YOLO-ReT: Towards High Accuracy Real-time Object Detection on Edge GPUs

Prakhar Ganesh¹*, Yao Chen¹*, Yin Yang², Deming Chen¹,³, Marianne Winslett¹,³

¹ Advanced Digital Sciences Center, Illinois at Singapore
² College of Science and Engineering, Hamad Bin Khalifa University, Qatar
³ University of Illinois at Urbana-Champaign, USA
Background & Motivation

- Transfer Learning in Object Detection
- Multi-Scale Feature Interaction
Transfer Learning in Object Detection

Feature Extraction Backbone

Rapidly Increasing channel number in the last few layers

Classification Head (Fully Connected Layers)

Image Net

COCO/VOC

Feature Extraction Backbone

Detection Head
What if we had randomly initialised the **last layer**?
Transfer Learning in Object Detection

What if we had randomly initialised the last three layers?
Transfer Learning in Object Detection

What if we had used **no transfer learning at all?**

Feature Extraction Backbone

Detection Head

COCO/VOC
Transfer Learning in Object Detection

- Transfer learning plays an important role in model training, specially in a low data setting.
- Not every layer of a pre-trained model is equally useful. Initial layers are known to be task-agnostic, and last layers can be task-specific.
- Despite the existence of active research in transfer learning, most SOTA models in vision have not adapted to this behavior.
Multi-Scale Feature Interaction

Feature Extraction Backbone

Multi-Scale Feature Interaction

Detection Head

Segmentation Head

Tracking Head

... and many more
Multi-Scale Feature Interaction

FPN  PANet  NAS-FPN  BiFPN

repeated blocks

Tan, Mingxing, Ruoming Pang, and Quoc V. Le. "EfficientDet: Scalable and efficient object detection." CVPR. 2020.
Multi-Scale Feature Interaction

- Existing work focuses on some combination of top-down and/or bottom-up approaches.
- With the increasing complexity of these modules, the tradeoff between accuracy and efficiency has started saturating.
- NAS-based architectures have revealed the importance of direct connections between non-adjacent feature scales.

Ghiasi, Golnaz, Tsung-Yi Lin, and Quoc V. Le. "NAS-FPN: Learning scalable feature pyramid architecture for object detection." CVPR. 2019.
YOLO-\text{ReT}

- Importance of Individual Layers
- Backbone Truncation
- Raw Feature Collection and Redistribution
The diagram illustrates a complete model architecture. It begins with a **Truncated Backbone** that includes truncated last CNN layers, with features labeled as $I/4$, $I/8$, $I/16$, and $I/32$.

Following the backbone, a **Raw Feature Collection and Redistribution Module** is shown, which involves operations such as **MaxPool**, **Upsample**, **Pointwise Conv**, **MBConv Block with 5x5 kernel**, and **Weighted Sum**. The module processes these features and redistributes them, preparing them for further processing.

Finally, the **Detection Head** is depicted, which involves operations like **Detection Neck** (E.g., FPN, PANet, BiFPN etc.) and output layers. The head outputs final detections, as indicated by the final outputs arrow.
Complete Model

Truncated Backbone

Raw Feature Collection and Redistribution Module

Detection Head

Truncated Last CNN Layers

I/4

I/8

I/16

I/32

P

P

P

P

P

MB

5x5

C

C

C

C

= MaxPool

= Upsample

= Concat

= Weighted Sum

= Pointwise Conv

= MBConv Block with 5x5 kernel

Detection Neck’ (Eg., FPN, PANet, BiFPN etc.)
Importance of Individual Layers
Backbone Truncation

- Initializing the last layers of the feature extraction backbone with transfer learning weights actually 'hurts' the performance.
- Since these last layers hold no transfer learning importance, they can be analysed purely from an architecture viewpoint.
- We propose that a truncated version of the feature extraction backbone is a better alternative to width reduction.
Complete Model

Truncated Backbone

Truncated Last CNN Layers

\[ \text{I/4} \quad \text{I/8} \quad \text{I/16} \quad \text{I/32} \]

Raw Feature Collection and Redistribution Module

\[ \text{P} \quad \text{P} \quad \text{P} \quad \text{P} \] +

\[ \text{MB} \]

5x5

\[ \text{C} \quad \text{C} \quad \text{C} \]

Detection Neck (Eg., FPN, PANet, BiFPN etc.)

Output Layers

Final Outputs

\[ \text{down} = \text{MaxPool} \quad \text{up} = \text{Upsample} \quad \text{C} = \text{Concat} \quad + = \text{Weighted Sum} \]

\[ \text{P} = \text{Pointwise Conv} \quad \text{MB} = \text{MBConv Block with 5x5 kernel} \]
Raw Feature Collection and Redistribution
Raw Feature Collection and Redistribution

Simplistic Design

Minimal Network Fragmentation
(Each Collection and Redistribution Path Can be Executed in Parallel)
Raw Feature Collection and Redistribution

Direct Connection Paths Even Between Non-Adjacent Scales
Raw Feature Collection and Redistribution

Independent of the number of Output Scales
Raw Feature Collection and Redistribution

Some Other Multi-Scale Feature Interaction

Cannot replace the meticulousness provided by other Multi-Scale Feature Interaction methods

But can be easily integrated as an additional feature processing
Evaluation

- Experiment Setup
- Component Ablation
- State of the art Models
Experiment Setup

- We tested with 3 lightweight feature extraction backbones (MobileNetV2x0.75, x1.4, and EfficientNet-B3) and various feature interaction methods (FPN, PANet and BiFPN).
- We evaluated our methods on Pascal VOC and COCO datasets.
- We tested our models with on-device performance latencies, on Jetson Nano, Jetson Xavier NX and Jetson AGX Xavier.
Qualitative Heatmap Study

(a) Multi-Scale Raw Features

(b) Multi-Scale Raw Features passed through our RFCR Module

(c) Multi-Scale Raw Features passed through our RFCR module and PANet

(d) Multi-Scale Raw Features passed directly through PANet without the RFCR Module
Qualitative Heatmap Study

(a) Multi-Scale Raw Features

(b) Multi-Scale Raw Features passed through our RFCR Module

(c) Multi-Scale Raw Features passed through our RFCR module and PANet

(d) Multi-Scale Raw Features passed directly through PANet without the RFCR Module
Qualitative Heatmap Study

(a) Multi-Scale Raw Features

(b) Multi-Scale Raw Features passed through our RFCR Module

(c) Multi-Scale Raw Features passed through our RFCR module and PANet

(d) Multi-Scale Raw Features passed directly through PANet without the RFCR Module
Qualitative Heatmap Study

(a) Multi-Scale Raw Features

(b) Multi-Scale Raw Features passed through our RFCR Module

(c) Multi-Scale Raw Features passed through our RFCR module and PANet

(d) Multi-Scale Raw Features passed directly through PANet without the RFCR Module
## State of the art Models

| Model                     | Input Resolution | FPS   | AP50 (Detailed Results in Paper) |
|---------------------------|------------------|-------|----------------------------------|
|                           |                  | Nano  | NX     | AGX    | VOC    | COCO   |
| Tiny-YOLOv3               | 416              | 27.36 | 66.55  | 91.71  | 61.30  | 33.10  |
| Tinier-YOLO               | 416              | 30.14 | 68.73  | 92.09  | 65.70  | 34.00  |
| YOLO-ReT-MobileNetV2 x 0.75 | 320             | 33.19 | 71.64  | 95.97  | 68.75  | 34.91  |
| YOLO Nano                 | 416              | 13.62 | 54.03  | 85.81  | 69.10  | --     |
| YOLO-ReT-MobileNetV2 x 1.4 | 320             | 23.01 | 65.37  | 93.49  | 70.35  | 35.77  |
| YOLO Fastest              | 320              | 42.41 | 76.13  | 126.82 | 61.02  | --     |
| YOLO-ReT-MobileNetV2 x 1.4 | 224             | 43.16 | 84.32  | 113.94 | 62.91  | 31.63  |
## State of the art Models

| Model                   | Input Resolution | FPS         | AP<sup>50</sup> (Detailed Results in Paper) |
|-------------------------|------------------|-------------|---------------------------------------------|
|                         |                  | Nano | NX | AGX | VOC | COCO |
| Tiny-YOLOv3             | 416              | 27.36 | 66.55 | 91.71 | 61.30 | 33.10 |
| Tinier-YOLO             | 416              | 30.14 | 68.73 | 92.09 | 65.70 | 34.00 |
| YOLO-ReT-MobileNetV2 x 0.75 | 320          | **33.19** | 71.64 | **95.97** | **68.75** | **34.91** |
| YOLO Nano               | 416              | 13.62 | 54.03 | 85.81 | 69.10 | --    |
| YOLO-ReT-MobileNetV2 x 1.4 | 320         | 23.01 | 65.37 | 93.49 | 70.35 | 35.77 |
| YOLO Fastest            | 320              | 42.41 | 76.13 | **126.82** | 61.02 | --    |
| YOLO-ReT-MobileNetV2 x 1.4 | 224        | **43.16** | 84.32 | 113.94 | **62.91** | **31.63** |

Code available at: github.com/prakharg24/yoloret
Thank You