Going Deeper than Deep Learning for Massive Data Analytics under Physical Constraints

Bita Darvish Rouhani[1], Azalia Mirhoseini[2], Farinaz Koushanfar[1]
[1]UC San Diego, [2]Google
bita@uscd.edu, azalia@google.com, farinaz@ucsd.edu

ABSTRACT

Deep Neural Networks (DNNs) are a set of powerful yet computationally complex learning mechanisms that are projected to dominate various artificial intelligence and massive data analytic domains. Physical viability, such as timing, memory, or energy efficiency, are standing challenges in realizing the true potential of DNNs. We propose DeLight, a set of novel methodologies which aim to bring physical constraints as design parameters in the training and execution of DNN architectures. We use physical profiling to bound the network size in accordance to the pertinent platform's characteristics. An automated customization methodology is proposed to adaptively conform the DNN configurations to meet the characterization of the underlying hardware while minimally affecting the inference accuracy. The key to our approach is a new content- and resource-aware transformation of data to a lower-dimensional embedding by which learning the correlation between data samples requires significantly smaller number of neurons. We leverage the performance gain achieved as a result of the data transformation to enable the training of multiple DNN architectures that can be aggregated to further boost the inference accuracy. An accompanying API is also developed, which can be used for rapid prototyping of an arbitrary DNN application customized to the platform. Proof-of concept evaluations for deployment of different imaging, audio, and smart-sensing applications demonstrate up to 100-fold performance improvement compared to the state-of-the-art DNN solutions.

1. INTRODUCTION

Deep learning (a.k.a., deep neural network) has provided a paradigm shift in our ability to comprehend raw data by going beyond traditional linear and polynomial learning approaches [1]. Despite the powerful learning capabilities of deep learning algorithms, the computational overhead associated with DNN models hinders their applicability in resource constrained settings.

There are several advantages in devising resource efficient deep learning methodologies: (i) It benefits scaling of DNN models by maximizing the number of parameters that can be trained within the limits of a given computational or timing budget. (ii) It enables on-chip processing of sensor data acquired on portable embedded platforms such as autonomous vehicles, smart phones, and wearables while evading the requirement to offload personal content to the clouds. (iii) It empowers deployment of several DNN configurations within the confine of the underlying resource provisioning to empirically identify the best model.

We propose DeLight, an *end-to-end* learning framework that enables simultaneous *training* and *execution* of DNN models customized to the pertinent resource provisioning [4]. The resource efficiency of DNN training and execution is explicitly governed by the number of neurons per layer of a DNN architecture. Traditionally, the input layer size of a DNN model is dictated by the feature space size of the incoming data measurements. DeLight proposes the use of a new context- and resource-aware data projection as the primary step to reduce the dimensionality of the input layer of DNNs customized to platform resources and constraints. Our approach subsequently shrinks the dimensionality of the overall DNN model, resulting in significant cost reduction while delivering the same inference accuracy.

The proposed data projection is a pre-processing step that adaptively maps the stream of input data to a corresponding ensemble of lower-dimensional subspaces. The mapping highlights the most informative portions of the data, shrinking the DNN training and execution workload. Our approach leverages the degree of freedom in producing several possible projection subspaces to enable customization with respect to the underlying physical constraints. The achieved resource efficiency in DeLight framework enables concurrent training of multiple DNN topologies which can be combined together to further boost the inference accuracy while adhering to the physical platform resources and constraints.

2. RELATED WORK

The existing DNN realizations for resource-limited platforms are mostly cloud-based models that provide, for example, speech and object recognition in mobile commercial services [1]. A number of earlier works have focused on applying sparsity regularization techniques to reduce the number of parameters in a DNN model [2] or using approximate computing for minimizing the energy cost of evaluating DNNs [3]. Although these approaches yield significant performance improvement in executing DNNs, they do not explicitly optimize physical performance metrics for training deep neural networks as they are mainly post-processing techniques that are adopted after the DNN model has been fully trained. To

CODES/ISSS '16, October 01-07, 2016, Pittsburgh, PA, USA
 ISBN 978-1-4503-4483-8/16/10…$15.00
DOI: http://dx.doi.org/10.1145/2968456.2976766

the best of our knowledge, DeLight is the first to propose an automated pre-processing approach to customize both training and execution of DNNs while optimizing the resulting physical resource consumption on the target platform.

The use of auto-encoders or resource-aware dimensionality reduction techniques have been suggested in the literature for feature extraction or facilitating shallow classification methods such as nearest neighbor, or support vector machine [5, 6]. None of the prior works, however, have customized data projection as a way to achieve physical efficiency for training and execution of DNNs in resource-limited settings. DeLight, for the first time, proposes a systematic approach to optimize the data and DNN circuitry for the underlying physical resources [4].

3. DeLight FRAMEWORK

Here we outline DeLight framework to facilitate *training* and *execution* of DNNs in resource-constrained settings.

3.1 Training Phase

Figure 1 shows the global flow of DeLight for training of a DNN model. DeLight's training phase includes three major steps: (i) Physical profiling (ii) Data projection customization, and (iii) Updating DNN parameters using the projected data embedding.

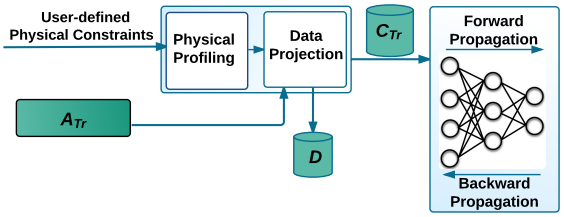

Figure 1: Global flow of DeLight: training phase.

Physical Profiling. To update the DNN parameters, each batch of training data is processed in two steps: (i) Forward Propagation (FP), and (ii) Backward Propagation (BP). Table 1 details the computation and communication cost of each step. We use T_i^{FP} and T_i^{BP} to denote the forward and backward propagation runtimes associated with model i. DeLight characterizes the physical coefficients listed in Table 1 by executing a set of micro-benchmarks that emulate basic operations involved in forward and backward pass. The characterization is a one-time process that incurs a fixed negligible overhead. A similar approach can be used to model energy consumption.

Table 1: DNN Computation and Communication Costs.

Computation and Communication Costs
$T_i^{FP} = \alpha_{flop} \sum_{s=1}^{S-1} n_i^{(s)} n_i^{(s+1)} + \alpha_{act} \sum_{s=1}^{S} n_i^{(s)}$ S: *total number of DNN layers* $n_i^{(s)}$: *number of neurons in the* s^{th} *layer of model* i α_{flop}: *multiply-add cost* α_{act}: *activation function cost*
$T_i^{BP} = 2\alpha_{flop} \sum_{s=1}^{S-1} n_i^{(s)} n_i^{(s+1)} + \alpha_{err} \sum_{s=1}^{S} n_i^{(s)}$ α_{flop}: *multiply-add cost* α_{err}: *propagation error cost*
$T_i^{Comm} = \alpha_{net} + \frac{N_{bits} \times N_{neurons}^{shared}}{BW_i}$ α_{net}: *constant network latency* N_{bits}: *number of signal representation bits* $N_{neurons}^{shared}$: *number of shared neurons* BW_i: *operational memory bandwidth*

DeLight uses the aforementioned cost metrics to determine a bound on the total number of training iterations (N^{itr}), as well as the overall size of the possible networks that can be performed within the specified physical platform limitations. The physical limitations can be either dictated by the arriving rate of sensor measurements or the maximum energy that an embedded platform can provide.

Platform Aware Data Projection. DeLight's data projection is a pre-processing step. It works by projecting the input data $A_{m\times n}$ to an ensemble of lower-dimensional embeddings by seeking the best suited dictionary matrix $D_{m\times l}$ and a corresponding coefficient matrix $C_{l\times n}$ s.t., the required number of neurons per layer of a DNN topology for delivering a target inference accuracy, is minimized. In brief, DeLight optimization is as follows:

$$\underset{l,D,C}{minimize}\ (N_{net})\ s.t.\ \|A - DC\|_F \leq \epsilon \|A\|_F \qquad (1)$$
$$\|y - \tilde{y}\|_F \leq \delta^u$$

where N_{net} indicates the size of the DNN topology that should fit the bounds acquired by platform characterization. Here, y is the ground truth data label, and $\tilde{y}$ is the predicated inference label. $\|\cdot\|_F$ denotes the Frobenius norm, ϵ is an intermediate approximation error that casts input matrix (A) rank, and δ^u implies the user-defined inference error threshold. For a particular dataset, there are several data representations corresponding to different sizes of the projection subspace (l) that satisfy the conditions in Eq. (1). DeLight leverages the degree of freedom in producing several data embeddings to customize costly DNN training and execution to the limits of the energy resources and computational budget. As we experimentally verify, $l \ll m$ can be achieved in real-world large datasets.

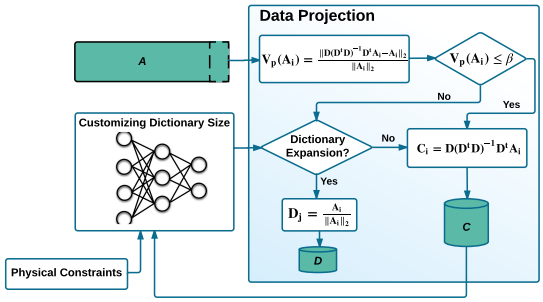

Figure 2: Adjusting projection subspace for evolving data.

Figure 2 illustrates a high-level block diagram of DeLight data projection unit. DeLight adjusts its data projection to accommodate for the new structural trends that might be added as data evolves over time. For each new arriving sample, DeLight first computes an approximation error achieved by projecting the input sample into the subspace spanned by the current dictionary matrix D. If the approximation error is less than a threshold (β), it implies that the existing subspace ensemble is good enough to present that batch of data samples. Otherwise, DeLight expands it dictionary matrix to include the new data structures suggested by the recently added data measurements. To do so, it adds the normalized value of the new samples to the projection matrix B and updates the coefficient matrix C to fit the new projection subspace while avoiding the cost of re-applying the projection for the entire dataset. We use $\beta = 0.1$ in our evaluations presented in Section 4.

Bag of DNN Models. To further boost the inference accuracy, DeLight uses the achieved performance gain to

Table 3: Performance improvement achieved by DeLight over the state-of-the-art deep learning approach.

Application	Baseline			DeLight					Performance Improvement			
	Input DNN	Training Runtime	Training Energy	Customized DNN	Pre-processing Runtime	Pre-processing Energy	Training Runtime	Training Energy	Training Runtime	Training Energy	Execution Runtime	Memory Footprint
Imaging (10%)	$200 \times 230 \times 230 \times 9$	42.1 *min*	2394 *J*	$70 \times 160 \times 160 \times 9$	0.7 *min*	16 *J*	11.6 *min*	417 *J*	3.4×	5.5×	2.6×	2.8×
Smart-Sensing (5%)	$5625 \times 2000 \times 500 \times 19$	138.4 *min*	33948 *J*	$100 \times 500 \times 100 \times 19$	1.7 *min*	91 *J*	6.8 *min*	341 *J*	16.3×	78.8×	108.3×	40.3×
Audio (5%)	$617 \times 50 \times 26$	21.2 *min*	1762 *J*	$78 \times 50 \times 26$	0.6 *min*	21 *J*	8.3 *min*	391 *J*	2.3×	4.2×	6.2×	7.3×

train multiple DNNs each customized for a specific lower-dimensional data embedding. Let N_c denote the number of DNNs trained for a particular task. N_c should be determined such that sum of iterations to train these networks $\sum_{k=1}^{N_c} N_k^{itr}$ as well as the associated number of parameters $\sum_{k=1}^{N_c} \sum_{s=1}^{S-1} n_k^{(s)} n_k^{(s+1)}$ meet the physical constraints.

3.2 Execution Phase

Once the DNN model is trained to meet the desired inference accuracy, there are two main steps to predict the class label for each incoming test data (Figure 3). First, each test sample is projected based on the learned dictionary matrix D. Second, the corresponding coefficient vector C is fed into the trained DNN model to obtain its class label. The execution phase only requires a forward propagation for each transformed data sample. The same computational model is directly applicable to the execution phase by excluding the backward propagation.

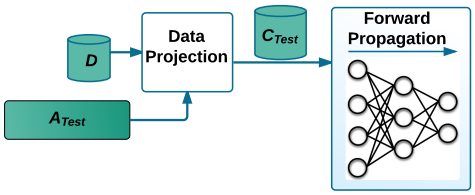

Figure 3: Global flow of DeLight: execution phase.

4. EVALUATIONS

Platform Setup. We use Nvidia Tegra K1 development kit as our hardware platform [7]. The Nvidia TK1 is an embedded processor designed for realizing different computer vision, robotics, security, automotive, and mobile sensing applications. It includes 192 CUDA cores and a 4-Plus-1 quad-core ARM Cortex A15 CPU with a 2 GB memory. We leverage all the available CPU cores on the specified platform to perform data projection using standard Message Passing Interface (MPI), while the DNN training and execution have been conducted using the CUDA cores. We adopt stochastic gradient descent with momentum [8] for back propagation and *Tangent-Hyperbolic* as the non-linear activation function for hidden layers.

Application Data. We evaluate DeLight using (i) Imaging, (ii) Smart-sensing, and (iii) Audio datasets that are key enablers in deployment of different autonomous learning tasks. Table 2 specifies our dataset size for each application.

Table 2: Size of evaluation datasets.

Imaging [9]	Smart-Sensing [10]	Audio [11]
200×54129	5625×9120	617×7797
$86.7MB$	$46.3MB$	$38.5MB$

Performance Improvement. Table 3 details the performance improvement achieved by DeLight compared to the *state-of-the-art implementation* currently used for solving deep learning problems in which Dropout technique is used to avoid over-fitting [12] and the raw data is used for DNN training with no pre-processing. $N_c = 1$ is used for comparison purposes. As shown, the one-time pre-processing overhead of DeLight is amortized when the iterative DL algorithms are run on the projected data.

5. CONCLUSION

We present DeLight, a novel end-to-end framework for realization of sensing and understanding tasks in resource-constrained settings using deep learning models. DeLight adaptively learns and customizes the hybrid structure of the streaming input data to improve system performance in terms of memory footprint, energy consumption, and/or runtime. Our framework provides designers with a user-friendly API for rapid prototyping and evaluation of an arbitrary learning task on multi-core CPU, and/or CPU-GPU platforms. We demonstrate three contemporary practical design experiences on a state-of-the-art IoT platform. Our experiments demonstrate up to 100-fold performance improvement compared to the best known prior solutions.

6. REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, 2015.

[2] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky, "Sparse convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[3] J. Kung, D. Kim, and S. Mukhopadhyay, "A power-aware digital feedforward neural network platform with backpropagation driven approximate synapses," in *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 2015.

[4] B. Rouhani, A. Mirhoseini, and F. Koushanfar, "Delight: Adding energy dimension to deep neural networks," in *International Symposium on Low Power Electronics and Design (ISLPED)*. ACM, 2016.

[5] A. Mirhoseini, B. Rouhani, E. Songhori, and F. Koushanfar, "Performml: Performance optimized machine learning by platform and content aware customization." *Design Automation Conference (DAC)*, 2016.

[6] B. D. Rouhani, E. M. Songhori, A. Mirhoseini, and F. Koushanfar, "Ssketch: An automated framework for streaming sketch-based analysis of big data on fpga," in *Field-Programmable Custom Computing Machines (FCCM), 2015 IEEE 23rd Annual International Symposium on*. IEEE, 2015, pp. 187–194.

[7] https://developer.nvidia.com/jetson-tk1, "Jetson tk1," 2015.

[8] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th international conference on machine learning (ICML-13)*, 2013.

[9] http://www.ehu.es/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes, "Remote sensing," 2015.

[10] https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones, "Uci machine learning repository," 2015.

[11] https://archive.ics.uci.edu/ml/datasets/isolet, "Uci machine learning repository," 2015.

[12] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, 2014.