Knowledge Distillation for YOLO Object Detection

M.Teichman, M.W(David). Kong, *Department of Mechanical and Industrial Engineering*
University of Toronto

***Abstract*— Many deep neural network models are increasingly deployed on resource-limit devices. As results, there is need to compress deep neural network models into small and efficient networks. One such method popular in academia is Knowledge Distillation, which compress learned features from large network model to smaller network model by learning to imitate the larger model. Our project investigates the feasibility and process of using Knowledge Distillation for 2D object detection. In our experiments we use You Only Look Once (YOLO) models to compress the knowledge from larger YOLO models into smaller YOLO models. We choose to perform traffic light detection for our 2D object detection task. The dataset was obtained from the University of Ulm, Germany. Overall, our experiments demonstrate that Knowledge Distillation can transfer knowledge to small network models, boosting detection recall and precision, and increase the networks' ability generalize while compressing network parameters [1, 2, 4].**

I. Introduction

The motivation and application behind Knowledge Distillation (KD) is to compress a large network into a smaller one that can be more easily deployed on resource-limited environments. KD can be used in many different application areas including:

1) *Edge Computing*
2) *Mobile Phone and Embedded Applications*
3) *Autonomous Vehicles and Robotics*

In edge computing, KD can be used to compress network parameters to reduce model inference times. This can reduce the communicate latency between IoT devices and edge computing services. For systems that relay on edge computing to provide real-time control signals or object detection, have a large uncompressed network can bottleneck these critical systems. For example, in manufacturing facilities its common to deploy computer vision models to detect faulty parts on the assembly lines. To reduce communication latency between factory and cloud-system, many factories use edge computing to reduce cloud communication latency as manufacturing lines generally have only couple of seconds to detect and remove the faulty part from the line. KD can be used to compress network models to ensure that the faulty part on factory line is detect and removed in time, while not disrupting the flow of manufacturing.

Furthermore, many mobile and embedded applications using augmented reality/ virtual reality (AR/VR) and machine learning require fast inference times to enhance application performance and user experience. As a result, KD can be used to compress neural network models to boost both application performance and experience. Additionally, KD can be used help deploy mobile application on older/slower resource-limited mobile phones.

Lastly, KD can be used for autonomous vehicles and robotics applications as many vehicles and robots have limited on-board computer resources. Compressing model parameters using KD can increase inferencing speeds and reduce resource compute consumption. Fast object detection can also mean faster detection of hazards giving the robotic platforms more time to react and navigate their environments.

Knowledge Distillation was originally developed by Geoffrey Hinton at el. in 2015 and was originally developed for image classification techniques [3]. The technique utilized the soft target output of teacher to distill dark knowledge into the student. This approach, however, does not work with object detection as these networks not only produce object classifications, but also bounding boxes. It was not until hint learning was proposed that academia started publishing papers in way to use KD training with object detection models. In hint learning, the feature maps of teacher network are used to teach the hidden layers of student. The goal of hint learning is to train the feature maps of student to imitate the teachers. In our project we utilize similar technique known as Fine-Grained Feature Imitation, which was developed by T.Wang al et. See the background information selection for more details [1].

You Only Look Once (YOLO) is a popular object detection algorithm developed by J.Redmon et al. in 2015 and widely been used in field of object detection due to it fast and accurate detection speeds [5]. Compared to other object detection networks such as Mask-RCNN and Faster-RCNN, YOLO utilizes a single-stage object detector, which means the entire image is processed once and is much more efficient compared to multi-stage object detectors. Our decision to use YOLO was largely based on network ability to train faster compared to other networks [5].

II. Project Description

The main objective of the project is to use fine-grained feature imitation to compress knowledge from large YOLO model into a smaller YOLO model. We use a YOLOv5 implementation by Ultralytics, which provides YOLO models of various sizes (extra-large, large, medium small and nano). The secondary objective of the project is to build object detector that can detect traffic lights using the traffic light dataset provided by University of Ulm [4].

III. Background

The Knowledge Distillation mechanism that is used in the project is fine-grained feature imitation. The main principle behind the algorithm is to train student model to imitate the teachers' high level feature map response. This is done by constructing a set of locations that estimates an object instance in a set of images. The calculated set of estimated anchor locations are called fine-grained location feature maps. These fine-grained location features are then used to distill knowledge to the student [1, 2].

The fine-grained location features are calculated by first generated a series of estimated region masks. These masks are calculated using ground truth bounding boxes and anchor priors. The masks are represented by the variable I_{ij} and are termed the fine-grained imitation masks. Combining the fine-grained imitation masks and feature maps extract from the teacher model a set of fine-grained feature imitation features are generated to train the student model. Figure 1 shows an overview of the algorithm [1].

The imitation loss function can be written as the following equation below.

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

$$\text{where } N_p = \sum_{i=1}^{W} \sum_{j=1}^{H} I_{ij}.$$

The overall training loss of the student model is then combined with the original object detection loss and the imitation loss defined above. The tunning parameter λ is added as imitation loss weight balancing factor [1].

$$L = L_{gt} + \lambda L_{imitation},$$

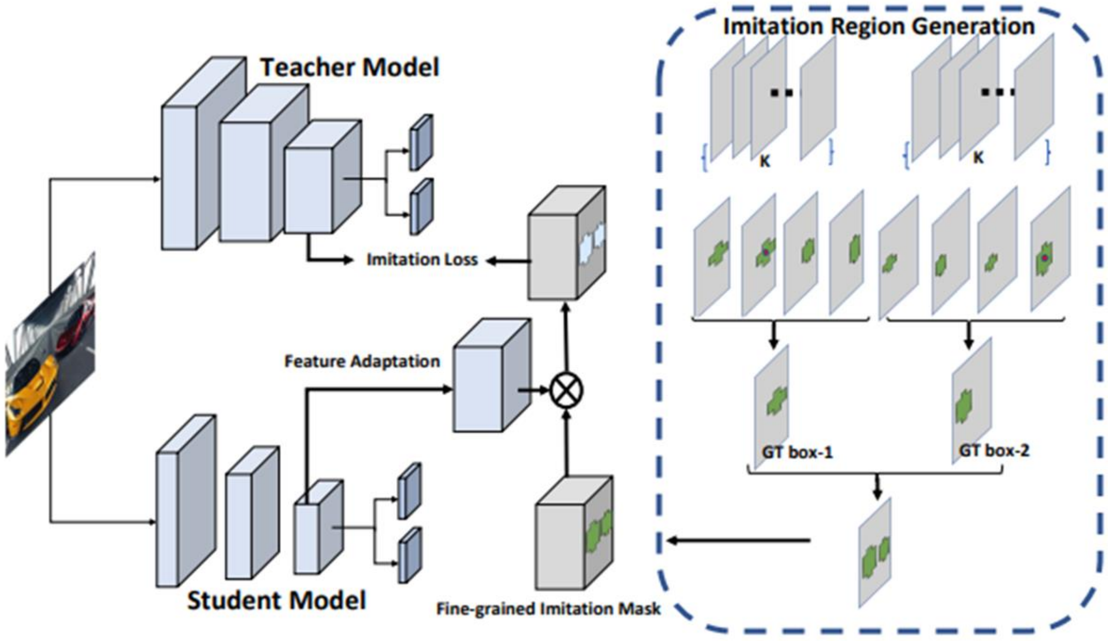

Figure 1. Fine-grained Feature Imitation Summary [1]

The following equation below defines the knowledge distillation objective function that is minimized to teach the student model to imitate the teacher's feature maps. Where s represents the student model's feature map and t represents the teacher's guided feature map [1].

$$l = \sum_{c=1}^{C} (f_{\text{adap}}(s)_{ijc} - t_{ijc})^2,$$

F_{adap} represents the adaption layer which is added to help adjust the student's feature map to match the size of the teacher's feature map [1].

IV. Methodology

The DriveU traffic light dataset by the University of Ulm, consists of a total of 292,245 total traffic light annotations. Images in the dataset consists of sequential video clips taken in 10 different German cities. Furthermore, the dataset consists of a total of 620 different unique classes. Traffic light labels vary by perspective orientation, relevance, orientation, number of lights, light state, occlusion state, and light pictogram. To simply the complexity of the dataset, the labels were converted into two main classes: relevant traffic lights and non-relevant traffic lights. Relevant traffic lights were considered relevant if light was visible to the car and had a fully defined state (green, yellow, red, and arrow etc.) Any traffic light that was partially hidden or not related to vehicle include pedestrian, tram and bike lights were labeled as non-relevant.

UDrive Traffic light dataset was converted into YOLO formatted dataset. Dataset was split into 24,520 for training, 800 for validation, 4,345 for testing. We only utilized 5 out of 10 German cities, due to dataset size exceeding our hardware space. The images were originally sized as 1024 by 2048, but the images split and resized to 640 by 640. Data augmentations were applied to training and validation sets including image blurs and contrast equalizations, which balanced the image bright between different intersection scenes.

Our project experiments consist of various trials and combinations of students and pre-trained teacher models. Each teacher was pretrained on traffic light dataset for maximum of 30 epochs. The batch size was kept as possible based on video memory limits. All YOLO models were pretrained on COCO dataset prior to transfer learning training. Additionally, we found that keeping batch size the largest possible size yielded the best object detection results. After teacher has completed pre-training, KD training is undergone with YOLO student model of smaller size. The maximum length of the KD training was 30 epochs, all the experiments were repeated twice to establish baselines for comparison.

Table 1. shows the code repositories that were utilized during the project. We utilized two Nvidia RTX 3050 laptop GPUs for project training. Which had 4 GB of dedicated video ram.

Table 1. Online Code Repositories Used in the Project [6, 7, 8]

Repository	Link	Description
YOLOv5	https://github.com/ultralytics/yolov5	Unofficial implementation of YOLOv5 used to build and train using distilled knowledge learning.
Knowledge Distillation	https://github.com/wonbeomjang/yolov5-knowledge-distillation	Implementation of knowledge distillation between YOLO models, we planning to use code to develop knowledge distillation training loop.
Knowledge Distillation	https://github.com/tranleanh/mobilenets-ssd-pytorch	Use the code to incorporated changes to knowledge distillation training loop.

V. Results

All experimental results can be summarized into the two tables below. Table 2 summarizes the pre-training experiments of the teachers. All the YOLO models were trained twice including YOLO small and YOLO nano, which were used as baseline comparison models to the KD trained models.

Table 3. summarizes the KD experiments where the pre-trained teacher in the previous section are used in knowledge distillation training of the student models. All student models are pre-trained on the COCO dataset.

Table 2. Teacher Experimental Training Summary

Model	**FLOPs**	**mAP50**	**mAP50-95**	**Relevant mAP50**	**Relevant mAP50-95**	**Not Relevant mAP50**	**Not Relevant mAP50-95**	**Epochs**
Yolov5n	4.5	36.6	15.55	45.25	18.9	14.92	4.83	30
Yolov5s	16.5	42.9	19.8	66.0	32.9	19.7	6.77	30
Yolov5m	49	49.95	23.75	73.35	37.8	26.5	9.505	30
Yolov5l	109.1	51.4	25.2	74.65	39.7	28.2	10.67	30
Yolov5x	205.7	53.95	27.0	76.55	41.95	31.3	12.045	30

Table 3. Knowledge Distillation Training Summary

Model (Student)	Teacher	mAP50	mAP50-95	Relevant mAP50	Relevant mAP50-95	Not Relevant mAP50	Not Relevant mAP50-95
Yolov5n	Yolov5l	50.9	23.2	77.9	38.3	24.0	8.03
Yolov5s	Yolov5l	54.5	26.2	80.0	41.9	29.1	10.5
Yolov5m	Yolov5l	59.7	29.7	81.7	46.0	35.4	13.3
Yolov5n	Yolov5x	49.6	22.6	75.4	37.5	23.8	7.76
Yolov5s	Yolov5x	55.4	26.4	80.9	41.8	30.0	11.1
Yolov5m	Yolov5x	58.3	29.6	81.2	45.8	35.4	13.3
Yolov5n	Yolov5m	50.1	22.8	75.8	37.8	24.4	7.89
Yolov5s	Yolov5m	55.3	26.9	80.0	42.5	30.5	11.3

The figure 2. shows a summary plot of knowledge distillation experiments, where the mAP score is compared with floating point operation per second (FLOPs) measurement. The dashlines in the plot represent the different YOLO models which can be separated by FLOPs for a given forward pass. Orange line presents the average student score without KD training the values were obtained from table 2. The green line presents average student score using KD training. Which was calculated by averaging all group model sizes together. From the gap between green and orange lines it can be observe that knowledge distillation boosts score for medium, small, and nano size YOLO models. The blue line represents the best teacher scores achieved, only YOLO medium, large and extra-large were used as teachers, as seen in table 3.

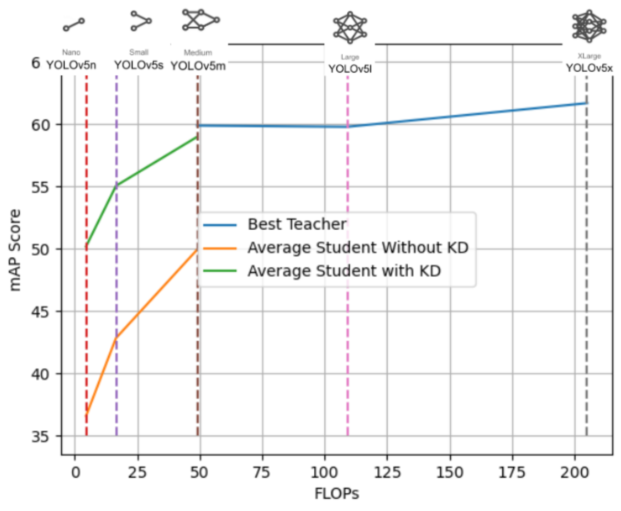

Figure 2. Knowledge Distillation Summary

The figure A1 in the appendix shows an example comparision output of traffic light detection of the trained YOLO large teacher model and train YOLO small student model. The large YOLO model appears to perform better as it able to identify all relevant traffic light, but has inference time delay of nearly four times the small YOLO model.

VI. Discussion

Overall, Knowledge Distillation training was effective at boosting student performance. This evident in the data as both mAP50 and mAP50-95 scores are increase from the baseline scores without KD training.

The YOLO models trained during the project still struggle to identify all class instances of relevant traffic lights and still have issues on the placement of boundary box. This is evident in the figure in the appendix. As the smaller YOLO model student missed two traffic lights. Both models however struggle on the placement of the bounding box. This likely due to lack of epochs, the models are all likely underfitting the data, especially when comes to boundary box. It also important to note that at end of 30 epochs of training, the regression loss of boundary box will remain relatively high.

Lastly, some of experiments performed were inconsistent, during the training of teacher models, we found that object detection performance would vary between training the teachers on different hardware environments.

VII. Conclusions

Overall, results from project showed that Knowledge Distillation training using a pre-trained teacher was relatively successful at boosting student model performance and compressing network features. There are however number of improvements that can made to the project.

1) Increase number of epochs during teacher training and KD training. As all the YOLO models still appear to be underfitting the data.

2) Utilize fine-tuning transfer learning of the teacher to boost results and speed up training. This could be done by experimentally freezing upper convolutional layers of the teacher.

3) Use multiple teachers to train a single student in KD training. Multiple teachers have the advantage of distilling knowledge of various unique feature maps and enhance student generalization.

4) Research quantization and network pruning for further student parameter compression. To optimize student inferencing speeds.

5) Experiments with more advanced object detectors such as object detection transformers.

References

[1] T. Wang, L. Yuan, X. Zhang, and J. Feng, "Distilling object detectors with fine-grained feature imitation," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[2] L. Yuan, X. Zhang, and J. Feng, "Learning Efficient Object Detection Models with Knowledge Distillation," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[3] G. Hinton, O. Vinyals, J. Dean, "Distilling the Knowledge in a Neural Network," 2014 (NIPS) Deep Learning Workshop. 2015

[4] A. Fregin, J. Muller, U. Krebel, and K. Dietmayer, "The DriveU Traffic Light Dataset: Introduction and comparison with existing datasets," 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018.

[5] J. Redmon, S. Divvala, R. Girshick, A. Farhadi "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2016

[6] Jocher, G. "YOLOv5 by Ultralytics" 2022 https://github.com/ultralytics/yolov5

[7] W. Jang "Knowledge Distillation". 2022. https://github.com/wonbeomjang/yolov5-knowledge-distillation

[8] Tran Le Anh MobileNet-SSD and MobileNetV2-SSD/SSDLite with PyTorch. 2022. https://github.com/tranleanh/mobilenets-ssd-pytorch

VIII. APPENDIX

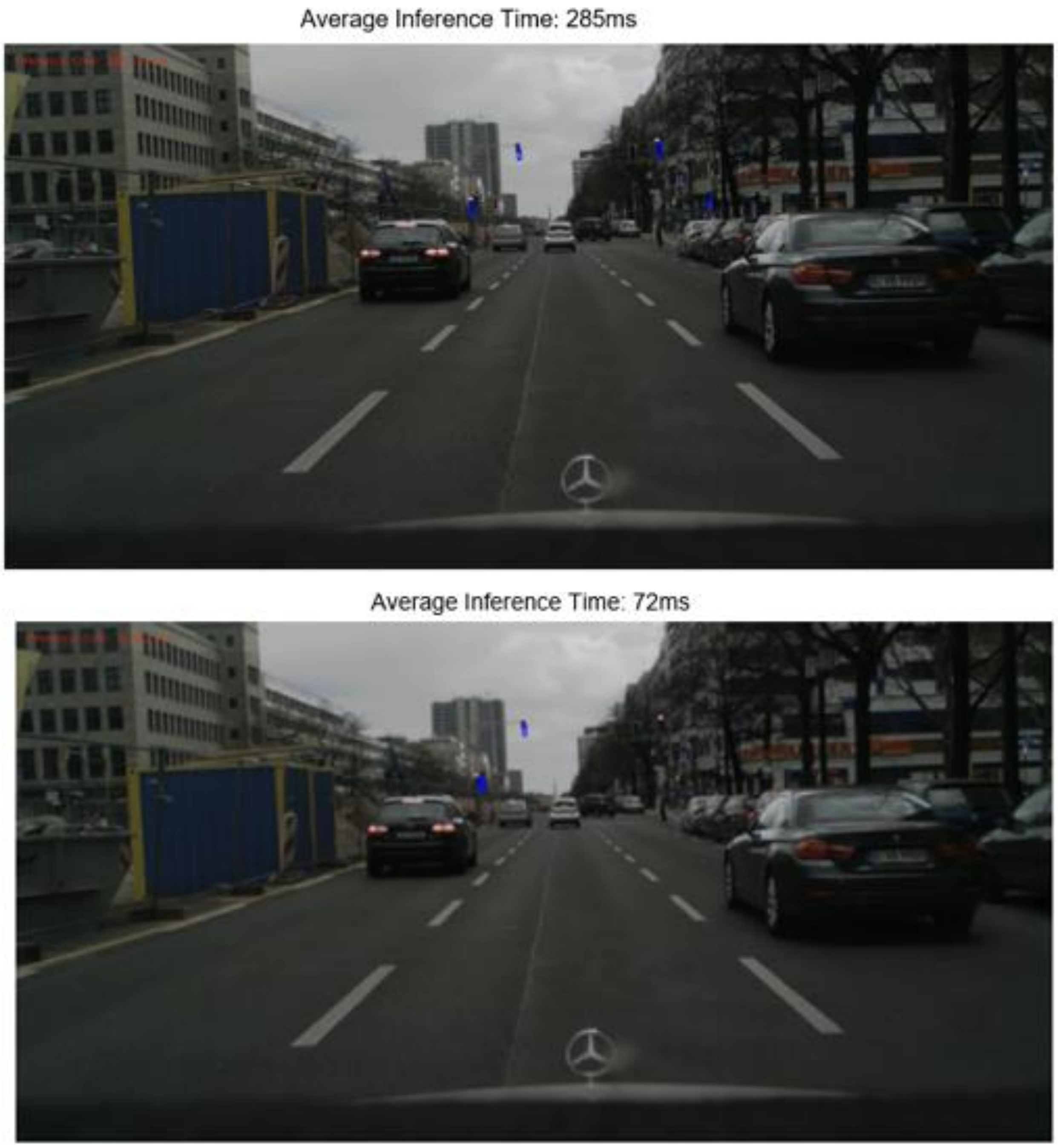

Figure A1. Object Detection Example using YOLO Large Teacher(Top) and YOLO Small Student(Bottom)