Object Detection and Tracking on Jetson Nano

Zeyuan Yin Chengqian Gao Songyao Jin Liang Li

Mohamed bin Zayed University of Artificial Intelligence

Motivation

Problem

The demand for AI power on edge devices is recently increasing. But most existing approaches face a dilemma between the high performance on the accuracy and the high cost of the high computational and memory requirements [1], which means the state-of-the-art models' ability will be restricted under the limited computational resources.

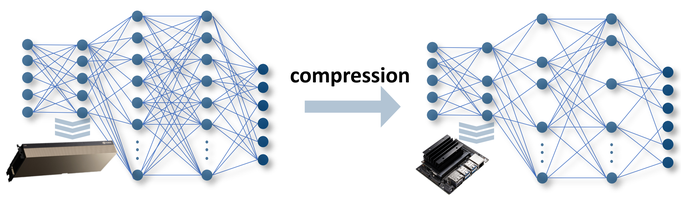

Figure 1. Three phases: train model on GPU, model compression, and deploy the tiny model on jetson nano

Objective

We focus on object detection and tracking deployed on edge computing devices and also take inference efficiency into consideration when doing object detection and tracking tasks. Thus, we try some model compress methods, like pruning [2] and quantization [3], to accelerate the model's inference.

Data & Devices

Data

- **Image Data.** We use the fruit category in OpenImage Dataset as image data to train and evaluate an object detection model. We use all categories in Cifar-100 in the evaluation of image classification of backbone models.
- **Video Data.** We shoot a video of the fruits as video data in the object tracking task.

Devices

We use NVIDIA A100 GPU to train and evaluate the model and convert the model to an onnx file that can run on the NVIDIA Jetson Nano. We report the experimental results of test models on these two computational platforms in the inference phase.

- **NVIDIA A100 GPU.** The NVIDIA A100 is a dual-slot 10.5-inch PCI Express Gen4 card based on the Ampere GA100 GPU, which allows organizations to build large-scale machine learning infrastructure.
- **NVIDIA Jetson Nano.** The Jetson Nano Developer Kit is a small, powerful computer with having CPU, GPU, high-speed interfaces, memory, power management, and more, allowing it to run multiple neural networks in parallel.

Approach

Object Detection

We choose SSD (Single Shot MultiBox Detector) [4] as the object detection framework with the following three convolutional neural networks as backbone modules: VGG, SqueezeNet, and MobileNetV1.

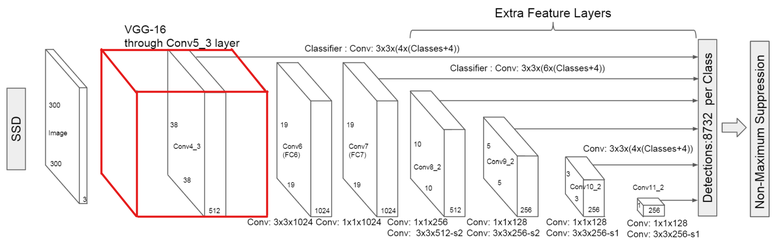

Figure 2. An illustration of SSD

Model Compression

We use the following 2 model compression methods on CNN backbone modules:

- **Neuron pruning.** We discard the neurons with the smallest values measured by L1 regularization and use channel pruning to remove the selected neurons.
- **Quantization.** Quantization is a common-used method to reduce the amount of computation by using low, precise values to replace full precise values.

Evaluation Metrics

We use some metrics to evaluate the efficiency between original and accelerated models on the following three related tasks:

- **Image Classification.** We evaluate the Top-1 accuracy of backbone models and record running time.
- **Object Detection.** We evaluate and mAP of SSD with backbones and record running time.
- **Object Tracking.** We record running time per image and FPS when tracking objects on video.

Results

In Table 1, we measured latency (ms) and top-1 accuracy for the original model and modified model on NVIDIA A100 GPU where we set the pruning rate to 10%.

Model	Baseline		Pruning		Quantization	
	Latency	Top1 accuracy	Latency	Top1 accuracy	Latency	Top1 accuracy
VGG16	47.404	0.7176	38.706	0.3303	34.832	0.7176
MobileNetV1	18.570	0.6721	16.639	0.6242	15.834	0.6723
SqueezeNet	19.013	0.6938	17.997	0.1842	16.917	0.6937

Table 1. Image Classification Task.

In Table 2, we measured the latency (ms) and meant average precision (mAP) for the original model and modified model on different devices where we set the pruning rate to 1%.

Device	Backbone	Baseline		Pruning		Quantization	
		Latency	mAP	Latency	mAP	Latency	mAP
A100	VGG	5.489	0.517	5.221	0.519	5.153	0.518
	MobileNetV1	6.733	0.462	5.698	0.462	6.277	0.450
	SqueezeNet	6.437	0.342	5.835	0.312	6.016	0.332
Jetson Nano	VGG	191.132	0.456	187.212	0.449	199.125	0.451
	MobileNetV1	34.112	0.391	33.923	0.384	33.121	0.390
	SqueezeNet	21.791	0.290	22.114	0.288	22.811	0.289

Table 2. Object Detection Task.

In Table 3, we measured the performance on video data where we set the pruning rate to 1%.

Device	Backbone	Baseline		Pruning		Quantization	
		Latency	FPS	Latency	FPS	Latency	FPS
Jetson Nano	VGG	229	4	227	4	221	4
	MobileNet	40.5	16	41	15	40	16
	SqueezeNet	26.8	22	23.5	22	23.6	22

Table 3. Object Tracking Task.

Figure 3 shows the consecutive frames tracking apples and bananas on Jetson Nano.

Figure 3. Object tracking for fruit category videos.

Conclusion & Future Work

Conclusion

1. We train an SSD object detection model with CNN backbones for fruit detection.
2. We apply the well-trained model on Jetson Nano and implement the object tracking on videos.
3. We use pruning/quantization methods to accelerate the MobileNet backbone module and improve the FPS index of object tracking.

Future Work

1. Implement removal of the discarded neurons to speed up inference rather than only setting it to zero.
2. Expend data from fruits to more object categories.
3. Design more efficient backbone models.

References

[1] Niall O'Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh.
Deep learning vs. traditional computer vision.
In *Science and information conference*, pages 128–144. Springer, 2019.

[2] Song Han, Huizi Mao, and William J Dally.
Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding.
International Conference on Learning Representations (ICLR), 2016.

[3] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan.
Deep learning with limited numerical precision.
In *International conference on machine learning*, pages 1737–1746. PMLR, 2015.

[4] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg.
Ssd: Single shot multibox detector.
In *European conference on computer vision*, pages 21–37. Springer, 2016.