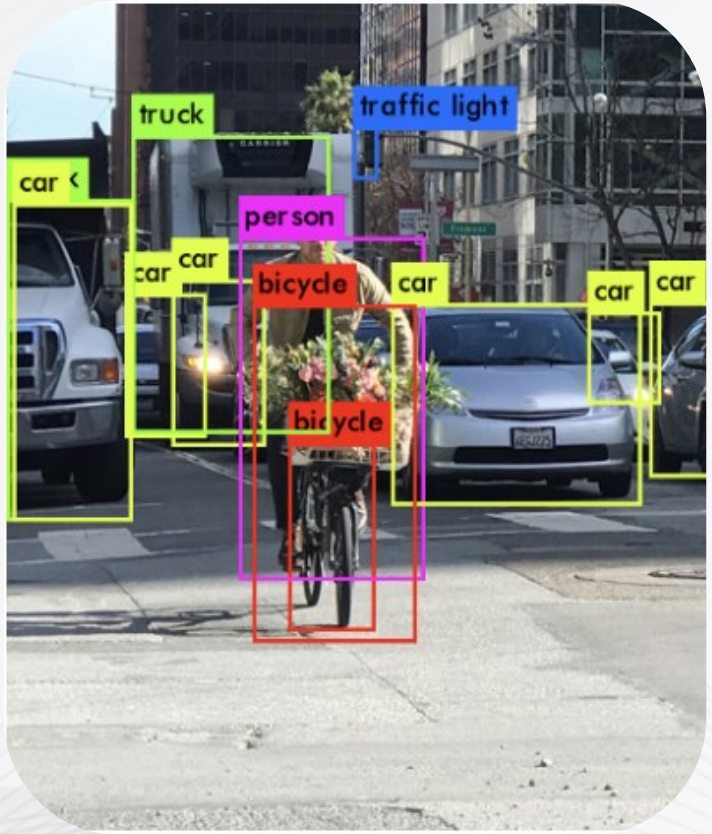
Computer Vision – You Only Look Once

Prof. Dr.Eng FITRI UTAMINGRUM, ST., MT

brone.ub.ac.id

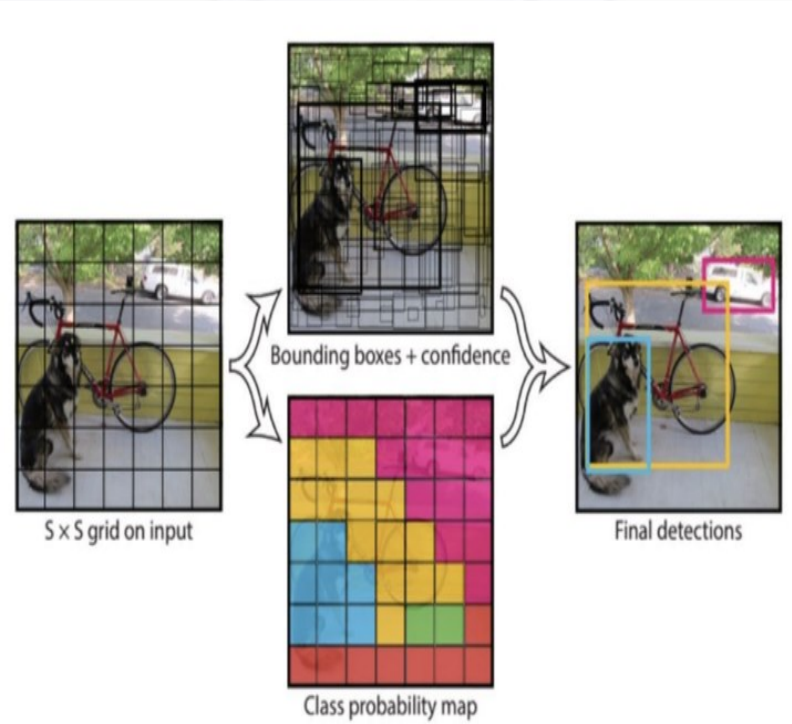
YOU ONLY LOOK ONCE



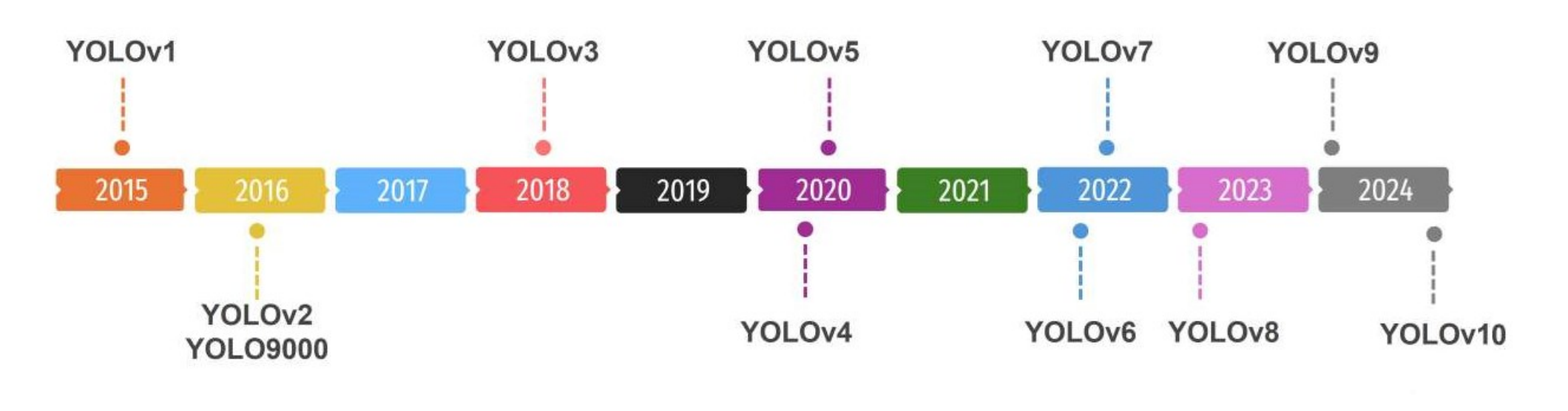
You Only Look Once (YOLO) adalah sebuah kelompok arsitektur deep learning yang digunakan untuk melakukan deteksi objek dalam gambar dan video. Arsitektur yang digunakan oleh YOLO berbeda dengan R-CNN dimana sistem deteksi terdiri hanya dari jaringan *neural network* yang memungkinkan YOLO dijalankan secara *real-time*

CARA KERJA YOLO

1. Gambar input dibagi menjadi grid berukuran S x S
2. Setiap grid akan digunakan sebagai referensi objek yang terdapat pada lokasi tersebut
3. YOLO menghasilkan bounding box untuk setiap objek yang terdeteksi
4. Bounding box yang dihasilkan akan dibersihkan dengan menggunakan non-maximum suppression



PERKEMBANGAN YOLO



PERKEMBANGAN YOLO

YOLOv1

YOLOv1 adalah versi pertama dari YOLO yang diperkenalkan pada tahun 2015. Versi ini menggunakan arsitektur deep learning yang sederhana dengan 24 layer dan mampu menghasilkan deteksi objek dengan kecepatan 150~ fps.

YOLOv2

YOLOv2 diperkenalkan pada tahun 2016 dan merupakan pengembangan dari YOLOv1. YOLOv2 memiliki arsitektur yang lebih kompleks dengan 29 layer dan menggunakan teknik skip connection untuk meningkatkan akurasi deteksi serta arsitektur berjenis *fully-convolutional*.

PERKEMBANGAN YOLO

YOLOv3

YOLOv3 diperkenalkan pada tahun 2018. Arsitektur yang digunakan lebih kompleks dengan 53 layer. YOLOv3 membawa pengembangan *multi-scale feature* yang memungkinkan YOLO melakukan deteksi dengan ukuran objek yang sangat bervariasi

YOLOv4

YOLOv4 diperkenalkan pada tahun 2020. Versi ini memiliki arsitektur yang mirip dengan YOLOv3 dengan pengembangan pada bagian *multi-scale feature* yang lebih baik dan proses training yang lebih *robust*

PERKEMBANGAN YOLO

YOLOv5

YOLOv5 diperkenalkan pada tahun 2020. Arsitektur ini tidak jauh berbeda dengan YOLOv4 namun terdapat perubahan signifikan pada proses training dengan menggunakan AutoAnchor untuk menentukan bentuk Anchor yang paling sesuai

YOLOv6

YOLOv6 diperkenalkan pada tahun 2022. Pada versi ini terdapat pengembagan yang cukup besar pada bagian *backbone* yang menggunakan RepVGGBlock yang meningkatkan pararelisme model dan penggunaan fungsi Focal Loss yang membantu dalam proses training

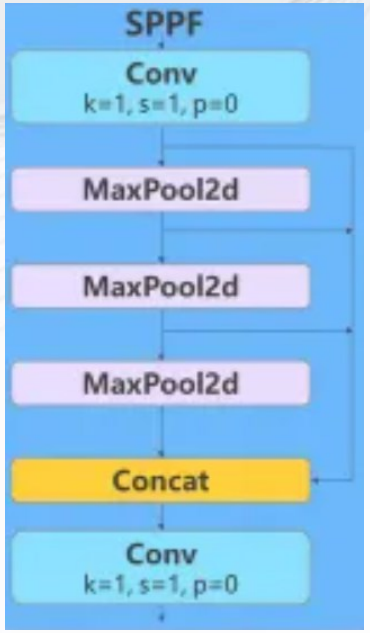
PERKEMBANGAN YOLO

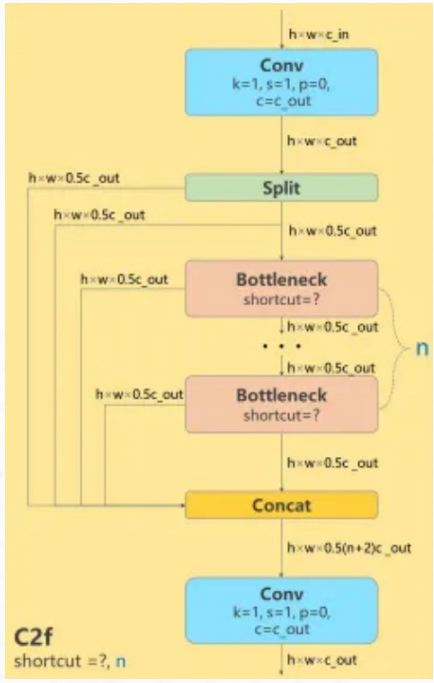
YOLOv7

YOLOv7 diperkenalkan pada tahun 2022. Pengembangan yang dilakukan pada versi ini berupa penambahan E-ELAN (Extended Efficient Layer Aggregation Network), auxillary head, serta *model scaling* yang memungkinkan adanya pengaturan pada ukuran model sehingga dapat disesuaikan dengan kebutuhan dengan mudah.

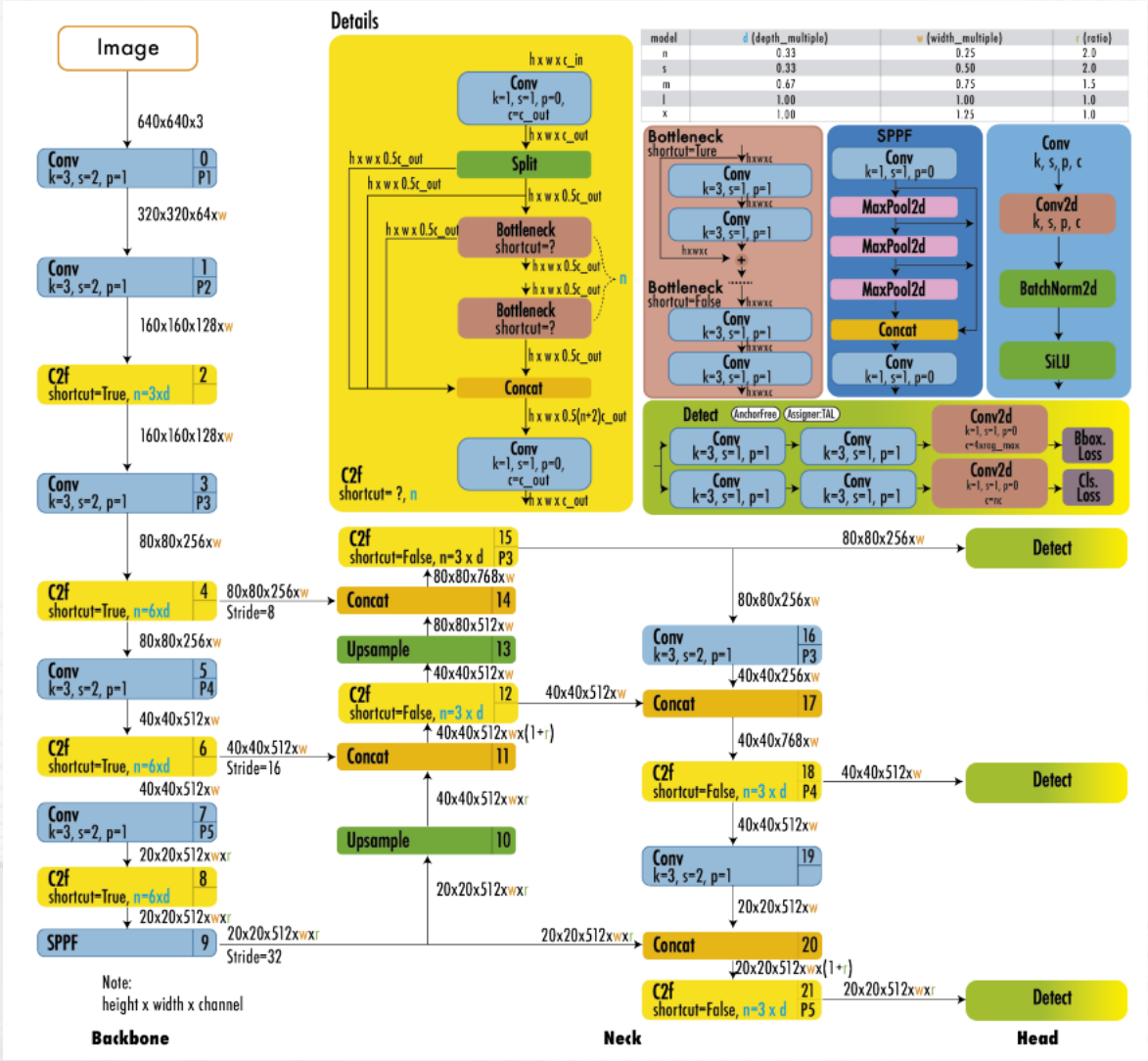
YOLOv8

- YOLOv8 merupakan pengembangan dari versi YOLO sebelumnya dengan menggunakan basis YOLOv5 yang diperkenalkan pada tahun 2023
- Pengembangan yang ada pada YOLOv8 meliputi :
  - Model Anchor-free dan SPPF yang mempercepat proses komputasi
  - Distribution Focal Loss yang mencegah overfitting
  - Modul C2f dan fungsi aktivasi Swish
- Berbeda dengan versi YOLO sebelumnya, YOLOv8 juga dapat melakukan segmentasi dan pose estimation





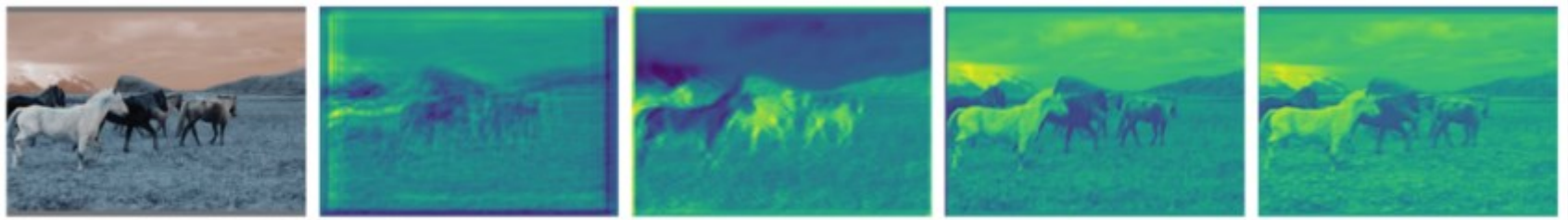
ARSITEKTUR YOLOv8



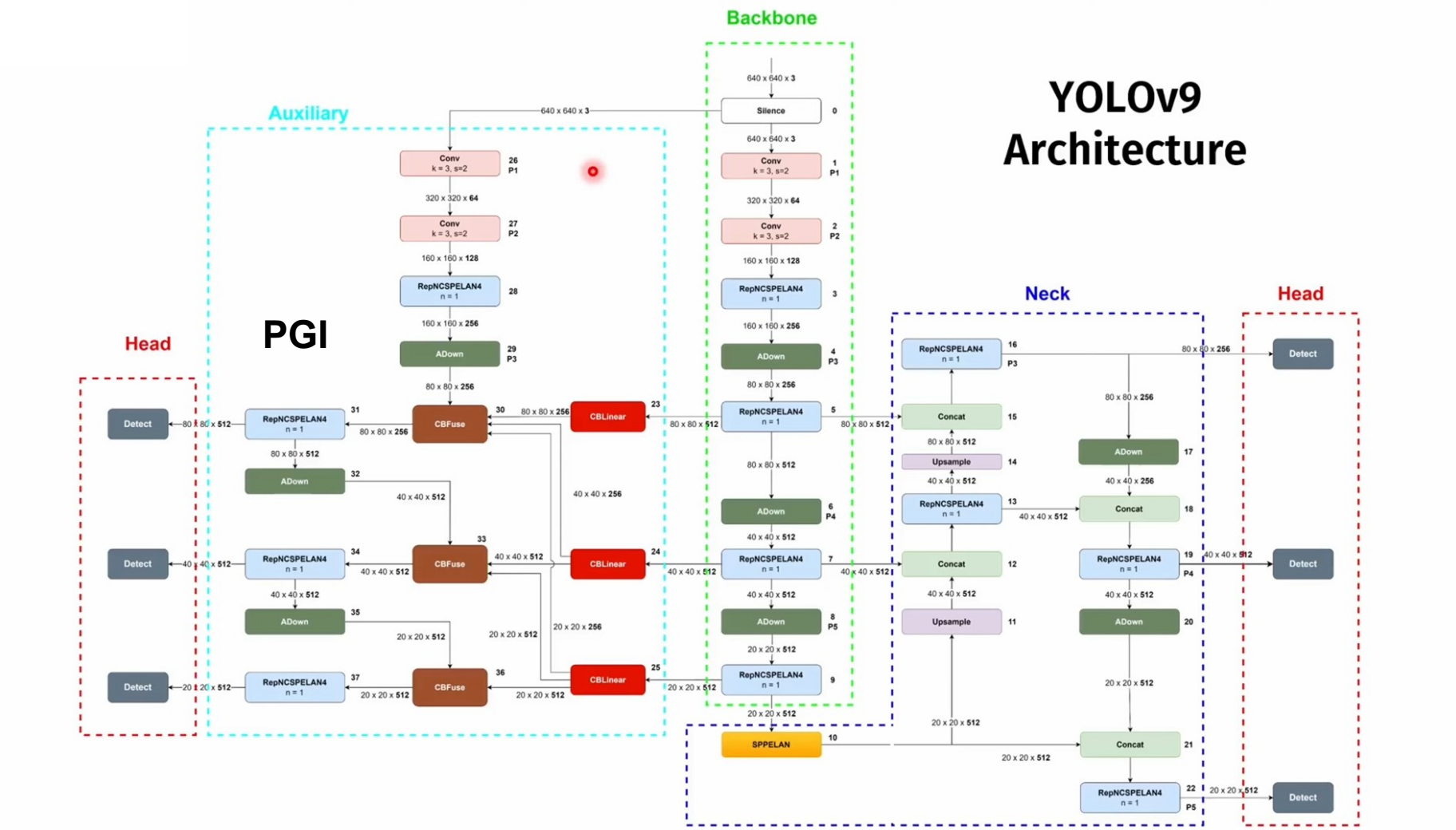
(Tevez et.al, 2023)

YOLOv9

- YOLOv9 juga merupakan pengembangan dari versi YOLO sebelumnya yang diperkenalkan pada tahun 2024. Namun, YOLOv9 dikembangkan dengan menggunakan basis YOLOv7
- Pengembangan yang ada pada YOLOv9 meliputi :
  - Programmable Gradient Information (PGI) yang mencegah hilangnya informasi pada layer yang dalam pada jaringan
  - Generalized Efficient Layer Aggregation Network (GELAN) yang merupakan pengembangan dari E-ELAN dengan akurasi dan kecepatan yang lebih tinggi
- Fokus pengembangan pada YOLOv9 cenderung pada peningkatan performa, namun tanpa meningkatkan kompleksitas komputasi



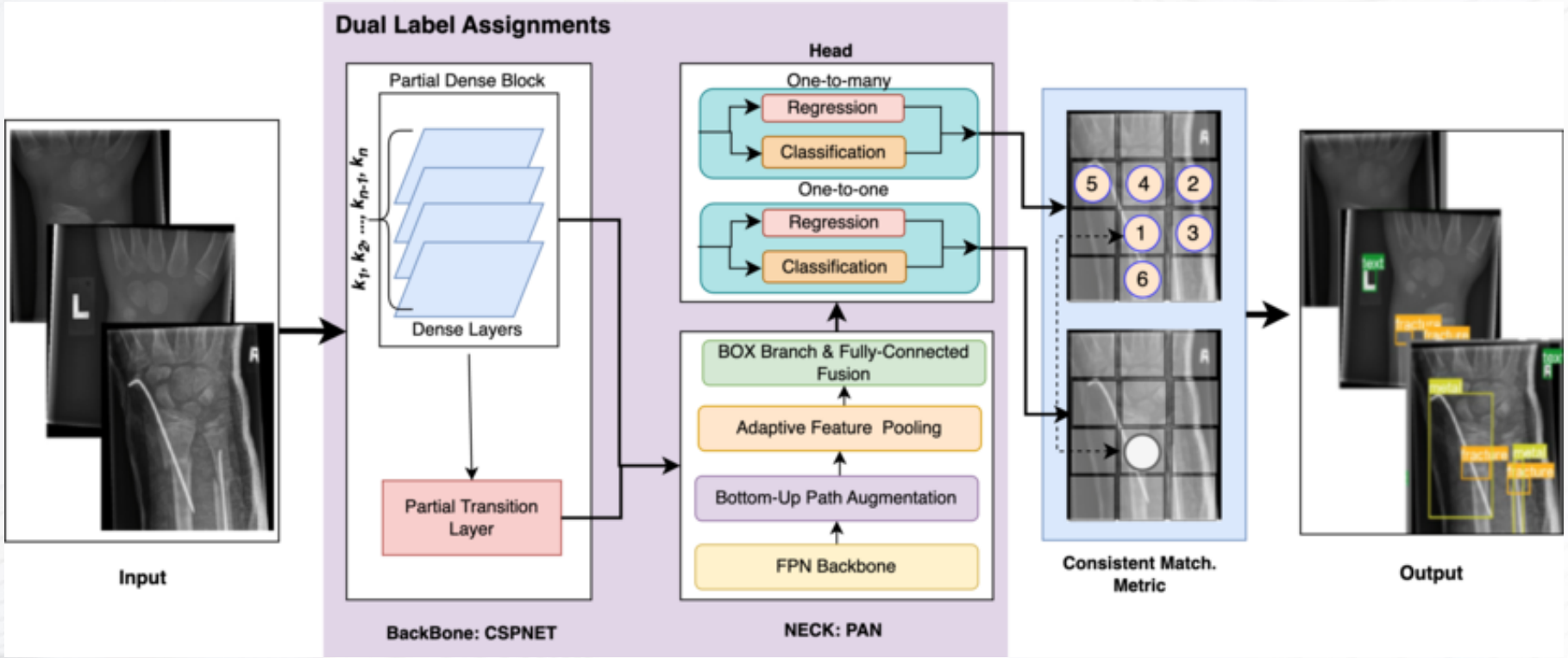
ARSITEKTUR YOLOv9



YOLOv10

- YOLOv10 merupakan pengembangan dari versi YOLO sebelumnya yang diperkenalkan pada tahun 2024. Versi ini dikembangkan dengan menggunakan basis YOLOv8
- Pengembangan yang ada pada YOLOv10 meliputi :
  - Penghilangan proses Non Maximum Supression (NMS) yang mempercepat proses deteksi dengan menggunakan *dual-assignment*
  - Penggunaan *depth-wise separable convolution* yang mempercepat proses klasifikasi
  - Penggunaan kernel berukuran besar yang membantu ekstraksi fitur pada layer konvolusi
- YOLOv10 merupakan algoritma SotA (State of the Art) untuk sistem deteksi yang memiliki performa lebih baik dibandingkan model berbasis transformer

ARSITEKTUR YOLOv10

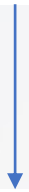


IMPLEMENTASI YOLOv10

- Berbeda dengan model *deep learning* lainnya yang umumnya diimplementasikan secara langsung menggunakan library seperti TensorFlow, PyTorch, JAX, Caffe, model YOLO umumnya diimplementasikan menggunakan library Ultralytics
- Model YOLO terbaru juga umumnya memiliki banyak varian ukuran dan dapat diatur lebih dalam lagi jika perlu.
- YOLO juga umumnya digunakan dengan menggunakan model *pretrained* COCO yang dapat membantu akurasi dan kecepatan training
- Sebagai contoh, akan diimplementasikan model YOLOv10 dengan varian ukuran nano

IMPLEMENTASI YOLOv10

```
python -m pip install ultralytics
```



```
from ultralytics import YOLO

model_scratch = YOLO("yolov10n.yaml")

model_pretrained = YOLO("yolov10n.pt")
```

Varian model YOLO

- Nano (n)
- Small (s)
- Medium (m)
- Big (b)
- Large (l)
- eXtra (x)

IMPLEMENTASI YOLOv10

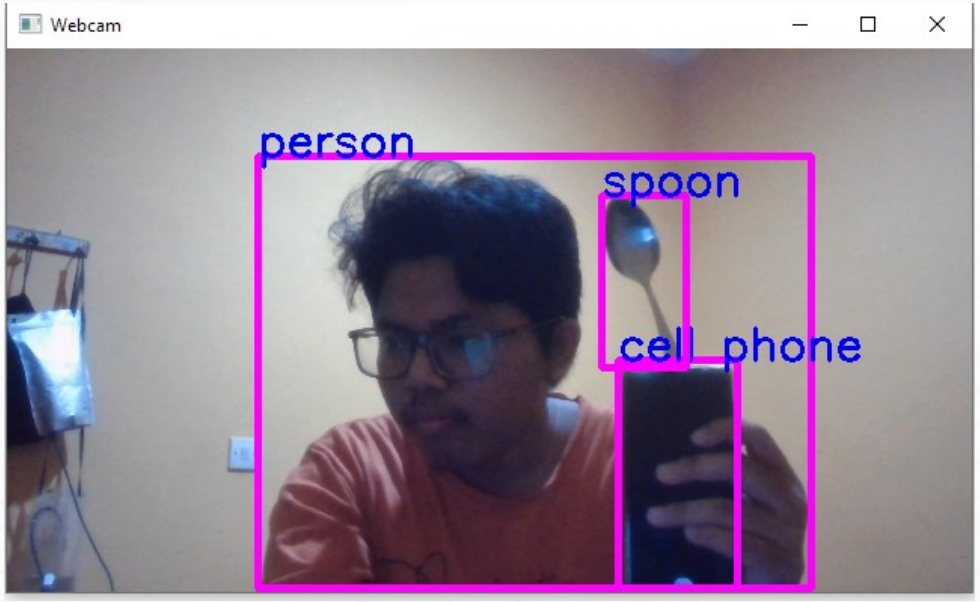
Proses training dapat dilakukan dengan menggunakan fungsi train(). Untuk melakukan inferensi, dapat dilakukan dengan memasukkan data secara langsung ke dalam model.

```
model_scratch.train(data="coco8.yaml", epochs=100, imgsz=640)

result = model_scratch("image.jpg")

result[0].show()
```

IMPLEMENTASI YOLOv10

```python
from ultralytics import YOLO
import cv2
import math

cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)

model = YOLO("yolov10n.pt")

classNames = ["person", "bicycle", "car", "motorbike", "aeroplane", "bus", "train", "truck", "boat",···

while True:
    success, img = cap.read()
    results = model(img, stream=True)

    for r in results:
        boxes = r.boxes

        for box in boxes:
            x1, y1, x2, y2 = box.xyxy[0]
            x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)

            cv2.rectangle(img, (x1, y1), (x2, y2), (255, 0, 255), 3)

            cls = int(box.cls[0])
            org = [x1, y1]
            font = cv2.FONT_HERSHEY_SIMPLEX
            fontScale = 1
            color = (255, 0, 0)
            thickness = 2

            cv2.putText(img, classNames[cls], org, font, fontScale, color, thickness)

    cv2.imshow('Webcam', img)
    if cv2.waitKey(1) == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Hasil Deteksi



IMPLEMENTASI YOLOv10

Yolov10n.yaml

```yaml
# Parameters
nc: 80 # number of classes
scales: # model compound scaling constants, i.e. 'model=yolov8n.yaml' will call yolov8.yaml with scale 'n'
  # [depth, width, max_channels]
  n: [0.33, 0.25, 1024]

# YOLOv8.0n backbone
backbone:
  # [from, repeats, module, args]
  - [-1, 1, Conv, [64, 3, 2]] # 0-P1/2
  - [-1, 1, Conv, [128, 3, 2]] # 1-P2/4
  - [-1, 3, C2f, [128, True]]
  - [-1, 1, Conv, [256, 3, 2]] # 3-P3/8
  - [-1, 6, C2f, [256, True]]
  - [-1, 1, SCDown, [512, 3, 2]] # 5-P4/16
  - [-1, 6, C2f, [512, True]]
  - [-1, 1, SCDown, [1024, 3, 2]] # 7-P5/32
  - [-1, 3, C2f, [1024, True]]
  - [-1, 1, SPPF, [1024, 5]] # 9
  - [-1, 1, PSA, [1024]] # 10
```

Backbone + Model Scale (n)

```yaml
head:
  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 6], 1, Concat, [1]] # cat backbone P4
  - [-1, 3, C2f, [512]] # 13

  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 4], 1, Concat, [1]] # cat backbone P3
  - [-1, 3, C2f, [256]] # 16 (P3/8-small)

  - [-1, 1, Conv, [256, 3, 2]]
  - [[-1, 13], 1, Concat, [1]] # cat head P4
  - [-1, 3, C2f, [512]] # 19 (P4/16-medium)

  - [-1, 1, SCDown, [512, 3, 2]]
  - [[-1, 10], 1, Concat, [1]] # cat head P5
  - [-1, 3, C2fCIB, [1024, True, True]] # 22 (P5/32-large)

  - [[16, 19, 22], 1, v10Detect, [nc]] # Detect(P3, P4, P5)
```

Head

TERIMA KASIH