Deployment and optimization pipeline for YOLOv5 models on detection and segmentation tasks

Yiyang (Fred) Shi
Department of Electrical and Computer Engineering
University of Toronto
Toronto, Canada
fred.shi@mail.utoronto.ca

Chengqi (William) Li
Department of Electrical and Computer Engineering
University of Toronto
Toronto, Canada
chengqi.li@mail.utoronto.ca

***Abstract*—Convolutional neural networks explores dense and rich visual information from cameras and are frequently used in Robotics to provide perception information for many time and safety critical tasks. Due to the real time operating nature of robotics applications, the model needs to perform inference in a compact and time-efficient manner. This project proposed and implemented a 2-stage deployment and optimization pipeline that utilizes knowledge distillation and quantization to reduce the model inference time without severely sacrificing accuracy. We performed verification individually on the 2 stages and compared the inference speed and accuracy metrics against the original models across different model architectures and objectives. Our proposed pipeline is able to achieve significant inference speed reduction while maintaining the model accuracy on a specific target hardware device, helping the model to compute more efficiently and require less resources.**

I. Introduction

Convolutional neural networks have been a popular area of research in recent years and many lightweight models are developed and used in robotics applications. The majority of models focus on utilizing dense and rich visual information completed by multiple cameras with different Field of Views and at different positions. YOLO-based object detector [1], [2] is one of the state-of-the-art 2D object detection models, which utilizes grids and anchors for a single-stage detection. Due to its compact nature, YOLO detector has been adapted for various robotics usages. However, the resources onboard are still limited and many time and safety critical tasks are dependent on the perception network outputs. Take the example of an autonomous vehicle, the computation tasks are divided into environment mappings, perception, motion planning and control along with system supervision, and the dedicated resources for each task are constrained. To be used in downstream simultaneous localization and mapping as well as behavior planning tasks, the perception information needs to be generated within a defined time frame and confined to the real-time system onboard strictly.

One way to generate inference data is to decentralize computation and upload/download vision information to a cloud computing cluster, which has plenty of resources. However, this approach is not ideal for time and safety critical tasks, as latency and network signals can vary from point to point. Also, the method is prone to data privacy issues and requires encryption for sensitive data.

Therefore, we would like to utilize and optimize for the resources available and propose a 2-stage deployment and optimization pipeline, which reduces inference latency and increase throughput without severely scarifying the model accuracy.

For this project, our work is summarized as follows:

1) We developed a model optimization pipeline consisting of 2 vertically stacked parts, which are knowledge distillation (KD) and quantization. They can be used individually or combined together.
2) We trained and fine tuned the YOLOv5 model on both object detection and semantic segmentation tasks using the Cityscapes dataset [3] with 10 selected object classes.
3) We verified the 2 stages individually and compared the performance of models with and without the optimization pipeline.

II. Related Work

A. Knowledge distillation

Knowledge distillation (KD) was first introduced by Hinton in the paper [4] as a model compression method to transfer the generalization ability of a cumbersome teacher model to a small student model for classification tasks. The student model in KD is trained with the combination of the ground truth label and the distilled knowledge from the teacher model. This provides a viable solution for resource-constrained hardware implementations of deep neural networks. The knowledge is transferred from the teacher model to the student by minimizing a loss function, aimed at matching softened teacher logits as well as ground-truth labels. This concept served as the fundamentals and inspired distillation implementation for other tasks.

This project focuses on the adaptation of the paper "Distilling Object Detectors with Fine-grained Feature Imitation" [5], which extends knowledge distillation to object detection tasks. Unlike classification tasks, the bounding box and object detection output can not be directly distilled. Instead, the paper focus on matching fine-grained feature maps from the last

layer of the detection model filtered by a fine-grained imitation mask to distill knowledge at near object locations. The mask is generated based on the ground truth bounding box and predefined anchor locations to outline object of interests along with features nearby.

B. Quantization

Quantization is the process of reducing the precision of the weights, biases, and activations such that they consume less memory [6]. Often quantization can be applied to a neural network with 32 bits float points representation and convert it to a lower bits representation to reduce model size and conserve memory. With lower bits representation, integer operations can be used over floating point ones and the same forward function would require fewer computations, therefore, leads to quicker inference time. Typically 8-bit integer (int8) is used, as many hardware optimization methods are available for this data type. However, information are lost and quantization-related errors will accumulate during this process. The lower precision datatype will result in less dynamic range and resolution and ultimately affect model precision.

Two types are quantization are available [7]. Post training quantization does not alter the training process or require training data, it applies quantization directly on trained model. Many modern frameworks like OpenVINO [8] balances the trade-off between compression and precision by taking in a subset of model dataset to calibrate the quantized parameters after initial quantization effort. Meanwhile, quantization aware training optimizes a model first and uses original training dataset to fine-tune the model to restore accuracy.

III. Method

A. Deployment and optimization pipeline overview

Inspired by related works in the previous section, we would like to propose a 2-stage deployment and optimization pipeline that contains 3 parts in total, which are Knowledge distillation training, Evaluation & filter and final Quantization for deployment. The pipeline should reduce the model inference time while maintaining the test precision and flexibility for different model architecture supports. The inputs for the pipelines are a pre-trained large cumbersome teacher model and a lightweight student model with optional student model checkpoints. After optimization, the model should output a best-performing, quantized model in Int8 precision format that is ready for deployment.

In this pipeline, we use knowledge distillation as a compression technique to "capture" and "distill" the knowledge from the teacher model to the student model which is easier to deploy. It modifies the existing training process and train the student model based on a combination of ground truth labels and teacher predictions (imitation loss from teacher feature map). As knowledge distillation is sensitive to hyper-parameters and could not guarantee the best results, the distilled student model output is compared against a pre-trained student model without KD for evaluation on defined metrics. The best performing model is then passed to the post-training quantization module for further compression and optimization for target hardware. Noticeably, knowledge distillation and quantization are two orthogonal processes and can be used individually or combined together.

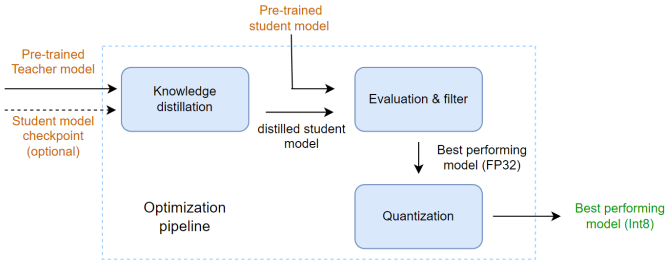


Fig. 1. Proposed deployment and optimization pipeline

B. Knowledge Distillation on object detection

We adopt the method of knowledge distillation in the object detection task from [5]. Knowledge distillation for object detection captures the valid knowledge of the object of interest with a fine-grained feature mask. The mask is generated according to 1, the feature map is divided into $g \times g$ grids, and for each grid, n anchors are initialized. Then, it calculates IOU for all initial anchors and ground truth bounding boxes, using $\Psi * Maximum\_IOU$ as the threshold to generate fine-grained imitation masks im. As shown in figure 2, the red and green bounding boxes are the anchor boxes near the ground truth object bounding box, where the red bounding boxes have the anchor with the largest IOU with the ground truth.

To handle the unmatched dimensions of the feature map from student model and teacher model, an extra 2D convolution feature adaptation layer is applied to the feature map from the student model. As shown in figure 3, the feature map of the student model passed through the feature adaptation layer masked by the imitation mask is used to calculate the imitation loss of the student model.

With feature adaptation layer f_{adap}, student model feature map s, teacher model feature map t and imitation mask Im, the imitation loss $L_{imitation}$ can be expressed as:

$$L_{imitation} = \frac{1}{2N_p} \sum_{i=1}^{W} \sum_{j=1}^{H} \sum_{c=1}^{C} Im_{ij} (f_{\text{adap}}(s)_{ijc} - t_{ijc})^2,$$
$$\text{where } N_p = \sum_{i=1}^{W} \sum_{j=1}^{H} Im_{ij}. \tag{1}$$

The overall loss function for the student model is:

$$L = L_{gt} + \lambda L_{imitation} \tag{2}$$

C. Quantization

Quantization maps the high resolution FP32 weights and activation function parameters in a model to lower bits Int8 representation to conserve memory and reduce computation

Algorithm 1: Imitation mask generation

input : Ground truth bounding boxes gt of size $b \times k \times 4$, $[batch, max(gt_{bbox}), [x_{min}, x_{max}, y_{min}, y_{max}]]$
input : Feature map f of size $b \times w \times h$, $w = h$
input : IOU hyper-parameter $\Psi \in [0, 1]$
input : Grid size $g \times g$
input : n Initial anchors for each grid
output: Imitation mask Im of size $b \times w \times h$, $w = h$ initialize with 0.

```
for i ← 1 to b do
    generate anchors in a batch with initial anchors and grid size;
    anchors ← GenerateAnchors(n, g);
    find IOU matrix of shape [g × g × n, k] between all anchors and all ground truth bbox;
    IOU_matrix ← FindIOU(anchors, gt);
    max_IOU ← Max(IOU_matrix);
    for iou ← IOU_matrix do
        if iou > Ψ * max_IOU then
            correspond area in Im[i] = 1
        else
            pass
        end
    end
end
```

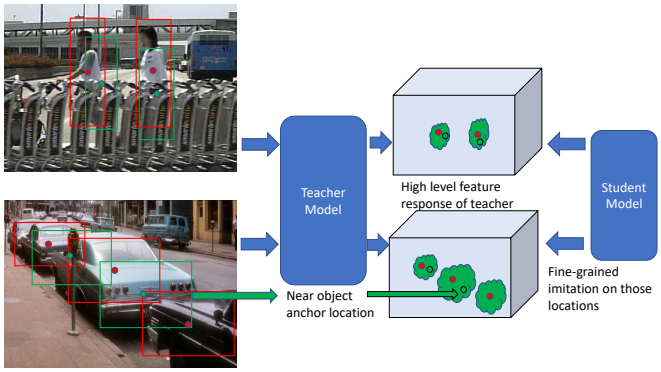


Fig. 2. Knowledge distillation with feature imitation [5]

efforts. The process is applied to the 2D convolution and fully-connected layers which contribute to majority of model floating point operations (FLOPS). Quantization can be applied per tensor or per channel with different slicing methods [6], only the tensor slicing is considered for this project. The symmetric quantization formula calculates input low and input high for a specific layer weights or activation function based on a scale parameter and 8 bits integer range. The floating-point zero is mapped directly to integer zero.

$$input_{Low} = scale * \frac{level_{Low}}{level_{High}} \tag{3}$$

$$input_{High} = scale \tag{4}$$

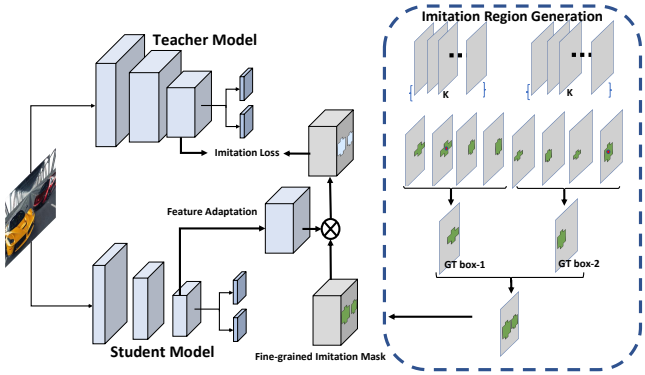


Fig. 3. Imitation mask generation and imitation loss via feature adaptation

TABLE I
LEVEL PARAMETERS FOR INT8 QUANTIZATION

Usage	Levels	$Level_{High}$	$Level_{Low}$
Weights	255	$2^7 - 1$	$-2^7 + 1$
Unsigned activation	256	$2^8 - 1$	0
Signed activation	256	$2^7 - 1$	$-2^7 + 1$

$$s = \frac{levels - 1}{input_{High} - input_{Low}} \tag{5}$$

The calculated $input_{Low}$ and $input_{High}$ in 3 and 4 are served as the quantization range that post process parameters will fall in. The pseudo scaling factor s is calculated as the ratio between the desired datatype levels and quantization range. For Int8 quantization, the bit levels value is set as for either 255 or 256 depending on the usages. The calculated $input_{Low}$ and $input_{High}$ are used to clamp the original layer parameters and perform quantization shown as follow.

$$\begin{aligned} clamp(input; input_{Low}; input_{High}) = \\ min(max(input, input_{Low}), input_{High})) \end{aligned} \tag{6}$$

$$output = \frac{round(clamp(input) * s)}{s} + input_{Low} \tag{7}$$

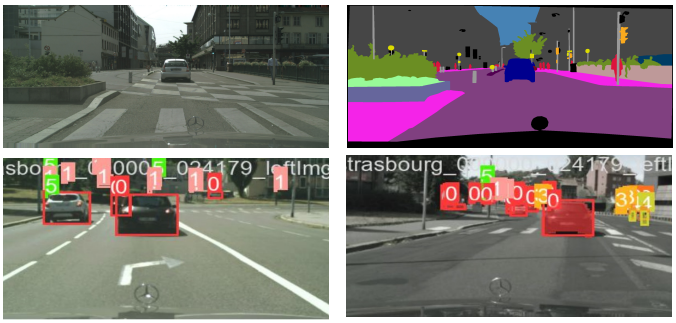
The pseudo scaling factor s is used along with a round operation to ensure that quantized inputs can be represented by the desired datatype. Quantization is not a lossless process as precision is compromised during the clamp operation. By tuning the scale parameter, we can achieve different levels of granularity and control the overall quantization loss. Different bit levels value and level ranges are used for weights and activation functions, details are shown in table I.

After initial quantization, a calibration dataset is used to collect statistics and fine tune quantization parameters (adjusting the scaling parameter), additional procedures like bias correction and channel alignment are not included in this paper and can be seen from OpenVINO documentation [8].

IV. Experiments

A. Model and dataset selection

For the experiment setup, we selected the one-stage state-of-the-art YOLOv5 model for both object detection and segmentation tasks and utilized the provided training and validation pipeline from [9].



Object *detection* labels | Object *semantic segmentation* masks

Fig. 4. Convert semantic segmentation masks to bounding boxes with masks

To simulate a perception application of driving and road scenarios, we replaced the original COCO dataset [10] with fine-grained Cityscapes semantic segmentation dataset, where detailed masks for all objects are recorded in various weather conditions and at different cities in Europe. In total, 2975 images from training set and 500 images from validation set are selected to provide comprehensive coverage and generalization of the road scenes. Among the 30 annotated semantic classes defined in CityScape dataset, we picked 10 most representative ones from vehicle, pedestrian and object categories. Noticeably, the numbers of instances per class in the training set are severely imbalanced with a heavy focus on cars, traffic signs and person as shown below in Figure 6.

Fig. 5. Mosaic augmentation

The ground truth semantic segmentation masks for the Cityscapes dataset are converted to 2D bounding boxes for object detection tasks using the maximum and minimum values along the x-axis and y-axis. For the segmentation task, the same bounding box computed for OD along with the original segmentation mask is used. The training set is augmented with scaling, rotation, color space adjustment. The

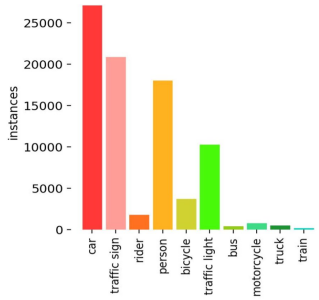


Fig. 6. Occurrence of instances per class

TABLE II
Model complexity between student and teacher

Model	Number of Parameters	FLOPS
YOLOv5x (teacher)	86,278,375	$2^7 - 1$
YOLOv5s (student)	7,046,599	$2^8 - 1$

Mosaic augmentation (mixing 4 training images with random scaling) shown in figure 5 to further enhance the training process and prevent over-fitting to the training data.

To prepare the models for our distillation and optimization pipeline, a cumbersome teacher of YOLOv5 extra large model and a relatively light weight student YOLOv5 small model are trained from scratch for 100 epochs with input image size of 640 x 640 pixels. To simulate the resource limited setup, we select the inference device as an Intel i7-8700 CPU, while training is performed on a GPU cluster service. The inference time and accuracy for both models are recorded on the CPU hardware establish baselines. The 2 models share the same architecture while different in network depth and width II. The selected extra large model has layer depth multiple of 4 and width multiple of 2.5 compared to student model resulting in better generalization capability in terms of mean average precision at the sacrifice of computational complexity.

B. Knowledge Distillation

For the teacher model, a YOLOv5 extra large model is trained on the selected Cityscapes dataset for 100 epochs. For the student model, a YOLOv5s model is trained with both ground truth labels and teacher model YOLOv5x with feature imitation.

As shown in figure 7, the imitation feature is selected to be the feature layer with the highest resolution. For both teacher and student models, the 16th layer is chosen for feature imitation.

In our experiment, the IOU hyper-parameter Ψ is set to 0.5 the λ in the loss function 2 is set to 0.01. Both YOLOv5x and YOLOv5s models are trained on the Cityscapes dataset with for 100 epochs as baseline results III and knowledge distillation model on a single GPU RTX A2000 for 300 epochs for comparison IV. Knowledge distillation on segementation tasksa is not implemented in this project, but it can accom-

TABLE III
MAP BASELINE AND COMPARISON FOR OBJECT DETECTION DISTILLATION

Model	mAP_{50}	mAP_{50-95}
Yolov5x.pt ∼100	0.523	0.311
Yolov5s.pt (from scratch) ∼100	0.418	0.223
Yolov5s.pt (KD) ∼100	0.377	0.198
Yolov5s.pt (KD) ∼200	0.422 (+0.004)	0.224 (+0.001)

TABLE IV
MAP FOR OBJECT DETECTION DISTILLATION

Class	all	car	traffic sign	rider	person	bicycle	traffic light	bus	motorcycle	truck	train
mAP_{50} Before KD	0.418	0.697	0.374	0.444	0.481	0.395	0.367	0.501	0.292	0.3	0.333
mAP_{50-95} Before KD	0.223	0.445	0.184	0.23	0.229	0.177	0.144	0.36	0.104	0.202	0.154
mAP_{50} After KD	0.422	0.707	0.382	0.441	0.479	0.38	0.358	0.455	0.295	**0.351**	0.334
mAP_{50-95} After KD	0.224	0.457	0.196	0.229	0.232	0.178	0.144	0.319	0.106	**0.243**	0.14

TABLE V
MAP AND INFERENCE LATENCY COMPARISON FOR OBJECT DETECTION TASK

Model	Average Inference speed (ms)	Size (MB)	mAP_{50}	mAP_{50-95}
YOLOv5x.onnx	620.7	329.0	0.523	0.311
YOLOv5x default quantization	291.2	84.1	0.527(+0.004)	0.307(-0.004)
YOLOv5x accuracy aware quantization	307.4	84.0	0.519(-0.004)	0.306(-0.005)
YOLOv5s.onnx	70.1	27.0	0.425	0.228
YOLOv5s default quantization	36.1	7.7	0.418(-0.007)	0.224(-0.004)
YOLOv5s accuracy aware quantization	38.5	84.0	0.416(-0.009)	0.219(-0.008)

plished in a similar way with the imitation loss computed from the 16th layer feature map.

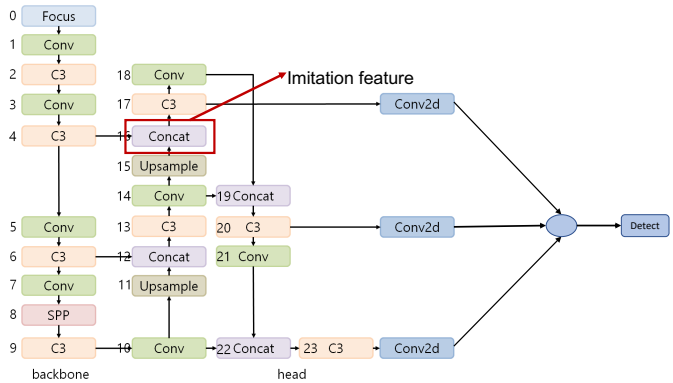


Fig. 7. The 16th layer is chosen for feature imitation

C. Optimization Details

The trained YOLOv5 small and extra large models for object detection and segmentation are first converted to ONNX model format for inference purpose. The mean average precision (mAP) across 10 classes and inference speed tested on the Intel i7-8700 CPU are recorded as the baseline. The models are then passed through OpenVINO model optimizer to perform various optimization techniques such as linear operation fusing and grouped convolution fusing. The OpenVINO framework provides an intermediate representation format (IR) and model is optimized for Intel specific hardware (i7-8700 CPU for our usage).

The default quantization and accuracy aware quantization methods are explored, and the original validation set from Cityscapes is used for calibration. The mAP from 0 to 0.5 (mAP_{50}) bounding box IOU and mAP from IOU of 50 percent to 95 percent with 5 percent increments(mAP_{50-95}) are used as the object detection models' calibration metric, while the mAP_{50} and mAP_{50-95} mask IOU are used for the segmentation models during quantization. We performed both default quantization and accuracy aware quantization methods with random 300 images selected from the validation set. The default quantization method prepares the layer for quantization by aligning ranges of Convolution layer output activation,

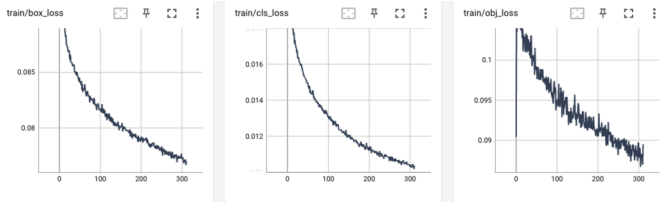


Fig. 8. loss/epochs during training

then it inserts a "Fake quantization" operation before each layer and tune the scaling parameter based on calibration dataset. Lastly, it adjusts the biases of Convolution and fully connected layer to minimize quantization error. The accuracy aware quantization method extends the default algorithm and computes the mismatch in target accuracy metric before initial quantization. It creates a ranking subset and reverts the layers that cause majority of mismatches in a sequential order until new accuracy drop is within the defined range. The quantization results are shown in table V for both methods with input image size of 640 x 640 pixels. The accuracy drop (mAP score) for accuracy aware training is set to 0.01. Smaller thresholds are also explored, but they will not lead to convergence.

D. Results and analysis

1) Knowledge Distillation: The knowledge distillation network was trained with 0.01 imitation weight, 0.05 bounding box weight and 0.5 for object class cross entropy loss weight for 300 epochs, the batch size is set as 16, hyper-parameters can be found in VI. With a pre-trained YOLOv5x model, the imitation loss is calculated with the inference result from the teacher model, the overall training time is 2x of the single model training pipeline.

TABLE VI
HYPER-PARAMETERS IN KNOWLEDGE DISTILLATION NETWORK TRAINING

learning rate	0.01
momentum:	0.937
weight decay:	0.0005
box loss weight:	0.05
class CE loss weight:	0.5
epochs:	300
IOU thresholding hyperparameter:	0.5

After training for 300 epochs, comparing with the YOLOv5s object detection model trained from scratch, the overall highest mAP_{50} and mAP_{50-95} of knowledge distillation student model YOLOv5s has negligible improvement on the Cityscapes validation set, the mAP of the class truck has increased for 5% and 4% on mAP_{50} and mAP_{50-95} respectively. Inspired by the implementation on Faster RCNN in the original paper, we select 5 anchors for each of the grids' in the 16^{th} layer feature map in the YOLOv5s model with 40×40 grid size. In the training of the baseline object detection model, the anchor size is determined with k-means using all ground truth bounding boxes in the training dataset. For further improvement, a fine tune of anchor boxes on the feature map is needed. Also, multiple feature layer (19^{th} and 22^{th}) imitation can be added for more fine-grained distillation.

2) Quantization: After comparing the student with KD mAP results against the one trained from scratch, we see no significant improvements. Therefore, the pre-trained student model is passed to the quantization block for further optimization. To verify the quantization performance and generalization capabilities, we optimized all available student and teacher models. From table V, we observe that quantization significantly reduces the model inference time across models (v5x and v5s) and across detection tasks (object detection and segmentation). Segmentation results can be found in Appendix. The quantized model inference times are reduced on average by about 48 percent while suffering a minimum performance drop. The mAP_{50} and mAP_{50-95} both dropped for less than 1 percent compared to the original models. The default quantization method can already perform well with a set of 300 calibration images, while the accuracy aware quantization performs slightly worse than the default method. This observation is uniform across models and tasks. As the accuracy aware quantization uses default method as intermediate archive, it performs further accuracy metrics analysis and removes quantization layers and repeats the calibration process until the metric requirements are satisfied. For object detection YOLOv5x models, the accuracy-aware quantized model takes 16.2 ms longer per image for inference time compared to the default one as some of the "FakeQuanization" layers are removed to satisfy accuracy metrics. The accuracy drops are within the defined 1 percent maximum drop range, but greater than default quantization drops. This can be caused by error accumulation within the model during the calibration process and calibration step size causing the parameters to traverse further away from local optima. In addition, the parameters produced by default quantization is already calibrated on the selected 300 images, further calibration performed on the same image set might be lead to worse generalization capability as it is over-fitting the quantized parameter to the specific dataset. Overall, quantization results are consistent and it greatly reduces model inference time without sacrificing the model accuracy.

V. CONCLUSION

This report proposed a 2-stage deployment and optimization pipeline that reduces the model inference speed without severely sacrificing model accuracy. The pipeline uses knowledge distillation to train/fine-tune the student model first and then uses quantization for further optimization for inference performance. We performed verification individually between the 2 stages and compared the accuracy metrics and inference speed between the optimized models and original models. We conclude that the proposed 2-stage deployment and optimization pipeline can reduce inference speed significantly while maintaining flexibility for adaptation across model architectures and objectives (object detection and segmentation).

REFERENCES

[1] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020. 1

[2] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018. 1

[3] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[4] G. Hinton, O. Vinyals, J. Dean, *et al.*, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015. 1

[5] T. Wang, L. Yuan, X. Zhang, and J. Feng, "Distilling object detectors with fine-grained feature imitation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4933–4942, 2019. 1, 2, 3

[6] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 2, 3

[7] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," *Low-Power Computer Vision*, p. 291–326, 2022. 2

[8] "Quantizing models post-training¶." 2, 3

[9] G. Jocher, "YOLOv5 by Ultralytics," 5 2020. 4

[10] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014. 4

APPENDIX

TABLE VII
MAP AND INFERENCE LATENCY COMPARISON FOR SEMANTIC SEGMENTATION TASK

Model	Average Inference speed (ms)	Size (MB)	mAP_{50}	mAP_{50-95}
YOLOv5x.onnx	700.5	337.0	0.38	0.17
YOLOv5x default quantization	291.2	86.1	0.38(+0.000)	0.167(-0.003)
YOLOv5x accuracy aware quantization	379.3	86.0	0.376(-0.004)	0.165(-0.005)
YOLOv5s.onnx	97.7	28.7	0.27	0.106
YOLOv5s default quantization	55.5	8.1	0.267(-0.003)	0.105(-0.001)
YOLOv5s accuracy aware quantization	8.1	84.0	0.264(-0.006)	0.104(-0.002)