DeepTrim: An automated platform-aware domain-customized framework for Deep Learning

Proposal

1 Introduction

Deep Learning (DL) methodologies are known to achieve superior inference for several important problems by moving beyond traditional linear or polynomial analytical machine learning. Despite DL's powerful learning capability, the associated high computational overhead (i.e., timing, storage, and energy) limits its performance in resource-constrained settings. For vision applications, in particular, the high dimensionality of the input data and the complexity of the underlying tasks hinder efficient implementation of the existing standard algorithms. However, due to intrinsic correlations present in visual data, there exists a broad category of optimization schemes that could potentially be investigated towards reducing the complexity of DL-models for vision applications.

Practical DL models incur a large memory footprint, requiring extensive communication with power-hungry external memories which, in turn, severely degrades the performance. In addition, the time required for building or updating a model is critical since DL models are largely experimental: The less time it takes to construct and evaluate one model, the more models can be empirically compared. Thus, there is a need for new DL methods that minimize the performance cost on the target platform such as memory access and inter-processor communications. The broad usage of DL on various vision applications and diverse platforms ranging from resource-constrained embedded devices to data centers signifies the importance of customizing the algorithms to the pertinent data and platform settings.

A number of recent studies have focused on reducing the DL memory footprint, execution time, or energy consumption. On the one hand, content-aware approaches use techniques such as dimensionality reduction [1] and data encoding [2] to leverage intrinsic characteristics of the data in order to simplify a DL model. On the other hand, platform-aware methods in computer engineering map a given algorithm to a platform, but typically do not change the model or data abstraction [3]. Although these works demonstrate notable improvement in the realization of the pertinent DL model, to the best of our knowledge, no prior work provides a systematic automated solution that can simultaneously customize DL algorithms with respect to the data and the platform.

This proposal presents DeepTrim, a novel holistic end-to-end automated data-dependent framework for efficient realization of DL algorithms for important classes of computer vision applications. DeepTrim leverages customization techniques to optimally tailor DL algorithms to the underlying data geometry and platform constraints for achieving resource efficiency while training, executing, and dynamically updating of DL models. DeepTrim addresses five specific challenges associated with the studied methods: (i) Reduction in execution time for learning, inference, and model updating; (ii) Modifications for resource efficiency without losing accuracy for training, inference, and model updating; (iii) Optimizing the performance of execution for pre-trained networks on 3 families of platforms (CPUs, CPU-GPU, and FPGA) by data/platform-aware customization; (iv) Online and infield updating of pre-trained models to cope with the dynamics in the data; and (v) Automation of the aforementioned methodologies to ease adoption and customization. In particular, DeepTrim is devised based on six modular but inter-linked thrusts:

- Developing automated resource profiling benchmark suits for various hardware platforms including multi- and many-core CPUs, GPUs, and FPGAs.

- Designing a set of automated context and platform-aware pre-processing techniques to improve performance by systematic alignment of data to lower-dimensional subspaces.
- Incepting the idea of tensor decomposition to minimize the required resource provisioning for DL execution in accordance with the rank of underlying DL kernels (layers).
- Proposing a novel DL graph traversal methodology to reduce the communication overhead for moving data to/from the main memory and message-passing between various cores on the target platform.
- Providing DL-specific compression methodologies and parameter encoding techniques for efficient DL execution on constrained embedded systems with severe memory and computational budget constraints.
- Devising accompanying end-to-end APIs to ensure DeepTrim's ease of adoption on three classes of platforms including CPUs, GPUs, and FPGAs. Our proposed methodology is extensible to other families of computing hardware.

The proposed techniques leverage PI's strong background and prior relevant work in engineering and design automation of machine learning algorithms. The PI's group has reported the best-ever performance optimization for linear ML algorithms both in terms of finding new theoretical bounds, and new algorithms that reach the bounds [4, 5]. The PI's group is the first to report training of DL on mobile platforms [1]. The PI has a proven track record in technology transfer, and in outreach to diverse groups, especially women and minorities.

2 State of the Art

As documented by several recent studies, DL approaches have demonstrated superb learning capabilities in a variety of understanding tasks including classification, feature extraction, pattern recognition, and regression [6, 7]. The relevant literature on the subject of DL is vast and spans several fields, from computer vision, machine learning, and statistical signal processing, to information theory, natural language processing, and biology, to computer architecture and hardware acceleration. Surveying the fast expanding DL literature is outside the scope of the present proposal. Instead, we focus on the most relevant results. Due to the cross-disciplinary nature of this proposal, we further elaborate on the specific relevant work in each subsection of the proposal.

We note that an important challenge set in the DL is related to mapping of computations to increasingly parallel multi-core and/or heterogeneous modern architectures. The cost of computing and moving data to/from the memory and inter-cores is different for each computing platform. The existing solutions for performing DL can be broadly divided into two distinct categories: (i) Data scientists, on the one hand, have been mainly focused on optimizing DL efficiency through data and neural network manipulations and pruning with little or no attention to the platform characterization [8, 9, 10]. (ii) Computer engineers, on the other hand, have developed hardware-accelerated solutions for an efficient domain-customized realization of DL models oblivious to data geometry, e.g., [11, 12].

Although existing works demonstrate significant improvement in the realization of particular DL models, to the best of our knowledge, none of the prior works have devised an end-to-end solution where the higher level data-dependent signal projection can benefit the physical performance. Our hypothesis is that devising platform-aware signal transformations greatly improves the performance of the learning algorithm by holistic customization to the limits of the hardware resources, data, and DL neural network.

3 Statement of the work

In this Section, we first describe our target applications that belong to the computer vision domain. The global overview and detailed explanations of DeepTrim framework are discussed in Sections 3.3 through 3.5. Each subsection concludes with a set of enumerated goals and deliverables. The schedule of the deliverables is outlined in Section 5.

3.1 Applications and Evaluation Metrics

Applications: DeepTrim focuses on four important DL vision applications and their implementation on real testbeds:

• **Visual Object Tracking:** Object tracking is one of the most important tasks in numerous computer vision applications. The use of convolutional neural networks (CNNs) for the purpose of object tracking has shown promising results in the existing literature [13, 14, 15, 16]. However, the existing methods do not consider the scenario of execution on resource-limited devices. Performing robust, reliable, and real-time object tracking is essential for mobile visual computing such as autonomous collision avoidance and positioning, e.g., in drones, and smart cars. Thus, it is crucial to optimize the performance of the heavyweight DL models to best conform to the underlying platform constraints. Our goal is to develop automated customization tools to efficiently deploy large-scale CNN-based trackers on various platforms. We plan to customize the state-of-the-art CNNs for the Visual Object Tracking (VOT) challenge [17] and the OTB dataset [18] to demonstrate the effectiveness of different DeepTrim customization techniques.

• **Face Detection:** Face recognition plays an impactful role in omnipresent visual applications such as video surveillance [19], computational photography [20], and human-computer interface [21], to name a few. Previous work has shown that the accuracy of face detection can be significantly improved using CNNs [22, 23, 24, 25]. Considering the data-intensive and complex nature of this task, we will employ our customization techniques to maximize the efficiency for the underlying platform. In particular, we will evaluate DeepTrim on the Wider Face [26] and UMD Faces [22] datasets.

• **Medical Image Analysis:** Popular cited applications of DL include aiding the automation of the visual diagnosis process for health care applications [27, 28, 29, 30]. The high cardinality of the processed data as well as the complexity of the diagnosis process motivate optimizing the performance of DL for such applications. Our evaluations would be based on the Messidor dataset [31] which aims to evaluate segmentation and indexing techniques in the field of retinal ophthalmology.

• **Object Classification for Autonomous Cars:** Autonomous vehicles require precise and real-time computer vision capabilities to have a clear understating of the surrounding

objects and make sound driving decisions. CNN-based approaches have led to an auspicious improvement in the detection accuracy [32, 33, 34]. The excessive runtime of complex CNNs along with the power limitations of the available hardware on vehicles hinder deployment of excessively large DL models. As such, the current DL approaches need to be tailored to the platform constraints. We plan to evaluate DeepTrim on the German Traffic Sign Detection [35] and KITTI [36] datasets as examples of vision applications in this domain.

Evaluation Metrics: To evaluate DeepTrim, we will demonstrate the improvement for each of the aforementioned applications. In particular, while fixing the inference accuracy, we will compare the baseline (in terms of the best published prior work) and our customized neural networks in terms of throughput, latency, resource utilization, and energy consumption. Our study will uncover the effect of the customization techniques applied on top of each application. Specifically, we will compare the aforementioned metrics for the various customization techniques introduced in the remainder of the proposal.

3.2 Global Flow of DeepTrim

Figure 1 presents, in a high level, the global flow of DeepTrim unified for all 4 applications. The resource profiling module translates the physical constraints into meaningful terms determining the parameters for the DL customization routines. The customization modules are realized using parameterized kernels that can be optimally configured based on the resource profiling output and performance expectations. This approach ensures that the optimization objective considers both physical constraints and application requirements, e.g. accuracy and timing.

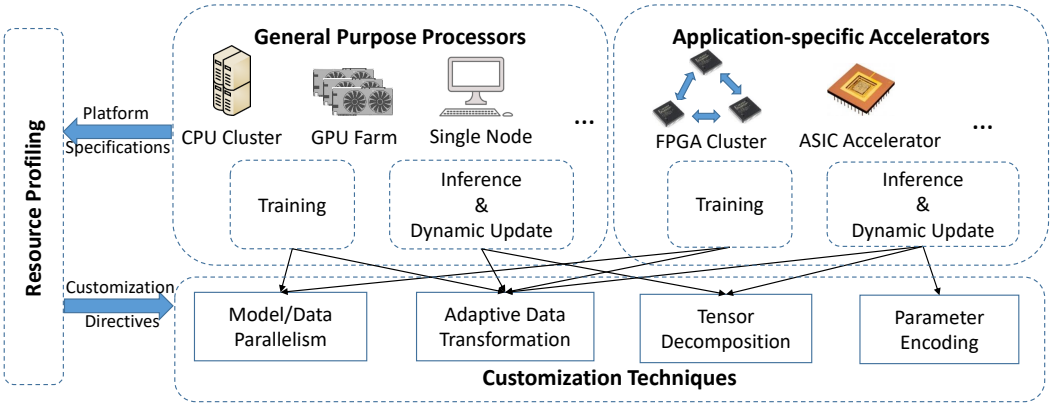


Figure 1: Overview of DeepTrim framework.

3.3 Resource Profiling

The structure of a neural network has a direct impact on the corresponding memory footprint and overall system performance. As such, the realization of DL operations can be highly diverse depending on the target platform and its memory architecture. For instance, based on the dimensionality of the matrices being multiplied, a matrix multiplication can be compute-bound, bandwidth-bound, or occupancy-bound on a specific platform. For each family of platforms, we plan to develop subroutines that run fundamental operations in the forward

and backward propagation with varying sizes to identify the underlying hardware in terms of memory, processing power, and other physical metrics. Examples of such operations include convolution, matrix multiplication, non-linearities of the memory hierarchy, and inter-core communication. We will develop multiple versions of these kernels for different target platforms (CPU, GPU, and FPGA) to hide design and optimization complexities from the application developers. DeepTrim provides the flexibility of allowing developers to add more kernels to accommodate for more complicated deep learning models.

Our prior results have demonstrated that resource-aware customization can significantly improve performance of DNN architectures [1, 37, 38, 39, 40]. In this project, we aim to further focus on developing resource-aware methodologies for complex neural networks used in computer vision tasks. Particularly, the results of the resource profiling step set upper bounds on the range of parameters of the optimization problem, facilitating automated implementation of the customization techniques.

The output of the resource profiling step includes information about the maximum size of the each layer that fits into the target platform. Besides, physical profiling indicates the throughput, energy consumption, and memory footprint of the optimized implementation on the specified platform. DeepTrim takes a set of user-defined performance metrics such as runtime, power consumption, and/or memory as its input. To optimize for the target performance metrics, DeepTrim uses the output of physical resource profiling as a guideline to perform platform-aware customization to optimally utilize available resources and meet user-defined constraints. Note that this one-time procedure will be fully automated and incur a negligible cost compared to the training or execution of deep learning models. A summary of deliverables for this thrust is as follows.

Objectives and Deliverables:

- **RP-I**: Investigating platform-specific resource constraints to characterize DL complexity on each platform. The goal is to find the source of inefficiency (e.g. high memory access overhead) in various platforms. The pertinent resources are then abstracted such that they can be incorporated into DL models (e.g., relating memory footprint to vector cardinality).

- **RP-II**: Developing subroutines that aim to identify the hardware by performing fundamental DL operations for CPU, GPU, and FPGA. Note that DeepTrim allows developers to add subroutines for other platforms of interest to ensure flexibility.

- **RP-III**: Providing an API to automatically generate resource provisioning guidelines for the customization methodologies. The API will enable data analysts to simultaneously use our platform-aware and data-aware customization and acceleration techniques without learning the hardware aspects.

3.4 Customization Techniques

We now outline our customization methods. The proposed techniques are expandable to ensure compatibility with emerging algorithms and applications.

3.4.A Adaptive Data Transformation

Motivation and problem statement: The input size of a DL model is conventionally dictated by the feature space size of the input data samples used for training the DL model. Several complex modern datasets that are structured but not inherently low-rank can be modeled by a composition of multiple lower-rank subspaces, reducing the input dimensionality of the neural networks. In this thrust, we plan to devise adaptive light-weight context and resource-aware data projection algorithms that can leverage this fact for real-world scenarios.

State-of-the-art: Previous dimensionality reduction methods such as principal component analysis (PCA) and Column Subset Selection (CSS) are either infeasible in large-scale or not applicable in streaming scenarios where the projection subspace should be dynamically updated. There are at least two limitations with such methods: (i) The prior work in data projection domain are oblivious to hardware specification, meaning that they always return a dictionary of size equal to the rank of the pertinent matrix. (ii) Traditional methods such as PCA or SVD incur a quadratic computational complexity, making them costly choices for large datasets. In addition, these techniques induce a high re-computational cost for streaming applications where the dataset dynamically grows over time.

Our approach: The goal of DeepTrim is to incorporate tools that gradually learn the projection subspace to form the low-dimensional data. The data projection approach should be linear in complexity to incur a negligible overhead. DeepTrim's data projection works by factorizing the original input data $A_{m\times n}$ into a *dictionary matrix* $D_{m\times l}$ and a *data embedding* $C_{l\times n}$ such that:

$$\underset{l, D_{m\times l}, C_{l\times n}}{minimize}(\delta_{valid}) \quad s.t. \quad \|A - DC\|_F \leq \epsilon \|A\|_F, \tag{1}$$

where δ_{valid} is the validation error acquired by training the neural network using the projected data embedding (C), $\|\cdot\|_F$ denotes the Frobenius norm, and ϵ is an intermediate approximation error that casts the rank of the input matrix A. If we denote the number of columns in matrix D and the dimensionality of each input sample by l and m respectively, the cardinality of the projected inputs will reduce from m to l. Given that $l \ll m$, the data projection scheme reduces dimensionality and facilitates the corresponding vision task.

DeepTrim approximates the solution to Eq.1 by gradually adding the learned features to the dictionary matrix D and finding the minimal l that satisfies the optimization objective. Let A be a data with a *mixture of lower-rank signal embeddings* and A_s be a random subset of A such that $|A_s| = n_0$, where $|.|$ is the cardinality or the number of columns in the data matrix A. As we experimentally have verified in [39], A_s closely models the data structure of A as $n_0 \to n$. We will build up our customization approach based on this convenient property to find the dictionary matrix D and the mixture of lower-rank data embeddings C optimized for the DL configuration imposed by the hardware. We plan to further expand the objectives of this thrust to streaming scenarios where the dataset A dynamically evolves over time. For this purpose, we will develop adaptive light-weight dictionary learning methodologies that can conform to environmental variations.

Objectives and Deliverables:

- **A-I**: Developing a platform-aware data transformation to reduce data dimensionality of neural network input data. The approach should incur a negligible overhead to be applicable to real-world streaming scenarios.

- **A-II**: Devising an automation scheme and the accompanying APIs for the data projection approaches to ensure ease of adaptation to different sets of physical constraints imposed by the platform and/or application data.

- **A-III**: Evaluating the adaptive data transformation methodology for the target computer vision applications on CPU, GPU, and FPGA platforms. The resulting performance would be compared with the state-of-the-art data projection methodologies in terms of runtime, power, and memory footprint.

3.4.B Tensor Decomposition for DL Parameters

Motivation and problem statement: Training over-parameterized neural networks facilitates the convergence of the model to a local optima. However, the trained models are significantly redundant in the way they are represented [41]. Once the model is trained and ready to deploy, this redundancy can be removed to improve inference efficiency. The computation and storage costs of CNNs are dominated by convolutional and fully-connected layers, respectively. In this thrust of the proposal, we aim to utilize tensor decomposition techniques along with our automation tools to customize CNNs. Decomposing convolutional kernels and fully-connected weight matrices result in computation and storage optimizations, respectively. Note that this track is different from the data transformation technique (Section 3.4.A) in that data transformation targets the input data but tensor decomposition aims to compress DL model parameters (weights).

State-of-the-art: Matrix decomposition and tensor decomposition can be utilized in fully-connected and convolutional layers to reduce the memory footprint and the computational complexity, respectively. For fully-connected layers, truncated singular value decomposition has been used [42] to compress the weight matrices without a significant drop of accuracy. To reduce the computational complexity of convolutional layers, tensor decomposition approaches have been utilized to approximate a given 4D tensor as a product of two 3D tensors [43]. In this context, both CP decomposition [44] and Tucker decomposition [45] have been investigated.

The focus prior work has been on developing tensor decomposition methods that achieve minimal reconstruction accuracy. However, these works are oblivious to hardware constraints and application performance requirements, motivating further investigation in this direction. In addition, exploring the possibility of dynamic model updating on the decomposed parameters is another track to pursue. We plan to explore novel tensor decomposition techniques in the literature and integrate them into DeepTrim to improve the performance of DL inference and dynamic updating.

Our approach: The most critical drawback of the existing work is the lack of automation in their proposed methodologies. These works involve hand-crafted design space explorations, making the process of model optimization tedious. A high-rank decomposition delivers an

accurate CNN, but the computational burden is also increased with higher ranks. In other words, the decomposition rank introduces a trade-off between model accuracy and execution efficiency. In order to have an automated scheme, two ingredients are necessary. First, alternative automated rank selection methodologies must be developed to facilitate the process of automated tensor decomposition without loss of accuracy. Second, an abstraction of the underlying platform shall be coupled with the decomposition scheme to ensure execution efficiency.

To address the first challenge, we will add resource awareness to the existing rank selection methods such as "global analytic variational Bayesian matrix factorization" utilized in [45]. This step will incorporate hardware limitations to the rank selection and enable DeepTrim to optimally explore the tradeoff between execution efficiency and model accuracy. The second challenge will be addressed by translating hardware limitations into abstractions that are meaningful and usable by the decomposition scheme.

Objectives and Deliverables:

- **B-I**: Investigating and adding resource awareness to the existing rank selection methodologies. We will develop automated schemes to determine the rank of decomposition based on resource constraints.

- **B-II**: Exploring hardware abstraction techniques that can facilitate the process of platform-aware customization. This step will provide support for three platforms of interest in this proposal (CPU, GPU, and FPGA).

- **B-III**: Providing an accompanying API that can perform decomposition of CNN parameters while hiding technical details from the user. The API will receive high-level user-defined variables such as the required CNN accuracy and hardware specifications and decompose the CNN parameters accordingly.

3.4.C Model/Data Parallelism

Motivation and problem statement: Even though DL neural networks have shown superior results compared to their linear machine learning counterparts, their success has been mainly based on experimental evaluations. As such, reducing the DL performance cost, particularly training runtime, enables realization of different DL models within the confine of the available computational resources to empirically identify the best one. Data and model parallelism can be used to facilitate training of DL models on parallel architectures by simultaneous training of local models [1, 39]. The goal is to transform the global DL training task into the parallel training tasks of multiple smaller size local DL networks while minimally affecting the accuracy. The cost of computing and moving data to/from the memory and inter-cores is different for each computing platform. Therefore, a parallelism approach should consider the platform-specific trade-off between memory communication and local calculations to improve the performance of costly iterative DL computations.

State-of-the-art: Traditionally, a breadth-first model partitioning approach is used to distribute the training workload of a DL network. Figure 2(a) illustrates the conventional

approach used for model parallelism in training DL models as proposed in [8, 46]. In such scenarios, the activations of neural connections that cross the machine boundaries should be exchanged in between different computing nodes to complete one round of forward/backward propagation for each data sample. What exacerbates the DL training cost in this context is the variance in processing time across different computing nodes, leading to unnecessary wait time until the slowest node finishes a given phase of computation. To mitigate the effect of this variation, authors in [8] suggest the use of asynchronous partial gradient updating in training DL networks. Although, [8] achieves meaningful runtime reduction by parallelism in training DL networks with sparse connectivity, for non-sparse (fully-connected) DL graphs updating becomes a bottleneck given the greedy layer-wise nature of back propagations and the higher communication overhead of fully-connected networks. We emphasize that even though the effectiveness of asynchronous computation and network averaging have been investigated in the literature [8, 47], DeepTrim is the first framework that leverages this flexibility for platform customization and minimizing the cost of moving data in hardware. Our approach greatly accelerates training of DL networks by platform-specific customization. It, in turn, scales up the size of DL networks that could be implemented on a constrained system-on-chip platform by breaking the global DL into sub-circuits that could be trained independently while adhering to the physical limits of the platform.

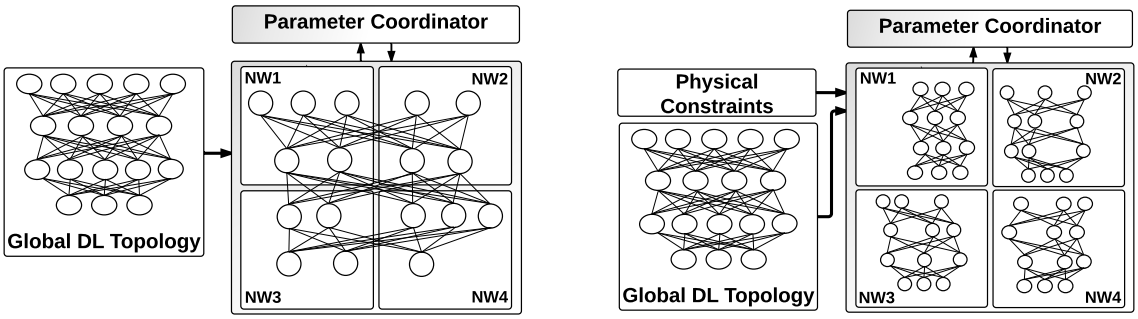


Figure 2: Network parallelism: the existing breadth-first DL partitioning approaches require communicating the partial updates multiple times for processing each batch of data. On the contrary, our depth-first approach limits such communications to only once at the end of each training iteration.

Our approach: We propose a *depth-first graph traversal* approach for model break-down to enable independent training of local replicas which minimizes the communication cost. This approach enables us to conform the DL configuration to the underlying physical constraints while lowering the communication overhead per training iteration. To optimize for the target performance metric, our approach uses the output of resource profiling as guidelines to break down the global DL model into subsequent local models that fit the pertinent resource provisioning as shown in Figure 3. As illustrated in Figure 2(b), DeepTrim transforms the global DL network into multiple overlapping local networks that are formed by sub-sampling the neurons of the initial model. Each local network has the same depth as the global model but has far fewer edges. As such, local networks can be updated independently without

having to wait for partial gradient updates from successive layers to be communicated in between different computing nodes. The updates are then periodically aggregated through the parameter coordination unit to optimize the weights of the global model. The number of neurons per local network is a configuration choice which is dictated by the pertinent physical resources (i.e., memory bandwidth, cache size, and/or available power).

To ensure uniform updating of the global network parameters, each local network is formed by random sub-sampling of a set of neurons based on a weighted probabilistic distribution. In this way, the neurons and their associated weights that have not been updated within prior local networks incur a higher probability to be selected as a part of subsequent local networks. Sub-sampling the neurons in a global DL model instead of edges has the benefit of having local networks with a fixed circuit footprint and dense (non-sparse) structures. However, sub-sampling the edges results in local networks with fixed number of connections but sparse structures. Having fully-connected local networks is particularly of interest from the hardware implementation perspective: the computations for each local network can be performed more efficiently following a specific computing pattern using dense matrix-vector multiplications as opposed to having local networks with sparse structures that incur chaotic memory access patterns. Each local network is initiated as a sub-sample of the current values of the global parameters and independently trained using different batches of the input training data. The local weights updates ΔW are periodically pushed back to be aggregated with the partial updates of other local networks working in parallel. Note that the use of stale parameters in some local networks causes no major issue in training DL models [39]. The proposed approach will enable customization of large DL models for training in multi-processor clusters as well as resource-constrained settings.

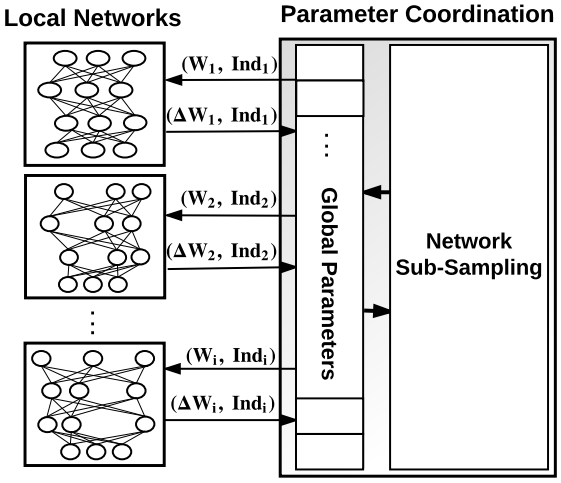


Figure 3: The updates of different local networks are asynchronously aggregated to optimize the global model's parameters.

Objectives and Deliverables:

- **C-I**: Developing novel platform-specific performance cost metrics to optimize the parallel DL neural network architecture for the underlying resources and physical constraints.

- **C-II**: Development of an end-to-end framework to perform platform-aware data/model parallelism on CPUs, GPUs, and FPGAs. Note that our proposed methodology is universal and directly applicable to other families of computing hardware.

- **C-III**: Evaluating the depth-first approach for target computer vision applications. We will demonstrate the performance improvement and scalability of our approach on

different platforms ranging from tiny embedded devices to data center scale servers by comparing throughput, latency, and power consumption with the best prior art.

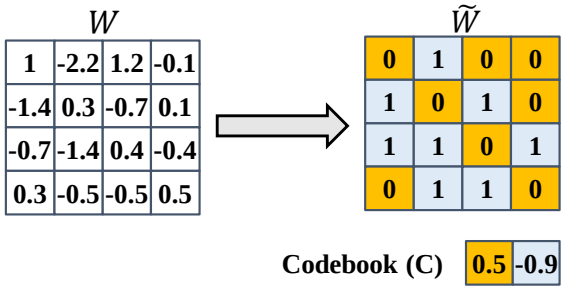
3.4.D Parameter Encoding

Motivation and problem statement: This customization technique specifically targets the execution and dynamic updating of CNNs on reconfigurable devices such as FPGA. There are two challenges to be addressed in this domain. First, the memory requirement of CNNs often exceeds the on-chip storage capacity available in the FPGA, making it inevitable to store the parameters and the intermediate activation units in an off-chip memory; consequently, the achievable throughput becomes bounded by the memory access bandwidth. Second, The computation burden shall be reduced based on the specifications of the underlying hardware (the FPGA) and the required CNN accuracy.

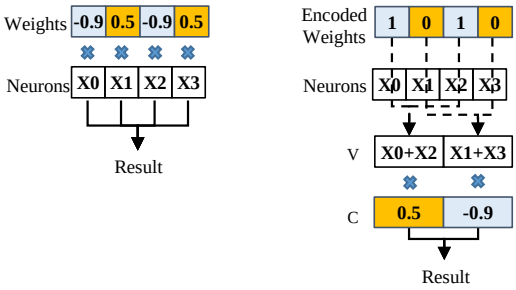
State-of-the-art: Many authors have suggested the use of fixed-point parameters to improve performance [48, 49, 50, 51, 52]. Although this approach has shown great promises in particular CNNs, the methodology is application-specific and does not provide a generic solution for the tasks requiring a higher numerical precision [48]. Using encoded floating-point parameters for memory reduction is investigated in the existing literature. Random hash functions and shared parameters have been utilized to reduce the number of free parameters during training [53, 54]. One downside of the previous works is that they involve tedious ASIC/FPGA designs to enable efficient use of the encoded parameters [55, 2]. To the best of our knowledge, the prior solutions require hand-crafted model customization for different layers of neural networks; thus, adaptive methodologies and user-friendly automated tools need to be developed to make the methods attractive to system-level designers. Another issue is that prior works have focused on encoding the parameters (weights), while most memory usage in the convolutional layers is imposed by the activation units. Unlike model parameters whose values are fixed during execution, the activation units values change during execution, hence, existing encoding methods cannot be applied to the activation units; novel methodologies that can perform online encoding need to be developed.

Our approach: The idea of parameter encoding is illustrated in Figure 4(a). For a parameter matrix (tensor) W in a certain fully-connected (convolutional) layer, the encoding algorithm seeks a codebook of K values $C = \{c_1, c_2, ..., c_K\}$ to approximate the elements of W. Each element of W can then be encoded according to its corresponding alphabet. Since the number of values in the codebook is K, the elements of the encoded weight matrix $\widetilde{W}$ are represented with $log(K)$ bits. For instance, if there are 8 alphabets in the codebook, the parameters will be encoded using $log(8) = 3$ bits which is significantly smaller than the 32 bits required to represent the floating point number.

Factorized Dot Product: The matrix multiplication and convolution operations in CNNs are narrowed down to computing dot products of vector parts in the execution phase. If the parameters are encoded, such dot products can be computed using factorized multiplications as illustrated in Figure 4(b). The multiply-add operations involving parameters of the same color can be factorized. For this purpose, an intermediate vector V can be used, where the $i-th$ element of V holds the sum of all input neurons corresponding to the $i-th$



(a) Illustration of the encoding method for a 4 × 4 matrix. Left: the original matrix containing floating-point numerals. Right: the encoded matrix of 1-bit numerals alongside the codebook of floating-point values.



(b) Conventional dot product (left) versus dot product with factorized coefficients (right)

alphabet: $V[i] = \sum_{\widetilde{W}[j]=i} X[j]$. Once V is computed, a lightweight dot product over V and the codebook values returns the correct output according to Eq. 2.

$$dot(W, X) = \sum_{i=1}^{N} W[i] \times X[i] = \sum_{k=1}^{K} V[k] \times C[k] \tag{2}$$

For a dot product involving N multiply-add operations, the factorization scheme reduces the number of multiplication from N to K. Given that $K \ll N$, both the memory footprint and the computational complexity are reduced using the encoded parameters. Nevertheless, software libraries that run encoded convolutional neural networks on parallel general purpose processors might not be as efficient as the baseline due to the overhead of decoding. Therefore, the bulk of the research in this track will include developing hardware accelerators on re-programmable devices (FPGAs).

We plan to develop two inter-connected APIs. The first API performs a pre-processing step on a given CNN. For a certain application, the encoded CNN will be examined with different codebook sizes, then the CNN accuracy associated with each codebook size will be reported to the FPGA designer. He/she then utilizes the second API that automatically customizes a certain encoded CNN topology, for each specific FPGA platform and CNN topology. We plan to utilize open source high-level synthesis hardware design languages such as Vivado HLS [56]. In particular, we will develop parameterized kernel functions that provide the basic building-blocks of encoded CNNs (e.g. factorized dot products) whose throughput and resource utilization can be controlled by the kernel parameters. Using the two APIs, designers can explore the trade-offs between model accuracy and implementation cost for different combinations of applications and platforms.

Objectives and Deliverables:

- **D-I**: Development of new algorithms that are capable of performing online encoding over the activation units. The work will allow CNNs to be fine-tuned on embedded devices to cope with environmental variations.

- **D-II**: Development of an API that enables automatic configuration of the encoding bit-width. This API will take a pre-trained neural network along with different codebook sizes to explore the CNN accuracy associated with different alphabets.

- **D-III**: Providing automated tools for *hardware* customization of the encoded neural networks realized on different FPGA platforms.

- **D-IV**: Evaluating the encoded CNNs on different reconfigurable embedded devices. We will compare the power, throughput, and energy consumption of the encoded CNN with state-of-the-art FPGA implementation of the corresponding non-encoded CNN.

3.5 Packaging and Automation

An important contribution of our work will be to facilitate practical application of DeepTrim for the target tasks, data, and platforms. As discussed previously, we aim to provide libraries of building-block kernels that are able to perform the customization techniques. In particular, the kernels should have the following characteristics:
Platform-specific: Multiple realizations of the kernels should be available for different target hardware platforms (i.e., GPU, CPU, FPGA).
Content-aware: The kernels can identify hyper-parameters (e.g. the codebook size in the encoding module) required to deliver a certain level of CNN accuracy. This feature enables the methods to be effectively utilized based on the underlying application.
Parameterized: We explore solutions that tailor the algorithm to the underlying hardware constraints. DL algorithms often enjoy a degree of freedom that introduces a tradeoff between execution overhead and CNN accuracy. The parameterized kernels will enable practitioners to exchange accuracy for a more efficient solution in terms of power, runtime, and energy.

Objectives and Deliverables:

- **Eval:** DeepTrim will provide support for training, inference, and dynamic updating of DL models on multi- and many-core CPU, CPU-GPU, and CPU-FPGA platforms. We will evaluate DeepTrim for four target computer vision applications on each hardware platform and report performance improvements in terms of the outlined metrics.

- **Pack:** The final outcome of this track is a global API that uses the local APIs developed in previous tracks (i.e. resource profiling and customization techniques). This approach enables us to partition the development of DeepTrim into short-term and long-term objectives. The global API receives high-level directives specifying the customization technique and optimization objectives such as memory footprint, power budget, run-time, and classification accuracy. It then utilizes these directives along with the reports from the resource profiling module to optimally configure the DL model in accordance with user inputs and hardware specifications. The automated nature of this process will enable systematic realization of DL models that jointly enjoy data-aware and platform-aware customization.

3.6 Contributions and broader impact

The proposed framework tackles the five challenges related to efficient training, inference, and dynamic updating of DL models mentioned in Section 1. In this section, we summarize the properties of DeepTrim that target each challenge specifically.

(I) Run-Time Improvement: To avoid performance bottlenecks, our customization techniques must balance the computation and communication costs such that the runtime is minimized for different applications executed on various devices. DeepTrim will address this challenge using parameterized kernels that insert hardware specifications to algorithm design and computation scheduling.

(II) Efficient Resource Utilization: DeepTrim will model the target platform by benchmarking fundamental operations using vendor supplied tests. The platform model along with external constraints will be used to tailor the tasks of training, inference, and dynamic updating to the underlying hardware. DeepTrim performs model modifications and efficient mapping of computations to optimally leverage the available computational resources for performance improvement.

(III) Data-aware and Platform-aware Customization: DeepTrim incorporates data characteristics as well as platform specifications in its optimization objective. As such, DeepTrim simultaneously enjoys two levels of optimization(s) that were previously performed separately, enabling efficient implementation of DL algorithms with regards to application requirements and hardware constraints.

(IV) Dynamic Updating of Pre-trained Models: The nature of the proposed customization techniques of DeepTrim must be compatible with environmental changes that require dynamic model updating. An example of such compatibility is discussed in Section 3.4 for adaptive data transformation in streaming scenarios. As a result, DeepTrim provides a scalable framework for dynamic model updating on resource-constrained devices running applications that are subject to temporal variations of their pertinent data.

(V) Automation: An accompanying API facilitates the use of the proposed framework for designers, relaxing them from manual optimizations of DL algorithms for the target applications and platforms. DeepTrim enables straightforward prototyping of hugely optimized DL models by fast exploration of the design space.

Broader Impact: PI Koushanfar is offering a new course on implementation of Deep Learning in 2018. She will integrate the applications and APIs from this research so students are exposed to real-world and contemporary challenges which would make them appreciate the importance of the techniques they learn. The course will also be offered online with accompanying open-source material so the interested engineers and researchers. We plan to provide online sides and tutorial for our APIs. The PI has recently established a new research center at UCSD, called Machine-Integrated Computing and Security (MICS) with focus on bringing hardware to make data analytics and security much more efficient. Another main objective of MICS is women empowerment. The present grant is relevant to the goals of MICS, and the personnel we are hiring for this project would form a diverse group.

The outputs of this project will have a tremendous impact. Massive datasets are being collected for scientific, industrial, government and defense purposes. Several of these areas are looking towards using variations of Deep Learning. This project plans to bring order(s) of magnitude improvement in computation which will enable analysts in all these different fields to use our APIs hardware-software library and build their own accelerated DL algorithms. The PI has a total of 10 patents and has a history of technology transfer. She will leverage her new research center (MIPS) and collaborations with industry to ensure a timely technology transfer to industry.

4 Personnel

The project duration is 1 year with an estimated budget of $320K to fund 1 post-doc, 2 grad student, 1 month PI Summer, 5 travels, and supplies/computer equipment as detailed in the budget justification. The post-doc and students will be hired once the availability of the funds is clear.

PI Farinaz Koushanfar is a professor and Henry Booker Faculty Scholar of Electrical and Computer Engineering (ECE) in University of California, San Diego (UCSD) where she is the co-Director of the new UCSD center on Machine-Integrated Computing and Security (MICS), and the director of the Adaptive Computing and Embedded Systems (ACES) Lab. Professor Koushanfar received her Ph.D. in Electrical Engineering and Computer Science and M.A. in Statistics, both from UC Berkeley in 2005. Koushanfar's research interests include design automation (DA), and in particular DA of domain-specific machine learning applications, embedded and cyber-physical systems design, and security. Dr. Koushanfar is the founder of Women ExCEl. She has received a number of prestigious awards and honors for her research, mentorship and teaching, including the Presidential Early Career Award for Scientists and Engineers (PECASE) from President Obama, the ACM SIGDA Outstanding New Faculty Award, MIT TR-35, Young Faculty/CAREER Awards from NSF, DARPA, ONR and ARO, CISCO IoT Grand Challenge Award. She is a fellow of the National Academy of Engineering (NAE) Kavli Frontiers of Engineering program.

5 Management Plan

The PI will work with the post-doc and the two students to accomplish the deliverables itemized in Section 3. The table below demonstrates the schedule for the deliverables and assignment to the personnel. On the table, the Post-doc is denoted by (P), and the students are referred by (S1) and (S2) respectively. The post-doc will work with students on each subproblem. To ensure a seamless integration, the postdoc is in charge of all APIs.

Q1	Q2	Q3	Q4
RP-1 (S1) RP-2 (S2) RP-III (P)	A-I (P, S1) A-II (P) A-III(S1) B-I (P, S2) B-II (P) B-III (S2)	C-I (P, S1) C-II (P) C-III (S1) D-I (P, S2) D-II (P) D-III (P) D-IV (S2)	Pack (P) Eval (S1, S2)

References

[1] B. D. Rouhani, A. Mirhoseini, and F. Koushanfar, "Delight: Adding energy dimension to deep neural networks," in *ISLPED*, 2016.

[2] M. Samragh, M. Imani, F. Koushanfar, and T. Rosing, "Looknn: Neural network with no multiplication," in *DATE*, 2017.

[3] Zaharia *et al.*, "Spark: Cluster computing with working sets.," *HotCloud*, 2010.

[4] A. Mirhoseini, B. D. Rouhani, E. M. Songhori, and F. Koushanfar, "Perform-ml: Performance optimized machine learning by platform and content aware customization," in *Design Automation Conference (DAC)*, June, 2016 2016.

[5] A. Mirhoseini, B. D. Rouhani, E. M. Songhori, and F. Koushanfar, "Extdict: Extensible dictionaries for data- and platform-aware large-scale learning," in *International Parallel & Distributed Processing Symposium (IPDPS)*, 2017.

[6] L. Deng, J. Li, J. T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero, "Recent advances in deep learning for speech research at microsoft," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8604–8608, 2013.

[7] L. Deng and D. Yu, *Deep Learning: Methods and Applications*. Hanover, MA, USA: Now Publishers Inc., 2014.

[8] Dean *et al.*, "Large scale distributed deep networks," in *NIPS*, 2012.

[9] A. Coates, B. Huval, T. Wang, D. Wu, B. Catanzaro, and N. Andrew, "Deep learning with cots hpc systems," in *Proceedings of the 30th international conference on machine learning*, pp. 1337–1345, 2013.

[10] T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman, "Project adam: Building an efficient and scalable deep learning training system," in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pp. 571–582, 2014.

[11] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 161–170, ACM, 2015.

[12] J. Jin, V. Gokhale, A. Dundar, B. Krishnamurthy, B. Martini, and E. Culurciello, "An efficient implementation of deep convolutional neural networks on a mobile coprocessor," in *Circuits and Systems (MWSCAS), 2014 IEEE 57th International Midwest Symposium on*, pp. 133–136, IEEE, 2014.

[13] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4293–4302, 2016.

[14] Y. Chen, X. Yang, B. Zhong, S. Pan, D. Chen, and H. Zhang, "Cnntracker: online discriminative object tracking via deep convolutional neural network," *Applied Soft Computing*, vol. 38, pp. 1088–1098, 2016.

[15] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, *et al.*, "T-cnn: Tubelets with convolutional neural networks for object detection from videos," *arXiv preprint arXiv:1604.02532*, 2016.

[16] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3074–3082, 2015.
[17] "Visual object tracking challenge." `http://www.votchallenge.net/`.
[18] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2411–2418, 2013.
[19] C. Pagano, E. Granger, R. Sabourin, G. L. Marcialis, and F. Roli, "Adaptive ensembles for face recognition in changing video surveillance environments," *Information Sciences*, vol. 286, pp. 75–101, 2014.
[20] P. N. Belhumeur, D. W. Jacobs, D. J. Kriegman, and N. Kumar, "Localizing parts of faces using a consensus of exemplars," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 12, pp. 2930–2940, 2013.
[21] A. Bulling and H. Gellersen, "Toward mobile eye-based human-computer interaction," *IEEE Pervasive Computing*, vol. 9, no. 4, pp. 8–12, 2010.
[22] R. R. C. D. C. R. C. Ankan Bansal, Anirudh Nanduri, "Umdfaces: An annotated face dataset for training deep networks," in *Arxiv Preprint arXiv:1611.01484*, November 2016.
[23] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, "A convolutional neural network cascade for face detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5325–5334, 2015.
[24] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1701–1708, 2014.
[25] S. S. Farfade, M. J. Saberian, and L.-J. Li, "Multi-view face detection using deep convolutional neural networks," in *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pp. 643–650, ACM, 2015.
[26] S. Yang, P. Luo, C.-C. Loy, and X. Tang, "Wider face: A face detection benchmark," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5525–5533, 2016.
[27] H. H. Vo and A. Verma, "New deep neural nets for fine-grained diabetic retinopathy recognition on hybrid color space," in *Multimedia (ISM), 2016 IEEE International Symposium on*, pp. 209–215, IEEE, 2016.
[28] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, "Convolutional neural networks for medical image analysis: full training or fine tuning?," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1299–1312, 2016.
[29] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1285–1298, 2016.
[30] Y. Bar, I. Diamant, L. Wolf, and H. Greenspan, "Deep learning with non-medical training used for chest pathology identification," in *SPIE Medical Imaging*, pp. 94140V–94140V, International Society for Optics and Photonics, 2015.

[31] E. Decencire, X. Zhang, G. Cazuguel, B. Lay, B. Cochener, C. Trone, P. Gain, R. Ordonez, P. Massin, A. Erginay, B. Charton, and J.-C. Klein, "Feedback on a publicly distributed database: the messidor database," *Image Analysis & Stereology*, vol. 33, pp. 231–234, Aug. 2014.

[32] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer, "Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," *arXiv preprint arXiv:1612.01051*, 2016.

[33] S. Ishibushi, A. Taniguchi, T. Takano, Y. Hagiwara, and T. Taniguchi, "Statistical localization exploiting convolutional neural network for an autonomous vehicle," in *Industrial Electronics Society, IECON 2015-41st Annual Conference of the IEEE*, pp. 001369–001375, IEEE, 2015.

[34] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[35] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark," in *International Joint Conference on Neural Networks*, no. 1288, 2013.

[36] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[37] B. D. Rouhani, A. Mirhoseini, and F. Koushanfar, "Going deeper than deep learning for massive data analytics under physical constraints," in *CODES+ISSS*, 2016.

[38] B. D. Rouhani, A. Mirhoseini, and F. Koushanfar, "Tinydl: Just-in-time deep learning solution for constrained embedded systems," in *ISCAS*, 2017.

[39] B. D. Rouhani, A. Mirhoseini, and F. Koushanfar, "Deep3: Leveraging three levels of parallelism for efficient deep learning," in *DAC*, 2017.

[40] M. Samragh, M. Ghasemzadeh, and F. Koushanfar, "Customizing neural networks for efficient fpga implementation," in *FCCM*, 2017.

[41] M. Denil, B. Shakibi, L. Dinh, N. de Freitas, *et al.*, "Predicting parameters in deep learning," in *NIPS*, 2013.

[42] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in Neural Information Processing Systems*, pp. 1269–1277, 2014.

[43] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," *arXiv preprint arXiv:1405.3866*, 2014.

[44] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," *arXiv preprint arXiv:1412.6553*, 2014.

[45] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," *arXiv preprint arXiv:1511.06530*, 2015.

[46] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[47] 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.

[48] Lin *et al.*, "Overcoming challenges in fixed point training of deep convolutional networks," *arXiv preprint arXiv:1607.02241*, 2016.

[49] Lin *et al.*, "Fixed point quantization of deep convolutional networks," *arXiv preprint arXiv:1511.06393*, 2015.

[50] Lin *et al.*, "Neural networks with few multiplications," *arXiv preprint arXiv:1510.03009*, 2015.

[51] M. Courbariaux, J.-P. David, and Y. Bengio, "Low precision storage for deep learning," *arXiv preprint arXiv:1412.7024*, 2014.

[52] Gupta *et al.*, "Deep learning with limited numerical precision.," in *ICML*, 2015.

[53] Chen *et al.*, "Compressing neural networks with the hashing trick.," in *ICML*, 2015.

[54] Han *et al.*, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," *CoRR, abs/1510.00149*, vol. 2, 2015.

[55] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: efficient inference engine on compressed deep neural network," in *Proceedings of the 43rd International Symposium on Computer Architecture*, pp. 243–254, IEEE Press, 2016.

[56] "Vivado high-level synthesis." `https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html`.