

# Combining Primal and Dual Representations in Deep Restricted Kernel Machines Classifiers

Francesco Tonin, Panagiotis Patrinos, and Johan A. K. Suykens

ESAT-STADIUS, KU Leuven, Heverlee 3001, Belgium.  
`{francesco.tonin, panos.patrinos, johan.suykens}@esat.kuleuven.be`

**Abstract.** In the context of deep learning with kernel machines, the deep Restricted Kernel Machine (DRKM) framework allows multiple levels of kernel PCA (KPCA) and Least-Squares Support Vector Machines (LSSVM) to be combined into a deep architecture using visible and hidden units. We propose a new method for DRKM classification coupling the objectives of KPCA and classification levels, with the hidden feature matrix lying on the Stiefel manifold. The classification level can be formulated as an LSSVM or as an MLP feature map, combining depth in terms of levels and layers. The classification level is expressed in its primal formulation, as the deep KPCA levels, in their dual formulation, can embed the most informative components of the data in a much lower dimensional space. The dual setting is independent of the dimension of the inputs and the primal setting is parametric, which makes the proposed method computationally efficient for both high-dimensional inputs and large datasets. In the experiments, we show that our developed algorithm can effectively learn from small datasets, while using less memory than the convolutional neural network (CNN) with high-dimensional data. and that models with multiple KPCA levels can outperform models with a single level. On the tested larger-scale datasets, DRKM is more energy efficient than CNN while maintaining comparable performance.

**Keywords:** Kernel methods · Manifold learning · Primal-Dual Representations.

## 1 Introduction

The deep Restricted Kernel Machine (DRKM) was introduced in [18] to find synergies between kernel methods and deep learning. While recent research has focused on leveraging RKM in generative models [17,16] and unsupervised learning [21,20,22], there has been little investigation into the use of DRKM for classification problems. The *single*-level RKM classification framework was used in [10] to propose a multi-modal classification method based on tensor learning using a single model weight tensor with all the modes sharing a common latent space. The *semi*-supervised RKM was used in conjunction with message-passing kernel PCA (KPCA) levels in [1] for semi-supervised node classification in graphs using *only* the dual representation. In [18], deep reduced set kernel-based models were

applied to classification datasets, and an unconstrained DRKM with a primal-estimation scheme using a Least-Squares Support Vector Machines (LSSVM) classifier was evaluated on two UCI datasets. However, its unconstrained formulation introduces additional stabilization terms in the objective and does not combine primal and dual in the same model. A DRKM for multi-level kernel PCA (DKPCA) was introduced in [22] as a method for unsupervised representation learning employing multiple levels of kernel PCA with orthogonality constraints. However, this method does not consider a classification objective and only employs the dual representation. Overall, effective classification algorithms for deep (i.e., with multiple levels) RKMs are still an open problem and previous studies have only considered either primal or dual models.

In this paper, we propose a new DRKM classifier employing multiple levels of KPCA and an LSSVM or MLP classification level. We combine the dual representation of the KPCA with the primal representation of the LSSVM/MLP: the dual representation is better suited for the kernel trick and feature extraction, while the primal representation is more computationally efficient for larger datasets and can directly take advantage of the nonlinear embedding from the multiple KPCA levels. Given that the size of the kernel matrix is independent of the dimension of the inputs, the dual problem is typically suited for handling problems with high-dimensional input data, while solving the primal is often suitable for large numbers of data. Combining the parametric primal representation and the kernel-based dual in the DRKM provides a powerful framework for efficient algorithms for both high-dimensional inputs and larger-scale datasets. Therefore, the model is well-suited for Frugal AI, where the non-parametric levels can handle high-dimensional data with computational resources that are independent of the number of inputs, possibly reducing the dimensionality of the input space, such that the parametric classification level can be efficiently trained in a much lower dimensional space. Note that we employ the dual framework for the unsupervised KPCA levels and the primal framework for the supervised classification level. Employing unsupervised levels is advantageous when training with limited labelled data, as the model is built on unsupervised core models that can effectively induce patterns and representations from the unlabelled data. Our main contributions are as follows.

- We propose a new method for DRKM classification that couples the objectives of KPCA and LSSVM/MLP classification levels to create a deep architecture. Our method integrates the kernel-based dual representation in the unsupervised part and the parametric primal representation in the supervised part. In this way, the final model is applicable to general data in the sense that the DRKM can be applied to limited and larger amounts of data in either low or high dimensional inputs spaces.
- We define a constrained optimization problem with orthogonality constraints on the hidden features of the unsupervised levels and illustrate a training algorithm based on Projected Gradient Descent. In particular, we investigate how end-to-end training compares with unsupervised initialization of the KPCA levels followed by fine-tuning.

- We empirically evaluate our method on benchmark datasets comparing with the LSSVM/MLP classifiers alone, showing that increased depth can benefit performance in kernel methods when limited training data is available. Our approach is frugal in that it can effectively learn from limited training data thanks to its unsupervised KPCA core, and it can also efficiently deal with larger datasets thanks to the primal classification level. We additionally empirically show lower environmental impact than the CNN on a number of UCI datasets with thousands of dimensions, thanks to the employed non-parametric dual representation, with similar classification performance.

## 2 Background

Consider the training data  $\{(x_i, y_i)\}_{i=1}^N$ , where  $x_i \in \mathbb{R}^d$  and  $y_i \in \{-1, 1\}$ . The objective of RKM binary classification [18] is:

$$J_0 = \sum_{i=1}^N (1 - y_i (\varphi(x_i)^T w + b)) h_i - \frac{\lambda}{2} h_i^2 + \frac{\eta}{2} w^T w, \quad (1)$$

where  $\varphi(x) \in \mathbb{R}^l$  is the feature map,  $w \in \mathbb{R}^l$  is the weight vector,  $h_i \in \mathbb{R}$  are the hidden features, and  $\lambda, \eta > 0$  are regularization constants. It can be shown that (1) is a lower bound to the LSSVM classification objective [19,18]. In the shallow case where there is no other level, the solution of (1) in the conjugated features  $h_i$  is given by a linear system obtained from the stationarity of  $J_0$  [18].

Regarding the KPCA level, let  $s$  be the number of selected principal components. In the LS-SVM setting, the KPCA problem can be written as minimizing a regularization term and finding directions of maximum variance [19]:

$$\begin{aligned} \underset{W, e_i}{\text{minimize}} \quad & \tilde{J}_{\text{KPCA}} = \frac{\eta}{2} \text{Tr}(W^T W) - \frac{1}{2} \sum_{i=1}^N e_i^T \Lambda^{-1} e_i \\ \text{subject to} \quad & e_i = W^T \varphi(x_i), \quad i = 1, \dots, N, \end{aligned}$$

where  $W \in \mathbb{R}^{l \times s}$  is the interconnection matrix,  $e_i \in \mathbb{R}^s$  are the score variables along the selected  $s$  projection directions, and  $\Lambda = \text{diag}\{\lambda_1, \dots, \lambda_s\} \succ 0, \eta > 0$  are regularization hyperparameters. The RKM formulation of KPCA [18] is given by an upper bound of  $\tilde{J}_{\text{KPCA}}$  obtained component-wise with the Fenchel-Young inequality  $\frac{1}{2\lambda} e^2 + \frac{\lambda}{2} h^2 \geq eh, \forall e, h \in \mathbb{R}$  which introduces the hidden features  $h$  and leads to the following objective with *conjugate feature duality*:

$$J_{\text{KPCA}} = - \sum_{i=1}^N \varphi(x_i)^T W h_i + \frac{1}{2} \sum_{i=1}^N h_i^T \Lambda h_i + \frac{\eta}{2} \text{Tr}(W^T W), \quad (2)$$

where  $h_i \in \mathbb{R}^s$  are the conjugated hidden features corresponding to each training sample  $x_i$ ; in representation learning,  $h_i$  is also known as the latent representation of  $x_i$  consisting of  $s$  latent variables or of  $s$  hidden features. By characterizing the stationary points of  $J_{\text{KPCA}}$  in (2), the following eigenvalue problem is

obtained

$$\frac{1}{\eta}KH = HA,$$

where  $K \in \mathbb{R}^{N \times N}$  denotes the kernel matrix induced by the positive-definite kernel function  $k : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$  with  $k(x_i, x_j) = \varphi(x_i)^\top \varphi(x_j)$  and the matrix  $H = [h_1, \dots, h_N]^\top$  incorporates the conjugate hidden features for all  $N$  data points. Since the primal form in (2) is not directly suitable for minimization because it is unbounded below, [21] derived a rewriting of (2) in the dual by introducing orthogonality constraints on the hidden features and eliminating the interconnection matrix. Using the stationarity of  $J_{\text{KPCA}}$  and the kernel trick, the first term of (2) can be rewritten as  $-\frac{1}{\eta} \text{Tr}(H^\top KH)$ , and the third term can be rewritten as  $\frac{1}{2\eta} \text{Tr}(H^\top KH)$ . Derivation details are given in [21]. A deep KPCA (DKPCA) framework combining multiple KPCA levels in a single objective was proposed in [22]. In that work, the principal components of multiple KPCA levels are coupled in their hidden units creating both forward and backward dependency across levels, showing greater representation efficiency for unsupervised learning. In this work, we propose a new method for classification based on a DKPCA architecture coupled with a classification level with end-to-end training combining dual and primal representations.

## 2.1 Related Work

Deep kernel learning tackles multiple latent spaces for greater flexibility, more informative hierarchical investigation of the data, and kernel-based interpretations. [14] considers the representation learned by successive network layers. In particular, they consider a deep neural network (DNN) of  $L$  layers  $f(x) = f_L \circ \dots \circ f_1(x)$  and analyze the induced kernels  $k_0 = k_{\text{RBF}}(x, x')$ ,  $k_1 = k_{\text{RBF}}(f_1(x), f_1(x'))$ ,  $\dots$ ,  $k_L = k_{\text{RBF}}(f_L \circ \dots \circ f_1(x), f_L \circ \dots \circ f_1(x'))$ , with  $k_{\text{RBF}}$  indicating the Gaussian kernel. In other words, each layer of the DNN is associated with a kernel defined on the output of that layer. Conversely, in this paper we consider several feature maps over multiple levels. In other words, following the terminology used in [18], in [14] deep learning is only performed over *layers*, while in our approach depth is given by multiple *levels*, each associated with a different feature map possibly consisting of multiple layers. For instance, a DRKM can consist of multiple DNNs, each associated with the feature map  $\varphi_j$  of level  $j$ . Therefore, in the framework of DRKM, [14] performs shallow learning, as it works with a single DNN. Additionally, in the DRKM the levels are coupled in terms of their hidden features, while in [14] the considered architecture is a composition of the outputs of each layer. A concatenation of operator-valued kernel layers was considered for data autoencoding in [12], but no extension to supervised learning was considered. In [6], shallow PCA is conducted to extract principal components, which are then applied to another KPCA, where each KPCA independently and sequentially optimizes its variance maximization. PCA is firstly performed to extract principal components of the data and then further dimensionality reduction is sequentially applied to the extracted features from the

previous (K)PCA layer. This serial approach makes each layer straightforwardly maximize its variance objective, which is independent of other layers.

### 3 Proposed Method

In this section, we describe the proposed combination of KPCA levels in the dual and a classification level in the primal. The former part extracts multiple levels of the most informative components of the given data through multiple feature maps. After such transformations, the classification level can effectively learn a decision boundary in a much lower dimensional subspace, motivating its representation in the primal. We start by describing the model formulation where the primal classifier is an LSSVM classifier following one or more KPCA levels. Next, we derive the model for the MLP classifier. Finally, we discuss the optimization algorithm and multiple initialization procedures.

#### 3.1 Shallow RKM - Primal LSSVM Classifier

We start by defining a DRKM classifier with a single KPCA level, i.e., the shallow RKM. Its architecture is the following.

- Level 1 consists of KPCA using as input the sample  $x_i$ . The features extracted by this level are characterized by its hidden features  $h_i^{(1)}$ .
- Level 2 consists of LSSVM classification using as input the hidden features  $h_i^{(1)}$  from the previous level and with output data  $y_i$ . This level is characterized by its weights  $w$ .

The training objective of the above architecture is obtained from the representation learning optimization problem of RKM with one KPCA level and an LSSVM classifier written in its RKM formulation. The solution in terms of the  $h_i$  of the latter is given by a linear system in the shallow case; however, in the case of multiple levels, end-to-end learning cannot be performed by solving the linear system. We propose an alternative training strategy for the deep case. First of all, we eliminate  $h_i$  in (1): from the stationarity of  $J_0$  it follows that

$$h_i = \frac{1}{\lambda} (1 - y_i (w^T \varphi(x_i) + b)). \quad (3)$$

Replacing (3) in (1),

$$J_0 = \sum_{i=1}^N \frac{1}{2\lambda} (1 - y_i (w^T \varphi(x_i) + b))^2 + \frac{\eta}{2} w^T w.$$

We propose to combine one level of KPCA in the RKM formulation with the objective of an LSSVM classifier in the following optimization problem:

$$\begin{aligned}
\min_{h_i^{(1)}, w, b} \quad & J_1 = -\frac{1}{2\eta_1} \text{Tr} \left( H^{(1)T} K^{(0)} H^{(1)} \right) + \frac{\eta}{2} w^T w \\
& + \frac{1}{2\lambda} \sum_{i=1}^N \left( 1 - y_i (w^T \varphi(h_i^{(1)}) + b) \right)^2 \\
\text{s.t.} \quad & H^{(1)T} H^{(1)} = I.
\end{aligned} \tag{4}$$

The objective of (4) consists of a KPCA term written in the dual formulation and a classification term in the primal. The former problem is independent of the input dimensionality due to the dual formulation; we use the primal formulation for the latter term because  $h_i^{(1)}$  is usually low-dimensional, as the number of principal components  $s_1$  is generally small. The classifier is expressed in the primal as  $\hat{y} = \text{sign}(w^T \varphi(h^{(1)}) + b)$ , where  $\hat{y}$  is the estimated label of an input sample  $x$  and  $h_i$  is computed following (3) with input  $h^{(1)}$ , which is the latent representation of  $x$ . Regarding the choice of  $\varphi$ , one can employ a linear mapping as often done in the last layer of deep neural network classifiers; the full model is still non-linear due to the first KPCA level. Moreover, the formulation in (4) is intended for binary classification problems; multiclass problems with  $p$  classes can be solved with a one-vs-all approach by training  $p$  binary classifiers.

*Remark 1 (Orthogonality constraints).* Contrary to the classifier proposed in [18], we use orthogonality constraints on the hidden features of the kernel PCA level. In the field of neural networks, it was shown that orthogonality constraints can lead to better performance [2] and better regularization [5,4]. For example, [4] shows that, if the weights of the neural network are points of the Grassmann manifold  $\mathcal{G}(1, n)$ , minimizing the soft orthogonality constraint  $L(\alpha, W) = \frac{\alpha}{2} \|W^T W - I\|_F^2$ , where  $W \in \mathbb{R}^{n \times p}$  is the weight matrix of the neural network, also minimizes a model complexity loss, introduced in [7], based on the KL divergence  $D_{\text{KL}}(Q(w|\beta) \| P(w|\alpha))$  between the posterior  $Q(w|\beta)$  and the prior  $P(w|\alpha)$  distribution of the neural network weights. In contrast to [4], our method is based on kernel methods.

### 3.2 Deep RKM - Primal LSSVM Classifier

We now define the DRKM classifier with  $n_{\text{levels}}$  KPCA levels: increased classifier depth may lead to a more compact representation of the target function and thus to better performance for complex classification problems. The architecture, illustrated in Fig. 1, is as follows.

- Level 1 consists of KPCA using as input the sample  $x_i$ . The features extracted by this level are characterized by its hidden features  $h_i^{(1)}$ .

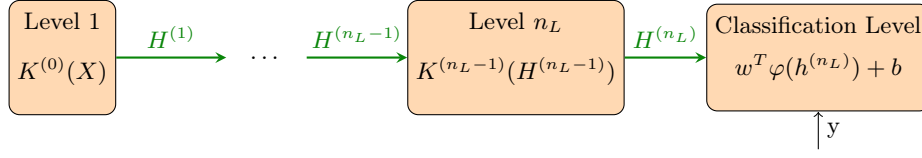


Fig. 1: Diagram of the proposed DRKM of (5) with  $n_L$  KPCA levels in dual and one final LSSVM classification level in primal. Each green arrow goes from the level that is characterized by the corresponding hidden features to the level where it is used as input.  $K^{(0)}(X)$  indicates the kernel matrix of the input data.

- Level  $j$ ,  $2 \leq j \leq n_{\text{levels}}$  consists of KPCA using as input the hidden features  $h_i^{(j-1)}$  from the previous level. The features extracted by this level are characterized by its hidden features  $h_i^{(j)}$ .
- Level  $n_{\text{levels}} + 1$  consists of LSSVM classification using as input the hidden features  $h_i^{(n_{\text{levels}})}$  from the previous level and with output data  $y_i$ . This level is characterized by its weights  $w$ .

The proposed objective to be minimized for  $n_{\text{levels}} \geq 2$  of KPCA levels is:

$$\begin{aligned}
 J_{n_{\text{levels}}}(h_i^{(1)}, \dots, h_i^{(n_{\text{levels}})}, w, b) = & -\frac{1}{2\eta_1} \text{Tr} \left( H^{(1)T} K^{(0)} H^{(1)} \right) \\
 & - \sum_{j=2}^{n_{\text{levels}}} \frac{1}{2\eta_j} \text{Tr} \left( H^{(j)T} K^{(j-1)}(H^{(j-1)}) H^{(j)} \right) \\
 & + \frac{1}{2\lambda} \sum_{i=1}^N \left( 1 - y_i (w^T \varphi(h_i^{(n_{\text{levels}})}) + b) \right)^2 + \frac{\eta}{2} w^T w,
 \end{aligned} \tag{5}$$

where  $H^{(j)} = [h_1^{(j)}, \dots, h_N^{(j)}]^T \in \mathbb{R}^{N \times s_j}$  incorporates the hidden features conjugated along  $s_j$  projection directions for all  $N$  data points, where  $s_j$  is the number of selected principal components by the  $j$ -th level. The kernel matrices are obtained as follows:  $K^{(0)} \in \mathbb{R}^{N \times N}$  is attained as  $(K^{(0)})_{ik} = k^{(0)}(x_i, x_k)$  and  $K^{(j-1)} \in \mathbb{R}^{N \times N}$  as  $(K^{(j-1)})_{ik} = k^{(j-1)}(h_i^{(j-1)}, h_k^{(j-1)})$ , where  $k^{(0)} : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$  is the kernel function of the first level and  $k^{(j-1)} : \mathbb{R}^{s_{j-1}} \times \mathbb{R}^{s_{j-1}} \mapsto \mathbb{R}$  is the kernel function of level  $j = 2, \dots, n_{\text{levels}}$ . The projection directions of shallow KPCA are uncorrelated due to the orthogonality of different principal components; similarly, we impose  $H^{(j)T} H^{(j)} = I$  in (5). Following [21], orthogonality between the hidden features of two different levels can be imposed to encourage the levels to learn new features of the data instead of repeating the same features in every level. The number of levels is an hyperparameter to be tuned in addition to the kernel hyperparameters and the number of components in each level. One can tune these hyperparameters using standard techniques such as defining a separate validation set or employing cross-validation. In general, the minimum number of levels required for good performance on a given set of observations

depends on the complexity of the training dataset. For an out-of-sample  $x^*$ , multiple strategies to compute its latent representation have been proposed in the context of RKMs [18,15,21,22]. Based on [17], we propose to apply a kernel smoother approach [9]  $h^{(n_{\text{levels}})*} = \frac{\sum_{i=1}^N \tilde{k}(x_i, x^*) h_i^{(n_{\text{levels}})}}{\sum_{i=1}^N \tilde{k}(x_i, x^*)}$ , where  $\tilde{k}$  is a localized kernel such as the Gaussian kernel with design parameter  $\tilde{\sigma}$ , representing the width of the considered local neighborhood. A larger  $\tilde{\sigma}$  gives lower variance, as more observed points are considered, but also higher bias. With smaller bandwidth, the similarity measure expressed by  $\tilde{k}$  is more local, so  $h^{(n_{\text{levels}})*}$  is closer to the latent representation of training points that are similar to  $x^*$ . Using this approach, the predicted class of  $x^*$  expressed in the primal model representation is:

$$\hat{y}^* = \text{sign} \left[ w^T \varphi \left( \frac{\sum_{i=1}^N \tilde{k}(x_i, x^*) h_i^{(n_{\text{levels}})}}{\sum_{i=1}^N \tilde{k}(x_i, x^*)} \right) + b \right]. \quad (6)$$

Note that  $h_i^{(n_{\text{levels}})}$  depends on the hidden features of the previous levels  $h_i^{(n_{\text{levels}}-1)}, \dots, h_i^{(1)}$  as they are jointly optimized in (5). The final architecture of the proposed DRKM is therefore a combination of non-parametric modelling through the hidden features of the last KPCA level computed through (6) and the parametric classifier with parameters  $w, b$ .

### 3.3 Deep RKM - Primal MLP Classifier

The classification DRKM can also be constructed with multiple KPCA levels in the RKM formulation and a final MLP classification level, instead of the LSSVM classifier described in the previous subsection. The MLP classifier is a single level that consists of a multi-layer feature map. Note here the difference between depth given by multiple KPCA levels and depth in the MLP: in the latter case deep learning is only performed over layers, while in the former case depth is given by multiple levels, each associated with a different feature map possibly consisting of multiple layers.

The training objective of the DRKM with a final MLP  $f_\theta(h_i^{(n_{\text{levels}})}) : \mathbb{R}^{s_{n_{\text{levels}}}} \rightarrow \mathbb{R}^p$  classification level with  $p$  classes is:

$$\begin{aligned} J(h_i^{(1)}, \dots, h_i^{(n_{\text{levels}})}, W, b, \theta) = & -\frac{1}{2\eta_1} \text{Tr} \left( H^{(1)T} K^{(0)} H^{(1)} \right) \\ & - \sum_{j=2}^{n_{\text{levels}}} \frac{1}{2\eta_j} \text{Tr} \left( H^{(j)T} K^{(j-1)} (H^{(j-1)}) H^{(j)} \right) \\ & + \frac{1}{2\lambda N} \sum_{i=1}^N \mathcal{L}(f_\theta(h_i^{(n_{\text{levels}})}), y_i) + \frac{\eta}{2} \text{Tr} W^T W, \end{aligned} \quad (7)$$

where  $\mathcal{L}(\hat{y}_i, y_i) = -\sum_{k=1}^p y_{i,k} \log \frac{\exp \hat{y}_{i,k}}{\sum_{j=1}^p \exp \hat{y}_{i,j}}$  is the cross entropy loss with  $\hat{y}_i = f_\theta(h_i^{(n_{\text{levels}})})$  and  $y_{i,k}$  a binary indicator of input  $x_i$  belonging to class  $k$ .

The logits predicting the class of an out-of-sample point  $x^*$  are given by  $f_\theta(h^{(n_{\text{levels}})})$ , where  $h^{(n_{\text{levels}})}$  is obtained through the kernel smoother approach. One motivation to employ the MLP classification level is that the MLP approach also naturally handles the multi-class case, resulting in multiple logits, one for each class. The next subsection describes the optimization algorithm for the DRKM classifier consisting of multiple KPCA levels and a final LSSVM or MLP classification level.

### 3.4 Optimization

The nonlinear optimization problem with objective (5) or (7) has at least one global minimum due to the Weierstrass theorem, as the objective function is continuous and the feasible set is compact, and it may have multiple local minima since it is a non-convex problem. The constraint set is a Stiefel manifold  $\text{St}(s, N)$ , so one of the algorithms that have been proposed for optimization on the Stiefel manifold could be employed. For instance, one could exploit the Cayley transform to determine the search curve, such as in the algorithms proposed in [23,24]. However, as explained in [21], these methods could be numerically problematic because determining the search curve requires a matrix inversion at each iteration. Therefore, we propose to employ the Projected Gradient Descent (PGD) algorithm, which specifies an iterative algorithm projecting  $H$  onto the Stiefel manifold at each iteration  $k$ . The iterates for minimizing (5) are specified by

$$\begin{aligned} H^{k+1} &= \Pi_{\text{St}(s, N)}(H^k - \alpha_k \nabla J_2(H^k, w^k, b^k)), \\ w^{k+1} &= w^k - \alpha_k \nabla J_{n_{\text{levels}}}(H^k, w^k, b^k), \\ b^{k+1} &= b^k - \alpha_k \nabla J_{n_{\text{levels}}}(H^k, w^k, b^k), \end{aligned}$$

where  $\Pi_{\text{St}(s, N)}$  is the Euclidean projection onto the Stiefel manifold and  $\alpha_k$  is the stepsize selected via backtracking. This projection is computed using the compact SVD of  $H^k$ , while the weights  $w$  and bias  $b$  of the LSSVM classifier level are not projected onto the Stiefel manifold. The objective (7) with MLP classifier is also optimized with PGD, where the weights of the MLP are not projected on the Stiefel manifold but are trained using Adam [11].

The variables can be initialized randomly from the standard normal distribution and *end-to-end* training or with unsupervised initialization with *fine-tuning*. In the unsupervised initialization scheme, the KPCA levels are first trained in the DKPCA unsupervised setting [22] by considering only the unsupervised terms of the objective (5); after convergence, the full model is fine-tuned using labels.

## 4 Experimental Evaluation

In this section, we conduct numerical experiments on standard benchmark datasets to evaluate the DRKM classifier and explore its properties. Specifically, we focus on efficient learning for small and large datasets in low- and high-dimensions.

Table 2: Mean classification accuracy (%) on the test set of MNIST according to the number of KPCA levels. Each column corresponds to a training set size.

| Method  | $N = 50$            | $N = 100$           | $N = 250$           | $N = 500$           | $N = 750$           | $N = 1000$          |
|---------|---------------------|---------------------|---------------------|---------------------|---------------------|---------------------|
| 1-level | 57.27 (5.75)        | 59.76 (4.51)        | 71.24 (5.74)        | 75.15 (7.47)        | 79.68 (7.01)        | 80.59 (7.64)        |
| 2-level | <b>61.05</b> (1.24) | <b>65.97</b> (1.03) | <b>75.74</b> (0.50) | 81.11 (1.22)        | 83.89 (0.23)        | <b>85.29</b> (0.45) |
| 3-level | 61.02 (1.23)        | 65.64 (1.70)        | 75.69 (0.49)        | <b>81.35</b> (0.43) | <b>83.90</b> (0.20) | <b>85.29</b> (0.69) |

Table 3: Classification accuracy (%) in the small-data regime.

| Method                                            | MNIST        | ARCENE       | Sonar        | Protein      |
|---------------------------------------------------|--------------|--------------|--------------|--------------|
| MLP                                               | 87.33        | 74.67        | 77.85        | 57.29        |
| CNN                                               | <b>95.80</b> | 80.33        | 85.71        | 59.87        |
| LSSVM                                             | 91.75        | 79.67        | 88.09        | 57.30        |
| DRKM (DKPCA initialization, no fine-tuning)       | 92.72        | 79.67        | 81.95        | 60.13        |
| DRKM (DKPCA initialization, with fine-tuning)     | 92.73        | 82.00        | 84.71        | 61.24        |
| DRKM (end-to-end training, random initialization) | 92.78        | <b>82.33</b> | <b>90.27</b> | <b>61.48</b> |

#### 4.1 Experimental Setup

We compare our method with a standard LSSVM classifier with RBF kernel, and a multilayer perceptron (MLP), and a Convolutional Neural Network (CNN). We test on MNIST [13], two UCI datasets, ARCENE [8] and Sonar, on the bioinformatics dataset Protein, on the RCV1 dataset [3], and on the IMDB Drama dataset.

To assess that the proposed DRKM can effectively learn from small amount of training data, we chose a train/test split with few training samples; details are given in Table 1, where  $N$  and  $N_{\text{test}}$  are the number of used training and test instances, respectively,  $d$  the input dimension, and  $p$  the number of classes. We train all models for a maximum of 100 iterations. In the DRKM objective, we fix  $\eta = \eta_j = 1$ ,  $j = 1, \dots, n_{\text{levels}}$  and  $\lambda = 0.5$ . The MLP has one hidden layer with 10 neurons and ReLU activation. For the CNN, we employ 2D convolution for 2-dimensional datasets and 1D convolution for 1D datasets.

Table 1: Tested datasets.

| Dataset | $N$   | $N_{\text{test}}$ | $d$            | $p$ |
|---------|-------|-------------------|----------------|-----|
| MNIST   | 4000  | 10000             | $28 \times 28$ | 10  |
| ARCENE  | 112   | 60                | 10000          | 2   |
| Sonar   | 166   | 42                | 60             | 2   |
| Protein | 1489  | 13406             | 357            | 3   |
| RCV1    | 16193 | 4049              | 47236          | 2   |
| IMDB    | 96735 | 24184             | 1002           | 2   |

#### 4.2 Experimental Results

**Do deeper models perform better?** The classification accuracy on the test set of MNIST attained by a DRKM classifier with one, two, and three KPCA levels and one linear LSSVM classification level for multiple  $N$  (number of training

points) is shown in Table 2. All models use the same overall number of components:  $s_1 = 6$  for 1-level,  $s_1 = 3$  and  $s_2 = 3$  for 2-level,  $s_1 = s_2 = s_3 = 2$  for 3-level. The variance is due to random initialization and different selection of the subset of training points. The mean accuracy of the two-level model is higher than that for the one-level model across all  $N$ , suggesting that the increased depth led to better generalization. For instance, for  $N = 500$  the mean performance of the one-level model is 75.15%, while for the two-level model it is 81.11%. The three-level model performs similarly to the two-level model. The three-level model keeps only two principal components per level, which may not be enough to get improved results. Furthermore, a deeper architecture may be useful on datasets more difficult than MNIST with, for instance, multiple more realistic objects and a complex background, where greater model complexity could boost classification accuracy. In general, the variance of the model with one level is significantly larger than that of the models with two and three levels, which means that different initializations have a greater influence on the model with the lowest depth. In conclusion, classification performance can benefit from architectures with multiple levels.

| Method | Protein (full) | RCV1         | IMDB         |
|--------|----------------|--------------|--------------|
| MLP    | 63.01          | 93.38        | 97.88        |
| CNN    | 67.33          | <b>94.83</b> | <b>98.88</b> |
| LSSVM  | 58.33          | 94.27        | 97.78        |
| DRKM   | <b>67.61</b>   | <b>94.83</b> | <b>98.88</b> |

Fig. 2: Classification accuracy (%) for larger-scale data.

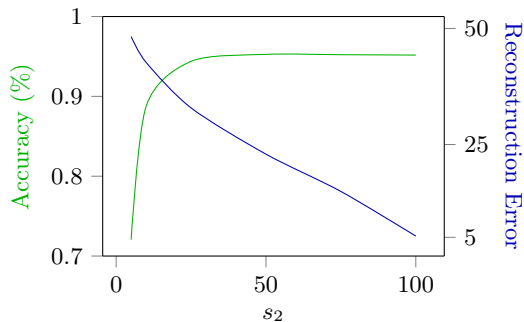


Fig. 3: Test classification accuracy (%) and training reconstruction error of DRKM on MNIST with varying  $s_2$ .

**How does the proposed method perform compared to related techniques?** Table 3 compares the classification accuracy (%) of a two-level DRKM with  $s_1 = s_2 = 10$  with MLP classification level against an MLP classifier, a CNN, and a standard nonlinear LSSVM classifier. Results are averaged over five runs. The two-level DRKM outperforms the LSSVM classifier on all datasets, showing that increased depth and end-to-end training can result in better performance. In general, the experiments suggest that the proposed model is well suited for both low-dimensional datasets such as Sonar, as the KPCA levels can derive a higher number of features than the dimension of the input dataset, and high-dimensional datasets such as ARCENE, because the formulations in

Table 4: Energy consumption in Wh when training with limited data.

| Method                                            | MNIST  | ARCENE | Sonar   | Protein  |
|---------------------------------------------------|--------|--------|---------|----------|
| MLP                                               | 0.0075 | 0.0007 | 0.001   | 0.000002 |
| CNN                                               | 0.1540 | 0.3746 | 0.036   | 0.00016  |
| LSSVM                                             | 0.1304 | 0.0002 | 0.00001 | 0.000008 |
| DRKM (DKPCA initialization, no fine-tuning)       | 0.0610 | 0.0006 | 0.0009  | 0.000007 |
| DRKM (DKPCA initialization, with fine-tuning)     | 1.177  | 0.0044 | 0.021   | 0.0012   |
| DRKM (end-to-end training, random initialization) | 0.316  | 0.0051 | 0.016   | 0.00026  |

(5) and (7) do not depend on the dimension  $d$  of the input space thanks to the combination of the dual and primal representations. This is not the case for the CNN, which achieves significantly better performance only with spatial data such as the MNIST 2D images. Table 2 evaluates DRKM with end-to-end training on larger datasets, where *Protein (full)* indicates the Protein dataset with  $N = 11916$  training points. The results show that the proposed method can be applied effectively to both small-scale datasets and larger datasets, for both lower-dimensional data (Protein) and higher-dimensional data (RCV1, IMDB). To further evaluate the performance on high-dimensional data, we train DRKM on MNIST for varying number of components  $s_2$  keeping fixed  $s_1 = 100$  with cosine kernel. The test classification performance and the training reconstruction error (MSE) are shown in Fig. 3. It can be seen that an initial sharp increase in accuracy corresponds to a distinctive drop in reconstruction error; further increasing  $s_2$  leads to a decrease in training reconstruction error but small improvement in test accuracy, showing that small values of  $s_2$  are enough to achieve good performance on MNIST.

**What is the effect of fine-tuning?** We compare end-to-end training to fine-tuning after DKPCA unsupervised initialization in Table 3. In addition, we consider a linear classifier trained on the deep features extracted by DKPCA [22], i.e., a DRKM with DKPCA initialization but no fine-tuning and a linear classification level. On Sonar and ARCENE, fine-tuning the hidden units initialized using unsupervised DKPCA initialization results in significantly higher performance compared to simply training a linear classifier on the feature extracted by DKPCA. Note that while random initialization give unfeasible initial points, the unsupervised initialization gives a feasible point. On the other hand, on the more complex MNIST dataset, DKPCA initialization already performs quite well compared to the other initialization schemes. While on MNIST the proposed optimization algorithm converges to a similar minimum when the initial  $h_i^{(1)}, h_i^{(2)}$  are found by DKPCA, on ARCENE end-to-end training with random initialization considerably outperforms the other initialization schemes, showing that the effect of the initialization schemes is dependent on the dataset at hand.

Table 5: Efficiency comparisons on running time and memory consumption on the limited data benchmarks where training time (s) and peak training memory usage (MB) are given.

| Method | Time (s) |        |       |         | Memory (MB) |         |       |         |
|--------|----------|--------|-------|---------|-------------|---------|-------|---------|
|        | MNIST    | ARCENE | Sonar | Protein | MNIST       | ARCENE  | Sonar | Protein |
| MLP    | 1.57     | 0.22   | 0.03  | 0.53    | 12.29       | 22.07   | 2.88  | 8.38    |
| CNN    | 4.55     | 11.36  | 0.33  | 2.88    | 74.46       | 5127.45 | 86.27 | 289.12  |
| LSSVM  | 0.91     | 0.07   | 0.002 | 0.05    | 50.41       | 22.08   | 0.59  | 0.60    |
| DRKM   | 0.98     | 0.15   | 0.07  | 0.42    | 67.89       | 43.52   | 5.24  | 40.53   |

Table 6: Efficiency comparisons on running time and memory consumption on the larger-scale benchmarks.

| Method | Time (s)       |       |        | Memory (MB)    |        |         |
|--------|----------------|-------|--------|----------------|--------|---------|
|        | Protein (full) | RCV1  | IMDB   | Protein (full) | RCV1   | IMDB    |
| MLP    | 5.26           | 5.92  | 23.80  | 10.32          | 5.18   | 32.44   |
| CNN    | 25.79          | 99.18 | 372.64 | 326.43         | 720.43 | 700.12  |
| LSSVM  | 0.95           | 1.42  | 8.26   | 10.88          | 16.68  | 19.26   |
| DRKM   | 16.46          | 33.72 | 365.17 | 139.20         | 148.78 | 1343.54 |

**What is the environmental impact?** The experiments above show that the combination of unsupervised KPCA in dual and supervised level in primal is effective when training with limited data. In efficient AI, it is also important to consider computational efficiency and the corresponding environmental impact of Machine Learning algorithms. In Table 4, we report the energy consumption in the small-data regime in Wh tracked using CodeCarbon.io<sup>1</sup>. Overall, DRKM has comparable or lower energy consumption than CNN while showing higher performance on the 1D datasets. The evaluations on the larger-scale data are presented in Table 7, where DRKM shows lower energy consumption than CNN while maintaining competitive performance.

We evaluate the efficiency of DRKM by comparing in terms of running time and memory consumption with three commonly used methods: MLP, CNN, and LSSVM. The results are presented in Tables 5 and 6. Table 5 shows the efficiency comparisons when limited data is available, where we measured the training time (in seconds) and peak training memory usage (in megabytes). We can observe that

Table 7: Energy consumption (Wh) for larger-scale data.

| Method | Protein (full) | RCV1  | IMDB  |
|--------|----------------|-------|-------|
| MLP    | 0.020          | 0.029 | 0.076 |
| CNN    | 0.115          | 0.44  | 2.96  |
| LSSVM  | 0.0023         | 0.003 | 0.017 |
| DRKM   | 0.052          | 0.092 | 1.87  |

the combination of non-parametric and parametric levels makes it possible for the DRKM to achieve competitive

<sup>1</sup> <https://codecarbon.io/>

results, with significantly lower memory consumption compared to CNN. For instance, in the high-dimensional ARCENE datasets, the DRKM uses 43.52 MB compared to 5127.45 MB used by CNN. The significant advantage in memory usage is due to the dual KPCA levels, whose dimension is independent of the input dimensions. The KPCA levels can perform dimensionality reduction such that the parametric classification level in the DRKM operates from an input space of much lower dimension. This indicates that our DRKM method strikes a balance between computational efficiency and memory utilization, making it a favorable choice compared to the CNN for applications where data has thousands of dimensions but memory resources are limited. Table 6 presents the efficiency comparisons on larger-scale benchmarks. The results demonstrate that DRKM exhibits favorable efficiency characteristics, with lower training times compared to CNN. Moreover, despite the larger dataset sizes, DRKM maintains a reasonable memory consumption, outperforming CNN in this regard in all tested datasets except the largest one. Overall, the experimental results demonstrate that the DRKM approach shows competitive efficiency in terms of both energy consumption, running time and memory usage when compared to established methods, suggesting its potential as an effective and resource-efficient solution in Frugal AI applications.

## 5 Conclusion

We propose a DRKM classifier based on the DKPCA representation learning method in its dual representation combined with a primal classification level expressed as LSSVM or as MLP. Our proposed DRKM classifier exhibits several advantageous characteristics that make it well-suited for Frugal AI. Firstly, the integration of unsupervised KPCA levels in our model enables superior performance compared to both LSSVM and MLP/CNN when dealing with tabular classification datasets with limited training points. This indicates the ability of our model to extract meaningful features from scarce data, making it particularly useful in resource-constrained scenarios. Moreover, our study demonstrates that employing multiple KPCA levels in the DRKM classifier surpasses the performance of using a single KPCA level. This finding highlights the significance of incorporating multiple perspectives in representation learning, leading to enhanced classification accuracy. Finally, our method showcases versatility in handling datasets of varying input dimensions and number of samples. We show that DRKM has lower energy consumption than the CNN on a number of higher-dimensional UCI datasets thanks to the employed dual representation, whose dimension is independent of the number of inputs. The combination of dual and primal components of the DRKM allows for efficient learning across different data settings, as the kernel-based levels are better suited for high-dimensional data and the parametric classification level is suitable for larger number of data. Overall, the above findings demonstrate the suitability of the DRKM classifier for Frugal AI applications, where limited resources and diverse data dimensions are common challenges. By leveraging its dual and primal components, our model

demonstrates promising capabilities in achieving good performance and efficient training in Frugal AI scenarios. Future work can investigate the performance of the DRKM classifier with fixed-size methods [19] to further improve its scalability on very large-scale datasets with millions of samples, such as ImageNet. For computer vision tasks, it would also be interesting to explore the application of specific kernels to enhance its performance on image data.

## Acknowledgements

This work is jointly supported by ERC Advanced Grant E-DUALITY (787960), iBOF project Tensor Tools for Taming the Curse (3E221427), Research Council KU Leuven: Optimization framework for deep kernel machines C14/18/068, KU Leuven Grant CoE PFV/10/002, and Grant FWO G0A4917N, EU H2020 ICT-48 Network TAILOR (Foundations of Trustworthy AI - Integrating Reasoning, Learning and Optimization), and the Flemish Government (AI Research Program), and Leuven.AI Institute. This work was also supported by the Research Foundation Flanders (FWO) research projects G086518N, G086318N, and G0A0920N; Fonds de la Recherche Scientifique — FNRS and the Fonds Wetenschappelijk Onderzoek — Vlaanderen under EOS Project No. 30468160 (SeLMA).

## References

1. Achten, S., Tonin, F., Patrinos, P., Suykens, J.A.K.: Semi-supervised classification with graph convolutional kernel machines. *arXiv* (2023)
2. Bansal, N., Chen, X., Wang, Z.: Can we gain more from orthogonality regularizations in training deep CNNs? In: *NeurIPS* (2018)
3. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)* **2**(3), 1–27 (2011)
4. Cho, M., Lee, J.: Riemannian approach to batch normalization. In: *NeurIPS* (2017)
5. Cogswell, M., Ahmed, F., Girshick, R., Zitnick, L., Batra, D.: Reducing Overfitting in Deep Networks by Decorrelating Representations. In: *ICLR* (2016)
6. Deng, X., Tian, X., Chen, S., Harris, C.J.: Deep Principal Component Analysis Based on Layerwise Feature Extraction and Its Application to Nonlinear Process Monitoring. *IEEE Transactions on Control Systems Technology* **27**(6), 2526–2540 (2019). <https://doi.org/10.1109/TCST.2018.2865413>
7. Graves, A.: Practical variational inference for neural networks. In: *NeurIPS* (2011)
8. Guyon, I., Gunn, S.R., Ben-Hur, A., Dror, G.: Result Analysis of the NIPS 2003 Feature Selection Challenge. In: *NeurIPS* (2004)
9. Hastie, T., Tibshirani, R., Friedman, J.: *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media (2009)
10. Houthuys, L., Suykens, J.A.: Tensor-based restricted kernel machines for multi-view classification. *Information Fusion* **68**, 54–66 (2021)
11. Kingma, D.P., Ba, J.L.: Adam: A method for stochastic optimization. In: *ICLR* (2015)
12. Laforgue, P., Cl  men  on, S., d’Alch  -Buc, F.: Autoencoding any data through kernel autoencoders. In: *International Conference on Artificial Intelligence and Statistics*. PMLR (2019)
13. LeCun, Y., Cortes, C., Burges, C.: MNIST handwritten digit database. <http://yann.lecun.com/exdb/mnist> (2010)
14. Montavon, G., Braun, M.L., M  ller, K.R.: Kernel analysis of deep networks. *Journal of Machine Learning Research* **12**(9) (2011)

15. Pandey, A., Schreurs, J., Suykens, J.A.K.: Robust Generative Restricted Kernel Machines using Weighted Conjugate Feature Duality. In: LOD (2020)
16. Pandey, A., Schreurs, J., Suykens, J.A.K.: Generative Restricted Kernel Machines: A framework for multi-view generation and disentangled feature learning. *Neural Networks* **135**, 177–191 (2021)
17. Schreurs, J., Suykens, J.: Generative kernel pca. In: ESANN. pp. 129–134 (2018)
18. Suykens, J.A.K.: Deep Restricted Kernel Machines Using Conjugate Feature Duality. *Neural Computation* **29**(8), 2123–2163 (2017)
19. Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B., Vandewalle, J.: *Least Squares Support Vector Machines*. World Scientific (2002)
20. Tonin, F., Pandey, A., Patrinos, P., Suykens, J.A.K.: Unsupervised Energy-based Out-of-distribution Detection using Stiefel-Restricted Kernel Machine. In: IJCNN (2021)
21. Tonin, F., Patrinos, P., Suykens, J.A.K.: Unsupervised learning of disentangled representations in deep restricted kernel machines with orthogonality constraints. *Neural Networks* **142**, 661–679 (2021)
22. Tonin, F., Tao, Q., Patrinos, P., Suykens, J.A.K.: Deep kernel principal component analysis for multi-level feature learning. *arXiv* (2023)
23. Wen, Z., Yin, W.: A feasible method for optimization with orthogonality constraints. *Mathematical Programming* **142**(1), 397–434 (2013)
24. Zhu, X.: A Riemannian conjugate gradient method for optimization on the Stiefel manifold. *Computational optimization and Applications* **67**(1), 73–110 (2017)