EfficientNet Paper Notes

Links:

Tags:

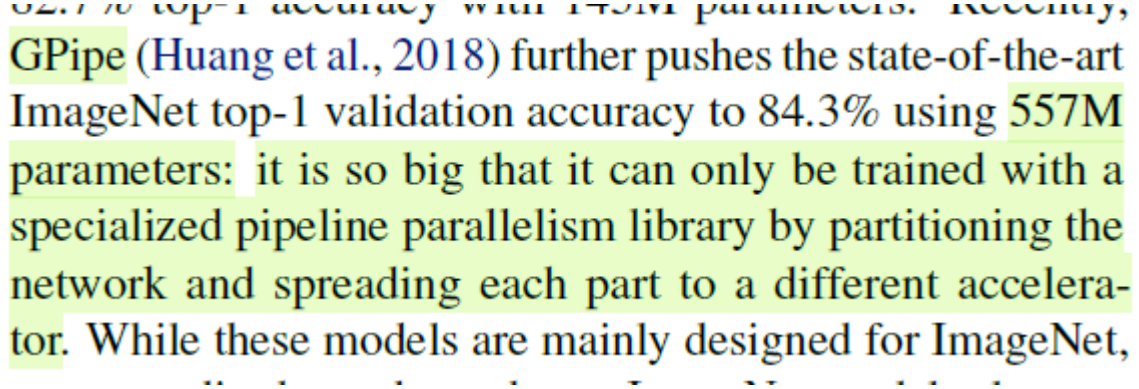
#paper #ML

Context:

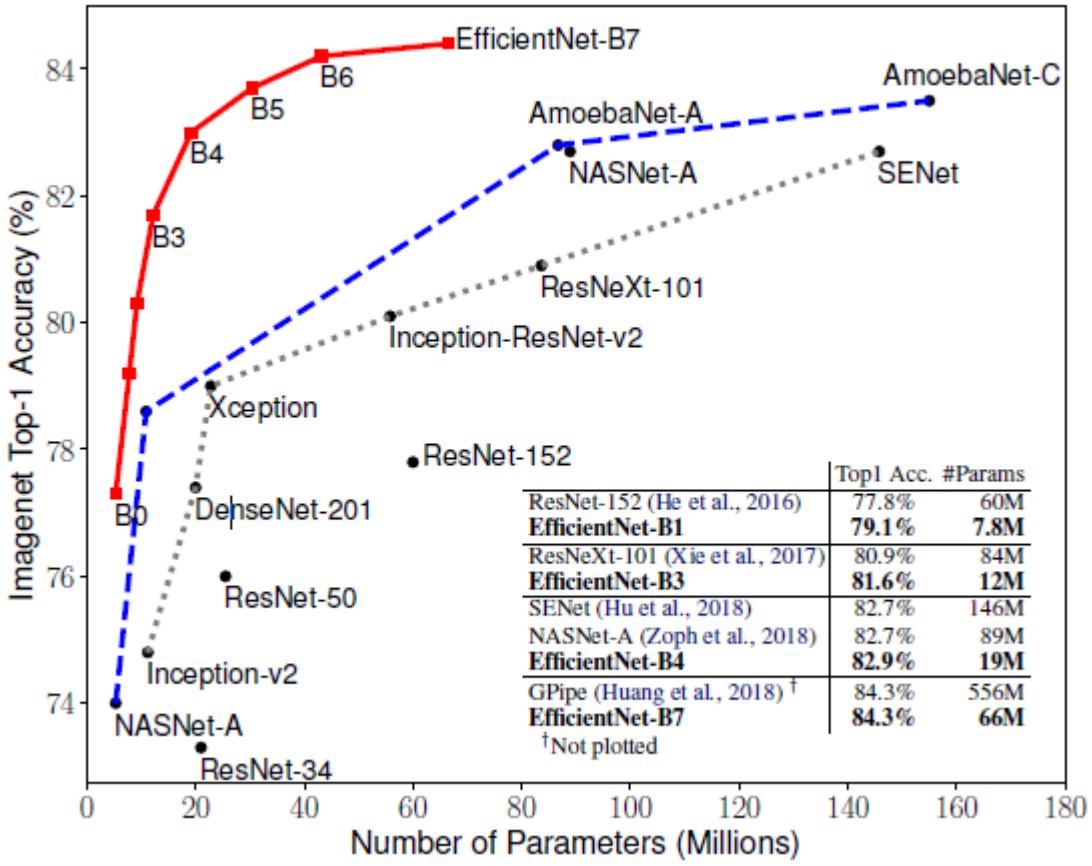
Paper Notes

Findings:

Motivation:


GPipe (Huang et al., 2018) further pushes the state-of-the-art ImageNet top-1 validation accuracy to 84.3% using 557M parameters: it is so big that it can only be trained with a specialized pipeline parallelism library by partitioning the network and spreading each part to a different accelerator. While these models are mainly designed for ImageNet,

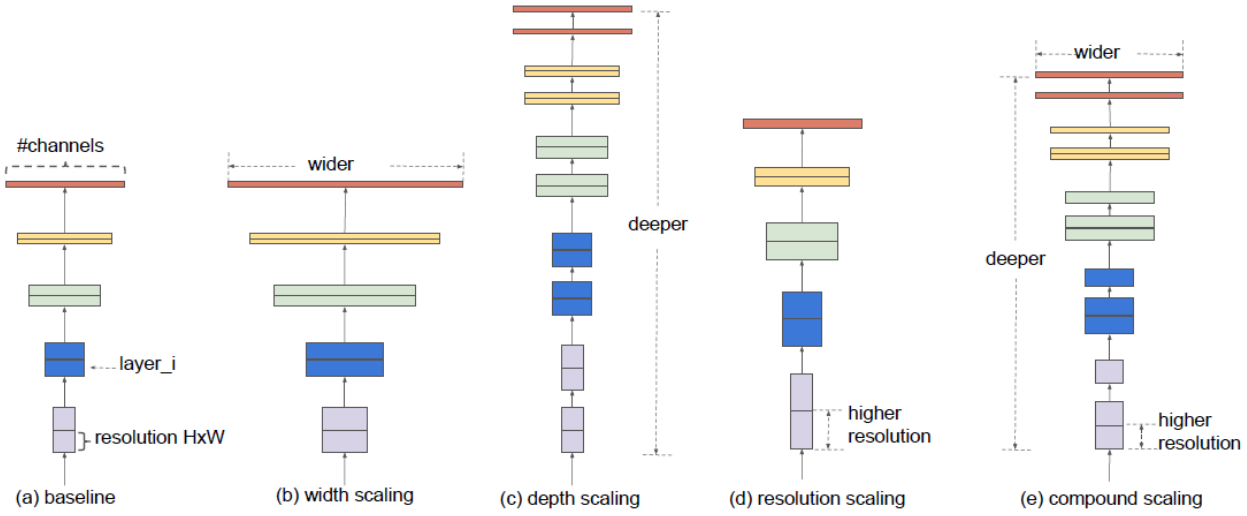
- In the race to building more accurate models the models got so big that they had to use specialized parallelism libraries to run the parts of the model on different GPUs/accelerators
- In this paper we are going to see how a conv nets are to be scaled such that they are **efficient and accuracy gainers**
- This paper proposes a new technique which balances the depth, width, resolution for better performance using a **new scaling method(compound scaling method)**
- These networks are **8.4x smaller and 6.1x faster** on inference than the best existing CNN then



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted

- Scaling the
  - **Depth**: Increases the **expressive power** of the model, allowing it to learn more **complex hierarchical features**.
  - **Width**: Increases the **number of neurons per layer**, allowing the model to capture **more diverse patterns**.
  - **Channels**: Expands the **number of feature maps**, improving the ability to extract **richer and more detailed representations**.



- The idea is that we create a *Baseline model* then scale it up according to the resolution of the input image or available computational resources

Compound Model Scaling

- **Mathematical Representaion of CNNs**:

  A ConvNet Layer i can be defined as a function: $Y_i = \mathcal{F}_i(X_i)$, where $\mathcal{F}_i$ is the operator, Y_i is output tensor, X_i is input tensor, with tensor shape $\langle H_i, W_i, C_i \rangle$[1], where H_i and W_i are spatial dimension and C_i is the channel dimension. A ConvNet $\mathcal{N}$ can be represented by a list of composed layers: $\mathcal{N} = \mathcal{F}_k \odot ... \odot \mathcal{F}_2 \odot \mathcal{F}_1(X_1) = \bigodot_{j=1...k} \mathcal{F}_j(X_1)$. In practice, ConvNet layers are often partitioned into multiple stages and all layers in each stage share the same architecture: for example, ResNet (He et al., 2016) 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:

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

  - where $\mathcal{F}_i^{L_i}$ denotes layer F_i is repeated L_i times in stage i, $\langle H_i, W_i, C_i \rangle$ denotes the shape of input tensor X of layer

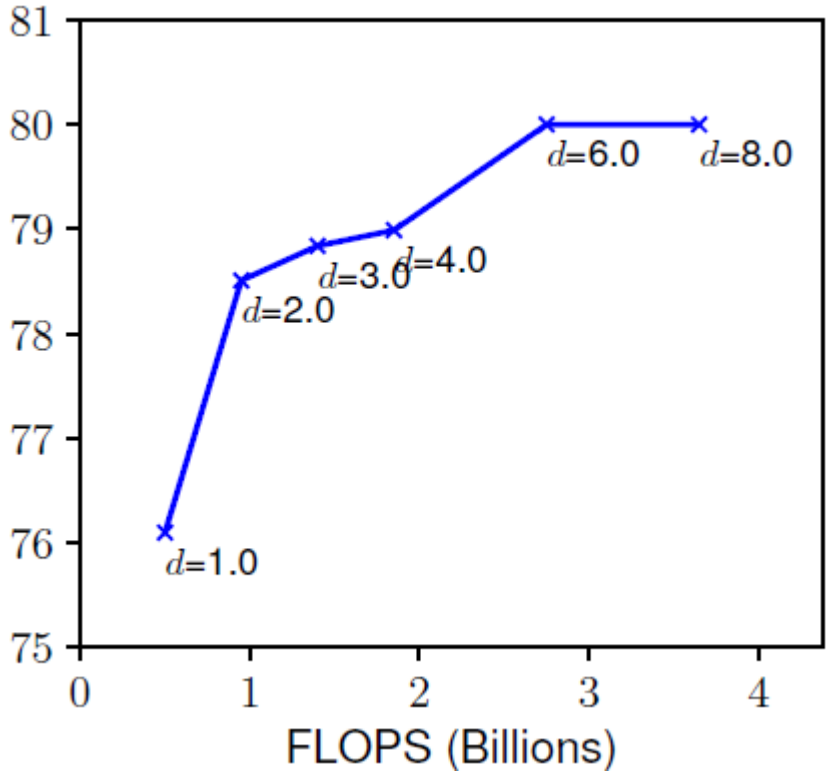
  - Unlike regular ConvNet designs that mostly focus on finding the best layer architecture $\mathcal{F}_i$, model scaling tries to expand the network length (L_i), width (C_i), and/or resolution (H_i, W_i) without changing $\mathcal{F}_i$ predefined in the baseline network. By fixing $\mathcal{F}_i$, model scaling simplifies the design

stant ratio. Our target is to maximize the model accuracy for any given resource constraints, which can be formulated as an optimization problem:

$$\begin{aligned} \max_{d,w,r} \quad & Accuracy(\mathcal{N}(d,w,r)) \\ s.t. \quad & \mathcal{N}(d,w,r) = \bigodot_{i=1\ldots s} \hat{\mathcal{F}}_i^{d\cdot\hat{L}_i}\left(X_{\langle r\cdot\hat{H}_i, r\cdot\hat{W}_i, w\cdot\hat{C}_i\rangle}\right) \\ & \text{Memory}(\mathcal{N}) \leq \text{target\_memory} \\ & \text{FLOPS}(\mathcal{N}) \leq \text{target\_flops} \end{aligned} \tag{2}$$

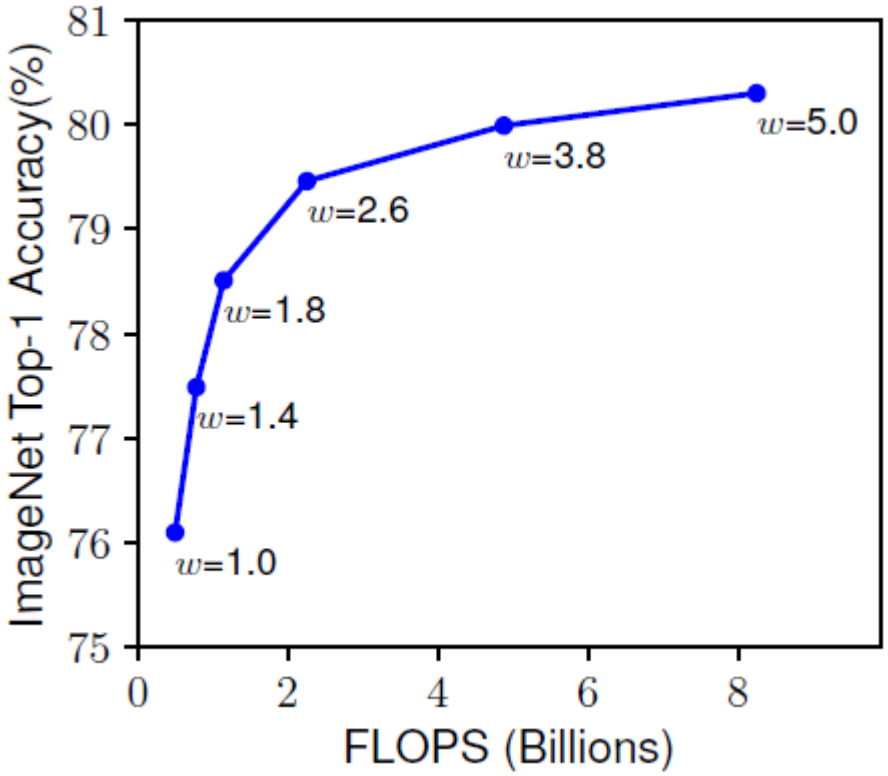
where w, d, r are coefficients for scaling network width, depth, and resolution; $\hat{\mathcal{F}}_i, \hat{L}_i, \hat{H}_i, \hat{W}_i, \hat{C}_i$ are predefined parameters in baseline network (see Table 1 as an example).

- Optimization problem, hence we use back prop to optimize the accuracy of the network for the given resources
- **Scaling Dimensions**:
  - **Depth**:
    - Deeper nets can capture more complex and rich features(Contextual details)
    - But more difficult to train due to vanishing gradient problem
    - To tackle this skip connections, batch norm can be used

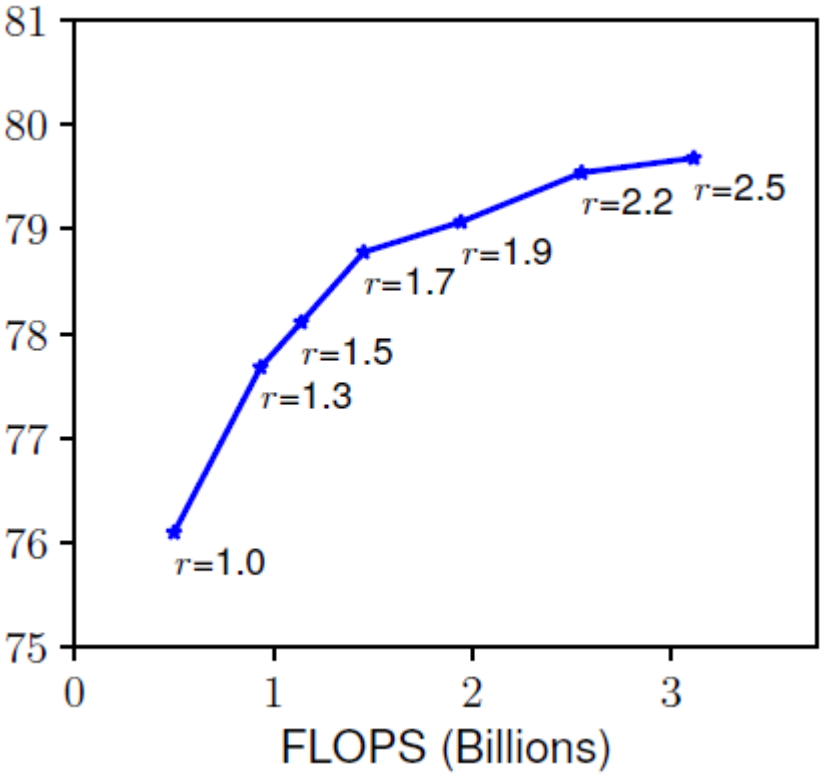


  - **Width**:

- commonly scaled for low depth models
- Capture more finer details(Low level details)
- easy to train
- but wide and shallow net cannot get the contextual info in the image like deep nets



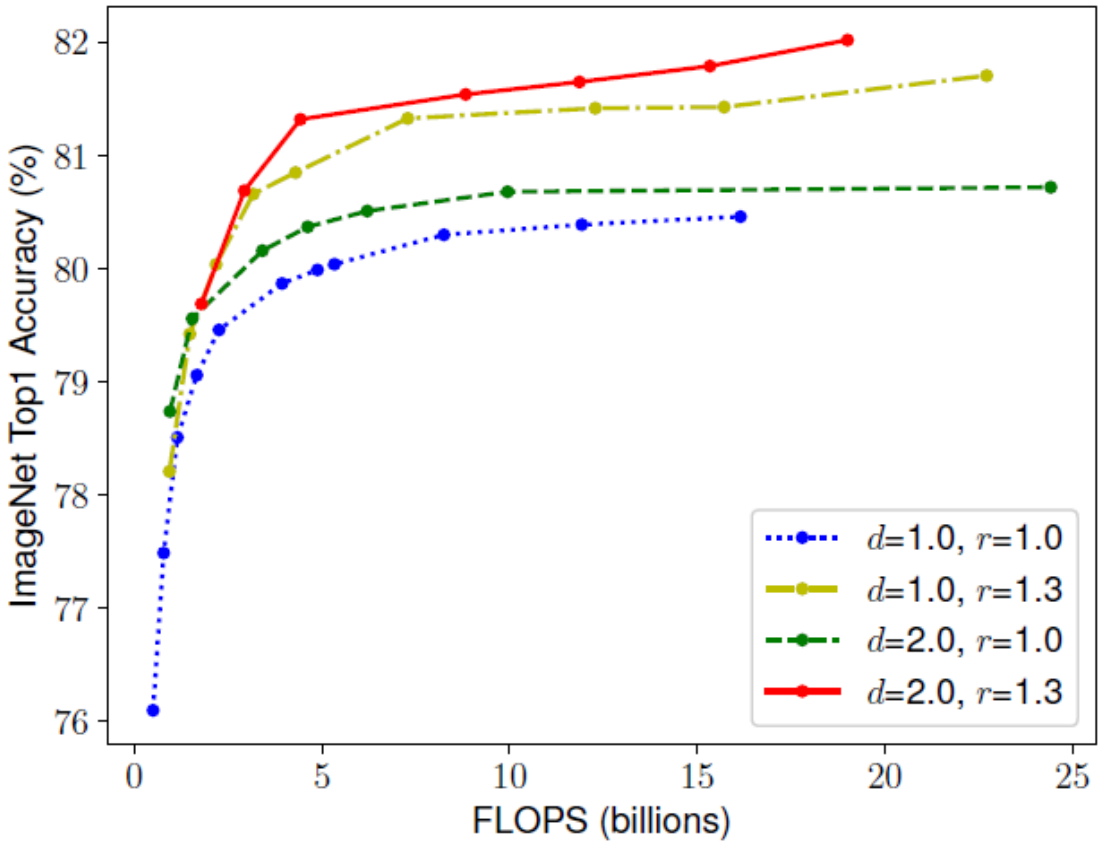
- **Resolution**:
  - Basically giving more finer details to capture for the net into the input
  - CNNs starts taking in with resolutions from 224x224(VGG Net) to 480x480(GPipe) recently to 600x600
  - higher the resolution input higher is the accuracy
  - but still plateaus for very high resolution inputs



- ishes for very high resolutions ($r = 1.0$ denotes resolution 224x224 and $r = 2.5$ denotes resolution 560x560).

  - ***Observation 1***: **Scaling up the depth, width, resolution increases accuracy**

- **Compound Scaling**:



- As we can see the net with a balance of resolution and depth gives the pest performances whereas the others plateaus out at lower accuracies

In this paper, we propose a new **compound scaling method**, which use a compound coefficient ϕ to uniformly scales network width, depth, and resolution in a principled way:

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

where α, β, γ are constants that can be determined by a small grid search. Intuitively, ϕ is a user-specified coefficient that controls how many more resources are available for model scaling, while α, β, γ specify how to assign these extra resources to network width, depth, and resolution respectively. Notably, the FLOPS of a regular convolution op is proportional to d, w^2, r^2, i.e., doubling network depth will double FLOPS, but doubling network width or resolution will increase FLOPS by four times. Since convolution ops usually dominate the computation cost in ConvNets, scaling a ConvNet with equation 3 will approximately increase total FLOPS by $(\alpha \cdot \beta^2 \cdot \gamma^2)^\phi$. In this paper, we constraint $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$ such that for any new ϕ, the total FLOPS will approximately[3] increase by 2^ϕ.

-
- Here, FLOPS - Floating point operations per second
- 4x Depth = 2x Width = 2x Resolution = 4x FLOPS

- **EfficientNet Architecture**:
  - Here, fixing the conv layer/operator simplifies the architecture design problem for computational resources but restricts us to explain the different other parameters like Li, Ci, Hi, Wi
  - Since scaling the model doesn't change the layer operators, Having a good Baseline model is crucial
  - The baseline architecture is decided by optimizing using grid search between $ACC(m)\ vs\ [FLOPS(m)/T]^w$

- Where $ACC(m)$ is the accuracy of the model, $FLOPS(m)$ is the Floating point operations per second of the model, $T = 400M$ is the target FLOPS(available resources) and $w = -0.7$ is a hyperparameter
- **STEP 1**:

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

-

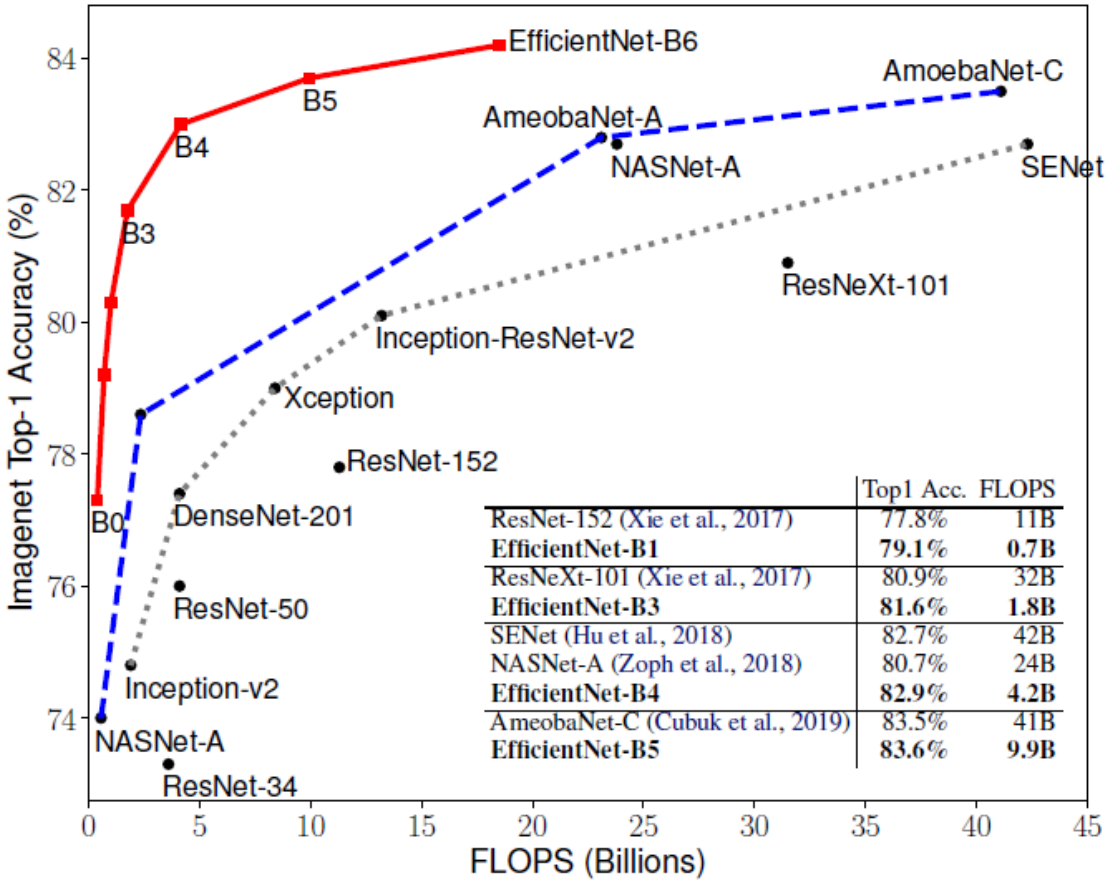
$$\begin{aligned} \max_{d,w,r} \quad & Accuracy(\mathcal{N}(d,w,r)) \\ s.t. \quad & \mathcal{N}(d,w,r) = \bigodot_{i=1\ldots s} \hat{\mathcal{F}}_i^{d\cdot\hat{L}_i}\left(X_{\langle r\cdot\hat{H}_i, r\cdot\hat{W}_i, w\cdot\hat{C}_i\rangle}\right) \\ & \text{Memory}(\mathcal{N}) \leq \text{target\_memory}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

- Taking these 2 equations as constraints and fixing $psi = 1$ and so we perform small grid search on Alpha, Beta, Gamma
- STEP 1: we first fix $\phi = 1$, assuming twice more resources available, and do a small grid search of α, β, γ based on Equation 2 and 3. In particular, we find the best values for EfficientNet-B0 are $\alpha = 1.2, \beta = 1.1, \gamma = 1.15$, under constraint of $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$.
- **STEP 2**:
- fix Alpha, Beta, Gamma and scale up psi under step 1's constraints
- STEP 2: we then fix α, β, γ as constants and scale up baseline network with different ϕ using Equation 3, to obtain EfficientNet-B1 to B7 (Details in Table 2).

- **Results***:

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPs	Ratio-to-EfficientNet
EfficientNet-B0	**77.1%**	**93.3%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x

- As we can see on comparing other nets on comparing it to the current EfficientNet-B0 we are fraction of Parameters and FLOPS it achieves excellent accuracies which are even better than the other best models like ResNet or DenseNet



	Top1 Acc.	FLOPS
ResNet-152 (Xie et al., 2017)	77.8%	11B
EfficientNet-B1	**79.1%**	**0.7B**
ResNeXt-101 (Xie et al., 2017)	80.9%	32B
EfficientNet-B3	**81.6%**	**1.8B**
SENet (Hu et al., 2018)	82.7%	42B
NASNet-A (Zoph et al., 2018)	80.7%	24B
EfficientNet-B4	**82.9%**	**4.2B**
AmeobaNet-C (Cubuk et al., 2019)	83.5%	41B
EfficientNet-B5	**83.6%**	**9.9B**

Figure 5. **FLOPS vs. ImageNet Accuracy** – Similar to Figure 1 except it compares FLOPS rather than model size.

- And these are other circuital networks comparing to EfficientNets
- **Training Details**:

We train our EfficientNet models on ImageNet using similar settings as (Tan et al., 2019): RMSProp optimizer with decay 0.9 and momentum 0.9; batch norm momentum 0.99;

weight decay 1e-5; initial learning rate 0.256 that decays by 0.97 every 2.4 epochs. We also use SiLU (Swish-1) activation (Ramachandran et al., 2018; Elfwing et al., 2018; Hendrycks & Gimpel, 2016), AutoAugment (Cubuk et al., 2019), and stochastic depth (Huang et al., 2016) with survival probability 0.8. As commonly known that bigger models need more regularization, we linearly increase dropout (Srivastava et al., 2014) ratio from 0.2 for EfficientNet-B0 to 0.5 for B7. We reserve 25K randomly picked images from the `training` set as a `minival` set, and perform early stopping on this `minival`; we then evaluate the early-stopped checkpoint on the original `validation` set to report the final validation accuracy.

- SiLU = $x(1/1 + e^{-}x)$
- **Transfer learning with EfficientNet**:
  - 8 other small datasets which are generally used for fine-tuning purposes were tested on EfficientNet models
  - EfficientNet models surpasses 5 out of 8 dataset's best trained models like NASNet, InceptionNet-v4, DAT, Gpipe etc, but only by using 9.6x fewer parameter
- **Comparing the activation maps with other scaling methods**:

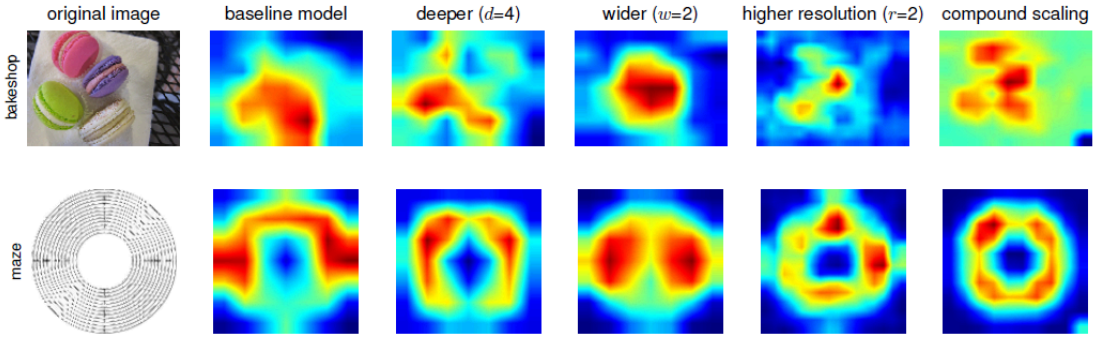


Figure 7. **Class Activation Map (CAM) (Zhou et al., 2016) for Models with different scaling methods-** Our compound scaling method allows the scaled model (last column) to focus on more relevant regions with more object details. Model details are in Table 7.

- We can observe the intuition of the points below,
  - **Depth**: increases the capacity of the model
  - **Width**: increases the receptive field
  - **Channels**: captures more fine-grained patterns
- from these activation maps.

----------END----------