Experiment of training Full Ternary Weights Network(FTWN) with darknet YOLO

Kenji Ogura

1 Abstract

Generally ternarizing weights of neural network model requires retraining after ternarization of weights. To reduce accuracy damage by ternarization I propose the staged retraining method(SRM) for full ternary weights network(FTWN). At SRM retraining and ternarizing pair consist of 4 pairs of Ternarizing and retraining to avoid local minumum problem. SRM widens ternarizing layers from full precision until FTWN step by step. To confirm effect of SRM at experiment I customized darknet framework to support weights ternarizing in convolutional layer and integrated training algorithm in it. I trained yolov2 and yolov3 models on customized darknet framework with VOC dataset 2012 and 2007 and obtained mAP as results. Full ternarized yolov2 with SRM performs 73.82% mAP against full precision's 76.85% mAP. Full ternarized yolov3 with SRM performs 71.26% mAP against full precision's 75.54% mAP.

2 Introduction

Some quantinization methods for model weights are proposed such as FP16, bfloat, fixed point 16bits, 8bit, ternary 2bits and XNor 1bit. Generally the inference task using full ternary weights -1,0,+1 with scaling factor Wl is considered as low accuracy than full precision weights. However for low power device such as mobile phone small weights will be efficient one of choices and 2bits ternary weights representaion may be x16 smaller than 32bit floating point. I consider that quatinization of weights requires retraining after quantinization. In this paper I propose Staged Retrainiing Method(SRM) for full ternary weights network and show the result as mAP.

3 Related works

I refer to papers which denote efficiency of quantinization about XNOR and Ternary weights.

- Training algorithm : "Alorithm 1" in "XNOR-Net: ImageNet Classification Using BinaryConvolutional Neural Networks"[3]
- Conversion system to Ternarize weights with scale factor Wl from full precision weigts : "2.2 Approximated solution with threshold-based ternary function" in "Ternary weight networks"[1]

I implement above Training algorithm and Conversion system into darknet code for this experiment.

4 Staged Retraining Method

I propose the staged retraining method(SRM) for full ternarization of model weights suppressing accuracy damage. To avoid local minimun problem during retraining ternarized model splitting retraining into some steps is important. In experiment SRM generates full ternarized weights for yolov2-voc[4], yolov3-voc[5] by splitting a training step into 4 stages. Roughly staging plan is below,

- Stage-0 : few layers without around of detector are ternarized(M0)
- Stage-1 : 40% of all layers are ternarized(M1)
- Stage-2 : 90% of all layers are ternarized(M2)
- Stage-3 : full ternarized(M3)

Each stages from M1 to M3 import weights from previous stage, such as stage-2 weights from stage-1. However stage-0 M0 imports usual full precision

no	layer	filters	size	input -> output	M0	M1	M2	M3
0	conv	32	3x3/1	416x416x3->416x416x32	F	F	F	T
1	max	-	2x2/2	416x416x32->208x208x32				
2	conv	64	3x3/1	208x208x32->208x208x64	F	F	T	T
3	max	-	2x2/2	208x208x64->104x104x64				
4	conv	128	3x3/1	104x104x64->104x104x128	F	F	T	T
5	conv	64	1x1/1	104x104x128->104x104x64	F	F	T	T
6	conv	128	3x3/1	104x104x64->104x104x128	F	F	T	T
7	max	-	2x2/2	104x104x128->52x52x128				
8	conv	256	3x3/1	52x52x128->52x52x256	F	F	T	T
9	conv	128	1x1/1	52x52x256->52x52x128	F	F	T	T
10	conv	256	3x3/1	52x52x128->52x52x256	F	F	T	T
11	max	-	2x2/2	52x52x256->26x26x256				
12	conv	512	3x3/1	26x26x256->26x26x512	F	T	T	T
13	conv	256	1x1/1	26x26x512->26x26x256	F	T	T	T
14	conv	512	3x3/1	26x26x256->26x26x512	F	T	T	T
15	conv	256	1x1/1	26x26x512->26x26x256	F	T	T	T
16	conv	512	3x3/1	26x26x256->26x26x512	F	T	T	T
17	max	-	2x2/2	26x26x512->13x13x512				
18	conv	1024	3x3/1	13x13x512->13x13x1024	F	T	T	T
19	conv	512	1x1/1	13x13x1024->13x13x512	F	T	T	T
20	conv	1024	3x3/1	13x13x512->13x13x1024	F	T	T	T
21	conv	512	1x1/1	13x13x1024->13x13x512	F	T	T	T
22	conv	1024	3x3/1	13x13x512->13x13x1024	F	T	T	T
23	conv	1024	3x3/1	13x13x1024->13x13x1024	T	T	T	T
24	conv	1024	3x3/1	13x13x1024->13x13x1024	T	T	T	T
25	route	16	-	-				
26	conv	64	1x1/1	26x26x512->26x26x64	T	T	T	T
27	reorg/2			26x26x64->13x13x256				
28	route	27	24	-				
29	conv	1024	3x3/1	13x13x1280->13x13x1024	T	T	T	T
30	conv	125	1x1/1	13x13x1024->13x13x125	F	F	F	T
31	detection	-	-	-				

Figure 1: Staging for yolov2-voc.cfg

no	layer	filters	size	input->output	M0	M1	M2	M3
0	conv	32	3x3/1	416x416x3->416x416x32	T	T	T	T
-	-	-	-	-	T	T	T	T
79	conv	512	1x1/1	13x13x1024->13x13x512	T	T	T	T
80	conv	1024	3x3/1	13x13x512->13x13x1024	F	F	F	T
81	conv	75	1x1/1	13x13x1024->13x13x75	F	F	F	T
82	yolo	-	-	-				
83	route	79	-	-				
84	conv	256	1x1/1	13x13x512->13x13x256	F	T	T	T
85	upsample			2x13x13x256->26x26x256				
86	route	85	61	-	F	T	T	T
87	conv	256	1x1/1	26x26x768->26x26x256	F	T	T	T
88	conv	512	3x3/1	26x26x256->26x26x512	F	T	T	T
89	conv	256	1x1/1	26x26x512->26x26x256	F	T	T	T
90	conv	512	3x3/1	26x26x256->26x26x512	F	T	T	T
91	conv	256	1x1/1	26x26x512->26x26x256	F	T	T	T
92	conv	512	3x3/1	26x26x256->26x26x512	F	F	F	T
93	conv	75	1x1/1	26x26x512->26x26x75	F	F	F	T
94	yolo	-	-	-				
95	route	91	-	-				
96	conv	128	1x1/1	26x26x256->26x26x128	F	T	T	T
97	upsample		F	2x26x26x128->52x52x128				
98	route	97	36	-				
99	conv	128	1x1/1	52x52x384->52x52x128	F	F	T	T
100	conv	256	3x3/1	52x52x128->52x52x256	F	F	T	T
101	conv	128	1x1/1	52x52x256->52x52x128	F	F	T	T
102	conv	256	3x3/1	52x52x128->52x52x256	F	F	T	T
103	conv	128	1x1/1	52x52x256->52x52x128	F	F	T	T
104	conv	256	3x3/1	52x52x128->52x52x256	F	F	F	T
105	conv	75	1x1/1	52x52x256->52x52x75	F	F	F	T
106	yolo	-	-	-				

Figure 2: Staging for yolov3-voc.cfg

weights. Figure.1 shows staging for yolov2-voc.cfg into 4 training stages and Figure.2 for yolov3-voc.cfg. On each figures 'F' denote full precision layer and 'T' denotes ternarized layer. To estimate SRM I implement ternary keyword for each convolution layer in cfg file like 'ternary=1' and I also implement ternarizing weights function including conversion system[1] and memory area saving for the ternarized weights in darknet framework.

I retrained yolov2-voc.cfg on 4 jobs, and checked each loss curves on Excell graph. And I use AlexeyAB darknet to get mAP of experiments.

5 Result of Experiment

Tables denote the result of retraining with SRM. VOC 2012, 2007 Dataset is used with training and VOC 2007 for estimation of mAP.

In Table.1 Iteration 41000(2000/class), steps x0.1 80% 90%, lr=0.001 at all stages official weights denotes full precision weights downloaded from darknet website for yolov2.

In Table.2 Iteration 100400(5000/class), steps x0.1 80% 90%, lr=0.001 at M0 and M1 Iteration 60400(3000/class), steps x0.1 80% 90%, lr=0.001 at M2 and M3. darknet53.conv.75 denotes full preci-

Stage	mAP	IOU	comments
-	76.85	54.67	official weights
M0	77.09	57.04	-
M1	76.44	56.18	-
M2	75.06	57.71	-
M3	73.82	54.90	full ternary

Table 1: result regard to yolov2

Stage	mAP	IOU	comments
-	75.54	62.78	darknet53.conv.75
M0	75.02	63.04	-
M1	73.69	63.75	-
M2	73.76	63.54	-
M3	71.26	61.61	full ternary

Table 2: result regard to yolov3

sion weights downloaded from darknet website and partialized until 75 layers.

6 Conclusion

If your applications using object detection task requires speed but not accuracy you can use full ternary weights network. To get efficient ternary weights you can use staged retraining method and generally full ternary weights is x16 smaller than fp32 representation. In experimet full ternary weights network via yolov2 or yolov3 perform accuracy within 4.5% mAP drops against full precision network.

References

[1] Fengfu Li and Bin Liu. Ternary weight networks. *ArXiv*, abs/1605.04711, 2016.

[2] Jiaolong Xu, Peng Wang, Haishun Yang, and Antonio.Lpez. Training a binary weight object detector by knowledge transfer for autonomous driving. *2019 International Conference on Robotics and Automation (ICRA)*, pages 2379–2384, 2018.

[3] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. *ArXiv*, abs/1603.05279, 2016.

[4] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.

[5] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.