DeLight: Adding Energy Dimension To Deep Neural Networks

Bita Darvish Rouhani[1], Azalia Mirhoseini[2], Farinaz Koushanfar[1]
Electrical and Computer Engineering Department
[1]UC San Diego, [2]Rice University
bita@uscd.edu, azalia@rice.edu, farinaz@ucsd.edu

ABSTRACT

Physical viability, in particular energy efficiency, is a key challenge in realizing the true potential of Deep Neural Networks (DNNs). In this paper, we aim to incorporate the energy dimension as a design parameter in the higher-level hierarchy of DNN training and execution to optimize for the energy resources and constraints. We use energy characterization to bound the network size in accordance to the pertinent physical resources. An automated customization methodology is proposed to adaptively conform the DNN configurations to the underlying hardware characteristics while minimally affecting the inference accuracy. The key to our approach is a new context and resource aware projection 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 projection to enable the training of different DNN architectures which can be aggregated together to further boost the inference accuracy. Accompanying APIs are provided to facilitate rapid prototyping of an arbitrary DNN application customized to the underlying platform. Proof-of-concept evaluations for deployment of different visual, audio, and smart-sensing benchmarks demonstrate up to 100-fold energy improvement compared to the prior-art DL solutions.

CCS Concepts

• **Computing methodologies** → **Neural networks;** • **Hardware** → **Energy metering;**

1. INTRODUCTION

Deep neural networks are a set of powerful yet computationally complex learning mechanisms that have provided a paradigm shift in the emerging field of machine learning and data inference [1]. Inspired by neural activities in the brain, DNNs model data through several successive layers of complex and non-linear features. While the non-linear and sophisticated nature of DNNs empowers achieving superb inference capability, it inevitably brings a new set of challenges concerning the scalability and energy efficiency.

Devising resource efficient DNN methodologies is particularly of interest from two distinct perspectives. On the one hand, reducing the overhead of DNNs benefits distributed training and execution of large DNNs so that more parameters could be learned on a single machine. As such, fewer computing nodes are required to perform a particular learning task using cloud servers. On the other hand, in many deep learning applications, training DNN models requires continuous processing of the evolving data constantly acquired by different sensors available on Internet-of-Thing (IoT) devices. Customizing DNNs to fit the limits of the underlying hardware enables on-chip processing of sensing data on constrained embedded platforms such as autonomous vehicles, smart phones, and wearables. This customization, in turn, evades the requirement to offload personal content to the clouds, which is especially important in real-world settings where having consistent access to the cloud servers can be highly expensive or even infeasible.

We propose DeLight, a *Light*weight automated *Deep* learning framework that enables efficient *training* and *execution* of DNNs customized to the underlying *energy resources* and *application data*. Our key observation is that the dimensionality of input data to a deep neural network has a direct impact on the overall size of the network, which subsequently dictates resource utilization for training and execution of the pertinent DNN model. To fulfill this observation, we introduce a new projection of data to an *ensemble of lower-dimensional subspaces* that is both aware of the *data context* and *platform resource constraints*. The data projection yields significant DNN compaction while minimally affecting the inference accuracy.

The proposed projection is an adaptive pre-processing methodology that incurs linear computational complexity within a guaranteed approximation error. It 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 customizing DeLight with respect to the energy constraints imposed by the platform and/or application data. We provide a systematic methodology to perform platform customization as well as projection error tuning for a target inference accuracy.

DeLight is devised based on a HW/SW co-design approach. Our proposed performance cost reduction approach enables training multiple DNN topologies within the confine of the pertinent computational budget and energy resources. The multiple models can be combined together to further boost the inference accuracy for a target application and platform. We present practical design experiences of using DeLight for various smart-sensing and visual understanding tasks on the Nvidia Tegra K1 processor. Tegra K1

ISLPED '16, August 08-10, 2016, San Francisco Airport, CA, USA

DOI: http://dx.doi.org/10.1145/2934583.2934599

is a popular embedded processor, widely used in contemporary IoT-enabled cars, robots, and smart phones [2]. To the best of our knowledge, no prior solution for *simultaneous DNN training and execution* on an embedded system has been reported in the literature. We also design multiple APIs that can implement DeLight on an arbitrary multi-core CPU and/or CPU-GPU system for rapid prototyping of different learning tasks customized to the platform. The explicit contributions of this paper are as follows:

- Proposing DeLight, the first end-to-end automated framework that explicitly targets energy efficiency for training and execution of deep neural networks. DeLight adaptively reduces the data and circuit footprint for implementing DNNs, which translate to meaningful savings in memory, runtime, power, and energy.
- Creating performance models for quantifying the computational costs of DNN's training and execution. The quantified cost metrics are used to conform the energy performance to the underlying platform.
- Introducing a context and resource aware data projection as a pre-processing step to reduce the dimensionality of a DNN input layer. Our methodology subsequently shrinks the dimensionality of the overall DNN, resulting in significant performance cost reduction while delivering the same inference accuracy.
- Devising an automated optimization approach and accompanying APIs to ensure ease of adaptation to different set of physical constraints imposed by the platform and/or application data. Evaluations of different visual, audio, and smart-sensing benchmarks demonstrate up to two orders-of-magnitude energy improvement compared to the prior-art DNN solutions.

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 [3]. A number of earlier works have focused on applying sparsity regularization techniques to reduce the number of parameters in a DNN model [4] or using approximate computing for minimizing the energy cost of evaluating DNNs [5]. Although these approaches yield significant performance improvement in executing DNNs, they do not explicitly optimize the energy metric or the impact of platform constraints in training deep neural networks as they are mainly post-processing techniques that are adopted after the DNN model has been fully trained. To 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 energy 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 [6, 7, 8]. None of the prior works, however, have customized data projection as a way to achieve energy 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 energy resources. Our customization leverages the trade-off between the output solution variations and system performance to provide an efficient approach for realization of various sensing and visual understanding tasks within the limits of the energy resources and computational budget.

3. PRELIMINARIES

Deep learning is an important area of machine learning that has provided a significant leap in our ability to comprehend raw data in a variety of complex learning tasks. Empirical experimentations have been the driving force behind the success of deep learning mechanisms with theoretical proofs explaining its behavior yet remaining mainly elusive [1]. Deep Neural Networks (DNNs) and Convolutional Neural Networks (CNNs) are the two main topologies widely used in deep learning domain [9]. These two types of neural networks differ from one another in a sense that a CNN architecture includes additional convolution layers on top of fully-connected networks that are the foundation of DNN topologies. As such, CNNs are better-suited for interpreting data measurements with strong local connectivity (e.g., visual data) while DNNs pose a more generic architecture that is directly applicable to various type of smart-sensing datasets. In this paper, we focus on DNNs to ensure ease of adoption for realization of different sensory applications.

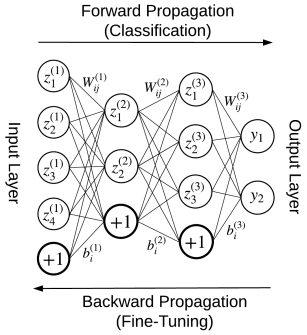

Figure 1: Schematic depiction of a 4-layer DNN.

Figure 1 demonstrates a 4-layer DNN topology. Training DNN models includes two main steps that are *iteratively* performed for multiple rounds of reprocessing input data until a certain level of inference accuracy is obtained: *forward propagation* and *backward propagation*. In the forward path, the model's prediction for the given input data is computed based on the current states of the DNN parameters per Equation (1):

$$z_i^{(s+1)} = f(\sum_{j=1}^{n^{(s)}} W_{ij}^{(s)} z_j^{(s)} + b_i^{(s)}), \quad (1)$$

where $n^{(s)}$ denotes the number of neurons (units) in the layer s and $z_i^{(s)}$ indicates the state of neuron i in the s^{th} layer. Here, we use $W_{ij}^{(s)}$ to specify the weight associated with the edge connecting unit j in layer s and unit i in the layer $s+1$. $b_i^{(s)}$ indicates the additive bias used for computing the state of unit i in layer $s+1$ and $f(.)$ denotes the non-linear activation function. In the backward path, a *gradient-based* approach is used to fine-tune network parameters in accordance to a specified loss function that is defined to capture the difference between network inferences (predictions) and the ground-truth labeled data. As such, for each batch of training data the network parameters are updated as follows:

$$W_{ij}^{(s)} = W_{ij}^{(s)} - \eta \frac{1}{b_s} \sum_{k=1}^{b_s} \frac{\partial E^{(s)}}{\partial W_{ij}^{(s)}}, \quad (2)$$

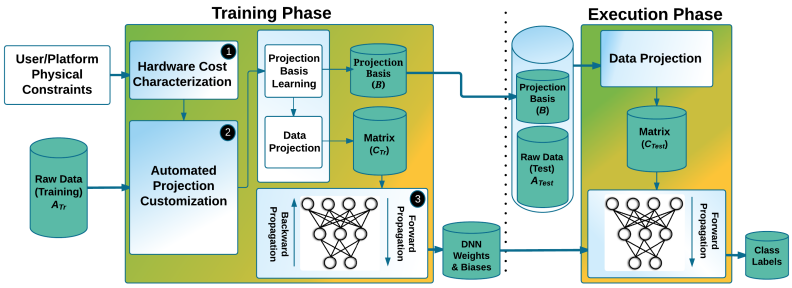

Figure 2: Global flow of DeLight: The raw input data is projected to an ensemble of lower-dimensional embeddings (matrix C) using a new energy aware methodology that is customized to benefit the subsequent DNN training and execution task.

where b_s is the data batch size, $(\partial E^{(s)}/\partial W_{ij}^{(s)})$ represents the propagated errors in the layer s, and η denotes the learning rate. Note that backward propagation is a step involved in the training of a DNN model while the DNN execution only includes a forward path through data samples.

4. DeLight GLOBAL FLOW

The energy 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 deep neural network 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 the pertinent resource provisioning and application data. Our approach subsequently shrinks the dimensionality of the overall DNN model, resulting in significant performance cost reduction while delivering the same inference accuracy.

DeLight's data projection is a pre-processing step. It works by projecting the original input data $A_{m\times n}$ to an ensemble of lower-dimensional embeddings by seeking the best suited projection basis $B_{m\times l}$ and a corresponding coefficient matrix $C_{l\times n}$ such that the required number of neurons per layer of a DNN topology for delivering a target inference accuracy, is minimized. In other words, DeLight aims to solve the following objective function:

$$\underset{l,B,C}{minimize}\ (N_{net})\ s.t.\ \|A - BC\|_F \leq \epsilon\|A\|_F \qquad (3)$$

$$\|y - \tilde{y}\|_F \leq \delta^u$$

where N_{net} indicates the size of the underlying DNN topology, y is the ground truth data label, and $\tilde{y}$ is the predicated inference label. Here, $\|\cdot\|_F$ denotes the Frobenius norm, ϵ is an intermediate approximation error that casts the rank of the input matrix A, and δ^u implies the user-defined inference error threshold. We use the parameter l to point out the size of the ambient space spanned by the learned projection basis B, and parameters m and n to denote the original feature space size and total number of data samples, respectively. Note that for a particular data collection, there are several data representations corresponding to different sizes of the projection subspace (l) that satisfy the conditions in Equation (3). DeLight leverages the degree of freedom in producing several possible data embeddings to customize costly DNN training and execution to the limits of the energy resources and computational budget. We show that $l \ll m$ can be achieved in real-world large datasets.

The overall flow of DeLight for training and execution of DNN models is illustrated in Figure 2. DeLight's training phase includes three major steps: (i) Hardware cost characterization, (ii) Automated projection customization, and (iii) Training the customized DNN architecture using the projected data embedding. In the execution phase, DeLight uses the trained DNN model to classify newly arriving samples. In Section 5.1, we propose metrics to quantify the computing cost of DNN training and execution, which directly impacts the resulting energy consumption. Our automated customization approach (Section 5.2) uses the performance cost models to tailor the subsequent learning task in accordance to the underlying energy and computational constraints while minimally affecting the inference accuracy. We also provide accompanying APIs to facilitate automation and adoption of DeLight for rapid prototyping of an arbitrary learning task using multi-core CPUs, and/or CPU-GPU nodes. Our APIs are built to work with Theano, a highly efficient and popular deep learning library [10].

5. DeLight FRAMEWORK

Algorithm 1 outlines the pseudocode of DeLight framework for training and execution of DNN models.

Algorithm 1 DeLight's Training & Execution Steps

Inputs: **Measurement matrices** (A_{Tr}, A_{Test}), **Training labels** (y_{Tr}), **Inference error threshold** (δ^u), **Tuning portion** (p), **Initial DNN network size** (N^u_{net}), **and Energy constraints** $(E^u = [Energy^u_{Tr},\ Energy^u_{Exe}])$

Output: **Projection basis** B, **DNN parameters** DNN_{param}, **and Predicted class labels** (y_{Test}).

1: $B \leftarrow$ **empty**
2: $[\tilde{A}_{Tr}, \tilde{y}_{Tr}] \leftarrow DataPartitioning(p, A_{Tr}, y_{Tr})$
3: $HW_{spec} \leftarrow HWCharacterization()$
4: $[N^{max}_{net}, T] \leftarrow EnergyCharacterization(HW_{spec}, N^u_{net}, E^u)$
5: $[\tilde{N}_{net}, l] \leftarrow NetCustomization(\tilde{A}_{Tr}, \tilde{y}_{Tr}, N^{max}_{net})$
6: $DNN^{init}_{param} \leftarrow RandomInitialization(\tilde{N}_{net})$
7: $[C_{Tr}, B] \leftarrow DataProjection(A_{Tr}, B, l)$
8: $DNN_{param} \leftarrow DNN(C_{Tr}, y_{Tr}, DNN^{init}_{param}, \delta, T)$

9: $i \leftarrow 0$
10: **while** *(true)* **do**
11: $\quad C^i \leftarrow DataProjection(A^i_{Test}, B)$
12: $\quad y^i_{Test} \leftarrow DNN_{forward}(C^i, DNN_{param})$
13: $\quad i \leftarrow i + 1$
end while

The core of our APIs is Algorithm 1. Lines 1-8 represent the steps involved in training a DNN model using DeLight, and lines 9-13 outline the required computational operations in the execution phase. Our APIs take the pertinent energy constraints ($Energy^u_{Tr}$ and $Energy^u_{Exe}$) into consideration and customizes the framework accordingly to adhere to the physical limitations imposed by the platform and/or the learning application. The user-defined algorithmic inputs of our APIs include: the raw data matrix A, training labels y_{Tr}, inference error threshold δ^u, and the initial DNN topol-

ogy N^u_{net}. N^u_{net} is a vector of integers whose first element indicates the feature space size of the raw input data (m), and its last element shows the desired number of inference classes. Any number in between the first and last elements denotes an approximation of the number of neurons per hidden layer in the chosen DNN topology. The outputs of our APIs are the learned projection basis B, DNN parameters DNN_{param}, and predicted class labels for test data y_{Test}.

DeLight finds an estimation of the underlying hardware specifications by running a *micro-benchmark* that emulates basic operations involved in the forward and backward propagation. It uses the acquired platform specifications to provide bounds on the total number of neurons per DNN model (N^{max}_{net}) and the number of training iterations (T) that can be performed within the confine of the given energy resources (Section 5.1). DeLight leverages the output of energy characterization as guidelines to customize the data and DNN circuitry to fit the hardware platform while minimally affecting the inference accuracy. It uses a small subset of the input data (e.g., $p = 5\%$) to model the underlying data dependencies and approximate the solution of Equation (3) (Section 5.2). Once DNN parameters are trained to meet the desired level of inference accuracy, the execution process only includes a forward path (Equation (1)) for each projected data sample. Note that for DNN execution, the input test data is transformed to the new subspace using the already learned projection basis B. Steps 7-8 can be repeated to adjust the DNN model over time and accommodate for newly added training samples.

5.1 DeLight Energy Characterization

The overall energy consumption for training a DNN model can be characterized as:

$$Energy^{Tr} \simeq T(Energy^{FP} + Energy^{BP}), \quad (4)$$

where T denotes the total number of training iterations for each batch of input data, and $Energy^{FP}$ and $Energy^{BP}$ indicate the energy cost of Forward Propagation (FP) and Backward Propagation (BP), respectively. Note that this performance model is reduced to $Energy^{FP}$ for DNN execution. In the following, we model the energy cost of both forward and backward propagations in terms of the dedicated number of arithmetic operations and communications.

Computation Cost. The cost of arithmetics is a direct function of the number of floating-point operations involved in the training and execution of a DNN model. In the forward path, the state of each neuron is computed according to Equation (1). For processing a data batch of size b_s, DeLight approximates the energy cost of a forward propagation as $Energy^{FP}_{comp} \simeq b_s(\alpha_{flop}\sum_{s=1}^{S-1} n^{(s)}n^{(s+1)} + \alpha_{act}\sum_{s=1}^{S} n^{(s)})$, where S denotes the total number of hidden layers, and $n^{(s)}$ is the number of neurons in the layer s. α_{flop} indicates the energy cost of one multiply-add operation, and α_{act} is the energy cost of computing the non-linear activation function $f(.)$ for a scalar on the target platform. Commonly used activation functions include *Logistic Sigmoid*, *Tangent-hyperbolic (Tanh)*, and *Rectified Linear Unit (ReLu)*.

In the backward path, DeLight approximates the energy cost of processing a data batch of size b_s as $b_s(2\alpha_{flop}\sum_{s=1}^{S-1} n^{(s)}n^{(s+1)} + \alpha_{err}\sum_{s=1}^{S} n^{(s)})$, where the first term reflects the cost of multiplications and additions that should be performed to update the network parameters and the second term denotes the energy cost of computing the pertinent propagation error in each neuron. Note that in a distributed setting, the aforementioned performance models represent the cost of computations for a local network. Thereby, the overall $Energy^{FP}_{comp}$ and $Energy^{BP}_{comp}$ should be computed as the sum of all local energy models.

Communication Cost. In a distributed or multi-core setting, the number of required communications in each forward and backward path is dominated by the number of shared weights between different nodes. Let us denote the number of shared parameters by $N^{shared}_{weights}$. The communication cost of forward and backward propagations can be modeled as $\gamma N^{shared}_{weights}$, where γ characterizes the impact of the operational memory bandwidth in the target platform. DeLight finds an estimation of the effective α_{flop}, α_{act}, α_{err}, and γ by running a micro-benchmark on the target platform.

As shown, the energy consumption for DNN training and execution is governed by the overall size of the underlying DNN model. DeLight uses the proposed cost metrics to determine a bound on the total number of training iterations (T), as well as the overall size of the possible networks (N^{max}_{net}) that could be performed within the specified energy constraints for DNN training ($Energy^u_{Tr}$) and execution ($Energy^u_{Exe}$). These energy constraints can be either dictated by the arriving rate of sensor measurements or the maximum power that an embedded platform can provide. Our performance model is applicable to DNN training and execution both on distributed and multi-core platforms. Due to the space limits, we only include the evaluation results for an embedded multi-core processor in Section 6.

5.2 DeLight Automated Customization

Given the non-linear ad-hoc nature of DNNs, it is a computationally complex problem to find the explicit solution of the objective function defined in the Equation (3). Many complex large datasets that are not inherently sparse or low-rank can be modeled by an ensemble of lower-rank subspaces [8]. Let A be a data with an ensemble of lower-rank data embeddings. As we experimentally verify in Section 6, a close estimation of the underlying data dependencies can be achieved by using a small random subset of the data. We leverage this data property to find the best-suited projection basis B and the corresponding data embedding C optimized for the pertinent DNN configuration imposed by the platform resources. In particular, DeLight approximates the solution of Equation (3) by using a small subset of the raw input data to interpret the convergence rate corresponding to different data embeddings (e.g., different projection subspace sizes l) and customize the framework for the underlying energy resources while minimally affecting the accuracy.

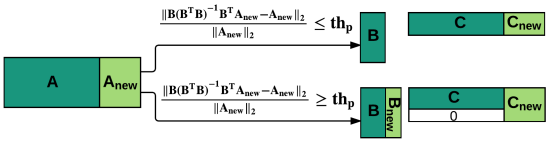

Figure 4: Adjusting projection subspace for dynamic data.

Dynamic Data. In many applications, the dataset A may dynamically grow over time. DeLight adjusts its data projection to accommodate for the new structural trends that might be added by the newly arriving data samples. As shown in Figure 4, DeLight first computes the approximation error achieved by projecting the newly added samples

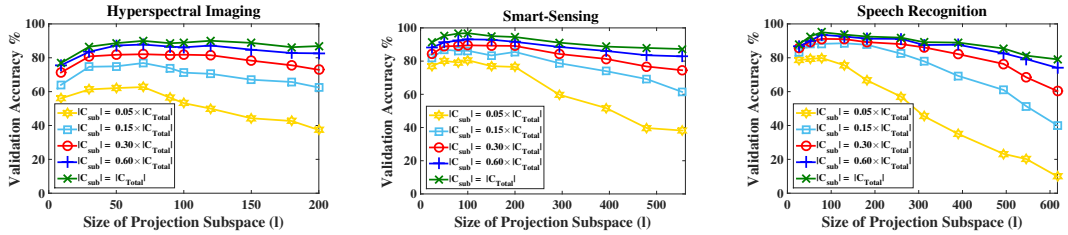

Figure 3: An effective estimation of the underlying data structure can be obtained using only a small subset of the data.

into the subspace spanned by the current projection basis B. If the approximation error is less than a threshold (th_p), it implies that the existing subspace ensemble is good enough to present that batch of data samples. Otherwise, DeLight expands the projection basis 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 $th_p = 0.1$ in our evaluations presented in Section 6.

Model Aggregation. To further boost the inference accuracy, DeLight uses the achieved performance gain to train multiple DNN models each customized for a specific lower-dimensional embedding of the data. Let N_c denote the number of DNNs trained for a particular task. N_c should be determined such that the sum of iterations to train these networks $\sum_{k=1}^{N_c} T_k$ 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 hardware physical resource constraints.

6. EVALUATIONS

Hardware Setup. In our evaluations, Nvidia Tegra K1 development kit is used as the hardware platform [2]. 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 [11] for back propagation and *Tanh* as the activation function for hidden layers. $N_c = 1$ is used in our evaluations.

Application Data. We based our evaluations to perceive knowledge from three classes of data.

(i) Imaging [12]: Hyperspectral imaging is a promising tool for classifying man-made and naturally occurring materials on the earth's surface using reflectance spectra. Given the large size of hyperspectral data, it is highly desirable to locally classify these images using system-on-chips of satellites rather than transferring large amounts of data to the earth's stations. We target hyperspectral imaging classification as one of our practical design experiences.

(ii) Smart-Sensing [13]: Analysis of smart-sensing data collected by many sensors embedded in IoT platforms, such as accelerometers and gyroscopes, is a common step in the realization of various learning tasks. We base one of our applications to evaluate DeLight's performance in analyzing such data to classify daily and sport activities.

(iii) Speech Recognition [14]: Processing audio data is another indispensable mechanism involved in devising different voice activated learning tasks that appear in mobile sensing, robotics, and computer vision applications. As our third application, we corroborate DeLight's practicability to analyze audio datasets. Our data consists of approximately 1.25 hours of speech collected by 150 speakers.

6.1 DeLight Physical Performance

Pre-processing Overhead. The size of the ambient space spanned by the projection matrix B is one of the key tunable parameters that characterizes DeLight's energy performance and accuracy. Figure 3 illustrates the validation accuracy as a function of the projection subspace size l for different subsets of each dataset. As shown, there is an optimal l that maximizes the inference accuracy for a given input data and platform. DeLight customizes the projection subspace size l for the underlying physical constraints by using only a small subset of the data as explained in Section 5.2. Note that a higher inference accuracy is obtained when more data samples are used for training. This in turn enables making the acquired model more accurate as training data evolves over time. We use $|C_{sub}|$ to denote the number of samples in each subset of the data and $|C_{Total}|$ to indicate the total number of training samples.

Table 1: DeLight's pre-processing energy and time overhead.

Application ($m \times n$)	Imaging (200×54129)		Smart-Sensing (5625×9120)		Speech Recognition (617×7797)	
	Time	Energy	Time	Energy	Time	Energy
Tuning	34.7 s	13.8 J	91.4 s	82.26 J	31.1 s	18.6 J
Projection	6.1 s	2.5 J	11.9 s	9.52 J	4.7 s	2.5 J
Overal	40.8 s	16.3 J	103.3 s	91.78 J	35.8 s	21.1 J

Table 1 shows DeLight's total pre-processing time and energy overhead, which accounts for both tuning the algorithmic parameters and projection of data. The tuning is performed using a small random subset of each dataset. Once the optimal parameter l is found in each application, the remaining data including training and testing samples are projected using the customized projection basis. As such, the data projection incurs a linear overhead with respect to the number of data samples n. The selected value l for the imaging, smart-sensing, and audio data is 70, 100, and 78, respectively (Figure 3). The dimensionality m reported in Table 1 denotes the original feature space size in each application. Note that hardware characterization is a one-time process with a fixed, negligible overhead.

Performance Improvement. Table 2 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 [15] and the raw data is used for DNN training with no pre-processing.

To process the hyperspectral data within 10% inference error, it takes 42.1 minutes and 2394 Joules to train a DNN of size $(200 \times 230 \times 230 \times 9)$ using the state-of-the-art Theano implementation. DeLight reduces this time and energy over-

Table 2: Performance improvement achieved by DeLight over the state-of-the-art deep learning approach.

Application	Training Runtime	Training Energy	Execution Runtime	Memory Footprint
Imaging	3.6×	5.7×	2.6×	2.8×
Smart-Sensing	20.9×	99.6×	108.3×	40.3×
Speech Recognition	2.8×	4.3×	6.2×	7.3×

head to less than 11.6 minutes and 417 Joules on the same platform by reducing the DNN size to $(70 \times 160 \times 160 \times 9)$ while delivering the same level of accuracy. In the same context, 2.8 times reduction in the pertinent memory footprint is achieved. We note that the inference accuracy achieved by DeLight is comparable to that of a CNN-based implementation; e.g., authors in [16] report achieving 9.8% inference error in classifying the hyperspectral data using a CNN-based model. However, DeLight's data and platform aware approach allows us to train and use a DNN model with 13.4 times less parameters to deliver the same inference accuracy.

To train a DNN of size $(5625 \times 2000 \times 500 \times 19)$ for classifying daily and sport activities within 5% inference error, it takes 138 minutes and 33948 Joules using the prior-art solution. However, our pre-processing approach makes it feasible to deliver the same level of accuracy with less than 7 minutes and 341 Joules on the same platform by scaling down the required number of neurons per DNN layer $(100 \times 500 \times 100 \times 19)$. This model compaction enables DeLight to gain *two orders-of-magnitude* (99.6) savings in energy consumption and 40.3 times reduction in the memory footprint. As mathematically discussed in Section 5, our customization results in more improvement when applied to datasets with a larger number of input features m.

In processing the audio data within 5% inference error, DeLight gains 2.8 and 4.3 times savings in the training runtime and energy consumption. In the same context, DeLight also achieves 7.3 times reduction in the required memory footprint. The selected topology for this dataset is a 3-layer DNN with 50 neurons per hidden layer and 26 units in the output layer. For this application, $l = 78$ is used for data projection, while the raw samples each have 617 features.

Discussion. One may speculate that classic data projection methods such as Principal Component Analysis (PCA) or Singular Value Decomposition (SVD) can replace our pre-processing step and result in comparable performance. However, in our evaluations, we observed poor performance when PCA is used as the primary dimensionality reduction technique before training the DNN. For instance, a test error of 72.6% is obtained with a DNN of size $(100 \times 500 \times 100 \times 19)$ when the smart-sensing data is transformed using PCA (DeLight achieves 5% inference error with the same DNN configuration). Besides the poor performance in terms of accuracy, there are three main limitations with PCA or SVD: (i) The computational complexity of such methods is quadratic, which makes them costly choices for projecting large datasets. (ii) These data projection techniques are not well-suited for evolving data where it dynamically grows over time. (iii) Unlike our data projection, such methods are oblivious to the coarse-grained parallelism existing in the data. Thereby, they are not flexible to be customized for different energy constraints imposed by the platform.

7. CONCLUSION

We present DeLight, an automated end-to-end framework that is devised to perform DNN training and execution customized to limited energy and computational resources. It adaptively leverages data geometry and platform customization to improve energy efficiency using our proposed data projection as a pre-processing step. The accompanying APIs provided by DeLight enables data scientists to easily adopt our framework for rapid prototyping of an arbitrary learning application 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 energy improvement compared to the best known prior solutions.

Acknowledgments. This work was supported in parts by the Office of Naval Research grant (N00014-11-1-0885).

8. REFERENCES

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

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

[3] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer *et al.*, "Recent advances in deep learning for speech research at microsoft," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] A. Mirhoseini, E. Dyer, E. Songhori, R. Baraniuk, and F. Koushanfar, "Rankmap: A platform-aware framework for distributed learning from dense datasets," *Preprint:1503.08169*, 2015.

[9] L. Deng and D. Yu, "Deep learning: methods and applications," *Foundations and Trends in Signal Processing*, vol. 7, no. 3–4, 2014.

[10] J. Bergstra, O. Breuleux, G. Bastien, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: a cpu and gpu math expression compiler," in *Proceedings of the Python for scientific computing conference (SciPy)*, 2010.

[11] 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.

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

[13] https://archive.ics.uci.edu/ml/datasets/Daily+and+Sports+Activities, "UCI machine learning repository," 2015.

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

[15] 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.

[16] W. Hu, Y. Huang, L. Wei, F. Zhang, and H. Li, "Deep convolutional neural networks for hyperspectral image classification," *Journal of Sensors*, vol. 501, 2015.