**: `http://https://arxiv.org/abs/1606.05336`

[6] **The Expressive Power of Neural Networks: A View from the Width**: `http://https://arxiv.org/abs/1709.02540`