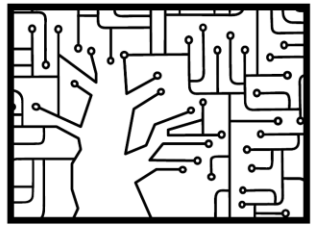
Deep Learning for Computer Vision: Efficient Architectures – Tutorial

Akhiad Bercovich



Agenda

- Measures of efficiency

- Examples of efficient architectures:
  - SqueezeNet
  - MobileNet V1-V3
  - MnasNet
  - EfficientNets
  - RegNets
  - Performers

- Not covering efficiency improving methods (pruning, quantization, etc.)

WAIC

https://machinethink.net/blog/mobile-architectures/

Measuring Efficiency

- Number of parameters
- MAdds/MACs
- FLOPs

- Training/inference time
- Memory usage
- Power consumption

- Hardware specific

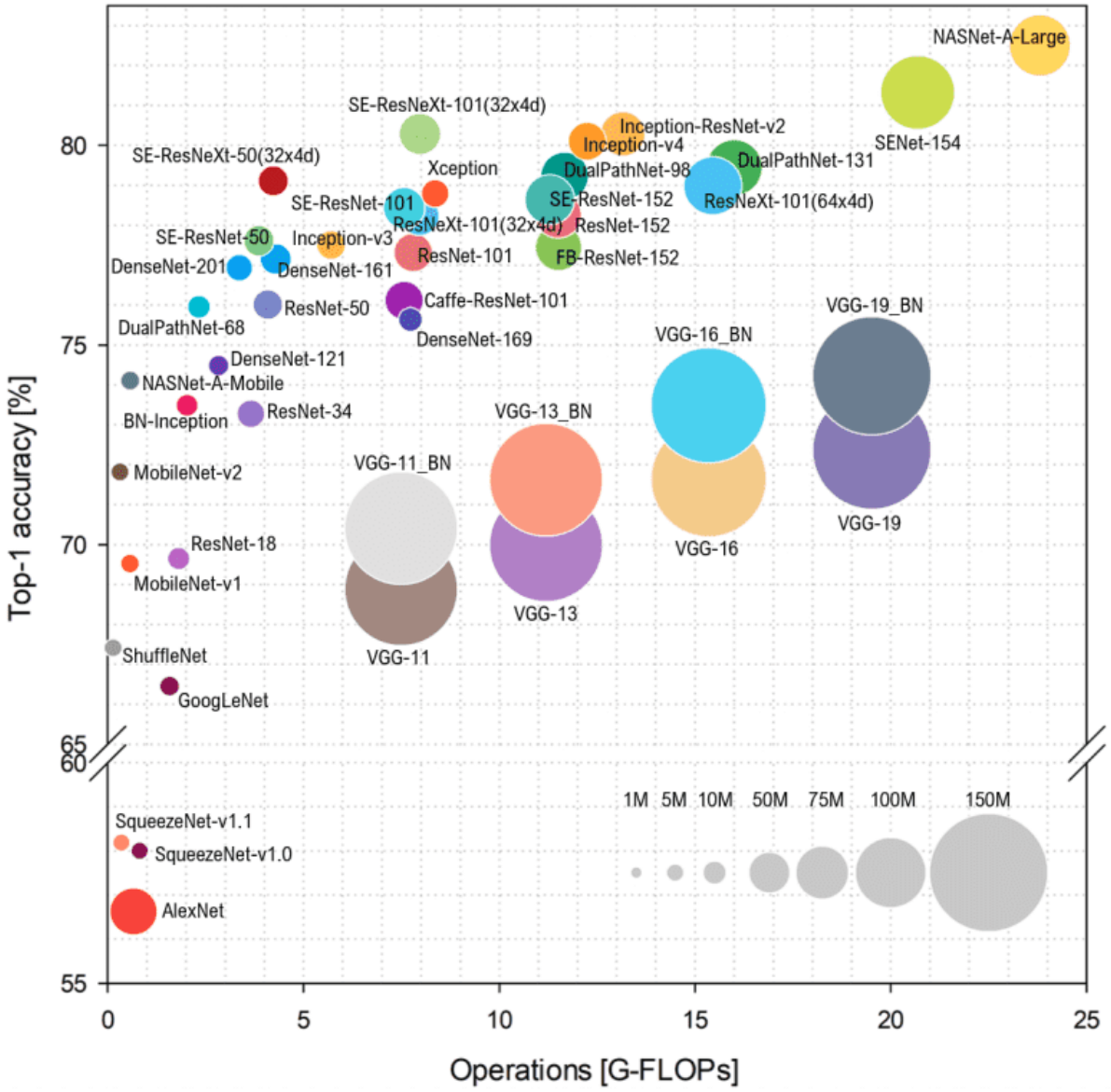
```
pytorch_total_params = sum(p.numel() for p in model.parameters())
pytorch_total_params = sum(p.numel() for p in model.parameters() if p.requires_grad)

import torch.autograd.profiler as profiler

$ from fvcore.nn import FlopCountAnalysis
$ flops = FlopCountAnalysis(model, input)
$ flops.total()
274656
$ flops.by_module_and_operator()
{'': Counter({'conv': 194616, 'addmm': 80040}),
 'conv1': Counter({'conv': 48600}),
 'conv2': Counter({'conv': 146016}),
 'fc1': Counter({'addmm': 69120}),
 'fc2': Counter({'addmm': 10080}),
 'fc3': Counter({'addmm': 840})}
```

```
torch.cuda.Event(enable_timing=True)
torch.cuda.synchronize()
```

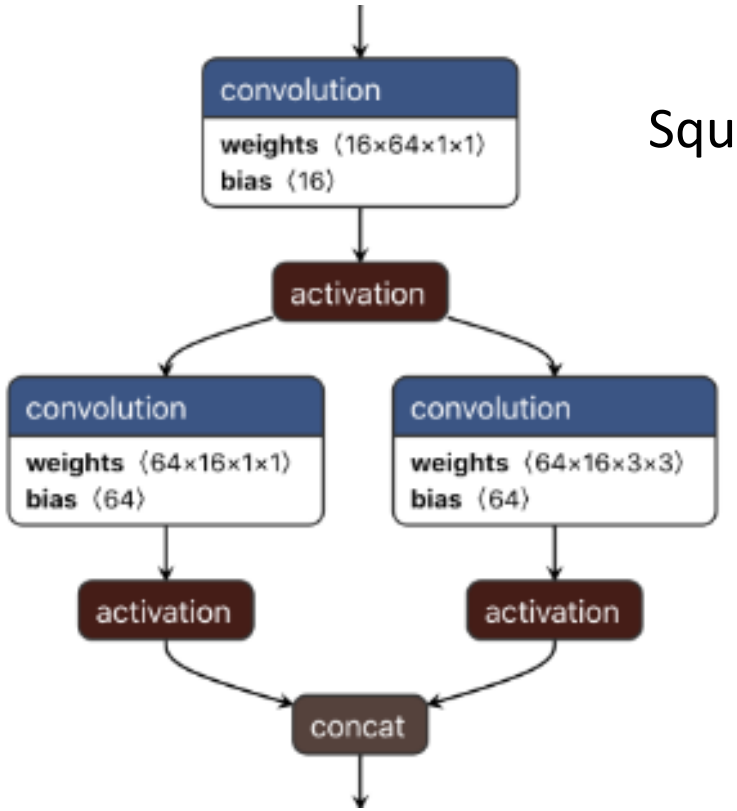
Efficient Architectures



SqueezeNet

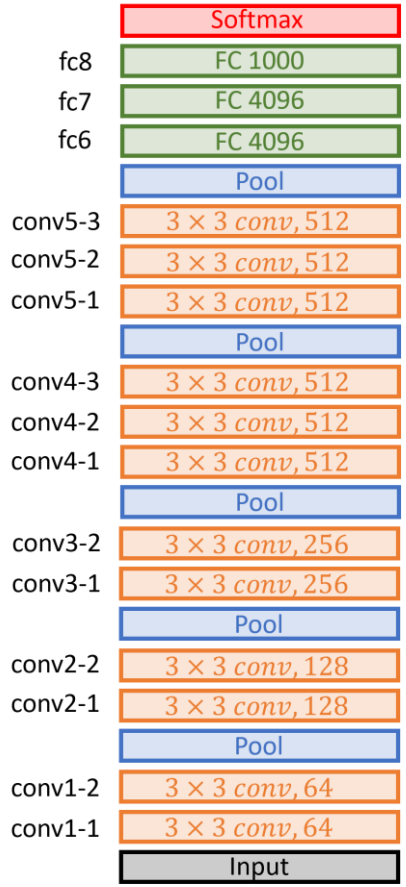
SQUEEZENET: ALEXNET-LEVEL ACCURACY WITH 50X FEWER PARAMETERS AND <0.5MB MODEL SIZE

Fire module



Squeeze

Expand



VGG16

MobileNetV1

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications

Depthwise separable convolution

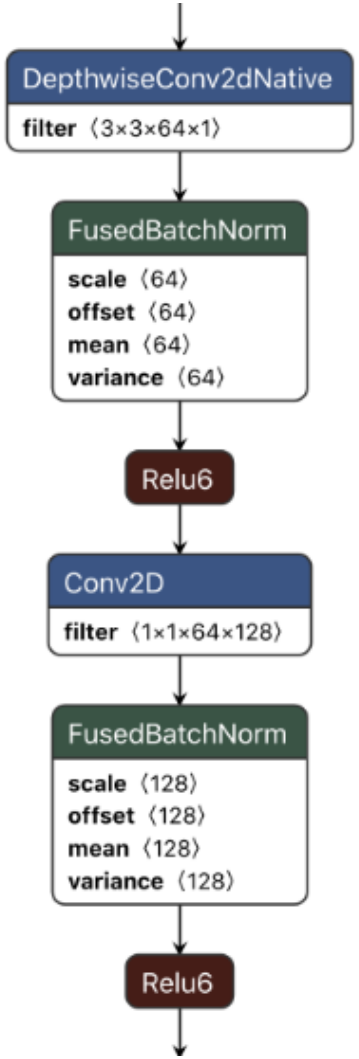
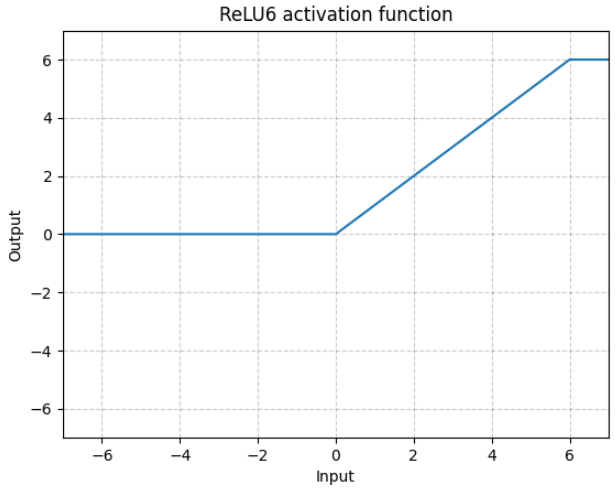


Table 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

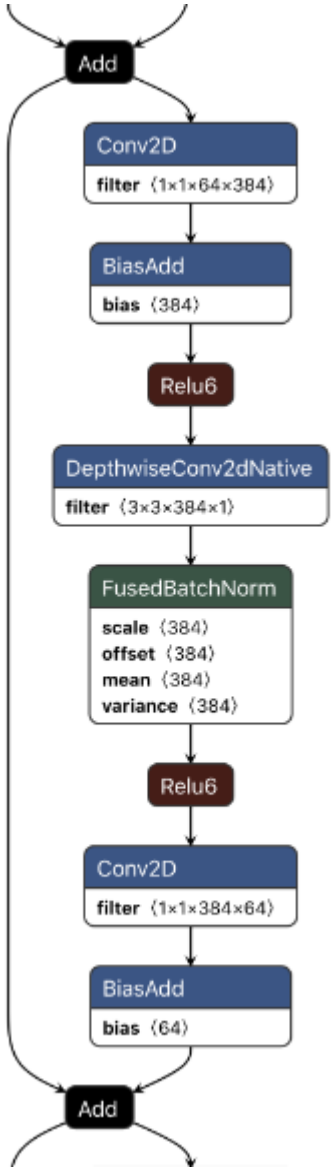
Table 9. Smaller MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.50 MobileNet-160	60.2%	76	1.32
Squeezenet	57.5%	1700	1.25
AlexNet	57.2%	720	60



MobileNetV2

MobileNetV2: Inverted Residuals and Linear Bottlenecks



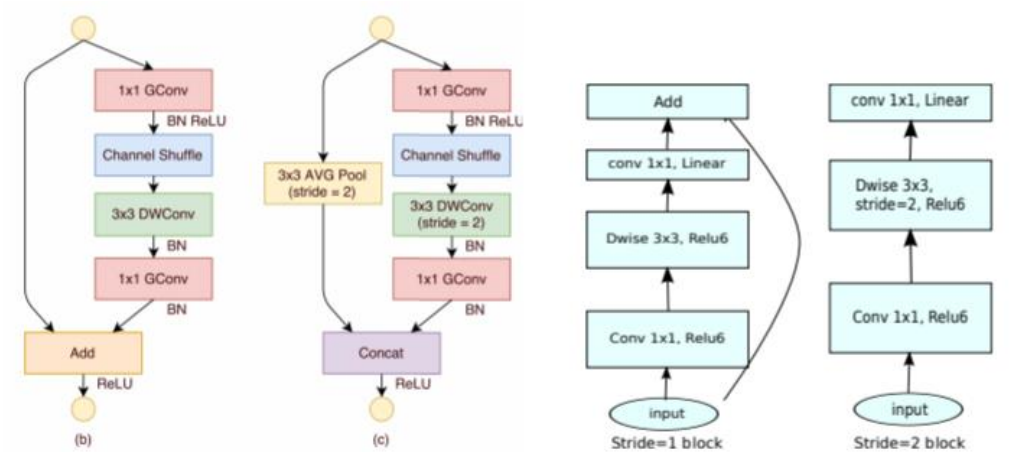
Expansion layer

Depthwise separable convolution V2

Depthwise layer

Bottleneck layer

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**



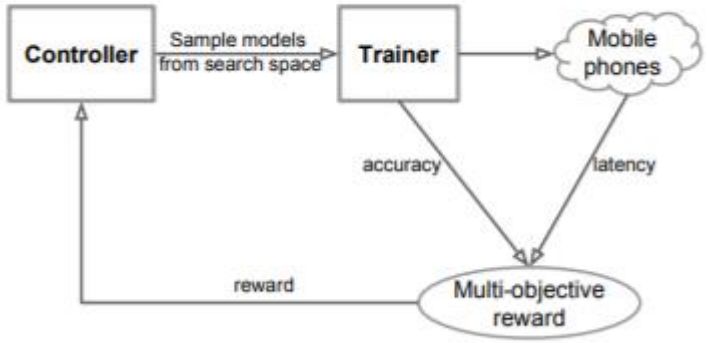
(c) ShuffleNet [20] (d) Mobilenet V2

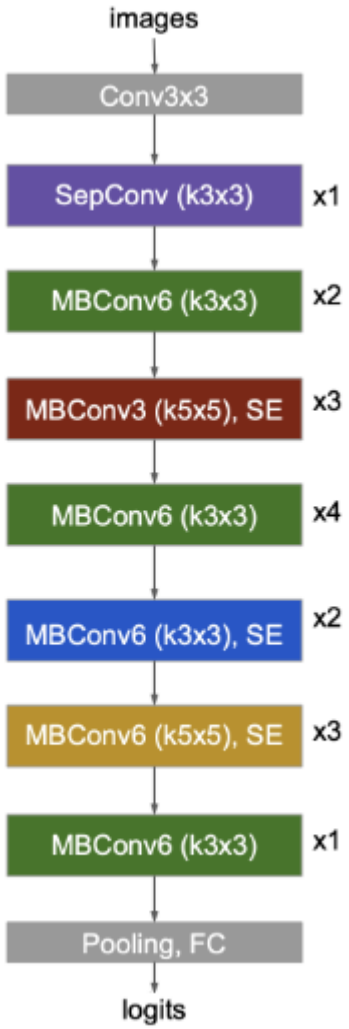
Mnasnet

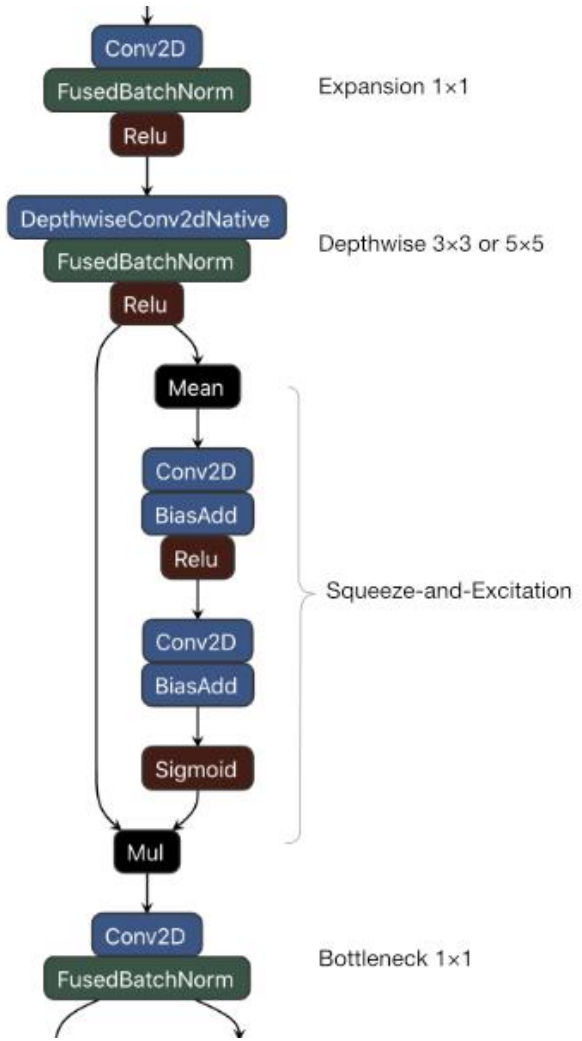
MnasNet: Platform-Aware Neural Architecture Search for Mobile

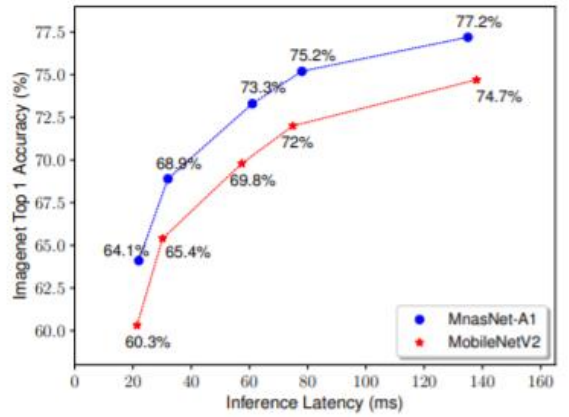
Mobile - neural architecture search

RNN









(a) Depth multiplier = 0.35, 0.5, 0.75, 1.0, 1.4, corresponding to points from left to right.

MobileNetV3

Searching for MobileNetV3

NetAdapt

H-swish = x * ReLU6(x + 3) / 6

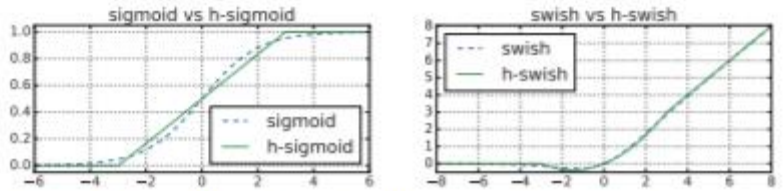
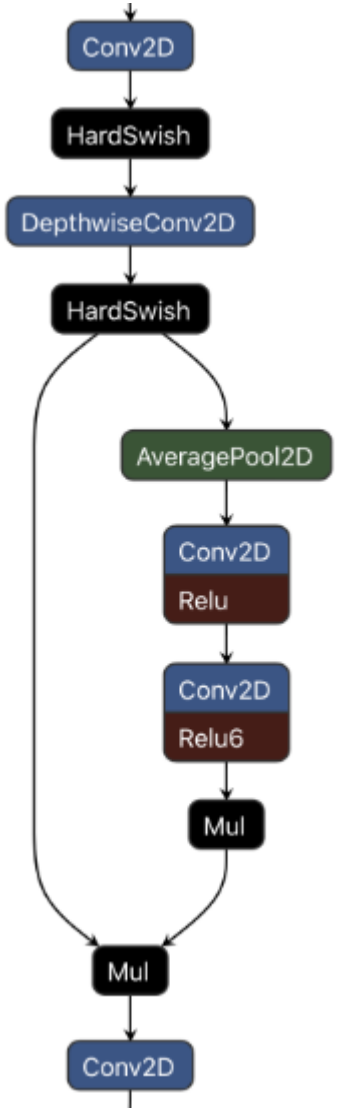
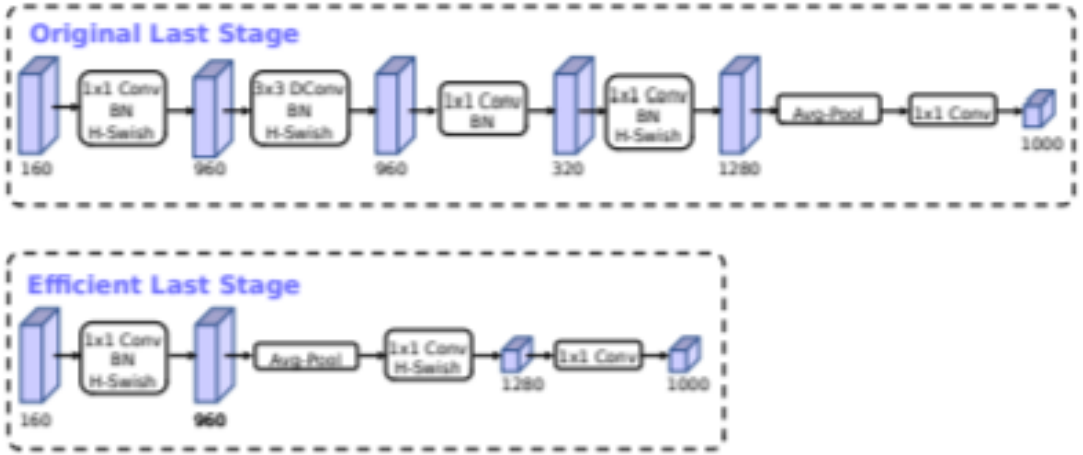
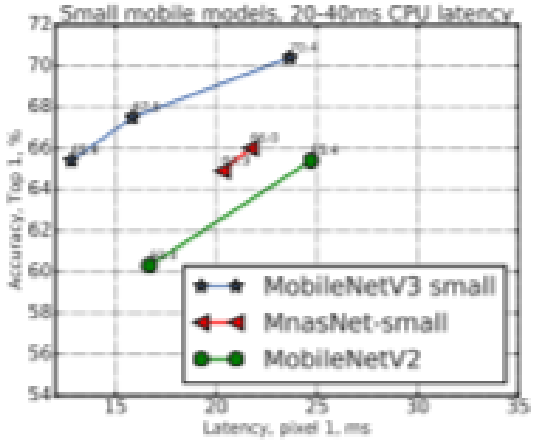


Figure 6. Sigmoid and swish nonlinearities and ther "hard" counterparts.



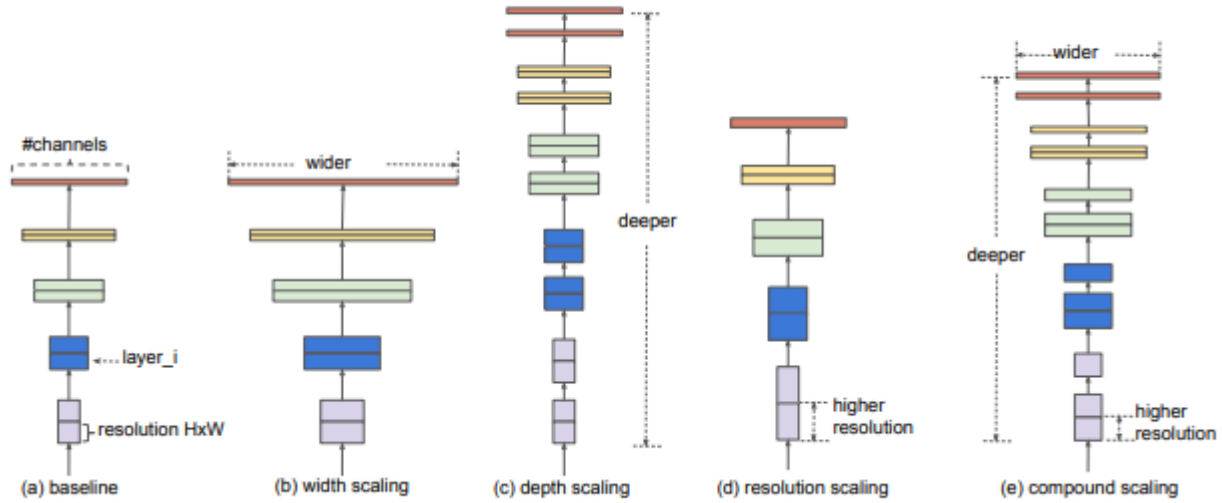




EfficientNets

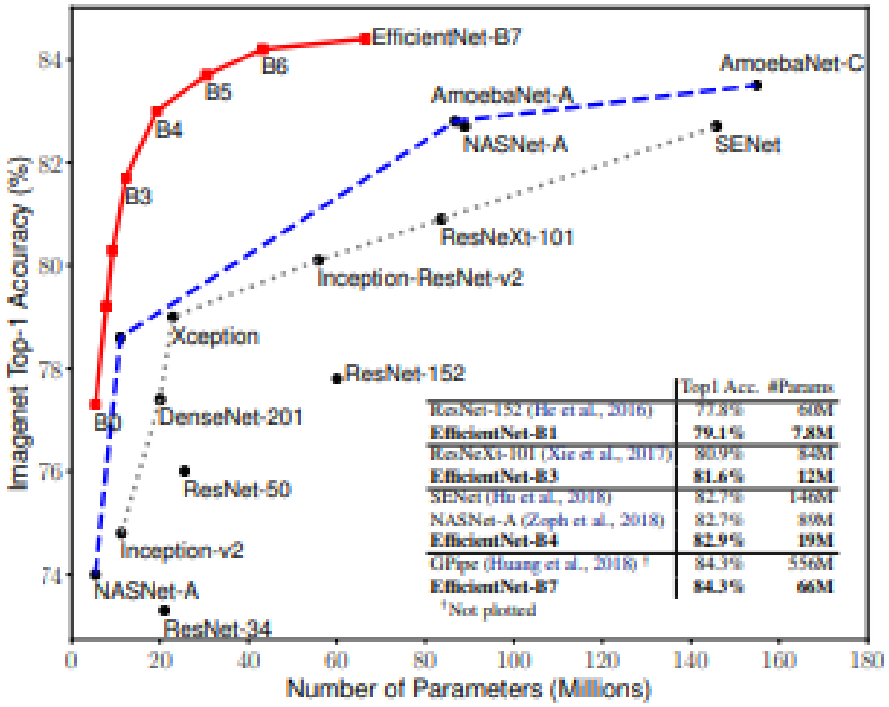
EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks



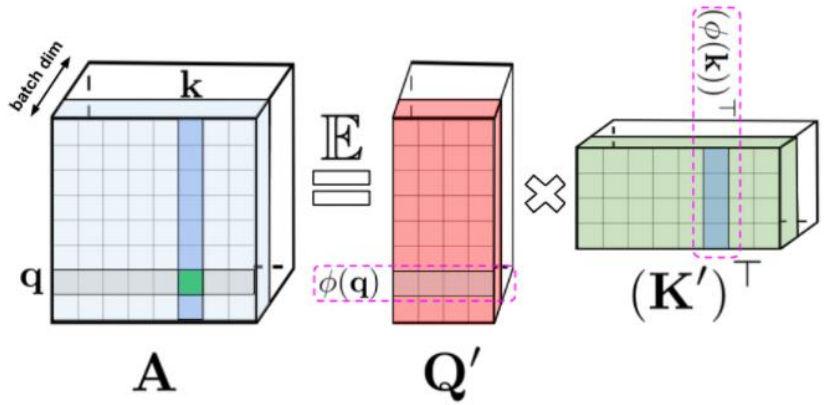
Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14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

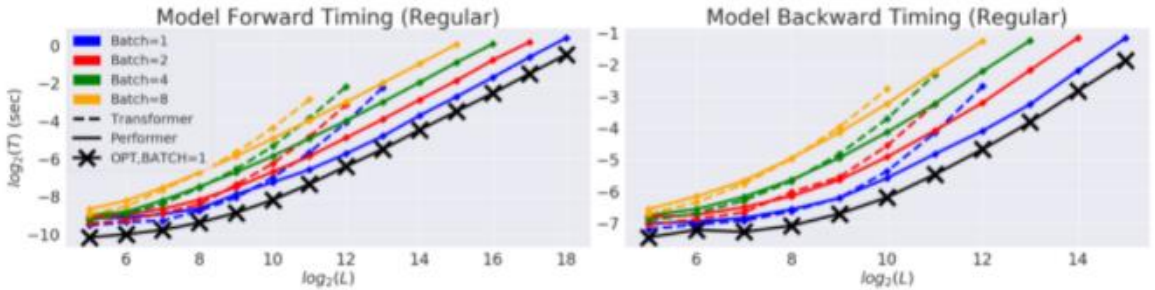
$$\text{depth: } d = \alpha^\phi$$
$$\text{width: } w = \beta^\phi$$
$$\text{resolution: } r = \gamma^\phi$$
$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$
$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

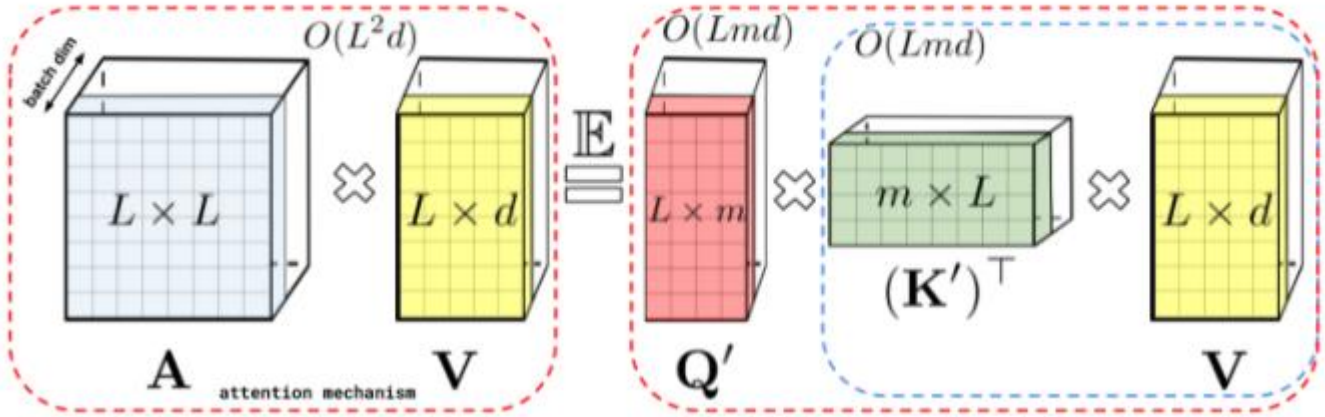


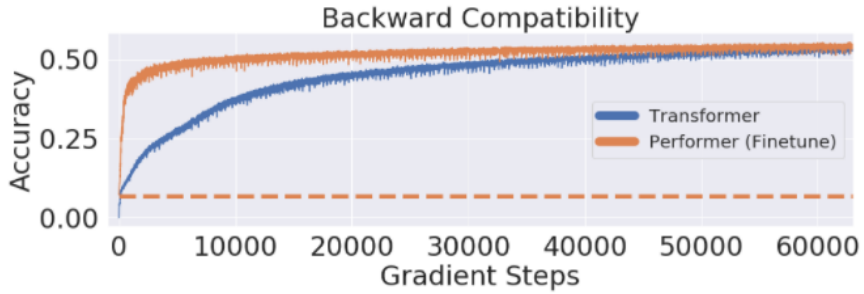
Performers

RETHINKING ATTENTION WITH PERFORMERS





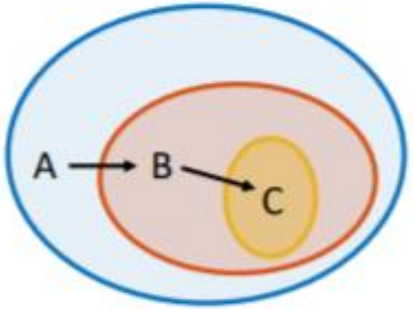


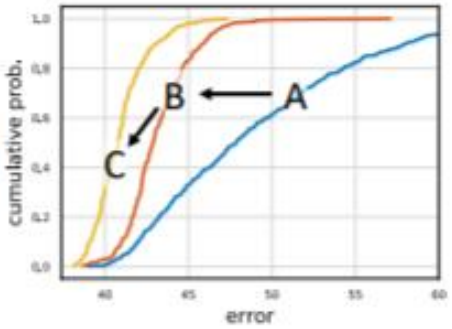


$$\mathrm{SM}(\mathbf{x},\mathbf{y}) = \mathbb{E}_{\omega\sim\mathcal{N}(0,\mathbf{I}_d)}\left[\exp\left(\omega^\top\mathbf{x}-\frac{\|\mathbf{x}\|^2}{2}\right)\exp\left(\omega^\top\mathbf{y}-\frac{\|\mathbf{y}\|^2}{2}\right)\right]$$

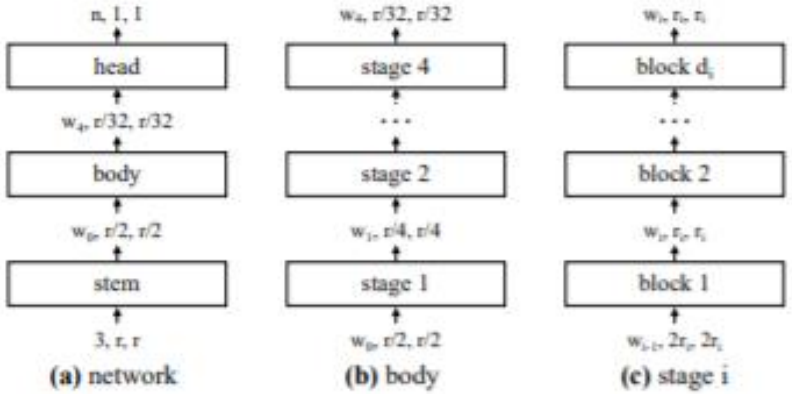
Regnets

Designing Network Design Spaces





	flops (B)	params (M)	acts (M)	batch size	infer (ms)	train (hr)	top-1 error ours [orig]
EFFICIENTNET-B0	0.4	5.3	6.7	256	34	11.7	**24.9** [23.7]
REGNETY-400MF	0.4	4.3	**3.9**	1024	19	5.1	25.9
EFFICIENTNET-B1	0.7	7.8	10.9	256	52	15.6	**24.1** [21.2]
REGNETY-600MF	0.6	6.1	**4.3**	1024	19	5.2	24.5
EFFICIENTNET-B2	1.0	9.2	13.8	256	68	18.4	**23.4** [20.2]
REGNETY-800MF	0.8	6.3	**5.2**	1024	22	6.0	23.7
EFFICIENTNET-B3	1.8	12.0	23.8	256	114	32.1	22.5 [18.9]
REGNETY-1.6GF	1.6	11.2	**8.0**	1024	39	10.1	**22.0**
EFFICIENTNET-B4	4.2	19.0	48.5	128	240	65.1	21.2 [17.4]
REGNETY-4.0GF	4.0	20.6	**12.3**	512	68	16.8	**20.6**
EFFICIENTNET-B5	9.9	30.0	98.9	64	504	135.1	21.5 [16.7]
REGNETY-8.0GF	8.0	39.2	**18.0**	512	113	28.1	**20.1**

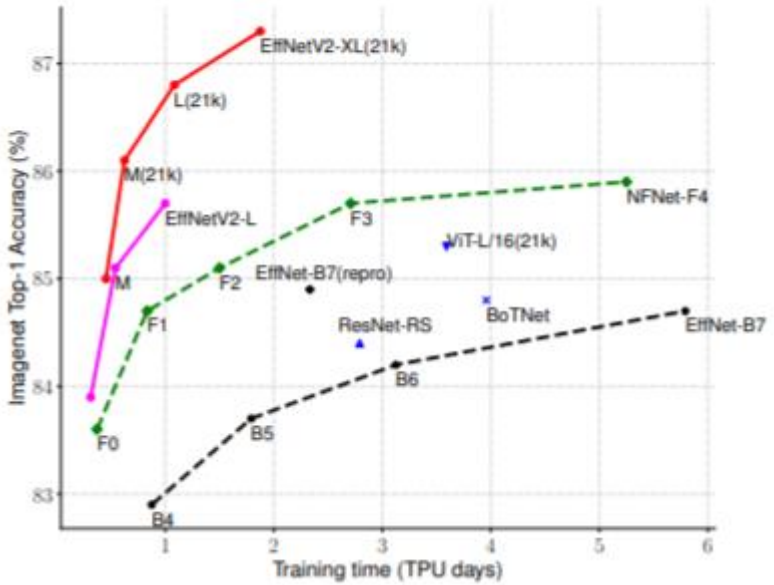


(a) network **(b)** body **(c)** stage i

(a) X block, s=1 **(b)** X block, s=2

It’s official, another season is coming!

EfficientNetV2: Smaller Models and Faster Training



WAIC