COMPUTER VISION-MOBILENET

Prof. Dr. Eng Fitri Utaminingrum., ST., MT

brone.ub.ac.id

MAIN IDEA MOBILE NET

- ➔ Arsitektur MobileNet diciptakan untuk diaplikasikan ke **perangkat mobile** dan **embedded vision**
- ➔ MobileNet menggunakan **depthwise separable convolutions** untuk membangun light weight deep neural networks
- ➔ Memiliki **ukuran model dan kompleksitas** yang lebih **kecil**
- ➔ Memperkenalkan 2 hyper-parameter yaitu **Width Multiplier** dan **Resolution Multiplier** yang memungkinkan model builder memilih model dengan ukuran yang tepat untuk diterapkan pada aplikasi

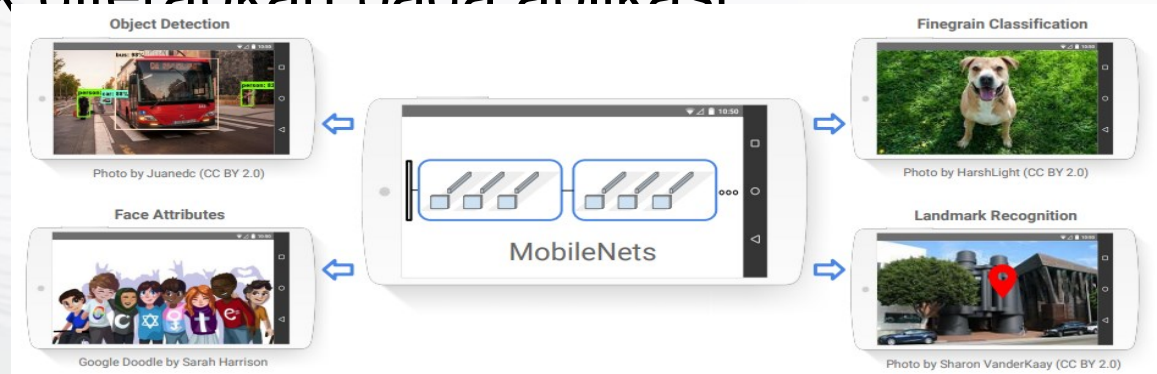
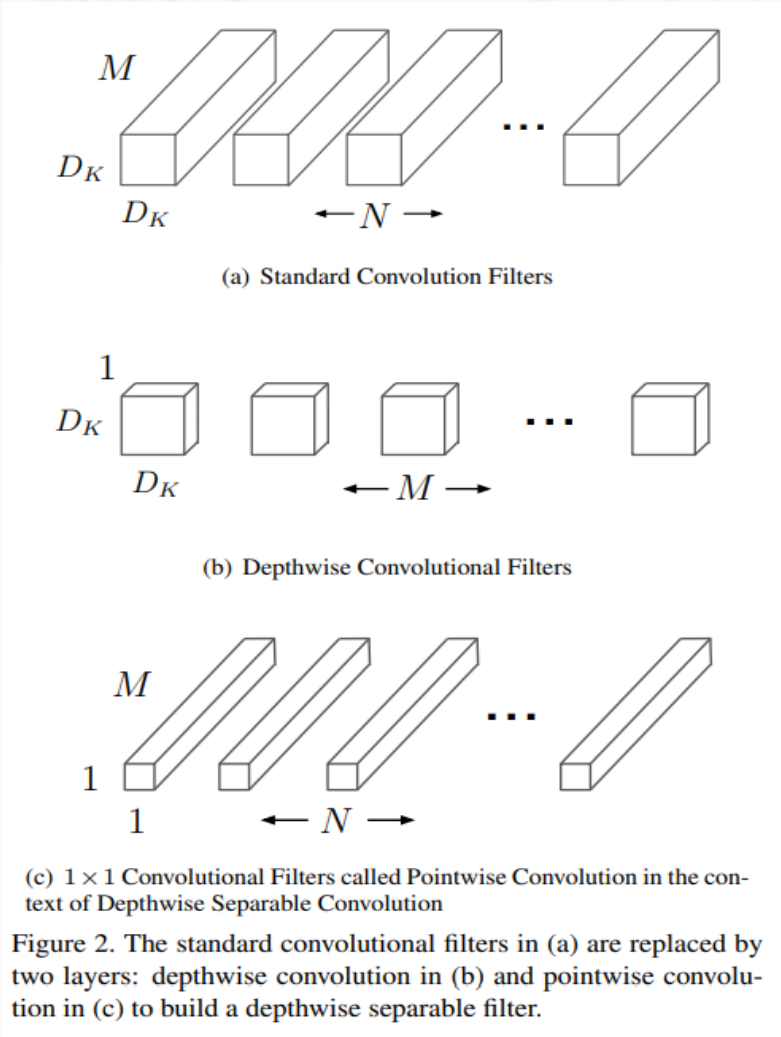


Figure 1. MobileNet models can be applied to various recognition tasks for efficient on device intelligence.

Apa itu Depthwise Separable Convolutions ?

Ide dasarnya adalah untuk mengganti full convolutional operator dengan factorized convolutions yang membagi konvolusi menjadi dua lapisan terpisah yaitu **depthwise convolution** dan **pointwise convolution**.



(a) Standard Convolution Filters

(b) Depthwise Convolutional Filters

(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Figure 2. The standard convolutional filters in (a) are replaced by two layers: depthwise convolution in (b) and pointwise convolution in (c) to build a depthwise separable filter.

Computational Cost

Standard Conv → $D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F$

Depthwise Separable Conv → $D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$

$$Depthwise\ Conv : Standard\ Conv = \frac{Depthwise\ Separable\ Conv}{Standard\ Conv}$$

$$\frac{Depthwise\ Conv}{Standard\ Conv} = \frac{D_K^2 \times M \times D_F^2 + M \times N \times D_F^2}{D_K^2 \times M \times N \times D_F^2}$$

$$\frac{Depthwise\ Conv}{Standard\ Conv} = \frac{M \times D_F^2 (D_K^2 + N)}{M \times D_F^2 \times D_K^2 \times N}$$

$$\frac{Depthwise\ Conv}{Standard\ Conv} = \frac{D_K^2 + N}{D_K^2 \times N}$$

$$\frac{Depthwise\ Conv}{Standard\ Conv} = \frac{1}{N} + \frac{1}{D_K^2}$$

Contoh apabila N = 512, dan DK = 3. Maka :

$$\frac{Depthwise\ Conv}{Standard\ Conv} = \frac{1}{512} + \frac{1}{3^2}$$

$$\frac{Depthwise\ Conv}{Standard\ Conv} = \frac{1042}{9216} = 0{,}113$$

Hyperparameter pada MobileNet

Menggunakan Hyper-parameter:

Width Multiplier (α)

Resolution Multiplier (ρ)

$$D_K \cdot D_K \cdot \alpha M \cdot D_F \cdot D_F + \alpha M \cdot \alpha N \cdot D_F \cdot D_F$$

$$D_K \cdot D_K \cdot \alpha M \cdot \rho D_F \cdot \rho D_F + \alpha M \cdot \alpha N \cdot \rho D_F \cdot \rho D_F$$

$$Standard\ Conv = M \times D_F^2 \times D_K^2 \times N$$
$$Standard\ Conv = 512 \times 14^2 \times 3^2 \times 512 = 462.422.016$$

$$Depthwise\ Conv = M \times D_F^2 (D_K^2 + N)$$
$$Depthwise\ Conv = 512 \times 14^2 (3^2 + 512) = 52.283.392$$

$$Depthwise\ Conv, \boldsymbol{\alpha} = \boldsymbol{\alpha} \times M \times D_F^2 (D_K^2 + \boldsymbol{\alpha} \times N)$$
$$Depthwise\ Conv, \boldsymbol{\alpha} = \mathbf{0,75} \times 512 \times 14^2 (3^2 + \mathbf{0,75} \times 512) = 29.578.752$$

$$\text{[illegible]}thwise\ Conv, \rho = \alpha \times M \times (\rho \times D_F)^2 (D_K^2 + \alpha \times N)$$
$$Depthwise\ Conv, \quad = 0{,}75 \times 512 \times (\mathbf{0,714} \times 14)^2 (3^2 + 0{,}75 \times 512) = 15.079.129{,}4546$$

Arsitektur Standard Conv VS MobileNet v1

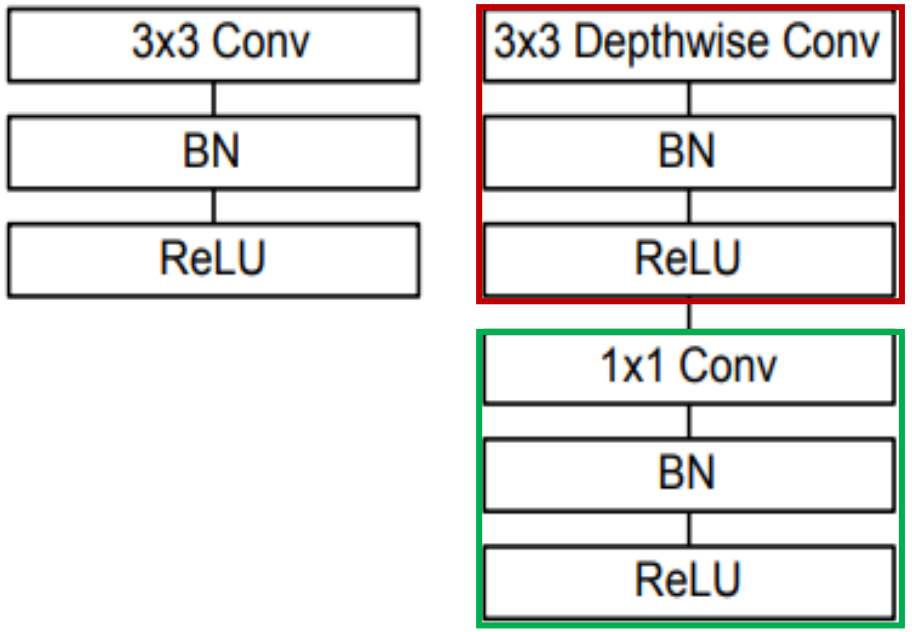


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

Depthwise Convolution
(spatial filtering)

Pointwise Convolution
(linear combination)

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

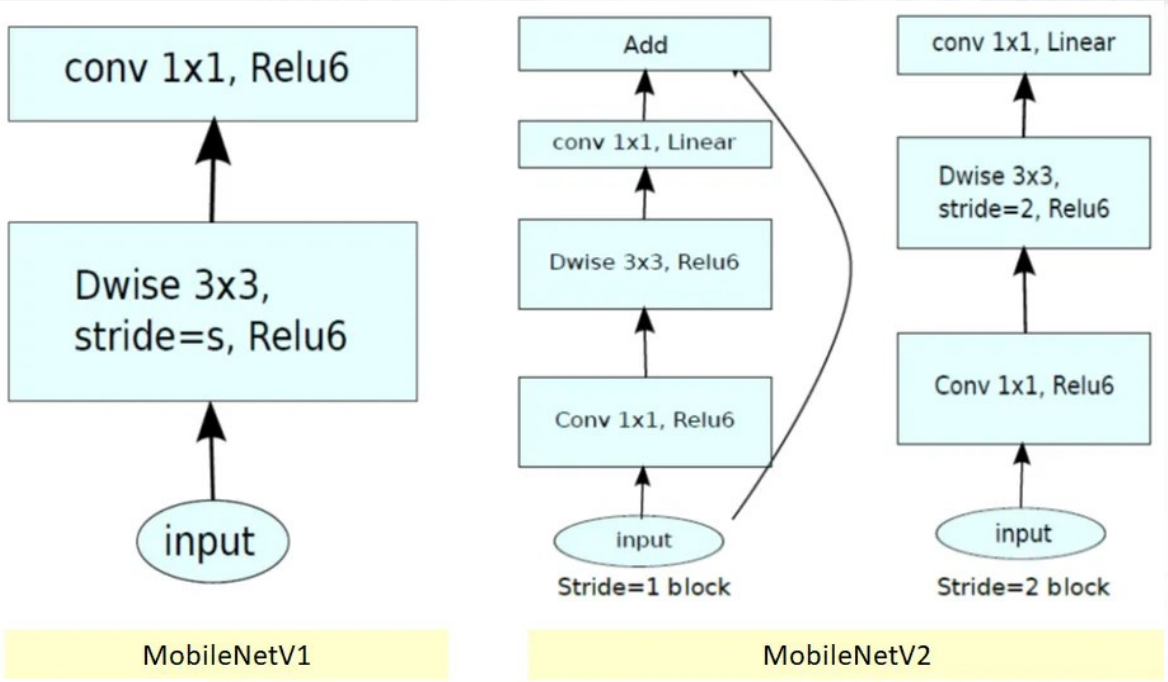
Table 4. Depthwise Separable vs Full Convolution MobileNet

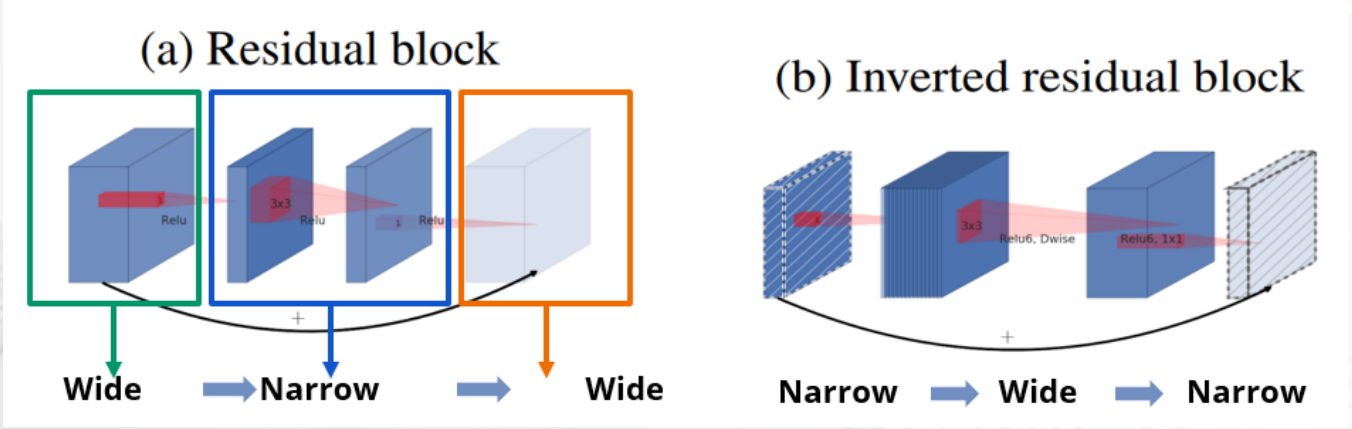
Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

MobileNet hanya mengalami penurunan akurasi sebesar 1%, tetapi Mult-Adds dan parameternya sangat berkurang.

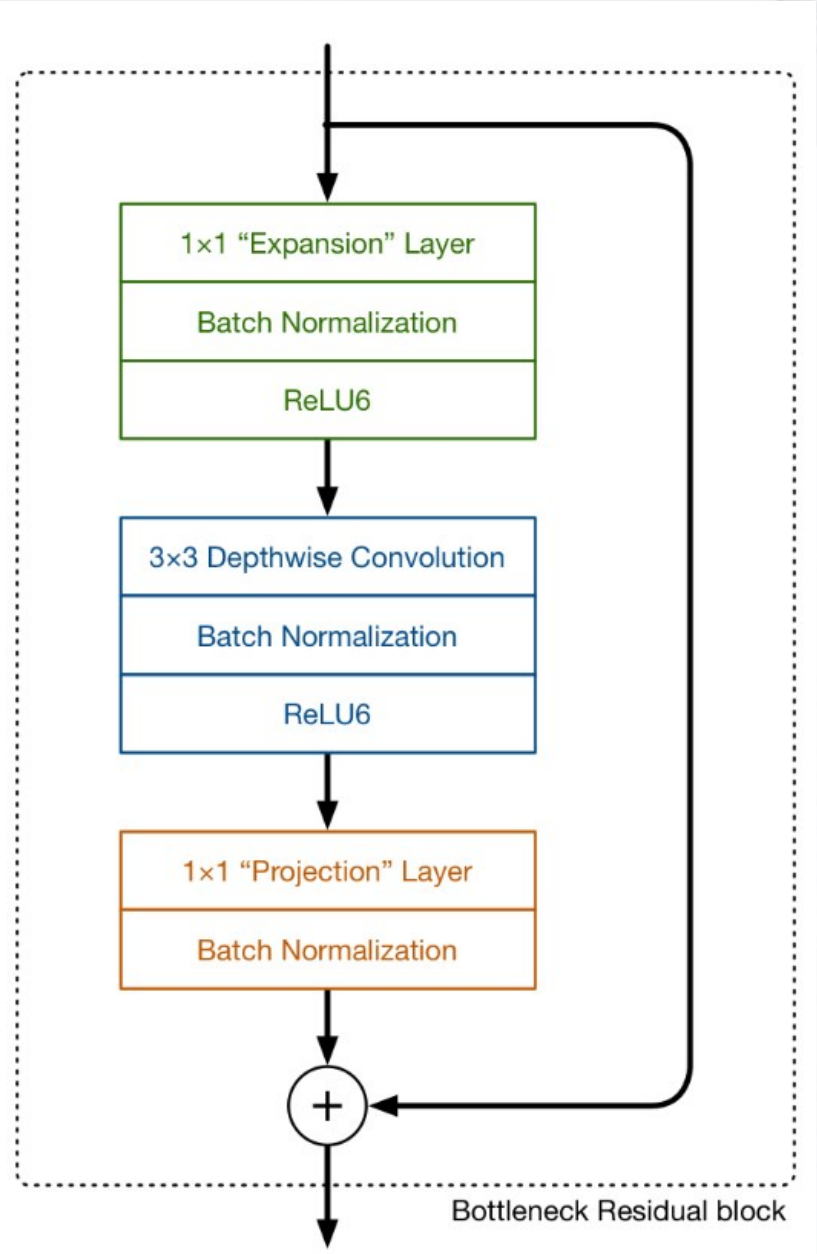
MobileNet v2, What's New?

MobileNet V2 mengembangkan pendahulunya dengan 2 ide utama yaitu, **Inverted Residuals** dan **Linear Bottlenecks.**





- **Expansion Layer**: Memperbanyak data (meningkatkan jumlah channel) yang melewatinya. Data diperluas berdasarkan faktor ekspansi (t). Default faktor ekspansi adalah 6 (t = 6).
- **ReLU6**: lebih robust daripada ReLU biasa ketika menggunakan komputasi presisi rendah. ReLU6 mencegah aktivasi menjadi terlalu besar dengan membatasi nilai aktivasi maks 6.
- **Projection Layer**: membuat jumlah channel lebih kecil, berkebalikkan dengan MobileNetV1 dimana pointwise layer menjaga jumlah channel tetap sama. Layer ini juga disebut dengan bottleneck layer.

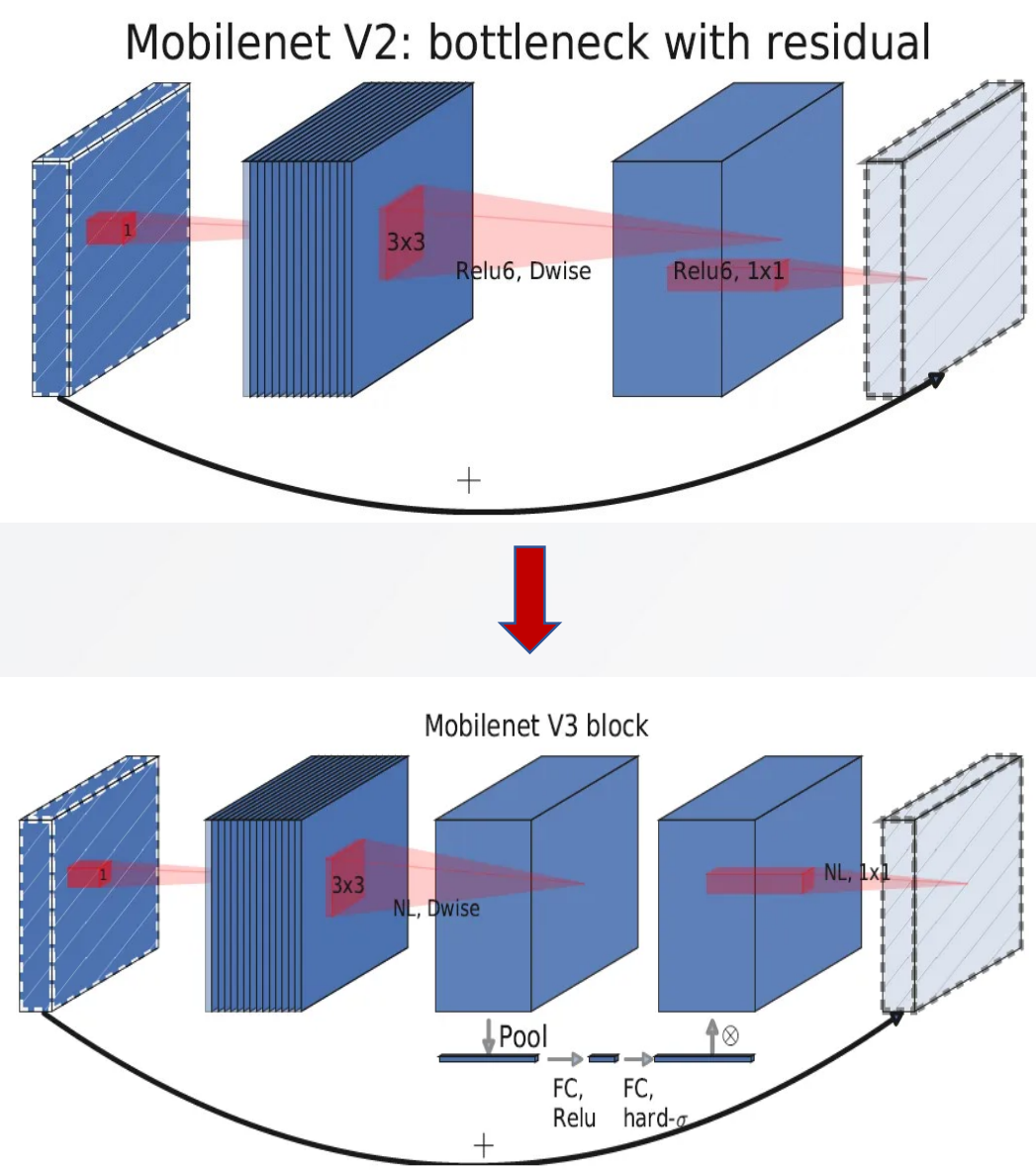


Next Generation! MobileNet v3

- Generasi berikutnya dari MobileNet yang didasarkan pada **combination of complementary search techniques** dan **novel architecture design**.

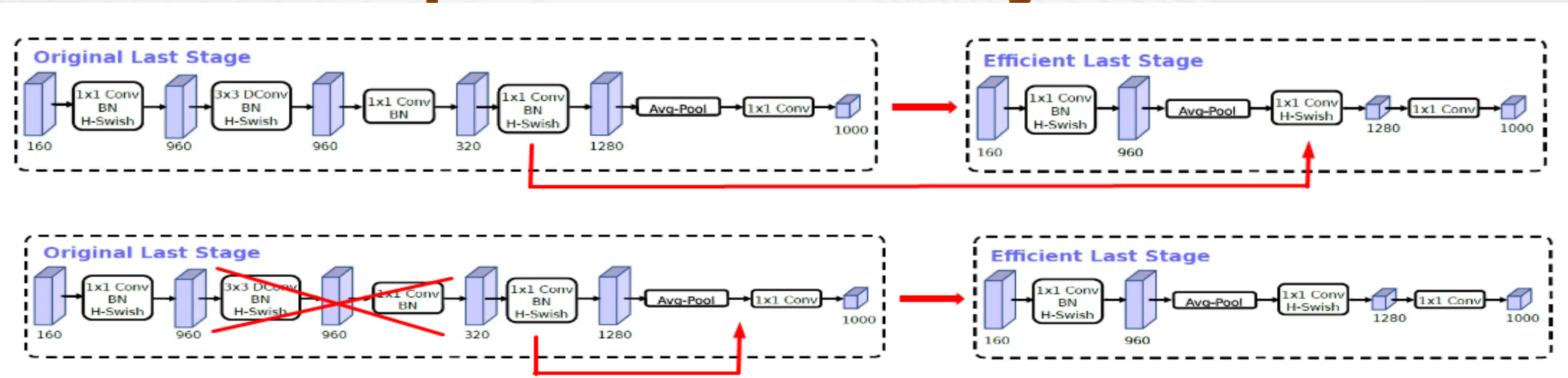
- MobileNetv3 menggunakan **hardware-aware network architecture search (NAS)** untuk mencari struktur jaringan global dengan mengoptimalkan setiap blok jaringan kemudian menggunakan algoritma **NetAdapt** untuk mencari jumlah filter per lapisan.

- 2 jenis arsitektur MobileNetv3: **MobileNetv3-Small** dan **MobileNetv3-Large** yang ditargetkan untuk kasus penggunaan low dan high resources.



- Pada MobileNetv3 terdapat penggabungan squeeze dan excitation block ke arsitektur inti.
- Bertujuan untuk meningkatkan kualitas representasi yang dihasilkan oleh jaringan secara eksplisit dengan memodelkan ketergantungan antar channel dari fitur konvolusi.
- Menghasilkan desain arsitektur yang lebih robust

Network Improvements - Layer Removal



- Pada jaringan awal, 1x1 final conv layer diperluas dari 320 menjadi 1280 fitur dimensi yang artinya ada biaya untuk tambahan latensi (cost of extra latency)
- Dengan memindahkan layer tersebut setelah Avg-Pool, final set dari fitur dikomputasi pada 1x1 spatial resolution bukan 7x7, ini membuatnya lebih efisien dalam hal komputasi dan latensi karena bottleneck projection layer sebelumnya tidak lagi diperlukan untuk mengurangi komputasi.
- Tahap terakhir ini mengurangi latensi sebesar 7 milidetik yang merupakan 11% dari running time dan mengurangi jumlah operasi sebesar 30 juta M-Adds tanpa kehilangan akurasi.

Network Improvements - Swish non-linearity

- swish non-linearity terbukti pada eksperimen dapat meningkatkan akurasi, akan tetapi fungsi sigmoid mahal secara komputasi.
- Fungsi swish ini kemudian di modifikasi menjadi hard-swish, tanpa mengurangi akurasi tetapi secara komputasi lebih murah
- Dengan menggunakan h-swish, inisial set filter yang awalnya 32 pada 3x3 full conv, diperkecil menjadi 16, hal ini

$$\text{swish}\, x = x \cdot \sigma(x)$$

$$\text{h-swish}[x] = x \frac{\text{ReLU6}(x+3)}{6}$$

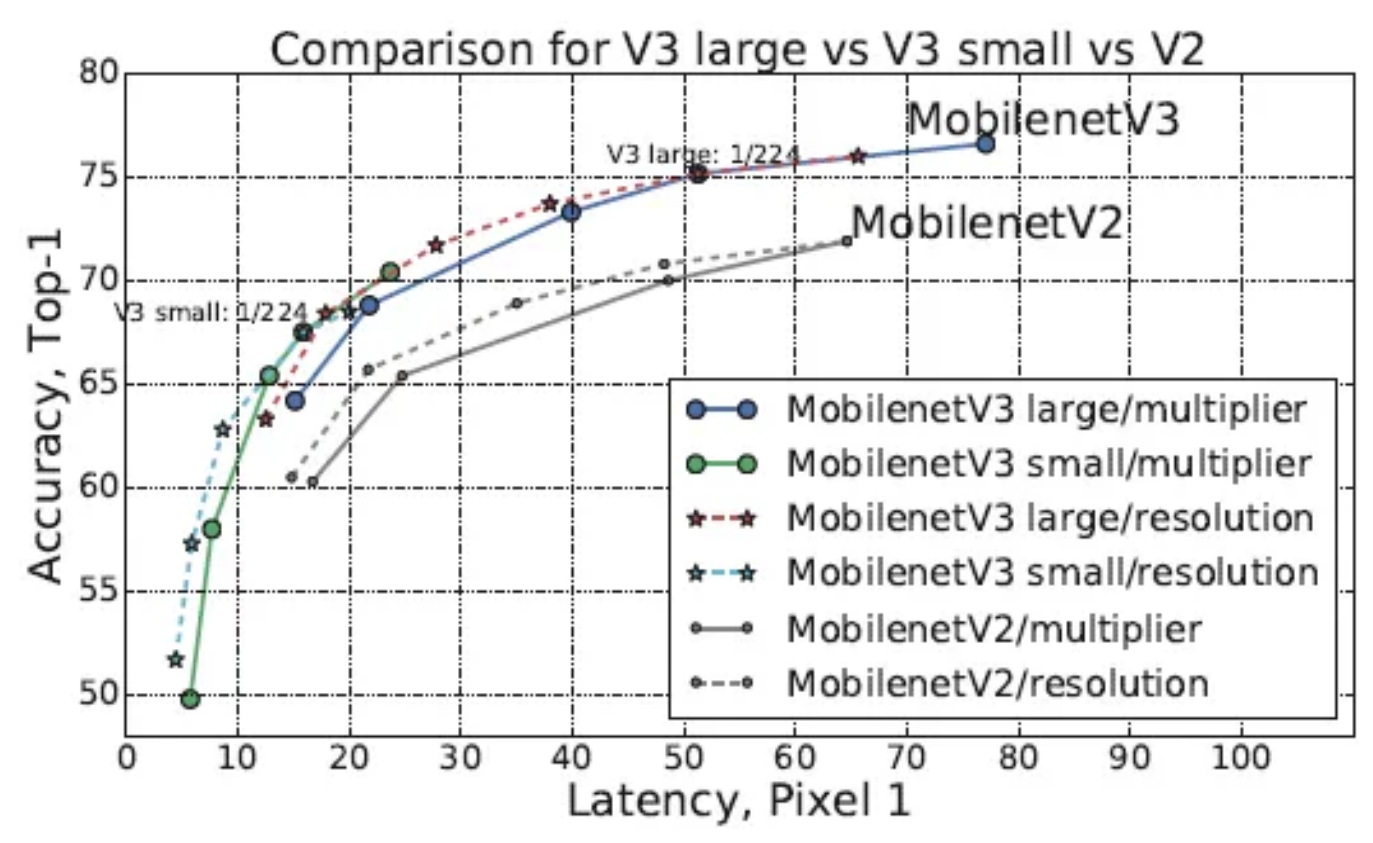
Arsitektur MobileNetv3 - Large

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	-	RE	1
$112^2 \times 16$	bneck, 3x3	64	24	-	RE	2
$56^2 \times 24$	bneck, 3x3	72	24	-	RE	1
$56^2 \times 24$	bneck, 5x5	72	40	✓	RE	2
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 3x3	240	80	-	HS	2
$14^2 \times 80$	bneck, 3x3	200	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	480	112	✓	HS	1
$14^2 \times 112$	bneck, 3x3	672	112	✓	HS	1
$14^2 \times 112$	bneck, 5x5	672	160	✓	HS	2
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	conv2d, 1x1	-	960	-	HS	1
$7^2 \times 960$	pool, 7x7	-	-	-	-	1
$1^2 \times 960$	conv2d 1x1, NBN	-	1280	-	HS	1
$1^2 \times 1280$	conv2d 1x1, NBN	-	k	-	-	1

Arsitektur MobileNetv3 - Small

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d, 3x3	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	✓	RE	2
$56^2 \times 16$	bneck, 3x3	72	24	-	RE	2
$28^2 \times 24$	bneck, 3x3	88	24	-	RE	1
$28^2 \times 24$	bneck, 5x5	96	40	✓	HS	2
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	240	40	✓	HS	1
$14^2 \times 40$	bneck, 5x5	120	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	144	48	✓	HS	1
$14^2 \times 48$	bneck, 5x5	288	96	✓	HS	2
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	bneck, 5x5	576	96	✓	HS	1
$7^2 \times 96$	conv2d, 1x1	-	576	✓	HS	1
$7^2 \times 576$	pool, 7x7	-	-	-	-	1
$1^2 \times 576$	conv2d 1x1, NBN	-	1280	-	HS	1
$1^2 \times 1280$	conv2d 1x1, NBN	-	k	-	-	1

SE denotes whether there is a Squeeze-And-Excite in that block. NL denotes the type of nonlinearity used. Here, HS denotes h-swish and RE denotes ReLU. NBN denotes no batch normalization. s denotes stride



Dibandingkan dengan MobileNetV2, MobileNetV3 masih lebih unggul dengan tradeoff yang lebih baik antara akurasi dan latensi.

Eksperimen MobileNet dengan Keras API

https://keras.io/api/applications/mobilenet/#mobilenet-function

MobileNet function

```
tf.keras.applications.MobileNet(
    input_shape=None,
    alpha=1.0,
    depth_multiplier=1,
    dropout=0.001,
    include_top=True,
    weights="imagenet",
    input_tensor=None,
    pooling=None,
    classes=1000,
    classifier_activation="softmax",
    **kwargs
)
```

MobileNet V1

```
base_model = MobileNet(weights='imagenet', include_top=False)
base_model.trainable = False
x=base_model.output
x=GlobalAveragePooling2D()(x)
x=Dense(320,activation='relu')(x)
x=Dense(320,activation='relu')(x)
preds=Dense(2,activation='softmax')(x)

modelmnetv1=Model(inputs=base_model.input, outputs=preds)
modelmnetv1.compile(optimizer=SGD(learning_rate =0.0001, momentum=0.9), loss='binary_crossentropy',metrics = ['accuracy'])
history = modelmnetv1.fit(x=train_batches, validation_data =  valid_batches, epochs = 20)
```