E9 246 : Advanced Image Processing Final Project

YOLOv5 Object Detection Android App

by

Aakash Sunil Shedsale

SR No.: 22275

Motivation

- Real-time Object Detection: YOLO is known for its ability to perform real-time object detection with impressive speed and accuracy. Traditional or RCNN based object detection algorithms cannot be used in real-time object detection because of low speed.
- Accessibility and Convenience: By deploying YOLO on Android devices, users can perform real-time object detection tasks without relying on internet connectivity or cloud-based services. This enhances accessibility and convenience, especially in scenarios where real-time processing is crucial or where network connectivity is limited.
- On-device Processing: Running object detection algorithms directly on Android devices enables on-device processing, eliminating the need to upload sensitive data to external servers for processing. This enhances privacy and security by keeping data local to the device.

YOLOv5 Training

- Dataset: PASCAL-VOC, # train images: 16551, # val images: 4952, # classes: 20
- Input size: (1,3,640,640); Output size: (1,25200,25)
- Performance on validation set after training for 100 epochs:

Class	Instances	P	R	mAP50	mAP50-95:
all	12032	0.802	0.794	0.827	0.631
aeroplane	285	0.929	0.832	0.898	0.665
bicycle	337	0.92	0.858	0.915	0.7
bird	459	0.826	0.767	0.809	0.583
boat	263	0.669	0.707	0.739	0.5
bottle	469	0.738	0.761	0.806	0.589
bus	213	0.869	0.869	0.9	0.788
car	1201	0.821	0.903	0.917	0.752
cat	358	0.875	0.824	0.858	0.689
chair	756	0.652	0.648	0.678	0.502
cow	244	0.75	0.861	0.855	0.681
diningtable	206	0.738	0.752	0.77	0.574
dog	489	0.852	0.777	0.825	0.653
horse	348	0.897	0.879	0.899	0.689
motorbike	325	0.846	0.828	0.872	0.631
person	4528	0.867	0.821	0.874	0.64
pottedplant	480	0.674	0.565	0.616	0.394
sheep	242	0.759	0.835	0.85	0.668
sofa	239	0.715	0.736	0.753	0.616
train	282	0.857	0.848	0.867	0.657
tvmonitor	308	0.777	0.805	0.831	0.65

Model conversion: from .pt to .ptl

- Required to remove dependence of model from python runtime environment
- Model summary: 157 layers, 7064065 parameters, 15.9 GFLOPs
- Procedure followed for conversion:

Algorithm 1 Trace and Save YOLOv5 Model for Mobile Deployment

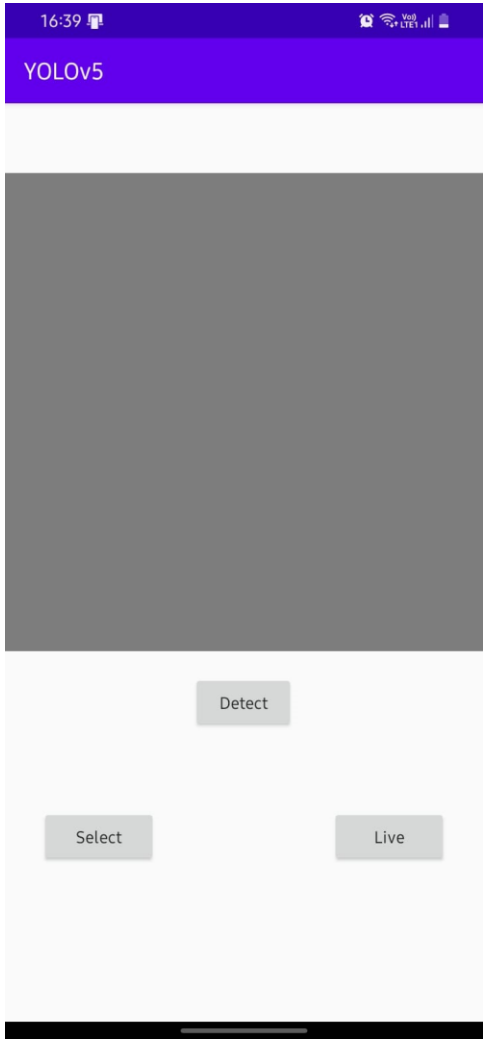
Require: Configuration (config), Example input
Ensure: Traced YOLOv5 model saved to a file for mobile deployment

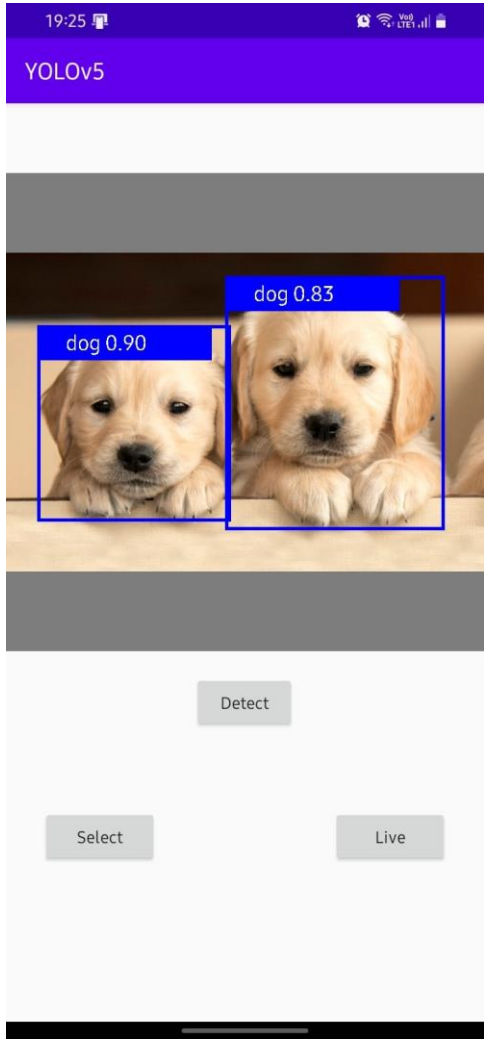
1: **Initialization:** Initialize YOLOv5 model with desired configuration (config) **model** ← YOLOv5(config)
2: **Load Pretrained Model Weights:** Load pretrained weights from a file MODEL.LOAD_STATE_DICT("weighths.pt")
3: **Set Evaluation Mode:** **model.eval()**
4: **Define Example Input:** Define example input of type and size same as the model input **input** ← RANDOM_ARRAY_OF_FLOATS(model.input_size)
5: **Create Trace:** Create a trace of the model using TORCH.JIT.TRACE **traced_model** ← TORCH.JIT.TRACE(model, input)
6: **Optimize For Mobile:** Optimize the traced model for mobile deployment using TORCH.UTILS.MOBILE_OPTIMIZER **optimized_model** ← TORCH.UTILS.MOBILE_OPTIMIZER.OPTIMIZE_FOR_MOBILE(traced_model)
7: **Export mobile interpreter version model:** Export the optimized model compatible with mobile interpreter optimized_model._SAVE_FOR_LITE_INTERPRETER("model.ptl")

- Original model size: 13.7 MB, Exported model size: 27.1 MB

Android app

- App UI:

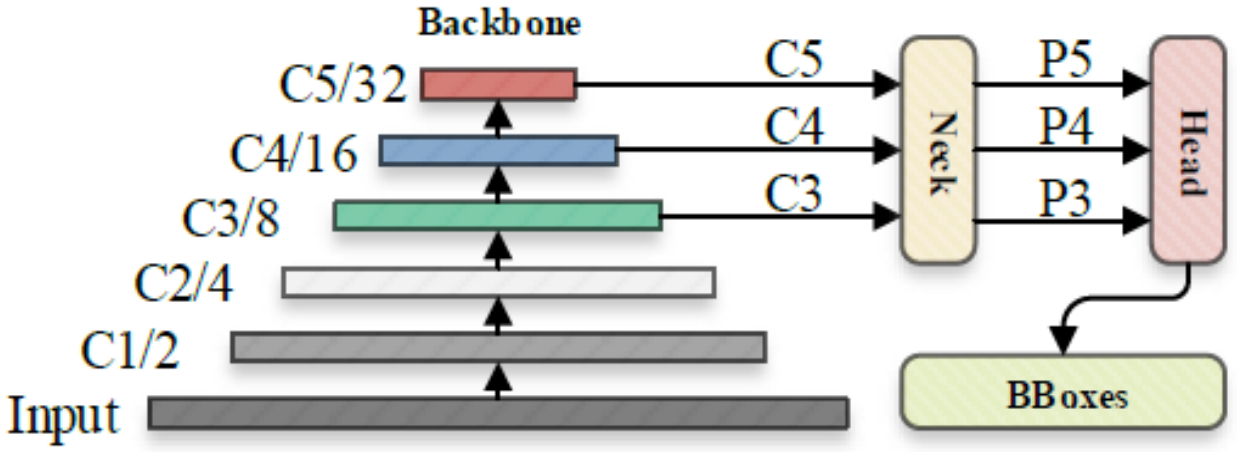




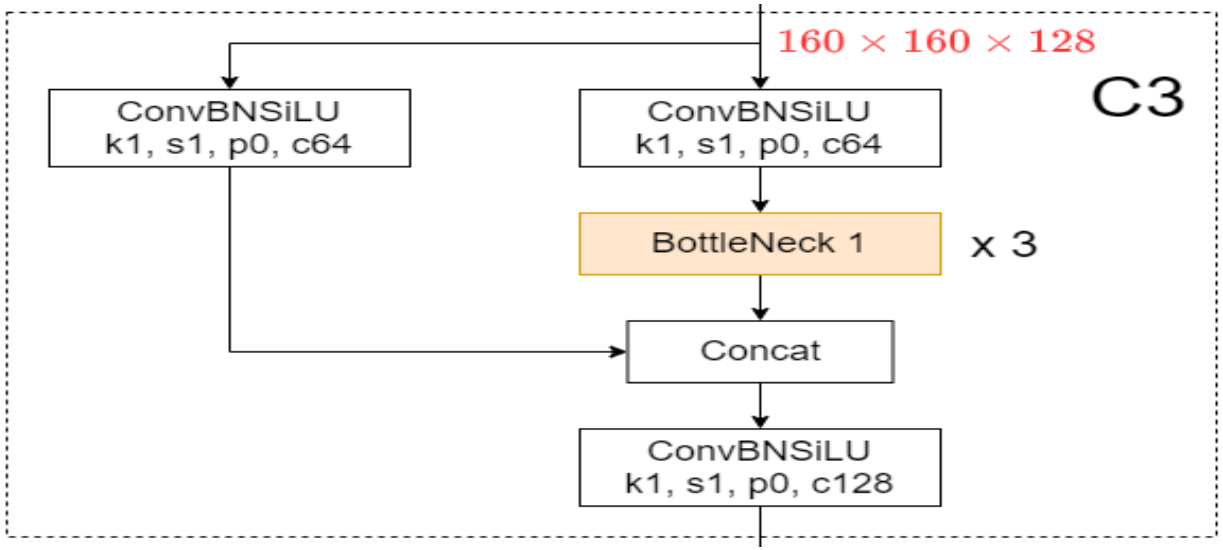
- .apk file size: 225.88 MB
- Link to demo: Here

YOLOv5 network architecture

- Backbone: CSPDarknet53, SPPF
- Neck: FPN and PAN
- Head: Same as YOLOv4 and YOLOv3



Backbone: CSPDarknet53



Cross Stage Partial Network is used to:

i. Strengthen feature propagation

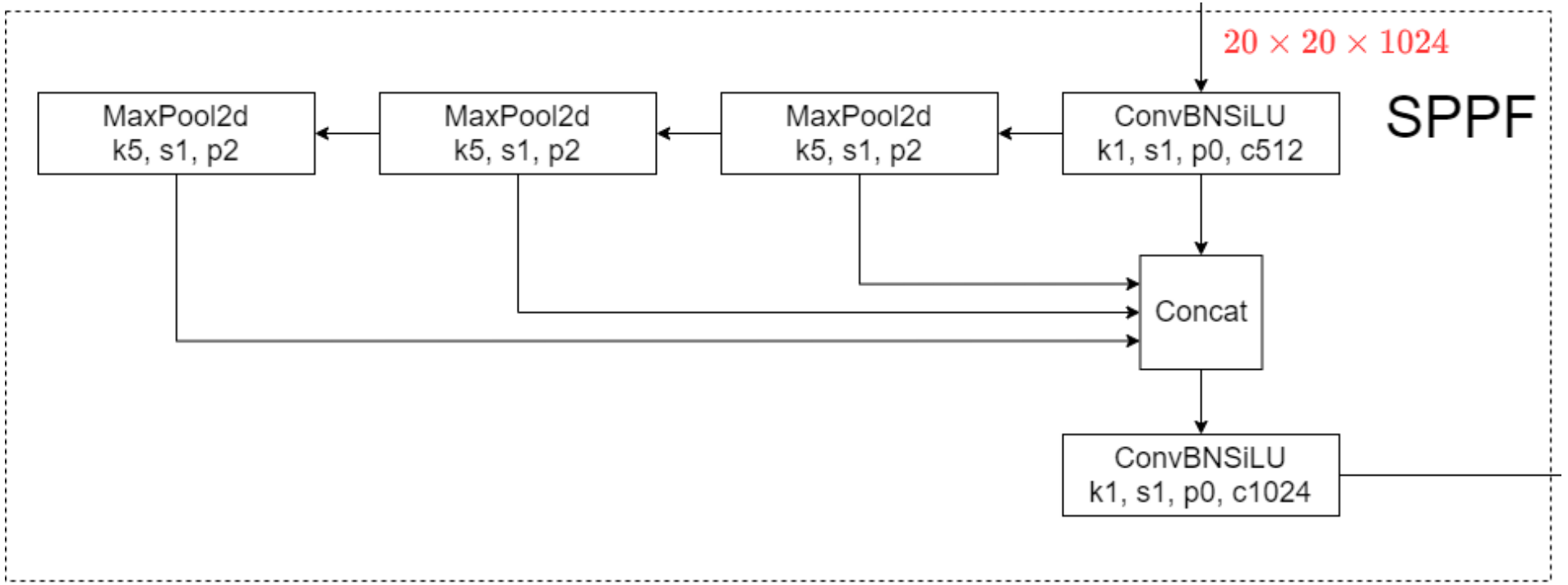
ii. Alleviate the vanishing gradient problem

iii. Encourage the network to reuse features

iv. Reduce the number of network parameters

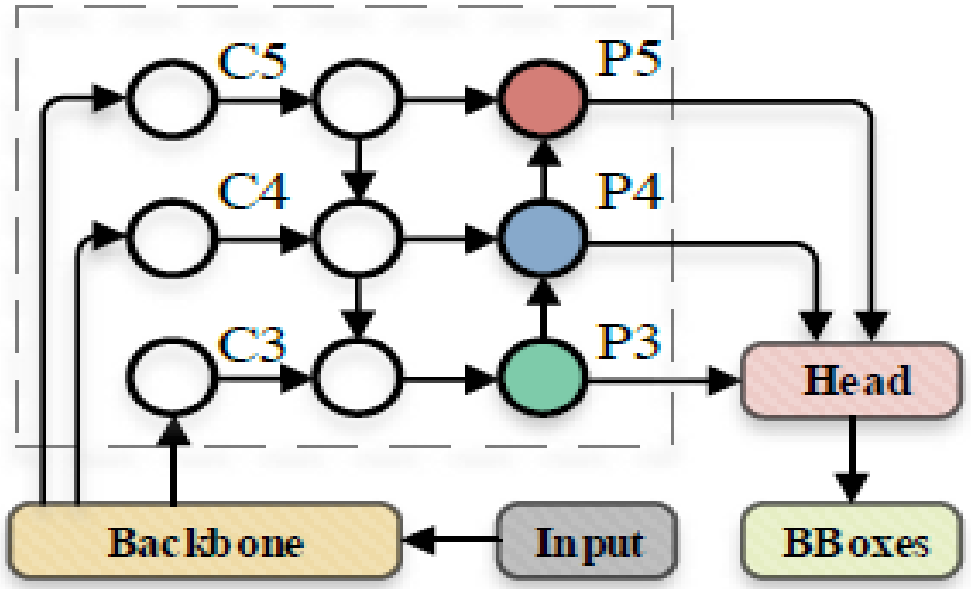
Backbone: SPPF

- Spatial pyramid pooling (SPP): Captures information from different spatial scales. SPPF is faster version of SPP.



Neck: FPN and PAN

- FPN: Feature pyramid network
- PAN: Path aggregation network



Head

- YOLOv5 uses the same head as YOLOv3 and YOLOv4.
- It is composed from three convolution layers and fully connected layers that predict the location of the bounding boxes (x, y, height, width), the confidence scores and the objects classes.
- The equation to compute the target coordinates for the bounding boxes have changed from previous versions

$$b_x = \sigma(t_x) + c_x \qquad b_x = (2 \cdot \sigma(t_x) - 0.5) + c_x$$
$$b_y = \sigma(t_y) + c_y \qquad b_y = (2 \cdot \sigma(t_y) - 0.5) + c_y$$
$$b_w = p_w \cdot e^{t_w} \qquad b_w = p_w \cdot (2 \cdot \sigma(t_w))^2$$
$$b_h = p_h \cdot e^{t_h} \qquad b_h = p_h \cdot (2 \cdot \sigma(t_h))^2$$

(a) (b)

- Output size: 3 anchor boxes per grid cell; 3 different scales: 80x80 (P3:640/8), 40x40 (P4:640/16) and 20x20 (P5:640/32); 3x(80x80 + 40x40 + 20x20) = 25200 bounding boxes; Each bounding box has (x, y, w, h, pc) + # classes predictions

The objectness losses of the three prediction layers(P3, P4, P5) are weighted differently.

Loss function

- Uses BCE Loss for classification and objectness loss; CIoU loss for localization
- Total $Loss = \lambda_1 L_{cls} + \lambda_2 L_{obj} + \lambda_3 L_{loc}$
- The objectness losses of the three prediction layers(P3, P4, P5) are weighted differently $L_{obj} = 4.0 \cdot L_{obj}^{small} + 1.0 \cdot L_{obj}^{medium} + 0.4 \cdot L_{obj}^{large}$

Challenges Faced

- ‘Detect’ function available with YOLOv5 GitHub repo doesn’t accept model in .ptl format hence cannot validate model on VOC dataset
- New to java hence understanding the code taken from GitHub repo to create app and making changes to it was difficult
- YOLOv5 has 10 different release versions (https://github.com/ultralytics/yolov5/releases); When official ultralytics library (provides latest release version) is used to train YOLOv5 model and deploy it in Android app, many errors occurred; Android app needs release version v6.2
- Bug in code taken from GitHub repo to build Android app: Sorts detected confidence scores in ascending order and preforms non-max suppression which is wrong; code modified to sort detected confidence scores in descending order
- Needed to change versions of few imported android libraries to make code taken from GitHub repo work
- YOLOv5 has no official paper hence understanding it required going through different blogs and articles

References

- Code to build Android app is taken from: https://github.com/pytorch/android-demo-app/tree/master/ObjectDetection
- Blogs and articles to understand YOLOv5:

i. https://blog.roboflow.com/yolov5-improvements-and-evaluation/

ii. https://iq.opengenus.org/yolov5/

iii. Liu, Haiying, et al. "SF-YOLOv5: A lightweight small object detection algorithm based on improved feature fusion mode." *Sensors* 22.15 (2022): 5817.

- Blog to know PyTorch model conversion process for mobile deployment: https://medium.com/@adrian.errea.lopez/from-pytorch-model-to-mobile-application-50bc5729ed83

Thank you