RETRAIN-FREE FULLY CONNECTED LAYER OPTIMIZATION USING MATRIX FACTORIZATION

Yi Sun, Xuejiao Liu, Luhong Liang

Hong Kong Applied Science and Technology Research Institute (ASTRI), Hong Kong
E-mails: {mikesun, xjliu, luhongliang}@astri.org

ABSTRACT

The complexity of Deep Neural Networks (DNNs) hinders their implementation on embedded system with limited hardware resources. To deal with this issue, this paper presents a novel optimization algorithm based on semi-Nonnegative Matrix Factorization for Fully Connected layers (semi-NMF-based FC optimization). Compared with previous network surgery techniques, our proposed method optimizes network structure in a more implementation-friendly manner with full controllability and no requirement on retraining using training data. Simulations using AlexNet and VGG-16 on ImageNet task show the effectiveness and efficiency of semi-NMF-based FC optimization.

Index Terms— Matrix factorization, neural network optimization, image classification, computer vision

1 Introduction

Deep Neural Networks (DNNs) are becoming widely recognized and have been adopted as the state-of-the-art solutions to most computer vision problems. Supported by the development of computational power, DNNs are designed with high complexity for better performances. However, such computation- and storage-intensive structures are hard to be fitted into embedded and mobile devices due to limited storage and power cap [1]. The complexity needs to be reduced. Significant redundancies in DNN have also been reported [2].

Existing works is known as Network Surgery or Pruning [3, 4, 5, 6, 7, 8]. Though achieving impressive compression rates, surged DNNs using the state-of-the-art techniques have two limitations if to be implemented on dedicated hardwares such as FPGA. The first one is the sparse and irregular network structures. Sparsity and irregularity induce extra overheads and provide little control on resultant structures which reduces the flexibility of hardware design. Retraining is the second limitation which could consume even more time and resources than training the original network. The unaccessible data in commercial cases is another obstacle for retraining.

To address these two issues, we propose a semi-Nonnegative Matrix Factorization based optimization algorithm specifically for Fully Connected Layers (semi-NMF-based FC optimization). Nonnegative matrix factorization (NMF) has been widely adopted in data-mining-related researches [9, 10, 11]. Semi-NMF is modified from NMF and proposed by Ding et al. [12]. To the best of our knowledge, we are the first to utilize semi-NMF technique on optimizing neural network structure. Specifically, we modify the original semi-NMF algorithm and propose an iterative optimization algorithm based on DNN features. The proposed algorithm factorizes selected FC weight matrix into smaller matrices with controllable structures and is able to keep network performance without retraining using training data.

We evaluate the effectiveness and efficiency of semi-NMF-based FC optimization on AlexNet [13] and VGG-16 [14] for ImageNet [15] task. Though not the most recent-invented networks, these two networks possess high FC ratio in terms of network complexity. Such FC-heavy feature is challenge when design dedicated hardware, especially considering the limited off-chip memory bandwidth. Optimized AlexNet contains 80% less weights in FCs with a 0.43% top-1 accuracy drop and 0.14% top-5 accuracy drop. Optimized VGG-16 contains 84% less weights in FCs providing 0.45% top-1 accuracy drop and 0.27% top-5 accuracy drop. Both results are computed by CPU and consume 10% to 30% execution time compared with retraining on GPU. Moreover, semi-NMF-based FC optimization is orthogonal to existing surgery approaches enabling further combinations and improvements.

This paper is structured as follows. Section 2 gives a brief recap about related works. We introduce semi-NMF in Section 3. Section 4 discuss test results and verifications of FPGA implementation. Finally, we conclude our work in Section 5 followed by future plans.

2 Related Works

Biased weight decay by Hanson et al. [16] was one of the early approaches for network pruning. Optimal Brain Damage from LeCun et al. [4] and Optimal Brain Surgeon from Hassibi et al. [3] improve the pruning based on the Hessian of the loss function and demonstrate higher accuracy compared with magnitude-based pruning.

Recent works from Han et al. [7] improved the pruning

This paper is supported by HKSAR ITC project #ARD/166.

performance to $9\times$ on AlexNet and $13\times$ on VGG-16 without accuracy loss. The state-of-the-art surgery, *Dynamic Surgery* proposed by Guo et al. [8], combined pruning with splicing. It compressed LeNet-5 to $0.9\%(108\times)$ of the original size and AlexNet to $5.7\%(17.7\times)$ without accuracy loss. *Dynamic Surgery* relied on extra binary mask matrices to store surgery statues. Surging and retraining were conducted simultaneously but required more iterations. AlexNet was surged with 700K iterations exceeding the 450K used for original model training. Han et al. [1] further combined surgery with quantization and Huffman coding and achieved $35\times$ and $49\times$ compression rate in terms of network model file size. Despite the amazing compression rate, the presence of retrain and the encoding and decoding require excessive data and dedicated hardware design.

Matrix decomposition is another set of approaches. Non-negative matrix factorization has been successfully adopted in data-mining-related tasks. Lee and Seung introduced NMF on learning the parts of objects in [10] and analyzed two NMF for multivariate data decomposition [11]. Paatero and Tapper proposed an optimized model based on NMF to estimate data values [9]. In terms of network structure, Denil et al. approximated weight matrices by low-rank decompositions achieving $1.6\times$ speedup on convolutional layers with 1% accuracy loss [2]. Denton et al. [17] and Zhang et al. [6] adopted similar approaches for $2\times$ to $4\times$ faster inference speed with less than 1% accuracy drop. Tara et al. utilized low-rank matrix factorization on output FC layer for speech recognition [5] leading to 50% to 80% reduction with similar performances.

3 Network Optimization and Matrix Factorization

To address the limitations in existing approaches, we propose a semi-NMF-based FC optimization algorithm.

Without loss of generality, a FC with M inputs and N outputs can be represented by a $M \times N$ weight matrix. If the matrix is factorized into two regular and dense matrices of dimension $M \times r$ and $r \times N$, we can compress the original matrix by controlling r so that $M \times r + r \times N \ll M \times N$. The key problem is that when features of the original matrix, such as its rank, are not controllable, how to maintain the network performance without retraining the factorized matrices.

We solve this problem by modifying the original semi-NMF technique based on DNN features to realize the compression while keeping network performance.

In this section, we first formulate the FC optimization problem and briefly describe the original semi-NMF algorithm. Then we introduce the design of our algorithm in detail.

3.1 Problem Formulation

The behavior of a FC can be formulated as

$$Y = XV, \tag{1}$$

where $X \in R^M$ is the input vector with dimension M and $Y \in R^N$ is the output vector with dimension N. V is the weight matrix with dimension $M \times N$ and is used to map the inputs X to outputs Y. The optimization on FCs is to find a new weight matrix $V^{'}$ by solving the following objective function:

$$\min \Delta \sum_{l=1}^{L} \left(V_l - V_l^{'} \right), \ s.t. \ C_{V_l{'}} \ll C_{V_l}, \tag{2}$$

where L is the amount of FCs to be optimized. V_l and $V_l^{'}$ are the original and optimized weight matrix of lth FC and $l = 1, 2, ..., L$. C_{V_l} and $C_{V_l{'}}$ are the complexities of weight matrix V_l and $V_l^{'}$.

Existing techniques rely mostly on retraining to solve Eq. (2) Resultant $C_{V_l{'}}$ can be very low but hard to control.

3.2 Semi-NMF

We propose a semi-NMF based optimization approach to solve Eq. (2) in order to skip retraining, control resultant matrix easily, and gain more implementation-friendly structures.

Without loss of generality, semi-NMF factorizes weight matrix $V^{'}$ into two matrices W and H by the following equation [12]:

$$V_{\pm} \approx V_{\pm}^{'} = W_{\pm} H_{+}^{T}, \tag{3}$$

where $V \in R^{M \times N}, W \in R^{M \times r}, H \in R^{N \times r}$. $\pm$ represents mixed sign values and $+$ means non-negative values only.

Semi-NMF is motivated by K-means clustering. Difference between original matrix V and factorized matrices W and H can be written as follows [12]:

$$E\left(W, H\right) = \left\| V - WH^T \right\|^2 = \sum_{n=1}^{N} \sum_{i=1}^{r} h_{ni} \|v_n - w_r\|^2, \tag{4}$$

where V is a mixed sign weight matrix with v_n, $n = 1, 2, ..., N$ representing its nth entry. W can be seen as cluster centroids having positive and negative entries w_i, $i = 1, 2, ..., r$. H represents cluster indicators with only non-negative entries h_{ni}. $\|\bullet\|^2$ is the divergence functions. We adopt squared error (or *Frobenius Norm*) in our work.

Minimizing $E\left(W, H\right)$ is a NP-hard problem. It is solved by updating factors W and H iteratively until it converges. The general process is described in **[12]**. First, *K*-means clustering is used to initialize H. Then we fix H and update W by:

$$W = VH\left(H^T H\right)^{-1}, \tag{5}$$

With the updated W, we update H by:

$$H_{ni} \leftarrow H_{ni} \sqrt{\frac{(V^T W)_{ni}^{+} + \left[H(W^T W)^{-}\right]_{ni}}{(V^T W)_{ni}^{-} + \left[H(W^T W)^{+}\right]_{ni}}}. \tag{6}$$

Eq. (5) and Eq. (6) consist of one optimization iteration. Termination criterion is given by a threshold δ_E. If $E(W, H) \leq \delta_E$, the optimization is terminated. Otherwise the updated H will be used in Eq. (5) for the next iteration.

3.3 Proposed optimization algorithm

Our proposed optimization algorithm adapts original semi-NMF with features from DNN FCs. More specifically, instead of using the error function in Eq. (4) directly, we redefine the objective function in Eq. (2) as:

$$\min \Delta \left(V_l - V_l^{'} \right) = \min \left\{ E\left(W, H\right), \Delta_{Acc} \right\}, \qquad (7)$$

where ΔAcc is the loss in test accuracies.

In order to solver Eq. (7), we propose another threshold δ_{Acc} to be used together with δ_E as the termination criterion. For every k iterations, after updating H by Eq. (6), we perform inference over the whole network using the current W and H on factorized FCs and record the Δ_{Acc} compared with the original network. The optimization is terminated if and only if $E\left(W, H\right) \leq \delta_E$ ***and*** $\Delta_{Acc} \leq \delta_{Acc}$.

Pseudo code for our proposed optimization algorithm is shown in **Algorithm 1**.

Algorithm 1 Semi-NMF-based FC optimization

Given the trained network `Net` *with all* V_l, $l = 1, 2, ..., L$
Given δ_E, δ_{Acc}, $MaxItr$, r_l
$l = 1$
while $l \leq L$ **do**
    Initialize W_l, H_l *by K-means clustering*
    $itr = 1$
    Calculate $E\left(W_{litr}, H_{litr}\right)$ *by Eq. (4)*
    Replace original V_l *by* W_{litr} *and* H_{litr}
    Perform inference on `Net` *and record* Δ_{Acc}

    while $(E\left(W_{litr}, H_{litr}\right) > \delta_E$ **or** $\Delta_{Acc} > \delta_{Acc})$ **and** $itr \leq MaxItr$ **do**
        $W_{litr+1} = V_l H_{litr} \left({H_{litr}}^T H_{litr} \right)^{-1}$
        $H_{litr+1} \leftarrow H_{litr} \sqrt{\frac{\left({V_l}^T W_{litr}\right)^+ + \left[H_{litr}\left({W_{litr}}^T W_{litr}\right)^-\right]}{\left({V_l}^T W_{litr}\right)^- + \left[H_{litr}\left({W_{litr}}^T W_{litr}\right)^+\right]}}$
        Calculate $E\left(W_{litr}, H_{litr}\right)$ *by Eq. (4)*
        Replace W_{litr} *and* H_{litr} *by* W_{litr+1} *and* H_{litr+1}
        Perform inference on `Net` *and record* Δ_{Acc}
        $itr = itr + 1$

    $l = l + 1$

By adjusting δ_E, δ_{Acc}, L and r, our proposed algorithm is able to control the structure and dimension of optimized FCs. Network performance is also kept without retraining on training data. Resultant weight matrices are regular, dense, and with smaller dimensions making them more implementation-friendly.

4 Experiments

In this section, we show the effectiveness of semi-NMF on both general-purposed processors (CPU & GPU) and dedicated hardware (FPGA).

We test two well-known CNNs, AlexNet [13] and 16-layer VGG (VGG-16) [14], on ImageNet task [15]. All trainings and inferences are carried out on a desktop with i7 dual CPU @ $4.00GHz$, 16GB RAM, single GTX1080 GPU. Programming environment is CAFFE [18], CUDA 7.5 [19] together with cuDNN 5.1 [20].

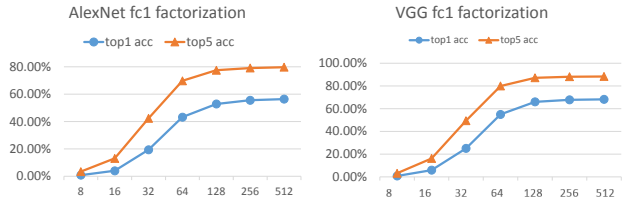

Fig. 1. Semi-NMF vs retraining optimization. Left: AlexNet fc1. Right: VGG16 fc1

AlexNet has 5 trainable Convs (conv1 - conv5) and 3 FCs (fc1-fc3). All Convs have around 2.3M weights in total. fc1 has 38M weights with dimension 9216×4096. fc2 has 17M weights represented by a 4096×4096 matrix. fc3 generates predictions for image type by a 4096×1000 weight matrix adding up to 4M weights. Original AlexNet has top1 accuracy at 56.33% and top-5 accuracy at 79.53% for 1000-type ImageNet-2012 task. VGG-16 has 13 trainable Convs (conv1-conv13) with about 14.1M weights and 3 FCs (fc1-fc3). fc1 has dimension 25088×4096 which is about 102.8M weights. fc2 and fc3 have identical dimensions with AlexNet fc2 and fc3, respectively. VGG-16 has around 137.6M weights in total. Original VGG-16 achieves top-1 accuracy 68.34% and top-5 accuracy 88.44%.

We first test semi-NMF-based FC optimization on single FC, fc1, on both CNNs with $r = [8, 16, 32, 64, 128, 256, 512]$. Results are shown in Fig. 1. Proposed algorithm provides similar results on both networks. As r increases, optimized network gives higher prediction accuracies. The increasing saturates rapidly after r exceeds 256. When r reaches 512, both optimized networks retain highly similar performances as original networks.

Table 1 shows detailed comparisons, including results from applying semi-NMF-based FC optimization on both fc1 and fc2 layers. $r_{fc2} = 256$ and 512 are tested. Column *%Whole* shows the percentage of remained weights over the whole original network. Column *%Layer* is the percentage of remained weights over the optimized layers. Process time column records the run time for all tests and iterations for those with retraining.

Optimized network structure is named as fc$i$-r_i-NetName, where fc$i$ is the ith FC that is factorized. There may be multiple FCs factorized. r_i is the selected value of r and NetName is the model name.

Two major conclusions can be drawn. First, on both networks, semi-NMF-based FC optimization adopted on single FC is able to keep the performance with around 50% weights reduced in a controllable and regular pattern. When adopted on fc1 and fc2, semi-NMF-based FC optimization gives around 70% to 80% weights reduction with about $0.4\% - 1\%$

Table 1. Tests of semi-NMF on AlexNet and VGG16 on ImageNet task

Model	*%Whole*	*%Layer*	Top1 Accuracy	Top5 Accuracy	Process Time
Original AlexNet	—	—	56.33%	79.53%	∼2400min, 450K iters
Dynamic Surgery AlexNet [8]	5.7%	5.7%	56.91%	80.01%	∼3800min, 700K iters
fc1-256-AlexNet	43.84%	8.97%	55.59%	79.10%	∼48min
fc1-256-AlexNet retrain	43.84%	8.97%	55.24%	78.97%	∼1060min, 200K iters
fc1-512-AlexNet	49.43%	17.94%	56.42%	79.72%	∼188min
fc1-512-AlexNet retrain	49.43%	17.94%	54.15%	78.05%	∼540min, 100K iters
fc1-1024-AlexNet	60.6%	35.87%	56.71%	79.89%	∼404min
fc1-1024-AlexNet retrain	60.6%	35.87%	56.22%	79.46%	∼1200min, 200K iters
fc1-256-fc2-256-AlexNet	19.41%	10.01%	54.65%	78.61%	∼77min
fc1-512-fc2-256-AlexNet	24.99%	16.21%	55.42%	79.14%	∼224min
fc1-512-fc2-512-AlexNet	28.43%	20.02%	55.90%	79.39%	∼260min
Original VGG16	—	—	68.34%	88.44%	—
fc1-256-VGG16	30.83%	7.27%	67.79%	88.11%	∼65min
fc1-512-VGG16	36.25%	14.54%	68.21%	88.31%	∼405min
fc1-256-fc2-256-VGG16	20.17%	8%	67.31%	87.95%	∼100min
fc1-256-fc2-512-VGG16	21.69%	9.76%	67.4%	87.98%	∼224min
fc1-512-fc2-256-VGG16	25.6%	14.25%	67.7%	88.12%	∼440min
fc1-512-fc2-512-VGG16	27.12%	16.01%	67.89%	88.17%	∼480min

top-1 accuracy drop and $0.2\% - 0.9\%$ top-5 accuracy drop. Second, semi-NMF-based FC optimization requires no retraining and runs $3\times$ to $21\times$ faster on CPU than retraining the same structure and $79\times$ faster than dynamic surgery on GPU.

When building CNN layers on dedicated hardware such as FPGA or ASIC, paralleled multipliers are the units calculating outputs according to inputs and weights. Since there are normally much more input nodes than parallel multipliers, internal buffers are necessary to store partial results for later accumulation. These multipliers require higher bit-depth in order to preserve precision. Our proposed approach can significantly reduce this buffer size to improve inference efficiency. Moreover, factorized matrices require much less bandwidth in between external memories and no extra codec circuits leading to more cost-effective designs.

We calculate FPGA resource savings based on two of our optimized networks fc1-512-AlexNet, and fc1-1024-AlexNet. FPGA model is Xilinx FPGA Kintect 7-325t. Bit-depth is set to 16bit fixed-point for both data and weights. 64 paralleled multipliers for AlexNet. These amounts are small but well-aligned to practical design where more units are reserved for Conv calculations. Comparisons are shown in Table 2.

As can be seen, our proposed algorithm effectively reduces FPGA resources requirement to implement AlexNet. DRAM bandwidth and internal buffer size can be reduced linearly as the reduction in weights. Process cycle can be more efficiently reduced by smaller r.

Table 2. Comparisons of original and optimized AlexNet in terms of FPGA implementation

CNN Model	Process Cycles	DRAM bandwidth (Kbit/image)	Internal buffer size (Kbit)
Original AlexNet	589,832	36,873	138
fc1-512-AlexNet	81,936	6,665	47
fc1-1024-AlexNet	213,008	13,321	60

5 Conclusion and Future work

The complexity of DNN is preventive for mobile device and dedicated hardware implementations. To deal with this issue, we propose a semi-NMF-based FC optimization algorithm to provide highly regular, fully controllable, and retrain-free optimization of DNN structures.

We have tested our algorithm AlexNet and VGG-16 for ImageNet-2012 1000-type image classification task. Results demonstrate that semi-NMF-based FC optimization can achieve highly competitive top-1 and top-5 accuracies compared with reference models and state-of-the-art surgery algorithms. Though semi-NMF keeps more weights, it runs 3-10 times faster on CPU compared with GPU-based retraining. Estimations of hardware resources savings also support our claim.

In terms of future work, one of the most important directions is to customize the error function for semi-NMF. By including characteristics of DNN into the error function, we hope to find better approximations for the original weight matrix.

6 References

[1] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," *CoRR*, vol. abs/1510.00149, 2015. [Online]. Available: http://arxiv.org/abs/1510.00149

[2] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. de Freitas, "Predicting parameters in deep learning," *CoRR*, vol. abs/1306.0543, 2013. [Online]. Available: http://arxiv.org/abs/1306.0543

[3] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in Neural Information Processing Systems 5, [NIPS Conference]*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, pp. 164–171. [Online]. Available: http://dl.acm.org/citation.cfm?id=645753.668069

[4] Y. L. Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems*. Morgan Kaufmann, 1990, pp. 598–605.

[5] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 6655–6659.

[6] X. Zhang, J. Zou, X. Ming, K. He, and J. Sun, "Efficient and accurate approximations of nonlinear convolutional networks," *CoRR*, vol. abs/1411.4229, 2014. [Online]. Available: http://arxiv.org/abs/1411.4229

[7] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," *CoRR*, vol. abs/1506.02626, 2015. [Online]. Available: http://arxiv.org/abs/1506.02626

[8] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient dnns," *CoRR*, vol. abs/1608.04493, 2016. [Online]. Available: http://arxiv.org/abs/1608.04493

[9] P. Paatero and U. Tapper, "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values," *Environmetrics*, vol. 5, no. 2, pp. 111–126, 1994.

[10] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[11] ——, "Algorithms for non-negative matrix factorization," in *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. MIT Press, 2001, pp. 556–562. [Online]. Available: http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf

[12] C. H. Q. Ding, T. Li, and M. I. Jordan, "Convex and semi-nonnegative matrix factorizations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 45–55, Jan. 2010. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2008.277

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, p. 2012.

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[16] S. J. Hanson and L. Y. Pratt, "Comparing biases for minimal network construction with back-propagation," in *Advances in Neural Information Processing Systems 1*, D. S. Touretzky, Ed. Morgan-Kaufmann, 1989, pp. 177–185. [Online]. Available: http://papers.nips.cc/paper/156-comparing-biases-for-minimal-network-construction-with-back-propagation.pdf

[17] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," *CoRR*, vol. abs/1404.0736, 2014. [Online]. Available: http://arxiv.org/abs/1404.0736

[18] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[19] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with cuda," *Queue*, vol. 6, no. 2, pp. 40–53, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1365490.1365500

[20] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cudnn: Efficient primitives for deep learning," *CoRR*, vol. abs/1410.0759, 2014. [Online]. Available: http://arxiv.org/abs/1410.0759