Mixed-Precision Kernel Recursive Least Squares

Lee, J., Nikolopoulos, D. S., & Vandierendonck, H. (2020). Mixed-Precision Kernel Recursive Least Squares. *IEEE Transactions on Neural Networks and Learning Systems*. Advance online publication. https://doi.org/10.1109/TNNLS.2020.3041677

Published in:
IEEE Transactions on Neural Networks and Learning Systems

Document Version:
Peer reviewed version

Mixed Precision Kernel Recursive Least Squares

JunKyu Lee, Dimitrios S. Nikolopoulos, Hans Vandierendonck
{junkyu.lee, d.nikolopoulos, h.vandierendonck}@qub.ac.uk

***Abstract*—Kernel Recursive Least Squares (KRLS) is a widely used online machine learning algorithm for time series predictions. In this paper, we present the mixed precision KRLS, producing equivalent prediction accuracy to double precision KRLS with a higher training throughput and a lower memory footprint. The mixed precision KRLS applies single precision arithmetic to the computation components being not only numerically resilient but also computationally intensive. Our mixed precision KRLS demonstrates the 1.32, 1.15, 1.29, 1.09, and 1.08× training throughput improvements using 24.95, 24.74, 24.89, 24.48, and 24.20% less memory footprint without losing any prediction accuracy compared to double precision KRLS for a 3 dimensional non-linear regression, a Lorenz chaotic time series, a Mackey-Glass chaotic time series, a sunspot number time series, and a sea surface temperature time series respectively.**

***Index Terms*—Mixed Precision Training; Kernel Method; Online Learning; Budget Machine Learning**

I. Introduction

The recent tremendous technology improvement from networks and storage devices has provided the machine learning community with the opportunity to utilize large data sources. Along with this trend, online learning algorithms have emerged to deal with abundant training data with moderate computing resources [1]. Unlike a batch learning, online learning algorithms adapt the weights per training sample in real time. By consequence, online learning algorithms often require higher training throughput using less memory footprint. For example, higher throughput is required to deal with real-time data flood [2] and the implementation of an online learning algorithm should be fit into the memory budget given to an embedded system or a wearable device [3]. Many time series applications often use online learning algorithms, requiring higher training throughput given computational resources.

Kernel Recursive Least Squares (KRLS) [4] is a widely used online learning algorithm thanks to its simple RLS mechanism, no local minima, fast convergence, and good prediction accuracy [3], [5]–[9]. However, KRLS still requires costly operations and memory footprint depending on the training data, limiting its applicability to real-time systems [10]. For example, the training time cost for KRLS is $O(m_t^2)$, where m_t is the number of the selected support vectors. The required memory footprint is also $O(m_t^2)$. Most research efforts to overcome such limitations amended the KRLS algorithm of [4]. For example, Kernel Normalized Least Mean Squares (KNLMS) [10] utilized least mean squares algorithm instead of recursive least squares, limiting the prediction accuracy. The fixed memory budget KRLS was proposed for a computing platform having a relatively limited memory resource [3]. The sliding windows KRLS [5] considered only recent training samples to track abrupt changes in a time-varying system. However, most KRLS variants require more hyperparameters compared to the KRLS of [4] to fit the amended algorithms for particular applications, limiting generalization performance (i.e., a discrepancy between in-sample error and out-sample error). This work takes a different approach in order to improve the training throughput of KRLS, which simultaneously achieves a lower memory footprint without increasing the number of hyperparameters.

Applying reduced precision arithmetic (i.e., lower precision arithmetic than the precision arithmetic used for the baseline implementation) to KRLS can be a promising approach towards high throughput training capability since reduced precision arithmetic reduces the memory footprint and the time spent transferring data across buses and interconnections [11], [12] and allows to pack arithmetic units densely in SIMD or matrix arrangement. E.g., the NVIDIA V100 processor can deliver a peak throughput of 125 TFLOPs on 4×4 matrix multiplication with 16-bit floating point arithmetic [13]. While arithmetic precision is an important lever to accelerate machine learning applications, many applications cannot tolerate uniformly reduced precision arithmetic as the lack of sufficient precision would result in inaccurate predictions. For example, the rounding errors magnified by reduced precision arithmetic can break an essential mathematical property of an algorithm where exact arithmetic is assumed, resulting in inaccurate predictions [14], [15]. Therefore, a mixed precision arithmetic approach is necessary. The idea of the mixed precision approach is to apply reduced precision only to the parts of the algorithm that are resilient to reduced precision, while others are performed at high precision to keep the equivalent accuracy. The throughput gain by the mixed precision approach depends on the relative computational portion for the part where lower precision arithmetic is applied. Higher portion, higher gain. Therefore, a mixed precision method applies reduced precision arithmetic to the computation parts being not only numerically resilient but also computationally intensive [16].

Some of the related work discusses how to apply reduced/mixed precision arithmetic for deep learning [17], [18]. The work of [17] discusses the impact of the reduced precision arithmetic with a stochastic rounding scheme on the training throughput for deep learning. The mixed precision training of [18] demonstrates equivalent accuracy results to double precision training empirically. However, most related works lack theoretical backgrounds of how the mixed precision approaches keep the equivalent prediction accuracy to double precision training. Lack of theoretical background limits the

J. Lee, D. S. Nikolopoulos, and H. Vandierendonck are with the Institute of Electronics, Communications and Information Technology (ECIT), Queens University Belfast, Northern Ireland, UK.

Manuscript received XXX, 2019; revised .

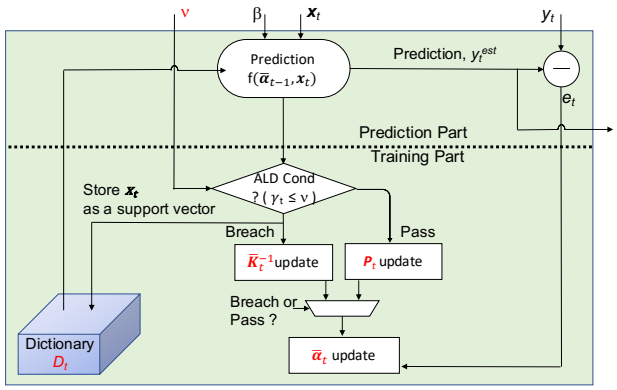

Fig. 1. Dataflow of KRLS

applications of the mixed precision arithmetic to machine learning applications. In contrast to such related work, we present the first mixed precision KRLS exploiting the inherent characteristics of the KRLS model chosen by cross-validation.

The main contributions in this paper are as follows:

- This paper presents the first mixed precision KRLS producing a higher training throughput with a lower memory footprint, while keeping the equivalent prediction accuracy compared to double precision KRLS.
- This paper discusses the condition number of the kernel matrix according to precision arithmetic and numerically sensitive computing components in KRLS.
- Our mixed precision KRLS brings 1.32, 1.15, 1.29, 1.09, and 1.08× training throughput improvements using 24.95, 24.74, 24.89, 24.48, and 24.20% lower memory footprint without losing prediction accuracy for a 3D nonlinear regression, a Lorenz time series, a Mackey-Glass 30 (MG30) time series, a sunspot number time series, and a sea surface temperature time series respectively.

We describe the KRLS algorithm in Section II, numerical properties for finite precision arithmetic KRLS in Section III, the mixed precision KRLS in Section IV, the experimental evaluation in Section V, related work in Section VI, and conclude this paper in Section VII.

II. Kernel Recursive Least Squares

The KRLS of [4] is a kernelized Recursive Least Squares (RLS) algorithm. Fig. 1 describes the dataflow of KRLS. KRLS takes an input sample $\mathbf{x}_t$, makes a prediction y_t^{est} based on $\mathbf{x}_t$, evaluates Approximation Linear Dependency (ALD) condition, and updates the weights $\bar{\boldsymbol{\alpha}}_t$ and the dictionary $\mathcal{D}_t$ according to an ALD condition evaluation result. KRLS utilizes two internal matrices (i.e., an inverse kernel matrix $\bar{\mathbf{K}}_t^{-1}$ and a covariance matrix $\mathbf{P}_t$) to update $\bar{\boldsymbol{\alpha}}_t$ at time t used for a prediction for a next sample. A KRLS model (e.g., hyperparameter set, ν and β) chosen by cross-validation depends on arithmetic precision. For example, if double precision is used for KRLS implementation, cross-validation returns a KRLS model that has the best prediction accuracy for double precision implementation, given a cross-validation data set. Applying lower precision arithmetic for entire computation to a double precision KRLS model can result in failure of prediction accuracy due to rounding errors magnified from the reduced precision arithmetic. Therefore, it is necessary to characterize the rounding error propagation properties (i.e., numerical properties) for individual computing components to see which computing components can be resilient to reduced precision arithmetic. This section discusses kernel method and KRLS algorithm, preparatory to discussing the numerical properties of KRLS.

A. Kernel Method

A kernel method maps an input space $\boldsymbol{\chi}$ into a higher dimensional Hilbert space (i.e., feature space) $\boldsymbol{\mathcal{H}}$: $\mathbf{x}_t \in \boldsymbol{\chi} \mapsto \boldsymbol{\phi}(\mathbf{x}_t) \in \boldsymbol{\mathcal{H}}$, where $\mathbf{x}_t$ is an $l_{in} \times 1$ vector and l_{in} is the dimensionality of the input $\mathbf{x}_t$. An inner product for a pair of mapped inputs $\boldsymbol{\phi}(\mathbf{x}_i)$ and $\boldsymbol{\phi}(\mathbf{x}_j)$ (i.e., features) is defined: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle$, where $\kappa(\cdot,\cdot)$ is an inner product between the two features $\boldsymbol{\phi}(\mathbf{x}_i)$ and $\boldsymbol{\phi}(\mathbf{x}_j)$ in $\boldsymbol{\mathcal{H}}$ using a Mercer kernel [19]. The inner products in $\boldsymbol{\mathcal{H}}$ are used in KRLS to predict a target. Every inner product using a Mercer kernel produces a positive value or zero (i.e., a positive semi-definite kernel). For example, the Gaussian kernel is one of well-known Mercer kernels and the inner product using the kernel is defined as:

$$\kappa(\mathbf{x}_t, \mathbf{x}_i) = \langle \boldsymbol{\phi}(\mathbf{x}_t), \boldsymbol{\phi}(\mathbf{x}_i) \rangle = exp^{-(\mathbf{x}_t - \mathbf{x}_i)^T (\mathbf{x}_t - \mathbf{x}_i)/(2\beta^2)} \tag{1}$$

where β is referred to the *kernel width parameter*. The inner product using a Mercer kernel is referred to "kernel trick", since the inner product in $\boldsymbol{\mathcal{H}}$ using Eq. (1) is performed without visiting $\boldsymbol{\mathcal{H}}$ [19]. This paper considers Gaussian kernel for KRLS, but any other Mercer kernel such as a polynomial kernel can also be used for KRLS [4].

In general, kernel methods do not employ enormous number of parameters unlike deep learning. The merit of KRLS comes from simple RLS mechanism, requiring relatively lower number of weights (e.g., 100 - 1000) for predictions [4], so that a higher training throughput can be obtained for online learning applications. The number of parameters in KRLS is directly related to the dictionary size that depends on the kernel width and the ALD threshold.

B. Kernel Recursive Least Squares Algorithm [4]

KRLS is a linear predictor on the support feature vector basis in $\boldsymbol{\mathcal{H}}$, performing a prediction at time t on an arrival of a new sample $\mathbf{x}_t$:

$$y_t^{est} = \Sigma_i^{m_{t-1}} \bar{\alpha}_i \langle \boldsymbol{\phi}(\bar{\mathbf{x}}_i), \boldsymbol{\phi}(\mathbf{x}_t) \rangle \tag{2}$$

where $\bar{\mathbf{x}}_i$ is the i^{th} support vector in $\mathcal{D}_{t-1}$ having m_{t-1} support vectors (i.e., the number of breaches for ALD condition of (3) until time $(t-1)$) and $\bar{\alpha}_i$ is the i^{th} component in $\bar{\boldsymbol{\alpha}}_{t-1}$. In order to select the support vectors, KRLS evaluates the ALD condition of:

$$\gamma_t := \min_{\mathbf{a}_t} \| \sum_{j=1}^{m_{t-1}} a_j \boldsymbol{\phi}(\bar{\mathbf{x}}_j) - \boldsymbol{\phi}(\mathbf{x}_t) \|^2 \le \nu \tag{3}$$

on every training sample at time t, where ν is referred to the *ALD threshold parameter*, $\mathbf{a}_t = (a_1, \ldots, a_{m_{t-1}})^T$, and $\bar{\mathbf{x}}_j$s are the ALD breach vectors (i.e., the support vectors) stored in $\mathcal{D}_{t-1}$. The γ_t indicates the square of the Euclidean distance between a new feature vector, $\boldsymbol{\phi}(\mathbf{x}_t)$, and the feature vector generated with support feature vector bases, which is closest to $\boldsymbol{\phi}(\mathbf{x}_t)$. If the condition of (3) does not hold, the sample $\mathbf{x}_t$ is stored in $\mathcal{D}_t$ as a new support vector.

The γ_t in (3) can be represented (refer to [4]):

$$\gamma_t = 1 - \bar{\mathbf{a}}_t^T \bar{\mathbf{k}}_{t-1}(\mathbf{x}_t) \tag{4}$$

where the kernel vector $\bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)$ and the $\bar{\mathbf{a}}_t$ can be sought using:

$$\bar{\mathbf{k}}_{t-1}(\mathbf{x}_t) = e^{-(\mathbf{x_t} - \bar{\mathbf{x}}_{1:m_{t-1}})^T (\mathbf{x_t} - \bar{\mathbf{x}}_{1:m_{t-1}})/(2\beta^2)} \tag{5}$$

$$\bar{\mathbf{a}}_t = \bar{\mathbf{K}}_{t-1}^{-1} \bar{\mathbf{k}}_{t-1}(\mathbf{x}_t) \tag{6}$$

where $\bar{\mathbf{K}}_{t-1} (= \bar{\boldsymbol{\Phi}}_{t-1}^T \bar{\boldsymbol{\Phi}}_{t-1})$ is referred to a kernel matrix and $\bar{\boldsymbol{\Phi}}_{t-1} = [\boldsymbol{\phi}(\bar{\mathbf{x}}_1), ..., \boldsymbol{\phi}(\bar{\mathbf{x}}_{t-1})]$. The prediction y_t^{est} is computed by an inner product:

$$y_t^{est} = \bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)^T \bar{\boldsymbol{\alpha}}_{t-1} \tag{7}$$

which is equivalent to Eq. (2). Once the target y_t is known, the prediction error e_t is computed:

$$e_t = y_t - y_t^{est} \tag{8}$$

The resultant KRLS algorithm is described in Fig. 1 [4].

The β and ν are chosen by a model selection process. We do not consider an explicit regularization parameter for KRLS since the regularization is supported by the ν employment [4], [20]. Algorithm 1 includes the parts Ⓢs where single precision arithmetic operations are applied for mixed precision KRLS, which will be discussed in detail in Section IV. The data type for a variable for mixed precision KRLS is represented with the parenthesis ( ) located at left side of the variable. For example, the (D), (S), (I) represent double precision, single precision, and integer data format respectively. The (S/D) means that the two data types of single and double precision are required for the variable for mixed precision KRLS.

Algorithm 1 Kernel Recursive Least Squares Algorithm [4]

Set Hyperparameters: ν and β

Initialization:
Get an initial sample $(\mathbf{x}_1, y_1)$
(D) $\mathcal{D}_1(:,1) <= \mathbf{x_1}$; (D) $\bar{\mathbf{K}}_1^{-1} = [1.0]$; (D) $\bar{\boldsymbol{\alpha}}_1 = (y_1)$;
(S) $\mathbf{P}_1 = [1.0]$; (I) $m_1 = 1$;

Training:
for $t = 2, 3, \ldots$ **do**
  Get a training sample $(\mathbf{x}_t, y_t)$
  Compute a kernel vector (D) $\bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)$ using Eq. (5).
  Compute an inference (D) y_t^{est} using Eq. (7)
  Compute a vector (S/D) $\bar{\mathbf{a}}_t$ using Eq. (6)
  Compute an ALD parameter (D) γ_t using Eq. (4)
  Compute a prediction error (D) e_t using Eq. (8)
  if $\gamma_t > \nu$ **then**
    ALD Breach Path
    $m_t = m_{t-1} + 1$
    $\mathcal{D}_t(:, m_t) = \mathbf{x_t}$
    $\bar{\mathbf{K}}_t^{-1} = \frac{1}{\gamma_t} \begin{bmatrix} \gamma_t \bar{\mathbf{K}}_{t-1}^{-1} + \bar{\mathbf{a}}_t \bar{\mathbf{a}}_t^T & \bar{\mathbf{a}}_t \\ -\bar{\mathbf{a}}_t^T & 1 \end{bmatrix}$
    $\mathbf{P}_t = \begin{bmatrix} \mathbf{P}_{t-1} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}$ Ⓢ
    $\bar{\boldsymbol{\alpha}}_t = \begin{bmatrix} \bar{\boldsymbol{\alpha}}_{t-1} - \frac{e_t}{\gamma_t} \bar{\mathbf{a}}_t \\ \frac{e_t}{\gamma_t} \end{bmatrix}$
  else
    ALD Pass Path
    $m_t = m_{t-1}$
    $\mathcal{D}_t = \mathcal{D}_{t-1}$
    $\bar{\mathbf{K}}_t^{-1} = \bar{\mathbf{K}}_{t-1}^{-1}$
    $(S)\mathbf{p}_t = \mathbf{P}_{t-1} \bar{\mathbf{a}}_t$ Ⓢ (9)
    $(S/D)\mathbf{q}_t = \frac{\mathbf{p}_t}{1 + \bar{\mathbf{a}}_t^T \mathbf{p}_t}$ Ⓢ (10)
    $\mathbf{P}_t = \mathbf{P}_{t-1} - \mathbf{q}_t \mathbf{p}_t^T$ Ⓢ (11)
    $\bar{\boldsymbol{\alpha}}_t = \bar{\boldsymbol{\alpha}}_{t-1} + e_t \bar{\mathbf{K}}_t^{-1} \mathbf{q}_t$ (12)
  end if
end for

III. Numerical Properties of KRLS

This section discusses floating point arithmetic and numerical properties of KRLS driven by floating point arithmetic.

A. Floating Point Arithmetic: IEEE 754

1) Data Format: The IEEE 754 floating point data format consists of three parts: sign, exponent, and mantissa. For example, a floating point number having a p-bit mantissa, an e-bit exponent, and a 1 sign bit are represented by:

$$sign \times (1 \times 2^0 + d_1 \times 2^{-1} + ... + d_p \times 2^{-p}) \times 2^{exponent - bias} \tag{13}$$

where $d_1 ... d_p$ represent binary digits, the 1 associated with the coefficient 2^0 is referred to the hidden 1, the $exponent$ is stored in offset notation, and the $bias$ is a positive constant. For double precision format, $p = 52$, $e = 11$, and $bias = 1,023$ and for single precision format, $p = 23$, $e = 8$, and $bias = 127$. The machine epsilon ϵ_{mach} is defined as $2^{-(p+1)}$ for a rounding nearest mode. For example, single precision machine epsilon is 2^{-24}, and double precision machine epsilon is 2^{-53}.

2) Rounding Errors based on IEEE 754: IEEE 754 standard [21] requires exact rounding for the four floating point arithmetic types such as addition, subtraction, multiplication, and division. In other words, the floating point arithmetic result should be identical to the one obtained from the final rounding after the exact calculation. Based on the IEEE 754 rounding to nearest mode, floating point arithmetic should follow Eq. (14) [22], [23].

$$fl(x_1 \odot x_2) = (x_1 \odot x_2)(1 + \epsilon_r) \tag{14}$$

where $|\epsilon_r| \leq \epsilon_{mach}$, ϵ_r represents an unavoidable rounding error quantity due to finite precision arithmetic, $\odot$ is one of the four floating point arithmetic operations, and $fl(\cdot)$ represents the result from the floating point arithmetic. For example, a floating point arithmetic for $x_3 = x_1 + x_2$ will be represented

as: $fl(x_1 + x_2) = x_1 + x_2 + \epsilon_r(x_1 + x_2) = x_3 + \delta x_3$, where $\delta x_3 = \epsilon_r(x_1 + x_2)$ represents the rounding error due to floating point arithmetic. We will use $fl(x)$ notation to represent the computed x by floating point arithmetic from this point forward.

3) Condition Number of Matrix: The condition number of a matrix A can be defined as:

$$Cond(A) = \|A\| \cdot \|A^{-1}\| \tag{15}$$

where $\|\cdot\|$ is a norm that measures a size of a vector or a matrix [23]. The condition number is often used to describe the error bound for the computed solution for a function. The condition numbers follow norms. For example, the 2 norm condition number can be represented by the largest singular value of the matrix A, $\sigma_{max}(A)$, divided by its smallest singular value, $\sigma_{min}(A)$ [23]:

$$Cond_2(A) = \|A\|_2 \cdot \|A^{-1}\|_2 = \sigma_{max}(A)/\sigma_{min}(A) \tag{16}$$

4) Numerical Stability [23]: The numerical stability used in numerical linear algebra indicates the behavior of an algorithm with the input perturbation. For example, an algorithm $\tilde{f}$ to solve a problem f is numerically stable if for all data $\mathbf{x} \in \mathcal{X}$,

$$\frac{\|\tilde{f}(\mathbf{x}) - f(\tilde{\mathbf{x}})\|}{\|f(\tilde{\mathbf{x}})\|} = O(\epsilon_{mach}) \tag{17}$$

for some perturbed input $\tilde{\mathbf{x}}$ satisfying:

$$\frac{\|\mathbf{x} - \tilde{\mathbf{x}}\|}{\|\mathbf{x}\|} = O(\epsilon_{mach}) \tag{18}$$

where $O(\epsilon_{mach})$ in Eq. (17) means that there exists a positive constant C satisfying: $\frac{\|\tilde{f}(\mathbf{x}) - f(\tilde{\mathbf{x}})\|}{\|f(\tilde{\mathbf{x}})\|} \leq C\epsilon_{mach}$, and the magnitude of C mainly depends on the dimension of $\mathbf{x}$. In words, Eq. (17) implies that "A stable algorithm gives nearly the right answer to nearly the right question." in [23].

B. Numerical Properties of Finite Precision Arithmetic KRLS

First, we describe numerically sensitive computing components in KRLS in Numerical Property 1 (NP1) due to the subtraction between two nearby numbers.

Numerical Property 1 (Cancellations of most significant digits in e_t and γ_t Computations)**.** *The computations of e_t and γ_t are exposed to cancellation of the most significant digits.*

Proof sketch: Finite precision arithmetic of e_t in Eq. (8) is exposed to cancellation [22] of the most significant digits when the estimation becomes closer to a real target. For example, suppose that $y_t = +(1.111) \times 2^0$ and $y_t^{est} = +(1.110) \times 2^0$ based on Eq. (13). Notice that y_t^{est} is a computed quantity which contains rounding error from previous computations. Assuming that the real value of $y_t^{est*} = +(1.11001) \times 2^0$ (i.e., the magnitude of rounding error is 2^{-5}.), the computed value of $fl(y_t - y_t^{est}) = 1.000 \times 2^{-3}$, while the real value of $(y_t - y_t^{est*})$ should be 1.100×2^{-4}. This results in the relative error of 33.3% of e_t. The γ_t in Eq. (4) is also exposed to cancellation when a new sample passes the ALD condition □.

If such erroneous $fl(\gamma_t)$ is used by the kernel matrix updates, KRLS can completely lose its learning ability. Therefore, a higher precision arithmetic is recommended for the e_t and the γ_t computations. Next, we describe the relation between the condition number of kernel matrix and ν in Numerical Property 2 (NP2).

Numerical Property 2 (Condition Number of Kernel Matrix according to ν)**.** $Cond_2(\bar{\mathbf{K}}_t^{-1}) \propto \frac{\|\bar{\mathbf{K}}_t\|_2}{\nu}$.

Proof sketch: Any two feature vectors in the dictionary are $\sqrt{\nu}$-separated (refer to [4]):

$$\|\boldsymbol{\phi}_{i,j}^{diff}\|_2 = \| \boldsymbol{\phi}(\bar{\mathbf{x}}_i) - \boldsymbol{\phi}(\bar{\mathbf{x}}_j) \|_2 > \sqrt{\nu} \tag{19}$$

where $i \neq j$. If $\nu = 0$, $\bar{\mathbf{K}}_t$ can be a singular matrix since it is possible that a $\boldsymbol{\phi}(\bar{\mathbf{x}}_i)$ can be approximated with a linear combination of other $\boldsymbol{\phi}(\bar{\mathbf{x}}_j)$s. Therefore, a smaller ν generally results in the better approximation for $\boldsymbol{\phi}(\bar{\mathbf{x}}_t)$ using the stored support feature vectors, while increasing the condition numbers of $\bar{\mathbf{K}}_t^{-1}$. For a mathematical proof, we adapt the singular value perturbation analysis discussed in [23]. When the size of the kernel matrix is saturated, $\|\boldsymbol{\phi}_{i,j}^{diff}\|_2 \rightarrow \sqrt{\nu}$ in (19). We can make $\bar{\mathbf{K}}_t$ a singular matrix by adding or subtracting the residual vector $\boldsymbol{\phi}_{i,j}^{diff}$ to $\boldsymbol{\phi}(\bar{\mathbf{x}}_i)$ in $\bar{\boldsymbol{\Phi}}_t$. Next, we denote $\sigma_{min}(\bar{\mathbf{K}}_t + \delta\bar{\mathbf{K}}_t)$ for the minimum singular value for the perturbed system $\bar{\mathbf{K}}_t + \delta\bar{\mathbf{K}}_t$, where $\bar{\mathbf{K}}_t + \delta\bar{\mathbf{K}}_t$ is a singular matrix (i.e., $\sigma_{min}(\bar{\mathbf{K}}_t + \delta\bar{\mathbf{K}}_t) = 0$). From [23], the difference between the two minimum singular values from the original matrix and the perturbed matrix is bounded to the perturbed quantity: $|\sigma_{min}(\bar{\mathbf{K}}_t + \delta\bar{\mathbf{K}}_t) - \sigma_{min}(\bar{\mathbf{K}}_t)| \leq \|\delta\bar{\mathbf{K}}_t\|_2 = O(\nu) \leq c_2\nu$ where c_2 is a positive value primarily depending on a matrix size. This leads to : $\sigma_{min}(\bar{\mathbf{K}}_t) \leq c_2\nu$. Therefore, $Cond_2(\bar{\mathbf{K}}_t) = \frac{\sigma_{max}(\bar{\mathbf{K}}_t)}{\sigma_{min}(\bar{\mathbf{K}}_t)} \geq \frac{\|\bar{\mathbf{K}}_t\|_2}{c_2\nu}$ and $Cond_2(\bar{\mathbf{K}}_t^{-1}) = Cond_2(\bar{\mathbf{K}}_t)$ based on Eq. (16). □

Corollary 2.1. *The maximum relative error of $\|\delta\bar{\mathbf{a}}_t\|_2/\|\bar{\mathbf{a}}_t\|_2$ in Eq. (6) is in proportion to $Cond_2(\bar{\mathbf{K}}_{t-1}^{-1})$.*

Proof sketch [23]: The maximum attainable error for Eq. (6) according to the perturbation $\frac{\|\delta\bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)\|_2}{\|\bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)\|_2}$ is:

$$\begin{aligned} max(\|\delta\bar{\mathbf{a}}_t\|_2/\|\bar{\mathbf{a}}_t\|_2) &= \sup_{\delta\bar{k}_{t-1}(x_t)} \frac{\|\bar{\mathbf{K}}_{t-1}^{-1}\delta\bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)\|_2}{\|\bar{\mathbf{K}}_{t-1}^{-1}\bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)\|_2} \\ &= \frac{\sigma_{max}(\bar{\mathbf{K}}_{t-1}^{-1})\|\delta\bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)\|_2}{\sigma_{min}(\bar{\mathbf{K}}_{t-1}^{-1})\|\bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)\|_2} \\ &\leq Cond_2(\bar{\mathbf{K}}_{t-1}^{-1})\frac{\|\delta\bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)\|_2}{\|\bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)\|_2} \end{aligned} \tag{20}$$

□

Therefore, high precision arithmetic is recommended for Eq. (5) and (6) when ν is low (i.e., when $Cond_2(\bar{\mathbf{K}}_{t-1}^{-1})$ is high.).

The γ_t in Eq. (4) should be positive to maintain KRLS learning capability since Eq. (4) is mathematically identical to Eq. (3). Corollary 2.2 describes potential breach of the positiveness of γ_t due to large $\|\delta\bar{\mathbf{a}}_t\|_2$.

Corollary 2.2 ($fl(\gamma_t)$ may be negative)**.** *It is possible that $fl(\gamma_t) < 0$ due to a large $\|\delta\bar{\mathbf{a}}_t\|_2$.*

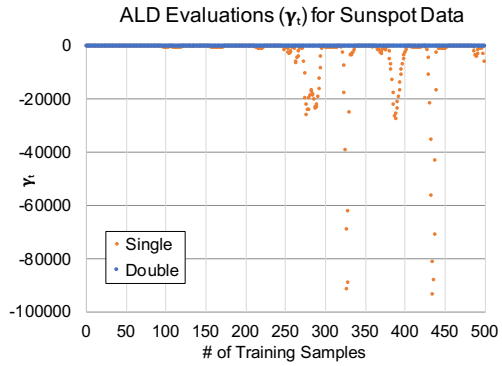

Fig. 2. γ_ts for Sunspot data in Section V-E

Proof sketch: $fl(\gamma_t) \approx 1 - \bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)^T(\bar{\mathbf{a}}_t + \delta\bar{\mathbf{a}}_t) = \gamma_t + \delta\gamma_t$, where the error $\delta\gamma_t = -\bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)^T\delta\bar{\mathbf{a}}_t$. When $\gamma_t < |\delta\gamma_t|$, negative $fl(\gamma_t)$s can occur. For example, if $\bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)^T = [1/2, 1/3], \bar{\mathbf{a}}_t^T = [1 - 2\times10^{-4}, 3/2]$, and $\delta\bar{\mathbf{a}}_t^T = [2\times10^{-4}, 3\times10^{-4}]$, then $\gamma_t = 10^{-4}$ and $\delta\gamma_t = -2\times10^{-4}$, incurring $fl(\gamma_t) = -10^{-4}$. □

Demonstration on sunspot time series: Fig. 2 represents the γ_ts computed in Eq. (4) according to single and double precision arithmetic KRLS for the sunspot data discussed in Section V-E. The double precision arithmetic correctly calculates the distance γ_t as a positive number, while single precision arithmetic breaks the property when the number of training samples exceeds 17. When the properties initially were broken, the $Cond_2(\bar{\mathbf{K}}_t^{-1})$ was $10^{7.5}$. The incorrect γ_ts from single precision arithmetic used to update $\bar{\mathbf{K}}_t^{-1}$s results in destroying the KRLS learning ability (as discussed in Fig. 16 in Section V-E).

IV. Mixed Precision KRLS

The idea of our mixed precision KRLS is to let double precision arithmetic deal with numerically sensitive computation components, while single precision arithmetic deals with numerically resilient computation components. For a baseline, double precision KRLS is chosen to be compared with mixed precision KRLS in terms of the prediction accuracy, the training throughput, and the memory footprint. Our mixed precision KRLS is defined as follows.

Mixed Precision KRLS
Single precision arithmetic is applied for Ⓢ parts in Fig. 1, while double precision arithmetic is applied for the other computations.

The mixed precision KRLS exploits inherent characteristics of the KRLS model chosen by cross-validation as shown in Fig. 3. Cross-validation suggests an appropriate KRLS model producing the minimum Mean Square Error (MSE) based on an applied arithmetic precision (e.g., double precision for a baseline) and a cross-validation data set. In general, a chosen KRLS model is subjected to have a low ν since a low ν can approximate a new incoming sample sufficiently well with the support vectors in the dictionary, resulting in a low prediction error. The low ν chosen by cross-validation incurs

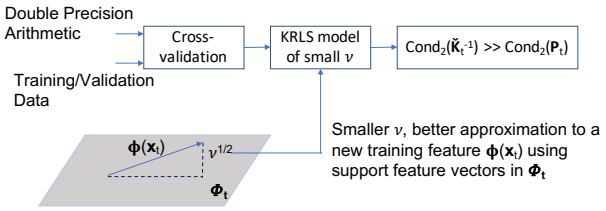

Fig. 3. The KRLS model chosen by Cross-Validation

a higher condition number of $\tilde{\mathbf{K}}_t^{-1}$ than $\mathbf{P}_t$ if double precision arithmetic is used for KRLS. Such property of the chosen KRLS model makes our mixed precision approach produce the equivalent prediction accuracy to double precision KRLS. For the throughput gain, the application of reduced precision arithmetic considers the computations only for the ALD pass path since every training in KRLS follows either ALD pass or ALD breach according to ALD condition evaluation of Eq. (3) and follows the ALD pass path dominantly (E.g., $\sim 100\%$ in practice.) [4].

The Mixed Precision Property 1 (MP1) describes the prediction accuracy property of the mixed precision arithmetic technique.

Mixed Precision Property 1. *Equivalent Prediction Accuracy to Double Precision KRLS*
The mixed precision KRLS produces equivalent accuracy to double precision KRLS if $\eta_1\epsilon_S Cond_2(\mathbf{P}_{t-1}) \lesssim Cond_2(\bar{\mathbf{K}}_{t-1}^{-1})\frac{\|\delta\bar{\mathbf{k}}_{t-1}\|_2}{\|\bar{\mathbf{k}}_{t-1}\|_2}$ *and Eq. (10) and (11) are numerically stable with single precision arithmetic, where* η_1 *is a positive number depending on the matrix size of* $\mathbf{P}_{t-1}$ *and* ϵ_S *is a machine epsilon for single precision arithmetic.*

Proof sketch: The numerical accuracy of the mixed precision KRLS will be validated based on the error decomposition scheme (e.g., unavoidable error + rounding error). Fig. 4 describes the error decomposition scheme for $\mathbf{p}_t$ computation which is one of the Ⓢ computing components in Algorithm 1. The $\bar{\mathbf{a}}_t$ computed by Eq. (6) can be decomposed as:

$$fl_D(\bar{\mathbf{a}}_t) = \bar{\mathbf{a}}_t + \delta\bar{\mathbf{a}}_t = \bar{\mathbf{a}}_t + \delta\bar{\mathbf{a}}_{tUNA} + \delta\bar{\mathbf{a}}_{tRND} \tag{21}$$

where $fl_D(\bar{\mathbf{a}}_t)$ represents the computed $\bar{\mathbf{a}}_t$ by double precision floating point arithmetic and $\delta\bar{\mathbf{a}}_t$ represents the error consisting of $\delta\bar{\mathbf{a}}_{tUNA} = \bar{\mathbf{K}}_{t-1}^{-1}\delta\bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)$ (i.e., unavoidable error) due to the previous computation error $\delta\bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)$ and $\delta\bar{\mathbf{a}}_{tRND}$ (i.e., rounding error) depending on the level of precision arithmetic used. Next, we compute $\mathbf{p}_t$ with single precision arithmetic:

$$\begin{aligned} fl_S(\mathbf{p}_t) &= fl_S(\mathbf{P}_{t-1}(\bar{\mathbf{a}}_t + \delta\bar{\mathbf{a}}_t)) \\ &= \mathbf{P}_{t-1}\bar{\mathbf{a}}_t + \mathbf{P}_{t-1}\delta\bar{\mathbf{a}}_t + \delta\mathbf{p}_{tRND} \\ &= \mathbf{p}_t + \delta\mathbf{p}_{tUNA} + \delta\mathbf{p}_{tRND} \end{aligned} \tag{22}$$

where $fl_S(\mathbf{p}_t)$ represents the computed result $\mathbf{p}_t$ by single precision floating point arithmetic and $\delta\mathbf{p}_{tUNA}(= \mathbf{P}_{t-1}\delta\bar{\mathbf{a}}_t)$ represents unavoidable error due to the previous computation error, and $\delta\mathbf{p}_{tRND}$ represents the rounding error. Since $\delta\mathbf{p}_{tUNA}$ is unavoidable regardless of the level of arithmetic precision, it will be desirable to provide the least sufficient

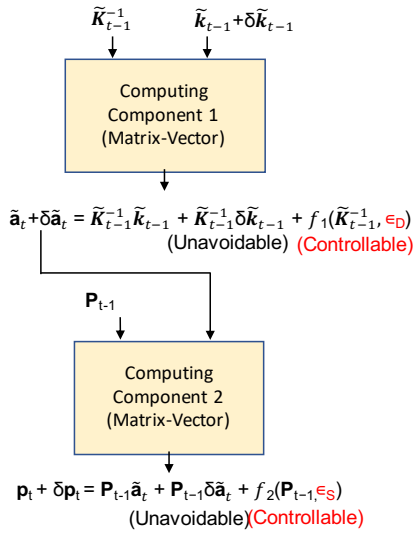

Fig. 4. Rounding Errors Propagation in KRLS

arithmetic precision that keeps the rounding error smaller than the unavoidable error, so that the impact of the rounding error on the accuracy of $\mathbf{p}_t$ can be negligible in practice. Therefore, we seek the numeric condition to satisfy $MIN(\|\delta \mathbf{p}_{tUNA}\|_2) \gtrsim MAX(\|\delta \mathbf{p}_{tRND}\|_2)$ for the first step to validate MP1. Representing $\|\delta \mathbf{p}_{tUNA}\|_2$ with singular values yields [23]:

$$\sigma_{min}(\mathbf{P}_{t-1})\|\delta \bar{\mathbf{a}}_t\|_2 \leq \|\delta \mathbf{p}_{tUNA}\|_2 \leq \sigma_{max}(\mathbf{P}_{t-1})\|\delta \bar{\mathbf{a}}_t\|_2 \tag{23}$$

where $\sigma_{max}(\mathbf{P}_{t-1})$ and $\sigma_{min}(\mathbf{P}_{t-1})$ represent the maximum and the minimum singular value each for $\mathbf{P}_{t-1}$. The $\|\delta \mathbf{p}_{tRND}\|_2$ is also bounded [24]:

$$\begin{aligned} \|\delta \mathbf{p}_{tRND}\|_2 &\leq \eta_1 \epsilon_S \|\mathbf{P}_{t-1}(\bar{\mathbf{a}}_t + \delta \bar{\mathbf{a}}_t)\|_2 \\ &\lesssim \eta_1 \epsilon_S \sigma_{max}(\mathbf{P}_{t-1})\|\bar{\mathbf{a}}_t\|_2 \end{aligned} \tag{24}$$

where η_1 is a positive value mainly depending on the size of the matrix $\mathbf{P}_{t-1}$. Therefore, the condition $MIN(\|\delta \mathbf{p}_{tUNA}\|_2) \gtrsim MAX(\|\delta \mathbf{p}_{tRND}\|_2)$ should satisfy:

$$\sigma_{min}(\mathbf{P}_{t-1})\|\delta \bar{\mathbf{a}}_t\|_2 \gtrsim \eta_1 \epsilon_S \sigma_{max}(\mathbf{P}_{t-1})\|\bar{\mathbf{a}}_t\|_2 \tag{25}$$

Eq. (25) yields:

$$\begin{aligned} \frac{\|\delta \bar{\mathbf{a}}_t\|_2}{\|\bar{\mathbf{a}}_t\|_2} &\gtrsim \eta_1 \epsilon_S \sigma_{max}(\mathbf{P}_{t-1})/\sigma_{min}(\mathbf{P}_{t-1}) \\ &= \eta_1 \epsilon_S Cond_2(\mathbf{P}_{t-1}) \end{aligned} \tag{26}$$

From Eq. (20), the unavoidable relative error $\|\delta \bar{\mathbf{a}}_{tUNA}\|_2/\|\bar{\mathbf{a}}_t\|_2$ is in proportion to $Cond_2(\bar{\mathbf{K}}_{t-1}^{-1})$ as follows [23]:

$$\frac{\|\delta \bar{\mathbf{a}}_{tUNA}\|_2}{\|\bar{\mathbf{a}}_t\|_2} \leq Cond_2(\bar{\mathbf{K}}_{t-1}^{-1}) \frac{\|\delta \bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)\|}{\|\bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)\|} \tag{27}$$

Since $\|\delta \bar{\mathbf{a}}_{tUNA}\|_2 >> \|\delta \bar{\mathbf{a}}_{tRND}\|_2$ in practice with double precision arithmetic, Eq. (27) can be represented:

$$\frac{\|\delta \bar{\mathbf{a}}_t\|_2}{\|\bar{\mathbf{a}}_t\|_2} \lesssim Cond_2(\bar{\mathbf{K}}_{t-1}^{-1}) \frac{\|\delta \bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)\|}{\|\bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)\|} \tag{28}$$

Combining it to Eq. (26),

$$\eta_1 \epsilon_S Cond_2(\mathbf{P}_{t-1}) \lesssim \frac{\|\delta \bar{\mathbf{a}}_t\|_2}{\|\bar{\mathbf{a}}_t\|_2} \lesssim Cond_2(\bar{\mathbf{K}}_{t-1}^{-1}) \frac{\|\delta \bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)\|}{\|\bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)\|} \tag{29}$$

Therefore,

$$\eta_1 \epsilon_S Cond_2(\mathbf{P}_{t-1}) \lesssim Cond_2(\bar{\mathbf{K}}_{t-1}^{-1}) \frac{\|\delta \bar{\mathbf{k}}_{t-1}\|_2}{\|\bar{\mathbf{k}}_{t-1}\|_2} \tag{30}$$

Next, we explain that finite precision arithmetic for the two other Ⓢ computations (i.e., $\mathbf{q}_t$ and $\mathbf{P}_t$) is numerically stable. The $\bar{\mathbf{a}}_t^T \mathbf{p}_t$ in $\mathbf{q}_t$ performs the dot product of the two identical vectors which is numerically stable [25]:

$$\bar{\mathbf{a}}_t^T \mathbf{p}_t = \bar{\mathbf{a}}_t^T \mathbf{A}_{t-1}^T \mathbf{A}_{t-1} \bar{\mathbf{a}}_t = (\mathbf{A}_{t-1}\bar{\mathbf{a}}_t)^T (\mathbf{A}_{t-1}\bar{\mathbf{a}}_t) \tag{31}$$

Therefore, computation for $\mathbf{q}_t$ is numerically stable, since the remaining basic operations such as the addition and the division (e.g., no division by zero) are numerically stable. The relative error for each element in the pairwise multiplications in the outer product in $\mathbf{P}_t$ is bounded by a machine epsilon. Likewise, the pairwise subtractions are numerically stable since cancellation errors for all the elements in $\mathbf{P}_t$ are not expected in practice. Therefore, if the condition of (30) holds, and the rounding errors from the two other numerically stable Ⓢ components are not large enough to affect the accuracy with single precision arithmetic, the accuracy of mixed precision KRLS is equivalent to double precision KRLS. □

To link the error decomposition scheme in Fig. 4 with the bias-variance trade off [26], [27], an individual computing component can be mapped to the entire computing component implementing a KRLS algorithm. This mapping decomposes the computed prediction value into the prediction error generated by exact arithmetic, "Unavoidable" error, and "Controllable" error. The unavoidable error comes from exact arithmetic for prediction function $f(\cdot)$ on input noise; i.e., $f(\delta \mathbf{x}_t)$ if the input $\mathbf{x}_t$ corrupted by the noise $\delta \mathbf{x}_t$. The controllable error, δy, decreases when higher precision arithmetic is applied. KRLS can be considered as a linear regression between a kernel vector and the weights based on Eq. (7). In the bias-variance decomposition for a linear regression that constructs a function $f(\mathbf{x}_t)$ based on a training set D, the expected mean squared error over the dataset D on an out-sample $\mathbf{x}_t$ ($\rightarrow \bar{\mathbf{k}}_{t-1}(\mathbf{x}_t)$ in KRLS) can be represented [26], [27]:

$$\begin{aligned} &\mathbb{E}_{\mathbf{D}}[(f(\mathbf{x}_t; D) - \mathbb{E}[y|\mathbf{x}_t])^2] \\ &= (\mathbb{E}_{\mathbf{D}}[f(\mathbf{x}_t; D)] - \mathbb{E}[y|\mathbf{x}_t])^2 \ (bias^2) \\ &+ \mathbb{E}_{\mathbf{D}}[(f(\mathbf{x}_t; D) - \mathbb{E}_{\mathbf{D}}[f(\mathbf{x}_t; D)])^2] \ (variance) \\ &+ \mathbb{E}[f(\delta \mathbf{x}_t)^2|\mathbf{x}_t] \ (noise^2_{data}) \\ &+ \mathbb{E}[\delta y^2|\mathbf{x}_t] \ (noise^2_{rnd}) \end{aligned} \tag{32}$$

where $f(\mathbf{x}_t; D)$ is an estimation on $\mathbf{x}_t$ based on a data set D, $\mathbb{E}[y|\mathbf{x}_t]$ is the optimal regression, $\mathbb{E}[f(\delta \mathbf{x}_t)^2|\mathbf{x}_t]$ is an expectation operator over $f(\delta \mathbf{x}_t)^2$, and $\mathbb{E}_{\mathbf{D}}$ is an expectation operator over possible Ds with a fixed number of training samples. Notice that Eq. (32) assumes that $f(\delta \mathbf{x}_t)$ follows white Gaussian noise, and the targets used for training are not

corrupted by noise. If a higher regularization is employed, the error from the bias increases and the error from the variance decreases. Therefore, a regularization can be chosen by a model section process to minimise the summation of the two terms of the bias and the variance. In KRLS, a higher regularization is imposed by a wider kernel width and a higher ν when the number of training samples decreases in order to minimise the variance part. In contrast, if model selection uses the increased training samples, the kernel width chosen by model selection shrinks since a reduced kernel width increases the number of weight parameters and decreases the bias without increasing the variance by the increased training samples. In this case, it is highly possible that a narrower kernel width due to the increased training samples increases the dictionary size, resulting in a higher improved throughput ratio by mixed precision KRLS. The bias-variance trade-off has been discussed with finite training samples, noise in the inputs, and under-modeling in many works of literature [26], [27], but to our best knowledge, no literature discusses the bias-variance trade off with the rounding error. Out of the two rounding error quantities ("Unavoidable" and "Controllable"), we can only control the error quantity indicated by the "Controllable" part (i.e., $noise^2_{rnd}$ in Eq. (32)) by adjusting arithmetic precision, and the unavoidable error corresponds to the noise part (i.e., $noise^2_{data}$ in Eq. (32)), since the unavoidable error comes from the noise in the input samples. Our mixed precision KRLS does not increase the error of $noise^2_{rnd}$ compared to double precision KRLS while pure single precision KRLS increases the error of $noise^2_{rnd}$. Therefore, a cross-validation suggests a higher β and (or) a higher ν for a pure single precision KRLS compared to double precision KRLS in order to decrease the errors from $noise^2_{rnd}$ and $variance$; $variance$ can be affected since a hyperparameter set chosen from model selection process depends on arithmetic precision.

The Mixed Precision Property 2 (MP2) describes the achievable throughput gain by the mixed precision KRLS.

Mixed Precision Property 2. *Throughput Gain by Mixed Precision KRLS*
The maximum throughput gain is $3\theta/(\theta+2)$, where θ is a speedup gain factor by employing a reduced arithmetic precision. For example, if single precision arithmetic is twice faster than double precision arithmetic, $\theta = 2$.

Proof sketch: We consider ALD pass only for achievable throughput gain, since the frequency of ALD pass is $\sim 100\%$ with sufficient training samples [4]. In Algorithm 1, the total number of operations required for KRLS on ALD pass is: $2m_t^2$ (Eq. (6)) + $2m_t^2$ (Eq. (9)) + m_t^2 (Eq. (10)) + m_t^2 (Eq. (11)). Therefore, $6m_t^2$ operations are required per training. Notice that we ignore $O(m_t)$ computation. Using a run time model of $T(\epsilon_{double}) = \theta \times T(\epsilon_{single})$ where $T(\epsilon_{double})$ is an average time cost for a double precision arithmetic operation and $T(\epsilon_{single})$ for a single precision arithmetic operation, the training throughput gain by the mixed precision KRLS, $\mathcal{G}_{mixed}$, is:
$\mathcal{G}_{mixed} = 6m_t^2 T(\epsilon_{double})/(2m_t^2 T(\epsilon_{double}) + 4m_t^2 T(\epsilon_{single}))$
$= 3\theta/(\theta+2)$. □

For example, if $\theta = 2$, a $1.5\times$ throughput improvement is achievable by mixed precision KRLS. Even though a typecasting operation for mixed precision KRLS incurs computation overhead, the overhead is negligible in practice since only two typecasting operations for $O(m_t)$ storage are required per training: from single to double for $\mathbf{q}_t$ and from double to single for $\bar{\mathbf{a}}_t$ in Algorithm 1.

The mixed precision KRLS also brings the memory footprint savings by using single precision data for $\mathbf{P}_t$ as described in Mixed Precision Property 3 (MP3).

Mixed Precision Property 3. *Storage Saving*
Assuming that the required dictionary size is the same as double precision KRLS, the mixed precision arithmetic technique saves up to 25% of memory resource compared to double precision KRLS.

Proof sketch: In the proof, we only consider $O(m_t^2)$ for required storage. The double precision KRLS requires double precision storage for $l_{in} \times m_t$ for $\mathcal{D}_t$ and m_t^2 for $\bar{\mathbf{K}}_t^{-1}$ and $\mathbf{P}_t$, while the mixed precision KRLS utilises single precision data for $\mathbf{P}_t$. Therefore, the storage saving by the mixed precision KRLS is as follows:

$$\begin{aligned} &1 - \textit{Bytes required for mixed KRLS / Bytes for double KRLS} \\ &= 1 - (64(m_t^2 + l_{in} \times m_t) + 32m_t^2)/(64(2m_t^2 + l_{in} \times m_t)) \\ &= 1 - (3m_t + 2l_{in})/(4m_t + 2l_{in}) \end{aligned} \tag{33}$$

In case that $l_{in} << m_t$, the storage saved by mixed precision KRLS approaches 25%. □

V. Experimental Evaluation

We evaluate our mixed precision KRLS in terms of the prediction accuracy, the training throughput, and the memory usage for a 3D non-linear regression, a Lorenz time series, an MG30 time series, a sunspot number time series, and Nino 3.4 sea surface temperature time series. The prediction accuracy evaluation employs the log_{10}-based Root Mean Square Error (RMSE) learning curves, the error bars for the throughput data are generated from 10 experiments, and the improved throughput is measured by the throughput from mixed precision KRLS divided by the throughput from double precision KRLS. We utilize computing resources and computational problems as follows:
- CPU: Intel Xeon E2650 core having 2 sockets and 8 cores per socket. One of the cores is used for the measurement.
- DRAM: 64GB
- Software: GCC 4.8.5, Intel Math Kernel Library (MKL) 2017
- Training and Testing: KRLS learns the weights $\bar{\boldsymbol{\alpha}}_t$ and the dictionary $\mathcal{D}_t$ according to training samples. After a training pass, KRLS returns the trained $\bar{\boldsymbol{\alpha}}_t$ and $\mathcal{D}_t$, and they are used to test the prediction accuracy using test samples which are not used during training.

A. Cross-Validations

The hyperparameters have been chosen by 10-fold cross-validation using double precision arithmetic KRLS for the non-linear regression and by Monte Carlos Cross-Validation

(MCCV) [28] for other time series applications. The hyperparameter sets used for the cross-validations are described in Table I.

1) Nonlinear Regression: We employ a 10-fold cross-validation with the $10,000$ data samples. The targets from the training samples have been perturbed with Gaussian noise with the standard deviation error 0.1.

2) Time Series Applications: We employ the Monte Carlo cross-validation (MCCV) used in [28] for hyperparameter search for our time series applications including a Lorenz time series, a MG30 time series, a sunspot number time series, and a Nino 3.4 sea surface temperature time series to consider the temporal dependency of the data. The MCCV generates 20 random training/validation sets per hyperparameter set for all time series applications. The MCCV utilises $10,000$ data samples with a setting using $1,000$ initial training data samples, $1,000$ validation samples, and $8,000$ available random data samples for a MG30 time series and a Lorenz time series, $2,880$ data samples with a setting using $1,440$ initial training data samples, 288 validation samples, and $1,152$ available random data samples for the sunspot time series, and $1,440$ data samples with a setting using 720 initial training data samples, 240 validation samples, and 480 available random data samples for the Nino 3.4 sea surface temperature time series.

We chose the number of the validation samples as 10% of total data samples except the Nino 3.4 application so that sufficient data samples could be used for the validation samples for all cases to minimise the variance of the optimal parameters chosen from each MCCV; using 144 validation samples (i.e., 10% of the data samples) for the Nino 3.4 generated high variances of the optimal parameters chosen from each MCCV run out of 5 runs (e.g., the ranges of the optimal β, ν, and l_{in} in Table I varied between 6 and 10, 10^{-5} and 10^{-1}, and 7 and 10 respectively.) and increasing the validation samples to 240 let the variance minimised (e.g, the ranges of the optimal β, ν, and l_{in} varied between 6 and 10, 10^{-5} and 10^{-3}, and 8 and 9 respectively.). The average number of the training samples in the MCCV can be estimated as (the number of the initial training samples)$+0.5\times$(the number of available random data samples). In order to maintain sufficient training samples required to achieve a desired level of performance, we utilised 50% of the data samples for the initial training samples for the applications with the MCCV data samples less than $3,000$ and 10% of the data samples for the initial training samples for the applications with the MCCV data samples equal to $10,000$. An appropriate time embedding dimension should be carefully chosen according to an application; an excessively high dimension leads to massive computation, while an excessively low dimension leads to wrong computational results. In order to choose an appropriate time embedding dimension (i.e., minimum sufficient dimension that has the equivalent accuracy performance compared to a higher embedding dimension), we increase the time embedding dimension in the MCCV hyperparameter sets until a lower embedding dimension produces lower MSEs compared to a higher embedding dimension with βs and νs chosen by MCCV, and we choose the lower dimension for a time embedding dimension for the time series applications. This method is similar to the work of [29] that extends the embedding dimension by one until no false neighbors are detected to seek the minimum sufficient dimension.

3) Cross-Validation Outcomes: Table II describes the cross-validation outcomes using double precision arithmetic for the four applications. Our mixed precision KRLS is validated by the identical hyperparameter outcomes to double precision KRLS with the equivalent MSEs. For single precision KRLS algorithms, our cross-validations suggest higher νs with higher MSEs compared to double precision KRLS for the five applications. A pure single precision arithmetic can be used for KRLS with a hyperparameter set chosen by the cross-validation using single precision KRLS, but it will be another research topic to investigate the trade-off between its accuracy loss and its throughput improvement according to the level of arithmetic precision.

TABLE I
HYPERPARAMETER SETS FOR CROSS-VALIDATION

Application	Params	1^{st}	2^{nd}	3^{rd}	4^{th}
Nonlinear	β	3	4	5	6
	ν	10^{-5}	10^{-4}	10^{-3}	10^{-2}
Lorenz	β	1	2	3	4
	ν	10^{-4}	10^{-3}	10^{-2}	10^{-1}
	l_{in}	3	4	5	6
MG30	β	$\frac{1}{2\sqrt{2}}$	$\frac{1}{2}$	$\frac{1}{\sqrt{2}}$	1
	ν	10^{-4}	10^{-3}	10^{-2}	10^{-1}
	l_{in}	5	6	7	8
Sunspot	β	80	90	100	110
	ν	10^{-5}	10^{-4}	10^{-3}	10^{-2}
	l_{in}	4	5	6	7
Sea Temp	β	4	6	8	10
	ν	10^{-7}	10^{-5}	10^{-3}	10^{-1}
	l_{in}	7	8	9	10

TABLE II
CROSS-VALIDATION RESULTS

Application	β	ν	l_{in}	m	MSE
Nonlinear	4	10^{-4}	3	740	1.05×10^{-3}
Lorenz	3	10^{-3}	5	242	0.17
MG30	0.5	10^{-4}	7	753	2.63×10^{-5}
Sunspot	90	10^{-4}	6	129	3.09
Sea Temp	10	10^{-5}	9	131	0.11

B. Nonlinear Regression

KRLS learns the three-dimensional function $y = \sin(x_1) + x_2/10 + \cos(x_3)$ with uniformly distributed x_1, x_2, and x_3 with a range of $[-10, 10]$. Up to $100,000$ samples are used for training and unused $1,000$ samples are used for the testing. Fig. 5 represents the targets (i.e., ys) from the test samples.

1) Prediction Accuracy: Fig. 6 describes the learning curves for a chosen KRLS model by cross-validation when it employs single, double and mixed precision arithmetic respectively. The mixed precision KRLS achieves identical prediction accuracy to the double precision KRLS upto the most significant six decimal digits in terms of $log_{10}(RMSEs)$

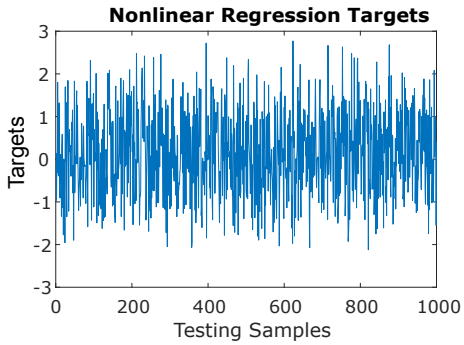

Fig. 5. Nonlinear Regression Targets from Test Samples

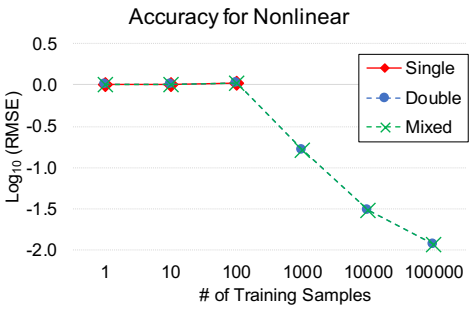

Fig. 6. Accuracy according to Level of Precision Arithmetic for Nonlinear

at the six data points in Fig. 6. Single precision arithmetic invalidated KRLS learning property when the number of training samples reaches $1,000$. The accuracy from single precision arithmetic were reported as '-NaN"s at the data points over 100. The single precision arithmetic broke the mathematical property of the positiveness of γ_t when the number of the samples was over 100 due to magnified rounding errors as stated in Corollary 2.2. The saturated condition numbers are $Cond_2(\bar{\mathbf{K}}_t^{-1}) = 10^{8.2}$ and $Cond_2(\mathbf{P}_t) = 10^{4.1}$ when the number of training samples reach $100,000$.

2) Training Throughput and Storage Usage: The improved throughputs with standard deviation error bars are shown in Fig. 7. The mixed precision KRLS improves the training throughput by $1.32\times$ when the number of training samples is $100,000$. The number of the support vectors is 775 for both double and mixed precision KRLS when the number of training samples is $100,000$. The average throughput of double precision KRLS was 854 training samples/sec when the number of training samples is $100,000$. In [4], when the number of the training samples was $50,000$ to predict a 2D Sinc-Linear function, the number of the support vector was 72. Its runtime was reported as around 20 secs, implying that the average throughput was $2,500$ samples/sec with the number of the support vectors $10.8\times$ less than our case. Based on Eq. (33), the mixed precision KRLS brings 24.95% storage saving when the number of training samples is $100,000$, compared to double precision KRLS.

C. Lorenz Time Series

Lorenz is a well known chaotic time series describing dynamical atmosphere behavior that is closely associated

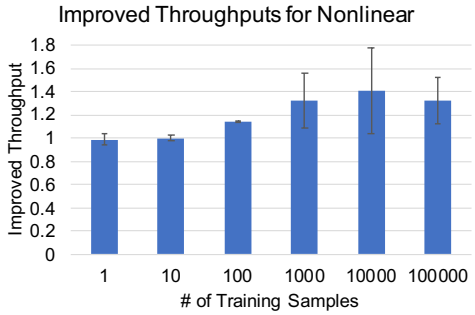

Fig. 7. Improved Throughputs for Nonlinear

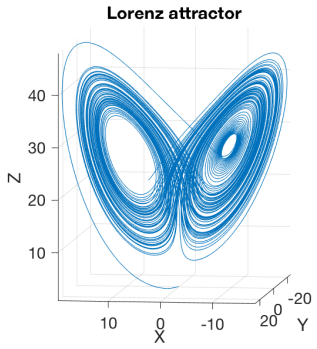

Fig. 8. Lorenz Attractor Plot

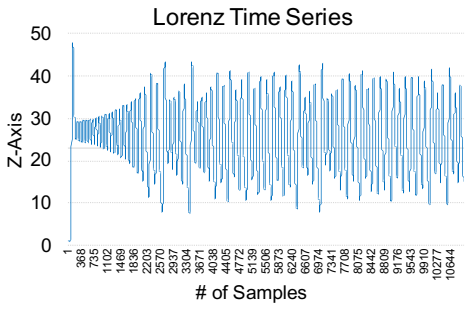

Fig. 9. Lorenz Time Series (Z-Axis)

with "butterfly effect" [30]. The Lorenz time series data are generated by the following equations [31]:

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y-x) \\ \frac{dy}{dt} &= x(\rho-z)-y \\ \frac{dz}{dt} &= xy-\lambda z \end{aligned} \tag{34}$$

where x, y, and z are correlated variables, while σ, ρ, and λ are parameters needed to be set. We employ the same parameters as [31]: $\sigma = 10$, $\rho = 28$, and $\lambda = \frac{8}{3}$. We set $x = 0$, $y = 1.0$, and $z = 1.05$ for the initial position. We generate the data and a Lorenz attractor plot with functions provided by [32]. The Lorentz attractor plot is shown in Fig. 8, and we predict the data projected onto the z axis as shown in Fig. 9.

We perform 5 step ahead predictions with KRLS; KRLS makes a prediction for z_{10} using the information of

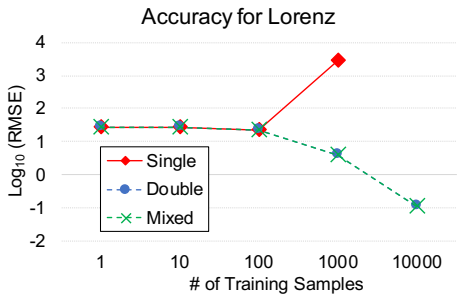

Fig. 10. Accuracy according to Level of Precision Arithmetic for Lorenz

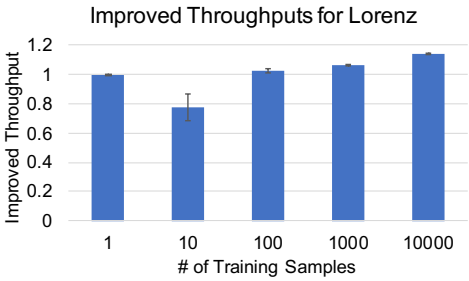

Fig. 11. Improved Throughputs for Lorenz

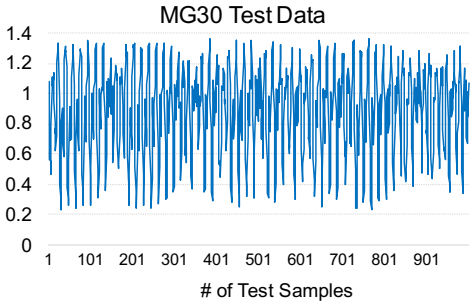

Fig. 12. MG30 Test Samples

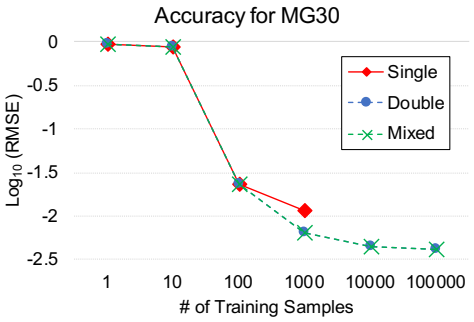

Fig. 13. Accuracy according to Level of Precision Arithmetic for MG30

$[z_5, z_4, z_3, z_2, z_1]$ when z_5 is a current sample. Earlier $10,000$ samples in Fig. 9 are used for the training and the other unused $1,000$ samples are used for the testing. The mean of the total Lorenz z data is 25.42 and the variance is 77.44.

1) Prediction Accuracy: Fig. 10 describes the learning curves for a chosen KRLS model by cross-validation when it employs single, double, and mixed precision arithmetic respectively. The mixed precision arithmetic achieves the identical prediction accuracy to the double precision KRLS upto the most significant 6 decimal digits in terms of $log_{10}(RMSEs)$ at the five data points in Fig. 10. The prediction accuracy from single precision arithmetic has been significantly degraded when the number of training samples is over 100; -NaN was reported when the number of samples is $10,000$. The saturated condition numbers are $Cond_2(\bar{\mathbf{K}}_t^{-1}) = 10^{8.8}$ and $Cond_2(\mathbf{P}_t) = 10^{5.6}$ when the number of training samples reach 10000.

2) Training Throughput and Storage Usage: The improved throughputs are shown with standard deviation error bars in Fig. 11. The mixed precision KRLS improves the training throughput by $1.15\times$ when the number of training samples is $10,000$. The number of the support vectors in the dictionary is 242 when the number of training samples is $10,000$. The average throughput of double precision KRLS was $6,577$ training samples/sec when the number of training samples is $10,000$. Based on Eq. (33), our mixed precision KRLS brings 24.74% storage saving when the number of training samples is $10,000$, compared to double precision KRLS.

D. Mackey-Glass Time Series

Mackey-Glass (MG) is a chaotic time series describing dynamical behaviors of physiological control systems and is widely used for a time series benchmark [33]. The MG30 data are generated by the MG equation:

$$\frac{dx_t}{dt} = -bx_t + \frac{ax_{t-\tau}}{(1 + x_{t-\tau}^{10})} \tag{35}$$

where $a = 0.2, b = 0.1$ and $\tau = 30$. The data are generated with the discrete time step size '1' and the initial value $x_1 = 1.2$. We sample the data every 6 time steps for the training and the test data, so that the KRLS can be used to predict a target 6 steps ahead. Based on the outcomes of cross-validation, we employ $\beta = \frac{1}{2}$, $\nu = 10^{-4}$ and the time embedding (i.e., input size) = 7. For example, KRLS makes a prediction for x_{37} using the information of $[x_{37}, x_{31}, x_{25}, x_{19}, x_{13}, x_7, x_1]$ when x_{37} is a current sample. Up to $100,000$ samples are used for the training and unused $1,000$ samples are used for the testing. Fig. 12 represents the test samples. The mean of the total MG30 data is 0.89 and the variance is 0.083.

1) Prediction Accuracy: Fig. 13 describes the learning curves for a chosen KRLS model by cross-validation when it employs single, double, and mixed precision arithmetic respectively. The mixed precision arithmetic achieves the identical prediction accuracy to the double precision KRLS upto the most significant six decimal digits in terms of $log_{10}(RMSEs)$ at the six data points in Fig. 13. The prediction accuracy from single precision arithmetic has been significantly degraded when the number of training samples is over $1,000$. The saturated condition numbers are $Cond_2(\bar{\mathbf{K}}_t^{-1}) = 10^{8.2}$ and $Cond_2(\mathbf{P}_t) = 10^{3.7}$ when the number of training samples reach $100,000$.

2) Training Throughput and Storage Usage: The improved throughputs are shown with standard deviation error bars in

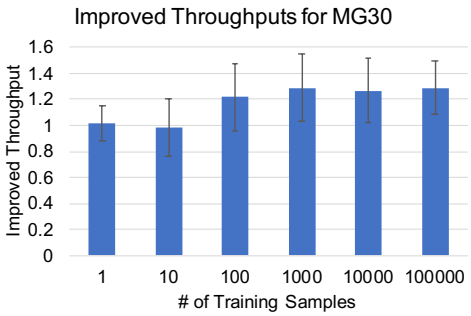

Fig. 14. Improved Throughputs for MG30

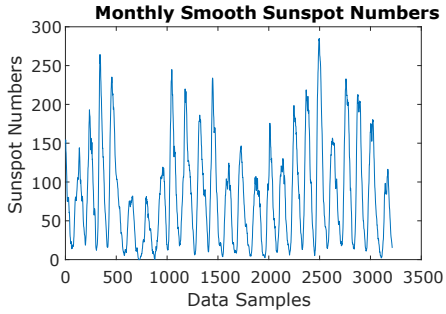

Fig. 15. Monthly Smooth Sunspot Numbers from [36]

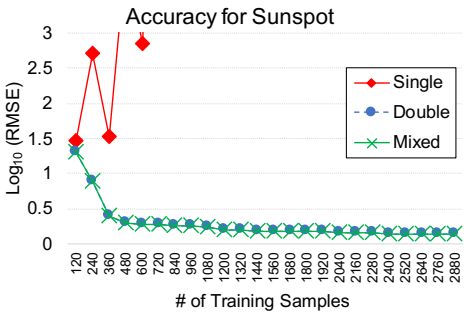

Fig. 16. Accuracy according to Level of Precision Arithmetic for Sunspot

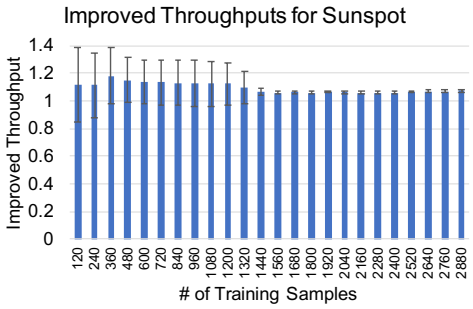

Fig. 17. Improved Throughputs for Sunspot

Fig. 14. The mixed precision KRLS improves the training throughput by $1.29\times$ when the number of training samples is $100,000$. The number of the support vectors in the dictionary is 804 when the number of training samples is $100,000$. The average throughput of double precision KRLS was 795 training samples/sec when the number of training samples is $100,000$. Based on Eq. (33), our mixed precision KRLS brings 24.89% storage saving when the number of training samples is $100,000$, compared to double precision KRLS.

E. Sunspot Number Times Series

The sunspot number also referred as Wolf number represents Sun activity with an integer number [34]. The sunspot number R is calculated with the formula [35]:

$$R = k(10g + s) \tag{36}$$

where s is the number of individual sunspots, g is the number of sunspot groups, and k is a scaling factor depending on observation location, equipment, and etc. We analyze the smooth monthly sunspot number data during the period 1750 to 2017 from [36]. Smoothing data is used for noise reduction [34]. Monthly data from earlier 240 years are used for training, and the remaining 28 years data are used for testing (i.e., $2,880$ training samples and 336 test samples.). KRLS performs one step prediction for the sunspot number in the next month. Fig. 15 shows the smooth monthly sunspot data. The mean is 82.4, and the variance is $10^{3.6}$.

1) Prediction Accuracy: Fig. 16 describes the learning curves for a chosen KRLS model by cross-validation when it employs single, double, and mixed precision arithmetic respectively. The mixed precision KRLS achieves the identical prediction accuracy to the double precision KRLS upto the most significant 4 decimal digits in terms of $log_{10}(RMSEs)$. The $log_{10}(RMSE)$ is 0.13 when the number of training samples is $2,880$. The prediction accuracy with pure single precision arithmetic has been significantly degraded when the number of training samples is higher than 120; $-NaN$s were reported for the accuracy beyond 600 training sample points. The single precision arithmetic broke the mathematical property of the positiveness of γ_t due to magnified rounding errors as stated in Corollary 2.2. The saturated condition numbers are $Cond_2(\bar{\mathbf{K}}_t^{-1}) = 10^{8.4}$ and $Cond_2(\mathbf{P}_t) = 10^{3.7}$ when the number of training samples reach $2,880$.

2) Training Throughput and Storage Usage: The improved throughput with standard deviation error bars are shown in Fig. 17. The mixed precision KRLS improves the training throughput by $1.09\times$ on average. The average throughput improvement was less than the nonlinear and the MG30 time series since the number of the support vectors was not large enough to exploit the memory bandwidth improvement by the mixed precision KRLS on the Intel Xeon E2650 architecture. The performance difference between double and mixed precision KRLS depends on the processor implementation and the number of the support vectors. The number of the support vectors in the dictionary was 140 when the number of training samples was $2,880$. The average throughput of double precision KRLS is $24,075$ training samples/sec when the number of training samples is $2,880$. Based on Eq. (33), our mixed precision KRLS brings 24.48% storage saving when the number of training samples is $2,880$, compared to double precision KRLS.

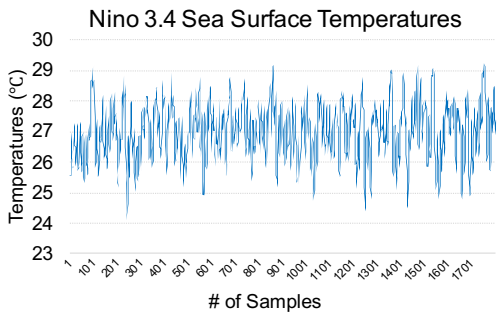

Fig. 18. Monthly Sea Surface Temperature Data from [38]

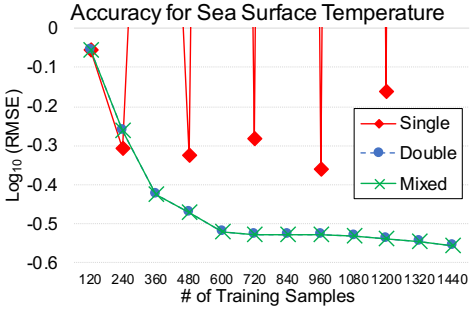

Fig. 19. Accuracy according to Level of Precision Arithmetic for Sea Surface Temperature

F. Nino 3.4 Sea Surface Temperature Time Series

The sea surface temperature is an essential parameter to predict global weather patterns, providing scientists with important information on the global climate system [37]. The sea surface temperatures prediction has wide range of applications such as climate and seasonal forecasting, validation of atmospheric models, sea turtle tracking, evaluation of coral bleaching, tourism, and commercial fisheries management. The Nino 3.4 represents an ocean region locating at 5N-5S and 170W-120W [38]. We analyze the monthly Nino 3.4 sea surface temperature data during the period from 1870 to 2019 downloaded from [38]. Monthly data from earlier 120 years are used for training and the remaining 30 years data are used for testing (i.e., $1,440$ training samples and 360 test samples.). KRLS performs one step prediction for the sea surface temperature in the next month. Fig. 18 shows the smooth monthly sunspot data. The mean is 26.96 and the variance is 0.79.

1) Prediction Accuracy: Fig. 19 describes the learning curves for a KRLS model chosen by MCCV employing single, double, and mixed precision arithmetic respectively. The mixed precision KRLS achieves the identical prediction accuracy to the double precision KRLS upto the most significant six decimal digits in terms of $log_{10}(RMSEs)$. The $log_{10}(RMSE)$ is -0.56 when the number of training samples is $1,440$. The prediction accuracy with pure single precision arithmetic has been significantly degraded when the number of training samples is higher than 240; the maximum $log_{10}(RMSE)$ was 12.42 at $1,320$ training sample points. The saturated condition numbers are $Cond_2(\bar{\mathbf{K}}_t^{-1}) = 10^{9.6}$ and $Cond_2(\mathbf{P}_t) = 10^{3.4}$ when the number of training samples reach $1,440$.

2) Training Throughput and Storage Usage: The improved throughput with standard deviation error bars are shown in Fig. 20. The mixed precision KRLS improves the training throughput by $1.08\times$ on average. The number of the support vectors in the dictionary was 136 when the number of training samples was $1,440$. The average throughput of double precision KRLS is $14,828$ training samples/sec when the number of training samples is $1,440$. Based on Eq. (33), our mixed precision KRLS brings 24.20% storage saving when the number of training samples is $1,440$, compared to double precision KRLS.

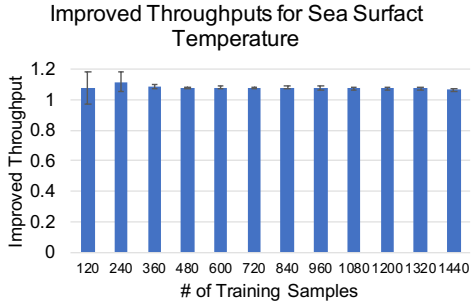

Fig. 20. Improved Throughputs for Sea Surface Temperature

G. Discussion

With practically infinite training samples for online learning applications, KRLS is computationally feasible with a guaranteed prediction accuracy, which is proven in [4]. Our mixed precision KRLS produces the equivalent prediction accuracy to double precision KRLS since the cross-validations let the condition of MP1 satisfied. For example, the cross-validations choose $\nu = 10^{-5}$ to 10^{-4} for double precision KRLS, leading to the $Cond_2(\bar{\mathbf{K}}_t^{-1}) = 10^{8.2}$ to $10^{9.6}$, while $Cond_2(\mathbf{P}_t) \approx 10^{3.4}$ to $10^{3.7}$ in our experiments. If arithmetic precision much lower than single precision was used for the Ⓢ parts in Algorithm 1, the prediction accuracy would be degraded. The β affects the training throughput since a lower β implies a larger size of the kernel matrix, but β does not have a direct relation to the condition number of the kernel matrix. Our key findings from the experiments include:

- The prediction accuracy of the mixed precision KRLS is equivalent to the double precision KRLS, while the pure single precision KRLS was not able to produce equivalent prediction accuracy to double precision KRLS.
- The throughput gains by mixed precision KRLS are 1.32, 1.15, 1.29, 1.09, and $1.08\times$ for a 3D nonlinear regression, a Lorenz time series, a MG30 time series, a sunspot number time series, and a Nino3.4 sea surface temperature time series applications respectively. The throughput gain factor depends on the number of the support vectors stored in the dictionary and a computing platform.
- 24-25% of memory footprints can be saved by mixed precision KRLS for the five applications.

VI. Related Work

A. KRLS variants

Many KRLS variants were developed with different purposes. In [5], the Sliding Window KRLS (SW-KRLS) was proposed to improve the prediction accuracy for time variation systems. On every training sample, this algorithm stores a new sample into the dictionary and removes the most outdated sample to keep the dictionary size fixed. Therefore, it might lose some important information from past, compared to the KRLS of [4]. In [3], the Fixed Memory Budget KRLS (FB-KRLS) is similar to the sliding windows KRLS, but it discards the least significant sample out of the dictionary rather than the most outdated sample every time step. In [9], the Quantized KRLS (QKRLS) reduces the computational cost by mapping the inputs into a smaller quantized space having finite states. For example, if Euclidean distance between any of the support vectors and a new input vector is larger than a threshold, the input is added as a new support vector. In [10], KNLMS algorithm was proposed that utilises least mean squares algorithm with kernel method instead of recursive least squares in order to improve the throughput, resulting in the prediction accuracy loss compared to the KRLS of [4]. In [7], Adaptive Normalised Sparse QKRLS (ANS-QKRLS) was proposed to support fast and accurate prediction for multivariate time series applications. The ANS-QKRLS is a combined algorithm employing QKRLS, KNLMS, and original KRLS. In [8], the KRLS algorithm was amended to detect network anomalies. In [39], the input signal $\mathbf{x}_t$ was decomposed on frequency domain by Fourier analysis and the weights on the frequency domain were updated by a recursive least squares algorithm; this scheme was able to save computational resource and operation cost when the input signal could be well reconstructed by small number of frequency components. There are some research to adapt the KRLS of [4] for dynamic systems. For example, the KRLS Tracker (KRLS-T) that adapts the KRLS for time varying nonlinear regression was proposed in [40], and the Extended KRLS (EX-KRLS) of [6] adapted the KRLS for nonlinear dynamic system identifications by exploiting a linear state space model used in Kalman filter of [41]. Some research efforts utilized FPGAs to improve the training throughputs for online learning algorithms. In [42], the KRLS was implemented on FPGA to support low latency training, and in [43], KNLMS was implemented on FPGA in fully pipelined fashion in order to improve the training throughput.

B. Comparison with other KRLS variants

Although this work focuses on improving the throughput of KRLS using less hardware resource by employing mixed precision arithmetic, it would be interesting to compare empirical evidence of our mixed precision KRLS with other existing KRLS approaches. In [7], many KRLS variant algorithms including KRLS, SW-KRLS, EX-KRLS, FB-KRLS, KRLS-T, QKRLS, and ANS-QKRLS were compared each other with a Lorenz chaotic time series, an El Nino-Southern Oscillation (ENSO) indexes time series, and a yearly sunspots time series data set in terms of prediction accuracy and throughput. Overall, the prediction accuracy and the throughput highly depend on the datasets. For 30 step ahead predictions in Lorenz time series, KRLS shows the highest RMSE prediction accuracy, while ANS-QKRLS shows the highest training throughput. For the 5 step ahead prediction with sunspot time series, ANS-QKRLS shows the highest prediction accuracy, while KRLS shows the highest training throughput. It is not practically feasible to decide the best KRLS algorithm given a dataset as to hardware resource, power, throughput, and prediction accuracy at the same time, since one algorithm can be good for prediction accuracy, requiring more memory foot prints and lower training throughput. In other words, different KRLS variants were developed with different purposes. However, we notice that the number of hyperparameters of KRLS is much less than any other KRLS algorithms, implying higher generalisation prediction ability compared to other KRLS algorithms and reduced efforts required for hyperparameter search (Table II in [7] describes the required numbers of hyperparameters according to the different KRLS algorithms.). The original KRLS also shows equivalent empirical prediction performance with other KRLS variants with various times series applications in [7]. Therefore, we have decided to develop mixed precision arithmetic for the original KRLS. To the best our knowledge, this is the first work to apply mixed precision training for KRLS in order to improve the training throughput using less memory footprints without losing the prediction accuracy compared to double precision KRLS. Our mixed precision method can be applicable to a computing platform on which further computational speed can be achieved by reduced precision arithmetic.

VII. Conclusion

Since online learning algorithms including KRLS require the throughput higher than real-time data arrival rate given a computing resource budget [1], it is crucial for online learning algorithms to have a higher training throughput using less computational resources. This paper presents the mixed precision KRLS producing equivalent prediction accuracy to double precision KRLS with a higher training throughput and a lower memory footprint. The mixed precision KRLS applies lower precision arithmetic to numerically resilient computations, while higher precision arithmetic to other computations. In our experiments, the mixed precision KRLS demonstrates the improved throughputs by 1.32, 1.15, 1.29, 1.09, and 1.08$\times$ using reduced memory footprints by 24.95, 24.74, 24.89, 24.48, and 24.20% without losing prediction accuracy compared to double precision KRLS for a 3D nonlinear regression, a Lorenz time series, a MG30 time series, a sunspot number time series, and a Nino3.4 sea surface temperature time series applications respectively. Applying single precision arithmetic to the entire computation in the double precision KRLS models chosen by the model selection processes let the models fail to learn for the five applications.

Future work includes exploration of methodology to combine this work with deep neural networks and with the KRLS variants such as [3], [5]–[7], [40]. We desire to apply the similar process of analysing, identifying properties and then tuning to neural networks.

ACKNOWLEDGEMENT

This project has received funding by the European Commission Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 798209 (Entrans) and the grant agreement No. 732631 (OPRECOMP).

REFERENCES

[1] L. Bottou and Y. L. Cun, "Large scale online learning," in *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

[2] S. Shalev-Shwartz, "Online learning and online convex optimization," *Found. Trends Mach. Learn.*, vol. 4, no. 2, pp. 107–194, Feb. 2012. [Online]. Available: http://dx.doi.org/10.1561/2200000018

[3] S. Van Vaerenbergh, I. Santamaria, L. Weifeng, and J. C. Principe, "Fixed-budget kernel recursive least-squares," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pp. 1882–1885.

[4] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, Aug 2004.

[5] S. V. Vaerenbergh, J. Via, and I. Santamaria, "A sliding-window kernel rls algorithm and its application to nonlinear channel identification," in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, May 2006.

[6] W. Liu, I. Park, Y. Wang, and J. C. Principe, "Extended kernel recursive least squares algorithm," *IEEE Transactions on Signal Processing*, vol. 57, no. 10, pp. 3801–3814, Oct 2009.

[7] M. Han, S. Zhang, M. Xu, T. Qiu, and N. Wang, "Multivariate chaotic time series online prediction based on improved kernel recursive least squares algorithm," *IEEE Transactions on Cybernetics*, pp. 1–13, 2018.

[8] T. Ahmed, M. Coates, and A. Lakhina, "Multivariate online anomaly detection using kernel recursive least squares," in *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, May 2007, pp. 625–633.

[9] B. Chen, S. Zhao, P. Zhu, and J. C. Príncipe, "Quantized kernel recursive least squares algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 9, pp. 1484–1491, Sept 2013.

[10] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Transactions on Signal Processing*, vol. 57, no. 3, pp. 1058–1067, March 2009.

[11] V. Sze, Y. Chen, J. Emer, A. Suleiman, and Z. Zhang, "Hardware for machine learning: Challenges and opportunities," in *Custom Integrated Circuits Conference (CICC)*. IEEE, 2017, pp. 1–8.

[12] G. L. Steele, Jr. and J.-B. Tristan, "Adding approximate counters," in *Proceedings of the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP '16. New York, NY, USA: ACM, 2016, pp. 15:1–15:12. [Online]. Available: http://doi.acm.org/10.1145/2851141.2851147

[13] N. Corporation, "NVIDIA Tesla V100 GPU architecture," Aug. 2017, wP-08608-001_v1.1.

[14] J. M. Cioffi, "Limited-precision effects in adaptive filtering," *Circuits and Systems, IEEE Transactions on*, vol. 34, no. 7, pp. 821–833, 1987.

[15] G. E. Bottomley and S. T. Alexander, "A novel approach for stabilizing recursive least squares filters," *Signal Processing, IEEE Transactions on*, vol. 39, no. 8, pp. 1770–1779, 1991.

[16] J. Langou *et al.*, "Exploiting the performance of 32 bit floating point arithmetic in obtaining 64 bit accuracy (revisiting iterative refinement for linear systems)," in *SC Conference, Proceedings of the ACM/IEEE*, 2006.

[17] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, pp. 1737–1746. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045118.3045303

[18] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, "Mixed precision training," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=r1gs9JgRZ

[19] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[20] W. Liu, J. C. Principe, and S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*. Wiley Publishing, 2010.

[21] "IEEE standard for binary floating-point arithmetic," *ANSI/IEEE Std 754-1985*, pp. 0–1, 1985.

[22] D. Goldberg, "What every computer scientist should know about floating-point arithmetic," *ACM Comput. Surv.*, vol. 23, no. 1, pp. 5–48, Mar. 1991. [Online]. Available: http://doi.acm.org/10.1145/103162.103163

[23] L. N. Trefethen, *Numerical Linear Algebra*. SIAM, 1998.

[24] M. Jankowski and H. Woźniakowski, "Iterative refinement implies numerical stability," *BIT Numerical Mathematics*, vol. 17, no. 3, 1977.

[25] T. Ogita, S. M. Rump, and S. Oishi, "Accurate sum and dot product," *SIAM Journal on Scientific Computing*, vol. 26, no. 6, pp. 1955–1988, 2005. [Online]. Available: https://doi.org/10.1137/030601818

[26] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning From Data*. AMLBook, 2012.

[27] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Comput.*, vol. 4, no. 1, p. 1–58, Jan. 1992. [Online]. Available: https://doi.org/10.1162/neco.1992.4.1.1

[28] R. Fonseca-Delgado and P. Gomez-Gil, "An assessment of ten-fold and monte carlo cross validations for time series forecasting," in *2013 10th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, 2013, pp. 215–220.

[29] M. B. Kennel, R. Brown, and H. D. I. Abarbanel, "Determining embedding dimension for phase-space reconstruction using a geometrical construction," *Phys. Rev. A*, vol. 45, pp. 3403–3411, Mar 1992. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.45.3403

[30] E. Lorenz, "Predictability: Does the flap of a butterfly's wings in brazil set off a tornado in texas." 1972.

[31] E. N. Lorenz, "Deterministic nonperiodic flow," *Journal of the Atmospheric Sciences*, vol. 20, no. 2, pp. 130–141, 1963. [Online]. Available: https://doi.org/10.1175/1520-0469(1963)020⟨0130:DNF⟩2.0.CO;2

[32] M. Igor, "Lorenz attaractor plot."

[33] M. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, no. 4300, pp. 287–289, 1977. [Online]. Available: http://science.sciencemag.org/content/197/4300/287

[34] K. Jinno, S. Xu, R. Berndtsson, A. Kawamura, and M. Matsumoto, "Prediction of sunspots using reconstructed chaotic system equations," *Journal of Geophysical Research: Space Physics*, vol. 100, no. A8, pp. 14 773–14 781, 1995. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/95JA01167

[35] F. Clette, L. Svalgaard, J. M. Vaquero, and E. W. Cliver, "Revisiting the Sunspot Number. A 400-Year Perspective on the Solar Cycle," *Space Science Reviews*, vol. 186, pp. 35–103, Dec. 2014.

[36] "SILSO data/image, Royal Observatory of Belgium, Brussels," http://www.sidc.be/silso/.

[37] NOAA. (2006) Why do scientists measure sea surface temperature? [Online]. Available: https://oceanservice.noaa.gov/facts/sea-surface-temperature.html

[38] K. Trenberth and N. C. for Atmospheric Research Staff. (2020) The climate data guide: Nino sst indices (nino 1+2, 3, 3.4, 4; oni and tni). [Online]. Available: https://climatedataguide.ucar.edu/climate-data/nino-sst-indices-nino-12-3-34-4-oni-and-tni

[39] Z. Qin, B. Chen, and N. Zheng, "Random fourier feature kernel recursive least squares," in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 2881–2886.

[40] M. Lázaro-Gredilla, S. V. Vaerenbergh, and I. Santamaría, "A bayesian approach to tracking with kernel recursive least-squares," in *2011 IEEE International Workshop on Machine Learning for Signal Processing*, Sept 2011, pp. 1–6.

[41] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 03 1960. [Online]. Available: https://doi.org/10.1115/1.3662552

[42] Y. Pang, S. Wang, Y. Peng, N. J. Fraser, and P. H. W. Leong, "A low latency kernel recursive least squares processor using FPGA technology," in *2013 International Conference on Field-Programmable Technology (FPT)*, Dec 2013, pp. 144–151.

[43] N. J. Fraser, J. Lee, D. J. M. Moss, J. Faraone, S. Tridgell, C. T. Jin, and P. H. W. Leong, "FPGA implementations of kernel normalised least mean squares processors," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 10, no. 4, pp. 26:1–26:20, Dec. 2017.