The EfficientNet architecture needs to be described in detail and thoroughly explain all its concepts and innovations.

EfficientNet is a convolutional neural network architecture and scaling method that uniformly scales all dimensions—depth, width, or resolution—using a composite scaling factor. Many consider it to be the most powerful CNN architecture. Unlike conventional methods, which scale these factors arbitrarily, EfficientNet scales the network's width, depth, and resolution uniformly with a set of fixed scaling coefficients. For example, if we want to use 2^n times more computational resources, we can simply increase the network depth by α^n, the width by β^n, and the image size by γ^n, where α, β, and γ are fixed coefficients determined through a search on a small baseline model.

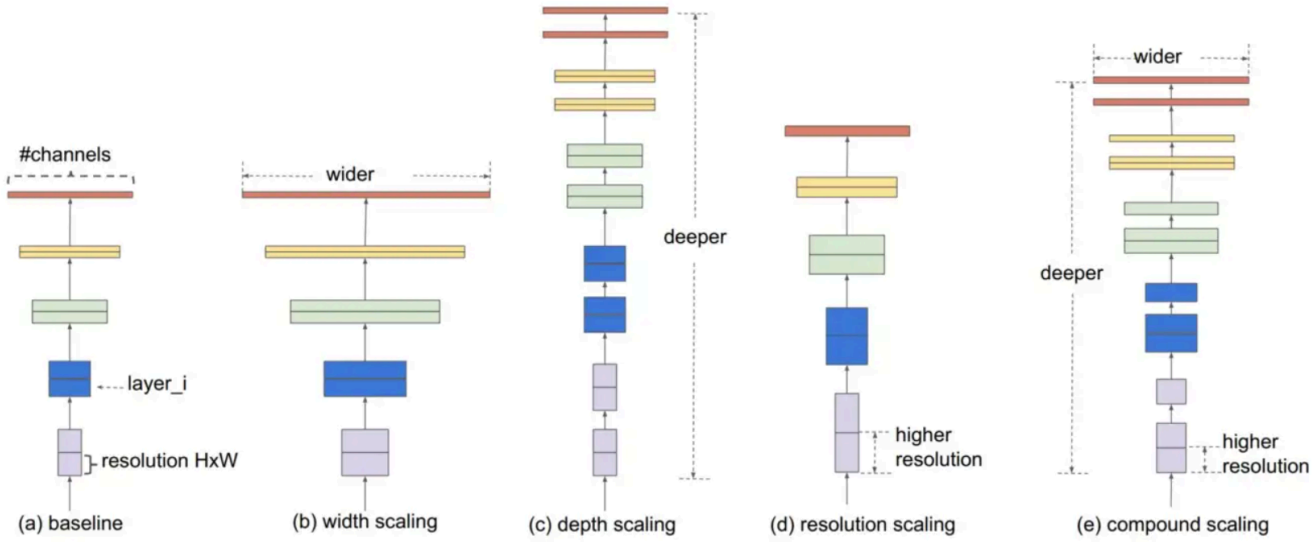
EfficientNet uses a technique called compound scaling to scale models in a simple yet effective way. Instead of scaling width, depth, or resolution randomly, compound scaling uniformly scales each dimension with a fixed set of scaling coefficients. Using this method, its developers created seven models of different sizes that surpassed the accuracy of the most advanced convolutional neural networks while being much more efficient.

$$\text{Depth } d = \alpha^{\emptyset}, \text{Width } w = \beta^{\emptyset}, \text{Resolution } r = \gamma^{\emptyset}, (1)$$

$$\text{such that } \alpha . \beta^2 . \gamma^2 \approx 2$$

$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

Compound Scaling: In this approach, experts found that while scaling one dimension at a time improves model performance, balancing scales across all three dimensions—width, depth, and image resolution—optimally enhances the overall model performance.

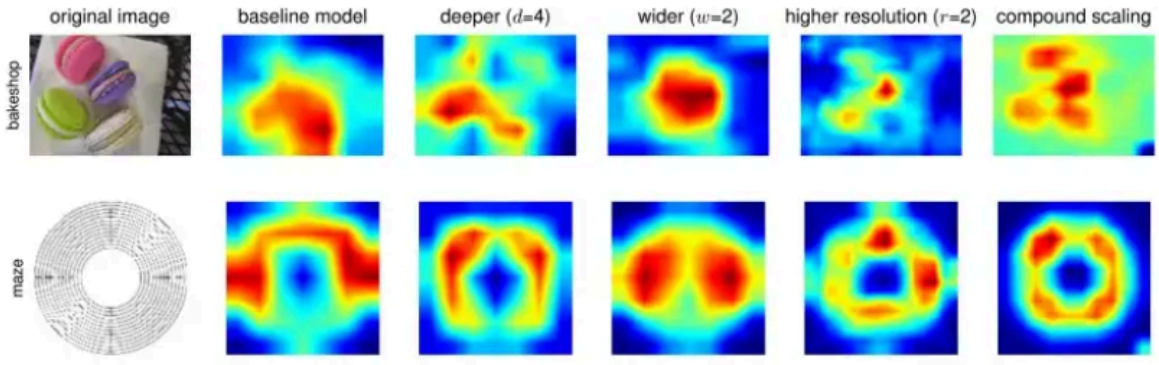


The values of α, β, and γ are determined using a network search algorithm. ϕ determines the computational resources of the network and is defined by the user. These values are limited by a factor showing that γ^2×β^2×α is approximately 2. FLOPS (floating-point operations per second) for a convolution operation scales as d×w^2×r^2, indicating that FLOPS double when network depth is doubled and quadruple when network width or resolution is doubled. In CNNs, convolutional operations control computational costs. Scaling the network using the above equation increases FLOPS. Since γ^2×β^2×α is approximately 2, for each new ϕ, the total FLOPS increases by 2^ϕ.

When ϕ is set to 1, and assuming that twice the resources are available, performing a small network search gives us the values for α, β, and γ. For ϕ = 1, the values are α = 1.2, β=1.1, and γ=1.15, maintaining that γ^2×β^2×α=2. This model is EfficientNet-B0. In the next step, the values of α, β, and γ are fixed, and the baseline network is scaled with different values of ϕ to obtain EfficientNet-B1 to EfficientNet-B7 models. Better performance might be achieved by searching for α, β, and γ in larger models, but the search cost becomes significantly higher. Therefore, to address this issue, the search is conducted from the beginning using a small network.

If the input image is larger, the network requires more layers to increase the receptive field and more channels to capture more patterns on the larger image. The compound scaling technique also helps to improve the efficiency and accuracy of previous CNN models, such as MobileNet and ResNet, achieving accuracy improvements of 1.4% and 0.7%, respectively, compared to other random scaling methods.

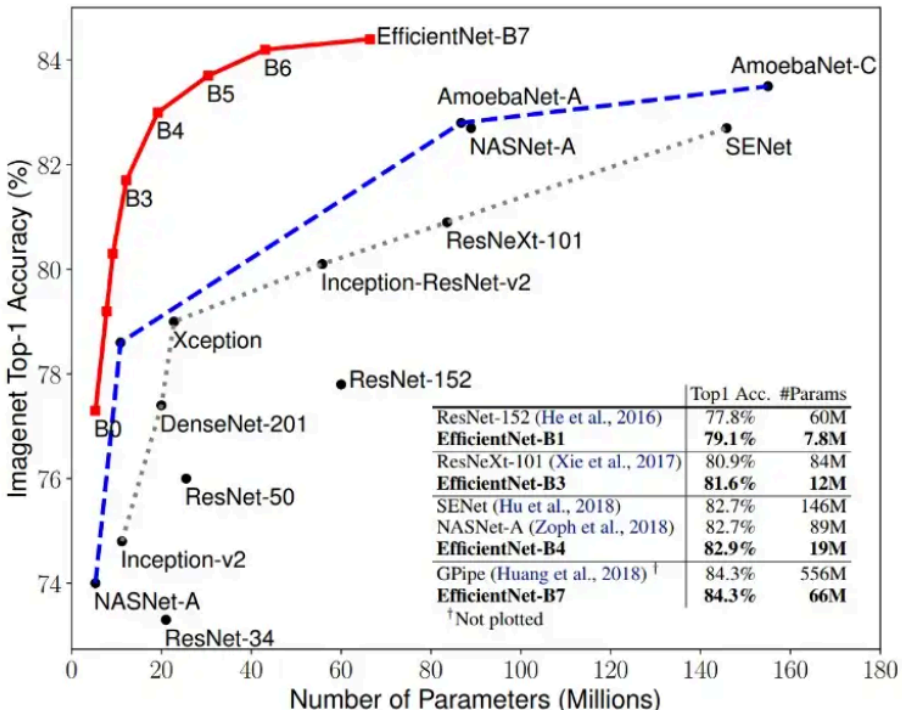
As you can see in the figure below, this model provides better Class Activation Maps (CAM), which focus more on relevant areas with greater detail and pave the way for better model interpretability.



As shown in the figure, the compound scaling method allows the scaled model (last column) to focus more on relevant areas with finer details of the object.

Model Performance: The performance of this model has been very good compared to other models, with the largest version, EfficientNet-B7, achieving state-of-the-art performance on the ImageNet and CIFAR-10 datasets. On ImageNet, it achieved approximately 84.4% accuracy (Top-1) and 97.3% accuracy (Top-5). Additionally, the model size was 8.4 times smaller and 6.1 times faster than the previous CNN models. It also achieved 91.7% accuracy on the CIFAR-10 dataset and 98.8% accuracy on the Flowers dataset.

TOP - 5 TOP - 1



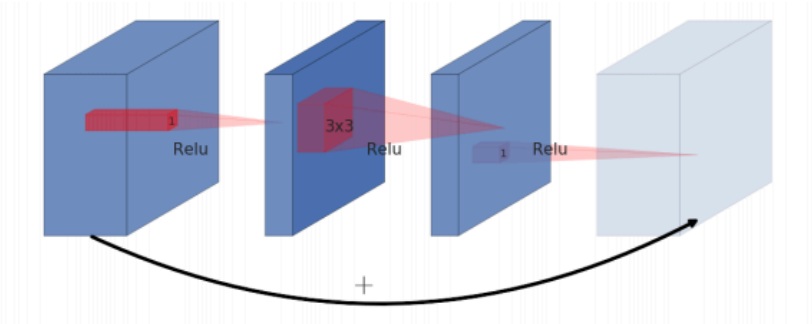
	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018)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

†Not plotted

Model Architecture: The building block of this architecture is the mobile inverted bottleneck, known as MBConv, which is also referred to as the inverted residual block with an additional SE (Squeeze-and-Excitation) block. These two blocks are explained below:

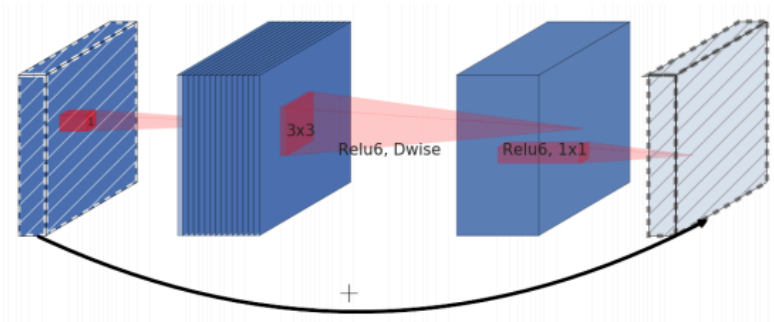
MBConv: To ensure information flows efficiently through deep neural networks, we typically use residual blocks. Residual blocks connect the start and end of a convolution block with skip connections. Channels in the convolution block first widen at the start, narrow through the depth of the block, and widen again at the end due to the added information. Therefore, the structure of a typical residual block is wide > narrow > wide in terms of channel count.

(a) Residual block

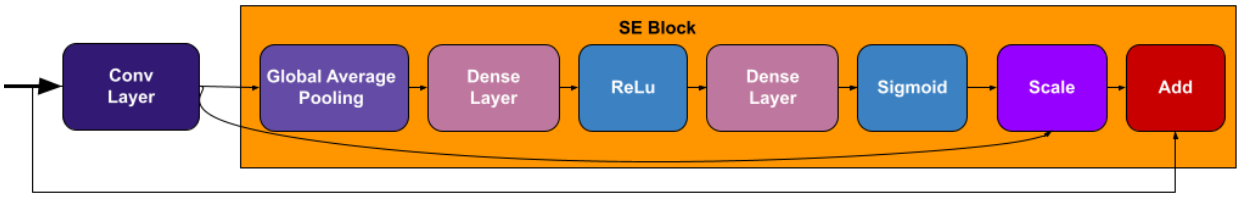


However, the inverted residual block follows the opposite pattern: narrow > wide > narrow.

(b) Inverted residual block



SE: These building blocks are used in CNNs to enhance the inter-channel dependency by recalibrating the dynamic feature channels. This means that instead of equally weighting all channels, the network dynamically assigns higher weights to the most important channels. The figure below shows the components of this block.



As a result, this model outperforms all previous CNN models on most benchmark datasets. Furthermore, the compound scaling method can also be applied to effectively scale other CNN architectures. This allows EfficientNet models to be scaled in such a way that they achieve higher power with fewer parameters and FLOPS on ImageNet and other commonly used machine learning datasets.

The ResNext and the Inception-ResNet should be compared and discussed in terms of their advantages and disadvantages.

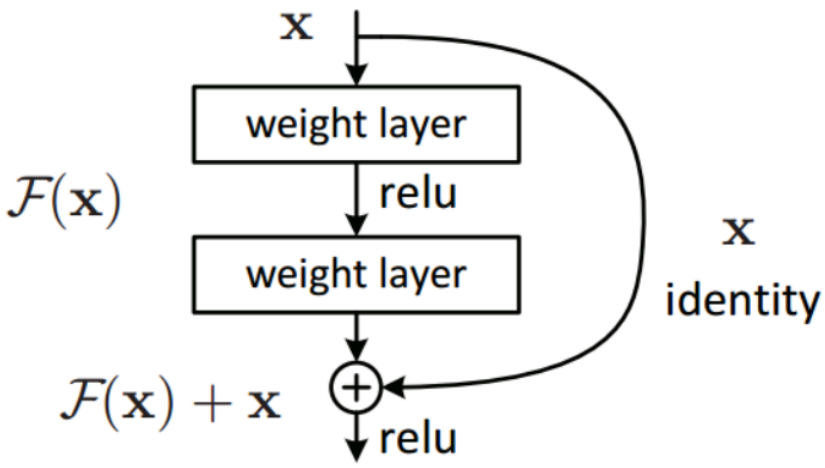
The Inception model was created to reduce the computational load of deep neural networks while achieving advanced performance. As networks become deeper, computational efficiency decreases. Therefore, when developing Inception, the focus was on finding a solution to scale up neural networks without increasing computational costs. We know that ResNet focuses on computational cost, but Inception-ResNet focuses on computational accuracy. Intuitively, deeper networks should not perform worse than shallower networks, but in practice, deeper networks often underperform due to optimization challenges, not overfitting. In short, the deeper the network, the harder it is to optimize.

ResNeXt introduces a new dimension called cardinality as a key factor, in addition to depth and width. ResNeXt architecture extends the deep residual network by replacing the standard residual block with a block that uses the "split-transform-merge" strategy (i.e., branching paths within a cell), which is also used in Inception models. Simply put, instead of performing convolution over the entire input feature map, the input is split into lower-dimensional channels, and separate convolution filters are applied before merging the results.

Inception employs a special technique called skip connection to address the vanishing gradient problem. The main idea is that instead of mapping $x \rightarrow f(x)$, you map $x \rightarrow f(x)+g(x)$, where:

- x: Input
- f(x): Output
- g(x): Residual

When x equals f(x), meaning the input equals the output, g(x) becomes zero. Thus, the CNN learns to map the input to the output by zeroing out the weights of the intermediate layers.



Additionally, using the ReLU activation function provides an extra advantage by eliminating the vanishing gradient issue.

This model is primarily designed for detecting various features in an image. It can use different kernels to detect different entities in the image depending on their importance and area within the frame. Another advantage is that it expands the network space, allowing the best network to be selected through training. Each initial module can capture prominent features at different levels. On the other hand, this model is highly engineered, using many tricks to improve both speed and accuracy. One of its positive attributes is its continuous evolution.

However, due to the complex architecture and structure of its internal modules, the training and testing time for the Inception-ResNet model is generally longer than for other models. Additionally, it has relatively high computational costs compared to models like MobileNet and AlexNet.