Deep Sketched Output Kernel Regression for Structured Prediction

Tamim El Ahmad[1](✉), Junjie Yang[1], Pierre Laforgue[2], and Florence d'Alché-Buc[1]

[1] LTCI, Télécom Paris, IP Paris, Palaiseau, France
tamim.elahmad@telecom-paris.fr
[2] Department of Computer Science, University of Milan, Milan, Italy
pierre.laforgue@unimi.it

Abstract. By leveraging the kernel trick in the output space, kernel-induced losses provide a principled way to define structured output prediction tasks for a wide variety of output modalities. In particular, they have been successfully used in the context of surrogate non-parametric regression, where the kernel trick is typically exploited in the input space as well. However, when inputs are images or texts, more expressive models such as deep neural networks seem more suited than non-parametric methods. In this work, we tackle the question of how to train neural networks to solve structured output prediction tasks, while still benefiting from the versatility and relevance of kernel-induced losses. We design a novel family of deep neural architectures, whose last layer predicts in a data-dependent finite-dimensional subspace of the infinite-dimensional output feature space deriving from the kernel-induced loss. This subspace is chosen as the span of the eigenfunctions of a randomly-approximated version of the empirical kernel covariance operator. Interestingly, this approach unlocks the use of gradient descent algorithms (and consequently of any neural architecture) for structured prediction. Experiments on synthetic tasks as well as real-world supervised graph prediction problems show the relevance of our method.

Keywords: Structured prediction · Deep learning · Kernel methods

1 Introduction

Learning to predict complex outputs, such as graphs or any other composite object, raises many challenges in machine learning [3,19,51]. The most important of them is undoubtedly the difficulty of leveraging the geometry of the

T. El Ahmad and J. Yang—Equal contribution.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-70352-2_6.

output space. In supervised graph prediction, for instance, it is often required to use node permutation-invariant and node size-insensitive distances, such as the Fused Gromov-Wasserstein distance [69]. In that regard, surrogate methods such as Output Kernel Regression [25,33,71] offer a powerful and flexible framework by using the kernel trick in the output space. By appropriately choosing the output kernel, it is possible to incorporate various kinds of information, both in the model and in the loss function [13,15,49]. One important limitation of this approach, however, is that the induced output features may be infinite-dimensional.

If leveraging the kernel trick in the input space may be a solution [12,16], such non-parametric methods are usually outperformed by more expressive models such as neural networks when input data consist of images or texts. In the context of structured prediction, deep learning has led to impressive results for specific tasks, such as semantic segmentation [37] or the protein 3D structure prediction [32]. To create versatile deep models, the main approach explored in the literature is the energy-based approach, which consists of converting structured prediction into learning a scalar score function [4,27,42,43]. However, these methods usually fail to go beyond structured prediction problems which can be reformulated as high-dimensional multi-label classification problems, as pointed out by [26]. Besides, this approach requires a two-step strategy, since the energy function is first learned thanks to the training data, and then maximized at inference time. To obtain an end-to-end model, [5] uses direct risk minimization techniques, and [67] introduces inference networks, a neural architecture that approximates the inference problem. In this work, we choose to benefit from the versatility of kernel-induced losses, and deploy it to neural networks. To this end, we address the infinite-dimensionality of the output features by computing a finite-dimensional basis within the output feature space, defined as the eigenbasis of a sketched version of the output empirical covariance operator.

Sketching [45,73] is a dimension-reduction technique based on random linear projections. In the context of kernel methods, it has mainly been explored through the so-called Nyström approximation [56,72], or via specific distributions such as Gaussian or Randomized Orthogonal Systems [40,75]. Previous works tackle sketched scalar kernel regression by providing a low-rank approximation of the Gram matrix [2,20], reducing the number of parameters to learn at the optimization stage [40,75], providing data-dependent random features [39,72,74], or leveraging an orthogonal projection operator in the feature space [56]. This last interpretation has been used to learn large-scale dynamical systems [47], and structured prediction [22].

In our proposition to solve structured prediction from complex input data, we make the following contributions:

- We introduce Deep Sketched Output Kernel Regression, a novel family of deep neural architectures whose last layer predicts a data-dependent finite-dimensional representation of the outputs, that lies in the infinite-dimensional feature space deriving from the kernel-induced loss.

- This last layer is computed beforehand, and is the eigenbasis of the sketched empirical covariance operator, unlocking the use of gradient-based techniques to learn the weights of the previous layers for any neural architecture.
- We empirically show the relevance of our approach on a synthetic least squares regression problem, and provide a strategy to select the sketching size.
- We show that DSOKR performs well on two text-to-molecule datasets.

2 Deep Sketched Output Kernel Regression

In this section, we set up the problem of structured prediction. Specifically, we consider surrogate regression approaches for kernel-induced losses. By introducing a last layer able to make predictions in a Reproducing Kernel Hilbert Space (RKHS), we unlock the use of deep neural networks as hypothesis space.

Consider the general regression task from an input domain $\mathcal{X}$ to a structured output domain $\mathcal{Y}$ (e.g., the set of labeled graphs of arbitrary size). Learning a mapping from $\mathcal{X}$ to $\mathcal{Y}$ naturally requires taking into account the structure of the output space. One way to do so is the *Output Kernel Regression* (OKR) framework [10,12,16,25,71], which is part of the family of surrogate regression methods [14,15].

Output Kernel Regression. A positive definite (p.d.) kernel $\mathrm{k} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a symmetric function such that for all $n \geq 1$, and any $(y_i)_{i=1}^n \in \mathcal{Y}^n$, $(\alpha_i)_{i=1}^n \in \mathbb{R}^n$, we have $\sum_{i,j=1}^n \alpha_i \, \mathrm{k}\,(y_i, y_j)\, \alpha_j \geq 0$. Such a kernel is associated with a canonical feature map $\psi\colon y \in \mathcal{Y} \mapsto \mathrm{k}(\cdot, y)$, which is uniquely associated with a Hilbert space of functions $\mathcal{H} \subset \mathbb{R}^{\mathcal{Y}}$, the RKHS, such that $\psi(y) \in \mathcal{H}$ for all $y \in \mathcal{Y}$, and $h\,(y) = \langle h, \psi(y) \rangle_{\mathcal{H}}$ for any $(h, y) \in \mathcal{H} \times \mathcal{Y}$. Given a p.d. kernel k, ψ its canonical feature map and $\mathcal{H}$ its RKHS, the OKR approach that we consider in this work exploits the kernel-induced squared loss:

$$\Delta(y, y') \coloneqq \|\, \psi(y) - \psi(y') \|_{\mathcal{H}}^2 = \mathrm{k}(y, y) - 2\,\mathrm{k}(y, y') + \mathrm{k}(y', y')\,. \tag{1}$$

The versatility of loss (1) stems from the large variety of kernels that have been designed to compare structured objects [7,24,38]. In multi-label classification, for instance, choosing the linear kernel or the Tanimoto kernel induces respectively the Hamming and the F1-loss [65]. In label ranking, Kemeny and Hamming embeddings define respectively Kendall's τ distance and the Hamming loss [38,50]. For sequence prediction tasks, n-gram kernels have been proven useful [17,33,50], while an abundant collection of kernels has been designed for graphs, based either on bags of structures or information propagation, see Appendix B and [7] for examples.

If kernel-induced losses can be computed easily thanks to the kernel trick, note that most of them are however non-differentiable. In particular, this largely compromises their use within deep neural architectures, that are however key to achieve state-of-the-art performances in many applications. In this work, we close this gap and propose an approach that benefits from both the expressivity of neural networks for input image/textual data, as well as the relevance of kernel-induced losses for structured outputs. Formally, let ρ be a joint probability distribution on $\mathcal{X} \times \mathcal{Y}$. Our goal is to design a family $(f_\theta)_{\theta \in \Theta} \subset \mathcal{Y}^{\mathcal{X}}$ of neural networks with outputs in $\mathcal{Y}$ that can minimize the kernel-induced loss, i.e., that can solve

$$\min_{\theta \in \Theta} \mathbb{E}_{(x,y)\sim\rho}\Big[\left\|\psi(y) - \psi\left(f_\theta(x)\right)\right\|_{\mathcal{H}}^2 \Big]. \tag{2}$$

To do so, we assume that we can access a training sample $\{(x_1, y_1), \ldots, (x_n, y_n)\}$

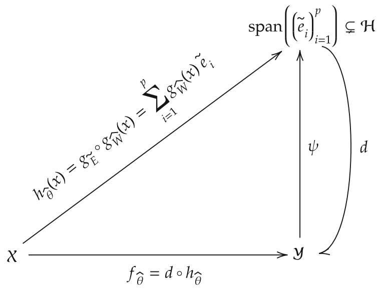

$\operatorname{span}\left(\left(\tilde{e}_i\right)_{i=1}^{p}\right) \subsetneq \mathcal{H}$
$h_{\hat\theta}(x) = g_{\tilde{E}} \circ g_{\widehat{W}}(x) = \sum_{i=1}^{p} g_{\widehat{W}}(x)_i \tilde{e}_i$
ψ
d
$\mathcal{X}$
$\mathcal{Y}$
$f_{\hat\theta} = d \circ h_{\hat\theta}$

Fig. 1. Illustration of DSOKR model.

drawn i.i.d. from ρ. Since learning f_θ through ψ is difficult, we employ a two-step method. First, we solve the surrogate empirical problem

$$\hat\theta \in \underset{\theta \in \Theta}{\arg\min}\ L(\theta) = \underset{\theta \in \Theta}{\arg\min}\ \frac{1}{n}\sum_{i=1}^{n} \|h_\theta(x) - \psi(y)\|_{\mathcal{H}}^2, \tag{3}$$

where $(h_\theta)_{\theta\in\Theta} \subset \mathcal{H}^{\mathcal{X}}$ is a family of neural networks with outputs in $\mathcal{H}$. We then retrieve the solution by solving for any prediction the pre-image problem

$$f_{\hat\theta}(x) = \underset{y \in \mathcal{Y}}{\arg\min}\ \|h_{\hat\theta}(x) - \psi(y)\|_{\mathcal{H}}^2. \tag{4}$$

This approach nonetheless raises a major challenge. Indeed, the dimension of the canonical feature space $\mathcal{H}$ may be infinite, making the training very difficult. The question we have to answer now is: *how can we design a neural architecture that is able to learn infinite-dimensional output kernel features?*

Neural Networks with Infinite-Dimensional Outputs. We propose a novel architecture of neural networks to compute the function h_θ with values in $\mathcal{H}$, as illustrated in Fig. 1. Let $p \geq 1$, our architecture is the composition of two networks: an input neural network, denoted $g_W : \mathcal{X} \to \mathbb{R}^p$, with generic weights $W \in \mathcal{W}$, and a last layer composed of a unique *functional* neuron, denoted $g_{\mathrm{E}} : \mathbb{R}^p \to \mathcal{H}$, that predicts in $\mathcal{H}$. The latter depends on the kernel k used in the loss definition, and on a finite basis $\mathrm{E} = ((e_j)_{j=1}^p) \in \mathcal{H}^p$ of elements in $\mathcal{H}$. We let $\theta = (W, \mathrm{E})$, and for any $x \in \mathcal{X}$, we have

$$h_\theta(x) \coloneqq g_{\mathrm{E}} \circ g_W(x), \tag{5}$$

where g_W typically implements a $L-1$ neural architecture encompassing, multilayered perceptrons, convolutional neural networks, or transformers. Instead, g_{E} computes a linear combination of some basis functions $\mathrm{E} = (e_j)_{j=1}^p \in \mathcal{H}^p$

$$g_{\mathrm{E}} : z \in \mathbb{R}^{\mathrm{p}} \mapsto \sum_{j=1}^{p} z_j e_j \in \mathcal{H}\,. \tag{6}$$

With this architecture, computations remain finite, and the input neural network outputs the coefficients of the basis expansion, generating predictions in $\mathcal{H}$.

Remark 1 (Input Neural net's last layers). Since the neural network g_W learns the coordinates of the surrogate estimator in the basis, its last layers are always mere fully connected ones, regardless of the nature of the output data at hand.

2.1 Learning Neural Networks with Infinite-Dimensional Outputs

Learning the surrogate regression model h_θ now boils down to computing $\theta = (W, \mathrm{E})$. We propose to solve this problem in two steps. First, we learn a suitable E using only the output training data $(\psi(y_i))_{i=1}^n$ in an unsupervised fashion. Then, we use standard gradient-based algorithms to learn W through the frozen last layer, minimizing the loss on the whole supervised training sample $(x_i, \psi(y_i))_{i=1}^n$.

Estimating the Functional Last Unit $g_{\mathbf{E}}$. A very first idea is to choose E as the non-orthogonal dictionary $\psi(y_j)_{j=1}^n$. But this choice induces a very large output dimension (namely, $p = n$) for large training datasets.

An alternative consists in using Kernel Principal Component Analysis (KPCA) [58]. Given a marginal probability distribution over $\mathcal{Y}$, let $\mathrm{C} = \mathbb{E}_y[\psi(y) \otimes \psi(y)]$ be the covariance operator associated with k, and $\widehat{\mathrm{C}} = (1/n)\sum_{i=1}^n \psi(y_i) \otimes \psi(y_i)$ its empirical counterpart. Let S be the sampling operator that transforms a function $f \in \mathcal{H}$ into the vector $(1/\sqrt{n})(f(x_1), \dots, f(x_n))^\top$ in $\mathbb{R}^n$, and denote by $\mathrm{S}^\#$ its adjoint. We have $\mathrm{S}^\# \colon \alpha \in \mathbb{R}^n \mapsto (1/\sqrt{n})\sum_{i=1}^n \alpha_i \, \psi(y_i) \in \mathcal{H}$, and $\widehat{\mathrm{C}} = \mathrm{S}^\# \, \mathrm{S}$. KPCA provides the eigenbasis of $\widehat{\mathrm{C}}$ by computing the SVD of the output Gram matrix, for a prohibitive computational cost of $\mathcal{O}(n^3)$. In practice, though, it is often the case that the so-called *capacity condition* holds [15, 22], i.e., that the spectrum of the empirical covariance operator enjoys a large eigendecay. It is then possible to efficiently approximate the eigenbasis of $\widehat{\mathrm{C}}$ using random projections techniques [45], also known as sketching, solving this way the computational and memory issues.

Sketching for Kernel Methods. Sketching [73] is a dimension reduction technique based on random linear projections. Since the goal is to reduce the dependency on the number of training samples n in kernel methods, such linear projections can be encoded by a randomly drawn matrix $\mathrm{R} \in \mathbb{R}^{\mathrm{m} \times n}$, where $\mathrm{m} \ll n$.

Standard examples include Nyström approximation [46], where each row of R is randomly drawn from the rows of the identity matrix I_n, also called sub-sampling sketches, and Gaussian sketches [75], where all entries of R are i.i.d. Gaussian random variables. As they act as a random training data sub-sampler and then largely reduce both the time and space complexities induced by kernel methods, sub-sampling sketches are the most popular sketching type applied to kernels, while Gaussian sketches are less computationally efficient but offer better statistical properties. Hence, given a sketching matrix $\mathrm{R} \in \mathbb{R}^{\mathrm{m} \times n}$, one can defines $\tilde{\mathcal{H}}_{\mathcal{Y}} = \operatorname{span}((\sum_{j=1}^{n} \mathrm{R}_{\mathrm{ij}}\, \psi(y_j))_{i=1}^{\mathrm{m}})$ which is a low-dimensional linear subspace of $\mathcal{H}$ of dimension at most m. One can even compute the basis $\tilde{E}$ of $\tilde{\mathcal{H}}_{\mathcal{Y}}$, providing the last layer $g_{\tilde{E}}$.

Sketching to Estimate $\boldsymbol{g_E}$. We here show how to compute the basis $\tilde{E}$ of $\tilde{\mathcal{H}}_{\mathcal{Y}}$. Let $\mathrm{m} < n$, and $\mathrm{R} \in \mathbb{R}^{\mathrm{m} \times n}$ be a sketching matrix. Let $\widetilde{\mathrm{K}} = \mathrm{R}\,\mathrm{K}\,\mathrm{R}^\top \in \mathbb{R}^{\mathrm{m} \times \mathrm{m}}$ be the sketched Gram matrix, and $\left\{(\sigma_i(\widetilde{\mathrm{K}}), \tilde{\mathbf{v}}_i), i \in [\mathrm{m}]\right\}$ its eigenpairs, in descending order. We set $\mathrm{p} = \operatorname{rank}\left(\widetilde{\mathrm{K}}\right)$. Note that $p \leq \mathrm{m}$, and that $p = \mathrm{m}$ for classical examples, e.g. full-rank K and sub-sample without replacement or Gaussian R. The following proposition provides the eigenfunctions of the sketched empirical covariance operator.

Proposition 1. *[22, Proposition 2] The eigenfunctions of the sketched empirical covariance operator* $\widetilde{\mathrm{C}} = \mathrm{S}^{\#}\mathrm{R}^\top\mathrm{R}\,\mathrm{S}$ *are the* $\tilde{e}_j = \sqrt{\frac{n}{\sigma_j(\widetilde{\mathrm{K}})}}\,\mathrm{S}^{\#}\,\mathrm{R}^\top\,\tilde{\mathbf{v}}_j \in \mathcal{H}$*, for* $j \leq \mathrm{p}$.

Hence, computing the eigenfunctions of $\widetilde{\mathrm{C}}$ provides a basis of $\mathcal{H}$ of dimension p. Note that in sketched KPCA, which has been explored via Nyström approximation in [63,64], one solves for $i = 1, \dots, \mathrm{m}$

$$f_i = \underset{f \in \mathcal{H}}{\arg\max} \left\{\langle f, \widehat{\mathrm{C}}\, f\rangle_{\mathcal{H}} : f \in \tilde{\mathcal{H}}_{\mathcal{Y}}, \|f\|_{\mathcal{H}} = 1, f \perp \{f_1, \dots, f_{i-1}\}\right\} \tag{7}$$

where $\tilde{\mathcal{H}}_{\mathcal{Y}} = \operatorname{span}((\sum_{j=1}^{n} \mathrm{R}_{\mathrm{ij}}\, \psi(y_j))_{i=1}^{\mathrm{m}})$. Let $\widetilde{\mathrm{P}}$ be the orthogonal projector onto the basis $(\tilde{e}_1, \dots, \tilde{e}_{\mathrm{p}})$, solving Equation (7) is equivalent to compute the eigenfunctions of the projected empirical covariance operator $\widetilde{\mathrm{P}}\,\widehat{\mathrm{C}}\,\widetilde{\mathrm{P}}$, i.e., to compute the KPCA of the projected kernel $\langle \widetilde{\mathrm{P}}\,\psi(\cdot), \widetilde{\mathrm{P}}\,\psi(\cdot)\rangle_{\mathcal{H}}$. Besides, as for the SVD of $\widetilde{\mathrm{C}}$, sketched KPCA needs the SVD of $\widetilde{\mathrm{K}}$ to obtain its square root, but also requires the additional $\widetilde{\mathrm{K}}^{1/2}\,\mathrm{R}\,\mathrm{K}^2\,\mathrm{R}^\top\,\widetilde{\mathrm{K}}^{1/2}$ SVD computation.

Remark 2 (Random Fourier Features). Another popular kernel approximation is the Random Fourier Features [44,52,57]. They approximate a kernel function as the inner product of small random features using Monte-Carlo sampling when the kernel writes as the Fourier transform of a probability distribution. Such an approach, however, defines a new randomly approximated kernel, then a new randomly approximated loss, which can induce learning difficulties due to the bias and variance inherent to the approximation. Unlike RFF, sketching is not limited to kernels writing as the Fourier transform of a probability distribution and to defining an approximated loss, it allows the building of a low-dimensional basis within the original feature space of interest.

Learning the Input Neural Network g_W. Equipped with the basis $\tilde{E} = (\tilde{e}_j)_{j \leq p}$, we can compute a novel expression of the loss $L(\theta) = L(\tilde{E}, W)$, see Appendix A for the proof.

Algorithm 1. Deep Sketched Output Kernel Regression (DSOKR)

```
input: training $\{(x_i, y_i)\}_{i=1}^n$, validation $\{(x_i^{\text{val}}, y_i^{\text{val}})\}_{i=1}^{n_{\text{val}}}$ pairs, test inputs $\{x_i^{\text{te}}\}_{i=1}^{n_{\text{te}}}$,
    candidate outputs test inputs $\{y_i^{\text{c}}\}_{i=1}^{n_{\text{c}}}$, normalized output kernel k, sketching
    matrix $\mathrm{R} \in \mathbb{R}^{\mathrm{m} \times n}$, neural network $g_W$
init  : $\widetilde{\mathrm{K}} = \mathrm{R}\,\mathrm{K}\,\mathrm{R}^\top \in \mathbb{R}^{\mathrm{m} \times \mathrm{m}}$ where $\mathrm{K} = (\mathrm{k}(y_i, y_j))_{1 \leq i,j \leq n} \in \mathbb{R}^{n \times n}$

// 1. a. Training of $g_E$: computations for the basis $\widetilde{E}$
• Construct $\widetilde{D}_{\mathrm{p}} \in \mathbb{R}^{\mathrm{p} \times \mathrm{p}}$, $\widetilde{V}_{\mathrm{p}} \in \mathbb{R}^{\mathrm{m} \times \mathrm{p}}$ such that $\widetilde{V}_{\mathrm{p}} \widetilde{D}_{\mathrm{p}} \widetilde{V}_{\mathrm{p}}^\top = \widetilde{\mathrm{K}}$ (SVD of $\widetilde{\mathrm{K}}$)
• $\widetilde{\Omega} = \widetilde{D}_{\mathrm{p}}^{-1/2} \widetilde{V}_{\mathrm{p}}^\top \in \mathbb{R}^{\mathrm{p} \times \mathrm{m}}$

// 1. b. Training of $g_W$: solving the surrogate problem
• $\tilde{\psi}(y_i) = \widetilde{\Omega}\,\mathrm{R}\,k^{y_i} \in \mathbb{R}^{\mathrm{p}}, \forall\, 1 \leq i \leq n$, $\tilde{\psi}(y_i^{\text{val}}) = \widetilde{\Omega}\,\mathrm{R}\,k^{y_i^{\text{val}}} \in \mathbb{R}^{\mathrm{p}}, \forall\, 1 \leq i \leq n_{\text{val}}$
• $\hat{W} = \underset{W \in \mathcal{W}}{\arg\min}\ \frac{1}{n} \sum_{i=1}^n \left\| g_W(x_i) - \tilde{\psi}(y_i) \right\|_2^2$ (training of $g_W$ with training
$\{(x_i, \tilde{\psi}(y_i))\}_{i=1}^n$ and validation $\{(x_i^{\text{val}}, \tilde{\psi}(y_i^{\text{val}}))\}_{i=1}^{n_{\text{val}}}$ pairs and Mean Squared Error loss)

// 2. Inference
• $\tilde{\psi}(y_i^{\text{c}}) = \widetilde{\Omega}\,\mathrm{R}\,k^{y_i^{\text{c}}} \in \mathbb{R}^{\mathrm{p}}, \forall\, 1 \leq i \leq n_{\text{c}}$
• $f_{\hat{\theta}}(x_i^{\text{te}}) = y_j^{\text{c}}$ where $j = \underset{1 \leq j \leq n_{\text{c}}}{\arg\max}\ g_{\hat{W}}(x_i^{\text{te}})^\top \tilde{\psi}(y_j^{\text{c}})$, $\forall\, 1 \leq i \leq n_{\text{te}}$

return $f_{\hat{\theta}}(x_i^{\text{te}}), \forall\, 1 \leq i \leq n_{\text{te}}$
```

Proposition 2. *Given the pre-trained basis $\tilde{E} = (\tilde{e}_j)_{j \leq p}$, $L(\tilde{E}, W)$ expresses as*

$$L(\tilde{E}, W) = \frac{1}{n} \sum_{i=1}^n \left\| g_W(x_i) - \tilde{\psi}(y_i) \right\|_2^2, \tag{8}$$

where $\tilde{\psi}(y) = (\tilde{e}_1(y), \dots, \tilde{e}_{\mathrm{p}}(y))^\top = \widetilde{D}_{\mathrm{p}}^{-1/2} \widetilde{V}_{\mathrm{p}}^\top \mathrm{R}\,\mathrm{k}^{\mathrm{y}} \in \mathbb{R}^{\mathrm{p}}$, $\widetilde{V}_{\mathrm{p}} = (\tilde{\mathbf{v}}_1, \dots, \tilde{\mathbf{v}}_{\mathrm{p}})$, $\widetilde{D}_{\mathrm{p}} = \mathrm{diag}(\sigma_1(\widetilde{\mathrm{K}}), \dots, \sigma_{\mathrm{p}}(\widetilde{\mathrm{K}}))$, *and* $\mathrm{k}^{\mathrm{y}} = (\mathrm{k}(y, y_1), \dots, \mathrm{k}(y, y_n))$.

Finally, given $\tilde{E}$ and Proposition 2, learning the full network h_θ boils down to learning the input neural network g_W and thus finding a solution $\hat{W}$ to

$$\min_{W \in \mathcal{W}} \frac{1}{n} \sum_{i=1}^n \left\| g_W(x_i) - \tilde{\psi}(y_i) \right\|_2^2. \tag{9}$$

A classical stochastic gradient descent algorithm can then be applied to learn W. Compared to the initial loss (3), the relevance of (9) is governed by the quality of the approximation of $\widehat{\mathrm{C}}$ by $\widetilde{\mathrm{C}}$. If our approach regularises the solution (the range of the surrogate estimator h_θ is restricted from $\mathcal{H}$ to E), this restriction may not be limiting if we set $\mathrm{m} \geq p$ high enough to capture all the information contained in $\widehat{\mathrm{C}}$. We discuss strategies to correctly set m at the beginning of Sect. 3.

Remark 3 (Beyond the square loss). Equipped with such an architecture $g_W \circ g_E$, one can easily consider any loss that writes $\Delta(y, y') = c(\| \psi(y) - \psi(y')\|^2_{\mathcal{H}})$, where $c : \mathbb{R}_+ \to \mathbb{R}_+$ is a non-decreasing sub-differentiable function. For instance, in the presence of output outliers, one could typically consider robust losses such as the Huber or ϵ-insensitive losses, that correspond to different choices of function c [30,41,62].

2.2 The Pre-image Problem at Inference Time

We focus now on the decoding part, i.e., on computing

$$d \circ h_{\hat{\theta}}(x) = \underset{y \in \mathcal{Y}}{\arg\min}\ \ \mathrm{k}(y, y) - 2 g_{\hat{W}}(x)^\top \tilde{\psi}(y) = \underset{y \in \mathcal{Y}}{\arg\max}\ \ g_{\hat{W}}(x)^\top \tilde{\psi}(y)$$

if we assume k to be normalized, i.e. $\mathrm{k}(y, y') = 1, \forall y, y' \in \mathcal{Y}$. For a test set $X^{\mathrm{te}} = (x_1^{\mathrm{te}}, \ldots, x_{n_{\mathrm{te}}}^{\mathrm{te}}) \in \mathcal{X}^{n_{\mathrm{te}}}$ and a candidate set $Y^{\mathrm{c}} = (y_1^{\mathrm{c}}, \ldots, y_{n_{\mathrm{c}}}^{\mathrm{c}}) \in \mathcal{Y}^{n_{\mathrm{c}}}$, for all $1 \leq i \leq n_{te}$, the prediction is given by

$$f_{\hat{\theta}}(x_i^{\mathrm{te}}) = y_j^{\mathrm{c}} \quad \text{where} \quad j = \underset{1 \leq j \leq n_{\mathrm{c}}}{\arg\max}\ g_{\hat{W}}(x_i^{\mathrm{te}})^\top \tilde{\psi}(y_j^{\mathrm{c}})\,. \tag{10}$$

Hence, the decoding is particularly suited to problems for which we have some knowledge of the possible outcomes, such as molecular identification problems [11]. When the output kernel is differentiable, it may also be solved using standard gradient-based methods. Finally, some ad-hoc ways to solve the pre-image problem exist for specific kernels, see e.g., [17] for the sequence prediction via n-gram kernels, or [38] for label ranking via Kemeny, Hamming, or Lehmer embeddings. The DSOKR framework is summarized in Algorithm 1.

3 Experiments

In this section, we first present a range of strategies to select the sketching size and an analysis of our proposed DSOKR on a synthetic dataset. Besides, we show the effectiveness of DSOKR through its application to two real-world Supervised Graph Prediction (SGP) tasks: SMILES to Molecule and Text to Molecule. The code to reproduce our results is available at: https://github.com/tamim-el/dsokr.

Sketching Size Selection Strategy. A critical hyper-parameter of DSOKR is the sketching size m. Indeed, the optimal choice is the dimension of the subspace containing the output features. However, to estimate this dimension, one has to compute the eigenvalues of K, which has the prohibitive complexity of $\mathcal{O}(n^3)$. Hence, a first solution is to compute the Approximate Leverage Scores (ALS) as described in [1]. This is an approximation of the eigenvalues of K that relies on sub-sampling $n_S < n$ entries within the whole training set. Moreover, we use another technique that we call *Perfect h*. Considering any pair (x, y) in a validation set, we replace $g_W(x)$ by the "perfect" coefficients of the expansion, i.e., for each $j = 1, \ldots, p$, $\langle \tilde{e}_j, \psi(y) \rangle_{\mathcal{H}}$ and define "perfect" surrogate estimator h_ψ as follows

$$h_\psi(x) = \sum_{j=1}^{\mathrm{p}} \langle \tilde{e}_j, \psi(y) \rangle_{\mathcal{H}} \; \tilde{e}_j = \sum_{j=1}^{\mathrm{p}} \tilde{\psi}(y)_j \; \tilde{e}_j \, . \tag{11}$$

Then, we evaluate the performance of this "perfect" surrogate estimator h_ψ on a validation set to select m. Hence, *Perfect h* allows to select the minimal m in the range given by ALS such that the performance of h_ψ reaches an optimal value.

3.1 Analysis of DSOKR on Synthetic Least Squares Regression

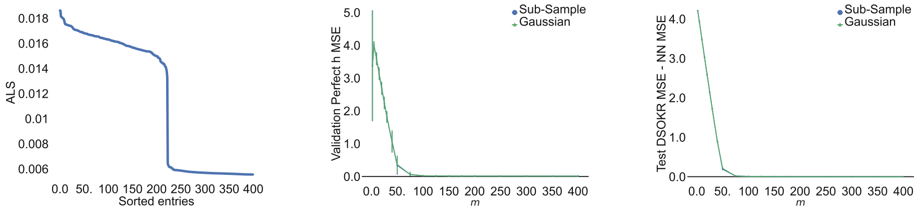


Fig. 2. Sorted 400 highest ALS (left), validation MSE of *Perfect h* w.r.t. m (center) and the difference between test MSE of DSOKR and NN w.r.t. m (right).

Dataset. We generate a synthetic dataset of least-squares regression, using then a linear output kernel, with $n = 50{,}000$ training data points, $\mathcal{X} = \mathbb{R}^{2{,}000}$, $\mathcal{Y} = \mathbb{R}^{1{,}000}$, and $\mathcal{H} = \mathcal{Y} = \mathbb{R}^{1{,}000}$. The goal is to build this dataset such that the outputs lie in a subspace of $\mathcal{Y}$ of dimension $d = 50 < 1{,}000$. Hence, given d randomly drawn orthonormal vectors $(u_j)_{j=1}^d$, for all $1 \leq i \leq n$, the outputs are such that $y_i = \sum_{j=1}^d \alpha(x_i)_j u_j + \varepsilon_i$, where α is a function of the inputs and $\varepsilon_i \sim \mathcal{N}(0, \sigma^2 I_{1{,}000})$ are i.i.d. with $\sigma^2 = 0.01$. We generate i.i.d. normal distributed inputs $x_i \sim \mathcal{N}(0, C)$, where $(\sigma_j(C) = j^{-1/2})_{j=1}^{2{,}000}$ and its eigenvectors are randomly drawn. Finally, we draw $H \in \mathbb{R}^{d \times 2{,}000}$ with i.i.d. coefficients from the standard normal distribution, and the outputs are given for $1 \leq i \leq n$ by

$$y_i = UHx_i + \varepsilon_i \, , \tag{12}$$

where $U = (u_1, \ldots, u_d) \in \mathbb{R}^{1{,}000 \times d}$. We generate validation and test sets of $n_{\mathrm{val}} = 5{,}000$ and $n_{\mathrm{te}} = 10{,}000$ points in the same way.

Experimental Settings. We first compute the ALS as described above. We take as regularisation penalty $\lambda = 10^{-4}$, sampling parameter $n_S = \sqrt{n}$ and probability vector $(p_i = 1/n)_{i=1}^n$ (uniform sampling). Then, we perform the sketching size selection strategy *Perfect h.* Note that using a linear output kernel, $\psi : y \in \mathbb{R}^{1,000} \mapsto y$, then $\tilde{e}_i = (1/\sqrt{\sigma_i(\widetilde{\mathrm{K}})})\tilde{\mathbf{v}}_i^\top \mathrm{R} Y$, where $Y = (y_1, \ldots, y_n)^\top \in \mathbb{R}^{n \times 1,000}$, and

$$h_{\hat{\theta}}(x) = Y^\top \mathrm{R}^\top \widetilde{V}_{\mathrm{p}} \widetilde{D}_{\mathrm{p}}^{-1/2} g_{\hat{W}}(x)\,. \tag{13}$$

Finally, we perform our DSOKR model whose neural network g_W is a Single-Layer Perceptron, i.e. with no hidden layer, and compare it with an SLP whose output size is $1,000$, and trained with a Mean Squared Error loss, that we call "NN". We select the optimal number of epochs thanks to the validation set and evaluate the performance via the MSE. We use the ADAM [36] optimizer. For the *Perfect h* and DSOKR models and any sketching size $\mathrm{m} \in [2, 400]$, we average the results over five replicates of the models. We use uniform sub-sampling without replacement and Gaussian sketching distributions.

Experimental Results. Figure 2 (left) presents the sorted 400 highest leverage scores. This gives a rough estimate of the optimal sketching size since the leverage scores converge to a minimal value starting from 200 approximately, which is an upper bound of the true basis dimension $d = 50$. Figure 2 (center) shows that *Perfect h* is a relevant strategy to fine-tune m since the obtained optimal value is $\mathrm{m} = 75$, which is very close to $d = 50$. This small difference comes from the added noise ε_i. Moreover, this value corresponds to the optimal value based on the DSOKR test MSE. In fact, Fig. 2 (right) presents the performance DSOKR for many m values compared with NN. DSOKR performance converges to the NN's performance for $\mathrm{m} = 75$ as well. Hence, we show that DSOKR attains optimal performance if its sketching size is set as the dimension of the output marginal distribution's range, which can be estimated thanks to the ALS and the *Perfect h* strategies. There is no difference between sub-sample and Gaussian sketching since the dataset is rather simple. Moreover, note that the neural network of the DSOKR model for $\mathrm{m} = 75$ contains $150,075$ parameters, whereas the NN model contains $2,001,000$ parameters. Then, our sketched basis strategy, even in the context of multi-output regression, allows to reduce the size of the last layer, simplifying the regression problem and reducing the number of weights to learn.

3.2 SMILES to Molecule: SMI2Mol

Dataset. We use the QM9 molecule dataset [54,55], containing around 130,000 small organic molecules. These molecules have been processed using RDKit[1], with aromatic rings converted to their Kekule form and hydrogen atoms removed. We also remove molecules containing only one atom. Each molecule contains up to 9 atoms of Carbon, Nitrogen, Oxygen, or Fluorine, along with three types

[1] RDKit: Open-source cheminformatics. https://www.rdkit.org.

of bonds: single, double, and triple. As input features, we use the Simplified Molecular Input Line-Entry System (SMILES), which are strings describing their chemical structure. We refer to the resulting dataset as **SMI2Mol**.

Experimental Set-Up. Using all SMILES-Molecule pairs, we build five splits using different seeds. Each split has 131,382 training samples, 500 validation samples, and 2,000 test samples. In DSOKR, g_W is a Transformer [68]. The SMILES strings are tokenized into character sequences as inputs for the Transformer encoder. To define the loss on output molecules, we cross-validate several graph kernels, including the Weisfeiler-Lehman subtree kernel (WL-VH) [60], the neighborhood subgraph pairwise distance kernel (NSPD) [18], and the core Weisfeiler-Lehman subtree kernel (CORE-WL) [48]. We use the implementation of the graph kernels provided by the Python library GraKel [61]. We employ SubSample sketching for the output kernel. The sketching size m is fixed using our proposed *Perfect h* strategy. Our method is benchmarked against SISOKR [22], NNBary-FGW [9], and ILE-FGW [9]. For ILE-FGW and SISOKR, we additionally use SubSample sketching [56] for input kernel approximation. To ensure a fair comparison, both SISOKR and ILE-FGW adopt the 3-gram kernel for the input strings, whereas NNBary-FGW and DSOKR use a Transformer encoder. The performance is evaluated using Graph Edit Distance (GED), implemented by the NetworkX package [28].

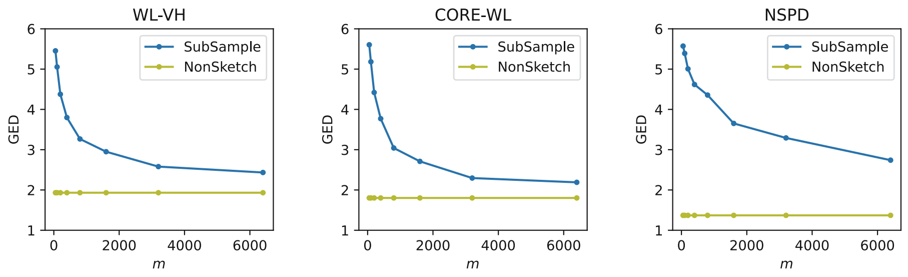
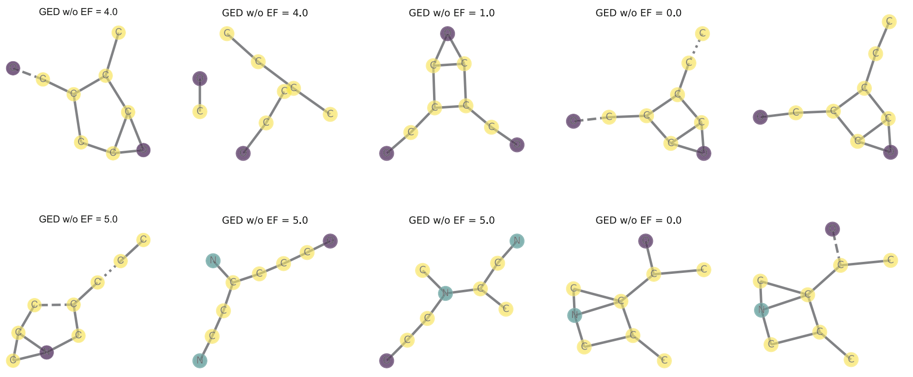


Fig. 3. The GED w/ edge feature w.r.t. the sketching size m for *Perfect h* for three graph kernels on SMI2Mol (m > 6400 is too costly computationally).

Table 1. Edit distance of different methods on SMI2Mol test set

	GED w/o edge feature ↓	GED w/ edge feature ↓
SISOKR	3.330 ± 0.080	4.192 ± 0.109
NNBary-FGW	5.115 ± 0.129	-
Sketched ILE-FGW	2.998 ± 0.253	-
DSOKR	$\mathbf{1.951 \pm 0.074}$	$\mathbf{2.960 \pm 0.079}$

Experimental Results. Figure 3 displays the GED obtained by *Perfect h* concerning various graph kernels. Based on this visualization, we have set the sketching sizes of WL-VH, CORE-WL, and NPSD to 3200, 3200, and 6400 respectively. Table 1 showcases the performance of various methods of SGP. Notably, DSOKR outperforms all baseline methods. It is evident that while graph kernels and the fused Gromov-Wasserstein (FGW) distance induce a meaningful feature space, the capabilities of SISOKR and ILE-FGW are constrained by the input kernels, thus highlighting the relevance of our proposed method. For further insight, a comparison of some prediction examples is provided in Fig. 4 and Appendix C.1.



(a) SISOKR (b) NNBary (c) ILE (d) DSOKR (e) True target

Fig. 4. Predicted molecules on the SMI2Mol dataset.

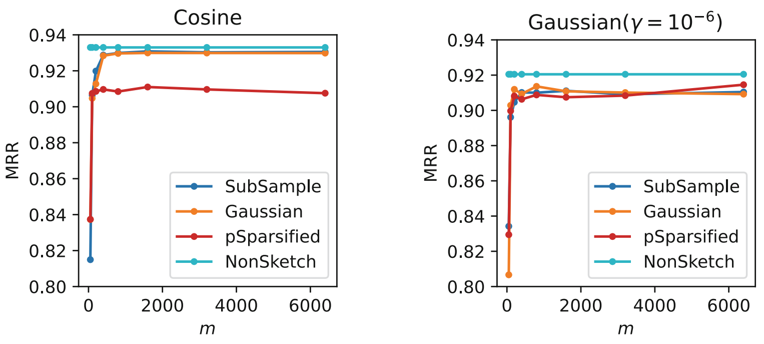


Fig. 5. The MRR scores on ChEBI-20 validation set w.r.t. m for *Perfect h* when the output kernel is Cosine or Gaussian on the ChEBI-20 dataset.

Table 2. Performance of different methods on ChEBI-20 test set. All the methods based on NNs use SciBERT as input text encoder for fair comparison. The number in the ensemble setting indicates the number of single models used.

	Hits@1 ↑	Hits@10 ↑	MRR ↑
SISOKR	0.4%	2.8%	0.015
SciBERT Regression	16.8%	56.9%	0.298
CMAM - MLP	34.9%	84.2%	0.513
CMAM - GCN	33.2%	82.5%	0.495
CMAM - Ensemble (MLP×3)	39.8%	87.6%	0.562
CMAM - Ensemble (GCN×3)	39.0%	87.0%	0.551
CMAM - Ensemble (MLP×3 + GCN×3)	44.2%	**88.7%**	0.597
DSOKR - SubSample Sketch	48.2%	87.4%	0.624
DSOKR - Gaussian Sketch	49.0%	87.5%	0.630
DSOKR - Ensemble (SubSample×3)	**51.0%**	88.2%	**0.642**
DSOKR - Ensemble (Gaussian×3)	50.5%	87.9%	**0.642**
DSOKR - Ensemble (SubSample×3 + Gaussian×3)	50.0%	88.3%	0.640

3.3 Text to Molecule: ChEBI-20

Dataset. The ChEBI-20 [21] dataset contains 33,010 pairs of compounds and descriptions. The compounds come from PubChem [34,35], and their descriptions (more than 20 words) from the Chemical Entities of Biological Interest (ChEBI) database [29]. The dataset is divided as follows: 80% for training, 10% for validation, and 10% for testing. The candidate set contains all compounds. The mean and median number of atoms per molecule is 32 and 25 respectively, and the mean and median number of words per description is 55 and 51 respectively.

Experimental Set-Up. For our method DSOKR, we use SciBERT [6] with an additional linear layer to parameterize g_W. The maximum length of the input tokens is set to 256. Mol2vec [31] is used as the output molecule representation, which is a vector of dimension 300. Based on the Mol2vec representation, we conduct cross-validation using the following kernels: Cosine kernel and Gaussian kernel with gamma chosen from $\{10^{-9}, 10^{-6}, 10^{-3}, 1\}$, along with the following three sketches: sub-sampling [56], Gaussian [75], and p-sparsified [23]. The sketching size for all combinations of the output kernels and sketches is determined using the *Perfect h* strategy. As for the baselines, we consider SciBERT Regression, Cross-Modal Attention Model (CMAM) [21], and SISOKR. In the case of SciBERT Regression, we address the regression problem using Mean Squared Error loss, where the output space is the embedding space of Mol2vec, within a function space parameterized by SciBERT. CMAM aims to enhance the cosine similarity between the text embedding and the corresponding molecule in true pairs by employing a contrastive loss function. Specifically, the former is derived from SciBERT, while the latter is generated using either a multi-layer perceptron (MLP) or a graph convolutional network (GCN) atop the Mol2vec representation. We reproduce the results of CMAM with the codes[2] released by [21]. In SISOKR, we use SciBERT embeddings as input features, leveraging the cosine kernel atop them. We maintain the identical output kernel sketching setup as in DSOKR. For all methods, we train the model using the best hyper-parameters with three random seeds and report the one with the best validation performance. The performance is evaluated with mean reciprocal rank (MRR), Hits@1 and Hits@10. We could not benchmark AMAN [76], as no implementation is publicly available.

Ensemble. In [21], the authors propose an ensemble strategy to enhance the results by aggregating the ranks obtained by different training of their models. If for each $1 \leq t \leq T$, R_t denotes the ranking returned by the model t, the new score is computed as follows

$$s(y_i) = \sum_{t=1}^{T} w_t R_t(y_i) \quad s.t. \quad \sum_{i=1}^{T} \omega_t = 1 \tag{14}$$

for each y_i in the candidate set. In our case, the computation of DSOKR's last layer g_E depends on a draw of the sketching matrix R, which means that DSOKR is particularly well-suited to the aggregation via multiple draws of the sketching matrix R_t and the training of the corresponding neural networks g_{W_t}. Hence, we explore two more ways of aggregating multiple DSOKR models, by averaging or maximizing these models' scores, i.e. for any input x and candidate y,

$$s(x,y) = \sum_{t=1}^{T} \omega_t \; g_{\hat{W}_t}(x)^\top \, \tilde{\psi}_t(y) \quad \text{or} \quad s(x,y) = \underset{1\leq t\leq T}{\arg\max} \; g_{\hat{W}_t}(x)^\top \, \tilde{\psi}_t(y). \tag{15}$$

We explore all three ensemble methods for DSOKR models and subsequently select the optimal one based on its validation performance.

Experimental Results. Figure 5 illustrates the validation MRR scores with *Perfect h*, for many m values, and either Cosine or Gaussian output kernels. It is evident that for both the Cosine kernel and Gaussian kernel (with $\gamma = 10^{-6}$) employing various sketching methods, the MRR score stabilizes as the sketching size exceeds 100, and that Cosine outperforms Gaussian. This observation allows us to choose $m = 100$, smaller than the original Mol2vec dimension, which is 300. Table 2 presents a comprehensive comparison of DSOKR with various baseline models. Firstly, comparing DSOKR with SISOKR reveals the critical importance of employing deep neural networks when dealing with complex structured inputs and DSOKR makes it possible in the case of functional output space. Secondly, the notable improvement over SciBERT Regression underscores the value of employing kernel sketching to derive more compact and better output features, thereby facilitating regression problem-solving. Lastly, DSOKR outperforms the sota CMAP for both single and ensemble models. See Appendix C.2 for more details.

[2] https://github.com/cnedwards/text2mol.

4 Conclusion

We designed a new architecture of neural networks able to minimize kernel-induced losses for structured prediction and achieving sota performance on molecular identification. An interesting avenue for future work is to derive excess risk for this estimator by combining deep learning theory and surrogate regression bounds.

Acknowledgments. Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Commission. Neither the European Union nor the granting authority can be held responsible for them. This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement 101120237 (ELIAS), the Télécom Paris research chair on Data Science and Artificial Intelligence for Digitalized Industry and Services (DSAIDIS) and the PEPR-IA through the project FOUNDRY.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Alaoui, A., Mahoney, M.W.: Fast randomized kernel ridge regression with statistical guarantees. In: NeurIPS, vol. 28 (2015)
2. Bach, F.: Sharp analysis of low-rank kernel matrix approximations. In: COLT, pp. 185–209 (2013)
3. Bakir, G., Hofmann, T., Smola, A.J., Schölkopf, B., Taskar, B.: Predicting Structured Data. The MIT Press, Cambridge (2007)
4. Belanger, D., McCallum, A.: Structured prediction energy networks. In: ICML, pp. 983–992 (2016)
5. Belanger, D., Yang, B., McCallum, A.: End-to-end learning for structured prediction energy networks. In: ICML, vol. 70, pp. 429–439 (2017)
6. Beltagy, I., Lo, K., Cohan, A.: SciBERT: a pretrained language model for scientific text. In: EMNLP-IJCNLP, pp. 3615–3620. Hong Kong, China (2019)
7. Borgwardt, K., Ghisu, E., Llinares-López, F., O'Bray, L., Rieck, B.: Graph kernels: state-of-the-art and future challenges. Found. Trends Mach. Learn. **13**(5–6), 531–712 (2020)
8. Borgwardt, K., Kriegel, H.: Shortest-path kernels on graphs. In: Fifth IEEE International Conference on Data Mining (ICDM'05), pp. 8 pp.– (2005)
9. Brogat-Motte, L., Flamary, R., Brouard, C., Rousu, J., D'Alché-Buc, F.: Learning to predict graphs with fused Gromov-Wasserstein barycenters. In: ICML, vol. 162, pp. 2321–2335 (2022)
10. Brouard, C., d'Alché-Buc, F., Szafranski, M.: Semi-supervised penalized output kernel regression for link prediction. In: ICML, pp. 593–600 (2011)
11. Brouard, C., Shen, H., Dührkop, K., d'Alché-Buc, F., Böcker, S., Rousu, J.: Fast metabolite identification with input output kernel regression. Bioinformatics **32**(12), 28–36 (2016)

12. Brouard, C., Szafranski, M., d'Alché Buc, F.: Input output kernel regression: supervised and semi-supervised structured output prediction with operator-valued kernels. JMLR **17**(1), 6105–6152 (2016)
13. Cabannes, V.A., Bach, F., Rudi, A.: Fast rates for structured prediction. In: COLT, pp. 823–865 (2021)
14. Ciliberto, C., Rosasco, L., Rudi, A.: A consistent regularization approach for structured prediction. In: NeurIPS, pp. 4412–4420 (2016)
15. Ciliberto, C., Rosasco, L., Rudi, A.: A general framework for consistent structured prediction with implicit loss embeddings. JMLR **21**(98), 1–67 (2020)
16. Cortes, C., Mohri, M., Weston, J.: A general regression technique for learning transductions. In: ICML, pp. 153–160 (2005)
17. Cortes, C., Mohri, M., Weston, J.: A general regression framework for learning string-to-string mappings. In: Predicting Structured Data (2007)
18. Costa, F., Grave, K.D.: Fast neighborhood subgraph pairwise distance kernel. In: ICML, pp. 255–262 (2010)
19. Deshwal, A., Doppa, J.R., Roth, D.: Learning and inference for structured prediction: a unifying perspective. In: IJCAI (2019)
20. Drineas, P., Mahoney, M.W., Cristianini, N.: On the nyström method for approximating a gram matrix for improved kernel-based learning. JMLR **6**(12) (2005)
21. Edwards, C., Zhai, C., Ji, H.: Text2Mol: cross-modal molecule retrieval with natural language queries. In: EMNLP, pp. 595–607 (2021)
22. El Ahmad, T., Brogat-Motte, L., Laforgue, P., d'Alché-Buc, F.: Sketch in, sketch out: accelerating both learning and inference for structured prediction with kernels (2023)
23. El Ahmad, T., Laforgue, P., d'Alché Buc, F.: Fast kernel methods for generic lipschitz losses via p-sparsified sketches. TMLR (2023)
24. Gärtner, T.: Kernels for Structured Data, Series in Machine Perception and Artificial Intelligence, vol. 72. WorldScientific (2008)
25. Geurts, P., Wehenkel, L., d'Alché Buc, F.: Kernelizing the output of tree-based methods. In: ICML, pp. 345–352 (2006)
26. Graber, C., Meshi, O., Schwing, A.: Deep structured prediction with nonlinear output transformations. In: NeurIPS, vol. 31 (2018)
27. Gygli, M., Norouzi, M., Angelova, A.: Deep value networks learn to evaluate and iteratively refine structured outputs. In: ICML, p. 1341–1351 (2017)
28. Hagberg, A.A., Schult, D.A., Swart, P.J.: Exploring network structure, dynamics, and function using NetworkX. In: Proceedings of the 7th Python in Science Conference, pp. 11–15 (2008)
29. Hastings, J., et al.: ChEBI in 2016: improved services and an expanding collection of metabolites. Nucleic Acids Res. **44**(D1), D1214–D1219 (2016)
30. Huber, P.J.: Robust estimation of a location parameter. Ann. Math. Stat. 73–101 (1964)
31. Jaeger, S., Fulle, S., Turk, S.: Mol2Vec: unsupervised machine learning approach with chemical intuition. J. Chem. Inf. Model. **58**(1), 27–35 (2018)
32. Jumper, J., et al.: Highly accurate protein structure prediction with alphafold. Nature **596**(7873), 583–589 (2021)
33. Kadri, H., Ghavamzadeh, M., Preux, P.: A generalized kernel approach to structured output learning. In: ICML, pp. 471–479 (2013)
34. Kim, S., et al.: PubChem 2019 update: improved access to chemical data. Nucleic Acids Res. **47**(D1), D1102–D1109 (2019)
35. Kim, S., et al.: PubChem substance and compound databases. Nucleic Acids Res. **44**(D1), D1202–D1213 (2016)

36. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR (2015)
37. Kirillov, A., et al.: Segment anything. In: ICCV, pp. 4015–4026 (2023)
38. Korba, A., Garcia, A., d' Alché-Buc, F.: A structured prediction approach for label ranking. In: NeurIPS, vol. 31 (2018)
39. Kpotufe, S., Sriperumbudur, B.K.: Gaussian sketching yields a J-L lemma in RKHS. In: Chiappa, S., Calandra, R. (eds.) AISTATS (2020)
40. Lacotte, J., Pilanci, M.: Adaptive and oblivious randomized subspace methods for high-dimensional optimization: sharp analysis and lower bounds. IEEE Trans. Inf. Theory **68**(5), 3281–3303 (2022)
41. Laforgue, P., Lambert, A., Brogat-Motte, L., d'Alché Buc, F.: Duality in RKHSS with infinite dimensional outputs: application to robust losses. In: ICML, pp. 5598–5607 (2020)
42. LeCun, Y., Chopra, S., Hadsell, R., Ranzato, A., Huang, F.J.: A tutorial on energy-based learning. In: Predicting Structured Data (2006)
43. Lee, J.Y., Patel, D., Goyal, P., Zhao, W., Xu, Z., McCallum, A.: Structured energy network as a loss. In: NeurIPS (2022)
44. Li, Z., Ton, J.F., Oglic, D., Sejdinovic, D.: Towards a unified analysis of random Fourier features. JMLR **22**(108), 1–51 (2021)
45. Mahoney, M.W., et al.: Randomized algorithms for matrices and data. Found. Trends® Mach. Learn. **3**(2), 123–224 (2011)
46. Meanti, G., Carratino, L., Rosasco, L., Rudi, A.: Kernel methods through the roof: handling billions of points efficiently. In: Advances in Neural Information Processing Systems (NeurIPS), vol. 33 (2020)
47. Meanti, G., Chatalic, A., Kostic, V., Novelli, P., Pontil, M., Rosasco, L.: Estimating Koopman operators with sketching to provably learn large scale dynamical systems. In: NeurIPS (2023)
48. Nikolentzos, G., Meladianos, P., Limnios, S., Vazirgiannis, M.: A degeneracy framework for graph similarity. In: IJCAI (2018)
49. Nowak, A., Bach, F., Rudi, A.: Sharp analysis of learning with discrete losses. In: AISTAT (2019)
50. Nowak, A., Bach, F., Rudi, A.: Consistent structured prediction with max-min margin Markov networks. In: ICML (2020)
51. Nowozin, S., Lampert, C.H.: Structured learning and prediction in computer vision. Found. Trends Comput. Graph. Vision **6** (2011)
52. Rahimi, A., Recht, B.: Random features for large scale kernel machines. In: NeurIPS, vol. 20, pp. 1177–1184 (2007)
53. Ralaivola, L., Swamidass, S.J., Saigo, H., Baldi, P.: Graph kernels for chemical informatics. Neural Netw. **18**(8), 1093–1110 (2005)
54. Ramakrishnan, R., Dral, P.O., Rupp, M., von Lilienfeld, O.A.: Quantum chemistry structures and properties of 134 kilo molecules. Sci. Data **1** (2014)
55. Ruddigkeit, L., van Deursen, R., Blum, L.C., Reymond, J.L.: Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17. J. Chem. Inf. Modeli. **52**(11) (2012)
56. Rudi, A., Camoriano, R., Rosasco, L.: Less is more: Nyström computational regularization. In: NeurIPS, vol. 28 (2015)
57. Rudi, A., Rosasco, L.: Generalization properties of learning with random features. In: NeurIPS, pp. 3215–3225 (2017)
58. Schölkopf, B., Smola, A., Müller, K.R.: Kernel principal component analysis. In: ICANN (1997)
59. Seidman, S.B.: Network structure and minimum degree. Soc. Netw. **5**(3), 269–287 (1983)

60. Shervashidze, N., Schweitzer, P., van Leeuwen, E.J., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-Lehman graph kernels. JMLR (2011)
61. Siglidis, G., Nikolentzos, G., Limnios, S., Giatsidis, C., Skianis, K., Vazirgiannis, M.: Grakel: a graph kernel library in python. JMLR (2020)
62. Steinwart, I., Christmann, A.: Sparsity of SVMs that use the epsilon-insensitive loss. In: NeurIPS (2008)
63. Sterge, N., Sriperumbudur, B., Rosasco, L., Rudi, A.: Gain with no pain: efficiency of kernel-PCA by Nyström sampling. In: AISTATS (2020)
64. Sterge, N., Sriperumbudur, B.K.: Statistical optimality and computational efficiency of Nystrom kernel PCA. JMLR **23**(337), 1–32 (2022)
65. Tanimoto, T.: An Elementary Mathematical Theory of Classification and Prediction. International Business Machines Corporation (1958)
66. Tripp, A., Bacallado, S., Singh, S., Hernández-Lobato, J.M.: Tanimoto random features for scalable molecular machine learning. In: NeurIPS (2023)
67. Tu, L., Gimpel, K.: Learning approximate inference networks for structured prediction. In: ICLR (2018)
68. Vaswani, A., et al.: Attention is all you need. In: NeurIPS (2017)
69. Vayer, T., Courty, N., Tavenard, R., Laetitia, C., Flamary, R.: Optimal transport for structured data with application on graphs. In: ICML (2019)
70. Weisfeiler, B., Leman, A.: The reduction of a graph to canonical form and the algebra which appears therein. NTI, Ser. **2**(9), 12–16 (1968)
71. Weston, J., Chapelle, O., Vapnik, V., Elisseeff, A., Schölkopf, B.: Kernel dependency estimation. In: NeurIPS, pp. 897–904. MIT Press (2003)
72. Williams, C., Seeger, M.: Using the Nyström method to speed up kernel machines. In: NeurIPS, vol. 13, pp. 682–688 (2001)
73. Woodruff, D.P.: Sketching as a tool for numerical linear algebra. Found. Trends Theor. Comput. Sci. **10**(1–2), 1–157 (2014)
74. Yang, T., Li, Y.F., Mahdavi, M., Jin, R., Zhou, Z.H.: Nyström method vs random Fourier features: a theoretical and empirical comparison. In: NeurIPS, vol. 25 (2012)
75. Yang, Y., Pilanci, M., Wainwright, M.J., et al.: Randomized sketches for kernels: fast and optimal nonparametric regression. Ann. Stat. **45**(3), 991–1023 (2017)
76. Zhao, W., Zhou, D., Cao, B., Zhang, K., Chen, J.: Adversarial modality alignment network for cross-modal molecule retrieval. IEEE Trans. Artif. Intell. **5**(1) (2024)