Deep neural networks performance optimization in image recognition

Rassadin A.

N. Novgorod, 29.03.2017

Agenda

1. Problem discussion
2. Stage I. Overview of performance optimization approaches: categorization and survey
3. Stage II. Evaluation of existing optimization methods with the respect to the real task
4. Stage III. Dataset preparation for knowledge transfer
5. Stage IV. Multi-objective optimization
6. Stage V. Final pipeline

Agenda

1. Problem discussion
2. Stage I. Overview of performance optimization approaches: categorization and survey ✔ *НУГ*
3. Stage II. Evaluation of existing optimization methods with the respect to the real task ✔ *ITNT-2017*
4. Stage III. Dataset preparation for knowledge transfer – *TBD*
5. Stage IV. Multi-objective optimization – *TBD*
6. Stage V. Final pipeline – *TBD*

Problem discussion

Say we want to build a powerful model for some task, e.g. *emotion recognition from single image*. However, we also have:

- lack of computational resources.

On the other hand, there can be available pre-trained model though still out of our computational capabilities to use it.

Moreover, we can not reproduce it with more compact architecture, say, due to

- lack of original or suitable dataset.

Our objective: build a powerful model within the limited resources.

Thus, we need to study methods for neural network performance optimization:

- *fixed* (pre-trained) model optimization,
- optimization through knowledge transfer - *multi-objective* optimization.

Methods overview: the general-kind optimization

- continuous architecture improvement (evolution)

  convolution spread up, replacement FC with convolutions, 1x1 convolutions, residual connections etc.
- hardware / driver optimization

  special-purpose processing and memory units (Google TPU, Nervana Engine, Movidius VPU, SnapDragon 820 etc.)
- optimization-precision trade-off

  vDNN, FP16, INT8
- special-purpose frameworks

  NNPack, DeepLearningKit,

  CNNdroid, tiny-dnn, Darknet
- general framework optimizations

Library	Class	Time (ms)	forward (ms)	backward (ms)
Nervana-neon-fp16	ConvLayer	230	72	157
Nervana-neon-fp32	ConvLayer	270	84	186
TensorFlow	conv2d	445	135	310
CuDNN[R4]-fp16 (Torch)	cudnn.SpatialConvolution	462	112	349
CuDNN[R4]-fp32 (Torch)	cudnn.SpatialConvolution	470	130	340
Chainer	Convolution2D	687	189	497
Caffe	ConvolutionLayer	1935	786	1148
CL-nn (Torch)	SpatialConvolutionMM	7016	3027	3988
Caffe-CLGreenTea	ConvolutionLayer	9462	746	8716

Methods overview: additional optimization

- Pruning

  Han et al. 2016, Molchanov et al. 2016

- Distillation The Knowledge

  Hinton et al. 2014, Romero et al. 2014

- Weights Hashing / Quantization

  Chen et al. 2015, Han et al. 2016

- Tensor Decompositions: TT, CP, Tucker, …

  Lebedev et al. 2015, Kim et al. 2015, Novikov et al. 2015, Garipov et al. 2016

- Binarization

  Courbariaux / Hubara et al. 2016, Rastegari et al. 2016, Merolla et al. 2016, Hou et al. 2017

- Architectural tricks (*simple* but yet *powerful* architecture)

  Hong et al. 2016, Iandola et al. 2016 etc.

- The *silver bullet* architecture --it's a kind of maaagic..

  Hasanpour et al. 2016

Comparisons. Optimization type

	Train - Memory	Train - Speed	Inference - Memory	Inference - Speed
Distillation the knowledge	-	-	+	+
HashedNets	?	?	+	?
Deep Compression	-	-	+	+
CP-Decomposition	-	-	+	+
TensorNet	?	-	+	?
BinaryNet	-	-	+	+
Binary-Weight-Network	-	-	+	+
XNOR-Net	-	+	+	+
SqueezeNet	N/A	N/A	+	?
PVANet	N/A	N/A	+	+
Tiny Darknet	N/A	N/A	+	+
BranchyNet	-	-	-	+

Comparisons. Scores

	Memory reduction while training	Memory reduction while inference	Inference speedup	Accuracy gain	Baseline model	Dataset
FitNets	-	36	13.36	-1.17	Maxout	CIFAR-10
HashedNets	-	64	?	0,24	*same-size*	MNIST
Deep Compression	-	49 (~4)	?	0.33	VGG-16	ImageNet
CP-Decomposition	-	12	4.5	-1	AlexNet	ImageNet
TensorNet	?	80	?	-1.1	*simple*	CIFAR-10
BinaryNet	~32 (theoretical)		3.4~23	1.53	Maxout	CIFAR-10
Binary-Weight-Network	-	67	58 (CPU)	-8.5	ResNet-18	ImageNet
XNOR-Net	-			-18.1		
SqueezeNet	?	50	1.	0.3	AlexNet	ImageNet
Tiny Darknet	?	60	2.9	1.5		
BranchyNet	-	-	1.9	-1,53	ResNet-110	CIFAR-10

Experiments. Formulation

Baseline - visual emotion recognition, Levi et al. 2015. Unfortunately, original EmotiW 2015 dataset not available and Radboud Faces Database cropped by face was used instead for training and evaluation:

- ±45° of rotation, the same balanced and independent train / test sets for all experiments.

Common experiments settings:

- **SquezeNet, CP-decomposition**, **HashedNets**, **BWN**, **XNOR-Net**
- author's code(guarantees exact implementation and results reproducibility);
- SGD with momentum equal to 0.9, fixed learning rate equal to 0.001;
- no data augmentation except channel-wise z-score for **SquezeNet**.

Evaluation settings:

- accuracy metric: test accuracy;
- speedup metrics:
    - epoch time for single forward pass and subsequent gradient update on GPU for mini-batch in one random sample, averaged over 1000 runs
    - GPU inference time for single random sample, averaged over 1000 runs

Experiments. Scores

	Epoch time, ms	Inference time, ms	Accuracy, %	Model size, MB
CP-Decomposition	N/A	7.74	87.5	-23.5%
***Baseline* (Caffe)**	**22.94**	**4.94**	**89.14**	**2.8**
HashedNets	294.8	158.2	96.31	-81.64%
Binary-Weight-Network	83.8	33.5	98.57	0% (11.6)
XNOR-Net	84.3	34.2	58.81	-2.4% (11.6)
XNOR-Net w/o weights b-n	43.4	34.1	88.32	-2.4% (11.6)
***Baseline* (Torch)**	**43.7**	**33.4**	**97.13**	**372.2**

Experiments. Gains

	Training speedup	Memory reduction while inference	Inference speedup	Parameters reduction	Accuracy loss
HashedNets	-	?	4.7 times slower	81.64%	slight
CP-Decomposition	N/A	0%	-	23.5%	slight
Binary-Weight-Network	4x (by epochs number)	0%	-	0%	sloght
XNOR-Net	-	2.4%	-	2.4%	huge

Knowledge transfer

We choose *VGG Faces* dataset with 2+ million faces from 2+ thouthands of identities. Since dataset have no emotional markup, we should mark out it first with the **baseline** model.

Next, we should train **descendant** model using *multi-loss optimization process* where we are trying not only predict the same labels but also restore the original probability distribution.

Unfortunately, we was unable to process even 4% of dataset on existing hardware..