DeepRebirth: Accelerating Deep Neural Network Execution on Mobile Devices

Dawei Li,[1] **Xiaolong Wang,**[1] **Deguang Kong**
[1]Samsung Research America, Mountain View, CA
{xiaolong.w, dawei.l}@samsung.com, {doogkong}@gmail.com

AAAI 2018

Presenter: Youngeun Kim
young.ryan.36524@gmail.com

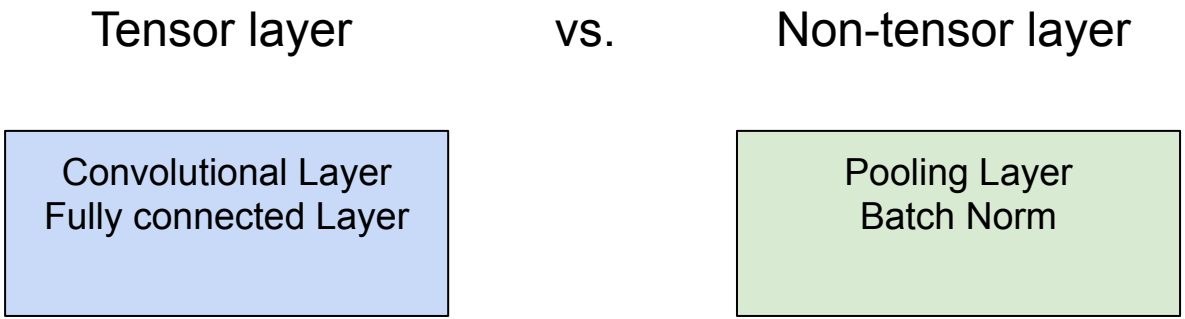
Neural Network Acceleration Study Season #2

Contents

Motivation

Current model compression method still cannot satisfy real-time processing.

The major obstacle is the excessive execution time of **non-tensor layers**

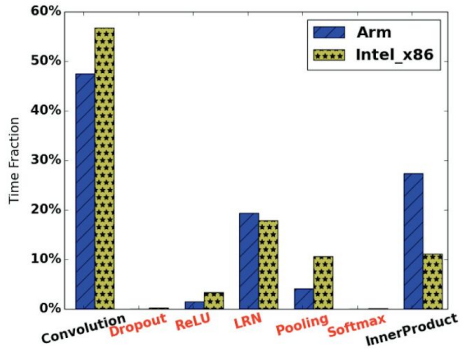


Motivation

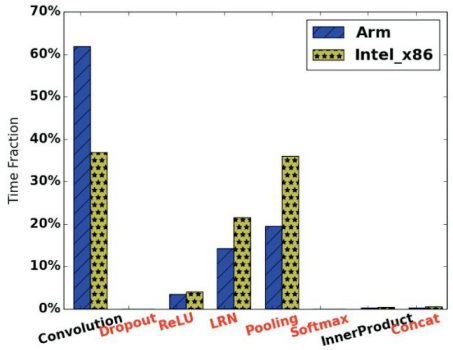
Time analysis on Tensor and Non-Tensor layers

Table 2: Percentage of Forwarding Time on Non-tensor Layers

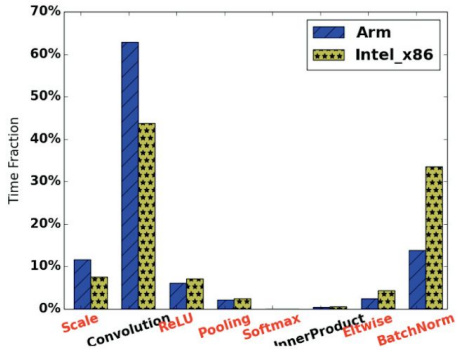
Network	Intel x86	Arm	Titan X
AlexNet	32.08%	25.08%	22.37%
GoogLeNet	62.03%	37.81%	26.14%
ResNet-50	55.66%	36.61%	47.87%
ResNet-152	49.77%	N/A	44.49%
Average	49.89%	33.17%	35.22%



(a) AlexNet



(b) GoogLeNet



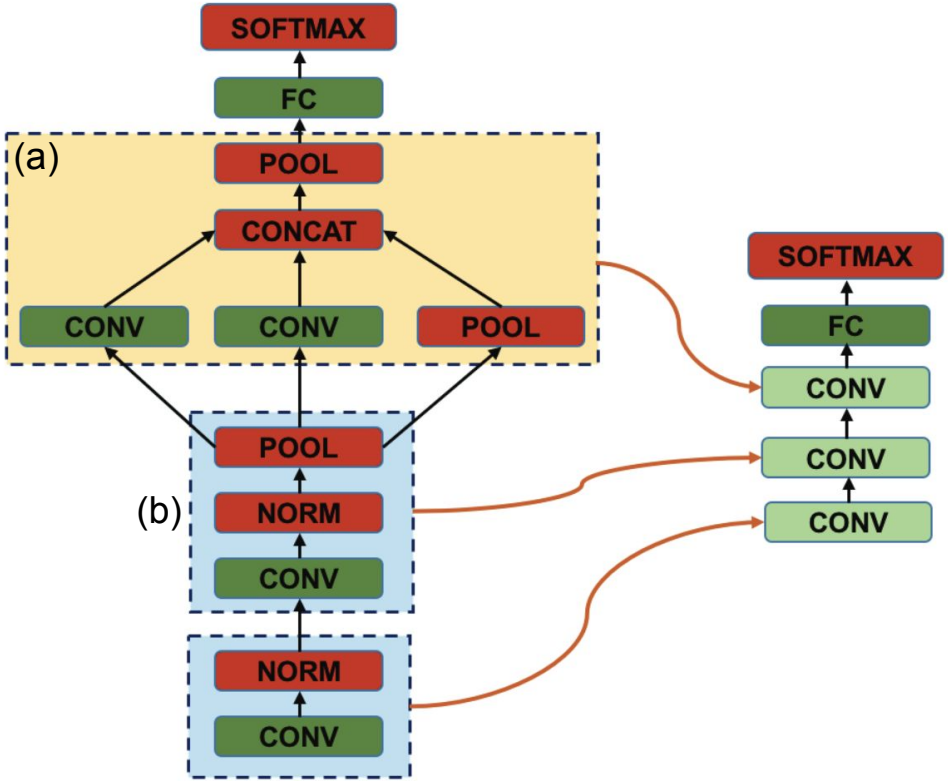
(c) ResNet-50

Figure 2: Time Decomposition for each layer. Non-tensor layers (e.g., dropout, ReLU, LRN, softmax, pooling, etc) shown in red color while tensor layers (e.g., convolution, inner-product) shown in black color.

Motivation

Two types of slimming methods

(a) branch slimming
(b) streamline slimming



Contribution

1) Existing works are designed to reduce the model with handling tensor-type layers, **however, handling non-tensor layer has not been discussed so far.**

2) In this paper, we emphasize and validate experimentally that the proposed **method is orthogonal to compression techniques on tensor-type layers**.

Method - Streamline Slimming

Pooling Layer

- Remove the pooling layer
- Set the stride value of the new convolution layer

None Pooling Layer

- Prune those layers from the original DNN

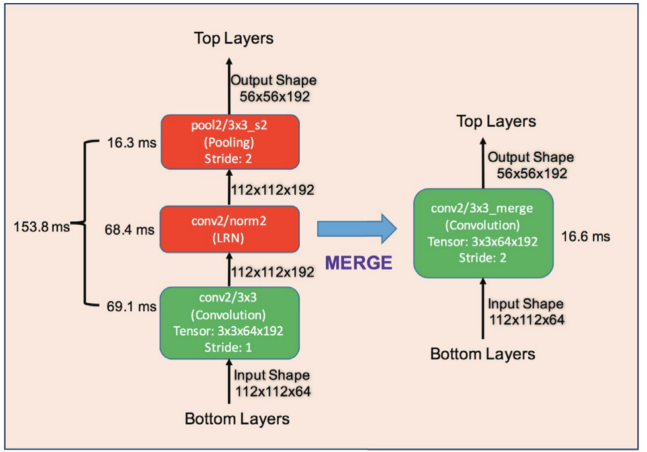


Figure 3: Streamline Slimming: The GoogLeNet example and the running time is measured using bvlc_googlenet model in Caffe on a Samsung Galaxy S5. Left panel: convolution (in green), LRN (in red), pooling (in red). Right Panel: single convolution layer. The three layers in the left panel are merged and regenerated as a convolution layer (i.e., slim layer) in the right panel.

Method - Streamline Slimming

Theoretical Analysis

$$X^i \xrightarrow{f_{\text{conv}}} Y^i_{\text{cv}} \xrightarrow{f_{\text{bn}}} Y^i_{\text{cv+bn}} \xrightarrow{f_{\text{sl}}} Y^i_{\text{cv+bn+sl}} \xrightarrow{f_{\text{pooling}}} Y^i_{\text{cv+bn+sl+pl}} \dot{\rightarrow} \cdots\cdots := Y^i_{\text{CNN}}$$

$$\begin{cases} f_{\text{conv}} : \mathbf{W}_{\text{conv}}, \mathbf{B}_{\text{conv}}; \\ f_{\text{bn}} : m, \quad \mu, \quad \sigma^{\mathbf{2}}; \\ f_{\text{sl}} : \gamma, \quad \beta; \\ f_{\text{pooling}} : p; \\ f_{\text{LRN}} : \kappa, \quad \rho \quad \alpha. \\ \cdots \end{cases}$$

Idea is to find a new mapping function

$$\tilde{f}(\tilde{\mathbf{W}}, \tilde{\mathbf{B}}) : \quad X^i \rightarrow Y^i_{\text{CNN}}$$

$$(\tilde{\mathbf{W}}^*, \tilde{\mathbf{B}}^*) = argmin_{\mathbf{W},\mathbf{B}} \sum_i \| Y^i_{\text{CNN}} - \tilde{f}(\mathbf{W}, \mathbf{B}; X^i) \|^2_F.$$

Method - Streamline Slimming

Theoretical Analysis: how to get closed form solution for the new Conv layer?

Lemma 2. *Let W_j, B_j, μ_j, σ_j^2, γ_j and b_j be the corresponding j-th dimension in the reshaped weight vector or bias vector in Eq.(3), and $\tilde{W}_j$, $\tilde{B}_j$ be the learned new convolution layer parameter in Eq.(5). Then, if Y^i_{CNN} is obtained after the three layers of f_{conv}, f_{bn}, f_{sl} in the sequence order, i.e., $Y^i_{CNN} := Y^i_{cv+bn+sl}$, we have closed form solution for the parameters in the new convolution layer:*

$$\begin{aligned} \tilde{W}_j &= \eta_j W_j, \\ \tilde{B}_j &= \eta_j B_j + \beta_j - \eta_j \frac{\mu_j}{m}, \\ \eta_j &= \frac{\gamma_j}{\sqrt{\frac{\sigma_j^2}{m}}}. \end{aligned} \tag{6}$$

Method - Streamline Slimming

Theoretical Analysis: how to get closed form solution for the new Conv layer?

Lemma 2. *Let W_j, B_j, μ_j, σ_j^2, γ_j and b_j be the corresponding j-th dimension in the reshaped weight vector or bias vector in Eq.(3), and $\tilde{W}_j$, $\tilde{B}_j$ be the learned new convolution layer parameter in Eq.(5). Then, if Y^i_{CNN} is obtained after the three layers of f_{conv}, f_{bn}, f_{sl} in the sequence order, i.e., $Y^i_{CNN} := Y^i_{cv+bn+sl}$, we have closed form solution for the parameters in the new convolution layer:*

$$\begin{aligned} \tilde{W}_j &= \eta_j W_j, \\ \tilde{B}_j &= \eta_j B_j + \beta_j - \eta_j \frac{\mu_j}{m}, \\ \eta_j &= \frac{\gamma_j}{\sqrt{\frac{\sigma_j^2}{m}}}. \end{aligned} \tag{6}$$

$$\begin{aligned} Y_j &= \gamma_j \Big(f_{\text{bn}} \cdot f_{\text{conv}}(X) \Big)_j + \beta_j, \quad \triangleright \text{Scaling} \\ &= \gamma_j \Big(\frac{f_{\text{conv}}(X)_j - \mu_j}{\sqrt{\sigma_j^2}} \Big) + \beta_j, \quad \triangleright \text{BN} \\ &= \gamma_j \Big(\frac{(W * X)_j + B_j - \frac{\mu_j}{m}}{\sqrt{\frac{\sigma_j^2}{m}}} \Big) + \beta_j. \quad \triangleright \text{Convolution} \end{aligned} \tag{8}$$

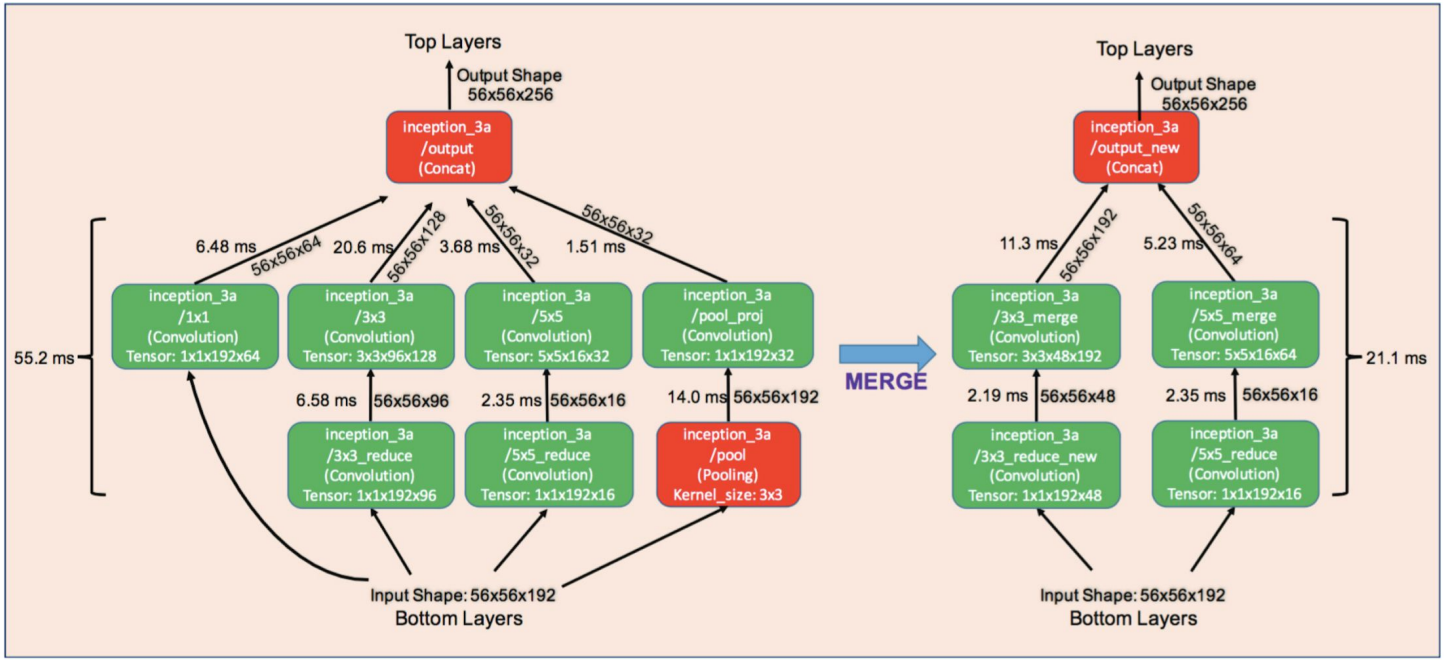
Let $\eta_j = \frac{\gamma_j}{\sqrt{\frac{\sigma_j^2}{m}}}$, then Eq.(8) is equivalent to:

$$Y_j = \underbrace{\eta_j (W * X)_j}_{weight} + \underbrace{\Big(\eta_j B_j - \frac{\eta_j \mu_j}{m} + \beta_j \Big)}_{bias}. \tag{9}$$

Compared to Eq.(7), we have $\tilde{W}_j = \eta_j W_j$ and $\tilde{B}_j = \eta_j B_j + \beta_j - \eta_j \frac{\mu_j}{m}$. This completes the proof. □

Method - Branch Slimming

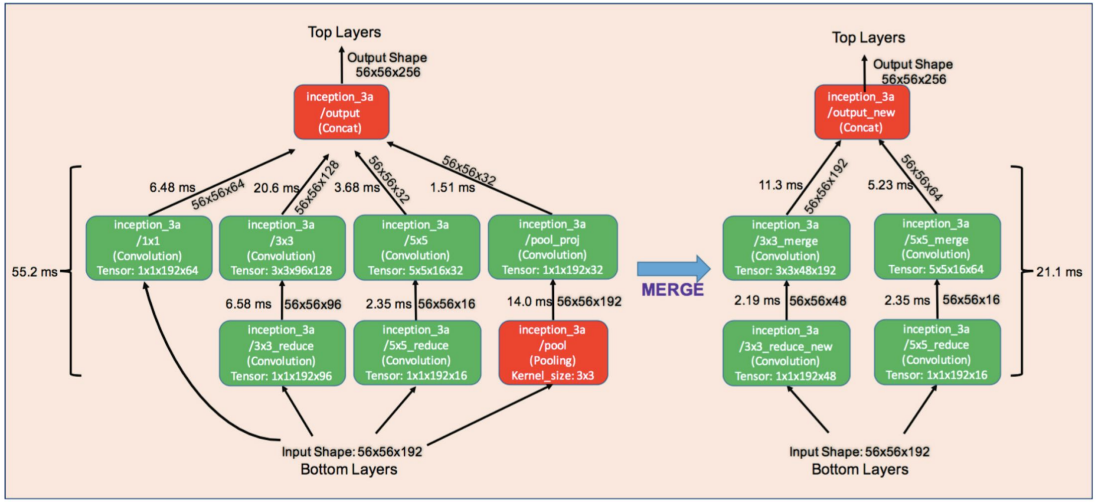
- Combining small size kernel with large kernel



Method - Branch Slimming

Branch reducing: for greater speed-up, we further reduce the number of feature maps generated by the 1×1 “reducer”

Tensor-Branch Slimming: A convolution branch with a smaller kernel size can be absorbed to a convolution branch with a larger kernel size



Method - Finetuning

1) In DeepRebirth, we leverage Xavier (Glorot and Bengio 2010) initialization to initialize the parameters in the new layer **while keeping the weights of other layers unchanged**

2) Generally, the proposed optimization scheme is applied from the **bottom layer to the top layer**

3) Alternative is to learn **multiple slim layers** at the same time

Experiments

Setting: caffe's GoogLeNet implementation (i.e.,bvlc_googlenet)

Compare with:

1) GoogLeNet
2) GoogLeNet-Slim
3) GoogLeNet-Tucker
4) GoogLeNet-Slim-Tucker
5) SqueezeNet

*Tucker decomposition approach (Kim et al. 2015)

Experiments

Table 4: Layer breakdown of GoogLeNet forwarding time cost

Layer	GoogLeNet	GoogLeNet -Tucker	GoogLeNet -Slim (ours)	GoogLeNet -Slim-Tucker (ours)
conv1	94.92 ms	87.85 ms	8.424 ms	6.038 ms
conv2	153.8 ms	179.4 ms	16.62 ms	9.259 ms
inception_3a	55.23 ms	85.62 ms	21.17 ms	9.459 ms
inception_3b	98.41 ms	66.51 ms	25.94 ms	11.74 ms
inception_4a	30.53 ms	36.91 ms	16.80 ms	8.966 ms
inception_4b	32.60 ms	41.82 ms	20.29 ms	11.65 ms
inception_4c	46.96 ms	30.46 ms	18.71 ms	9.102 ms
inception_4d	36.88 ms	21.05 ms	24.67 ms	10.05 ms
inception_4e	48.24 ms	32.19 ms	28.08 ms	14.08 ms
inception_5a	24.64 ms	14.43 ms	10.69 ms	5.36 ms
inception_5b	24.92 ms	15.87 ms	14.58 ms	6.65 ms
loss3	3.014 ms	2.81 ms	2.97 ms	2.902 ms
Total	**651.4 ms**	**614.9 ms (1.06x)**	**210.6 ms (3.09x)**	**106.3 ms (6.13x)**

Table 5: Execution time using different methods (including SqueezeNet) on different processors

Device	GoogLeNet	GoogLeNet -Tucker	GoogLeNet -Slim	GoogLeNet -Slim-Tucker	SqueezeNet
Moto E	1168.8 ms	897.9 ms	406.7 ms	**213.3 ms**	291.4 ms
Samsung Galaxy S5	651.4 ms	614.9 ms	210.6 ms	**106.3 ms**	136.3 ms
Samsung Galaxy S6	424.7 ms	342.5 ms	107.7 ms	**65.34 ms**	75.34 ms
Macbook Pro (CPU)	91.77 ms	78.22 ms	23.69 ms	**15.18 ms**	17.63 ms
Titan X	10.17 ms	10.74 ms	6.57 ms	7.68 ms	**3.29 ms**

Table 6: Storage, Energy and Runtime-Memory Comparison

Model	Energy	Storage	Memory	Max Batch Size on Titan X
GoogLeNet	984 mJ	26.72 MB	33.2 MB	350
GoogLeNet-Tucker	902 mJ	14.38 MB	35.8 MB	323
GoogLeNet-Slim	**447 mJ (2.2x)**	23.77 MB	13.2 MB	**882 (2.52x)**
GoogLeNet-Slim-Tucker	**226 mJ (4.4x)**	11.99 MB	14.8 MB	**785 (2.24x)**
SqueezeNet	288 mJ	4.72 MB	36.5 MB	321

Conclusion

1) Re-generating new tensor layers from optimizing non-tensor layers and their neighborhood units

2) DeepRebirth is also compatible with slimming approaches that aims to reduce tensor layers