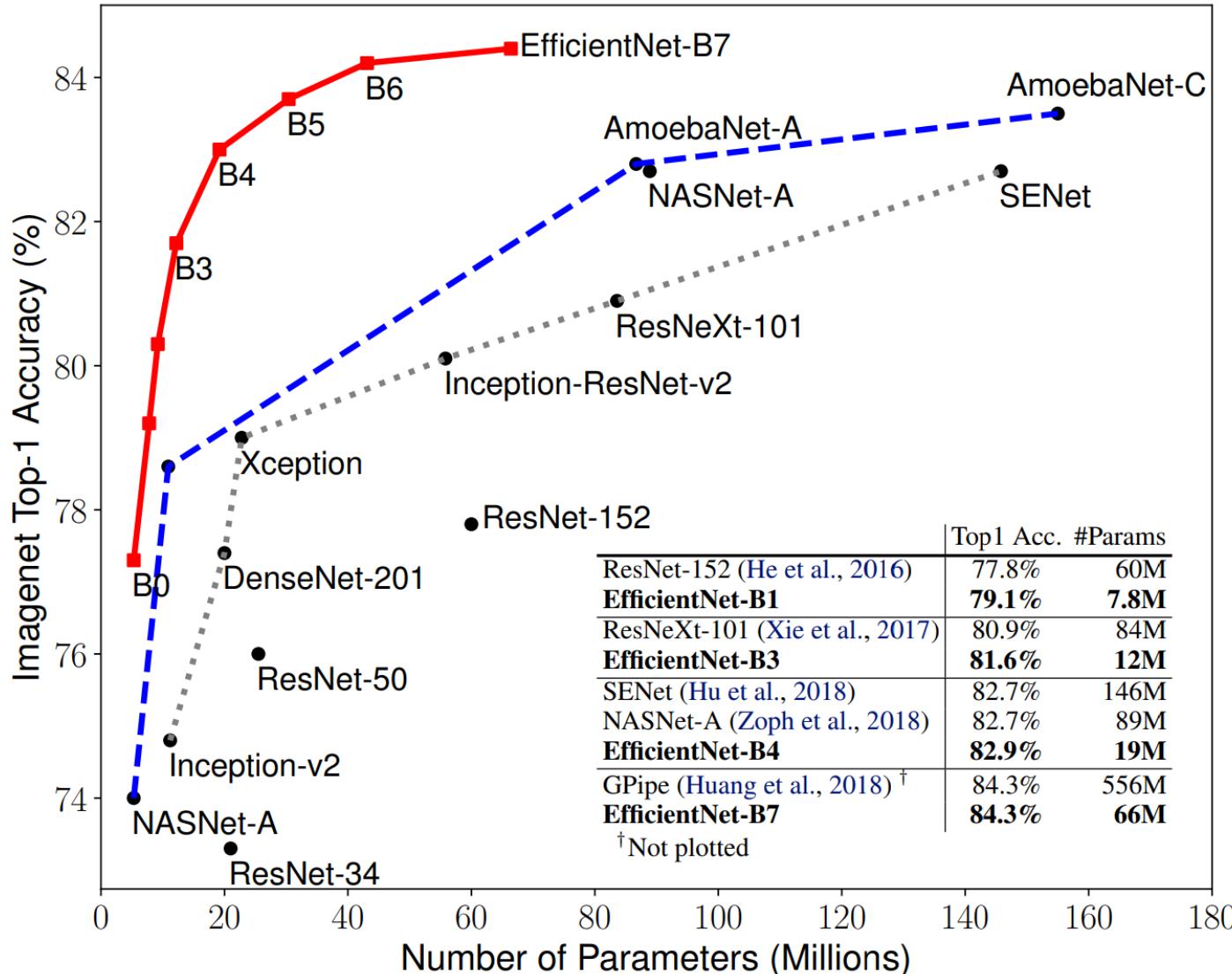


# What is EfficientNet?

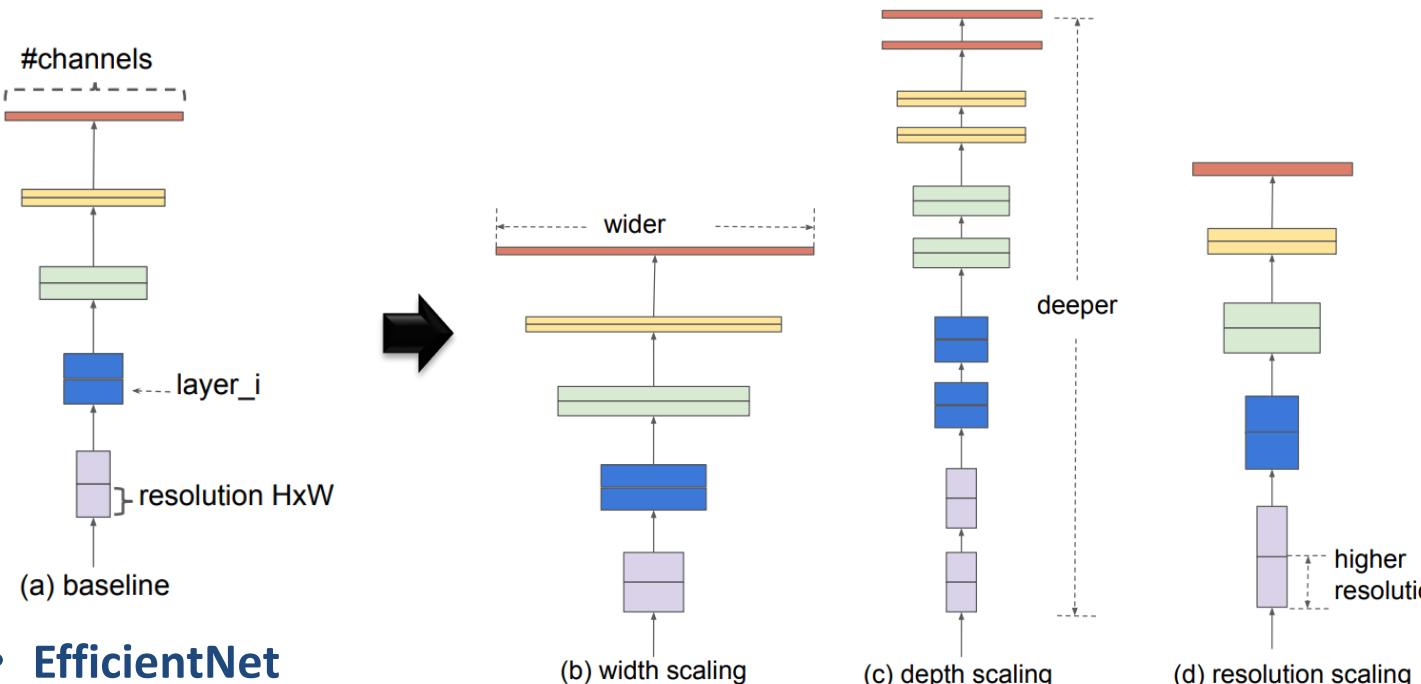
## EfficientNet: A New Way to Scale CNNs

- Proposed by Google (2019)
- Goal
  - better accuracy + fewer parameters
- Key idea
  - Compound Scaling
    - Instead of manually scaling depth/width/resolution, scale all together
- Achieves state-of-the-art accuracy with **high efficiency**



# Compound Scaling Explained

- How EfficientNet Scales Models Efficiently
  - Previous models scale only one aspect (depth or width or resolution)



## EfficientNet

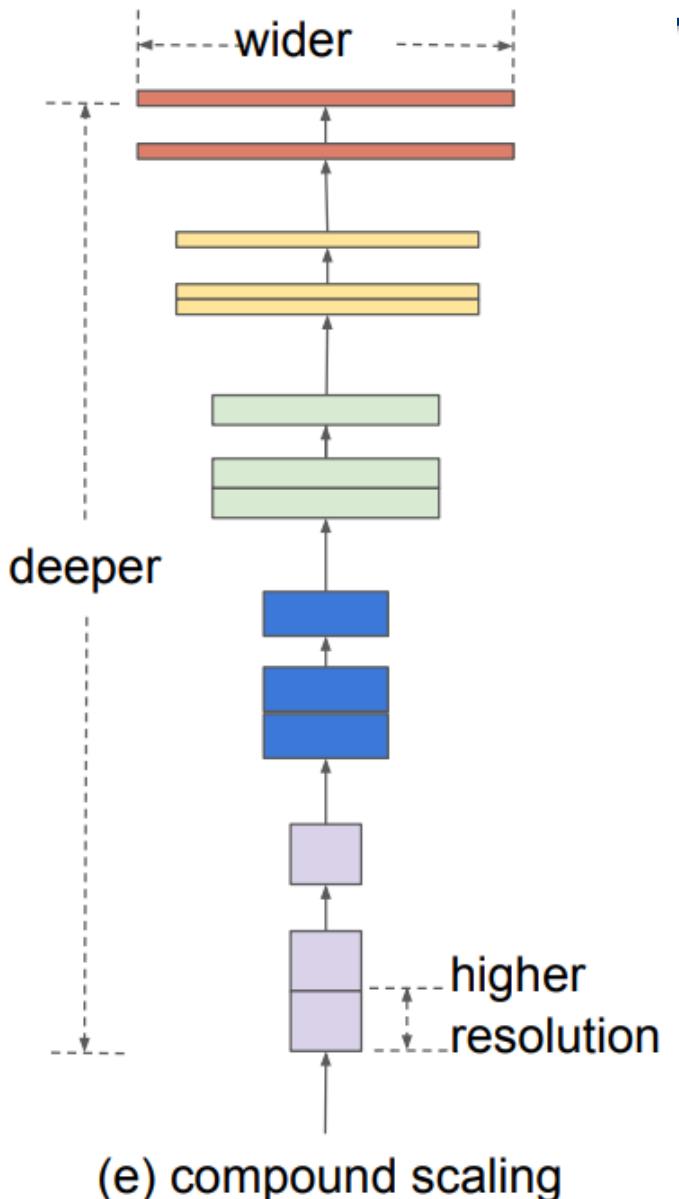
- Scale all 3 in a **balanced** way using a compound coefficient  $\phi$

### Scaling formula

✓ Depth:  $d = \alpha^\phi$ , Width:  $w = \beta^\phi$ , Resolution:  $r = \gamma^\phi$

### Constraint

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



(e) compound scaling

# EfficientNet's Compound Scaling – Mathematical View

---

- How EfficientNet Scales Architectures under Constraints

- EfficientNet assumes a **fixed layer structure** (pre-searched via NAS)
- Rather than redesigning the layers, it scales
  - Depth (number of layers):  $d$
  - Width (number of channels):  $w$
  - Resolution (input size):  $r$
- The goal is to **maximize accuracy** under **(1) Memory constraint and (2) FLOPs constraint**
- **Optimization Objective**
  - $\max_{d,w,r} \text{Accuracy}(\mathcal{N}(d,w,r))$ ; where  $\mathcal{N}$  is a model
  - *subject to:*  $\begin{cases} \text{Memory}(\mathcal{N}) \leq \text{target memory} \\ \text{FLOPs}(\mathcal{N}) \leq \text{target flops} \end{cases}$

# EfficientNet's Compound Scaling – Mathematical View

---

- How EfficientNet Scales Architectures under Constraints

- Optimization Objective

- $\max_{d,w,r} \text{Accuracy}(\mathcal{N}(d, w, r))$

- *subject to:*  $\begin{cases} \text{Memory}(\mathcal{N}) \leq \text{target memory} \\ \text{FLOPs}(\mathcal{N}) \leq \text{target flops} \end{cases}$

- Interpretation to Optimization Objective

- $\mathcal{N}$ : CNN model with depth  $d$ , width  $w$ , and resolution  $r$

- Scaling is applied to input shape  $(H_i, W_i, C_i)$  as

- ✓  $r \cdot H_i, r \cdot W_i, w \cdot C_i$  while increasing model depth  $d \cdot L_i$

# EfficientNet Architecture

---

- **Building Blocks of EfficientNet**

- Based on **MobileNetV2 MBConv blocks**
- Key components
  - **MBConv**: Depthwise separable conv + expansion + projection
  - **SE block (Squeeze-and-Excitation)**: channel-wise attention
  - **Swish activation**: smooth and non-monotonic
- Progressive stage-wise scaling of the backbone

# EfficientNet Architecture

---

- Building Blocks of EfficientNet

- Based on **MobileNetV2 MBConv blocks**

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	$224 \times 224$	32	1
2	MBConv1, k3x3	$112 \times 112$	16	1
3	MBConv6, k3x3	$112 \times 112$	24	2
4	MBConv6, k5x5	$56 \times 56$	40	2
5	MBConv6, k3x3	$28 \times 28$	80	3
6	MBConv6, k5x5	$14 \times 14$	112	3
7	MBConv6, k5x5	$14 \times 14$	192	4
8	MBConv6, k3x3	$7 \times 7$	320	1
9	Conv1x1 & Pooling & FC	$7 \times 7$	1280	1

- Key components

- **MBConv**: Depthwise separable conv + expansion + projection
- **SE block (Squeeze-and-Excitation)**: channel-wise attention
- **Swish activation**: smooth and non-monotonic

- Progressive stage-wise scaling of the backbone

# EfficientNet Architecture

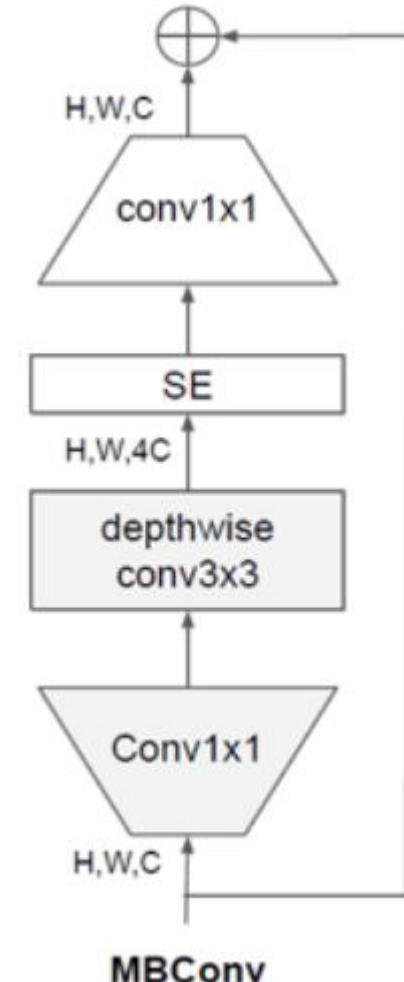
## ■ MBConv – Mobile Inverted Bottleneck

- **MBConv: Efficient Block for Lightweight CNNs**

- Introduced in **MobileNetV2**, reused in EfficientNet
- **Inverted bottleneck**
  - ✓ **Expand** → Depthwise Conv → SE Block → Project (1x1 conv)
- Includes residual connection if input and output shapes match

- **MBConv Structure**

- **1. Expansion (1x1 conv)**: increases channel size
- **2. Depthwise Conv (3x3)**: lightweight spatial feature extraction
  - ✓ Discussed in MobileNet (applying convolution to each channel)
- **3. SE Block**: Squeeze and excitation
- **4. Projection (1x1 conv)**: reduces channels back
- **5. Residual connection (optional)**
  - ✓ Efficient gradient propagation



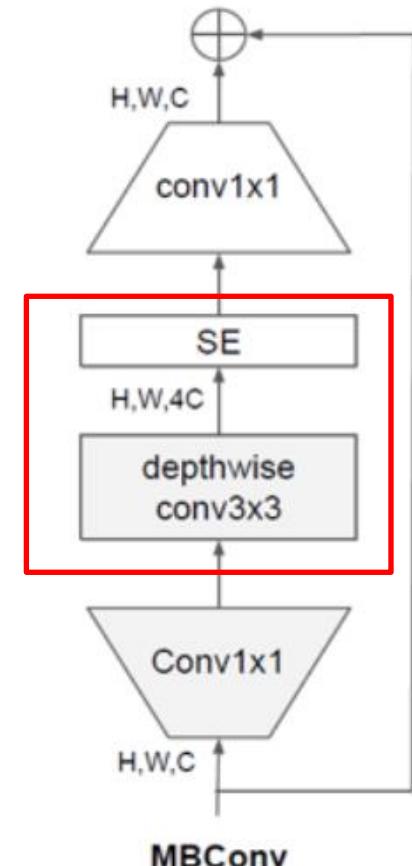
# EfficientNet Architecture

## ■ SE Block – Squeeze and Excitation in MBconv

### • Motivation – Why Channel Attention?

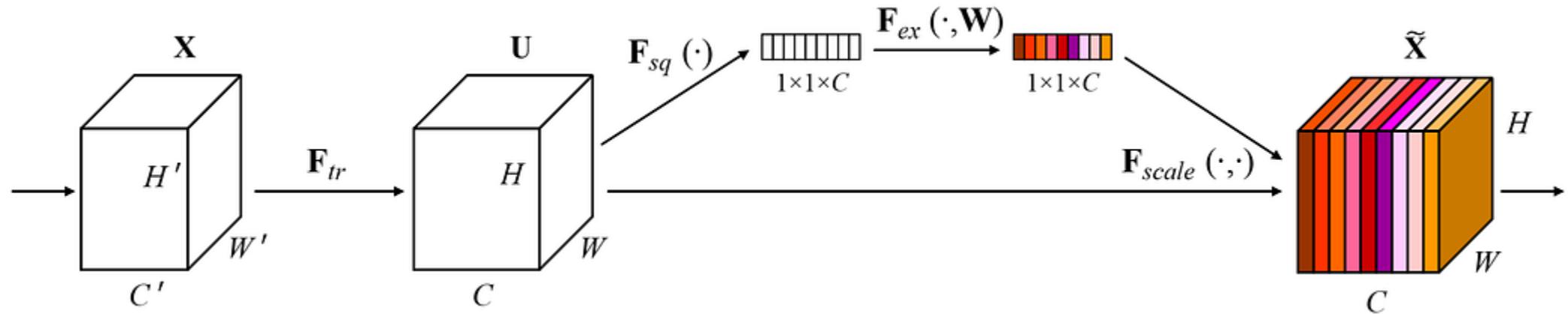
- Traditional convolutions primarily focus on **spatial information** within local regions (receptive field).
- Each channel is treated **independently**, without considering **inter-channel dependencies**.
- In **MBConv** (used in EfficientNet), **depthwise convolution** is applied per channel
  - ✓ This means there is **no interaction across channels** during spatial filtering.
  - ✓ The model cannot learn which channels are more important or how they relate to each other.

“Depthwise convolutions extract spatial features **independently for each channel**, but **do not capture relationships between channels**.”



# EfficientNet Architecture

- SE Block – Squeeze and Excitation in MBconv
  - Overview – SE Block Structure



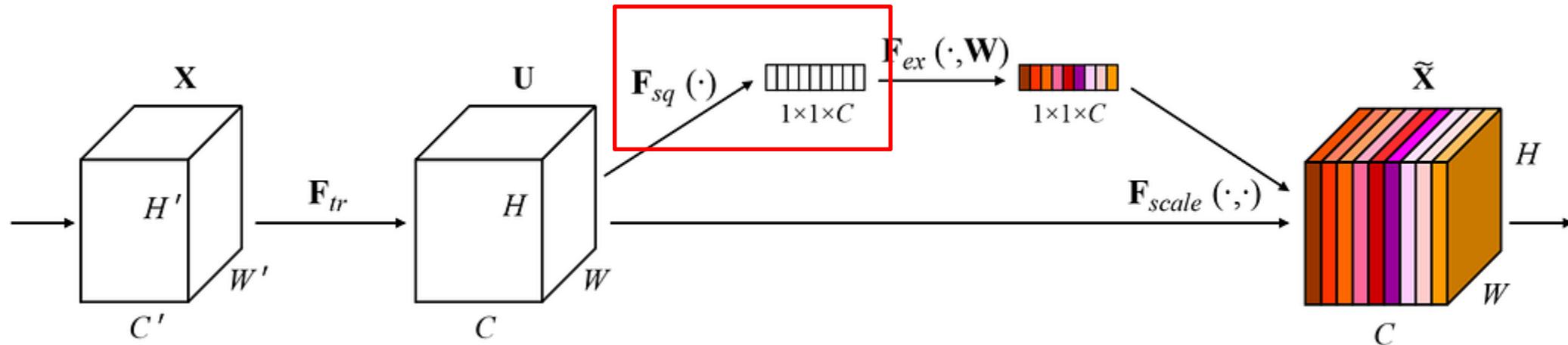
- Step 1.  $F_{tr}$ : Input feature map  $X \rightarrow \mathbf{U}$  via convolution
- Step 2. SE block applies
  - ✓(1) Squeeze  $F_{sq}$ : Global Average Pooling
  - ✓(2) Excitation  $F_{ex}$ : FC layers with ReLU & Sigmoid
  - ✓(3) Scale  $F_{scale}$ : Channel-wise multiplication
- Step 3. output: recalibrated feature map  $\tilde{X}$

# EfficientNet Architecture

- SE Block – Squeeze and Excitation in MBconv

- Squeeze – Global Information Embedding

- Applies **Global Average Pooling** to each channel



$$\checkmark \mathbf{z}_c = F_{sq}(\mathbf{u}_c) = \frac{1}{H \cdot W} \sum_{i=1}^H \sum_{j=1}^W \mathbf{u}_c(i, j) \Rightarrow \mathbf{z} \in \mathbb{R}^{1 \times 1 \times C}$$

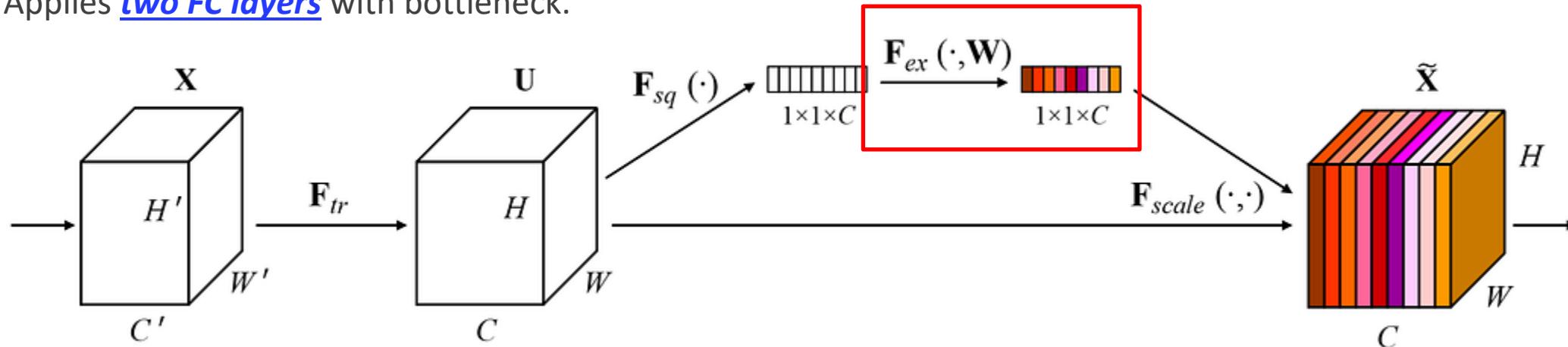
✓ Each channel is compressed into a scalar representing its global activation.

# EfficientNet Architecture

## ■ SE Block – Squeeze and Excitation in MBconv

- **Excitation – Learn Channel Dependencies (i.e., Learning What to Emphasize)**

- Applies two FC layers with bottleneck.



$$\checkmark \mathbf{s} = F_{ex}(\mathbf{z}, \mathbf{W}) = \sigma(g(\mathbf{z}, \mathbf{W}) = \sigma(\mathbf{W}_2 \cdot \delta(\mathbf{W}_1 \cdot \mathbf{z}))$$

➤  $\mathbf{z}$ : input vector (obtained from the squeeze step)

➤  $\delta$ : ReLU for FC1

➤  $\mathbf{W}_1 \in \mathbb{R}^{\frac{C}{r} \times C}$ : reduce channel dimension from  $C \rightarrow C/r$

➤  $\sigma$ : Sigmoid for FC2

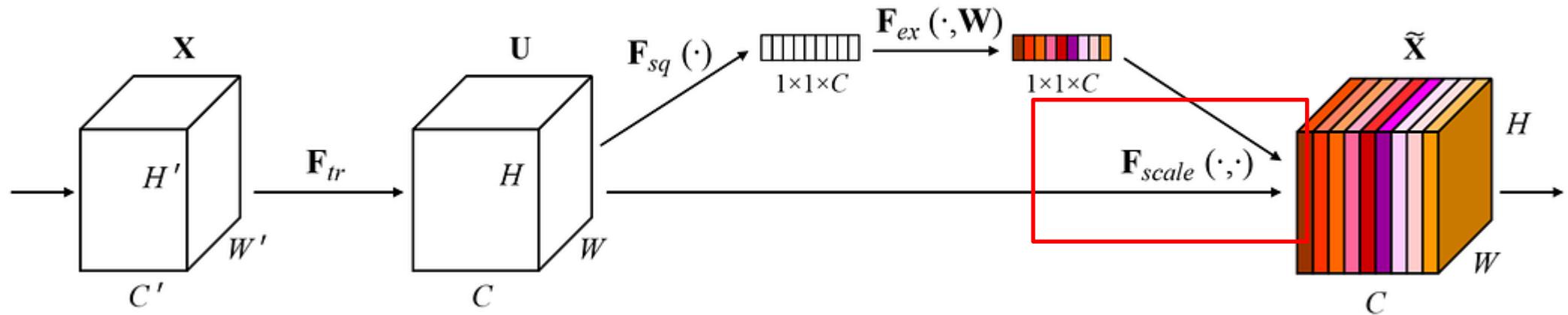
➤  $\mathbf{W}_2 \in \mathbb{R}^{C \times \frac{C}{r}}$ : expand back from  $C/r \rightarrow C$

➤  $r$ : reduction ratio (e.g., 16, 4) – a hyperparameter

**FC1:**  $\mathbf{W}_1 \cdot \mathbf{z} \rightarrow \text{ReLU} \rightarrow \mathbb{R}^{\frac{C}{r}}$   
**FC2:**  $\mathbf{W}_2 \cdot \mathbf{z} \rightarrow \text{Sigmoid} \rightarrow \mathbb{R}^C$

# EfficientNet Architecture

- SE Block – Squeeze and Excitation in MBconv
  - Scale – Channel-wise Recalibration (i.e., Reweighting the Feature Map)



- Output weights  $s \in R^{1 \times 1 \times C}$  are used to rescale  $U$
- Channel-wise multiplication  
$$\check{X}_c = F_{scale}(u_c, s_c) = s_c \cdot u_c$$
- This enhances the informative channels and suppresses less useful ones.

# EfficientNet Architecture

## ■ Swish Activation Function

- A Smooth Activation for Better Gradients

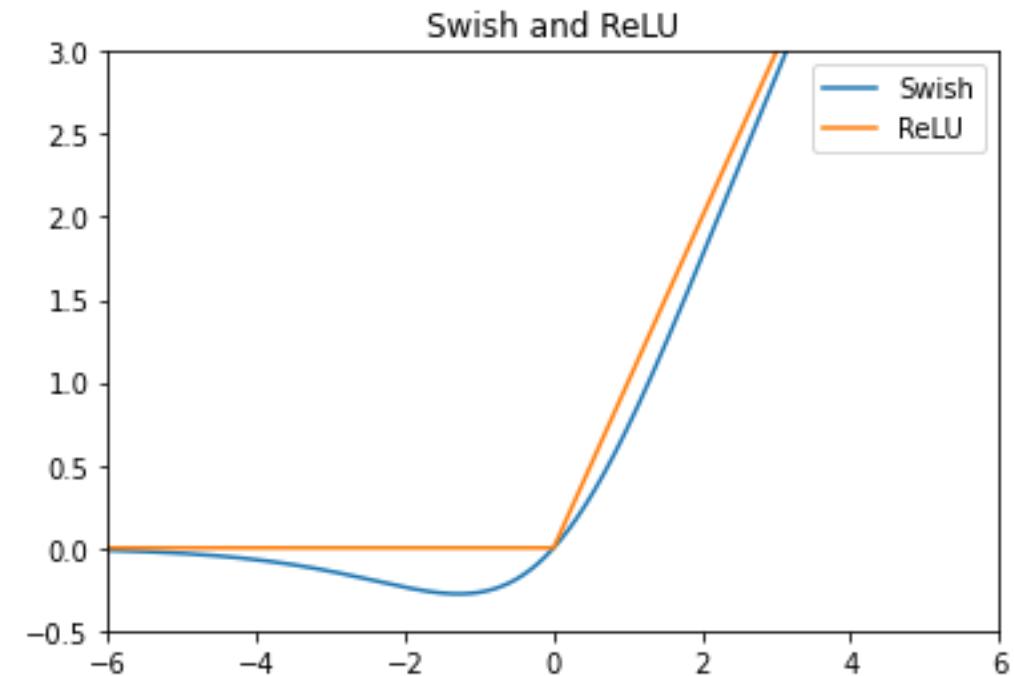
- Proposed by Google researchers

- Formula

- ✓  $\text{Swish}(x) = x \cdot \sigma(\beta x)$  (where  $\beta$  is often set to 1)

- Non-monotonic, smooth, and avoids dying neurons

- A Smooth Activation for Better Gradients



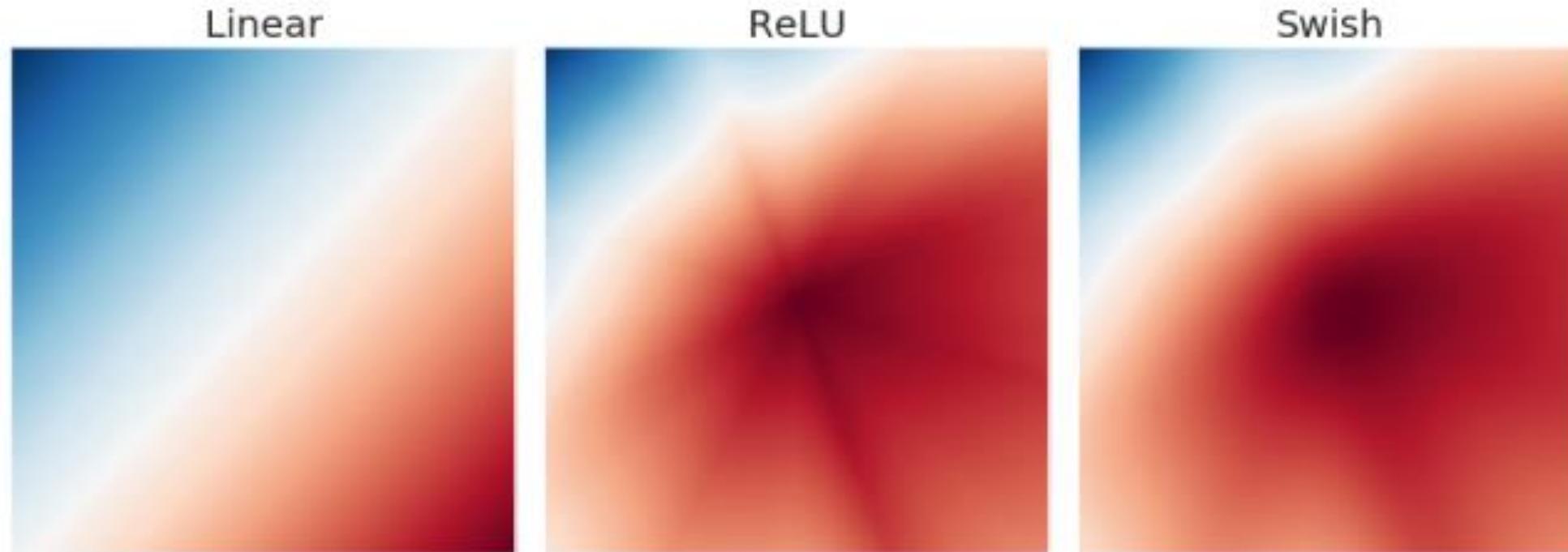
Function	Formula	Smooth?	Monotonic?	Used in
ReLU	$\max(0, x)$	No	Yes	Most CNNs
Swish	$x \cdot \sigma(\beta x)$	Yes	No	EfficientNet

# EfficientNet Architecture

---

## ■ Swish Activation Function

- Activation Map Comparison



- Activation maps from a 6-layer network output.
- ReLU: Sharp edges (star-shaped regions) → sudden activation changes → harder optimization
- Swish: Smoother transition → stable and gradual activation response  
→ This makes it easier for optimizers to follow smooth loss surfaces and avoid local traps

# EfficientNet Performance vs. Other Models

## EfficientNet Achieves Better Accuracy with Fewer FLOPs

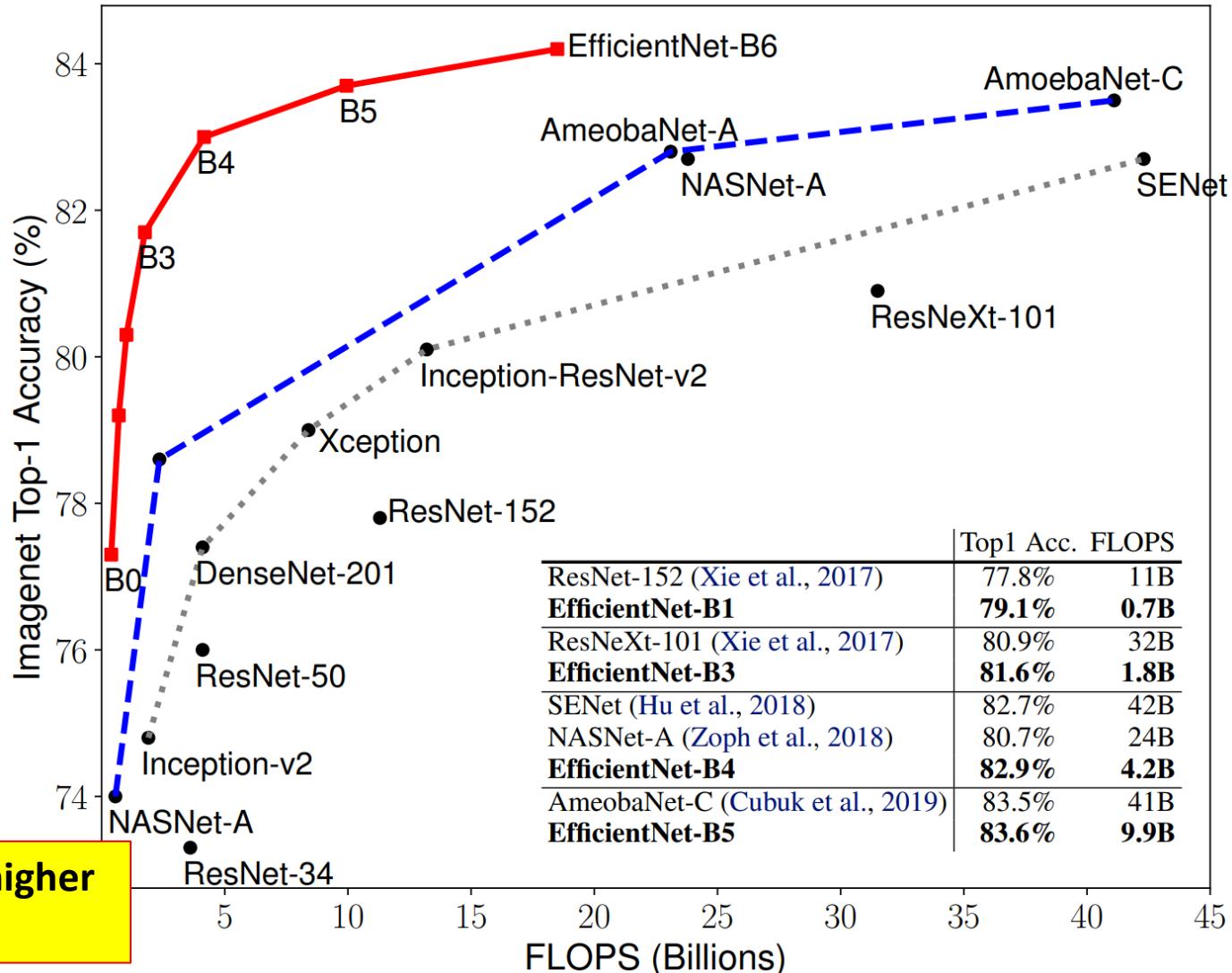
### Top-1 Accuracy vs. FLOPs on ImageNet

- Red Line
  - ✓ EfficientNet family (B0–B6)
- Blue Line
  - ✓ NAS-based models (NASNet, AmoebaNet)
- Gray Dotted Line
  - ✓ Other conventional models (ResNet, SENet, DenseNet, etc.)

### Meaning of B0 to B6

- B0: Base model
- B1–B6: Versions scaled from B0 using compound scaling

With **lower computational cost**, EfficientNet achieves **higher accuracy** than larger models



# MobileNet vs EfficientNet

---

## ■ Comparing MobileNet and EfficientNet

Feature	MobileNet	EfficientNet
Design Strategy	Manual + Heuristic	Neural Architecture Search + Compound Scaling
Building Blocks	Depthwise Separable Conv	MBConv + SE + Swish
Accuracy	Moderate (~70%)	High (up to 84.4%)
Params / FLOPs	Extremely Low	Efficient w/ High Accuracy
Use Cases	Real-time Mobile Vision	General-purpose High-Accuracy Visio