Minimum Precision Requirements of General Margin Hyperplane Classifiers

Charbel Sakr[†] *Student Member, IEEE*, Yongjune Kim[⋆], Naresh Shanbhag[†], *Fellow, IEEE*
[†]Dept. of Electrical and Computer Engineering, University of Illinois at Urbana Champaign
[⋆]Western Digital Research
Email: {sakr2, shanbhag}@illinois.edu; yongjune.kim@wdc.com

Abstract—**Margin hyperplane classifiers such as support vector machines have achieved considerable success in various classification tasks. Their simplicity makes them suitable candidates for the design of embedded intelligent systems. Precision is an effective parameter to trade-off accuracy and resource utilization. We present analytical bounds on the precision requirements of general margin hyperplane classifiers. In addition, we propose a principled precision reduction scheme based on the trade-off between input and weight precisions. We present simulation results that support our analysis and illustrate the gains of our approach in terms of reducing resource utilization. For instance, we show that a linear margin classifier with precision assignment dictated by our approach and applied to the 'two vs. four' task of the MNIST dataset is $\sim 2\times$ more accurate than a standard 8 bits low precision implementation in spite of using $\sim 2\times 10^4$ fewer 1 bit full adders and $\sim 2\times 10^3$ fewer bits for data and weight representation.**

Index Terms—**Fixed-point, precision, accuracy, resource constrained machine learning.**

I. Introduction

Machine learning algorithms have proven to be very effective in extracting complex patterns from data. However, due to their high computational and storage complexity, today these systems are deployed in the cloud and on large-scale general-purpose computing platforms such as CPU and GPU-based clusters [1]. A key challenge today is to incorporate inference capabilities into untethered (embedded) platforms such as cell phones, autonomous unmanned vehicles, and wearables. Such platforms, however, have stringent limits on available energy, computation, and storage resources. Enabling such *on-device intelligence* necessitates a fresh look at the design of resource-constrained learning algorithms and architectures.

Several techniques can be employed in order to reduce the implementation complexity of machine learning algorithms. Most of such techniques address only the storage complexity and memory requirements. For instance, parameter sparsification, popularly known as pruning [2], attempts to force a sparse, and hence memory efficient, representation of the model weights. Alternatively, parameter sharing [3] achieves the same goal via weight clustering. In contrast, finite-precision implementations [4] not only reduce memory requirements, but also have a profound impact on the computational complexity. Reducing the numerical precision while maintaining inference accuracy can be realized by leveraging the training algorithm as demonstrated in binarized neural networks [5]–[8]. While impressive results have been observed, such approaches do not provide any statistical guarantees on the degradation in accuracy of inference and incurs a significant overhead in terms of optimization effort.

An alternative approach to obtain the finite-precision requirements of machine learning algorithms is to analytically characterize the effects of quantization noise on its accuracy. Such a characterization has no optimization overhead and even provides accuracy guarantees. Previous work has demonstrated the effectiveness of such an approach to obtain the minimum precision requirements for just the inference (forward) path in deep neural networks [9], [10]. However, a corresponding analytical approach to obtain the precision of both the training (backward) path and inference remains elusive. Such an approach needs to be able to provide answers to questions such as: is there a way of choosing the minimum precision of data, weights, and internal signal representations? Are these precisions interdependent? How should one choose the precision of the training algorithm? Our work addresses these questions for the case of general margin hyperplane classifiers.

In fact, the questions listed above have been answered for the popular least mean-squared (LMS) adaptive filter [11]–[17] in the context of digital signal processing and communications systems. It turns out that there is a trade-off between data and coefficient precision in order to achieve a desired SQNR at the output. Furthermore, the precision of the LMS weight update block needs to be greater than that of the coefficient in the filter to avoid a premature termination of the convergence process. This paper leverages these insights in the design of fixed-point machine learning algorithms.

A. Contributions

We propose an analytical framework to predict precision vs. accuracy trade-offs into the design of fixed-point learning algorithms thereby eliminating the need for trial-and-error. In our preliminary work [18], we addressed the issue of fixed-point linear support vector machines (SVM). In this work, we build on these results to obtain complete and rigorous set of bounds for general margin hyperplane classifiers. Specifically, we consider classifiers using non-linear input mapping (NLIM), non-linear output mapping (NLOM), also known as kernels, and quadratic forms. We analyze the trade-off between data and weight precisions. In addition, we propose a principled approach to reduce the precision of various algorithmic parameters while maintaining fidelity to the ideal (floating-point) accuracy. We quantify the benefits of this precision reduction in terms of computational (# of 1 b full

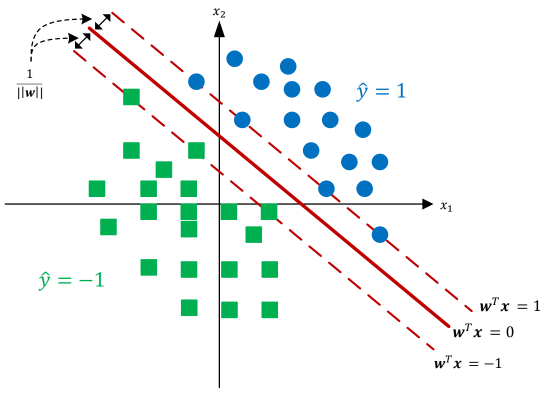


Fig. 1: Illustration of the geometry of a margin hyperplane classifier.

adders) and representational (# of bits) costs. We test and validate all our results through simulations on the Breast Cancer Dataset from the UCI Machine Learning Repository [19] and the MNIST Dataset for handwritten character recognition [20].

The rest of this paper is organized as follows: in Section II, we present necessary background. Sections III and IV contain analysis for classification and training, respectively. Simulation results are presented in Section V. We conclude in Section VI.

II. Background

A. The Classification Problem

Given a feature vector $\mathbf{x}$ of dimension D, with a corresponding unknown true label $y \in \{\pm 1\}$, it is desired to predict the class it belongs to. There are several approaches to solving the problem, amongst which are hyperplane classifiers. These separate the feature space by means of a hyperplane and assign a predicted label $\hat{y}$ based on the relative position of the feature vector with respect to the hyperplane. For generalizability, it is often desired to determine a maximum margin separating hyperplane in the feature space (see Fig. 1) as is the case for SVMs [21]. The classifier is said to have a soft margin when some of the feature vectors are allowed to lie within the margin and hence may be misclassified.

The simplest, yet effective, such classifier is a linear classifier which predicts the label as follows:

$$\mathbf{w}^T\mathbf{x} + b \underset{\hat{y}=-1}{\overset{\hat{y}=1}{\gtrless}} 0 \tag{1}$$

where $\mathbf{w}$ is the weight vector and b is the bias term. For notational convenience, we reformulate (1) into an equivalent *affine* form:

$$\mathbf{w}^T\mathbf{x} \underset{\hat{y}=-1}{\overset{\hat{y}=1}{\gtrless}} 0 \tag{2}$$

by absorbing the bias term b into the weight vector and extending the feature space by one: $\mathbf{w} \leftarrow \begin{bmatrix} b & \mathbf{w}^T \end{bmatrix}^T$ and $\mathbf{x} \leftarrow \begin{bmatrix} 1 & \mathbf{x}^T \end{bmatrix}^T$. Often, data statistics make the classification problem linearly non-separable in the input feature space. To circumvent this issue, one method is to map the input feature space to a higher dimension, i.e., $\mathbf{x} \rightarrow \phi(\mathbf{x})$ such that it becomes linearly separable, i.e.:

$$\mathbf{w}^T\phi(\mathbf{x}) \underset{\hat{y}=-1}{\overset{\hat{y}=1}{\gtrless}} 0 \tag{3}$$

We refer to this method as non-linear input mapping (NLIM). The non-linear map $\phi(\mathbf{x})$ typically lifts the data to a much higher dimension. Consequently, the dimension of the weight vector $\mathbf{w}$ in (3) is greater than that in (2).

Sometimes, it is impractical to map the input into a space where the data is linearly separable. The reason is that the corresponding dimension may be too large or even infinite. A remedy to this problem is the popular kernel trick which we refer to as non-linear output mapping (NLOM). The idea is to perform the similarity operation (projection, distance, etc.) in the original (lower dimensional) feature space and then apply a kernel to the result, as shown below:

$$\sum_{i=1}^{N_s} \alpha_i K(\mathbf{s}_i, \mathbf{x}) + b \underset{\hat{y}=-1}{\overset{\hat{y}=1}{\gtrless}} 0 \tag{4}$$

where $K(\cdot,\cdot)$ is the kernel, b is a bias term, $\mathbf{s}_i$'s are called support vectors, and α_i's are the N_s constants associated with the N_s support vectors. The support vectors found during training characterize the margin of the separating region. Their number depends on the size of the training set and the dimensionality of the input and output spaces. Note that since the support vectors are obtained through training, it is not possible to absorb the bias term into the kernel in a similar fashion as in (2).

For the dot product kernel $K(\mathbf{s}_i, \mathbf{x}) = \mathbf{s}_i^T\mathbf{x}$, it can be seen that (2) and (4) are identical by letting $\mathbf{w} = \sum_{i=1}^{N_s} \alpha_i \mathbf{s}_i$ and extending the feature space by one. A slightly more sophisticated kernel is the polynomial kernel: $K(\mathbf{s_i}, \mathbf{x}) = (\mathbf{s}_i^T\mathbf{x})^d$ where d is the order of the polynomial. Another popular kernel is the radial basis function (RBF): $K(\mathbf{s}_i, \mathbf{x}) = \exp(-\frac{1}{2}\left\|\mathbf{s}_i - \mathbf{x}\right\|^2)$. This kernel maps the data into an infinite-dimensional space.

In [22], a reformulation for the second order polynomial kernel SVM was proposed. Essentially, using the fact that $(\mathbf{s_i}^T\mathbf{x})^2 = \mathbf{x}^T(\mathbf{s}_i\mathbf{s}_i^T)\mathbf{x}$, one can reformulate (4) as:

$$\mathbf{x}^T\mathbf{K}\mathbf{x} \underset{\hat{y}=-1}{\overset{\hat{y}=1}{\gtrless}} 0 \tag{5}$$

where $\mathbf{K} = \sum_{i=1}^{N_s} \alpha_i \mathbf{s}_i\mathbf{s}_i^T$. Note that extending the feature space by one allowed us to absorb the bias term into the matrix $\mathbf{K}$. This reformulation is attractive for of two reasons: 1) the computational cost reduces from N_s inner products in $\mathcal{R}^D$ to $D+1$ inner products in $\mathcal{R}^D$ (typically $N_s \gg D$); 2) there is no need to store any support vectors.

B. Learning Classifier Parameters

It is possible to train hyperplane margin calssifiers on the fly using the stochastic gradient descent (SGD) algorithm. For a linear classifier, the instantaneous loss function is:

$$Q(y_n, \mathbf{x}_n, \mathbf{w}) = \lambda \left\|\mathbf{w}\right\|^2 + \max\{0, 1 - y_n\mathbf{w}^T\mathbf{x}_n\} \tag{6}$$

where y_n is the true label corresponding to the n^{th} sample $\mathbf{x}_n$. The hinge loss term in (6) contributes to the margin and λ is a regularizer. The optimum weight vector $\mathbf{w}$ that minimizes this loss function can be determined using SGD via the following updates [23]:

$$\mathbf{w}_{n+1} = (1-\gamma\ \lambda)\mathbf{w}_n + \begin{cases} 0 \quad \text{if} \quad y_n \mathbf{w}_n^T \mathbf{x}_n > 1, \\ \gamma\ y_n\ \mathbf{x}_n \quad \text{otherwise} \end{cases} \tag{7}$$

where γ is the learning rate. For NLIM classifiers, the weights can be trained in a similar manner as follows:

$$\mathbf{w}_{n+1} = (1-\gamma\ \lambda)\mathbf{w}_n + \begin{cases} 0 \quad \text{if} \quad y_n \mathbf{w}_n^T \phi(\mathbf{x}_n) > 1, \\ \gamma\ y_n\ \phi(\mathbf{x}_n) \quad \text{otherwise.} \end{cases} \tag{8}$$

For NLOM classifiers, we will consider only the decision directed mode. This is because NLOM requires knowledge of the support vectors that arise during training, and hence obtaining an SGD procedure to train such a system is difficult.

Finally, the quadratic form of (5) suggests a straightforward SGD training method. Indeed, as the gradient $\nabla_{\mathbf{K}}\left(\mathbf{x}^T\mathbf{K}\mathbf{x}\right) = \mathbf{x}\mathbf{x}^T$, the update equation is given by:

$$\mathbf{K}_{n+1} = (1-\gamma\ \lambda)\mathbf{K}_n + \begin{cases} 0 \quad \text{if} \quad y_n \mathbf{x}_n^T \mathbf{K}\mathbf{x}_n > 1, \\ \gamma\ y_n\ \mathbf{x}_n\mathbf{x}_n^T \quad \text{otherwise.} \end{cases} \tag{9}$$

where L_2 regularization is applied to $\mathbf{K}$.

Figure 2 depicts the architectures of the various classifiers considered with online training weight update blocks shown for linear, NLIM, and quadratic form classifiers. The multiplications and additions are element-wise. The precision assignment per dimension for each signal considered in the upcoming analysis is highlighted. These are input precision B_X, weight precision B_F, and weight update precision B_W. Next, we will obtain lower bounds on these precisions in order to achieve a desired level of accuracy.

III. Classifier Precision

In this section, we assume that the classifier has been pretrained in floating-point and we analytically study how much can it be quantized and how does its accuracy vary with its precision. In our analysis, we assume without loss of generality that all quantities of interest lie between ± 1. This can be achieved for data via scaling and for the weights by forcing saturation upon each iteration. Finally, unless otherwise stated, we allow the precision of internal signals to grow arbitrarily. That is we do not incorporate intermediate round-offs in our analysis. For instance, a length D dot product, with inputs and weights quantized to B_X and B_F bits, respectively, requires $(B_X + B_F)$-bit multipliers and $(B_X + B_F + \lceil \log_2(D) \rceil)$-bit adders. The upcoming analysis can be extended by considering round-off noise terms but would become much less tractable. Figure 1 shows the geometric configuration of a margin hyperplane classifier. This illustration reveals interesting insights upon which we build our analysis.

A. Geometric Lower Bounds

The first result exploits the *geometry* of the classifier to provide a lower bound on the precision which guarantees that the quantized feature vectors lying outside the margin are classified correctly. The geometric lower bounds (GLB) are conservative in the sense that they are sufficient conditions for fixed-point classifiers to have an average accuracy close to their trained floating-point counterparts.

Finite precision computation modifies (2) to:

$$(\mathbf{w}+\mathbf{q}_w)^T(\mathbf{x}+\mathbf{q}_x) \underset{\hat{y}=-1}{\overset{\hat{y}=1}{\gtrless}} 0 \tag{10}$$

where $\mathbf{q}_x \in \mathcal{R}^D$ and $\mathbf{q}_w \in \mathcal{R}^D$ are the quantization noise terms in $\mathbf{x}$ and $\mathbf{w}$ respectively. Each element of $\mathbf{q}_x$, except the first one, is a random variable uniformly distributed with support $[-\frac{\Delta_X}{2}, \frac{\Delta_X}{2}]$, where $\Delta_X = 2^{-(B_X-1)}$ is the input quantization step. The first term in $\mathbf{q}_x$ is zero. Similarly, each element of $\mathbf{q}_w$ is a random variable uniformly distributed with support $[-\frac{\Delta_F}{2}, \frac{\Delta_F}{2}]$, where $\Delta_F = 2^{-(B_F-1)}$ is the coefficient quantization step. This uniform assumption is standard [12], has been found to be accurate in signal processing and communications systems, and validated by the experimental results in our paper. Note that quantization perturbs both the feature vector $\mathbf{x}$ and the separating hyperplane defined by $\mathbf{w}$.

In what follows, for a feature vector $\mathbf{x}$, let us denote by $\hat{y}_{fl}(\mathbf{x})$ the label predicted by the floating-point classifier and by $\hat{y}_{fx}(\mathbf{x})$ the one predicted by the corresponding fixed-point classifier. The notation $\mathbf{a}\_$ denotes a vector $\mathbf{a}$ without the first element. We start with the linear classifier and consider the GLB on B_X.

Theorem 1 (Geometric Lower Bound on B_X for a Linear Classifier)**.**
Given B_F, and $\mathbf{w}$, $\forall \mathbf{x} \in \mathcal{R}^D$ such that $|\mathbf{w}^T\mathbf{x}| > 1$, $\hat{y}_{fx}(\mathbf{x}) = \hat{y}_{fl}(\mathbf{x})$ if

$$B_X > \log_2\left(\frac{||\mathbf{w}\_||\sqrt{D-1}}{1-2^{-B_F}||\mathbf{x}||\sqrt{D}}\right) \tag{11}$$

where $B_F > \log_2\left(||\mathbf{x}||\sqrt{D}\right)$.

Proof. See Appendix A. □

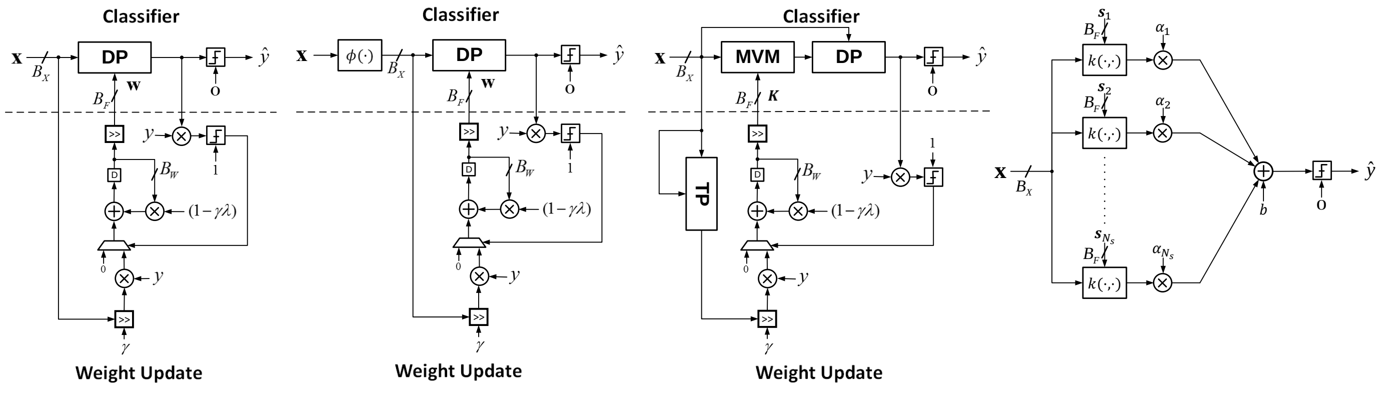
The GLB for a linear classifier reveals the following insights: 1) larger margin (i.e., smaller $||\mathbf{w}||$ in Fig. 1) allows a greater reduction of B_X, 2) there is a trade-off between B_X and B_F, and 3) input precision B_X increases with dimension D and $||\mathbf{x}||$. Figure 3 shows the trade-off between B_X and B_F for several values of the dimension D. In each case, the starting value of B_F corresponds to the condition of Theorem 1.

Note that Theorem 1 is specific to a single feature vector $\mathbf{x}$ and can be extended to a dataset.

Corollary 1.1 (Geometric Lower Bound on B_X for a Linear Classifier on a Dataset)**.**
Given B_F, and $\mathbf{w}$, $\forall \mathbf{x} \in \mathcal{R}^D$ such that $|\mathbf{w}^T\mathbf{x}| > 1$, $\hat{y}_{fx}(\mathbf{x}) = \hat{y}_{fl}(\mathbf{x})$ if

$$B_X > \log_2\left(\frac{||\mathbf{w}\_||\sqrt{D-1}}{1-2^{-B_F}\max\limits_{\mathbf{x}\in\mathcal{X}}||\mathbf{x}||\sqrt{D}}\right) \tag{12}$$

where $B_F > \log_2\left(\max\limits_{\mathbf{x}\in\mathcal{X}}||\mathbf{x}||\sqrt{D}\right)$.



(a) (b) (c) (d)

Fig. 2: Architectures of various margin hyperplane classifiers showing the precision assignments per dimension, and the classifier and weight update blocks for: (a) linear, (b) NLIM, (c) quadratic form, and (d) the NLOM classifier. DP denotes a dot product block, MVM denotes a matrix vector multiplier, and TP denotes a tensor product block.

Classifier type	Geometric Lower Bound on B_X given B_F	Geometric Lower Bound on B_F given B_X	Remarks
Linear	$B_X > \log_2\left(\frac{\|\|\mathbf{w}\_\|\|\sqrt{D-1}}{1-2^{-B_F}\|\|\mathbf{x}\|\|\sqrt{D}}\right)$ where $B_F > \log_2\left(\|\|\mathbf{x}\|\|\sqrt{D}\right)$.	$B_F > \log_2\left(\frac{\|\|\mathbf{x}\|\|\sqrt{D}}{1-2^{-B_X}\|\|\mathbf{w}\_\|\|\sqrt{D-1}}\right)$, where $B_X > \log_2\left(\|\|\mathbf{w}\_\|\|\sqrt{D-1}\right)$.	$\mathbf{x} \in \mathcal{R}^D$ $\|\mathbf{w}^T\mathbf{x}\| > 1$
NLIM	$B_X > \log_2\left(\frac{\|\|\mathbf{w}\_\|\|\sqrt{D_\phi-1}}{1-2^{-B_F}\|\|\phi(\mathbf{x})\|\|\sqrt{D_\phi}}\right)$ where $B_F > \log_2\left(\|\|\phi(\mathbf{x})\|\|\sqrt{D_\phi}\right)$.	$B_F > \log_2\left(\frac{\|\|\phi(\mathbf{x})\|\|\sqrt{D_\phi}}{1-2^{-B_X}\|\|\mathbf{w}\_\|\|\sqrt{D_\phi-1}}\right)$, where $B_X > \log_2\left(\|\|\mathbf{w}\_\|\|\sqrt{D_\phi-1}\right)$.	$\phi \in \mathcal{R}^{D_\phi}$ $\|\mathbf{w}^T\phi(\mathbf{x})\| > 1$
NLOM	$B_X > \log_2\left(\frac{\sqrt{D}\left\\|\sum_{i=1}^{N_s}\alpha_i\nabla_{\mathbf{x}}K(\mathbf{s}_i,\mathbf{x})\right\\|}{1-2^{-B_F}\sqrt{D}\sum_{i=1}^{N_s}\left\\|\alpha_i\nabla_{\mathbf{s}_i}K(\mathbf{s}_i,\mathbf{x})\right\\|}\right)$ where $B_F > \log_2\left(\sqrt{D}\sum_{i=1}^{N_s}\\|\alpha_i\nabla_{\mathbf{s}_i}K(\mathbf{s}_i,\mathbf{x})\\|\right)$.	$B_F > \log_2\left(\frac{\sqrt{D}\sum_{i=1}^{N_s}\\|\alpha_i\nabla_{\mathbf{s}_i}K(\mathbf{s}_i,\mathbf{x})\\|}{1-2^{-B_X}\sqrt{D}\left\\|\sum_{i=1}^{N_s}\alpha_i\nabla_{\mathbf{x}}K(\mathbf{s}_i,\mathbf{x})\right\\|}\right)$ where $B_X > \log_2\left(\sqrt{D}\left\\|\sum_{i=1}^{N_s}\alpha_i\nabla_{\mathbf{x}}K(\mathbf{s}_i,\mathbf{x})\right\\|\right)$.	$\mathbf{x} \in \mathcal{R}^D$ $\left\|\sum_{i=1}^{N_s}\alpha_i K(\mathbf{s}_i,\mathbf{x}) +b\right\| > 1$
Quadratic form	$B_X > \log_2\left(\frac{2\\|(\mathbf{K}\mathbf{x})\_\\|\sqrt{D-1}}{1-2^{-B_F}D\\|\mathbf{x}\\|^2}\right)$ where $B_F > \log_2\left(D\\|\mathbf{x}\\|^2\right)$.	$B_F > \log_2\left(\frac{D\\|\mathbf{x}\\|^2}{1-2^{-(B_X-1)}\\|(\mathbf{K}\mathbf{x})\_\\|\sqrt{D-1}}\right)$ where $B_X > \log_2\left(2\\|(\mathbf{K}\mathbf{x})\_\\|\sqrt{D-1}\right)$.	$\mathbf{x} \in \mathcal{R}^D$ $\|\mathbf{x}^T\mathbf{K}\mathbf{x}\| > 1$

TABLE I: Table of GLBs for linear, NLIM, NLOM, and quadratic form classifiers. The classifier parameters, for each case, are as defined in Section II. The precision assignments are as shown in Figure 2. Under these conditions we have $\hat{y}_{fx}(\mathbf{x}) = \hat{y}_{fl}(\mathbf{x})$. The first row identical to Theorem 1. The proofs of these bounds can be found in Appendix A.

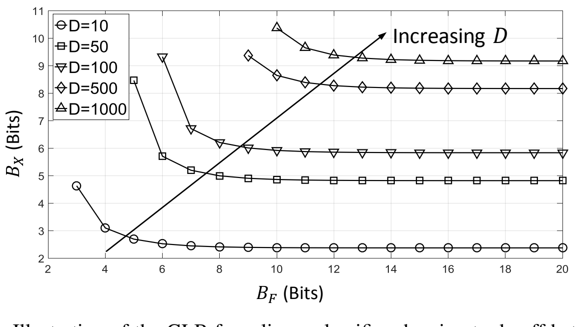


Fig. 3: Illustration of the GLB for a linear classifier showing trade-off between input (B_X) and weight (B_F) precisions, and dependence on the dimension (D). In this example, the values of $||\mathbf{w}\_||$ and $||\mathbf{x}||$ are taken to be the mean norm of the corresponding vectors assuming each entry is random and uniformly distributed between -1 and 1.

In Table I, we list the GLB for the different classifiers. These lower bounds on precision guarantee that any feature vector lying outside the margin of a floating-point classifier is identically classified by the fixed-point counterpart. The GLB is a guideline on the safe exploitation of the margin. Note that we may derive a GLB on B_X given B_F or on B_F given B_X. Each of these GLBs can be extended for a classifier operating on a dataset as in Corollary 1.1. Please refer to Appendix A for the proofs.

Figure 4 shows a comparison across linear, NLIM, and, quadratic form classifiers. It appears that the NLIM classifier has the highest input precision requirement, followed by the quadratic form and linear classifiers, in that order. The quadratic form classifier seems to have the highest weight precision requirement.

B. Probabilistic Upper Bounds

The GLB provides a lower bound on the precision requirements to ensure that fixed-point decisions are identical to

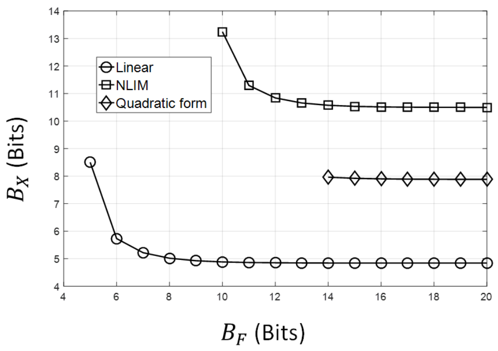


Fig. 4: Comparison of the GLB across classifier types. The input dimension is arbitrarily chosen to be $D = 50$. For the NLIM classifier, the map considered is the second order polynomial one ($D_\phi = D^2$). For each case, inputs and weights are random, uniformly distributed between -1 and 1, per dimension. To obtain average bounds, 1000 datasets are generated. The GLBs plotted are the mean bounds in each case.

floating-point for feature vectors lying outside the margin of the classifier. However, it does not provide quantitative guarantees on the fixed-point accuracy. Probabilistic upper bounds (PUBs), introduced next, upper bound the worst-case fixed-point accuracy for a given precision.

In what follows, we employ capital letters for random variables. We define probability of mismatch p_m between the decisions made by the floating-point and fixed-point algorithms as $p_m = \Pr\{\hat{Y}_{fx} \neq \hat{Y}_{fl}\}$, where $\hat{Y}_{fx}$ is the output of the fixed-point classifier and $\hat{Y}_{fl}$ is the output of the floating-point classifier. A small mismatch probability indicates that the classification accuracy of the fixed-point algorithm is very close to that of the floating-point algorithm. Indeed, if we know the accuracy of the floating-point system quantified by its probability of error $p_{e,fl} = \Pr\{\hat{Y}_{fl} \neq Y\}$ (Y is the true output) then we can obtain an upper bound on the fixed-point probability of error $p_{e,fx} = \Pr\{\hat{Y}_{fx} \neq Y\}$:

Proposition 1. *The fixed-point probability of error is upper bounded as follows:*

$$p_{e,fx} \leq p_{e,fl} + p_m \tag{13}$$

The right hand side represents the union bound of two events: 1) misclassification, and 2) incorrect classification due to quantization.

Note that $p_{e,fx}$ is the quantity of interest as it characterizes the accuracy of the fixed-point system. While $p_{e,fl}$ is a metric of the algorithm itself and does not depend on quantization, p_m is the term that captures the impact of quantization on accuracy and we hence use it as a proxy for $p_{e,fx}$ in order to evaluate the accuracy of a fixed-point system. In what follows, we obtain analytical upper bounds on p_m as a function of the precision of the fixed-point system.

It turns out that, for all the classifiers considered in Fig. 2, the mismatch probability p_m is upper bounded as follows:

Theorem 2 (Probabilistic Upper Bound on p_m)**.** *Given B_X and B_F, the upper bound on the mismatch probability of a hyperplane classifier is given by:*

$$p_m \leq \frac{1}{24}\left(\Delta_X^2 E_1 + \Delta_F^2 E_2\right) \tag{14}$$

Classifier type	E_1	E_2
Linear	$\mathbb{E}\left[\frac{\lVert\mathbf{w}\_\rVert^2}{\lvert\mathbf{w}^T\mathbf{X}\rvert^2}\right]$	$\mathbb{E}\left[\frac{\lVert\mathbf{X}\rVert^2}{\lvert\mathbf{w}^T\mathbf{X}\rvert^2}\right]$
NLIM	$\mathbb{E}\left[\frac{\lVert\mathbf{w}\_\rVert^2}{\lvert\mathbf{w}^T\phi(\mathbf{X})\rvert^2}\right]$	$\mathbb{E}\left[\frac{\lVert\phi(\mathbf{X})\rVert^2}{\lvert\mathbf{w}^T\phi(\mathbf{X})\rvert^2}\right]$
NLOM	$\mathbb{E}\left[\frac{\left\lVert\sum_{i=1}^{N_s}\alpha_i\nabla_{\mathbf{X}}K(\mathbf{s}_i,\mathbf{X})\right\rVert^2}{\left\lvert\sum_{i=1}^{N_s}\alpha_i K(\mathbf{s}_i,\mathbf{X})+b\right\rvert^2}\right]$	$\mathbb{E}\left[\frac{\sum_{i=1}^{N_s}\lVert\alpha_i\nabla_{\mathbf{s}_i}K(\mathbf{s}_i,\mathbf{X})\rVert^2}{\left\lvert\sum_{i=1}^{N_s}\alpha_i K(\mathbf{s}_i,\mathbf{X})+b\right\rvert^2}\right]$
Quadratic form	$4\mathbb{E}\left[\frac{\lVert(\mathbf{KX})\_\rVert^2}{\lvert\mathbf{X}^T\mathbf{KX}\rvert^2}\right]$	$\mathbb{E}\left[\frac{\lVert\mathbf{X}\rVert^4}{\lvert\mathbf{X}^T\mathbf{KX}\rvert^2}\right]$

TABLE II: List of E_1 and E_2 appearing in the PUB (Theorem 2) for linear, NLIM, NLOM, and quadratic form classifiers. For each case, the classifier parameters, as defined in Section II, are pre-trained in floating-point. The expectations are taken over random inputs. The precision assignments are as shown in Figure 2.

where, for each case, the values of E_1 and E_2 are listed in Table II.

Proof. See Appendix B. $\square$

In practice, the statistics (i.e., the expected values E_1 and E_2 in (14)) are calculated empirically. Note that the mismatch probability bound is increasing in Δ_X and Δ_F, which is expected as higher quantization noise variance leads to increased mismatch between fixed and floating-point algorithms.

Equation (14) reveals an interesting trade-off between input precision B_X and weight precision B_F. Indeed, in (14), the first term, $\Delta_X^2 E_1$, is the input quantization noise power gain while the second, $\Delta_F^2 E_2$, is the weight quantization noise power gain. Depending on the values taken by E_1 and E_2 it might be that one of the two terms dominates the sum. If so, then it would imply that one of B_F or B_X is unnecessarily large and hence can be reduced without increasing p_m. An intuitive first step to obtain a tighter upper bound is to make the two terms of comparable order, i.e., set $\Delta_X^2 E_1 = \Delta_F^2 E_2$ by constraining B_X and B_F as follows:

$$B_X - B_F = round\left(\log_2\sqrt{\frac{E_1}{E_2}}\right) \tag{15}$$

where $round()$ denotes the rounding operation. This is an effective way of taking care of one of the two degrees of freedom introduced by (14).

One way to employ (15) is to consider minimizing the upper bound in (14) subject to the constraint $B_X + B_F = c$ for some constant c. Indeed, it can be shown that (15) would be a necessary condition of the corresponding solution.

From the expressions of E_1 and E_2 in Table II we note two points. First, for each classifier, the denominator within the expectation operator represents the confidence in classification. This means that, the better the classifier separates the data, the smaller E_1 and E_2 are expected to be. Hence, better data separability implies better tolerance to quantization, which is to be expected. Second, the numerators represent functions of the magnitudes of weight and input vectors in the decision space.

Such magnitudes are direct functions of dimensionality and margin. Consequently, we may infer that higher dimensionality, or smaller margins, increase the values of E_1 and E_2, and hence lead to an increase in the precision requirements of the classifier. This correlates well with our observations in the case of the GLB.

The results presented in this section provide useful insights to determine suitable precision allocations for inputs and weights. First, the GLBs offer a condition under which the behavior of a fixed-point system is expected to be similar to that of the corresponding floating-point one. However, they do not provide analytical guarantees on the resulting accuracy. The PUBs do, and (15) captures an interesting trade-off between both precisions. In practice, it is possible to use both sets of bounds to efficiently determine minimal precisions for a fixed-point implementation. For example, (15) can first be used to determine the optimal difference between B_X and B_F, then the GLB can be used to find a suitable pair of (B_X, B_F). Finally, the PUB can be used to estimate the expected accuracy loss. We shall demonstrate this approach in Section V.

IV. Precision in Training

In the preceding discussion, all learning parameters were assumed to have been obtained after full-precision training. There are many applications where reduced-precision training has several merits. These include learning on edge devices in order to enable continuous tracking, have better privacy guarantees, and reduce communication overhead [24]. For such applications, it is reasonable to assume a full precision baseline with desirable convergence properties has already been designed and its information is available. Thus, setting feedforward precisions may be done via our proposed analysis above. In addition, weight update precision needs to be determined so that edge training exhibits statistically similar convergence behavior. In this section, we consider the problem of finding precision requirements on B_W in the updates when training is done in fixed-point with feedforward precisions set to B_X and B_F.

Consider a linear classifier. Note that the right hand side of (7) includes an attenuation term $(1-\gamma\ \lambda)\mathbf{w}_n$ and an update term equal to 0 or $\gamma\ y_n\ \mathbf{x}_n = \pm\gamma\mathbf{x}_n$ where $x_i \in [-1,1]$ for $i = 1\ldots D$. Without loss of generality, we assume that floating point convergence is achieved for $\lambda = 1$ and some small value of γ. Therefore, the attenuation factor $(1-\gamma\ \lambda)$ is less than or close to unity independent of B_W.

Our next result ensures that the non-zero update term in (7) is non-zero during training in spite of weight update quantization.

Theorem 3 (Weight Update Requirements for a Linear Classifier)**.**
The following lower bound on weight update precision B_W is a sufficient condition to ensure full convergence:

$$B_W \geq B_X - \log_2(\gamma) \tag{16}$$

when B_X and B_F are the input and coefficient precisions, respectively.

Proof. Each step in (7) has magnitude $|\gamma x|$ where x is a scalar value taken by the different components of $\mathbf{x}$. For any non-zero update to remain non-zero after quantization, we need $|\gamma x| > \frac{1}{2}b_{min}$ where $b_{min} = 2^{-(B_W-1)}$ is the value of the least significant bit (LSB) in the weight update block. So we require $|\gamma||x| > \frac{1}{2}2^{-(B_W-1)} = 2^{-B_W}$. This has to be satisfied for any non-zero value of x, but the minimum non-zero value taken by $|x|$ is $2^{-(B_X-1)}$. So we get $|\gamma|\cdot 2^{-(B_X-1)} > 2^{-B_W}$, which can be written as:

$$B_W > B_X - 1 - \log_2(\gamma) \Leftrightarrow B_W \geq B_X - \log_2(\gamma).$$

□

Theorem 3 provides a sufficient condition for convergence. As the SGD approximates the true gradient at every step, fewer bits than specified in (16) may work occasionally. We refer to these as boundary cases and present a detailed analysis on the corresponding learning behavior in Appendix C.

For a NLIM classifier, the exact same discussion applies. This is because (7) and (8) are structurally equivalent. For a quadratic form classifier the results changes a little bit. Indeed, every entry (i,j) in the matrix $\mathbf{K}$ has an update term equal to $|\gamma x_i x_j|$ in absolute value. The product of two scalars increases the precision by a factor of 2. Following the same argument as in Theorem 3, we have the following result:

Corollary 3.1 (Weight Update Requirements for a Quadratic Form Classifier)**.**
The following lower bound on weight update precision B_W is a sufficient condition to ensure full convergence:

$$B_W \geq 2B_X - \log_2(\gamma) \tag{17}$$

when B_X and B_F are the input and coefficient precisions, respectively.

While the required precision for full convergence has been slightly modified in this case, the discussion of the learning behavior for lower precisions is exactly similar to the one following Proposition 3. This is because that discussion observes the inputs starting from the most significant bit (MSB) and towards the LSB. The set $[-1,1]$ being closed under multiplication, the update terms are just as significant as those in the case of a linear classifier when looking at the MSB and onwards.

V. Simulation Results

In this section, we validate the analysis of Sections III and IV via simulation with the UCI Breast Cancer dataset [19] and the 'two vs. four' task applied to the MNIST dataset for handwritten character recognition [20].

A. Complexity in Fixed-Point

We identify two implementation costs associated with the implementation of signal processing and machine learning systems: the computational cost and the representational cost.

The computational cost is measured in numbers of 1 bit full adders ($\#FA$) to come up with one decision. Dot products are the predominant computing structures in machine learning systems. We assume they are realized in a multiply and accumulate (MAC) fashion where additions and multiplications are implemented using ripple carry adders and Baugh-Wooley multipliers, respectively. Consequently, the number of full adders used to compute the dot product between two D

Classifier type	$B_X = B_F$			$B_F - B_X$ dictated by (15)		
	FX Sim	GLB	PUB$_e$	FX Sim	GLB	PUB$_e$
Linear	(4,4)	(4,4)	(6,6)	(2,4)	(2,4)	(4,6)
NLIM	(4,4)	(6,6)	(8,8)	(4,7)	(4,7)	(6,9)
NLOM	(2,2)	(6,6)	(3,3)	(2,2)	(6,6)	(3,3)
Quadratic form	(5,5)	(6,6)	(5,5)	(2,5)	(4,7)	(3,6)

TABLE III: Summary of Fig. 5 illustrating minimum precision requirements for hyperplane classifiers on the Breast Cancer dataset dictated by FX Sim, GLB, and PUB$_e$ when $B_X = B_F$ and $B_F - B_X$ determined by (15).

dimensional vectors with entries quantized to B_X bits and B_F bits respectively is,

$$DB_XB_F + (D-1)(B_X + B_F + \lceil\log_2(D)\rceil - 1) \quad (18)$$

and the output is $(B_X + B_F + \lceil\log_2(D)\rceil)$ bits.

Equation (18) describes the computational cost of the linear classifier and the NLIM one (but with D replaced by D_ϕ). The quadratic form classifier will have a total computational cost equal to:

$$\begin{aligned} &D^2B_XB_F + D(D-1)(B_X + B_F + \lceil\log_2(D)\rceil - 1)\\ &+ DB_X(B_X + B_F + \lceil\log_2(D)\rceil)\\ &+ (D-1)(2B_X + B_F + 2\lceil\log_2(D)\rceil - 1). \end{aligned} \quad (19)$$

For NLOM, we consider the computational cost dependent on precision. For instance, the evaluation of the norm difference when using a RBF kernel, which has a total computational cost of:

$$N_s(DB_{X,F} + DB_{X,F}^2 + (D-1)(2B_{X,F} + \lceil\log_2(D)\rceil - 1)) \quad (20)$$

where $B_{X,F} = \max(B_X, B_F)$.

The representational cost is defined as the total number of bits needed to represent all parameters (inputs and weights). It is a measure of the complexity of storage and communications. Depending on the application, either one of the representational costs associated with weights or inputs may be more important.

B. Validation of Bounds

We consider two scenarios for each classifier type:

- Scenario A: $B_X = B_F$.
- Scenario B: $B_X - B_F$ defined by (15).

For each of the two scenarios:

1) We sweep the value of B_X (and the corresponding value of B_F) and perform fixed-point simulation (FX Sim) to obtain the associated true fixed-point classification error rate (p_e).
2) For each pair (B_X, B_F), we find the corresponding probabilistic upper bound on the classification error rate (PUB$_e$) based on Proposition 1 and the PUB (Theorem 2).
3) We find the smallest value of B_X (and hence B_F) that satisfy the GLB in Table I.

Figure 5 shows our results for linear, NLIM (second order polynomial map), quadratic form, and NLOM (RBF kernel) classifiers for the Breast Cancer dataset. The training and testing sets were obtained by independently sampling 500 random samples for each from the dataset. The classifiers were pretrained in floating-point using SGD with $\gamma = 2^{-10}$ and $\lambda = 1$ except for the NLOM classifier which was trained using the commercial LIBSVM package [25].

For the linear classifier, Fig. 5(a) shows the validity of the GLB and the PUB$_e$ for equal input and weight precisions. Indeed, fixed-point simulations for precisions larger than the GLB (4 bits) offer no significant accuracy gains while lower precisions seem to quickly degrade the accuracy. The PUB$_e$ also successfully upper bounds the error obtained in fixed-point. Figure 5(b) shows the benefits of using (15) which dictates in this example that $B_F = B_X + 2$. Indeed, not only are the GLB and PUB$_e$ still valid, but it is also possible to decrease B_X to 2 bits, as reflected by the GLB and supported by the fixed-point simulation, while maintaining good accuracy.

Similar trends are observed for NLIM (Fig. 5(c,d)) and quadratic form (Fig. 5(e,f)) classifiers. Indeed, the GLB reaches the precision value after which the fixed-point accuracy saturates to within 2 bits, and the PUB$_e$ successfully upper bounds the fixed-point probability of error. Interestingly, the PUB$_e$ is much tighter for the quadratic form classifier as compared to the NLIM classifier.

Figure 5(g) shows our results for the NLOM classifier. There are a total of 98 support vectors. In this case (15) results in $B_X = B_F$, this is why we have only one plot. Observe that the PUB$_e$ is much tighter than the GLB. Indeed, the PUB$_e$ tracks the fixed-point simulations to precisions as low as 3 bits. The GLB predicts 6 bits.

A detailed breakdown of the minimum precision requirements determined by FX Sim, GLB, and PUB$_e$ for all classifier types is presented in Table III.

So far, all results were performed on the Breast Cancer dataset. To show the generality of our results, we also conduct a similar experiment on the 'two vs. four' task on the MNIST dataset where we consider a linear classifier. Again, the classifier is first pretrained using SGD as discussed in Section II.

The training and test sets were obtain by selecting the 'two' and 'four' instances from the original MNIST dataset. Overall, we had 11800 training and 2014 testing samples, respectively. The classifier was trained using SGD with $\gamma = 2^{-10}$ and $\lambda = 1$. The results are shown in Fig. 6. Once again, we find the numerical results to be consistent with the analysis of Section III.

Interestingly, in the experiments on the Breast Cancer Dataset, (15) seemed to always yield a B_F larger than B_X by a few bits (2 or 3) except for the case of NLOM. For the MNIST experiment, this trend seems to continue and is even more pronounced as the difference $B_F - B_X = 6$ bits. In fact, this should not be surprising. Indeed, the feature vectors in the MNIST dataset are grayscale images whereas the weights define the separating hyperplane. It hence reasonable to expect the precision requirements of weights to be higher as slight changes to the separating hyperplane are more detrimental to the classification accuracy.

C. Complexity vs. Accuracy Trade-offs

We compare costs and performance for the following setups:

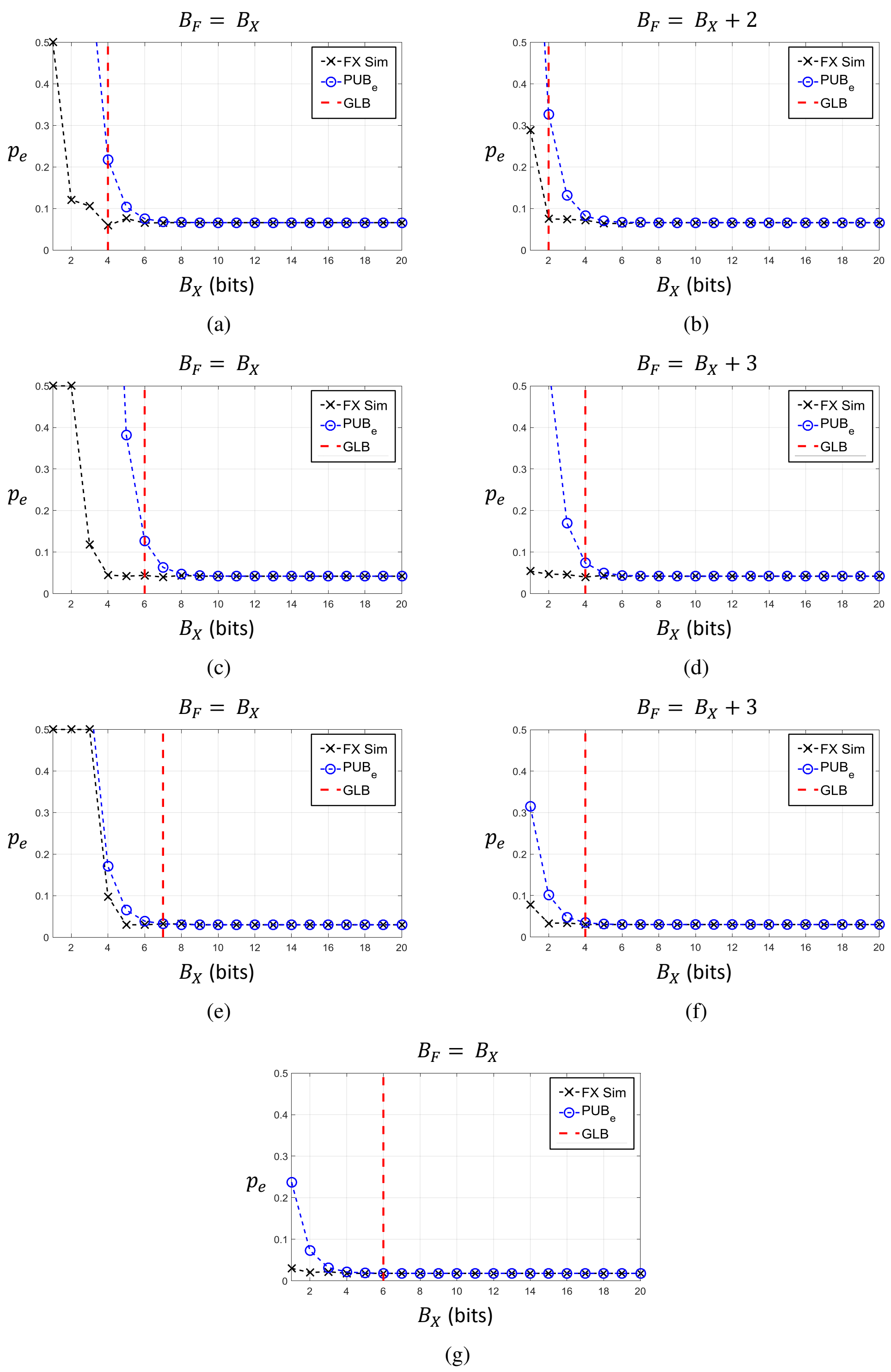
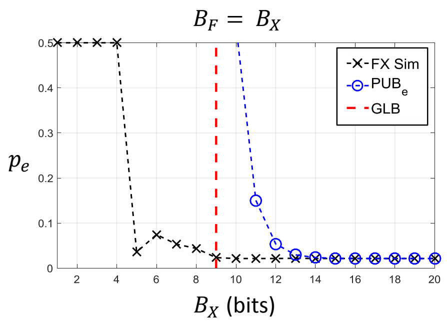
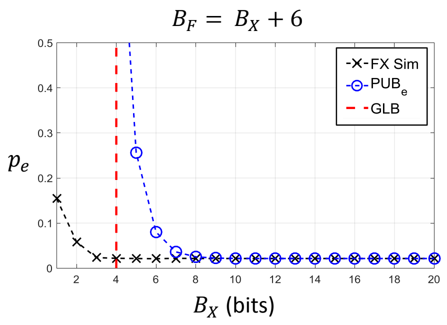


Fig. 5: Results for classification on the Breast Cancer Dataset: classification error rate p_e in fixed-point simulations (FX Sim), analytical probabilistic upper bound (PUB$_e$), geometric lower bound (GLB) for $B_X = B_F$ and $B_X - B_F$ determined by (15) for linear (a,b), NLIM (second order polynomial) (c,d), quadratic form (e,f), and NLOM (RBF) (g) classifiers. For the NLOM classifier, (15) dictates $B_X = B_F$ hence Scenarios A and B collapse into one. The PUB$_e$ is found to successfully upper bound the fixed-point error obtained through FX Sim. The GLB identifies a precision for which FX Sim settles. The use of (15) makes it possible to reduce precision but maintain accuracy.

1) minimum value of B_X specified by the GLB with $B_X = B_F$,
2) minimum value of B_X specified by the GLB with the difference between B_X and B_F satisfying (15),
3) 8 bits quantization for all parameters,
4) an arbitrary precision assignment not satisfying the bounds presented in Section III.



(a)



(b)

Fig. 6: Results for linear classification on the 'two vs. four' task for the MNIST Dataset: classification error rate p_e in fixed-point simulations (FX Sim), analytical probabilistic upper bound (PUB$_e$), geometric lower bound (GLB) for (a) $B_X = B_F$ and (b) $B_X - B_F = 6$ as dictated by (15). The PUB$_e$ is found to successfully upper bound the fixed-point error obtained through FX Sim. The GLB identifies a precision for which FX Sim settles. The use of (15) makes it possible to reduce precision but maintain accuracy.

Linear Classifier

(B_X, B_F)	Computational cost ($\#FA$)	Representational cost (bits)	Test error
$(8, 8)$	894	168	6.6%
$(4, 4)$	286	84	5.9%
$(2, 4)$	178	64	7.5%
$(2, 3)$	146	53	13.5%

NLIM (second order polynomial) Classifier

(B_X, B_F)	Computational cost ($\#FA$)	Representational cost (bits)	Test error
$(8, 8)$	5.9×10^3	1048	4.4%
$(6, 6)$	3.7×10^3	786	4.4%
$(4, 7)$	3.1×10^3	722	4.0%
$(3, 3)$	1.4×10^3	393	11.8%

Quadratic form Classifier

(B_X, B_F)	Computational cost ($\#FA$)	Representational cost (bits)	Test error
$(8, 8)$	11.9×10^3	1048	3.2%
$(7, 7)$	9.5×10^3	917	3.2%
$(4, 7)$	5.6×10^3	887	3.0%
$(4, 4)$	3.8×10^3	524	9.8%

NLOM (RBF) Classifier

(B_X, B_F)	Computational cost ($\#FA$)	Representational cost (bits)	Test error
$(8, 8)$	10×10^4	7920	1.8%
$(6, 6)$	6.6×10^4	5940	1.8%
$(1, 1)$	1.1×10^4	990	3.0%

TABLE IV: Results for the Breast Cancer Dataset. Comparison of computational cost, representational cost, and test error for linear, NLIM (second order polynomial), quadratic form, and NLOM (RBF) classifiers. The precision assignments considered are the standard low precision quantization (8,8), the minimum (B_X, B_F) satisfying the GLB when $B_X = B_F$, the minimum (B_X, B_F) satisfying the GLB when $B_F - B_X$ is dictated by (15), and one arbitrarily chosen assignment violating the bounds in Section III. In the case of the NLOM classifier, the second and third precision assignments are identical. In each case, the use of the GLB and (15) makes it possible to reduce complexity but maintain accuracy.

The first setup takes into account only one half of the theory proposed in Section III, while the second shows the possibility and benefits of leveraging both GLB and PUB for more aggressive, yet principled quantization. We chose 8 bits in the third setup as it is a typical representative of low precision/high performance arithmetic [26]. The fourth setup is intended to highlight the drawbacks of aggressive and unprincipled quantization.

(B_X, B_F)	Computational cost ($\#FA$)	Representational cost (bits)	Test error
$(9, 9)$	85×10^3	14×10^3	2.3%
$(8, 8)$	70×10^3	13×10^3	4.4%
$(4, 10)$	49×10^3	11×10^3	2.2%
$(3, 6)$	28×10^3	7×10^3	8.5%

TABLE V: Results for linear classification on the 'two vs. four' task for the MNIST Dataset: Comparison of computational cost, representational cost, and test error. The precision assignments considered are the minimum (B_X, B_F) satisfying the GLB when $B_X = B_F$, the standard low precision quantization (8,8), the minimum (B_X, B_F) satisfying the GLB when $B_F - B_X = 6$ as dictated by (15), one arbitrarily chosen assignment of (3,6) violating the bounds in Section III. The use of the GLB and (15) makes it possible to reduce complexity but maintain accuracy.

For the Breast Cancer dataset, Table IV shows how our principled quantization strategy makes it possible to operate at low costs while maintaining accuracy. Indeed, for the linear classifier, an unstructured precision assignment of 2 and 3 bits for inputs and weights, respectively, does reduce the computational and representational costs, but results in a relatively high test error. At the expense of only $\sim 1.2\times$ (178/146) computational cost and $\sim 1.2\times$ representational cost, $\sim 1.8\times$ classification error rate reduction is possible using a $(2, 4)$ quantization as dictated by (15) and the GB. This corresponds to $\sim 5\times$ reduction in computational cost and $\sim 2.6\times$ reduction in representational cost compared to the traditional 8 bits quantization. Similar trends are observed for the other classifiers.

Similarly, this comparison for the MNIST experiment is shown in Table V. Interestingly, we observe here is that the constraint $B_X = B_F$ is too harsh. Indeed, although the value $B_X = B_F = 9$ bits determined by the GLB yields a resulting accuracy $\sim 2\times$ better than the precision assignment of $B_X = B_F = 8$ bits, this should not be considered a satisfactory result. In fact, when taking into account the trade-off between B_X and B_F as described in (15), $B_F = B_X + 6$ here, we are able to reduce B_X down to only 4 bits. This corresponds to $\sim 1.7\times$ and $\sim 1.3\times$ decrease in the computational and representational costs with a negligible degradation in accuracy. Note that this quantization strategy results in a classifier $\sim 2\times$ more accurate than the one quantized to 8 bits in spite of using $\sim 2 \times 10^4$ fewer FAs and $\sim 2 \times 10^3$ fewer bits for data and weight representation. This highlights the importance of the analysis of the trade-off between input and weight precisions that was described in Section III.

D. Training Behavior

Here we illustrate the the impact of precision on training as a supporting evidence to the discussion in Section IV. We start with the Breast Cancer dataset. The feedforward precisions are set as follows: we choose $B_X = 6$ in order to separate the Theorem 3 scenario from the three boundary cases in Appendix C, and we choose B_F as specified by the GLB. That way, the feedforward precisions are minimal in a geometric sense and the full convergence of the algorithm is determined by the weight update precision (Theorem 3).

We shall consider a linear and a quadratic form classifier. We consider two scenarios: training with a small step size ($\gamma = 2^{-10}$) and training with a large step size ($\gamma = 2^{-5}$). The dataset is once again split into 500 training and 500 testing samples. Each time we show the convergence behavior of the training loss function, which is the objective being minimized, and the test error rate, which is the target metric to be minimized. Each experiment is conducted over 30 independent runs and the curves shown hereafter are ensemble averages over these runs. For each run, the initial weights are set to zeros and λ is set to 1.

Figure 7 shows convergence curves for a linear classifier trained on the Breast Cancer Dataset. We show convergence curves for weight update precisions of: a) $B_X - \log_2(\gamma)$ (Theorem 3), $1 - \log_2(\gamma)$ (Boundary Case 1), b) $2 - \log_2(\gamma)$ (Boundary Case 2), and $-\log_2(\gamma)$ (Boundary Case 3). As shown, in both scenarios, a weight update precision of $B_X - \log_2(\gamma)$ is enough to mimic floating-point behavior in fixed-point. For weight update precisions of $1 - \log_2(\gamma)$ and $-\log_2(\gamma)$, the occasional sign-SGD updates are not enough for convergence as discussed in Section IV and Appendix C. For a weight update precision of $2 - \log_2(\gamma)$, we do observe a decrease in the training loss function and the test error rate but the accuracies obtained are not as good as those of the floating-point trials. This demonstrates the sufficiency property of Theorem 3.

As far as the training loss function is concerned, similar trends are observed for the quadratic form (Fig. 8) classifier. Interestingly, the test error rate does go down in spite of imprecise updates. Nonetheless, we see that higher precision leads to greater overall accuracy. Note that we not only considered weight update precision of $B_X - \log_2(\gamma)$, but also $2B_X - \log_2(\gamma)$ as dictated by Corollary 3.1. Results show that this increased precision only marginally improves the convergence behavior.

Figure 9 illustrate these results for the ‘two vs. four’ classification task on the MNIST Dataset for linear classification with a learning rate of 2^{-10}. The feedforward precisions are again chosen to be minimal in the geometric sense. Those precisions were determined in the previous subsection to be 4 and 10 for the inputs and weights, respectively. The results here are very consistent with those observed for the experiments on the Breast Cancer Dataset. Indeed, it is clearly seen that a weight update precision of $B_W = B_X - \log_2(\gamma)$ as dictated by Theorem 3 is of paramount importance for successful convergence.

E. Discussion on Accuracy of Analysis

We conclude this section by discussing the accuracy of our analysis in predicting the precision requirements. The key sources of GLB's inaccuracy arises from the use of triangle and Cauchy-Schwarz inequalities, which can be loose, and assuming worst case quantization noise magnitudes. On the other hand, PUB$_e$ is obtained by application of the Chebyshev inequality, a relatively more conservative estimate. Nevertheless, both bounds consistently predict precision requirements within $0 \sim 2$ bits of the empirically obtained minimum thereby providing a good first estimate.

In addition, our results reveal that the accuracy of the bounds is a function of the classifier type. The GLB is tightest for linear classifiers because of their lower dimensionality. As dimensionality increases, the worst case quantization noise magnitude approximation gets repeatedly applied making the GLB looser for non-linear classifiers. In contrast, the PUB$_e$ is found tightest for non-linear classifiers. The use of the Chebyshev inequality loosens the bound for larger values of pm. Since non-linear classifiers have a stronger representational power, it is expected that they would exhibit smaller pm and consequently have a tighter PUB$_e$.

Finally, for training, Theorem 3 ensures avoidance of premature stoppage of convergence. The reason why there is a slight offset between FL and FX convergence is because feedforward precisions are selected so as to guarantee a small but non-zero mismatch probability. On the other hand, there is a slight slow down (with respect to FL training) in the case of the linear classifier applied to the Breast Cancer dataset. This is due weight clipping caused by the limited dynamic range of FX representation.

VI. Conclusion

We have presented a theoretical analysis of the behavior of general fixed-point margin hyperplane classifiers. The results presented consist of bounds based on the geometry and statistics of the classification task and on the convergence conditions of the training task. By characterizing the trade-off between input and weight precisions, efficient precision reduction scheme was presented. This framework eliminates the need for expensive trial-and-error. Furthermore, it presents guidelines for minimizing resource utilization. This utilization was captured by the computational and representational costs. Simulation results shown support the developed theory and highlight its benefits.

Several insights can be taken from our work. These include a trade-off between input and weight precision which is useful for minimizing the overall precision. Furthermore, it was observed from the GLB that precision increases logarithmically with the dimensionality of the problem. From the PUB, it was seen that the mismatch probability between fixed-point and floating-point classifier decays, at worst, exponentially with precision. It is in fact possible to derive a tighter bound on this mismatch probability using the Chernoff bound which marginally improves this dependence making it double

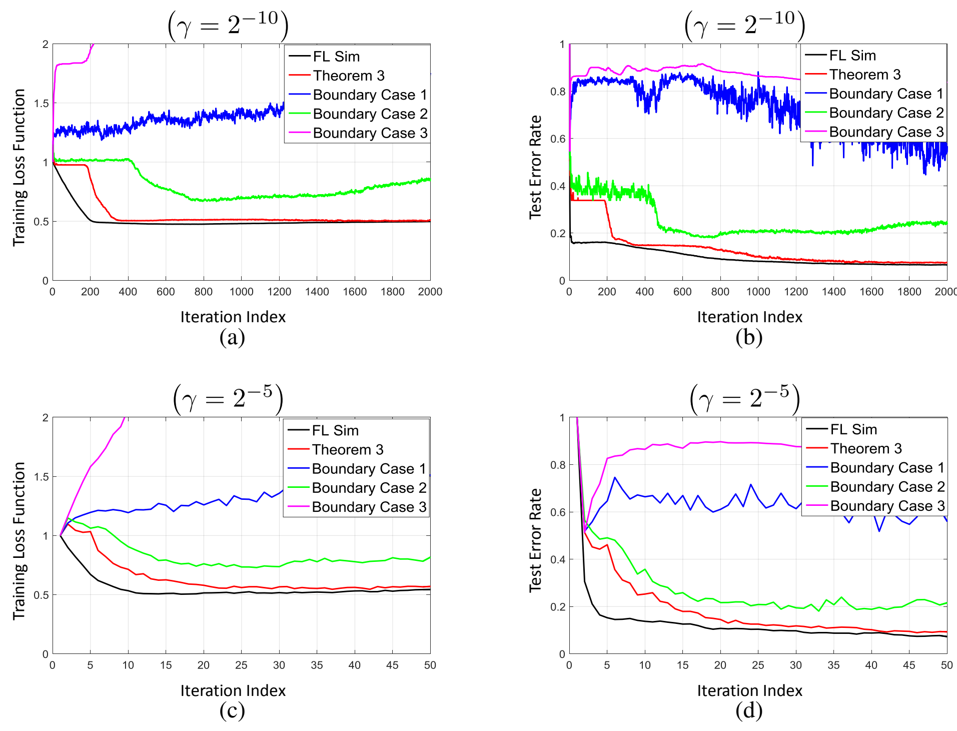


Fig. 7: Results for linear classifier training on the Breast Cancer Dataset: (a) ensemble training loss function (6) and (b) ensemble test error rate for a small learning rate; (c) ensemble training loss function and (d) ensemble test error rate for a large learning rate. FL Sim denotes the floating-point simulation. In each case, a weight update precision of $B_X - \log_2(\gamma)$ (Theorem 3) is enough to mimic floating-point behavior in fixed-point. Accuracy degradation is observed for lower precisions. Note that for clarity, we only include comparisons with boundary cases 1, 2, and 3.

exponential in precision. Finally, it was shown that the early stopping criterion applies to fixed-point training and provides a sufficient condition on the weight update precision for full convergence.

Future work includes a deeper dive into the topic of complexity vs. accuracy in machine learning. The presented work takes a conservative approach in the model of quantization. It is possible to shape quantization noise statistics using dithering during training. For instance, stochastic quantization or random noise injection can be used. Such approach might lead to greater reductions in precision. Another line of work is to study the effects of floating-point quantization. While fixed-point implementations are usually more efficient than floating-point ones, the latter benefit from a much wider dynamic range which could be beneficial to the robustness in classification. An orthogonal direction is to consider the structure of the algorithms themselves. It is well established that data-driven models inhibit large redundancies which can be exploited to trade-off complexity with robustness. The above constitute the first steps in the important task of developing a unified and principled framework to understand complexity vs. accuracy in the design and implementation of machine learning systems.

Appendix A
Proofs of Geometric Lower Bounds

The proof of the GLB is done in three steps:

- Step 1: Determine the output quantization noise N_o at the output of the classifier. In this step, we neglect cross products of quantization noise terms as their contribution is very small. For the NLOM classifier, this is equivalent to a first order Taylor expansion on the kernel. For the linear classifier we have:

$$N_o = \mathbf{q}_w^T\mathbf{x} + \mathbf{w}^T\mathbf{q}_x.$$

- Step 2: Upper bound the magnitude of N_o using the triangle and Cauchy-Schwarz inequalities. Input quantization noise terms are upper bounded by 2^{-B_X} and weight quantization noise terms by 2^{-B_F}. For the linear classifier we have:

$$\begin{aligned}|N_o| &\leq |\mathbf{q}_w^T\mathbf{x}| + |\mathbf{w}^T\mathbf{q}_x| \\ &\leq \|\mathbf{q}_w\|\,\|\mathbf{x}\| + \|\mathbf{w}\|\,\|\mathbf{q}_x\| \\ &\leq 2^{-B_F}||\mathbf{x}||\sqrt{D} + 2^{-B_X}||\mathbf{w}\_||\sqrt{D-1}\end{aligned}$$

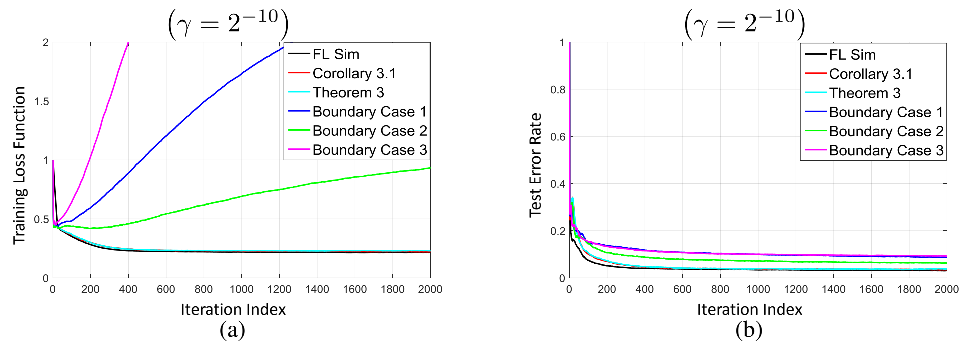
where the introduction of the dimension D is due to the expansion of the norms of $\mathbf{x}$ and $\mathbf{w}$.
- Step 3: Set the upper bound on $|N_o|$ to be less than the functional margin of the classifier which is equal to 1 (See Fig. 1).

Step 3, up to a rearrangement of terms is equivalent to the GLB as described in Section III.A. In Table VI, we list the output quantization noise and corresponding upper bound for each classifier type.

Appendix B
Proofs of Probabilistic Upper Bounds

The PUB is proved in 4 steps:

- Step 1: For a single input, obtain the total output quantization noise N_o. This is identical to the Step 1 in the proof of the GLB in Appendix A. This output



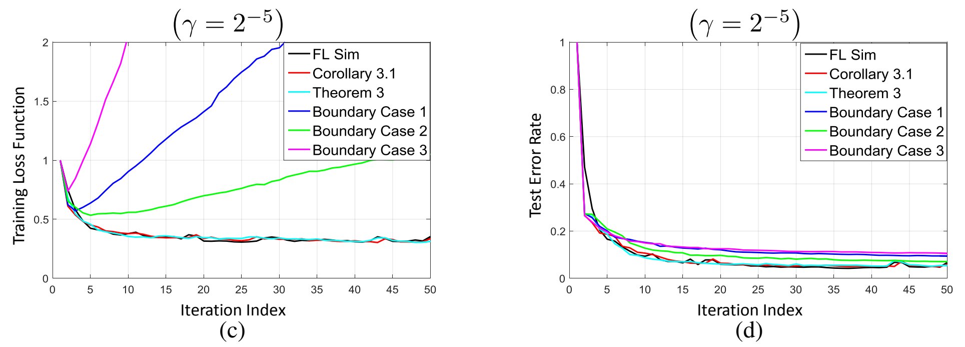


Fig. 8: Results for quadratic form classifier training on the Breast Cancer Dataset: (a) ensemble training loss function ((6), matrix (Frobenius) norm replaces vector norm) and (b) ensemble test error rate for a small learning rate ($\gamma = 2^{-10}$); (c) ensemble training loss function and (d) ensemble test error rate for a large learning rate ($\gamma = 2^{-5}$). FL Sim denotes the floating-point simulation. The weight update precision of $2B_X - \log_2(\gamma)$ (Corollary 3.1) only marginally improves the accuracy over that of $B_X - \log_2(\gamma)$. Note that for clarity, we only include comparisons with boundary cases 1, 2, and 3.

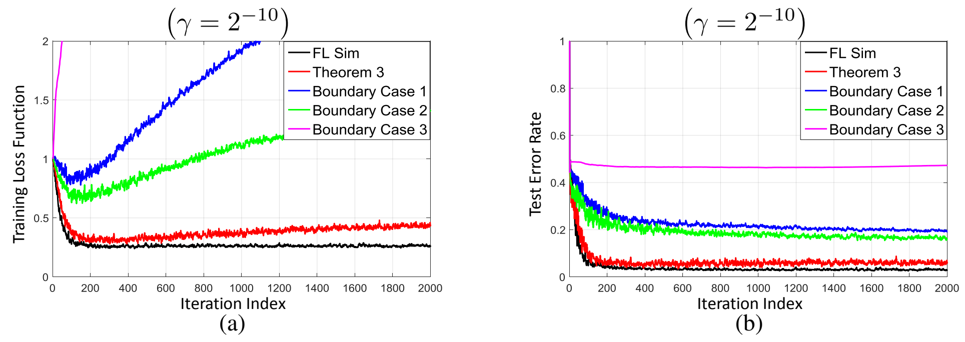


Fig. 9: Results for linear classifier training on the 'two vs. four' task for the MNIST Dataset: (a) ensemble training loss function (6) and (b) ensemble test error rate for a learning rate $\gamma = 2^{-10}$). FL Sim denotes the floating-point simulation. A weight update precision of $B_X - \log_2(\gamma)$ (Theorem 3) is enough to mimic floating-point behavior in fixed-point. Accuracy degradation is observed for lower precisions. Note that for clarity, we only include comparisons with boundary cases 1, 2, and 3.

quantization noise is a sum of independent input and weight quantization noise terms.

- Step 2: Compute the variance ($\sigma^2_{N_o}$) of this output quantization noise. Because of independence, it is the sum of the input ($\sigma^2_{q_x \to o}$) and weight ($\sigma^2_{q_w \to o}$) quantization noise variances referred to the output. For the linear classifier we have:

$$\sigma^2_{N_o} = \frac{\Delta_X^2}{12}||\mathbf{w}\_||^2 + \frac{\Delta_F^2}{12}||\mathbf{x}||^2.$$

- Step 3: Use this computed variance and Chebyshev's inequality to determine the probability of the quantization noise being larger in magnitude than the floating-point soft output z_o of the classifier. Because the quantization noise has a symmetric distribution, this probability needs to be divided by 2 (the mismatch is only caused when quantization noise and output have opposing signs). The upper bound is hence derived as follows:

$$p_m = \frac{1}{2}P(|N_o| > |z_o|) \leq \frac{\sigma^2_{N_o}}{2|z_o|^2} = \frac{\sigma^2_{q_x \to o} + \sigma^2_{q_w \to o}}{2|z_o|^2}.$$

- Step 4: Use the law of total probability over the data to obtain the averaged upper bound on p_m. For the linear classifier we obtain:

$$p_m \leq \frac{\Delta_X^2}{24}\mathbb{E}\left[\frac{||\mathbf{w}\_||^2}{|\mathbf{w}^T\mathbf{X}|^2}\right] + \frac{\Delta_F^2}{24}\mathbb{E}\left[\frac{||\mathbf{X}||^2}{|\mathbf{w}^T\mathbf{X}|^2}\right]$$

.

For each classifier type, we list the values of $\sigma^2_{q_x \to o}$, $\sigma^2_{q_w \to o}$, and z_o in Table VII. The values of E_1 and E_2 in Table II are equal to $\frac{12}{\Delta_X^2}\mathbb{E}\left[\frac{\sigma^2_{q_x \to o}}{|Z_o|^2}\right]$ and $\frac{12}{\Delta_F^2}\mathbb{E}\left[\frac{\sigma^2_{q_w \to o}}{|Z_o|^2}\right]$ for each classifier type, respectively.

Classifier type	Output quantization noise N_o	Upper bound on output quantization noise magnitude $\lvert N_o \rvert$
Linear	$\mathbf{q}_w^T\mathbf{x} + \mathbf{w}^T\mathbf{q}_x$	$2^{-B_F}\lVert\mathbf{x}\rVert\sqrt{D} + 2^{-B_X}\lVert\mathbf{w}\_\rVert\sqrt{D-1}$
NLIM	$\mathbf{q}_w^T\phi(\mathbf{x}) + \mathbf{w}^T\mathbf{q}_{\phi(x)}$	$2^{-B_F}\lVert\phi(\mathbf{x})\rVert\sqrt{D_\phi} + 2^{-B_X}\lVert\mathbf{w}\_\rVert\sqrt{D_\phi - 1}$
NLOM	$\sum_{i=1}^{N_s}\alpha_i\mathbf{q}_{s_i}^T\nabla_{\mathbf{s}_i}K(\mathbf{s}_i,\mathbf{x}) + \sum_{i=1}^{N_s}\alpha_i\mathbf{q}_x^T\nabla_{\mathbf{x}}K(\mathbf{s}_i,\mathbf{x})$	$2^{-B_F}\sqrt{D}\sum_{i=1}^{N_s}\lVert\alpha_i\nabla_{\mathbf{s}_i}K(\mathbf{s}_i,\mathbf{x})\rVert + 2^{-B_X}\sqrt{D}\left\lVert\sum_{i=1}^{N_s}\alpha_i\nabla_{\mathbf{x}}K(\mathbf{s}_i,\mathbf{x})\right\rVert$
Quadratic form	$2\mathbf{q}_x^T\mathbf{K}\mathbf{x} + \mathbf{x}^T\mathbf{q}_K\mathbf{x}$	$2^{-(B_X-1)}\lVert(\mathbf{K}\mathbf{x})\_\rVert\sqrt{D-1} + 2^{-B_F}\lVert\mathbf{x}\rVert^2 D$

TABLE VI: Output quantization noise N_o and corresponding upper bounds on its magnitude needed to prove the GLB for each classifier type. Note that $\mathbf{q}_x$, $\mathbf{q}_w$, $\mathbf{q}_{\phi(x)}$, $\mathbf{q}_{s_i}$, $\mathbf{q}_K$ are the quantization noise terms of $\mathbf{x}$, $\mathbf{w}$, $\phi(\mathbf{x})$, $\mathbf{s}_i$, and $\mathbf{K}$, respectively.

Classifier type	Input quantization noise variance referred to output $\sigma^2_{q_x\to o}$	Weight quantization noise variance referred to output $\sigma^2_{q_w\to o}$	Classifier floating-point output z_o
Linear	$\frac{\Delta_X^2}{12}\lVert\mathbf{w}\_\rVert^2$	$\frac{\Delta_F^2}{12}\lVert\mathbf{x}\rVert^2$	$\mathbf{w}^T\mathbf{x}$
NLIM	$\frac{\Delta_X^2}{12}\lVert\mathbf{w}\_\rVert^2$	$\frac{\Delta_F^2}{12}\lVert\phi(\mathbf{x})\rVert^2$	$\mathbf{w}^T\phi(\mathbf{x})$
NLOM	$\frac{\Delta_X^2}{12}\left\lVert\sum_{i=1}^{N_s}\alpha_i\nabla_{\mathbf{x}}K(\mathbf{s}_i,\mathbf{x})\right\rVert^2$	$\frac{\Delta_F^2}{12}\sum_{i=1}^{N_s}\lVert\alpha_i\nabla_{\mathbf{s}_i}K(\mathbf{s}_i,\mathbf{x})\rVert^2$	$\sum_{i=1}^{N_s}\alpha_i K(\mathbf{s}_i,\mathbf{x}) + b$
Quadratic form	$4\frac{\Delta_X^2}{12}\lVert(\mathbf{K}\mathbf{x})\_\rVert^2$	$\frac{\Delta_F^2}{12}\lVert\mathbf{x}\rVert^4$	$\mathbf{x}^T\mathbf{K}\mathbf{x}$

TABLE VII: Input ($\sigma^2_{q_x\to o}$) and weight ($\sigma^2_{q_w\to o}$) quantization noise variances referred to output and classifier floating-point output (z_o).

For the quadratic form case, we used the fact that, for a datapoint $\mathbf{x}$, $Var(\mathbf{x}^T\mathbf{q}_K\mathbf{x}) = \frac{\Delta_F^2}{12}\lVert\mathbf{x}\rVert^4$. This result is proved as follows:

$$Var(\mathbf{x}^T\mathbf{q}_K\mathbf{x}) = \mathbb{E}\left[(\mathbf{x}^T\mathbf{q}_K\mathbf{x})^2\right] = \mathbf{x}^T\mathbb{E}\left[\mathbf{q}_K\mathbf{x}\mathbf{x}^T\mathbf{q}_K^T\right]\mathbf{x}$$

$$= \mathbf{x}^T\mathbb{E}\left[\begin{bmatrix}\mathbf{q}_{K,1}^T\mathbf{x}\\ \vdots \\ \mathbf{q}_{K,D}^T\mathbf{x}\end{bmatrix}\begin{bmatrix}\mathbf{q}_{K,1}^T\mathbf{x} & \dots & \mathbf{q}_{K,D}^T\mathbf{x}\end{bmatrix}\right]\mathbf{x}$$

$$= \mathbf{x}^T\frac{\Delta_F^2}{12}\lVert\mathbf{x}\rVert^2\mathbf{I}_{D\times D}\mathbf{x} = \frac{\Delta_F^2}{12}\lVert\mathbf{x}\rVert^4$$

where $\mathbf{q}_{K,i}^T$ for $i = 1\dots D$ are the row vectors of $\mathbf{q}_K$ and $\mathbf{I}_{D\times D}$ is the identity matrix of size $D\times D$. The fourth equality holds because the quantization terms are independent of each other making the off-diagonal elements of the matrix in the third equation a product of two zero-mean independent terms.

Appendix C
Precision of Training for Boundary Cases

We analyze the learning behavior when the precision is less than the one predicted predicted by Theorem 3.

As $1-\gamma\lambda \approx 1$, we assume $(1-\gamma\lambda)w_{i,n} \approx w_{i,n}$ for $i = 1\dots D$. Let $K = B_W$ and $M = B_X$. Let $\tilde{x}_{i,n} = y_n x_n$ be the update term per dimension. The following cases describe the corresponding two's complement arithmetic:

<u>Case 1:</u> $B_W = 1 - log_2(\gamma) \Leftrightarrow \gamma = 2^{-(B_W-1)}$ then:

$$\begin{array}{rcccccc} \gamma = & 0 & .0 & \dots & 1 & & \\ w_{i,n} = & b_{w_0} & .b_{w_1} & \dots & b_{w_{K-1}} & & \\ \gamma\tilde{x}_{i,n} = & b_{x_0} & .b_{x_0} & \dots & b_{x_0} & b_{x_1}\dots b_{x_{M-1}} & \end{array}$$

where $\{b_{w_i}\}_{i=0}^{K-1}$ and $\{b_{x_i}\}_{i=0}^{M-1}$ are the binary expansions of $w_{i,n}$ and $x_{i,n}$, respectively. Hence, if $\tilde{x}_{i,n} \geq 0 \rightarrow w_{i,n+1} = w_{i,n}$, and if $\tilde{x}_{i,n} < 0 \rightarrow w_{i,n+1} = w_{i,n} - \gamma$. We hence obtain a sign-SGD behavior only when $\tilde{x}_{i,n} < 0$. Therefore, in case I, there is no way to guarantee convergence.

<u>Case 2:</u> $B_W = 2 - log_2(\gamma) \Leftrightarrow \gamma = 2^{-(B_W-2)}$ then:

$$\begin{array}{rcccccc} \gamma = & 0 & .0 & \dots & 1 & 0 & \\ w_{i,n} = & b_{w_0} & .b_{w_1} & \dots & b_{w_{K-2}} & b_{w_{K-1}} & \\ \gamma\tilde{x}_{i,n} = & b_{x_0} & .b_{x_0} & \dots & b_{x_0} & b_{x_1} & b_{x_2}\dots b_{x_{M-1}} \end{array}$$

Hence, if $\tilde{x}_{i,n} \geq 0.5 \rightarrow w_{i,n+1} = w_{i,n} + 0.5\gamma$, if $0 \leq \tilde{x}_{i,n} < 0.5 \rightarrow w_{i,n+1} = w_{i,n}$, if $-0.5 \leq \tilde{x}_{i,n} < 0 \rightarrow w_{i,n+1} = w_{i,n} - 0.5\gamma$ and, if $-1 \leq \tilde{x}_{i,n} < 0.5 \rightarrow w_{i,n+1} = w_{i,n} - \gamma$. We get a more precise but very noisy estimate of the gradient. We may observe inaccurate convergence.

<u>Case 3:</u> $B_W = -log_2(\gamma) \Leftrightarrow \gamma = 2^{-B_W}$ then:

$$\begin{array}{rcccccc} \gamma = & 0 & .0 & \dots & 0 & 1 & \\ w_{i,n} = & b_{w_0} & .b_{w_1} & \dots & b_{w_{K-1}} & & \\ \gamma\tilde{x}_{i,n} = & b_{x_0} & .b_{x_0} & \dots & b_{x_0} & b_{x_0} & b_{x_1}\dots b_{x_{M-1}} \end{array}$$

Hence, if $\tilde{x}_{i,n} \geq 0 \rightarrow w_{i,n+1} = w_{i,n}$, and if $\tilde{x}_{i,n} < 0 \rightarrow w_{i,n+1} = w_{i,n} - 2\gamma$. We again get a sign-SGD behavior only if $\tilde{x}_{i,n} < 0$ but this time the step size has doubled. Again, there is no way to guarantee any sort of convergence in this case. Things are even made worse because when updates do happen, they are two times greater than in the first case.

Acknowledgments

This work was supported in part by SONIC, one of the six SRC STARnet Centers, sponsored by MARCO and DARPA;

and by CBRIC, one of the six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

References

[1] A. Coates *et al.*, "Deep learning with COTS HPC systems," in *Proceedings of the 30th international conference on machine learning*, 2013, pp. 1337–1345.

[2] S. Han *et al.*, "Learning both Weights and Connections for Efficient Neural Network," in *Advances in Neural Information Processing Systems*, 2015, pp. 1135–1143.

[3] J. Wu, Y. Wang, Z. Wu, Z. Wang, A. Veeraraghavan, and Y. Lin, "Deep k-means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions," in *International Conference on Machine Learning*, 2018, pp. 5359–5368.

[4] S. Gupta *et al.*, "Deep Learning with Limited Numerical Precision," in *Proceedings of The 32nd International Conference on Machine Learning*, 2015, pp. 1737–1746.

[5] M. Kim *et al.*, "Bitwise Neural Networks," *arXiv preprint arXiv:1601.06071*, 2016.

[6] M. Courbariaux *et al.*, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems*, 2015, pp. 3123–3131.

[7] ——, "Binarynet: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[8] M. Rastegari *et al.*, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," *arXiv preprint arXiv:1603.05279*, 2016.

[9] C. Sakr, Y. Kim, and N. Shanbhag, "Analytical guarantees on numerical precision of deep neural networks," in *International Conference on Machine Learning*, 2017, pp. 3007–3016.

[10] C. Sakr and N. Shanbhag, "An analytical method to determine minimum per-layer precision of deep neural networks," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 1090–1094.

[11] M. Goel *et al.*, "Finite-precision analysis of the pipelined strength-reduced adaptive filter," *Signal Processing, IEEE Transactions on*, vol. 46, no. 6, pp. 1763–1769, 1998.

[12] C. Caraiscos and B. Liu, "A roundoff error analysis of the lms adaptive algorithm," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 1, pp. 34–41, 1984.

[13] W. Sung and K.-I. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *IEEE transactions on Signal Processing*, vol. 43, no. 12, pp. 3087–3090, 1995.

[14] R. Rocher *et al.*, "Accuracy evaluation of fixed-point LMS algorithm," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, vol. 5. IEEE, 2004, pp. V–237.

[15] C. Shi and R. W. Brodersen, "An automated floating-point to fixed-point conversion methodology," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, vol. 2. IEEE, 2003, pp. II–529.

[16] R. Gupta and A. O. Hero, "Transient behavior of fixed point LMS adaptation," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, vol. 1. IEEE, 2000, pp. 376–379.

[17] P. S. Diniz *et al.*, "Analysis of LMS-Newton adaptive filtering algorithms with variable convergence factor," *IEEE Transactions on Signal Processing*, vol. 43, no. 3, pp. 617–627, 1995.

[18] C. Sakr *et al.*, "Minimum precision requirements for the SVM-SGD learning algorithm," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017.

[19] A. Asuncion *et al.*, "UCI machine learning repository," 2007.

[20] Y. LeCun *et al.*, "The MNIST database of handwritten digits," 1998.

[21] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[22] K. H. Lee *et al.*, "Low-energy formulations of support vector machine kernel functions for biomedical sensor applications," *Journal of Signal Processing Systems*, vol. 69, no. 3, pp. 339–349, 2012.

[23] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[24] B. McMahan and D. Ramage, "Federated learning: Collaborative machine learning without centralized training data," 2017.

[25] C.-C. Chang *et al.*, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[26] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," Google Brain, Tech. Rep., 2016, arXiv preprint. [Online]. Available: https://arxiv.org/abs/1605.08695

Charbel Sakr is a PhD student at the University of Illinois at Urbana-Champaign. He obtained his Engineering degree from the American University of Beirut in 2015 graduating with High Distinction. He then joined the University of Illinois where he is now a PhD candidate in the department of Electrical and Computer Engineering. His research interests are in resource-constrained machine learning, with a focus on analysis and implementation of reduced precision algorithms and models. He is the recipient of the best in session award at Techcon 2017 and of the Rambus fellowship from the ECE department at the University of Illinois both in 2018-2019 and 2019-2020.

Yongjune Kim received the B.S. and M.S. degrees in Electrical and Computer Engineering from Seoul National University, Seoul, South Korea, in 2002 and 2004, respectively, and the Ph.D. degree in Electrical and Computer Engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2016. From 2007 to 2011, he was with Samsung Electronics and Samsung Advanced Institute of Technology, South Korea. From 2016 to 2018, he was a postdoctoral scholar with the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL, USA. Currently, he has been with Western Digital Research, Milpitas, CA, USA. His research interests include coding and information theory, energy-efficient computing, and machine learning. He received the IEEE Data Storage Best Student Paper Award, the Best Paper Award of the IEEE International Conference on Communications (ICC), the Best Paper Award (honorable mention) of the IEEE International Symposium on Circuits and Systems (ISCAS), and the Best Paper Award of the Samsung Semiconductor Technology Symposium.

Naresh R. Shanbhag (F'06) is the Jack Kilby Professor of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign. He received his Ph.D. degree from the University of Minnesota (1993) in Electrical Engineering. From 1993 to 1995, he worked at AT&T Bell Laboratories at Murray Hill where he led the design of high-speed transceiver chip-sets for very high-speed digital subscriber line (VDSL), before joining the University of Illinois at Urbana- Champaign in August 1995. He has held visiting faculty appointments at the National Taiwan University (Aug.-Dec. 2007) and Stanford University (Aug.-Dec. 2014). His research interests are in the design of energy-efficient integrated circuits and systems for communications, signal processing and machine learning. He has more than 200 publications in this area and holds thirteen US patents.

Dr. Shanbhag received the 2018 SIA/SRC University Research Award, became an IEEE Fellow in 2006, received the 2010 Richard Newton GSRC Industrial Impact Award, the IEEE Circuits and Systems Society Distinguished Lecturership in 1997, the National Science Foundation CAREER Award in 1996, and multiple best paper awards. In 2000, Dr. Shanbhag co-founded and served as the Chief Technology Officer of Intersymbol Communications, Inc., (acquired in 2007 by Finisar Corporation) a semiconductor start-up that provided DSP-enhanced mixed-signal ICs for electronic dispersion compensation of OC-192 optical links. From 2013-17, he was the founding Director of the Systems On Nanoscale Information fabriCs (SONIC) Center, a 5-year multi- university center funded by DARPA and SRC under the STARnet program.