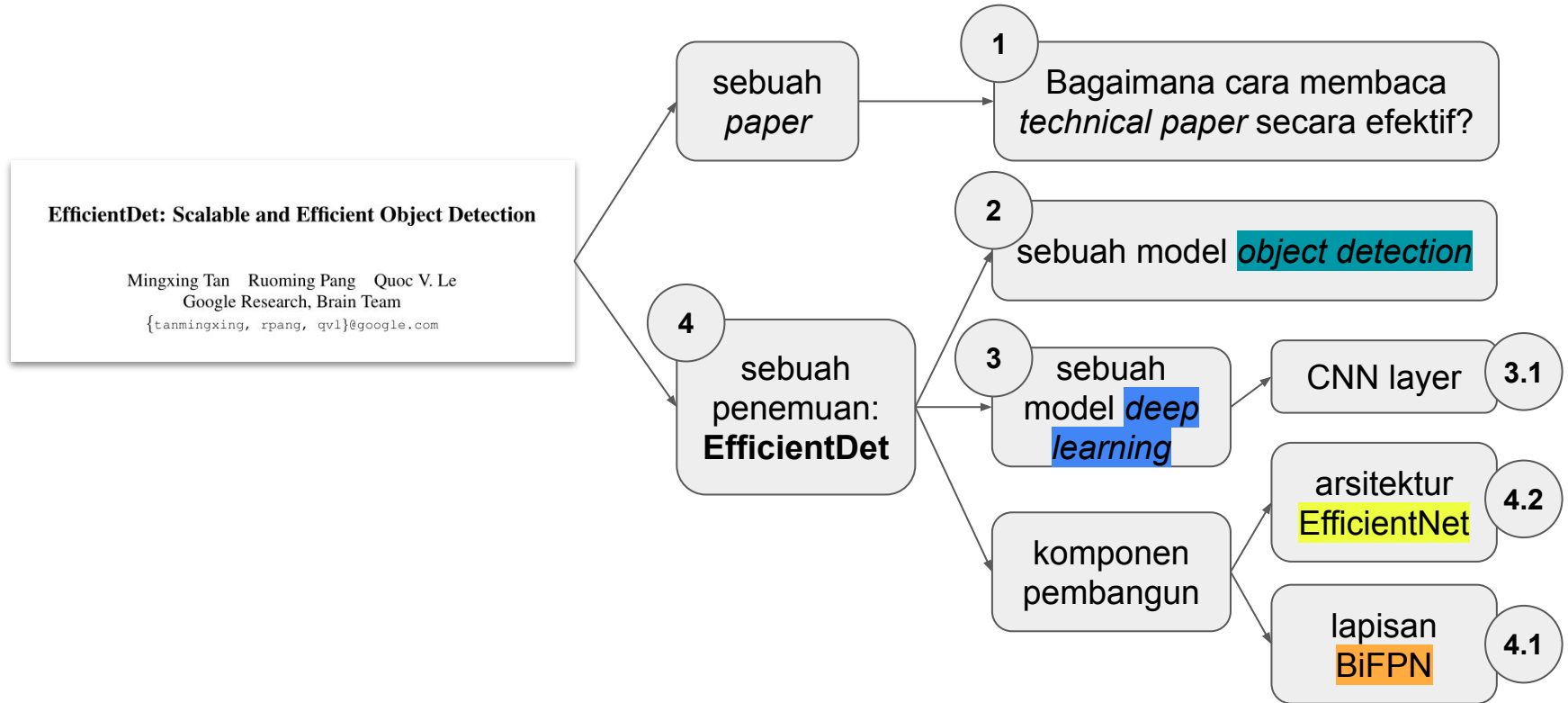


Pengenalan **EfficientDet**

Tan, M., Pang, R., & Le, Q. V. (2020). [Efficientdet: Scalable and efficient object detection](#).
In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10781-10790).

Ilma Aliya Fiddien
2 Juni 2022

Alur Penjelasan



1

Cara Membaca *Technical Paper*

EfficientDet: Scalable and Efficient Object Detection

Mingxing Tan Ruoming Pang Quoc V. Le
Google Research, Brain Team

{tanmingxing, rpang, qvl}@google.com

Alamat utama: <https://arxiv.org/abs/1911.09070v7>

Kode implementasi: <https://github.com/google/automl>

Blog: <https://ai.googleblog.com/2020/04/efficientdet-towards-scalable-and.html>

Langkah-Langkah

1. Cek identitas paper
2. Lihat sekilas judul bagian-bagian papernya
3. Di mana posisi paper ini di bidang yang ia kaji?
4. Baca bagian diskusi
5. Baca bagian abstrak
6. Lihat bagian metode dan hasil
7. Buat ringkasan

Langkah-Langkah

- 1. Cek identitas paper**
- 2. Lihat sekilas judul bagian-bagian papernya**
- 3. Di mana posisi paper ini di bidang yang ia kaji?**
4. Baca bagian abstrak/motivasi
5. Baca bagian kesimpulan, lalu diskusi
6. Lihat bagian metode dan hasil
7. Buat ringkasan

Cek Identitas Paper

- Tanggal publikasi (kebaruan)

Submission history

From: Mingxing Tan [\[view email\]](#)

[\[v1\]](#) Wed, 20 Nov 2019 18:16:09 UTC (1,440 KB)

[\[v2\]](#) Sun, 15 Mar 2020 22:55:42 UTC (1,124 KB)

[\[v3\]](#) Tue, 31 Mar 2020 18:47:08 UTC (1,086 KB)

[\[v4\]](#) Fri, 3 Apr 2020 20:34:21 UTC (1,086 KB)

[\[v5\]](#) Sun, 24 May 2020 07:12:44 UTC (1,086 KB)

[\[v6\]](#) Sun, 14 Jun 2020 18:28:53 UTC (1,086 KB)

[\[v7\]](#) Mon, 27 Jul 2020 15:55:16 UTC (1,318 KB)

- Penulis (kredensial)

Mingxing Tan Ruoming Pang Quoc V. Le

Google Research, Brain Team

{tanmingxing, rpang, qvl}@google.com

- Penerbitan (kualitas/gengsi)

Conference on Computer Vision and Pattern Recognition

From Wikipedia, the free encyclopedia

The **Conference on Computer Vision and Pattern Recognition (CVPR)** is an annual conference on computer vision and pattern recognition, which is regarded as one of the most important conferences in its field.^{[1][2][3]}

Conference on Computer Vision and Pattern Recognition	
Abbreviation	CVPR
Discipline	Computer vision
Publication details	
Publisher	IEEE
History	1985-present
Frequency	Annual

- Jumlah referensi (kedalaman/keluasan)
 - 51 references
- Jumlah sitasi (pengaruh)
 - 1972 citations ([google scholar](#))

Bagian-bagian paper

Abstract

1. Introduction
2. Related Work
3. BiFPN
 - 3.1. Problem Formulation
 - 3.2. Cross-Scale Connections
 - 3.3. Weighted Feature Fusion
4. EfficientDet
 - 4.1. EfficientDet Architecture
 - 4.2. Compound Scaling
5. Experiments
 - 5.1. EfficientDet for Object Detection
 - 5.2. EfficientDet for Semantic Segmentation

6. Ablation Study
 - 6.1. Disentangling Backbone and BiFPN
 - 6.2. BiFPN Cross-Scale Connections
 - 6.3. Softmax vs Fast Normalized Fusion
 - 6.4. Compound Scaling
7. Conclusion

Acknowledgements

References

Appendix

- 1.1 Hyperparameters
- 1.2 Image Resolution

Posisi paper ini di bidang yang ia kaji

EfficientDet: Towards Scalable and Efficient Object Detection (ai.googleblog.com)

As one of the core applications in computer vision, object detection has become increasingly important in scenarios that demand high accuracy, but have limited computational resources, such as [robotics](#) and [driverless cars](#). Unfortunately, many current [high-accuracy detectors](#) do not fit these constraints. More importantly, real-world applications of object detection are run on a variety of platforms, which often demand different resources. A natural question, then, is **how to design accurate and efficient object detectors that can also adapt to a wide range of resource constraints?**

In “[EfficientDet: Scalable and Efficient Object Detection](#)”, accepted at [CVPR 2020](#), **we introduce a new family of scalable and efficient object detectors**. Building upon our previous work on scaling neural networks ([EfficientNet](#)), and incorporating a novel bi-directional feature network (BiFPN) and new scaling rules, [EfficientDet](#) achieves state-of-the-art accuracy while being up to 9x smaller and using significantly less computation compared to prior state-of-the-art detectors. The following figure shows the overall network architecture of our models.

- Mencari efisiensi
- Membuat keluarga “object detector” baru
- Ada “upon previous work”
→ menambah bacaan
- Mencapai performa State-of-The-Art (SoTA)
→ terbaik pada saat itu

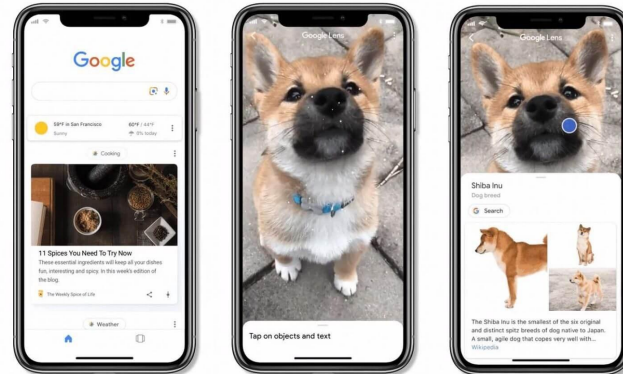
Motivasi

Abstract

Model efficiency has become increasingly important in computer vision. In this paper, we systematically study neural network architecture design choices for object detection and propose several key optimizations to improve efficiency. First, we propose a weighted bi-directional feature pyramid network (BiFPN), which allows easy and fast multi-scale feature fusion; Second, we propose a compound scaling method that uniformly scales the resolution, depth, and width for all backbone, feature network, and box/class prediction networks at the same time. Based on these optimizations and better backbones, we have developed a new family of object detectors, called EfficientDet, which consistently achieve much better efficiency than prior art across a wide spectrum of resource constraints. In particular, with single-model and single-scale, our EfficientDet-D7 achieves state-of-the-art 55.1 AP on COCO test-dev with 77M parameters and 410B FLOPs¹, being 4x – 9x smaller and using 13x – 42x fewer FLOPs than previous detectors. Code is available at <https://github.com/google/automl/tree/master/efficientdet>.

Efficiency:

- Model yang lebih kecil/ringan
- Processing yang lebih cepat
- Akurasi yang lebih tinggi



Solusi yang ditawarkan

Finally, we also observe that the recently introduced EfficientNets [39] achieve better efficiency than previous commonly used backbones. Combining EfficientNet backbones with our propose BiFPN and compound scaling, we have developed a new family of object detectors, named EfficientDet, which consistently achieve better accuracy with much fewer parameters and FLOPs than previous object detectors. Figure 1 and Figure 4 show the performance comparison on COCO dataset [25]. Under similar accuracy constraint, our EfficientDet uses 28x fewer FLOPs than YOLOv3 [34], 30x fewer FLOPs than RetinaNet [24], and 19x fewer FLOPs than the recent ResNet based NAS-FPN [10]. In particular, with single-model and single test-time scale, our EfficientDet-D7 achieves state-of-the-art 55.1 AP with 77M parameters and 410B FLOPs, outperforming previous best detector [45] by 4 AP while being 2.7x smaller and using 7.4x fewer FLOPs. Our EfficientDet is also up to 4x to 11x faster on GPU/CPU than previous detectors.

Key optimizations:

- Compound scaling method ([EfficientNet](#) sebagai *backbone*)
- Bi-directional feature pyramid network (BiFPN)

→ **EfficientDet**

Kesimpulan

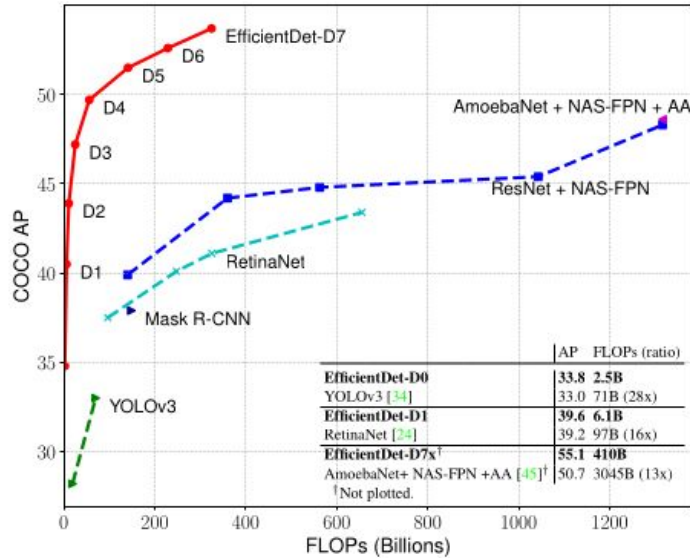
7. Conclusion

In this paper, we systematically study network architecture design choices for efficient object detection, and propose a weighted bidirectional feature network and a customized compound scaling method, in order to improve accuracy and efficiency. Based on these optimizations, we develop a new family of detectors, named *EfficientDet*, which consistently achieve better accuracy and efficiency than the prior art across a wide spectrum of resource constraints. In particular, our scaled EfficientDet achieves state-of-the-art accuracy with much fewer parameters and FLOPs than previous object detection and semantic segmentation models.

Diskusi

1. Membandingkan **EfficientNet** dengan model backbone lainnya
2. Membandingkan **BiFPN** dengan metode lainnya
3. Membandingkan Softmax vs **Fast Normalized Fusion**
4. Membandingkan **compound scaling** dengan single scaling resolution/depth/width

Hasil (1)



FLOPS= *floating point operations/sec*

AP= *average precision*

Metrik COCO data: IoU

IOU = Overlap/Union =

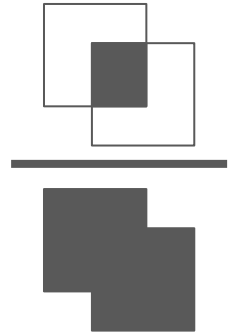


Figure 1: **Model FLOPs vs. COCO accuracy** – All numbers are for single-model single-scale. Our EfficientDet achieves new state-of-the-art 55.1% COCO AP with much fewer parameters and FLOPs than previous detectors. More studies on different backbones and FPN/NAS-FPN/BiFPN are in Table 4 and 5. Complete results are in Table 2.

Hasil (1)

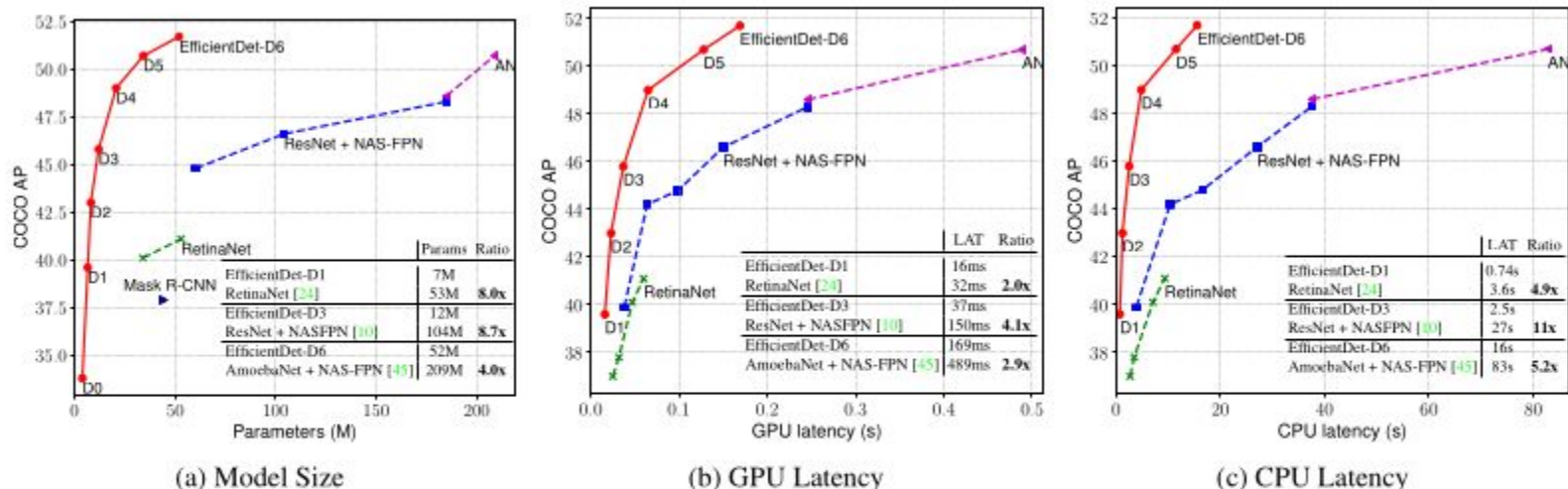


Figure 4: **Model size and inference latency comparison** – Latency is measured with batch size 1 on the same machine equipped with a Titan V GPU and Xeon CPU. AN denotes AmoebaNet + NAS-FPN trained with auto-augmentation [45]. Our EfficientDet models are 4x - 9x smaller, 2x - 4x faster on GPU, and 5x - 11x faster on CPU than other detectors.

Hasil (1 & 2)

Model	mIOU	Params	FLOPs
DeepLabV3+ (ResNet-101) [6]	79.35%	-	298B
DeepLabV3+ (Xception) [6]	80.02%	-	177B
Our EfficientDet[†]	81.74%	17M	18B

[†]A modified version of EfficientDet-D4.

Table 3: **Performance comparison on Pascal VOC semantic segmentation.**

	AP	Parameters	FLOPs
ResNet50 + FPN	37.0	34M	97B
EfficientNet-B3 + FPN	40.3	21M	75B
EfficientNet-B3 + BiFPN	44.4	12M	24B

Table 4: **Disentangling backbone and BiFPN** – Starting from the standard RetinaNet (ResNet50+FPN), we first replace the backbone with EfficientNet-B3, and then replace the baseline FPN with our proposed BiFPN.

Hasil (2)

	AP	#Params ratio	#FLOPs ratio
Repeated top-down FPN	42.29	1.0x	1.0x
Repeated FPN+PANet	44.08	1.0x	1.0x
NAS-FPN	43.16	0.71x	0.72x
Fully-Connected FPN	43.06	1.24x	1.21x
BiFPN (w/o weighted)	43.94	0.88x	0.67x
BiFPN (w/ weighted)	44.39	0.88x	0.68x

Table 5: **Comparison of different feature networks** – Our weighted BiFPN achieves the best accuracy with fewer parameters and FLOPs.

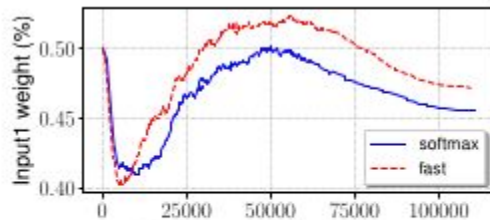
	AP	Parameters	FLOPs
ResNet50 + FPN	37.0	34M	97B
EfficientNet-B3 + FPN	40.3	21M	75B
EfficientNet-B3 + BiFPN	44.4	12M	24B

Table 4: **Disentangling backbone and BiFPN** – Starting from the standard RetinaNet (ResNet50+FPN), we first replace the backbone with EfficientNet-B3, and then replace the baseline FPN with our proposed BiFPN.

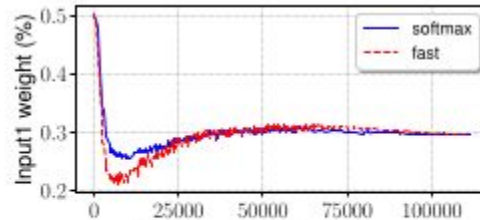
Hasil (3)

Model	Softmax Fusion AP	Fast Fusion AP (delta)	Speedup
Model1	33.96	33.85 (-0.11)	1.28x
Model2	43.78	43.77 (-0.01)	1.26x
Model3	48.79	48.74 (-0.05)	1.31x

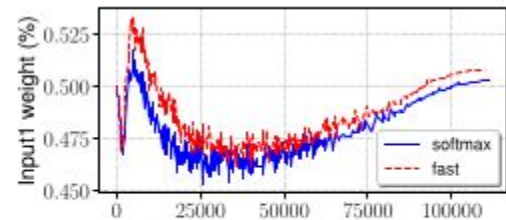
Table 6: **Comparison of different feature fusion** – Our fast fusion achieves similar accuracy as softmax-based fusion, but runs 28% - 31% faster.



(a) Example Node 1



(b) Example Node 2



(c) Example Node 3

Figure 5: **Softmax vs. fast normalized feature fusion** – (a) - (c) shows normalized weights (i.e., importance) during training for three representative nodes; each node has two inputs (input1 & input2) and their normalized weights always sum up to 1.

Hasil (4)

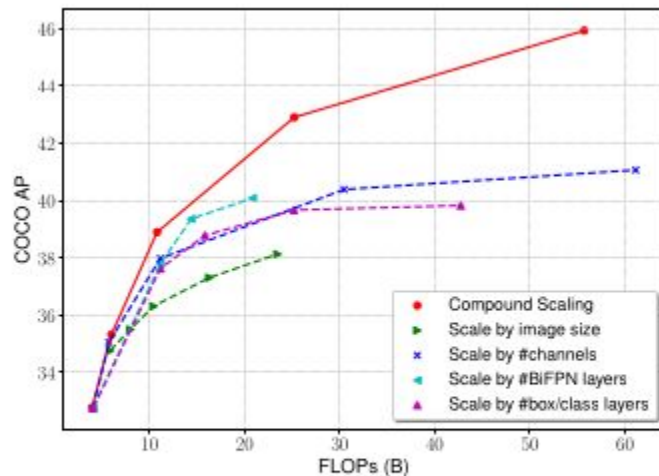
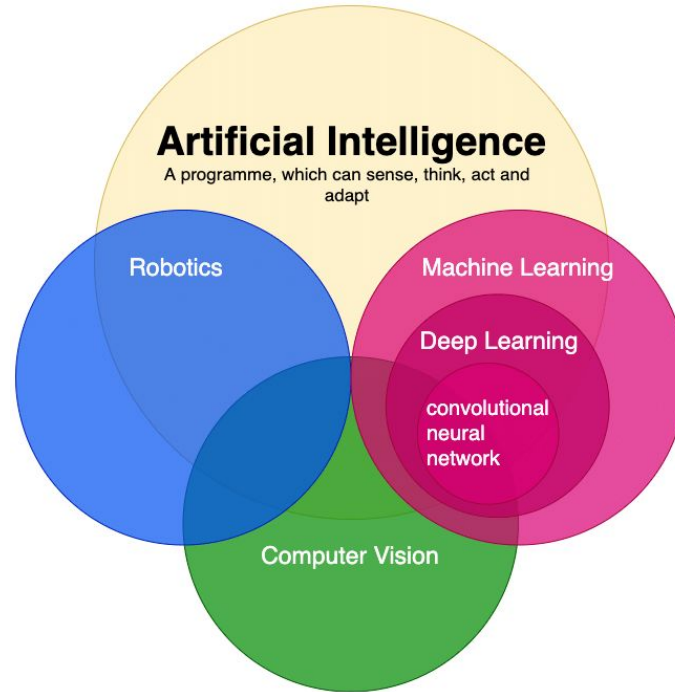


Figure 6: **Comparison of different scaling methods – compound scaling achieves better accuracy and efficiency.**

2

Object Detection

Artificial Intelligence, Computer Vision, Machine Learning



Motivasi Computer Vision

Use Cases

Automotive



Satellite & Drone Imagery



Gaming



Retail



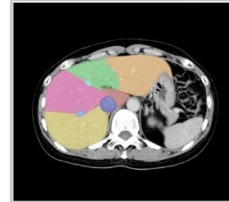
Agriculture



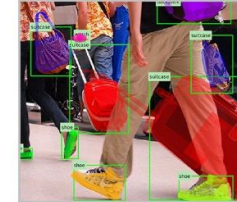
Media & Marketing



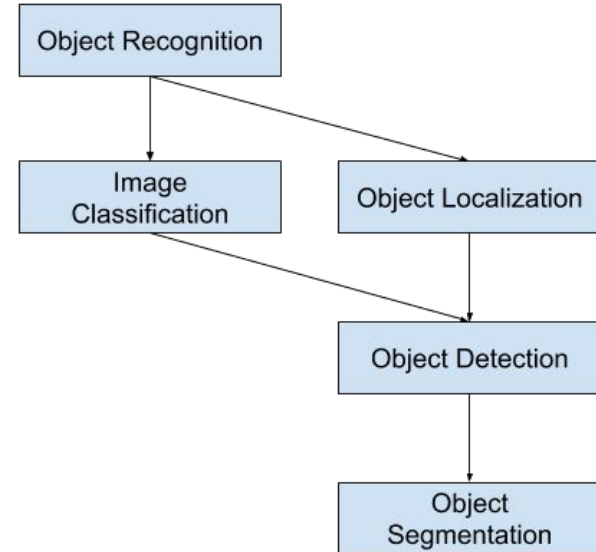
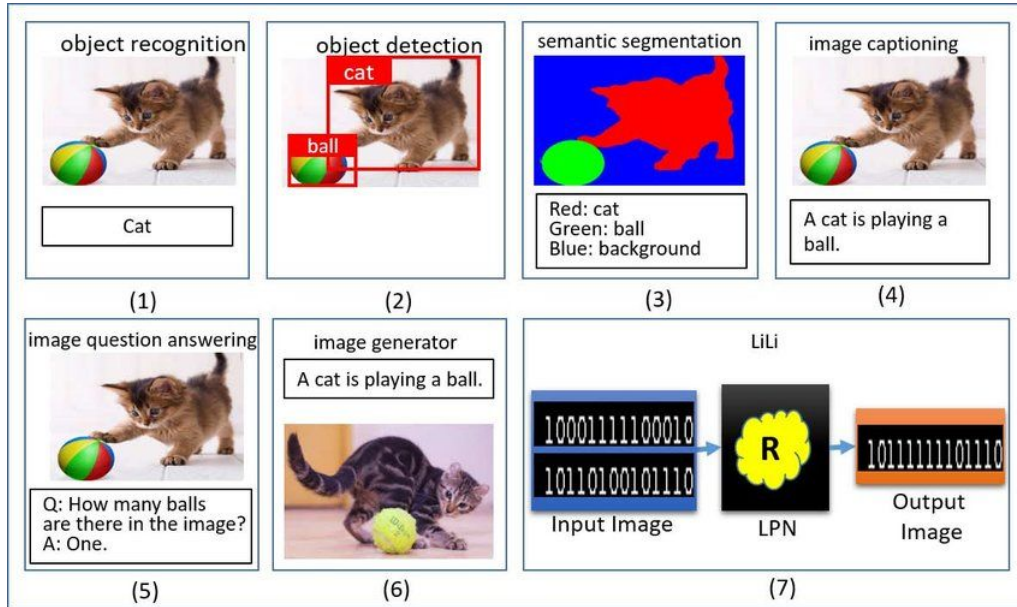
Medical



Fashion



Tugas-Tugas di Computer Vision



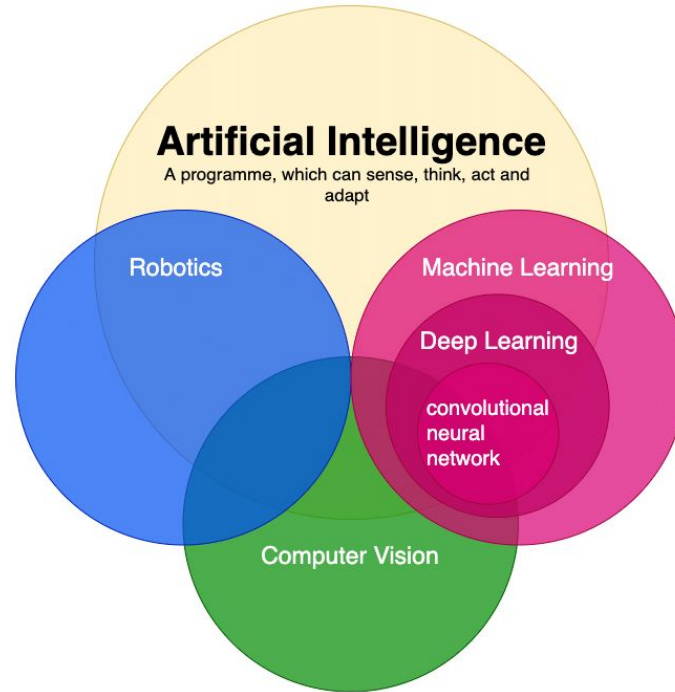
Guo, Qian & Qian, Yuhua & Liang, Xinyan & She, Yanhong & Li, Deyu & Liang, Jiye. (2019). [Logic could be learned from images.](#)

Brownlee, J. (2019, May 22). [A Gentle Introduction to Object Recognition With Deep Learning. Machine Learning Mastery.](#)

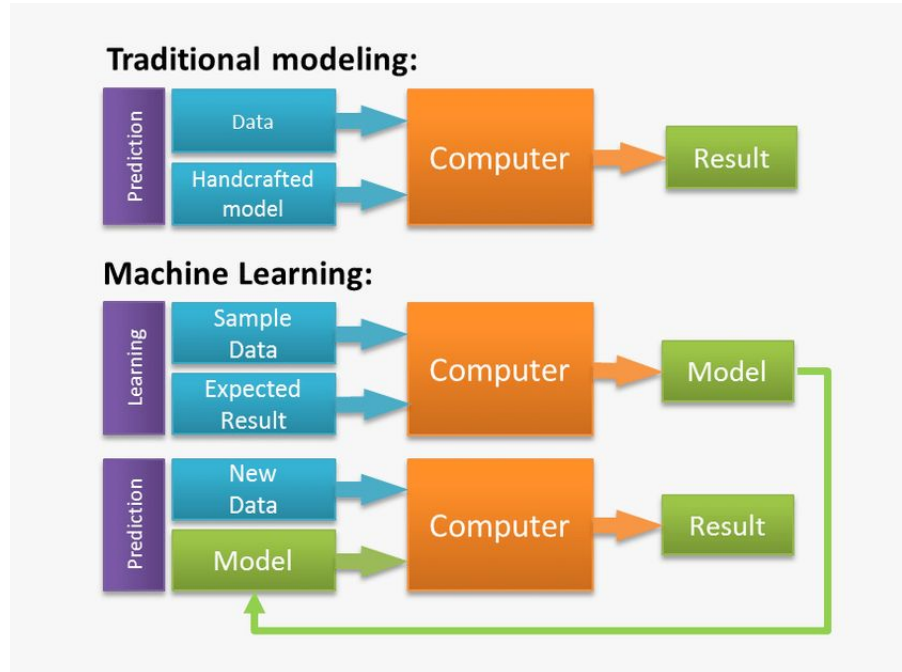
3

Model Machine Learning & Deep Learning

Artificial Intelligence, Computer Vision, Machine Learning



Artificial Intelligence, Computer Vision, Machine Learning



Learning Algorithm

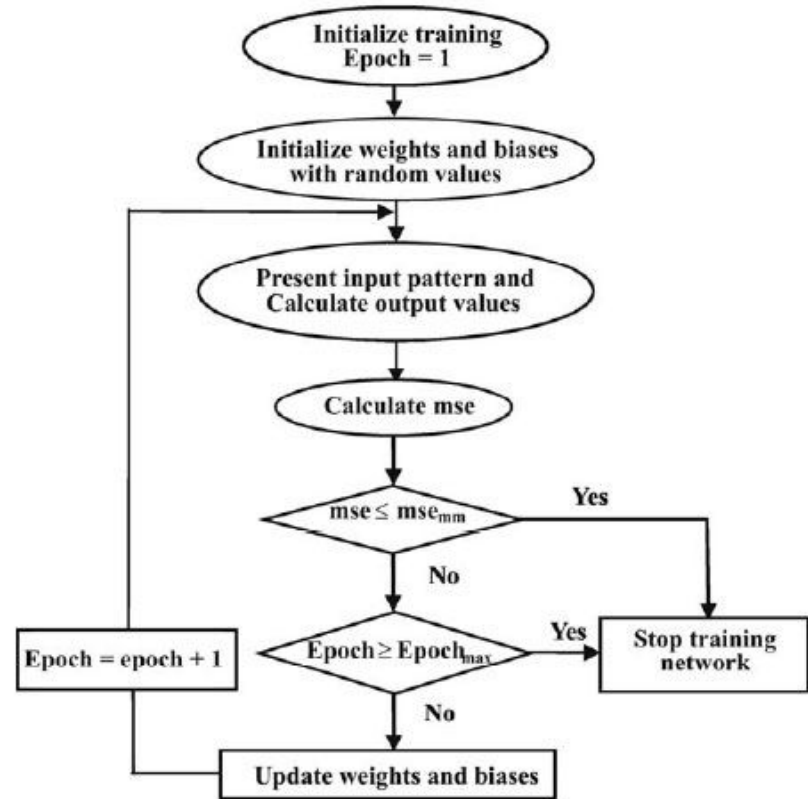
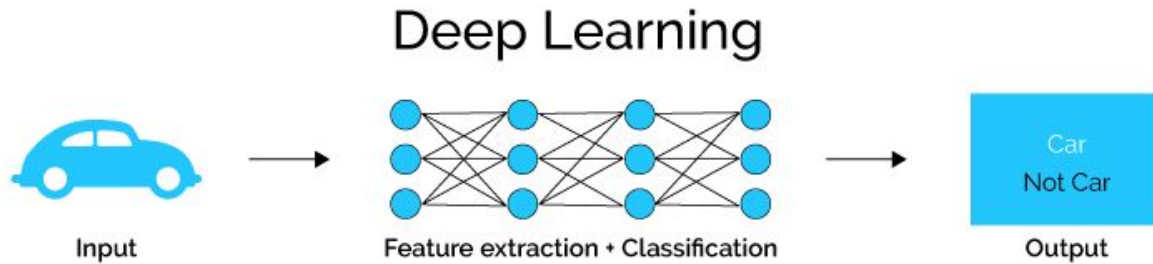
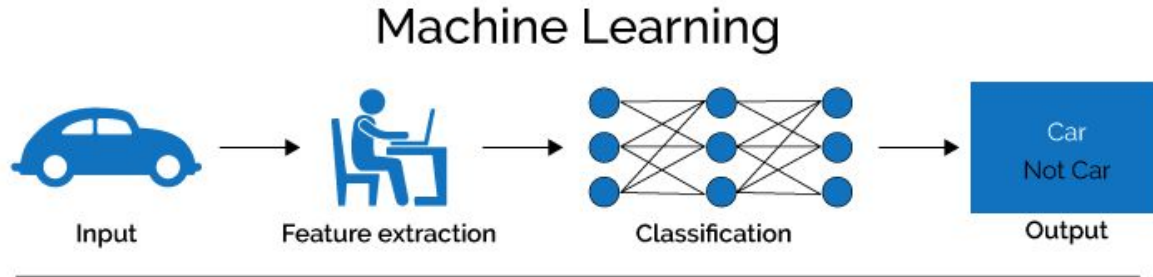
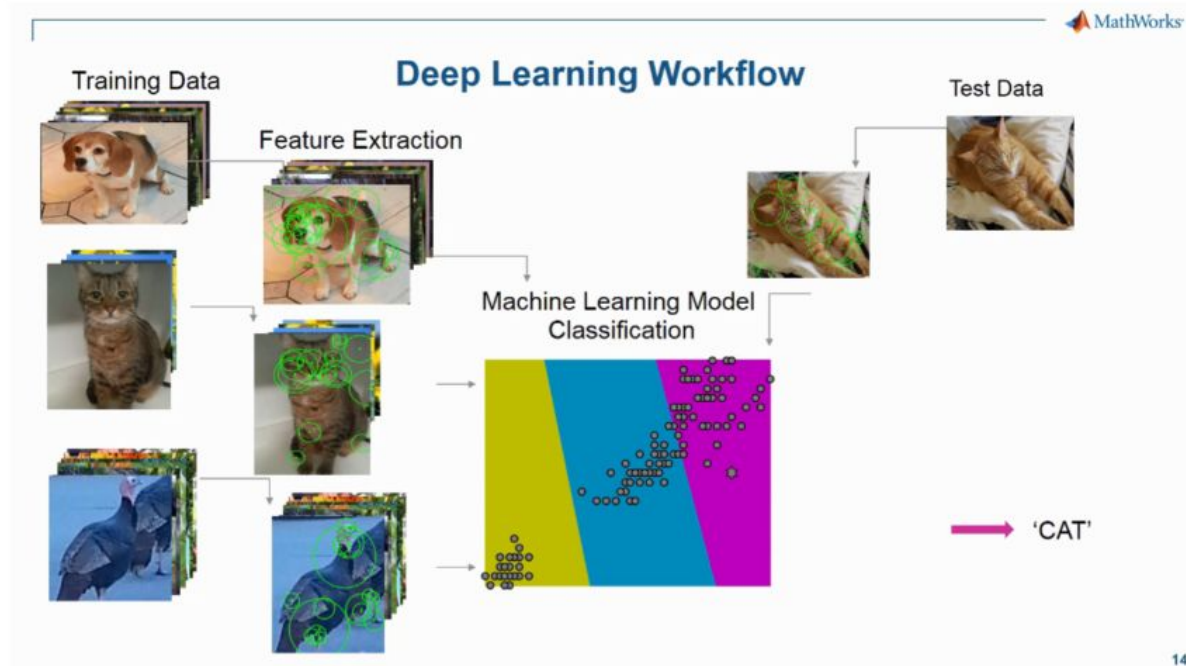


Figure 2: A training process flowchart

Machine Learning & Deep Learning

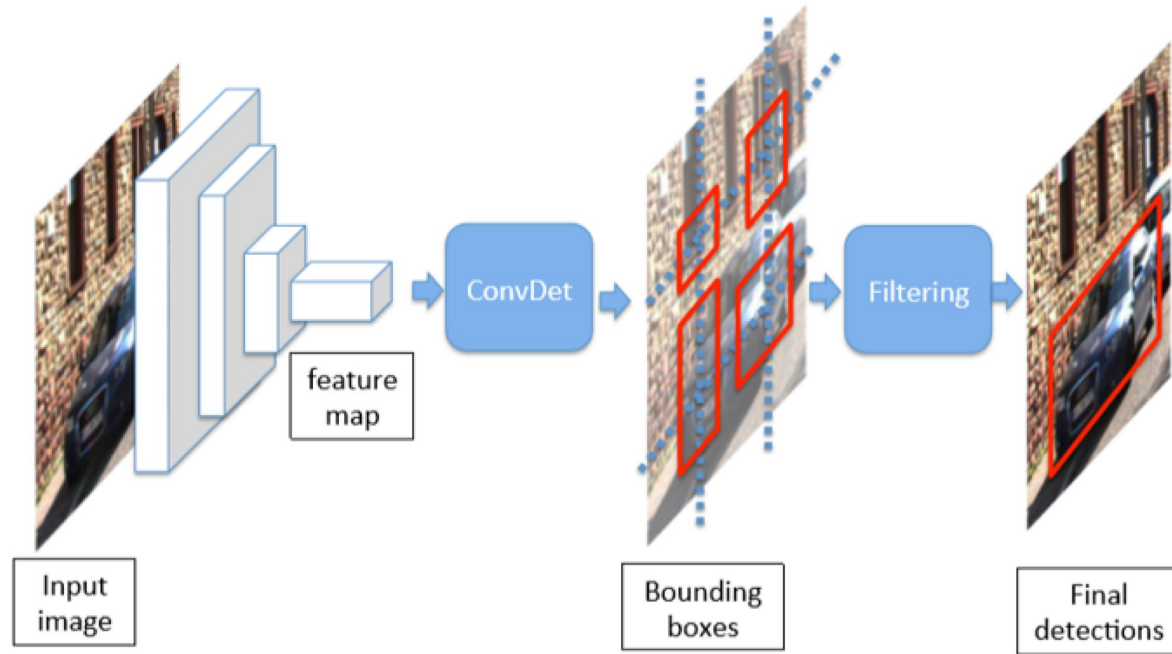


Machine Learning & Deep Learning

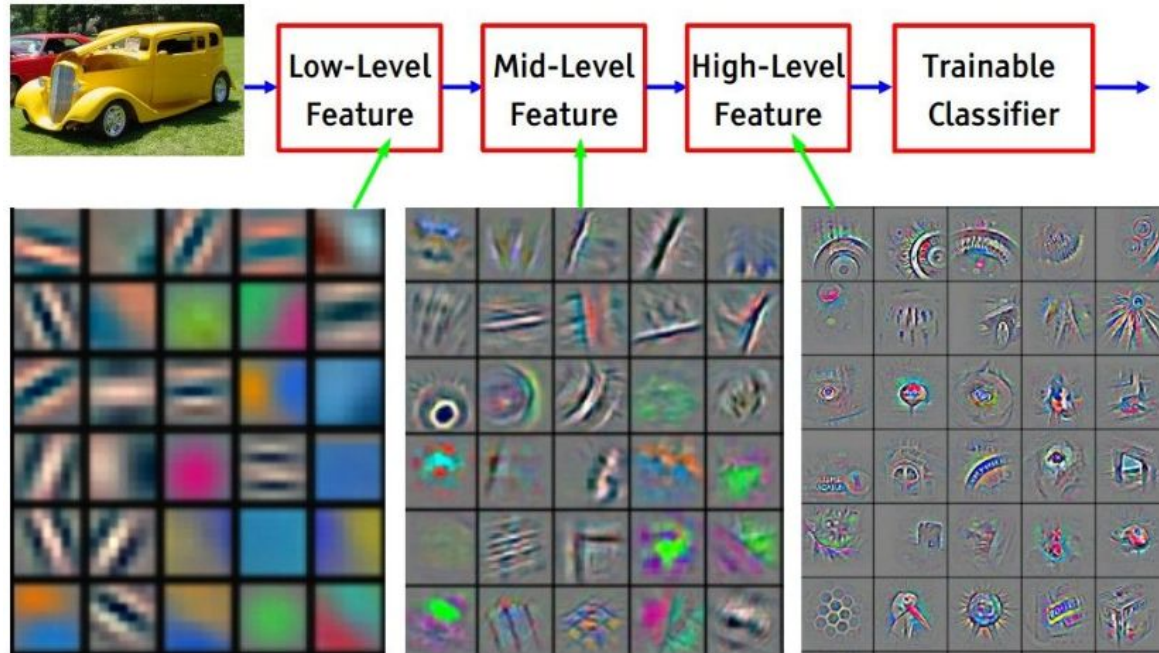


Johanna Pingel (2022). [Demos from "Object Recognition: Deep Learning" Webinar](#), MATLAB Central File Exchange. Retrieved July 1, 2022.

Object Detection model: regresi *bounding boxes* + klasifikasi objek

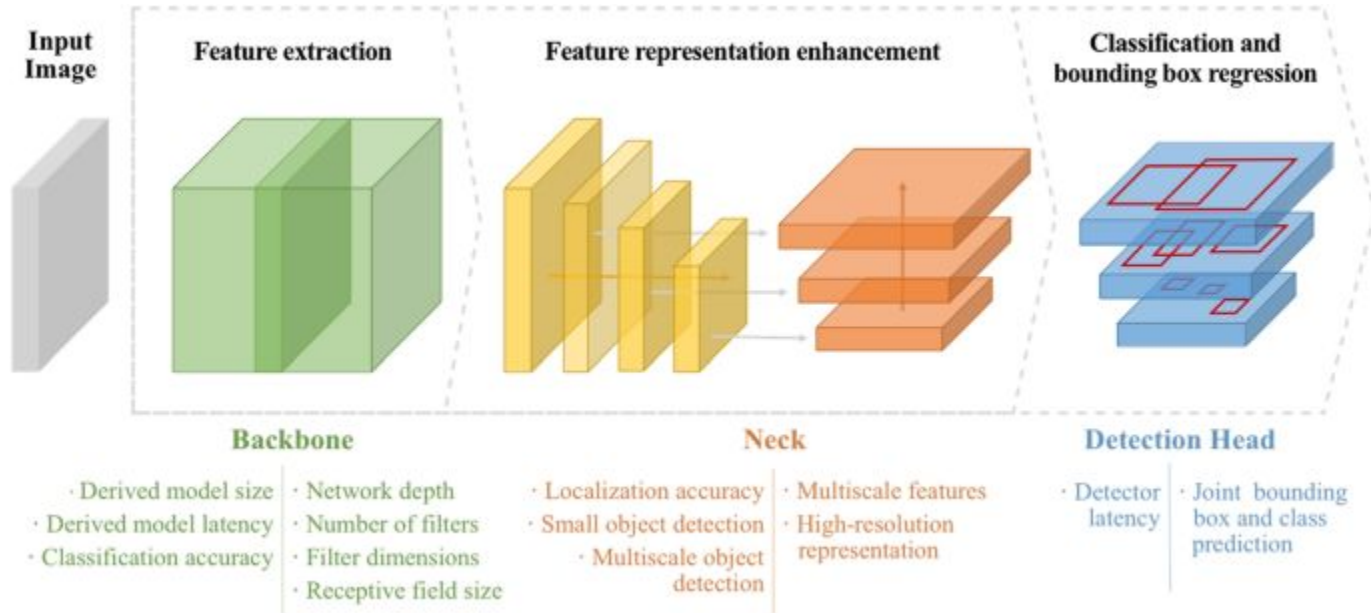


Features Map



<https://www.youtube.com/watch?v=cWzi38-vDbE>






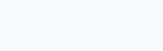



Struktur dari Model Object Detection (Deep Learning)



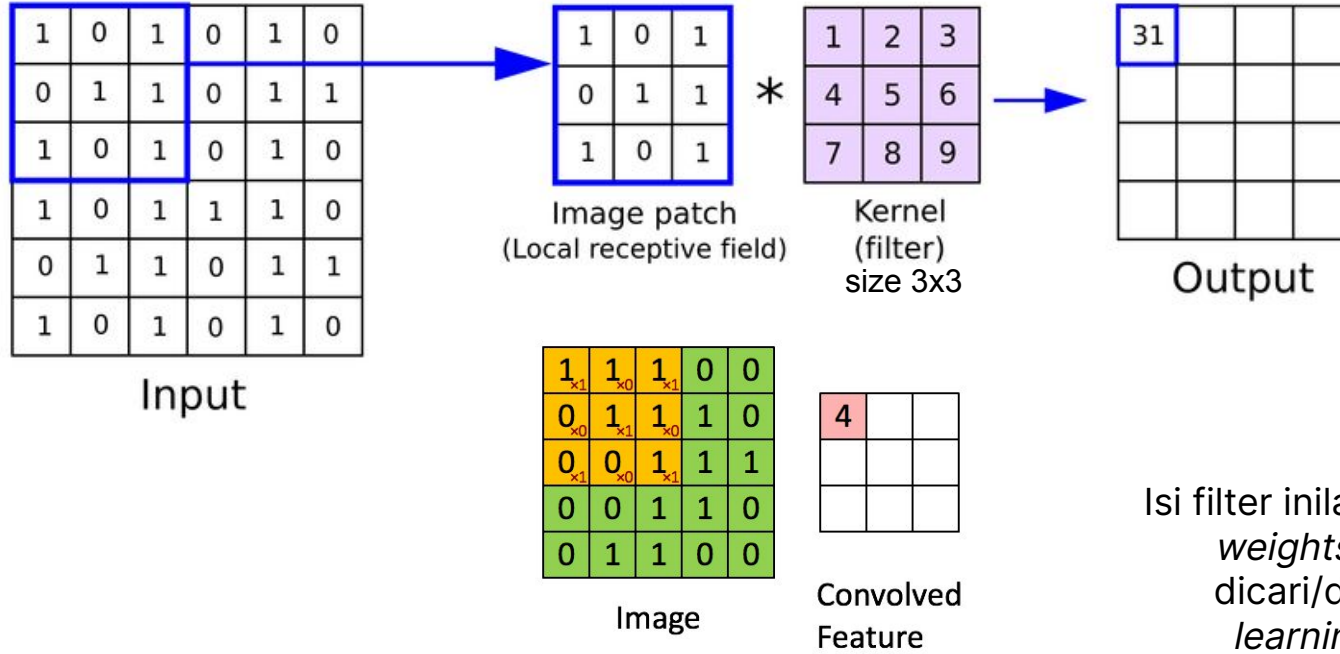
3.1

Convolutional Neural Networks (CNNs)

Convolution Filter

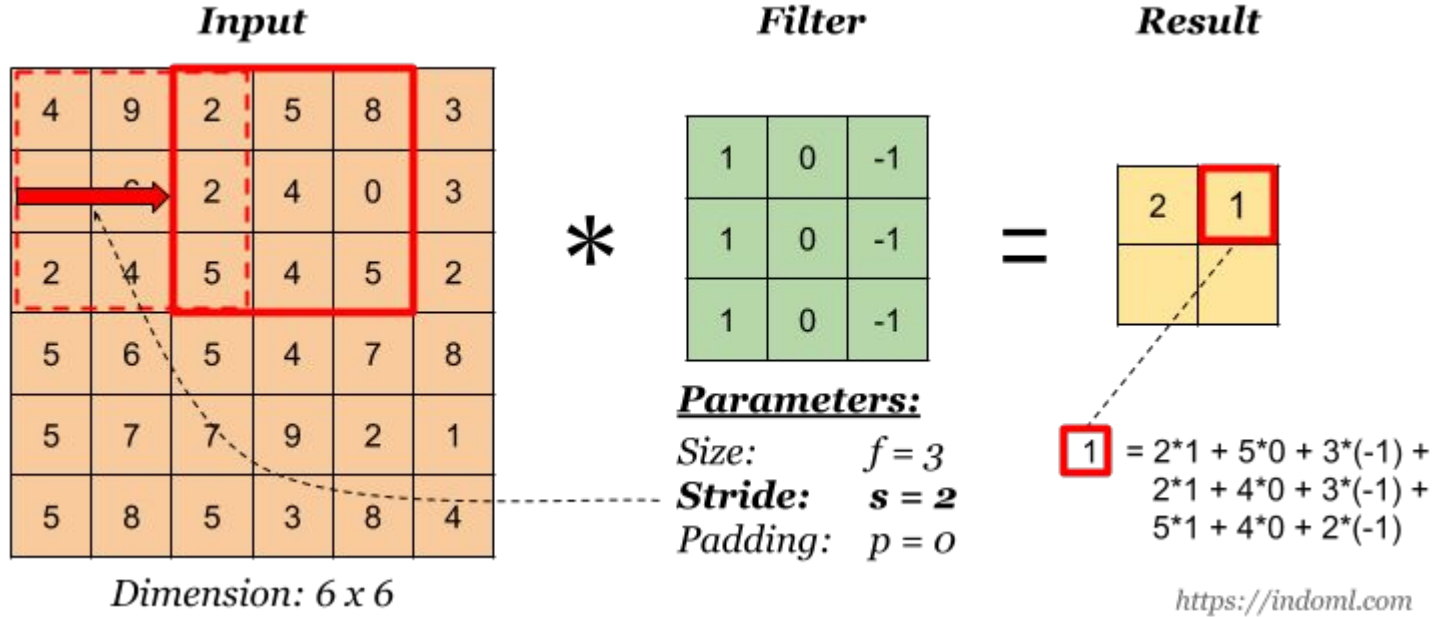
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$		Gaussian blur 3 × 3 (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
Ridge detection	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & -1 \end{bmatrix}$		Gaussian blur 5 × 5 (approximation)	$\frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$				
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$		Unsharp masking 5 × 5 Based on Gaussian blur with amount as 1 and threshold as 0 (with no image mask)	$\frac{-1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & -476 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$				

Convolution Filter

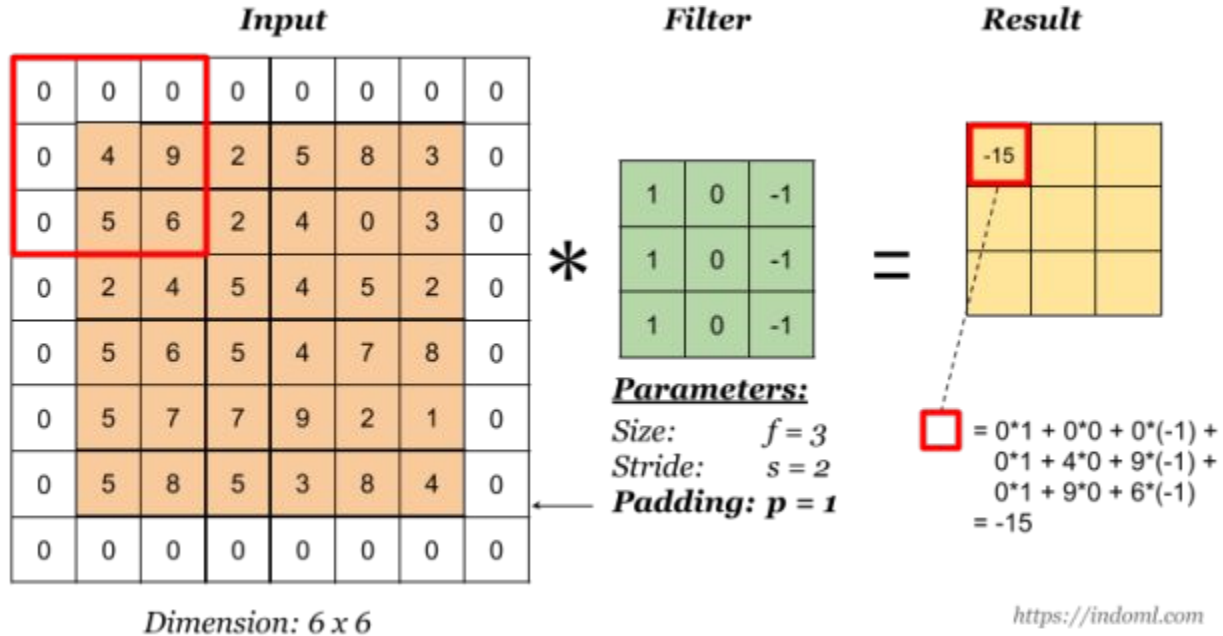


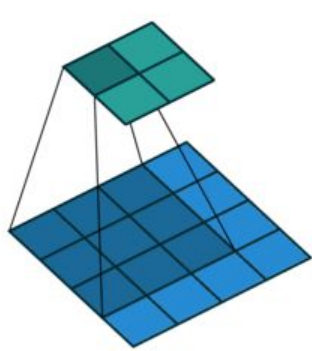
Isi filter inilah yang akan jadi *weights* model yang dicari/dipelajari oleh *learning algorithm*

Stride

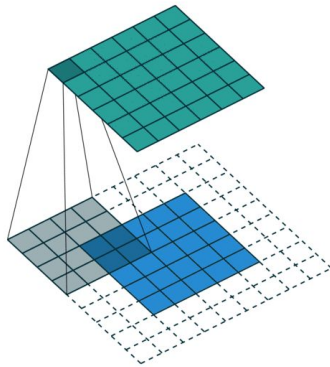


Padding

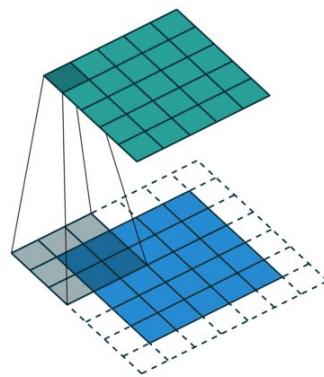




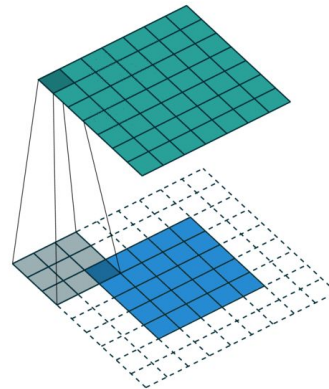
Filter 3×3 , no padding,
no stride



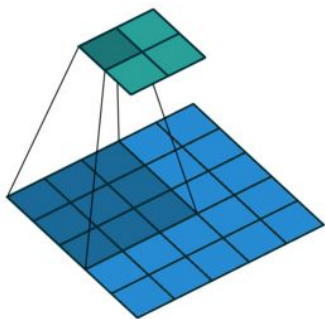
Filter 4×4 , padding=2
(arbitrary), no stride



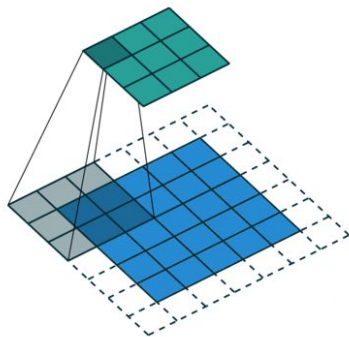
Filter 4×4 , half padding,
no stride



Filter 4×4 , full padding,
no stride

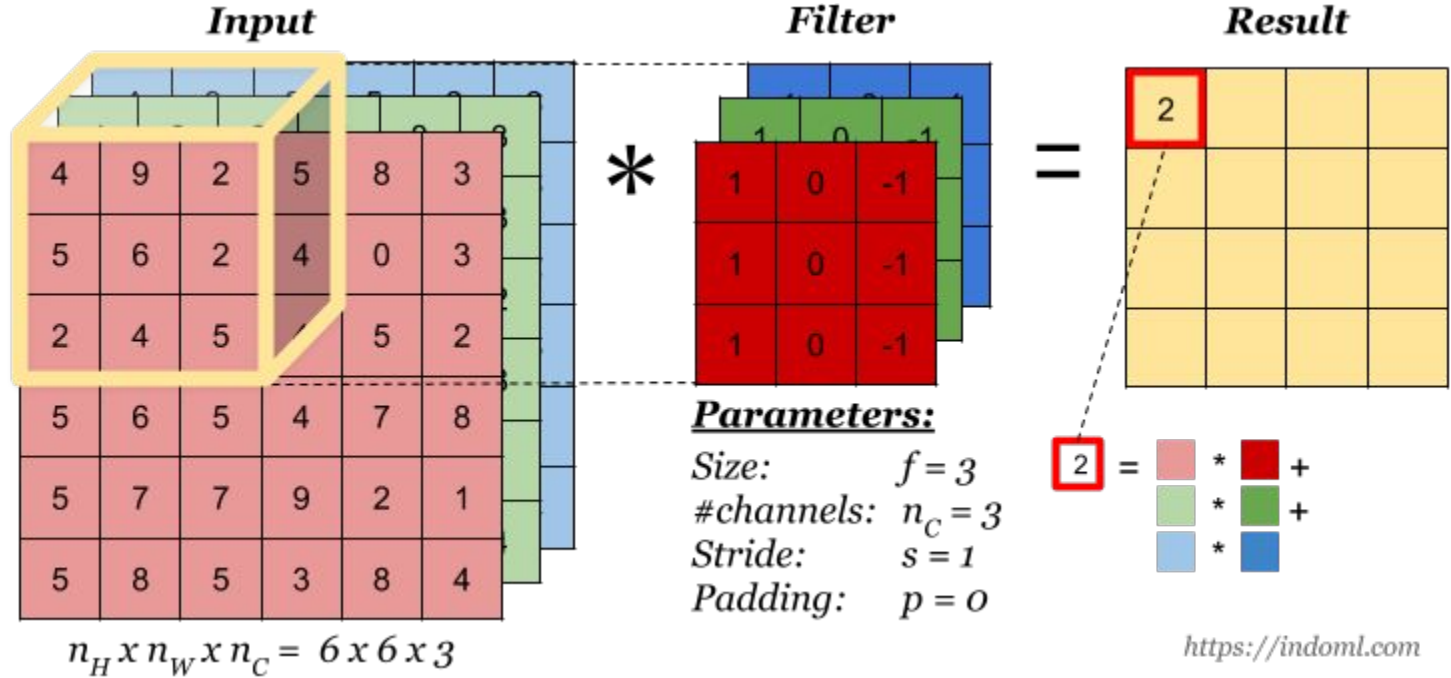


Filter 3×3 , no padding,
stride=2

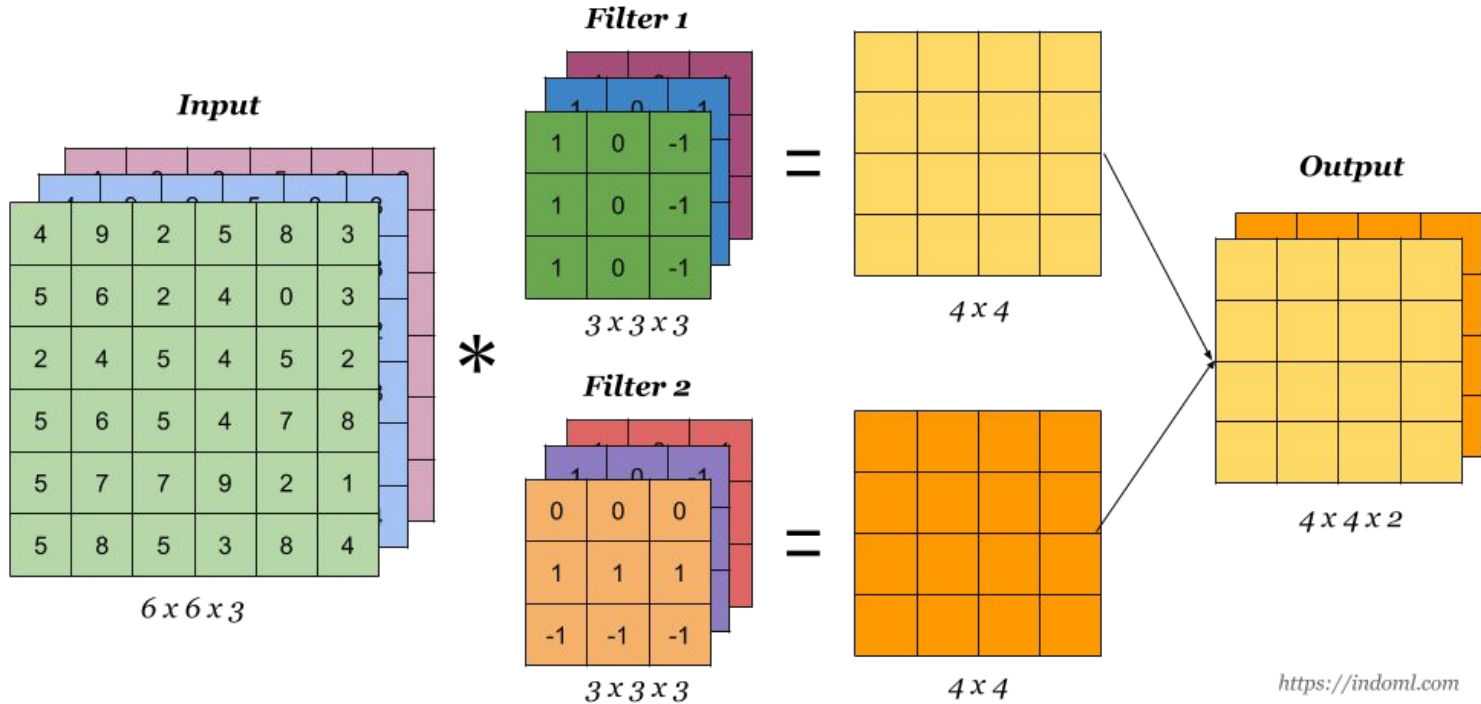


Filter 3×3 , padding=1,
stride=2

Multiple Channels

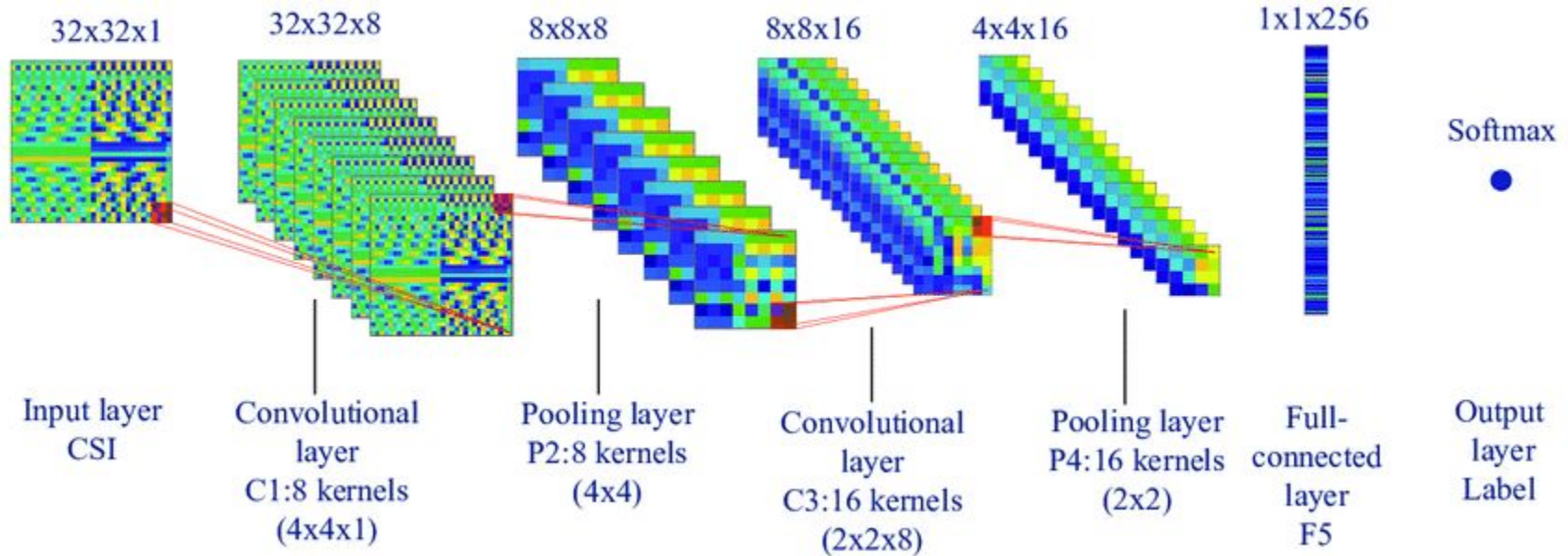


Multiple Filters



<https://indoml.com>

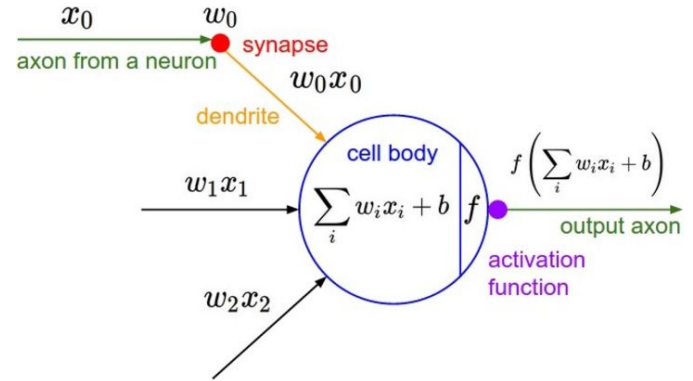
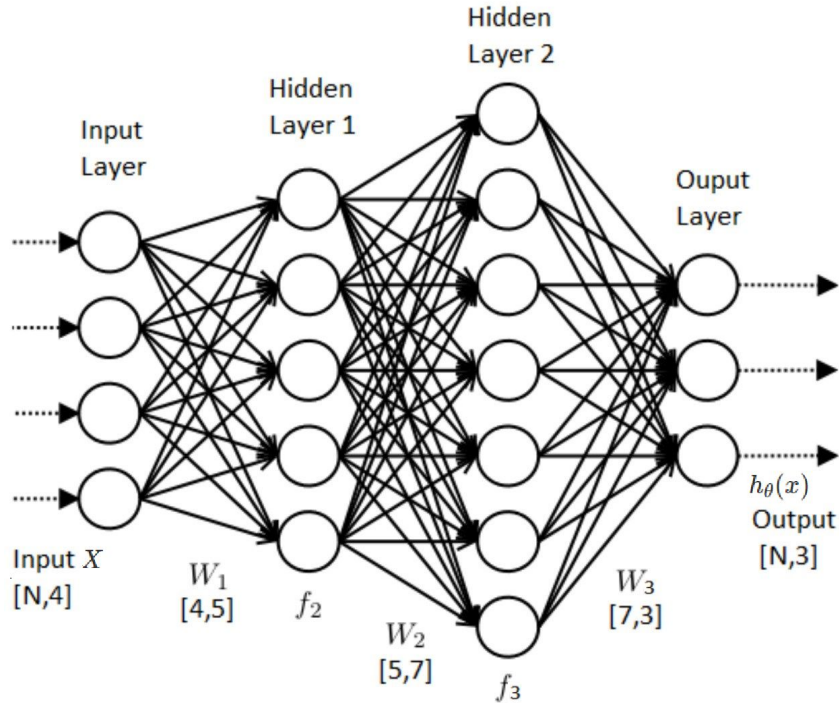
Convolutional Layers



3.2

Fully Connected Neural Networks

FCNNs

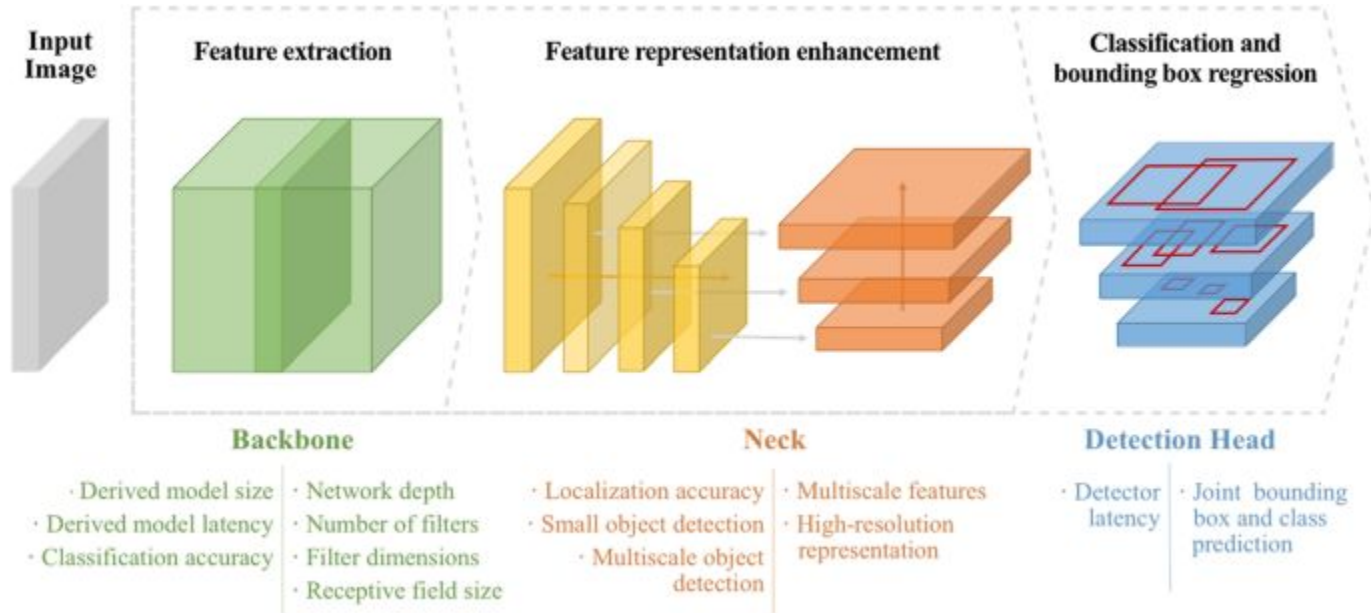


w (*weights*) dan b (*biases*) akan jadi parameter model yang dicari/dipelajari oleh *learning algorithm*

4

EfficientDet

Struktur dari Model Object Detection (Deep Learning)



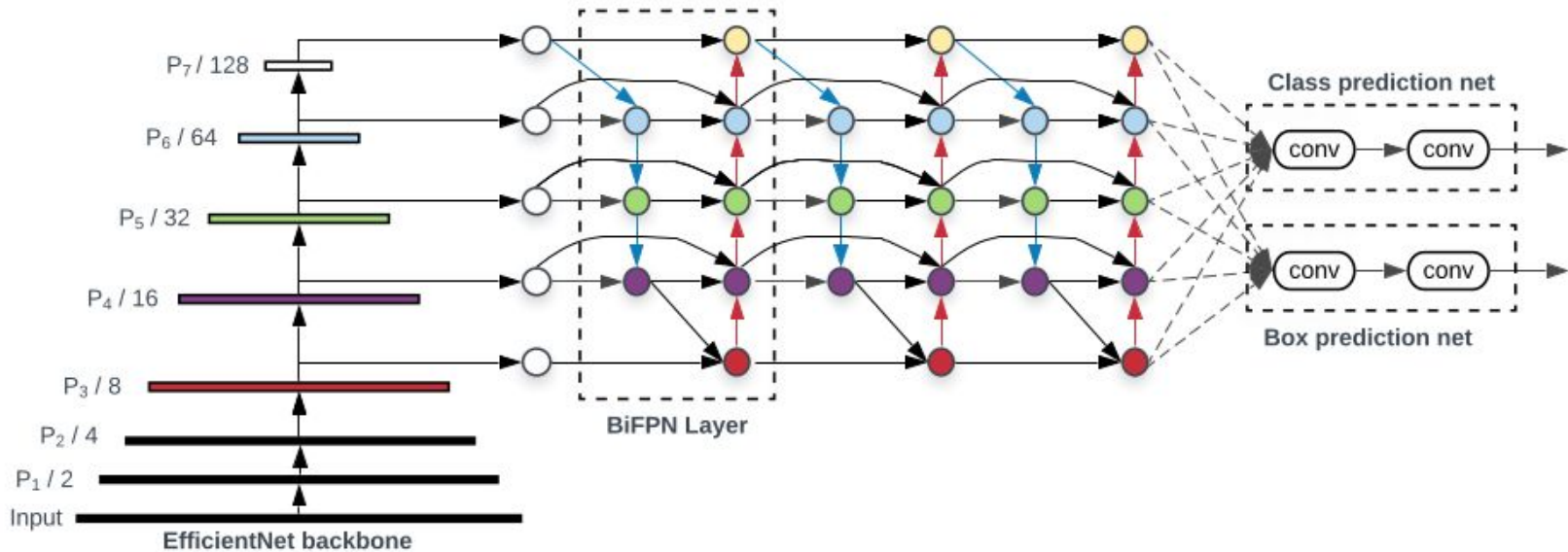


Figure 3: **EfficientDet architecture** – It employs EfficientNet [39] as the backbone network, BiFPN as the feature network, and shared class/box prediction network. Both BiFPN layers and class/box net layers are repeated multiple times based on different resource constraints as shown in Table 1.

Struktur EfficientDet

Backbone: Convolutional Neural Networks (CNNs)

EfficientNet - Compound scaling - dilatih dari dataset ImageNet

Neck: Feature Pyramid Network (FPN)

Bidirectional FPN (BiFPN)

Detection Head

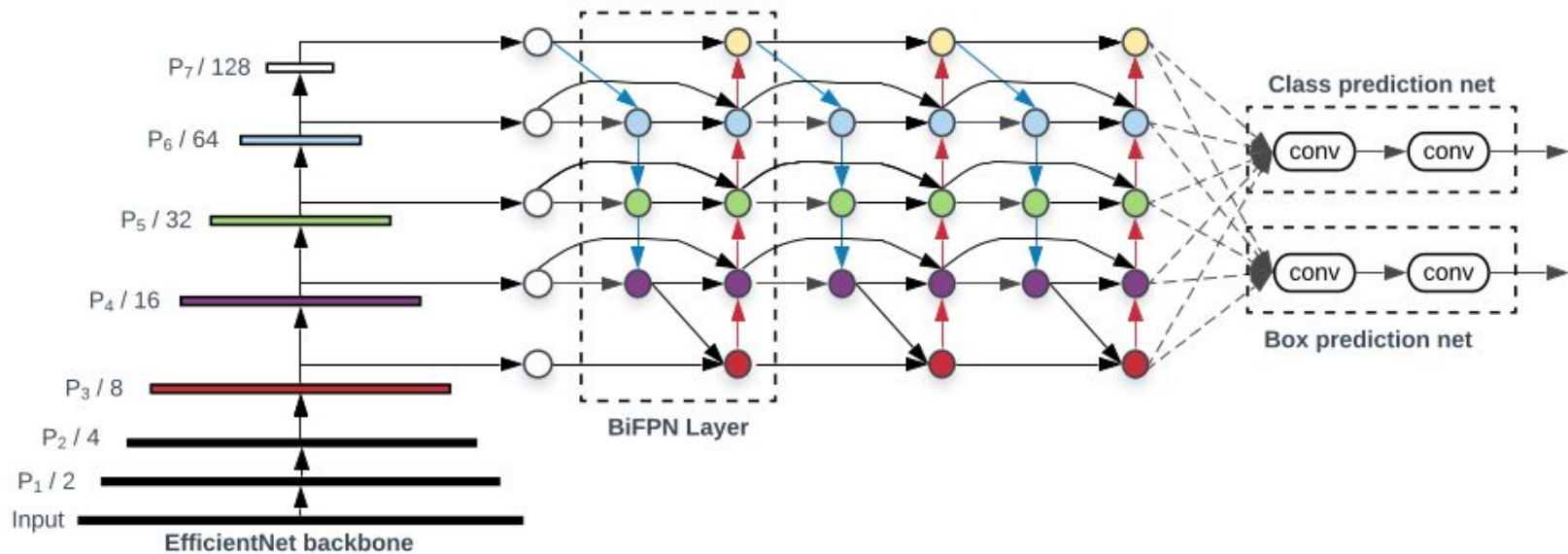
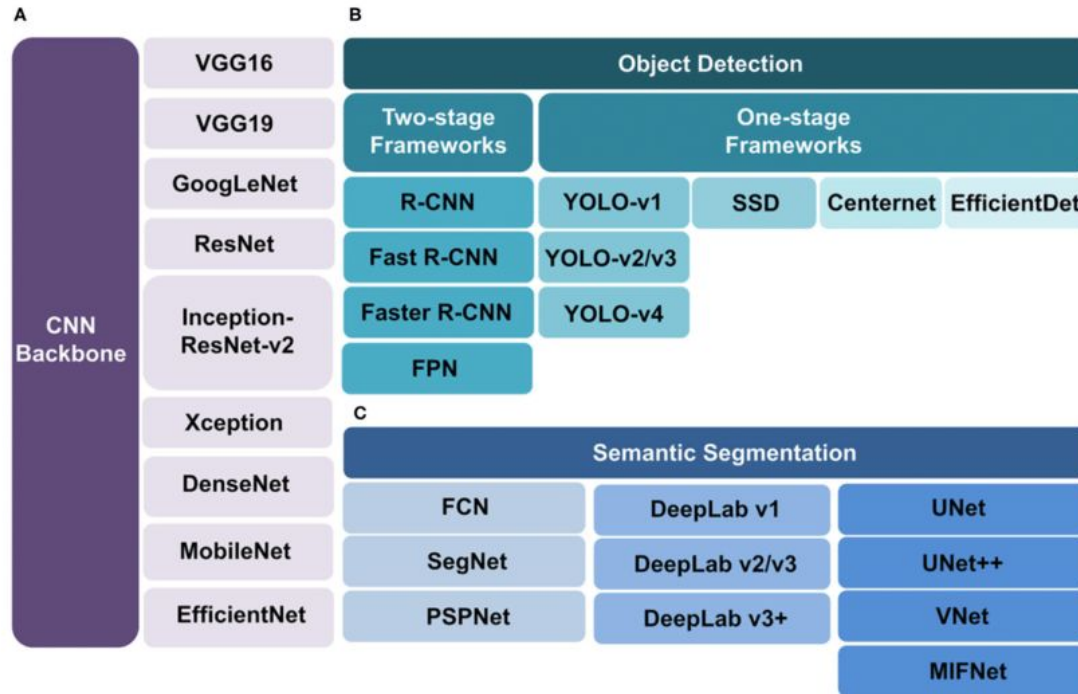


Figure 3: **EfficientDet architecture** – It employs EfficientNet [39] as the backbone network, BiFPN as the feature network, and shared class/box prediction network. Both BiFPN layers and class/box net layers are repeated multiple times based on different resource constraints as shown in Table 1.

Algoritma-algoritma dari CNN backbone, object detection, dan semantic segmentation



4.1

BiFPN

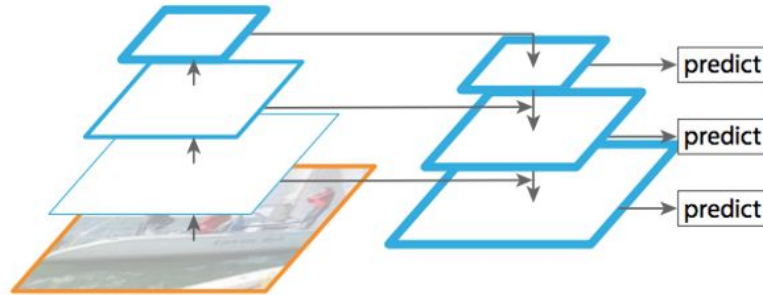
Bidirectional Feature Pyramid Network

Motivasi Feature Pyramid Network

Pemetaan gambar beresolusi tinggi mendeteksi detail yang tidak perlu.

Pemetaan gambar beresolusi rendah mendeteksi pola yang terlalu umum.

Cukup detil + cukup umum → Feature Pyramid Network



Motivasi BiFPN

Tidak semua *input features* berkontribusi sepadan terhadap *output features*.

→ Butuh strategi lain untuk mengetahui kombinasi *input features* mana yang lebih penting. (*multi-scale feature fusion*)

Feature Fusion

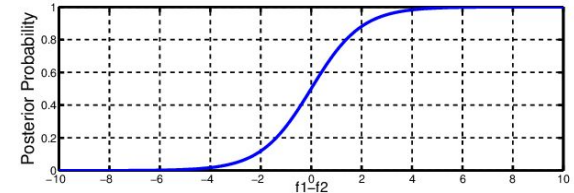
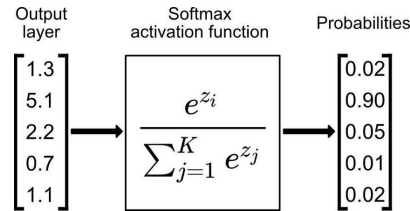
I = input

w = parameter model (weights & biases)

O = output

Unbounded fusion: $O = \sum_i w_i \cdot I_i,$

Softmax-based fusion: $O = \sum_i \frac{e^{w_i}}{\sum_j e^{w_j}} \cdot I_i.$



Fast normalized fusion: $O = \sum_i \frac{w_i}{\epsilon + \sum_j w_j} \cdot I_i$ $w_i \geq 0$ $\epsilon = 0.0001$

30% lebih cepat daripada
Softmax-based fusion

Jenis-Jenis FPN

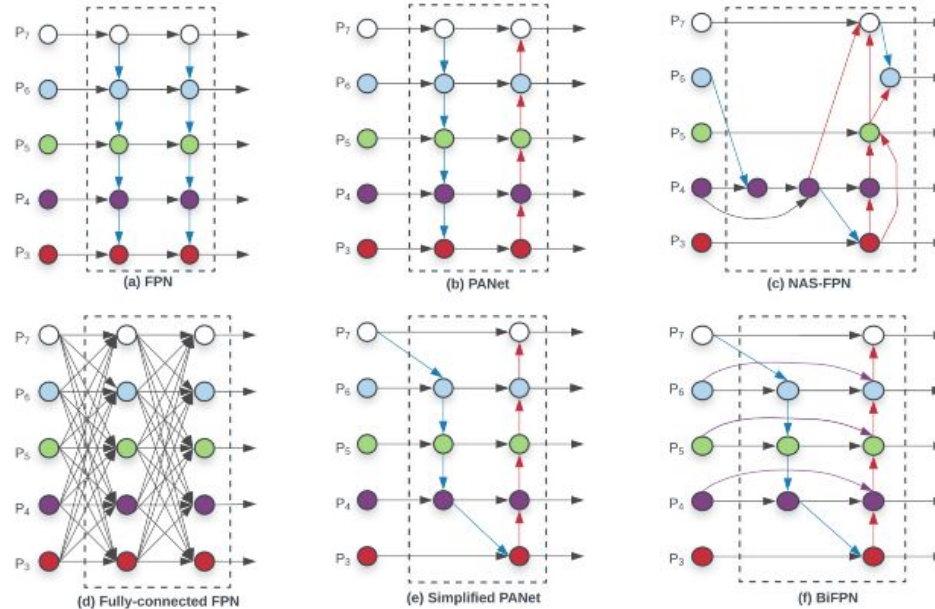
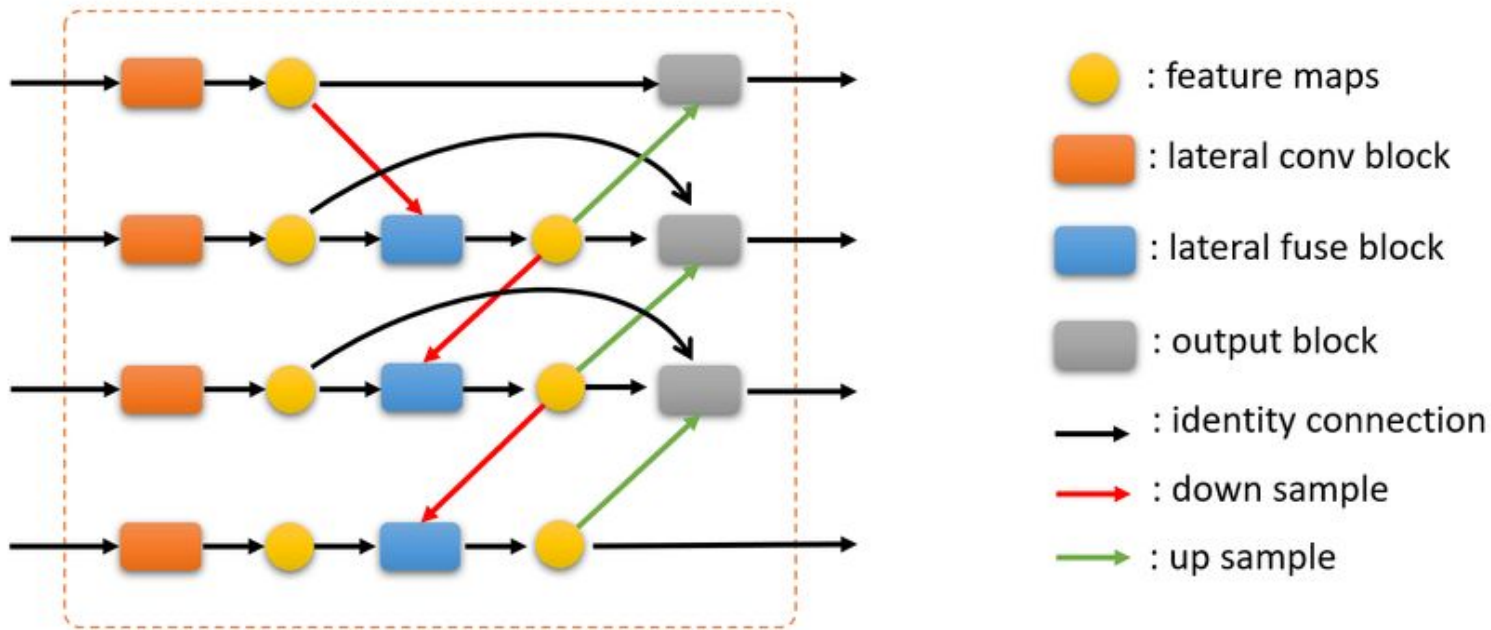


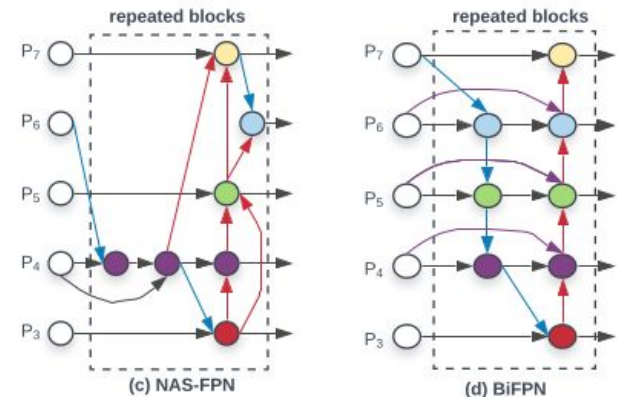
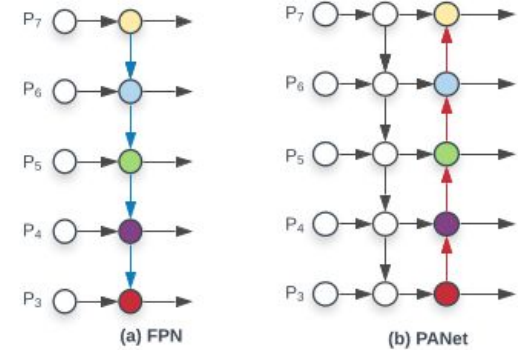
Figure 2: **Feature network design** – (a) FPN [16] introduces a top-down pathway to fuse multi-scale features from level 3 to 7 ($P_3 - P_7$); (b) PANet [19] adds an additional bottom-up pathway on top of FPN; (c) NAS-FPN [5] use neural architecture search to find an irregular feature network topology; (d)-(f) are three alternatives studied in this paper. (d) adds expensive connections from all input feature to output features; (e) simplifies PANet by removing nodes if they only have one input edge; (f) is our BiFPN with better accuracy and efficiency trade-offs.

Struktur BiFPN



Mengapa BiFPN?

- FPN konvensional hanya mengirimkan *informasi features* secara satu arah
- PANet menambahkan agregasi bottom-up tambahan (ada *cross-scale*) secara seragam
- NAS-FPN menggunakan *neural architecture search* untuk mencari konfigurasi *cross-scale* yang lebih baik, namun boros komputasi
- BiFPN menghapus *node* yang hanya punya satu input *edge*, dan menambahkan *edge* baru untuk *node* input-output dalam satu level yang sama



4.2

EfficientNet

Paper EfficientNet

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

Mingxing Tan¹ Quoc V. Le¹

Abstract

Convolutional Neural Networks (ConvNets) are commonly developed at a fixed resource budget, and then scaled up for better accuracy if more resources are available. In this paper, we systematically study model scaling and identify that carefully balancing network depth, width, and resolution can lead to better performance. Based on this observation, we propose a new scaling method that uniformly scales all dimensions of depth/width/resolution using a simple yet highly effective *compound coefficient*. We demonstrate the effectiveness of this method on scaling up MobileNets and ResNet.

To go even further, we use neural architecture search to design a new baseline network and scale it up to obtain a family of models, called *EfficientNets*, which achieve much better accuracy and efficiency than previous ConvNets. In particular, our EfficientNet-B7 achieves state-of-the-art 84.3% top-1 accuracy

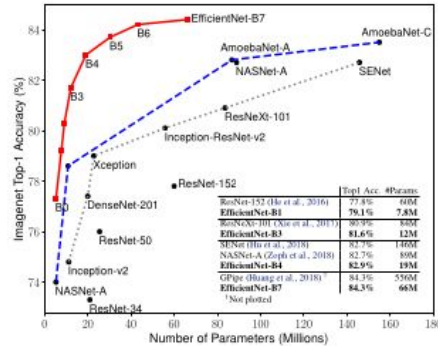


Figure 1. **Model Size vs. ImageNet Accuracy.** All numbers are for single-crop, single-model. Our EfficientNets significantly outperform other ConvNets. In particular, EfficientNet-B7 achieves new state-of-the-art 84.3% top-1 accuracy but being 8.4x smaller and 6.1x faster than GPipe. EfficientNet-B1 is 7.6x smaller and 5.7x faster than ResNet-152. Details are in Table 2 and 4.

Tan, M., & Le, Q. (2019, May). EfficientNet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning* (pp. 6105-6114). PMLR.

<https://arxiv.org/abs/1905.11946>

Model Scaling

Resolution scaling

Semakin besar resolusi (ukuran input), semakin banyak *complex features* dan *pattern* yang lebih detail (fine-grained)

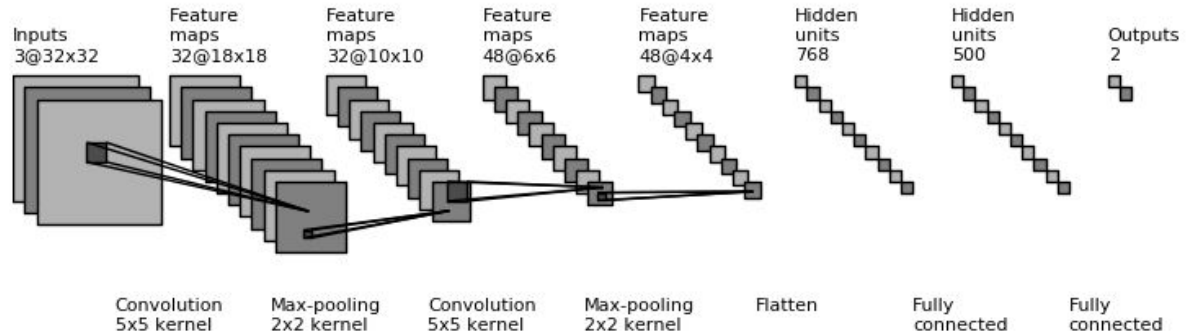


Width scaling

Untuk menangkap *pattern* yang lebih detail, dibutuhkan *channel* yang lebih banyak

Depth scaling

Untuk menangkap *features* yang lebih kompleks, dibutuhkan *layer* yang lebih banyak (lebih dalam)



EfficientNet - *compound scaling*

Memperbesar model dengan cara yang lebih seimbang.

ϕ ditentukan oleh user

$$\text{depth: } d = \alpha^\phi$$

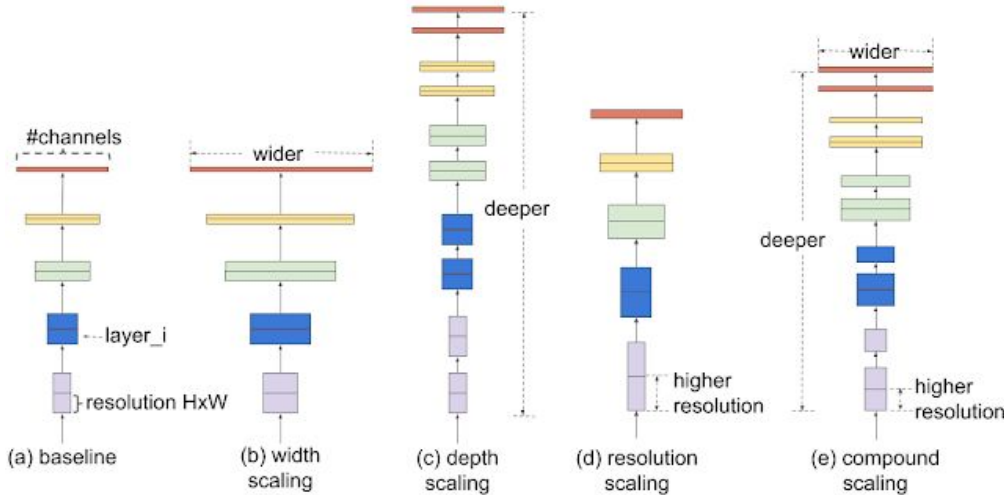
$$\text{width: } w = \beta^\phi$$

$$\text{resolution: } r = \gamma^\phi$$

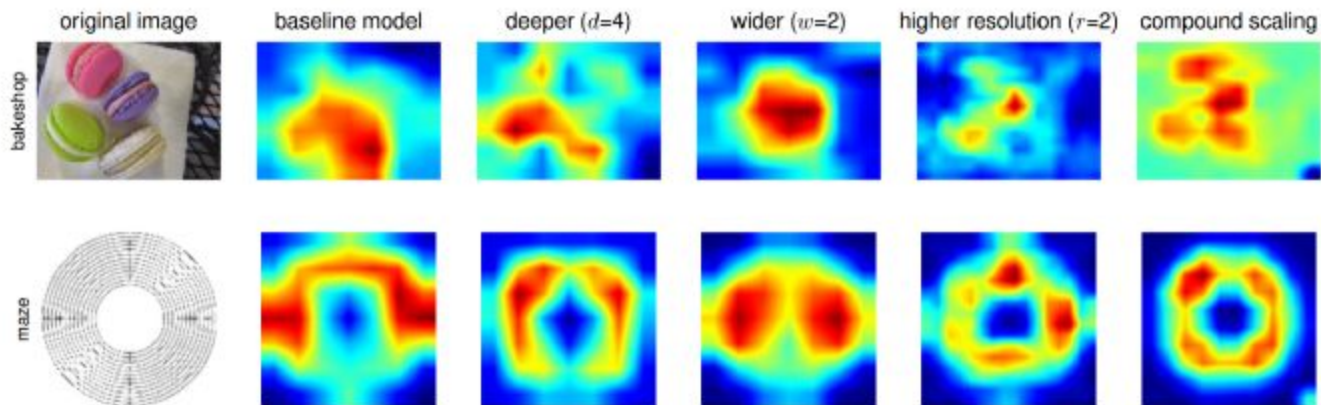
$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

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

Kenapa? Operasi di ars. convolutional berbanding lurus dengan d, w^2, r^2



EfficientNet - Efek *compound scaling*



<https://medium.com/mlearning-ai/understanding-efficientnet-the-most-powerful-cnn-architecture-eaeb40386fad>

EfficientNet - Perhitungan *compound scaling*

$f = \alpha \cdot \beta^\phi \cdot \gamma^\phi$ (network scaling factor)

$$\alpha = 1.1 \quad \beta = 1.15 \quad \gamma = 1.2 \quad \rightarrow \quad \alpha \cdot \beta^2 \cdot \gamma^2 = 2.0948399999999996$$

$\phi = 0$ | EfficientNet-D0

$$\begin{aligned} f &= 1.1 \cdot 1.15^0 \cdot 1.2^0 \\ &= 1.1 \end{aligned}$$

$$\text{depth: } d = \alpha^\phi$$

$$\text{width: } w = \beta^\phi$$

$$\text{resolution: } r = \gamma^\phi$$

$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

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

$\phi = 3$ | EfficientNet-D3

$$\begin{aligned} f &= 1.1 \cdot 1.15^3 \cdot 1.2^3 \\ &= 2.8908791999999999 \end{aligned}$$

$\phi = 6$ | EfficientNet-D6

$$\begin{aligned} f &= 1.1 \cdot 1.15^2 \cdot 1.2^2 \\ &= 7.597438680902396 \end{aligned}$$

EfficientNet - Arsitektur dasar (B0, $\phi=0$)

Table 1. EfficientNet-B0 baseline network – Each row describes a stage i with \hat{L}_i layers, with input resolution $\langle \hat{H}_i, \hat{W}_i \rangle$ and output channels \hat{C}_i . Notations are adopted from equation 2.

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	MBCConv1, k3x3	112×112	16	1
3	MBCConv6, k3x3	112×112	24	2
4	MBCConv6, k5x5	56×56	40	2
5	MBCConv6, k3x3	28×28	80	3
6	MBCConv6, k5x5	14×14	112	3
7	MBCConv6, k5x5	14×14	192	4
8	MBCConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Scaling formula

Input image resolution – Since feature level 3-7 are used in BiFPN, the input resolution must be dividable by $2^7 = 128$, so we linearly increase resolutions using equation:

$$R_{input} = 512 + \phi \cdot 128 \quad (3)$$

Box/class prediction network – we fix their width to be always the same as BiFPN (i.e., $W_{pred} = W_{bifpn}$), but linearly increase the depth (#layers) using equation:

$$D_{box} = D_{class} = 3 + \lfloor \phi/3 \rfloor \quad (2)$$

BiFPN network – we linearly increase BiFPN depth D_{bifpn} (#layers) since depth needs to be rounded to small integers. For BiFPN width W_{bifpn} (#channels), exponentially grow BiFPN width W_{bifpn} (#channels) as similar to [39]. Specifically, we perform a grid search on a list of values $\{1.2, 1.25, 1.3, 1.35, 1.4, 1.45\}$, and pick the best value 1.35 as the BiFPN width scaling factor. Formally, BiFPN width and depth are scaled with the following equation:

$$W_{bifpn} = 64 \cdot (1.35^\phi), \quad D_{bifpn} = 3 + \phi \quad (1)$$

Konfigurasi scaling

	Input size R_{input}	Backbone Network	BiFPN		Box/class
			#channels W_{bifpn}	#layers D_{bifpn}	#layers D_{class}
D0 ($\phi = 0$)	512	B0	64	3	3
D1 ($\phi = 1$)	640	B1	88	4	3
D2 ($\phi = 2$)	768	B2	112	5	3
D3 ($\phi = 3$)	896	B3	160	6	4
D4 ($\phi = 4$)	1024	B4	224	7	4
D5 ($\phi = 5$)	1280	B5	288	7	4
D6 ($\phi = 6$)	1280	B6	384	8	5
D7 ($\phi = 7$)	1536	B6	384	8	5
D7x	1536	B7	384	8	5

Table 1: **Scaling configs for EfficientDet D0-D6** – ϕ is the compound coefficient that controls all other scaling dimensions; *BiFPN*, *box/class net*, and *input size* are scaled up using equation 1, 2, 3 respectively.

Table 2. **EfficientNet Performance Results on ImageNet** (Russakovsky et al., 2015). All EfficientNet models are scaled from our baseline EfficientNet-B0 using different compound coefficient ϕ in Equation 3. ConvNets with similar top-1/top-5 accuracy are grouped together for efficiency comparison. Our scaled EfficientNet models consistently reduce parameters and FLOPs by an order of magnitude (up to 8.4x parameter reduction and up to 16x FLOPs reduction) than existing ConvNets.

	Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPs	Ratio-to-EfficientNet
$\phi=0$	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
$\phi=1$	EfficientNet-B1	79.1%	94.4%	7.8M	1x	0.70B	1x
	ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
	DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
	Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x	
$\phi=2$	EfficientNet-B2	80.1%	94.9%	9.2M	1x	1.0B	1x
	Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
	Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
$\phi=3$	EfficientNet-B3	81.6%	95.7%	12M	1x	1.8B	1x
	ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
	PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
$\phi=4$	EfficientNet-B4	82.9%	96.4%	19M	1x	4.2B	1x
	SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
	NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
	AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
	PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
$\phi=5$	EfficientNet-B5	83.6%	96.7%	30M	1x	9.9B	1x
	AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
$\phi=6$	EfficientNet-B6	84.0%	96.8%	43M	1x	19B	1x
	EfficientNet-B7	84.3%	97.0%	66M	1x	37B	1x
	GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

We omit ensemble and multi-crop models (Hu et al., 2018), or models pretrained on 3.5B Instagram images (Mahajan et al., 2018).

Tan, M., Pang, R., & Le, Q. V. (2020). [Efficientdet: Scalable and efficient object detection](#). In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10781-10790).

Kesimpulan

- ?

Implementasi Kode Penggunaan EfficientDet

<https://colab.research.google.com/drive/1W0uwwwvevyH-jVmCkBGkRoHFtB4xPToX>

Saduran dari <https://github.com/google/automl>