Sensors & Transducers
Published by IFSA Publishing, S. L., 2019
http://www.sensorsportal.com

Learning Binary Data Representation for Optical Processing Units

[1, *] Bogdan Kozyrskiy, [1] Maurizio Filippone, [2]Iacopo Poli, [2,3]Ruben Ohana, [2]Laurent Daudet, [2]Igor Carron

[1] Department of Data Science, EURECOM, 450 Route des Chappes, 06410 Biot, France
[2] LightOn, 2 rue de la Bourse, F-75002 Paris, France
[3] Laboratoire de Physique, Ecole Normale Supérieure, 24 rue Lhomond, 75005 Paris, France
[1] Tel.: +33 4 93 00 81 00
[*] E-mail: Bogdan.Kozyrskiy@eurecom.fr

Received: Accepted: Published:

Abstract: Optical Processing Units (OPUs) are computing devices that perform random projections of input data by exploiting the physical phenomenon of scattering a light source through a diffusive medium. Random projections calculated by OPUs have been used successfully for approximating kernel ridge regression for large datasets with low power consumption and at high speed. However, OPUs require the input data to be binary. In this paper, we propose to use shallow and deep neural networks (NN) as binary encoders to perform input data binarization. The difficulty in developing a binarization strategy which is learned in an end-to-end fashion along with kernel ridge regression parameters, is due to the non-differentiability of the operation performed by the OPU. We overcome this difficulty by considering OPUs as a black-box and by employing the REINFORCE gradient estimator, which allows us to calculate the gradient of the loss function with respect to the weights of the binarization encoder and to optimize these together with the parameters of kernel ridge regression with gradient-based optimization.

Through our experimental campaign on a variety of tasks and datasets, we show that our method outperforms alternative unsupervised and supervised binarization techniques.

Keywords: optimization, random features, linear regression, optical processing unit.

1. Introduction

Statistical models based on kernel methods offer powerful and theoretically well-understood tools for complex data modeling problems. The limitation of employing these kernel-based models in practice is that a naive implementation scales poorly with the size of the data set, and there has been a tremendous amount of work in the direction of mitigating this issue by introducing approximations.

In this context, Nyström approximations [1] and random features [2] are very popular techniques to scale kernel methods virtually to any number of data, thanks to mini-batch formulations [3], [4].

The focus of this work is on random feature approximations, where by kernel-based models are "linearized" by an equivalent linear model with a set of suitably constructed random basis functions. The motivation behind this work is to considerably accelerate the construction of random features, while reducing power consumption, by resorting to a

dedicated hardware, which we refer to Optical Processing Units (OPUs).

OPUs are computing devices which perform random projections of input vectors by exploiting the physical phenomenon of scattering a light source through a diffusive medium [5]. The random projection is then followed by a nonlinear operation, making the whole pipeline of computation exactly what is needed to construct random features to approximate kernel-based models. Crucially, OPUs offer the possibility to operate with a number of random features at the speed of light and with low-power consumption, representing a unique solution to further improve scalability of kernel machines. As an example, OPU-based random feature approximations have successfully been proposed to carry out approximate kernel ridge regression in [6], [7].

One limitation associated with working with OPUs is that, because of the hardware setup, input vectors need to be binarized. In addition, the random projection matrix characterizing the device is unknown, and can only be retrieved through an expensive calibration procedure.

In this paper, we propose a novel binarization strategy for OPUs which is learned along with the regression/classification task in an end-to-end manner, meaning that the parameters of the binarization part are learned along with the kernel-based model parameters. In order to achieve this, we overcome the limitation that OPU projection matrices are unknown by employing the so-called REINFORCE gradient estimator, which allows us to treat the OPU as a black-box. Through experiments on several UCI classification/regression problems, we show that our proposal outperforms alternative unsupervised and supervised binarization techniques. This paper is an extended version of [8]; compared to the shorter version, we expand on the methods by analyzing the bounds on the objective functions of the proposed approaches, and we expand on the experiment by considering a larger class of kernels and image-based classification problems.

2. Related work

In neural networks, binarization is generally targeting intermediate layer activations, and it may also stem from binarization of model parameters; in these cases, binarization is mostly introduced to reduce computational cost and memory consumption [9]. Neural networks with binary hidden layers find applications in binary autoencoders for hashing [10], data compression [11], and hard attention mechanism [12]. The binarization of layer activations is obtained by a suitable choice of activation functions; for instance, the sign or Heaviside functions for the deterministic case, or the sigmoid or $tanh$ functions combined with the Bernoulli distribution for the stochastic case [13], [14]. The most popular technique to propagate gradients through such activation functions is the so called straight-through estimator (STE) [15]. More recently, there have been proposals to replace the STE with another estimator through a relaxation technique, also known as the Gumbel Softmax-trick [16]. Also, different kinds of target propagation are used to learn suitable targets for each binary layer and then train the associated parameters with relaxation techniques or combinatorial optimization [17], [18], [19].

Focusing on OPUs, currently the standard approach to binarize data makes use of a binary autoencoder [11]. Such a binary autoencoder is trained independently from the OPU device, and it gives the possibility to perform the binarization operation by means of its encoder part. The autoencoder consists of a fully-connected encoder and decoder. The hidden layer has a Heaviside activation function, so its output is binary. The training procedure updates the weights of the decoder with backpropagation and weights of the encoder are forced to be equal to the weights of the decoder in order to be able to reproduce the input.

In this work, we aim to develop a supervised binarization model which is learned together with the supervised learning task. That is, we aim to provide a training procedure for the heterogeneous model consisting of the kernel ridge regression model approximated with random features and the binarization encoder before the OPU. In this context, a general-purpose framework called Method of Auxiliary Coordinates (MAC) was proposed in [19] with examples of application in [10] and [20]. The authors propose to introduce auxiliary variables into a deep neural network. These auxiliary variables are assigned the role of pre-activations for each layer, and they get replaced during the forward pass. The first step of the optimization targets the auxiliary variables, and, after this step, the parameters of each layer are optimized to regress on these variables, which take the role of layer-specific labels. This is very beneficial when some layers are discrete and vanilla backpropagation is not applicable. In [20], this approach is used to train a fully connected network with binary activation functions, using a STE to propagate a learning signal through the non-differentiable parts. Reference [10] is especially interesting because authors illustrate, how discrete binary layers can be optimized withing larger, non-binary model.

While splitting the optimization of the binarization and the model is a viable option, we still need a way to training each part individually. There is a wide variety of ways to obtain a solution for kernel ridge regression with the random feature approximation, so the most difficult point is how to optimize the part consisting of the binary encoder and the OPU, because it combines a non-differentiable function with an implicit random projection. These make the STE from [20] inapplicable. Also, we found that the combinatorial approach used in [10] and [17] is inapplicable for our case for two reasons. First, it is suitable only when the binary dimension is relatively small, which might be a limitation for a general

solution. Second, the combinatorial approach combined with MAC converges in one iteration to poor local optima, and this happens because of the model setup which is different from the ones in [10] and [17].

From a different point of view, it is possible to view our problem through the lenses of reinforcement learning, where it is necessary to propagate binary codes through the OPU instead of discrete actions through the black-box environment. Instead of maximizing the reward from the environment, we are trying to minimize the loss function. The classical algorithm to solve this problem is REINFORCE [1]. This allows one to calculate gradients of the reward with respect to parameters of the policy that generates actions. The applicability of this method to other settings with black-box elements was shown in [21]. There are various versions of this algorithm intended to reduce variance of the gradient of the parameters. Very frequently they are based on relaxations of the non-differentiable sampling procedure [22], or approximation of the black-box part of the model [23]. It also worth noting that there exist competitive alternatives to REINFORCE, such as the one in [24], later extended with variance reduction [25] or relaxation [26].

3. Background

3.1. Kernel Ridge Regression

In this paper, we focus on kernel ridge regression for supervised learning tasks. Let $\mathbf{X} = \mathbf{x}_1, \dots, \mathbf{x}_n$ be a set of input vectors $\mathbf{x} \in \mathbb{R}^d$ and let $\mathbf{y} = y_1, \dots y_n$ be a set of labels associated with the input vectors.

The labels y_i can be continuous or binary depending on whether the task is regression or classification. Kernel ridge regression is a statistical model which constructs a functional relationship between the inputs and the labels which belongs to the so-called Reproducing Kernel Hilbert Space (RKHS). The properties of such functions, such as smoothness, are characterized by the choice of a so-called kernel function $k(\cdot,\cdot): \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ [27], which is a positive semi-definite function of pairs of input points returning a scalar. The reproducing property of kernel functions is $\langle k(\mathbf{x},\cdot), k(\mathbf{y},\cdot)\rangle = k(\mathbf{x},\mathbf{y})$. Positive definiteness of kernel functions implies that we can express $k(\mathbf{x_i},\mathbf{x_j}) = \varphi(\mathbf{x_i})^T\varphi(\mathbf{x_j})$ for some set of (possibly infinite) basis functions $\varphi(\cdot)$.

In order to derive the conventional formulation of kernel ridge regression, it is useful to start from linear regression, where a set of model parameters $\mathbf{w}$ is introduced to express a linear relationship between input and labels. Then, one introduces the following optimization problem:

$$\hat{\mathbf{w}} = \underset{w}{\operatorname{argmin}} \frac{1}{2}\sum_{i=1}^{n}(y - \mathbf{w}^\top \mathbf{x})^2 - \frac{\lambda}{2}||\mathbf{w}||_2^2 \qquad (1)$$

The objective function contains two terms; the first is a model fitting term, while the second is a regularization term, which prevents the weights to become too large. The solution to this optimization problem is available in closed form, given that the objective is quadratic with respect to the parameters, yielding:

$$\hat{\mathbf{w}} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{y} \qquad (2)$$

Using standard algebraic manipulations involving the Woodbury identity, we can re-express the solution as:

$$\hat{\mathbf{w}} = \mathbf{X}^T(\mathbf{X}\,\mathbf{X}^\top + \lambda\mathbf{I})^{-1}\mathbf{y} \qquad (3)$$

While this is costly than the previous expression in the common case where $d < n$ (inversion of a $n \times n$ matrix rather than a $d \times d$ matrix), this formulation is useful to derive kernel ridge regression.

Imagining to introduce basis functions $\phi(\cdot) = \left(\phi_1(\cdot), \dots, \phi_D(\cdot)\right)^\top$, we can solve this new optimization problem

$$\hat{\mathbf{w}} = \underset{W}{\operatorname{argmin}} \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \mathbf{w}^\top\phi(\mathbf{x}_i)\right)^2 + \frac{\lambda}{2}||\mathbf{w}||^2 \qquad (4)$$

with solution

$$\hat{\mathbf{w}} = \Phi^\top(\Phi\,\Phi^\top + \lambda\mathbf{I})^{-1}\mathbf{y} \qquad (5)$$

Evaluating the model at a given input $\mathbf{x}_*$ yields:

$$\phi(\mathbf{x}_*)^\top\hat{\mathbf{w}} = \phi(\mathbf{x}_*)^\top\mathbf{\Phi}^\top(\mathbf{\Phi}\mathbf{\Phi}^\top + \lambda\mathbf{I})^{-1}\mathbf{y} \qquad (6)$$

In this expression, we recognize the scalar product of vectors of basis functions. What we can do then, is to express these scalar products as a kernel function and obtain:

$$\phi(\mathbf{x})^\top\hat{\mathbf{w}} = \mathbf{k}_*(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{y} \qquad (7)$$

where $\mathbf{k}_* = \left(k(\mathbf{x}_1,\mathbf{x}_*), \dots, k(\mathbf{x}_n,\mathbf{x}_*)\right)^\top$ and $K_{ij} = k(\mathbf{x}_i,\mathbf{x}_j)$. In practice, one first chooses a kernel function, and this induces a set of basis function; the beauty of this formulation is that one never explicitly works with the set of basis functions and all we need to use this model in practice is the evaluation of kernel functions among inputs.

3.2. Random Feature Approximation

One of main limitations of kernel methods is scalability to large datasets. The problem arises from the need to evaluate and perform algebraic operations with the so-called Gram matrix **K**. Because **K** is an $n \times n$ matrix, evaluating and storing **K** requires $\mathcal{O}(n^2)$ computations and storage, while any algebraic operations, such as factorizations to handle the inverse of $\mathbf{K} + \lambda\mathbf{I}$, requires $\mathcal{O}(n^3)$ operations. These prevent the applicability of kernel methods in their exact form to datasets of size beyond a few thousand. It is worth noting that some approaches have been proposed to solve algebraic operations in an iterative fashion and without the need to store **K** [28], [29], [30], but they still require $\mathcal{O}(n^2)$ computations for each iteration of their solvers. Furthermore, while the number of iterations of the solvers is much lower than n in practice, in the worst case it can be $\mathcal{O}(n)$, leading to a worst-case complexity of $\mathcal{O}(n^3)$.

The literature offers a number of solutions to scale kernel methods to large data linearly in the number of data, such as Nyström approximations [31] and random features [2]. In this work we focus in particular on random feature approximations, given that these have a practical implementation in hardware in the optical processing units that we consider in this work.

The random feature approximations form a class of approximations which attempt to construct a finite set of basis functions $\phi(\cdot) \in \mathbb{R}^D$ such that

$$k(\mathbf{x}_i, \mathbf{x}_j) \approx \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) \tag{8}$$

There are different ways to construct such sets of basis functions, depending on the kernel. For example, so-called random Fourier features are commonly employed to approximate the Gaussian kernel:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right) \tag{9}$$

Appealing to Bochner's theorem [2], this kernel, which is shift-invariant due to dependence on $\boldsymbol{\tau} = \mathbf{x}_i - \mathbf{x}_j$, admits an alternative expression as:

$$k(\boldsymbol{\tau}) = \int p(\boldsymbol{\omega}) \exp(\mathrm{i}2\pi\boldsymbol{\omega}\boldsymbol{\tau})\, d\boldsymbol{\omega} \tag{10}$$

where $p(\boldsymbol{\omega})$ is a proper density function and $\mathrm{i} = \sqrt{-1}$. Interpreting this as an expectation under $p(\boldsymbol{\omega})$, it is possible to approximate the integral as an expectation using Monte Carlo.

$$k(\boldsymbol{\tau}) = \frac{1}{D}\sum_r \exp\left(\mathrm{i}2\pi\boldsymbol{\omega}^{(r)}\boldsymbol{\tau}\right) \tag{11}$$

with $\boldsymbol{\omega}^{(r)} \sim p(\boldsymbol{\omega})$. Furthermore, it is possible to use simple trigonometric identities to verify that the complex exponential can be broken down as a scalar product with terms depending on $\mathbf{x}_i$ and $\mathbf{x}_j$ respectively

$$k(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{D}\phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) \tag{12}$$

with

$$\phi_r(\mathbf{x}) = \left(\sin\left(\mathbf{x}^\top\boldsymbol{\omega}^{(r)}\right), \cos\left(\mathbf{x}^\top\boldsymbol{\omega}^{(r)}\right)\right) \tag{13}$$

We refer the reader to [2], [32], [3], [33] for random features derived from alternative integral representation to the Fourier transform.

3.3. Random Features on Optical Processing Units

In this section we discuss Optical Processing Units (OPUs) in the context of random features. In the previous section we discussed random features as a way to approximate models involving kernels; for OPUs, instead, the device produces random features (fast and with little power consumption) and the question that we aim to address here is how to use these to implement approximate kernel machines.

OPU are computing devices which exploit the physical process of scattering of light to perform a random projection operation of a given vector. In particular, given a binary vector $\mathbf{x}_\mathrm{i} \in \mathbb{R}^d$, OPUs perform a multiplication by a random matrix **R** and apply the nonlinear activation function $||\cdot||^2$. In other words,

$$\phi(\mathbf{x}) = \frac{1}{\sqrt{D}}\|\mathbf{R}\mathbf{x}\|^2 \tag{14}$$

The matrix $\mathbf{R} \in \mathcal{C}^{D\times d}$ is a complex Gaussian matrix with elements $R_{ij} \sim \mathcal{CN}(0,1)$. Previous works have established that in the limit of an infinite number of random features, the equivalent kernel is the following [6]:

$$k(\mathbf{x}, \mathbf{y}) \approx \phi(\mathbf{x})\phi(\mathbf{y}) \overset{D\to\infty}{=} ||\mathbf{x}||^2||\mathbf{y}||^2 + (\mathbf{x}^T\mathbf{y})^2 \tag{15}$$

Therefore, when using OPUs for kernel ridge regression, we are implicitly working with this polynomial kernel.

Recently a new version of OPUs has been proposed and developed in [34], which allows one to perform linear random feature projections

$$\psi(\mathbf{x}) = C\mathbf{R}\mathbf{x} \tag{16}$$

where C is a fixed constant.

This novel type of OPU opens to the possibility to approximate a wide variety of kernels by choosing an appropriate activation function [2], [33]. For example,

it is possible to apply trigonometric activation functions to the outputs of the OPU:

$$\psi'(\mathbf{x}) = \begin{bmatrix}\sin(\psi(\mathbf{x}))\\ \cos(\psi(\mathbf{x}))\end{bmatrix} \tag{17}$$

This type of random features is called Random Fourier Features (RFF). It was proven in [2] that this kind of random features allows to approximate RBF kernels.

$$\frac{1}{D}\sum_{i=1}^{D}\psi(\mathbf{x})^{\top}\psi(\mathbf{y}) \tag{18}$$
$$= \frac{1}{D}\sum_{i=1}^{D}\left(\begin{bmatrix}\sin(\psi(\mathbf{x}))\\ \cos(\psi(\mathbf{x}))\end{bmatrix}^{\top}\begin{bmatrix}\sin(\psi(\mathbf{y}))\\ \cos(\psi(\mathbf{y}))\end{bmatrix}\right) =$$
$$= \mathbb{E}_{\boldsymbol{\omega}}[cos(\boldsymbol{\omega}(\mathbf{x}-\mathbf{y}))] = k_{RBF}(\mathbf{x},\mathbf{y})$$

As mentioned before, an important aspect of OPUs is that their input should be binary; this paper proposes a novel way to carry out a binarization of its input along with the kernel ridge regression task in an end-to-end fashion.

4. Methods

4.1. REINFORCE for Kernel Ridge Regression with Binarized Inputs

In order to be able to implement kernel ridge regression on OPUs we need to binarize the inputs $\mathbf{x}_i$, and we propose to do so by employing an encoder, implemented as a neural network, parameterized by a set of weights $\mathbf{W}_{\text{enc}}$. The encoder transforms the inputs to kernel-based models $\mathbf{x}_i$ and turns them into a set of Bernoulli-distributed binary random variables $\mathbf{z}_i$.

In particular, we denote by $f_k(\mathbf{x}, \mathbf{W}_{\text{enc}})$ the function implemented by the encoder which parameterizes the Bernoulli distribution associated with the *k*th element of the output, that is z_k. Recalling the random feature formulation of linear regression of **Section 3**, we propose the following approach to construct an approximate kernel-based model with binary inputs:

$$\tilde{y} = \mathbb{E}_{\mathbf{z}}\left[\mathbf{w}_{\text{regr}}^{\top}\phi(\mathbf{z})\right] + \varepsilon \tag{19}$$

where $\mathbf{z} \sim \text{Bernoulli}\big(f(\mathbf{x}, \mathbf{W}_{\text{enc}})\big)$ and $\mathbf{w}_{\text{regr}}$ are parameters of the linearized regression model. Note how in this formulation the binary vectors $\mathbf{z}$ are treated stochastically due to the expectation under the Bernoulli distribution induced by the encoder. The reason for this is that it allows us to employ the so-called REINFORCE gradient estimator, as we discuss next.

REINFORCE, also known as the log-derivative trick or score function estimator, offers a way to estimate the gradient of the expectation of a non-differentiable function $f(z)$ under the distribution of the input random vector variables z:

$$\nabla_\theta E_{p(z;\theta)} f(z) = \nabla_\theta \int p(z;\theta) f(z) dz = \tag{20}$$
$$\int \nabla_\theta p(z;\theta) f(z) dz =$$
$$\int p(z;\theta)\frac{\nabla_\theta p(z;\theta)}{p(z;\theta)} f(z) dz$$
$$= \mathbb{E}_{p(z;\theta)} \nabla_\theta \log p(\, z;\theta) f(z)$$
$$\approx \frac{1}{M}\sum_{i=1}^{M} \nabla_\theta \log p(z;\theta) f(z)$$

where M is number of samples drown from $p(z,\theta)$. Applying REINFORCE to our approximate kernel-based model yields the following optimization objective:

$$\min_{\mathbf{w}_{\text{regr}}, \mathbf{W}_{\text{enc}}} \mathbb{E}_{\mathbf{z}\sim \text{Bernoulli}(f(\mathbf{x},\mathbf{W}_{\text{enc}}))}\left[\mathcal{L}\left(\mathbf{y}, \phi(\mathbf{Z})\mathbf{w}_{\text{regr}}\right)\right] \tag{21}$$
$$+\lambda_{\text{enc}}\|\mathbf{W}_{\text{enc}}\|^2 + \lambda_{\text{regr}}\left\|\mathbf{w}_{\text{regr}}\right\|^2$$

In this expression, we denoted by $\mathcal{L}(\mathbf{y}, \tilde{\mathbf{y}})$ the loss function associated with the task at hand and by $\mathbf{Z}$ the matrix that contains binary encoded variables for the whole training set $\mathbf{X}$. We can optimize this objective by means of gradient-based techniques; for this we require that we are able to compute the gradient of the objective with respect to all parameters. The gradient of the first term of the objective with respect to $\mathbf{W}_{\text{enc}}$, which is the most involved part, is:

$$\nabla_{\mathbf{W}_{\text{enc}}} \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})}\left[\mathcal{L}\left(y, \mathbf{w}_{\text{regr}}^{\top}\phi(\mathbf{z})\right)\right] \approx \tag{22}$$
$$\approx \frac{1}{M}\sum_{i=1}^{M}\mathcal{L}\left(y, \mathbf{w}_{\text{regr}}^{\top}\phi(\mathbf{z})\right)\nabla_{\mathbf{W}_{\text{enc}}}\log q(\mathbf{z})$$

while the derivatives of the other terms are straightforward to compute. With this derivation, we observe that it is then possible to jointly optimize all parameters, leading to what it is commonly referred to as an *end-to-end* approach. In the remainder of this paper, we refer to this method as End-to-End SE, where SE stands for Supervised Encoder.

4.2. Variance Reduction

REINFORCE is known to suffer from large variance of the gradients. In order to reduce the variance of this estimator, we employ control variates [25]. In this approach, we add a set of random variables to the estimator, such that these variables have zero mean, so they do not alter the expectation of the gradient. The aim is to construct such variables so as to reduce the overall variance of the estimator:

$$\nabla_{\mathbf{W}_{\text{enc}}} \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z})}\left[\mathcal{L}\left(y, \mathbf{w}_{\text{regr}}^{\top}\phi(\mathbf{z})\right)\right] \approx \tag{23}$$

$$\frac{1}{M}\sum_{i=1}^{M} \nabla_{\mathbf{W}_{\text{enc}}} \log q\left(\mathbf{z}\right)\left(\mathcal{L}\left(y, \mathbf{w}_{\text{regr}}^{\top}\phi(\mathbf{z})\right) - \mathbf{v}\right)$$

$$\text{where } \mathbf{v} = \frac{1}{M-1}\sum_{i\neq j} \mathcal{L}\left(y, \mathbf{w}_{\text{regr}}^{\top}\phi(\mathbf{z})\right)$$

4.3. Lowering the Cost of REINFORCE

The estimation of the gradient of the End-to-End SE with respect to $\mathbf{W}_{\text{enc}}$ can be expensive when the number of random features is large. This is due to the fact that this requires multiple samples to be passed from the encoder through the random projection and the approximate kernel ridge regression model. In this section we propose a strategy to reduce the complexity of REINFORCE applied to our model, whereby we average set of basis functions under the resampling of the binary variables as follows:

$$\tilde{y} = \mathbf{w}_{\text{regr}}^{\top}\mathbb{E}_{\mathbf{z}}\left[\phi(\mathbf{z})\right] + \varepsilon \tag{24}$$

where $\mathbf{z} \sim \text{Bernoulli}\big(f(\mathbf{x}, \mathbf{W}_{\text{enc}})\big)$

With this new modeling assumption, the training is based on a modified optimization problem as follows:

$$\min_{\mathbf{w}_{\text{regr}}\mathbf{W}_{\text{enc}}} \mathcal{L}\left(\mathbf{y}, \mathbb{E}_{\mathbf{z}\sim\text{Bernoulli}\left(f(\mathbf{x},\mathbf{W}_{\text{enc}})\right)}[\phi(\mathbf{Z})]\mathbf{w}_{\text{regr}}\right) \tag{25}$$

$$+\lambda_{\text{enc}}\|\mathbf{W}_{\text{enc}}\|^2 + \lambda_{\text{regr}}\left\|\mathbf{w}_{\text{regr}}\right\|^2$$

Again, we can perform gradient-based optimization. Focusing on the first term, which is the nontrivial one to differentiate in the objective, we obtain

$$\nabla_{\mathbf{W}_{\text{enc}}}\mathcal{L} = \frac{d\mathcal{L}}{d\big(\mathbb{E}\phi(\mathbf{z})\big)}\nabla_{\mathbf{W}_{\text{enc}}}\mathbb{E}\big(\phi(\mathbf{z})\big) \tag{26}$$

where $\nabla_{\mathbf{W}_{\text{enc}}}\mathbb{E}\big(\phi(\mathbf{z})\big)$ is calculated with the REINFORCE estimator. In the remainder of the paper, we will refer to this method as Isolated Supervised Encoder (SE).

Regarding the comparison of the End-to-End SE and Isolated SE, we can note the following relationship between these models in the case of regression problems. In the data term of the optimization objective (25) we can put an expectation over the whole matrix product of the random features map and the regression weights instead of an expectation over the random features only. Then we can move the expectation in such a way that it is taken over the whole term within the squared norm. We can do this because within one gradient step iteration, neither $\mathbf{y}$ nor $\mathbf{W}_{\text{enc}}$ are considered as random variables. In this case the data term looks as follows:

$$\begin{aligned}\left\|\mathbf{y} - \mathbb{E}[\phi(\mathbf{Z})]\mathbf{w}_{\text{regr}}\right\|^2 &= \left\|\mathbf{y} - \mathbb{E}\left[\phi(\mathbf{Z})\mathbf{w}_{\text{regr}}\right]\right\|^2 \\ &= \left\|\mathbb{E}\left[\mathbf{y} - \phi(\mathbf{Z})\mathbf{w}_{\text{regr}}\right]\right\|^2\end{aligned} \tag{27}$$

In turn, End-to-End SE has a following data term as part of its optimization objective (21):

$$\mathbb{E}\left[\left\|\mathbf{y} - \phi(\mathbf{Z})\mathbf{w}_{\text{regr}}\right\|^2\right] \tag{28}$$

We can note that the squared loss is convex function. Thus, we can apply Jensen's inequality to obtain the following expression:

$$\begin{aligned}\mathbb{E}\left[\left\|\mathbf{y} - \phi(\mathbf{Z})\mathbf{w}_{\text{regr}}\right\|^2\right] &\geq \left\|\mathbb{E}\left[\mathbf{y} - \phi(\mathbf{Z})\mathbf{w}_{\text{regr}}\right]\right\|^2 \\ &= \left\|\mathbf{y} - \mathbb{E}\left[\phi(\mathbf{Z})\mathbf{w}_{\text{regr}}\right]\right\|^2\end{aligned} \tag{29}$$

As a result, End-to-End SE optimizes upper bound of the Isolated SE objective.

5. Results

5.1 Experiments on the UCI datasets

We compared the performance of the proposed approaches for a non-linear OPU (End-to-End SE and Isolated SE) and a linear OPU that uses trigonometric activations (End-to-End SE with RFF) against a model based on unsupervised autoencoder proposed in [11], an encoder trained with direct feedback alignment (DFA) [35] and a Kernel Ridge Regression (KRR) based on a Radial Based Function kernel (RBF). Results are reported in **Fig. 1** for several UCI regression and classification problems [36]. We want to emphasize that the main competitors of the proposed methods are the ones based on unsupervised autoencoder and encoder trained by DFA, because kernel ridge regression is unable to work with large datasets, and OPU-based regression just approximates this method and is intended to replace it on large datasets.

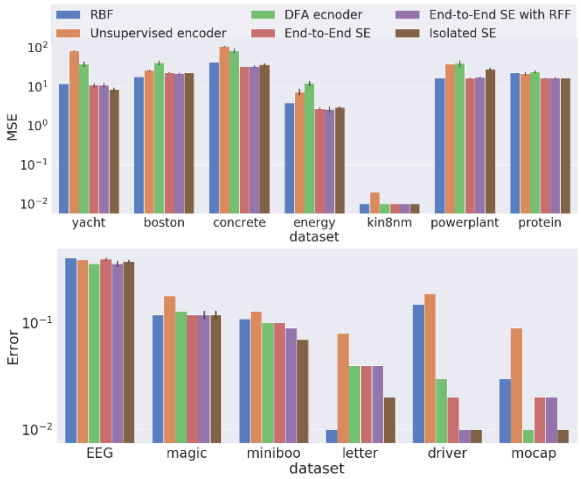

Fig. 1. Mean squared error (MSE) for regression (top) and negative error on classification (bottom) datasets comparison.

For KRR experiments we used Mean Squared Error (MSE) as a loss function. To apply KRR to the classification problems we replaced 0 and 1 in class labels with -1, 1 and solved a classification problem as a regression one using MSE loss as an optimization objective. For all other models we used MSE loss for the regression problems and Cross Entropy (CE) loss for the classification problems. We used a logistic activation function on the last layer for the classification tasks.

For Isolated SE and End-to-End SE as an encoding function $f(\mathbf{x}, \mathbf{W}_{\text{enc}})$ providing parameters for the Bernoulli distribution, we chose a single linear layer with a sigmoid activation.

$$f(\mathbf{x}, \mathbf{W}_{\text{enc}}) = \sigma\left(\mathbf{W}_{\text{enc}}^{\top}\mathbf{x}\right) \tag{30}$$

All hyperparameters for the DFA encoder, End-to-End SE and Isolated SE models (size of binary embedding, learning rate, l2 regularization for the encoder and the regression layer) were chosen with a random search during cross-validation. Kernel parameters of KRR were tuned by random search with cross-validation. This poses computational challenges for the large datasets (MiniBoo, MoCap), so we resort to random Fourier feature approximations for these cases.

For the models involving random features (both Fourier and OPU-generated ones) we have tuned the variance of the distribution that generates these random features. Concretely, assuming that the elements of the $\mathbf{R}$ matrix generating the random projections are distributed through the standard Normal distribution, we can obtain a new random matrix $\mathbf{R}'$ by multiplying $\mathbf{R}$ by any variance, for instance:

$$\phi'(\mathbf{x}) = c|\mathbf{R}'\mathbf{x}|^2 = c|\frac{\mathbf{R}}{\alpha}\mathbf{x}|^2 = c\frac{1}{\alpha^2}|\mathbf{R}\mathbf{x}|^2 \tag{31}$$

It is enough to multiply the output of the OPU by an additional parameter γ, such that $\gamma^2 = \frac{1}{\alpha^2}$, and optimize them with standard gradient descent. The parameter γ is not equivalent to the lengthscale parameter of the RBF kernel. In practice, it has an effect of outputscale parameter of RBF kernel, as it has simply a scaling effect on the kernel.

On the regression problems, both proposed methods outperformed their main competitors. On the classification problems, the DFA-based approach was better only on one dataset, and on all other datasets the proposed methods performed better or equally well. Regarding the type of a kernel approximated by the OPU, the experiments show that the linear OPU with trigonometric activations performs as good as OPU kernel for most of the datasets. It gives performance gains only for some classification problems. Considering the comparison between the proposed methods, we see that End-to-End SE is more stable and requires a significantly fewer number of samples from the encoder, although Isolated SE showed slightly better results on classification problems.

We considered including results obtained by running these models on the real OPU (**Fig. 2**). Unfortunately, the regression problems required such a large number of epochs that we could not perform the experiments in a reasonable amount of time.

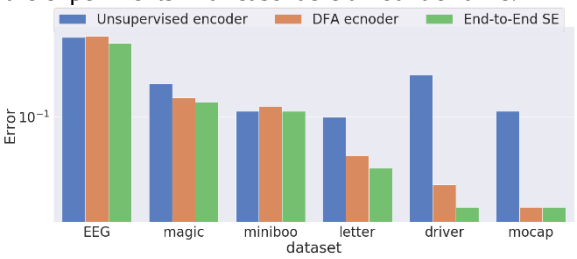

Fig. 2. Error comparison on classification (bottom) datasets for experiments on a real hardware.

We also tested the performance of our approach with respect to the number of samples required to employ REINFORCE. We found that End-to-End SE can achieve good results with a small amount of samples from the encoder, and the increase of amount of samples does not seem to improve performance.

Finally, we evaluated the effect of variance reduction on convergence speed and performance for End-to-End SE model. In **Fig. 3** we report results for one classification and one regression problem. The convergence curves indicate that the convergence speed is benefits from the gradient variance reduction.

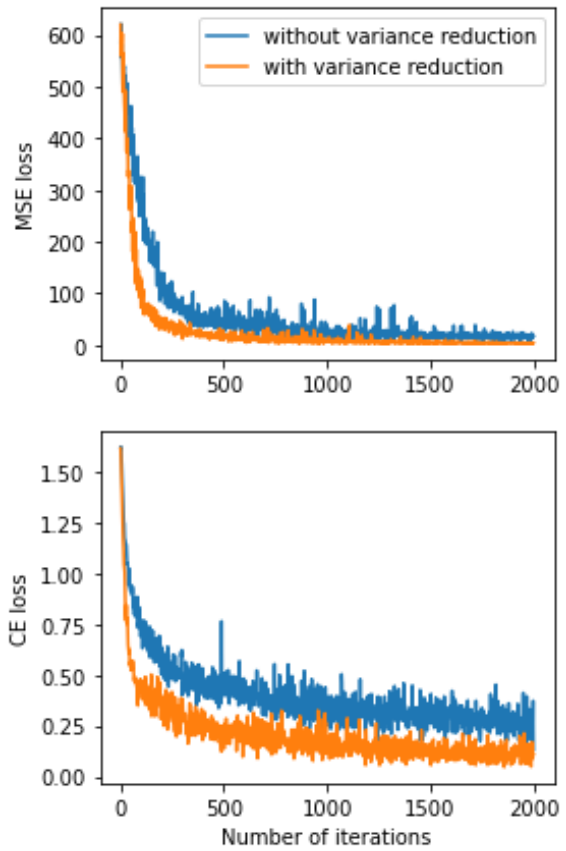

Fig. 3. Convergence of the training procedure on classification problem: mocap dataset (bottom) and regression problem: boston dataset (top).

5.2. Experiments on image data

In this section we evaluate an optical random feature regression approach for image classification tasks with several different binarization techniques including the proposed methods.

The kernel generated by the OPU (15) is an example of a polynomial kernel. Polynomial kernels, unlike more popular RBF kernels, take into account interaction between different feature dimensions. This property is especially important for image data because a relative alignment of pixels is crucial for image classification. Of course, when we are working with OPUs, kernel takes into account an alignment of different dimensions of the binary embedding of an image instead of the pixels. The relative alignment of different dimensions of the binary embedding most probably does not contain exactly the same information as the mutual alignment of pixels. But until the binary encoder does not have disentanglement properties, the mutual interaction of dimensions of the binary embedding have to contain an additional information about the image. That is why it is still important to use a kernel that is capable to take into account these relationships.

For the experiments on image data, we used two convolutional architectures of the binary encoder. First architecture was inspired by the LeNet model (**Table 1**).

Table 1. LeNet-based binary encoder architecture

Layer	Dimensions
Conv2D	5×5, 6 filters
MaxPooling	2×2
Conv2D	5×5, 16 filters
MaxPooling	2×2
Linear	576×512
Linear	512×d_{binary}
Binarization layer	d_{binary}

We performed experiments on three classical image classification datasets (**Table 2**). We compared End-to-End SE with a model that used autoencoder to train the encoder (AE) and a model that used direct feedback alignment (DFA) for the same purpose. The results are shown in **Table 3**.

Tab. 2. Image datasets used in the experiments

Dataset	Train/test size	Classes	Dimension
MNIST	60000/10000	10	1×28×28
F-MNIST	60000/10000	10	1×28×28
CIFAR10	50000/10000	10	3×32×32

Tab. 3. Classification error obtained by the model with the LeNet encoder

Dataset	AE	DFA	End-to-End SE
MNIST	0.06±0.02	0.31±0.03	0.01±0.00
F-MNIST	0.20±0.01	0.47±0.01	0.09±0.00
CIFAR10	0.55±0.01	0.81±0.02	0.32±0.01

We used the same encoder architecture with the same hyperparameters for each binarization method. The size of the binary embedding was set to 400, except of the unsupervised AE method. The reason why we trained the unsupervised AE differently for these experiments is because we were using complex convolutional models and it was hard to adapt the method proposed in [11]. This binarization approach requires to use exactly the same values of weights both for the encoder and for the decoder models. It means that this method requires to build a decoder model that is symmetrical to the encoder model. Achieving this property for convolutional neural networks is difficult because for the decoder it is difficult to pick equivalent symmetric operations for

convolutional and pooling layers in the encoder. Transposed convolutions and interpolation operations, that are used in decoders for image data, are suitable for training of the autoencoder by end-to-end backpropagation. They learn operations that are not symmetric to convolutions and pooling layers of the encoder. The method proposed in [11] assumes that only the decoder is trained and the encoder copies weights from it, that is impossible to do for asymmetric operations in decoder and encoder. That is why we trained the autoencoder model in a different way. The encoder of the AE model used a $tanh$ activation function at the output. We used a parameter β that controls steepness of the $tanh$ function. We slowly increased the value of this parameter from $\beta = 1$ to $\beta = 100$ in the process of training. At the end of the training, the function $tanh$ with a high value of the parameter β is almost equivalent to a shifted and scaled Heaviside function.

$$2h(x) - 1 \approx tanh(\beta x),\ \beta \to \infty \quad (32)$$

The results of the DFA approach signify that this type of gradient updates is not suitable for convolutional models. This observation is supported by other researchers [37], [38].

Unsupervised AE was able to provide acceptable accuracy for the MNIST dataset, but on CIFAR-10 its performance dropped significantly. A possible explanation is that simple convolutional AE is unable to extract reasonable binary representation of complex images. The AE model used in the experiments was able to reconstruct simple images from MNIST dataset. But the reconstruction quality of the same model was much worse for the CIFAR-10 dataset. When the dimensionality of the binary embedding was equal to 400, the AE model was unable to generate any sensible images. Thus, we had to increase the size of the binary embedding to 1024. But the reconstructed images were very blurry even with this modification.

Because of the poor performance of the unsupervised AE baseline, we decided to add another baseline to the comparison. For this experiment we used a RESNET-based convolutional network as the encoder. LBAE approach proposed in [39] implements an autoencoder with a binary latent space. The training procedure of this method is based on straight-through gradient estimator. The architecture of the binary encoder is represented in **Table 4** and **Table 5**. This encoder used leaky ReLU as an activation function. Batch-normalization was used before each activation function.

Tab. 4. Architecture of the RESNET-based binary encoder

Layer	Dimensions
Conv2D	3×3, 64 filters
Conv2D	4×4, 64 filters
Residual Block	3×3, 64 filters
Conv2D	4×4, 64 filters
Residual Block	3×3, 64 filters
Conv2D	4×4, 128 filters
Linear	4096×d_{binary}
Binarization layer	d_{binary}

Tab. 5. Residual block structure. The number of filters is specified in **Tab. 4**

Layer	Dimensions
Conv2d	3×3
Conv2d	3×3

Table 6 contains the results of the comparison between the LBAE-based and End-to-End SE-based encoders in terms of classification error.

Tab. 6. Classification error for models with the RESNET encoder

Dataset	LBAE	End-to-End SE
MNIST	0.15±0.01	0.01 ±0.00
F-MNIST	0.24±0.02	0.06±0.01
CIFAR-10	0.63±0.02	0.17±0.01

Both binarization approaches used the same RESNET-based architecture of the encoder network. We dropped the DFA approach from the comparison because of its poor performance.

The results of this experiment showed interesting property of the unsupervised approach for training of the binary encoder. The LBAE-based encoder with a deeper network performed worse than the simpler autoencoder with $tanh$ annealing in terms of classification error, while the LBAE approach was better in image reconstruction task. It seems, that the binary latent projection of image data, that is suitable for image reconstruction, is unsuitable for image classification.

As with the UCI data we evaluated the effect of variance reduction.

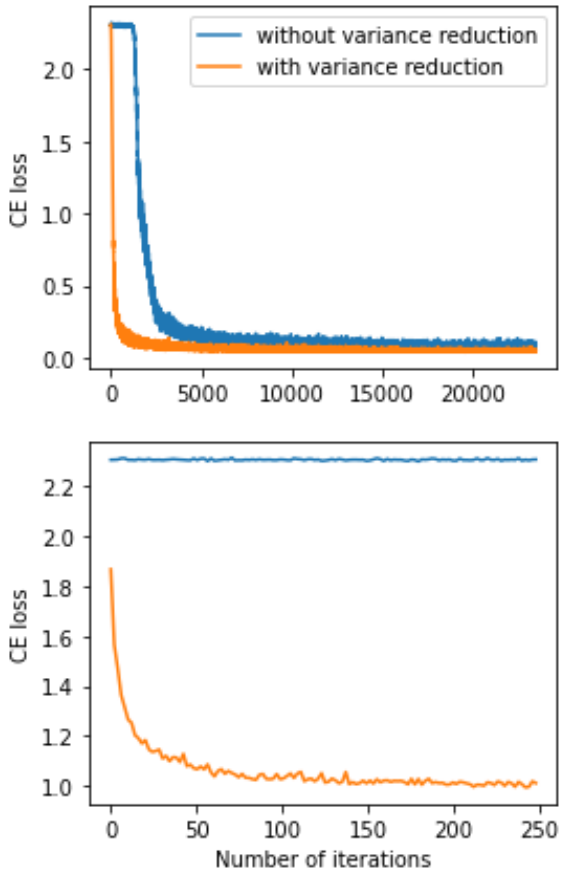

Fig. 4. Training loss with and without variance reduction

As we can see, variance reduction plays a crucial role for image classification. Without this technique the proposed method is unable to train the model for CIFAR-10. We assume, that it happens because in deeper models variance has a multiplicative effect when the number of layers increases.

6. Conclusion

Recent advances in alternatives to transistor-based hardware are bringing a new wave of sustainable computing solutions for machine learning [40]. This paper focuses on optical-based computing through OPUs [5], which perform randomized projections of binary input vectors at the speed of light with low energy consumption. In this paper, we considered these randomized computations to implement kernel-based models for regression and classification tasks through random feature approximations. In particular, we proposed a novel strategy to binarize the inputs of the given task so as to be able to employ OPUs inspired by reinforcement learning. The proposed strategy uses an encoder to map the inputs to a set of binary variables and employs the REINFORCE gradient estimator to estimate its parameters jointly with the parameters of the kernel-based model. We also explored ways to reduce the variance of the gradient estimator and accelerate convergence, which is key in a number of challenging modeling tasks such as image classification. Through a series of experiments, we showed that our proposal outperforms competitors based on unsupervised binarization and those that do not employ gradient information.

We are currently investigating our approach in the context of other kernel-based models, such as Gaussian processes [41], and their extension to deep models, such as Deep Kernel Learning [42] and Deep Gaussian processes [3].

References

[1] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, p. 229–256, 1992.

[2] A. Rahimi and B. Recht, "Weighted Sums of Random Kitchen Sinks: Replacing minimization with randomization in learning," in *Advances in Neural Information Processing Systems*, 2009.

[3] K. Cutajar, E. V. Bonilla, P. Michiardi and M. Filippone, "Random feature expansions for deep Gaussian processes," in *International Conference on Machine Learning*, 2017.

[4] J. Hensman, N. Fusi and N. D. Lawrence, "Gaussian Processes for Big Data," in *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, Arlington, 2013.

[5] A. Saade, F. Caltagirone, I. Carron, L. Daudet, A. Drémeau, S. Gigan and F. Krzakala, "Random projections through multiple optical scattering: Approximating kernels at the speed of light," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[6] R. Ohana, J. Wacker, J. Dong, S. Marmin, F. Krzakala, M. Filippone and L. Daudet, "Kernel computations from large-scale random features obtained by optical processing units," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[7] A. Cappelli, R. Ohana, J. Launay, L. Meunier, I. Poli and F. Krzakala, "Adversarial Robustness by Design through Analog Computing and Synthetic Gradients," *arXiv preprint arXiv:2101.02115*, 2021.

[8] B. Kozyrskiy, I. Poli, R. Ohana, L. Daudet, I. Carron and M. Filippone, "Binarization for Optical Processing Units via REINFORCE," in *Proceedings of the 3rd International Conference on Advances in Signal Processing and Artificial Intelligence*.

[9] H. Qin, R. Gong, X. Liu, X. Bai, J. Song and N. Sebe, "Binary neural networks: A survey," *Pattern Recognition*, vol. 105, p. 107281, 2020.

[10] M. A. Carreira-Perpinán and R. Raziperchikolaei, "Hashing with binary autoencoders," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[11] J. Tissier, C. Gravier and A. Habrard, "Near-lossless binarization of word embeddings," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

[12] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, 2015.

[13] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830,* 2016.

[14] J. W. T. Peters and M. Welling, "Probabilistic binary neural networks," *arXiv preprint arXiv:1809.03368,* 2018.

[15] Y. Bengio, N. Léonard and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432,* 2013.

[16] E. Jang, S. Gu and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144,* 2016.

[17] A. L. Friesen and P. Domingos, "Deep learning as a mixed convex-combinatorial optimization problem," *arXiv preprint arXiv:1710.11573,* 2017.

[18] D.-H. Lee, S. Zhang, A. Fischer and Y. Bengio, "Difference target propagation," in *Joint european conference on machine learning and knowledge discovery in databases*, 2015.

[19] M. Carreira-Perpinan and W. Wang, "Distributed optimization of deeply nested systems," in *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, Reykjavik, 2014.

[20] A. Choromanska, B. Cowen, S. Kumaravel, R. Luss, M. Rigotti, I. Rish, P. Diachille, V. Gurev, B. Kingsbury, R. Tejwani and others, "Beyond backprop: Online alternating minimization with auxiliary variables," in *International Conference on Machine Learning*, 2019.

[21] R. Ranganath, S. Gerrish and D. Blei, "Black box variational inference," in *Artificial intelligence and statistics*, 2014.

[22] G. Tucker, A. Mnih, C. J. Maddison, D. Lawson and J. Sohl-Dickstein, "Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models," *arXiv preprint arXiv:1703.07370,* 2017.

[23] W. Grathwohl, D. Choi, Y. Wu, G. Roeder and D. Duvenaud, "Backpropagation through the void: Optimizing control variates for black-box gradient estimation," *arXiv preprint arXiv:1711.00123,* 2017.

[24] M. Yin and M. Zhou, "ARM: Augment-REINFORCE-merge gradient for stochastic binary networks," *arXiv preprint arXiv:1807.11143,* 2018.

[25] W. Kool, H. van Hoof and M. Welling, "Buy 4 REINFORCE Samples, Get a Baseline for Free!," *Deep Reinforcement Learning Meets Structured Prediction Workshop at the International Conference on Learning Representations,* 2019.

[26] Z. Dong, A. Mnih and G. Tucker, "DisARM: An antithetic gradient estimator for binary latent variables," *arXiv preprint arXiv:2006.10680,* 2020.

[27] K. P. Murphy, Machine learning: a probabilistic perspective, MIT press, 2012.

[28] M. Filippone and R. Engler, "Enabling scalable stochastic gradient-based inference for Gaussian processes by employing the Unbiased LInear System SolvEr (ULISSE)," in *Proceedings of the 32nd International Conference on Machine Learning*, Lille, 2015.

[29] K. Cutajar, M. A. Osborne, J. P. Cunningham and M. Filippone, "Preconditioning Kernel Matrices," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, New York, NY, USA, 2016.

[30] K. Wang, G. Pleiss, J. Gardner, S. Tyree, K. Q. Weinberger and A. G. Wilson, "Exact Gaussian Processes on a Million Data Points," in *Advances in Neural Information Processing Systems*, 2019.

[31] C. Fowlkes, S. Belongie and J. Malik, "Efficient spatiotemporal grouping using the nystrom method," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 2001.

[32] Y. Cho and L. Saul, "Kernel methods for deep learning," *Advances in neural information processing systems*, vol. 22, 2009.

[33] J. Wacker, M. Kanagawa and M. Filippone, "Improved Random Features for Dot Product Kernels," *arXiv preprint arXiv:2201.08712*, 2022.

[34] I. Poli, J. Launay, K. Müller, G. Pariente, I. Carron, L. Daudet, R. Ohana and D. Hesslow, "Method and system for machine learning using optical data". USA Patent US 2021/0287079 A1, 16 September 2021.

[35] A. Nøkland, "Direct feedback alignment provides learning in deep neural networks," *arXiv preprint arXiv:1609.01596,* 2016.

[36] D. Dua and C. Graff, *UCI Machine Learning Repository*, 2017.

[37] S. Bartunov, A. Santoro, B. Richards, L. Marris, G. E. Hinton and T. Lillicrap, "Assessing the scalability of biologically-motivated deep learning algorithms and architectures," *Advances in neural information processing systems*, vol. 31, 2018.

[38] J. Launay, I. Poli and F. Krzakala, "Principled training of neural networks with direct feedback alignment," *arXiv preprint arXiv:1906.04554,* 2019.

[39] J. Fajtl, V. Argyriou, D. Monekosso and P. Remagnino, "Latent Bernoulli Autoencoder," in *International Conference on Machine Learning*, 2020.

[40] K. Berggren, Q. Xia and K. Likharev, "Roadmap on emerging hardware and technology for machine learning," *Nanotechnology*, vol. 31, no. 1, p. 012002, 2020.

[41] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer school on machine learning*, 2003.

[42] A. G. Wilson, Z. Hu, R. R. Salakhutdinov and E. P. Xing, "Stochastic variational deep kernel learning," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

Published by International Frequency Sensor Association, IFSA Publishing, S. L., 2019

(http://www.sensorsportal.com)