Learning Invariant Representations with Kernel Warping

Yingyi Ma **Vignesh Ganapathiraman** **Xinhua Zhang**
Department of Computer Science, University of Illinois at Chicago

Abstract

Invariance is an effective prior that has been extensively used to bias supervised learning with a *given* representation of data. In order to learn invariant representations, wavelet and scattering based methods "hard code" invariance over the *entire* sample space, hence restricted to a limited range of transformations. Kernels based on Haar integration also work only on a *group* of transformations. In this work, we break this limitation by designing a new representation learning algorithm that incorporates invariances *beyond transformation*. Our approach, which is based on warping the kernel in a data-dependent fashion, is computationally efficient using random features, and leads to a deep kernel through multiple layers. We apply it to convolutional kernel networks and demonstrate its stability.

1 Introduction

A broad range of machine learning problems naturally exhibit invariances that can be effectively leveraged as an informative prior. Spam filters are supposed to be stable under feature deletions, additions, and replacements [1], while detectors in image processing and computer vision are expected to deliver invariant response under transformations such as translation, rotation, scaling, etc [2, 3]. Manifold priors assume flat curvature or gradient of discriminant function in the vicinity of data samples [4–6], or in the direction of nearest-neighbor instances on a graph [7–9].

To model invariance, a large body of exiting methods first specify a *given* space of prediction functions and the representation of data (e.g., through the selection of kernels), and then use invariances to bias the search for the optimal predictor through the loss function and the regularizer. For example, the virtual sample approach, *a.k.a.* data augmentation [10], explicitly adds perturbed examples into the training set. Distributional robustness [11] tries to retain low loss when the *distribution* of training data is perturbed in the most adverse fashion. Besides extending the loss function, regularizers can also be augmented using, e.g., the tangent distance that enforces smoothness on the parametric trajectory of transformations such as rotation with a range of angles [3].

The major challenge here lies in the significant computational cost arising from the complexity of invariances. The virtual sample approach directly multiplies the training set size by the number of invariances. Distributional robustness often leads to intractable probabilistic inference tasks, and even in special classes where perturbations do yield a closed-form optimization objective, they are usually second order cone programming and semi-definite programming, which are still hard to scale in practice [12, 13]. The tangent distance regularizer typically leads to nonconvex problems even if the original risk minimization problem is convex.

To address these issues, approximate approaches have been studied. Sparse approximation that greedily finds the most violated invariance enjoys efficient quadratic programming [14], and it often outperformed virtual samples. However, it relies on the tractability of finding the most violated invariance, which significantly confined its applicability. In [15, 16], a key insight was taken that a variety of invariances can be *compactly* encoded as bounded linear functionals in a rich family of functions like reproducing kernel Hilbert spaces (RKHS). A representer theorem was then established, facilitating an efficient *convex* quadratic programming.

However, all these methods assume a *given* representation of data or function space, while the recent wisdom in deep learning reveals considerable benefit of *learning* the representations. Many deep architectures such as convolutional neural networks (CNNs) intrinsically enforce invariances via techniques like pooling and scattering transform [17, 18]. However these methods are all *parametric*, aiming to design models that induce invariance across the entire domain of samples. Although this is desirable, it also poses significant challenge in

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

model design and as a result many are restricted to translational invariance $u \mapsto u - \tau(u)$ for some *diffeomorphism* τ.

A similar parametric approach, but designed to gain convexity in learning, is to re-engineer kernel functions to make them invariant to a *group* of transformations [19, 20], and a popular class is the Haar integration kernel [21–23]. However, they do not allow multi-layer transformations and do not admit closed-form evaluations. Although finite approximation is possible, it is expensive, requiring $O(d/\epsilon^2)$ samples of transformation where d is the dimensionality of the underlying space [21], in addition to $O(d/\epsilon^2)$ samples to approximate the spectrum of translation-invariant kernels.

Finally, the restriction to a group of transformation precludes important invariances. In certain kinds of images such as persons, changes in pose and facial expressions may not form a group [24, §1.6]. Further, invariance may encompass useful priors beyond transformation. For example, flatness in differentiations such as directional derivative and curvature, graph Laplacian, and spatial smoothness under filtering (convolution with an origin-centered distribution). It is unclear how existing Haar kernel based methods can incorporate such priors.

The goal of this paper, therefore, is to develop a representation learning algorithm for this broadened range of invariance priors, while at the same time retaining the computational and spacial efficiency (§3). By using random features [25, 26] introduced in §3.1, such a representation can be transformed through multiple layers, yielding a deep kernel that offers the favorable decoupling between invariance modeling and the subsequent supervised learning. Our tool is a data-dependent warping of the RKHS based on the Riesz representer of invariances [15, 16], which also renders lower empirical Rademacher complexity for improved generalization.

We applied this technique to the Convolutional Kernel Network [CKN, 27–29] in §4, overcoming the gap between patch-level kernel and image-level invariance by using the direct finite approximation technique without going through the involved image-level kernel, an important merit of using the representers for bounded linear operators. We also established the stability of the multi-layer CKN with warped kernel (§5), and experimental results in §6 confirm the effectiveness of the warping technique. Finally, it is noteworthy that our invariance differs from equivariance or covariance [30], which concerns the commutativity of signal processing operations and image transformations.

2 Preliminaries

Suppose we have training examples $\{(x_i, y_i)\}_{i=1}^l$ where x_i's are the features lying in a domain $\mathcal{X}$, and y_i's are the labels. A *positive semi-definite* (PSD) kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ satisfies that for all $l \in \mathbb{N}$ and all $x_1, \ldots, x_l \in \mathcal{X}$, the Gram matrix $K := (k(x_i, x_j))_{ij} \in \mathbb{R}^{l \times l}$ is symmetric PSD. Commonly used kernels are on inner product spaces, such as polynomial kernel of degree r: $k(x_1, x_2) = (\langle x_1, x_2 \rangle + 1)^r$, and Gaussian kernels: $k(x_1, x_2) = \kappa_\sigma(x_1 - x_2)$ where $\kappa_\sigma(x) := \exp(-\|x\|^2 / (2\sigma^2))$.

Given a PSD kernel, a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ can be constructed with functions from $\mathcal{X}$ to $\mathbb{R}$. A feature mapping $\varphi : \mathcal{X} \to \mathcal{H}$ is defined as $\varphi(x) = k(x, \cdot)$. Many learning algorithms seek a function $f \in \mathcal{H}$ such that for an example x, the prediction $f(x) = \langle f, \varphi(x) \rangle$ is consistent with the given label. Comprehensive introductions can be found in [31–33].

Linear operators. A linear operator T from a normed space $\mathcal{V}$ to another normed space $\mathcal{W}$ has an operator norm defined as $\|T\| := \sup_{x \in \mathcal{V}, \|x\|_{\mathcal{V}} = 1} \|Tx\|_{\mathcal{W}}$. When it is finite, we call it a bounded linear operator. T is called a functional if $\mathcal{W} = \mathbb{R}$. For example, given an $x \in \mathcal{X}$, the evaluation operator from an RKHS $\mathcal{H}$ to $\mathbb{R}$ defined as $T(f) = f(x)$ is bounded because $|f(x)| = |\langle f, \varphi(x) \rangle_{\mathcal{H}}| \leq \|f\|_{\mathcal{H}} k(x, x)^{\frac{1}{2}}$. Indeed, RKHS is a Hilbert space with a bounded linear evaluation functional [34].

Bounded linear functionals on a Hilbert space $\mathcal{V}$ are particularly important because the Riesz representation theorem guarantees that any such a functional L can be represented by a unique $z \in \mathcal{H}$ such that $L(f) = \langle f, z \rangle$ for all $f \in \mathcal{V}$, and the Hilbert space norm of z is equal to $\|L\|$. This z can be evaluated by $z(x) = \langle z, k(x, \cdot) \rangle = L(k(x, \cdot))$ for all $x \in \mathcal{X}$, and their inner product can also be computed efficiently. These results play a fundamental role in our framework, allowing us to compactly represent functionals related to local invariances as elements of the RKHS.

2.1 Invariances as bounded linear operators

Although the most commonly used functional in nonparametric learning is the point evaluation for empirical risk minimization (ERM), other bounded linear functionals have also been used, e.g., to model local invariances [15, 35]. This has been particularly useful for semi-supervised learning where unlabeled data provides such invariance information in large supply. For example, the graph Laplacian [8, 36] has been shown very effective in leveraging unlabeled data. In general, ERM can be augmented with regularizers $\ell(L_i(f))$, where L_i is the transformation operator (e.g., $f(x) - f(y)$ for nearby x and y), and ℓ is the loss on it (e.g., to encourage smoothness by penalizing large deviation):

$$\min_{f \in \mathcal{H}} \frac{\|f\|_{\mathcal{H}}^2}{2} + \lambda \sum_{j=1}^{l} \ell_1(f(x_j), y_j) + \nu \sum_{i=1}^{m} \ell_2\left(L_i(f)\right). \quad (1)$$

[15, 35] showed that (1) enjoys a representer theorem, with the optimal function lying in the span of evaluation representers *and* the new representers z_i for L_i.

We next review some useful local invariances that can be modeled as bounded linear functionals. By Theorem 2.7-8 of [34], all linear functionals on *finite* dimensional normed space are bounded. This includes all linear and polynomial kernels. Our main interest, however, lies in infinite dimensional RKHS, where boundedness calls for more refined analysis depending on the specific L_i.

Graph Laplacian. Given the similarity measure w_{ij} between data points x_i and x_j, the graph Laplacian regularizer [7, 8, 36] can be written as $\sum_{ij} w_{ij}(f(x_i) - f(x_j))^2 = \sum_{ij} w_{ij}(L_{ij}(f))^2$, where $L_{ij}(f) = \langle f, k(x_i,\cdot) - k(x_j,\cdot)\rangle$ is linear and bounded.

Transformation invariance. Invariance to known local transformations is a commonly used prior for image processing [3] and natural language processing [14]. A signal such as an image can be represented by a function from the space of spatial coordinates $\Omega \subseteq \mathbb{R}^d$ to a Hilbert space $\mathcal{H}$ [3]. When its squared norm $\int_\Omega \|g(u)\|^2_{\mathcal{H}}\,\mathrm{d}u$ is finite, we write $g \in L^2(\Omega, \mathcal{H})$, which also forms a Hilbert space. Discrete images can be extended to continuous by convolution [3, § 2.3]. For example, an image can be considered as a function g that maps points in the plane $\Omega = \mathbb{R}^2$ to the intensity of the image at that point. Next, we consider a parametrized family of diffeomorphisms (differentiable bijective transformations with differentiable inverse) $t_{\boldsymbol{\alpha}} : \Omega \to \Omega$, which is also differentiable in the parameter $\boldsymbol{\alpha}$. For example, $t_{\boldsymbol{\alpha}}$ may mean translating by an offset (α_x, α_y), rotating by an angle α, and scaling by an amount α. They can be respectively formalized by

$$\begin{pmatrix} x \\ y \end{pmatrix} \xmapsto{t_{\boldsymbol{\alpha}}} \begin{pmatrix} x+\alpha_x \\ y+\alpha_y \end{pmatrix} \text{ or } \begin{pmatrix} x\cos\alpha - y\sin\alpha \\ x\sin\alpha - y\cos\alpha \end{pmatrix} \text{ or } \begin{pmatrix} x+\alpha x \\ y+\alpha y \end{pmatrix}.$$

These transformations induce a family of operators $T_{\boldsymbol{\alpha}} : \mathbb{R}^\Omega \to \mathbb{R}^\Omega$ as $T_{\boldsymbol{\alpha}}(g) = g \circ t^{-1}_{\boldsymbol{\alpha}}$, meaning the intensity of $T_{\boldsymbol{\alpha}}(g)$ at $u \in \Omega$ is $g(t^{-1}_{\boldsymbol{\alpha}}(u))$. In practice images are finite samples from Ω, but to present our core ideas concisely, we stick with continuous function g and extension to discretization is straightforward. Clearly, $f \mapsto f(g) - f(T_{\boldsymbol{\alpha}}(g))$ is linear and bounded.

Differentiation functional. Another common prior postulates that the discriminant function f changes mildly around sampled points, i.e., the norm of the gradient is small at these locations. Assuming differentiability as appropriate, we are interested in linear functionals $L_{\mathbf{x}_i,d}(f) := \frac{\partial f(\mathbf{x})}{\partial x^d}|_{\mathbf{x}=\mathbf{x}_i}$, where x^d is the d-th component of the vector $\mathbf{x}$. By [33, Corollary 4.36], this $L_{\mathbf{x}_i,d}$ must be bounded, and so are the higher order partial derivatives. In fact, $\|L_{\mathbf{x}_i,d}\| = \sigma^{-1}$ for Gaussian kernel κ_σ. Similarly directional derivative $f \mapsto \langle v, \nabla f(x_i)\rangle$ is also linear and bounded, where v could be a k-nearest neighbor direction. Furthermore, the following result from [15] empowers bounded linear operators to model transformations via directional derivatives in the parameters $\boldsymbol{\alpha}$,

Theorem 1. *Let $\mathcal{F}$ be a normed vector space of functions that map from the range of $T_{\boldsymbol{\alpha}}(g)$ to $\mathbb{R}$. Suppose the linear functional that maps $f \in \mathcal{F}$ to $\frac{\partial f(\mathbf{u})}{\partial u^j}\big|_{\mathbf{u}=\mathbf{u}_0}$ is bounded for any $\mathbf{u}_0$ and coordinate j. Then the derivative functional $L_{g,d}$: $f \mapsto \frac{\partial}{\partial \alpha^d}\big|_{\boldsymbol{\alpha}=\mathbf{0}} f(T_{\boldsymbol{\alpha}}(g))$ is linear and must be bounded on $\mathcal{F}$.*

By chain rule, $T_{\boldsymbol{\alpha}=\mathbf{0}}(g) = g$ implies $\frac{\partial}{\partial \alpha^d}\big|_{\boldsymbol{\alpha}=\mathbf{0}} f(T_{\boldsymbol{\alpha}}(g)) = \big\langle \nabla f(g), \nabla_{\alpha^d}\big|_{\boldsymbol{\alpha}=\mathbf{0}} T_{\boldsymbol{\alpha}}(g)\big\rangle$. The second argument represents the perturbation of the image under small changes of α^d. In practice, we can approximate it by the finite difference of the image under, say, rotation by one degree or shifting by one pixel. More generally we can consider the directional derivative operator $f \mapsto \langle \nabla f(g), \mathbf{v}\rangle$ for $\mathbf{v} \in L^2(\Omega, \mathcal{H})$, and we will use this operator in the sequel as a running example.

Local averaging. Another linear operator that effectively characterizes the smoothness of the image around a point y is by convolution with a distribution ρ on the space of $\mathcal{X}$, centered and peaked at zero: $f \mapsto \int_{\mathcal{X}} f(\tau)\rho(y-\tau)\,\mathrm{d}\tau - f(y)$. [15] also showed that it is bounded provided $\sup_{x\in\mathcal{H}} k(x,x) < \infty$.

3 Data-Dependent Kernel Warping for Modeling Invariance

Although the ERM framework in (1) is convenient to optimize, it does not provide novel representations of data that account for the invariances. To address this problem, our key observation is that when the ℓ_2 in (1) is the squared loss, we can construct a new RKHS whose norm deforms $\|f\|_{\mathcal{H}}$ in a way that favors the invariances $L_i(f)$. Indeed, we "define" a new RKHS $\mathcal{H}^\circ$, such that $\|f\|^2_{\mathcal{H}^\circ} = \|f\|^2_{\mathcal{H}} + \sum_{i=1}^m \langle z_i, f\rangle^2_{\mathcal{H}}$ (letting $\nu = 1$, and in practice, ν can be tuned as a hyperparameter). To show this induces a well-defined RKHS, we have the following theorem whose proof is in Appendix **??**.

Theorem 2. *Let $\mathcal{H}^\circ$ consist of the same set of functions in $\mathcal{H}$. Define the new inner product as*

$$\langle f, g\rangle_{\mathcal{H}^\circ} = \langle f, g\rangle_{\mathcal{H}} + \sum\nolimits_i \langle f, z_i\rangle_{\mathcal{H}} \langle g, z_i\rangle_{\mathcal{H}}\,. \quad (2)$$

Let $K_z = (\langle z_i, z_j\rangle_{\mathcal{H}})_{i,j=1}^m \in \mathbb{R}^{m\times m}$. Then $\mathcal{H}^\circ$ is an RKHS with the kernel function

$$k^\circ(x_1,x_2) = k(x_1,x_2) - z(x_1)^\top (I + K_z)^{-1} z(x_2),$$
$$\text{where} \quad z(x) = (z_1(x), \ldots, z_m(x))^\top. \quad (3)$$

In essence, the new kernel k° "warps" the original kernel k by accounting for the invariances z_i. The new RKHS makes *no change* to the space of functions, but instead warps the norm to adjust the preference over

these functions. In contrast to data-independent priors that enforce invariance over the *entire* sample space, our approach here is data dependent, enforcing invariances only around the observed training examples. The resulting kernel is similar to the cloud point kernel [37], but the latter is restricted to graph Laplacian only, while our framework flexibly models a much broader range of linear operators (and a completely new proof). Due to the fundamentality of our kernel warping and its essential role in the subsequent enhancement of convolutional kernel networks, we highlight several remarks.

Firstly, given a dataset $\{x_i\}_{i=1}^n$ with feature representation $\varphi(x_i) := k(x_i, \cdot)$, Theorem 2 guarantees that the Gram matrix $K^\circ := (k^\circ(x_i, x_j))_{i,j}$ can be written as follows by using (3), and consequentially K° is PSD:

$$\begin{aligned} K^\circ &= \Phi^\top\Phi - \Phi^\top Z(I + Z^\top Z)^{-1}Z^\top\Phi \\ &= \Phi^\top(I + ZZ^\top)^{-1}\Phi, \end{aligned} \tag{4}$$

where $\Phi = (\varphi(x_1), \ldots, \varphi(x_n))$, $Z = (z_1, \ldots, z_m) \in \mathcal{H}^m$. The second equality in (4) imitates the Woodbury matrix identity [38] and immediately establishes that K° is PSD. But it is only meant to illustrate intuitions and is not rigorous, because the columns of Z are functions in $\mathcal{H}$ instead of vectors.

Secondly, the new norm $\|f\|_{\mathcal{H}^\circ}^2 = \|f\|_{\mathcal{H}}^2 + \sum_i \langle z_i, f\rangle_{\mathcal{H}}^2$ can be justified by the fact that the empirical Rademacher complexity of $\mathcal{H}^\circ$ is $O(\sqrt{\operatorname{tr} K^\circ})$, and the generalization error depends on this complexity linearly [39]. By warping the kernel, the trace of K° is lower than that of K by $\sum_i z(x_i)(I + K_z)^{-1}z(x_i)$, hence reducing the generalization error. Although the unit ball of $\|\cdot\|_{\mathcal{H}^\circ}$ is smaller than that of $\|\cdot\|_{\mathcal{H}}$ (higher bias), the prior that the classifier f should render small values of $|\langle z_i, f\rangle_{\mathcal{H}}|$ means using the former ball does not effectively miss those hypotheses of interest.

Thirdly, the objective (1) can be rewritten as the standard ERM in $\mathcal{H}^\circ$ *without* explicit invariance regularizers: $\min_{f\in\mathcal{H}^\circ} \frac{\|f\|_{\mathcal{H}^\circ}^2}{2} + \lambda\sum_{j=1}^l \ell_1(\langle f, k^\circ(x_j, \cdot)\rangle_{\mathcal{H}^\circ}, y_j)$. This means x_j finds a new *representation* $\varphi^\circ(x_j) := k^\circ(x_j, \cdot)$ in $\mathcal{H}^\circ$. The mapping of $x \mapsto \varphi^\circ(x)$ can be decomposed into two steps: 1) $x \mapsto \varphi(x)$ as induced by the kernel k; and 2) a mapping $W : \mathcal{H} \to \mathcal{H}^\circ$ as $\sum_i \alpha_i\varphi(x_i) \mapsto \sum_i \alpha_i\varphi^\circ(x_i)$. Importantly, W is well defined, i.e., independent of the decomposition.

Theorem 3. *Define $W : \mathcal{H} \to \mathcal{H}^\circ$ by $\sum_i \alpha_i\varphi(x_i) \mapsto \sum_i \alpha_i\varphi^\circ(x_i)$, with standard completion in respective spaces. Then for any $\{\alpha_i, x_i\}$ and $\{\beta_j, y_j\}$, $\sum_i \alpha_i\varphi(x_i) = \sum_j \beta_j\varphi(y_j)$ implies $\sum_i \alpha_i\varphi^\circ(x_i) = \sum_j \beta_j\varphi^\circ(y_j)$.*

The proof is relegated to Appendix **??**. In general, W is not an identity mapping, but it is clearly linear. We call W a "*warping operator*" and denote it as W_Z whenever the set Z needs to be emphasized. We will keep W abstract and in computation only its finite approximation will be used (§3.1). But clearly W is not expansive, because by (3), $\|\varphi^\circ(x)\|_{\mathcal{H}^\circ}^2 = k^\circ(x, x) = k(x, x) - z(x)^\top(I + K_z)^{-1}z(x) \le \|\varphi(x)\|_{\mathcal{H}}^2$. See a more detailed proof in Appendix **??** (Property **??**). W is *not* contractive either as $\|\varphi^\circ(x)\|_{\mathcal{H}^\circ} = \|\varphi(x)\|_{\mathcal{H}}$ if $z_i(x) = 0$.

3.1 Finite-dimensional approximation of invariance and reproducing representers

Approximating nonlinear kernels by finite dimensional features (a.k.a. kernel linearization) has been popularly used to scale up kernel machines [25, 26]. It also underpins multi-layer kernel feature transformation [27]. We next show that our new invariance representers admit natural finite approximations (FAs) based on Fourier decomposition and RKHS subspace projection.

FA with Fourier approximation. Consider a shift-invariant kernel $k(x, y) := \kappa(x - y)$ where $x, y \in \mathbb{R}^d$ [26]. k is a positive definite kernel iff the Fourier transform of κ has all its coefficients being nonnegative. Bochner's theorem [40] guarantees that the Fourier transform $q(\omega)$ of κ is a proper probability distribution, and moreover

$$k(x, y) = \int q(\omega)e^{j\omega^\top(x-y)}\,\mathrm{d}\omega = \mathop{\mathbb{E}}_{\omega\sim q}[\xi_\omega(x)^\top\xi_\omega(y)], \tag{5}$$
$$\text{where} \quad \xi_\omega(x) := (\cos(\omega^\top x), \sin(\omega^\top x))^\top.$$

Given samples $\{\omega_i\}_{i=1}^p$ drawn i.i.d. from q, we can approximate $k(x, y)$ by $\langle\tilde\varphi(x), \tilde\varphi(y)\rangle$, where

$$\begin{aligned} \tilde\varphi(x) &= \{p^{-1/2}\xi_{\omega_i}(x)\}_{i=1}^p \\ &= \{p^{-1/2}[\cos(\omega_i^\top x), \sin(\omega_i^\top x)]\}_{i=1}^p \in \mathbb{R}^{2p}. \end{aligned} \tag{6}$$

We will call $\xi_{\omega_i}(x)$ a *channel* corresponding to ω_i.

Differentiation invariance. The FA of invariance representers z_i introduced in §2.1 can be similarly derived. Ideally, its inner product with the FA of $k(x, \cdot)$ should approximate $z_i(x)$. For example, consider the $z_{y,v}$ for $f \mapsto \langle v, \nabla f(y)\rangle$ with given v and y. Then $z_{y,v}(x) = v^\top\frac{\partial}{\partial y}k(x, y) = -v^\top\nabla\kappa(x - y)$. As the Fourier transform of $\nabla\kappa$ is $j\omega q(\omega)$, we get

$$\begin{aligned} z_{y,v}(x) &= -\int q(\omega)j\cdot\omega^\top v\cdot e^{j\omega^\top(x-y)}\,\mathrm{d}\omega \\ &= \mathop{\mathbb{E}}_{\omega\sim q}[-\omega^\top v\cdot\xi_\omega(x)^\top\xi_\omega(y)] \approx \langle\tilde z_{y,v}, \tilde\varphi(x)\rangle \end{aligned}$$
$$\text{where} \quad \tilde z_{y,v} := \{p^{-1/2}\omega_i^\top v\cdot[-\sin(\omega_i^\top y), \cos(\omega_i^\top y)]\}_{i=1}^p \in \mathbb{R}^{2p}, \text{ and } \tilde\varphi(x) \text{ is as in (6).} \tag{7}$$

So we have derived an FA of $z_{y,v}$ by using the same set of channels $\{\omega_i\}$ as employed by $\tilde\varphi(x)$.

Remark 1 (General recipe for FA of invariance)**.** *In hindsight, the result for gradient invariance is no surprising because given the FA of k in* (5)*, it follows*

$$z_{y,v}(x) = \left\langle v, \frac{\partial}{\partial y} k(x,y) \right\rangle = \left\langle v, \frac{\partial}{\partial y} \mathop{\mathbb{E}}_{w\sim q} [\xi_\omega(x)^\top \xi_\omega(y)] \right\rangle$$
$$= \mathop{\mathbb{E}}_{w\sim q} \left[\xi_\omega(x)^\top (\nabla \xi_\omega(y) \cdot v) \right], \quad (8)$$

where $\nabla\xi_\omega(y)$ is the Jacobian matrix. Clearly $\frac{\partial}{\partial y}\langle v, \xi_\omega(y)\rangle$ is exactly the $\tilde{z}_{y,v}$ we obtained above for gradient invariance. This convenient derivation shows that there is little difficulty in finding the FA of invariance, given that we know how to approximate $k(x,y)$.

FA with subspace projection. Another way to linearize kernels, as adopted by [25, 28, 41], is to compute feature representations by projecting to a finite subspace of the RKHS. We will refer to it as *projection FA*. Assuming the projection bases are $\{\omega_i\}_{i=1}^p$ and letting $G := (\omega_1, ..., \omega_p)$, the projection FA of evaluation representer for Gaussian kernel κ_σ can be written as

$$\tilde{\varphi}(x) = \{\tilde{\varphi}_{\omega_i}(x)\} = \kappa_\sigma(G^\top G)^{-1/2} \kappa_\sigma(G^\top x) \in \mathbb{R}^p, \quad (9)$$

where κ_σ is applied elementwise and $X^{-1/2}$ is the inverse square-root matrix of X. By Remark 1, the projection FA of gradient invariance is:

$$\tilde{z}_{y,v} = \{v^\top \nabla \tilde{\varphi}_{\omega_i}(y)\}_{i=1}^p = \tfrac{1}{\sigma^2} \kappa(G^\top G)^{\frac{-1}{2}} \{\kappa(\omega_i^\top y)\omega_i^\top v\}_{i=1}^p.$$

Local averaging. The local averaging operator z_y is $z_y(x) = \int_{\mathbb{R}^d} \kappa(x-u)\rho(y-u)\,\mathrm{d}u - \kappa(x-y)$. To derive its projection FA, recall that the Fourier transform of the convolution of two functions is the product of their Fourier transforms. Letting $q(\omega)$ and $\hat{\rho}(\omega)$ be the Fourier transforms of κ_σ and ρ resp, we derive:

$$z_y(x) = \int_{\mathbb{R}^d} q(\omega)(\hat{\rho}(\omega) - 1) e^{j\omega^\top (x-y)}\,\mathrm{d}\omega$$
$$= \mathop{\mathbb{E}}_{\omega\sim q} [(\hat{\rho}(\omega) - 1)\xi_\omega(x)^\top \xi_\omega(y)] \approx \langle \tilde{z}_y, \tilde{\varphi}(x)\rangle,$$
$$\text{where } \tilde{z}_y := \{(\hat{\rho}(\omega)-1)[\cos(\omega_i^\top y), \sin(\omega_i^\top y)]\}_{i=1}^p. \quad (10)$$

Again the Fourier FA uses the same ω_i as in $\tilde{\varphi}(x)$. We can also derive it by substituting with (5):

$$z_y(x) = \int_{\mathbb{R}^d} k(x,u)\rho(y-u)\,\mathrm{d}u - k(x,y) \quad (11)$$
$$= \mathop{\mathbb{E}}_{w\sim q} \left[\xi_\omega(x)^\top \left(\int_{\mathbb{R}^d} \xi_\omega(u)\rho(y-u)\,\mathrm{d}u - \xi_\omega(y) \right) \right].$$

Specializing $k(x,y)$ into $\kappa_\sigma(x-y)$, we easily extract the approximation $\tilde{z}_y$ as in (10).

3.2 FA of the warping operator

Given the invariance representers $z_i \in \mathcal{H}$ and Fourier samples $\{\omega_i\}$ for approximating the kernel k, we ultimately need to find an FA for a given function $f^\circ = Wf \in \mathcal{H}^\circ$. The most straightforward way is to take two steps. First, follow the approach in §3.1 to compute the FA of f and z_i from $\mathcal{H}$, denoted as $\tilde{f}$ and $\tilde{z}_i$ respectively. Then, by (4), the FA of f° can be computed by $\tilde{f}^\circ = (I + \tilde{Z}\tilde{Z}^\top)^{-1/2}\tilde{f}$, where $\tilde{Z} = (\tilde{z}_1, \ldots, \tilde{z}_n)$. To gain a more in-depth insight, let the nonzero eigenvalues of $\tilde{Z}\tilde{Z}^\top$ be $\sigma_1, \sigma_2, \ldots$, with eigenvectors $u_1, u_2, \ldots$. Let $\{v_j\}$ be an orthonormal basis of the null space of $\tilde{Z}\tilde{Z}^\top$. Then it follows that

$$\tilde{f}^\circ = \left(\sum\nolimits_i (1+\sigma_i^2) u_i u_i^\top + \sum\nolimits_j v_j v_j^\top \right)^{-1/2} \tilde{f}$$
$$= \left(\sum\nolimits_i (1+\sigma_i^2)^{-1/2} u_i u_i^\top + \sum\nolimits_j v_j v_j^\top \right) \tilde{f}$$
$$= \tilde{f} - \sum\nolimits_i (1 - (1+\sigma_i^2)^{-1/2})(u_i^\top \tilde{f}) u_i, \quad (12)$$

which can be easily approximated by using the top t singular values. Given these u_i after SVD on $\tilde{Z}$, we can compute $\tilde{\varphi}^\circ(x_i)$ for *all* i efficiently. Furthermore, $\{\omega_i\}$ can be learned in a supervised fashion jointly with the prediction model, and this end-to-end approach is detailed in Appendix **??**.

4 Warping Kernels for Convolutional Kernel Networks

In [27, 28], the convolutional kernel network (CKN) was proposed. Here we recap the description given by [29], and demonstrate how kernel warping can be seamlessly incorporated in CKN to achieve data-dependent transformation invariances.

Let the domain of the signal be $\Omega \subseteq \mathbb{R}^d$, which can be a two-dimensional grid $[0,1]^2$ with $d = 2$. The signal x_0 lies in $L^2(\Omega, \mathcal{H}_0)$, where typically $\mathcal{H}_0 = \mathbb{R}^3$ and $x_0(u)$ is the RGB value at location $u \in \Omega$. Then we transform x_0 through multiple layers of feature maps into RKHSs; $\mathcal{H}_1$, $\mathcal{H}_2$, etc, resulting in image representations $x_1 \in L^2(\Omega, \mathcal{H}_1)$, $x_2 \in L^2(\Omega, \mathcal{H}_2)$, etc.

Patch extraction operator Given a layer x_{k-1}, a patch $\mathcal{P}_k$ for layer k is defined as a mapping from a compact patch shape $S_k \subseteq \Omega$ to the feature representation $\mathcal{H}_{k-1}$, i.e., $\mathcal{P}_k := L^2(S_k, \mathcal{H}_{k-1})$. Naturally, given an image and a location $u \in \Omega$, we define a patch around u as

$$P_k x_{k-1}(u) = (v \mapsto x_{k-1}(u+v))_{v\in S_k} \in \mathcal{P}_k. \quad (13)$$

Note P_k is an operator from $L^2(\Omega, \mathcal{H}_{k-1})$ to $L^2(\Omega, \mathcal{P}_k)$, acting on x_{k-1} rather than $x_{k-1}(u)$.

Kernel mapping operator Given a patch in $\mathcal{P}_k$, we next transform its feature representation to an RKHS $\mathcal{H}_k$ induced by a kernel K_k with feature map $\varphi_k : \mathcal{P}_k \to \mathcal{H}_k$. Formally, we introduce a point-wise operator M_k such that for all $u \in \Omega$:

$$M_k P_k x_{k-1}(u) = \varphi_k(P_k x_{k-1}(u)) \in \mathcal{H}_k. \quad (14)$$

Pooling operator Finally, the next layer x_k is constructed by applying a pooling operation to enforce local shift-invariance. A linear convolution operator A_k is defined based on a Gaussian filter of scale σ_k with $h_{\sigma_k}(u) := \sigma_k^{-d} h(\frac{u}{\sigma_k})$ where $h(u) = (2\pi)^{-\frac{d}{2}} \exp(-\|u\|^2)$.

For all $u \in \Omega$, we define

$$x_k(u) = A_k M_k P_k x_{k-1}(u) \text{ (then use Bochner integral)}$$
$$= \int_{\mathbb{R}^d} h_{\sigma_k}(u-v) M_k P_k x_{k-1}(v)\, \mathrm{d}v \in \mathcal{H}_k. \quad (15)$$

Chaining the operators together over n layers, we get $x_n = \Phi_n(x_0) = A_n M_n P_n A_{n-1} M_{n-1} P_{n-1} \dots A_1 M_1 P_1 x_0$.

Output layer The final prediction layer is based on x_n and it can in turn employ a linear kernel or a Gaussian kernel $\kappa_\sigma(x_n - x'_n)$, amongst others. With this kernel, any loss function can be utilized that is suitable for the application problem, e.g. hinge loss. Note here the norm of x_n is based on the *image-level* inner product over $L^2(\Omega, \mathcal{H}_n)$, defined as (similarly for x_k of other layers):

$$\langle x_n, x'_n \rangle := \langle x_n, x'_n \rangle_\Omega := \int_{u\in\Omega} \langle x_n(u), x'_n(u) \rangle_{\mathcal{H}_n} \mathrm{d}u. \quad (16)$$

4.1 Incorporating Invariance to CKNs

Although CKN enjoys intrinsic invariance to translations [29], it would be desirable to instill other invariances. Achieving this with parametric forms of feed-forward layers (i.e. infinitely strong prior) is difficult in general, and so we naturally resort to the data-dependent approach based on kernel warping. However, kernels in CKN are defined over patches, while invariances apply to the entire image. It does not make sense to, say, rotate each patch individually. Despite this obstacle, our key contribution in this paper is to show that patch-wise kernels can be used to efficiently model invariance at image level.

Our key insight is that images x_k lie in the domain $L^2(\Omega, \mathcal{H}_k)$, which, under the inner product in (16), trivially forms an RKHS on $L^2(\Omega, \mathcal{H}_{k-1})$ (the domain of x_{k-1}). Its kernel function just tiles the kernel of $\mathcal{H}_k$ over all $u \in \Omega$, and we denote it as k_Ω. Therefore we can apply the warping operator on $(L^2(\Omega, \mathcal{H}_k), k_\Omega)$ to incorporate invariances *at the level of images*. Specifically, suppose at layer k we would like to introduce invariances represented by $z_i^k \in (L^2(\Omega, \mathcal{H}_k), k_\Omega)$ (the same domain as $M_k P_k x_{k-1}$). Letting $Z_k := \{z_1^k, \dots, z_m^k\}$, we can warp $M_k P_k x_{k-1}$ by $W_{Z_k} : (L^2(\Omega, \mathcal{H}_k), k_\Omega) \to (L^2(\Omega, \mathcal{H}_k), k_\Omega^\circ)$, where k_Ω° is the kernel of the warped RKHS. Recall warping does not change the set of functions in the RKHS (Theorem 2); it only changes the inner product and kernel function. We will keep k_Ω° implicit in this paper, and write W_k as a shorthand for W_{Z_k}. This leads to

$$x_k := A_k W_k y_k \in L^2(\Omega, \mathcal{H}_k),$$
$$\text{where} \quad y_k := M_k P_k x_{k-1} \in L^2(\Omega, \mathcal{H}_k). \quad (17)$$

Here we suppressed k_Ω and k_Ω° in the notation as the context is clear. Different layers can naturally employ different numbers of invariances, but to lighten the notation we tie them to m. It would be interesting if $A_k W_k M_k$ could be equivalently written as $A_k \bar{M}_k$ with some new kernel mapping operator $\bar{M}_k$ that *acts on patches*. Unfortunately, this is infeasible because the warping operator W_k is global, coupling all the patches.

Choice of invariances Z_k. The most straightforward way of choosing Z_k is to propagate all the raw training examples up to layer k, and then consider the union of transformation invariances based on these x_{k-1} (e.g., directional derivative, rotation, etc). This leads to a large number of invariances, posing significant challenges in computation. A simplified approach is that for each training example, Z_k only encompasses the transformation invariances based on the value of x_{k-1} *for that particular example*. This leads to a much smaller set of Z_k for each example, while, overall, we still exhaustively cover all the invariances on all training examples. We will refer to it as *sequential invariance*.

4.2 FA of kernel invariances and warping

Finally, both W_k (on images) and M_k (on patches) require finite approximations to keep computations tractable, and as we discussed in §3, this can be achieved for M_k by using the Fourier bases $B_k := \{\omega_1^k, \omega_2^k, \dots\}$ at layer k. A similar technique for W_k will be discussed shortly. The output layer also has a "finite" weight O, leading to a loss $\ell(\langle O, \tilde{x}_n \rangle)$ using Euclidean inner product.

Suppose the FA of Z_k is available as $\tilde{Z}_k := (\tilde{z}_1^k, \dots, \tilde{z}_m^k)$, then the FA of x_k as the result of warping the FA of y_k is $\tilde{x}_k = (I + \tilde{Z}_k \tilde{Z}_k^\top)^{-1/2} \tilde{y}_k$ by (12). This multiplication can be performed efficiently if $\tilde{Z}_k$ has a small number of columns, e.g., when sequential invariance is used.

The key challenge, however, is to construct $\tilde{Z}_k$, considering that we have kept k_Ω° implicit. What comes to our rescue is the technique of direct transformation of the finite approximation of reproducing elements, as discussed in Remark 1 and exemplified in (8) and (11). Recall that B_k is the set of Fourier samples (channels) ω used to construct the finite feature mappings ξ_ω for the *patch-level* kernel mapping φ_k; see (5), (8), and (11). For each ω, denote as $\chi_\omega(\tilde{x}_{k-1})$ the channel of $\tilde{y}_k$ corresponding to ω, which assembles this channel at all patches, rendering its value at $u \in \Omega$ as $\chi_\omega(\tilde{x}_{k-1})(u) := \xi_\omega(P_k \tilde{x}_{k-1}(u))$. So the FA $\tilde{y}_k$ simply collects all channels by $\tilde{y}_k = \chi_{B_k}(\tilde{x}_{k-1}) := \{\chi_\omega(\tilde{x}_{k-1}) : \omega \in B_k\} \in L^2(\Omega, \mathbb{R}^{|B_k|})$, paving the way for deriving the FA of invariances.

For example, when the invariance models the gradient at $\tilde{x}'_{k-1}$ in the direction of v, the corresponding FA $\tilde{z}^k$ is simply $\{\nabla \chi_\omega(\tilde{x}_{k-1}) \cdot v : \omega \in B_k\} \in L^2(\Omega, \mathbb{R}^{|B_k|})$. For rotation, v represents the change of $\tilde{x}'_{k-1}$ when it is rotated by one degree at layer $k-1$. Other invariances can be derived similarly.

5 Stability and Connection with CNN

An important property enjoyed by CKN is its stability under diffeomorphisms of the layer-wise kernel representation [18, 29]. It is natural to ask whether such stability is retained when kernel warping is incorporated to introduce data-dependent transformation invariance. It turns out that the answer is affirmative under a new notion of spatial smoothness for the warping operators.

Consider a C^1-diffeomorphism $\tau : \Omega \to \Omega$, and define the operator L_τ as $L_\tau x(u) = x(u - \tau(u))$. A representation Ψ is called stable under τ if $\sup_{x \neq 0} \|\Psi(L_\tau x) - \Psi(x)\| / \|x\| \leq C_1 \|\nabla\tau\|_\infty + C_2 \|\tau\|_\infty$ for some constants C_1 and C_2 [18]. Here $\nabla\tau$ is the Jacobian, $\|\nabla\tau\|_\infty = \sup_{u\in\Omega} \|\nabla\tau(u)\|$ (operator norm), and $\|\tau\|_\infty := \sup_{u\in\Omega} \|\tau(u)\|$ (Euclidean norm). Our first lemma breaks down the overall deformation of n layers into individual layers, and it extends Proposition 4 of [29] by adopting W_k.

Lemma 1. *Let $x_n = \Psi_n(x) := A_n W_n M_n P_n A_{n-1} W_{n-1} M_{n-1} P_{n-1} \dots A_1 W_1 M_1 P_1 A_0 x$, where W_n is identity and A_0 is arbitrarily close to identity (as in [29]). Denote as $[A, B] := AB - BA$ the commutator of linear operators A and B. Then for any $x \in L^2(\Omega, \mathcal{H}_0)$ and $x \neq 0$,*

$$\|x\|^{-1}\|\Psi_n(L_\tau x) - \Psi_n(x)\| \leq \|[A_n, L_\tau]\| + \|L_\tau A_n - A_n\| + \sum_{k=1}^{n} (\|[P_k A_{k-1}, L_\tau]\| + \|[W_{k-1}, L_\tau]\|) .$$

The proof is given in Appendix **??**. Compared with Proposition 4 of [29] where no warping is applied, we now have a new term $\|[W_k, L_\tau]\|$ to bound, while the first three terms on the right-hand side have already been bounded by [29]. For general W_k, it is not necessarily small even when τ and $\Delta\tau$ are small, so we next introduce a new notion of smoothness in order to retain stability.

Spatial smoothness of warping. Since W_k is a linear mapping, we can think of it as a kernel for the operator such that $(W_k y)(v) = \int_\Omega W_k(v, u) y(u) \, \mathrm{d}u$ for all $u, v \in \Omega$. Then our new spatial smoothness requirement for W_k is that $W_k(v, u)$ is block-wise L_w-Lipschitz continuous. That is, for all $v, v', u, u' \in \Omega$:

$$\|W_k(v, u') - W_k(v, u)\| \leq L_w \|u' - u\| ,$$
$$\|W_k(v, u) - W_k(v', u)\| \leq L_w \|v' - v\| ,$$

where the norms are appropriately chosen in respective spaces. Essentially this means nearby locations have similar weights of contribution to nearby pixels in the warped image. As $W = (I + ZZ^\top)^{-1/2}$ by (4), it is not hard to see that such smoothness holds for directional gradient and local average invariances, provided that their defining images are spatially Lipschitz continuous.

Our key novelty lies in the following bound on $\|[W_k, L_\tau]\|$, and its proof is available in Appendix **??**.

Theorem 4. *Suppose the warping operator W is spatially Lipschitz continuous with constant L_w, and the domain Ω is bounded. Then there exist constants C_1 and C_2 depending only on Ω, such that*

$$\|[W, L_\tau]\| \leq C_1 \|\nabla\tau\|_\infty + C_2 L_w \|\tau\|_\infty . \tag{18}$$

Combining Lemma 1, Theorem 4, and Lemma 5 and 6 of [29], we immediately get the final stability:

Theorem 5. *Suppose $\|\nabla\tau\| \leq 1/2$, Ω is bounded, and the warping operators W_k are all spatially Lipschitz continuous with constant L_w. Then there exist constants C_1, C_2, and C_3 depending only on Ω, and $C_4 = 2^d \|\nabla h\|_1$, such that*

$$\|\Psi_n(L_\tau x) - \Psi_n(x)\| \leq ((nC_1 + (n+1)C_3) \|\nabla\tau\|_\infty + (nC_2 L_w + C_4/\sigma_n) \|\tau\|_\infty) \|x\| .$$

This stability bound grows in both the depth n and the spatial steepness of the warping operator, which is consistent with our intuition. Compared with [29], we here additionally make a mild requirement that Ω is bounded. We also remark that some invariances might make $\|[W, L_\tau]\| = 0$, e.g., when the representers undergo the same transformation as the input. The special cases are left for future work.

5.1 Connection with CNNs

As demonstrated by [29], CKNs contain a set of convolutional neural networks (CNNs) with smooth and homogeneous activations. We can show that such a relationship is retained when kernel warping is introduced, and the new RKHS norm of the overall function allows CNNs to favor invariance-respecting configurations. We defer the details to Appendix **??** for space.

6 Experimental Results

We now present experiments to confirm that the warping operator W_Z enables one-hidden-layer networks and CKNs to learn invariant features.

One-hidden-layer network. Here we test our method on one-hidden-layer networks as described in §3, using both Fourier FA and projection FA with Gaussian kernel. In Fourier FA, we optimized $\{\omega_i\}_{i=1}^p$ in an unsupervised fashion. Given randomly sampled pairs $\{x_i, y_i\}_{i=1}^n$ and the features defined in (5), we optimized the Fourier bases ω_i and the weights η_l by minimizing $\sum_{i=1}^n R_i^2$ where $R_i = e^{-\frac{1}{\sigma^2}\|x_i - y_i\|^2} - \sum_{l=1}^p \eta_l(\cos(\omega_l^\top x_i)\cos(\omega_l^\top y_i)) + \sin(\omega_l^\top x_i)\sin(\omega_l^\top x_i))$.

In projection FA, the prototypes were learned by K-means as in [28]. To construct the warping operator, we followed [15] and used four directional derivatives as invariances that correspond to rotation, scaling, and translation on x or y. FourierW and Fourier stand for Fourier FA with and without warping, respectively. Same for Projection and ProjectionW. 3000 feature

Table 1: Test error for one-hidden-layer network and baselines.

Dataset	1-NN	SVM_{rbf}	VirSVM	Fourier	FourierW	Projection	ProjectionW
USPS	6.24±1.40	4.92±1.13	**4.48±1.26**	5.18±1.38	**4.52±1.52**	5.12±0.98	**4.56±1.22**
MNIST	5.02±1.21	3.26±0.06	2.78±0.07	3.22±0.06	2.76±0.18	3.16±0.09	**2.45±0.10**
EmnisLetter	22.02±5.05	14.76±1.33	13.88±1.04	15.00±2.16	14.02±1.98	14.56±2.52	**11.88±2.10**
Handwritten	9.82±1.06	6.19±0.74	**5.46±0.76**	6.30±0.69	5.82±0.66	6.13±0.82	**5.38±0.79**
FashionMnist	17.56±1.65	12.33±0.42	**12.06±0.46**	13.42±0.57	**12.55±0.32**	13.02±0.59	**12.23±0.48**
SVHN	37.18±3.73	32.41±1.42	**29.54±1.78**	31.58±1.06	30.96±1.21	31.03±2.00	**29.69±1.33**

Table 2: Test error rate for CKN and CNN

Method	USPS			MNIST			EmnistLetter		
	100	500	1k	1k	5k	10k	1k	5k	10k
CNN	18.97±1.90	6.38±0.48	4.64±0.32	4.65±0.43	2.31±0.21	1.72±0.15	24.03±0.97	13.01±0.43	10.98±0.36
CKN	13.88±1.55	3.37±0.17	2.32±0.13	3.68±0.27	1.96±0.10	1.42±0.09	20.08±0.68	11.98±0.31	9.62±0.17
CKNW	**13.04±1.47**	**3.05±0.17**	**1.90±0.14**	**3.09±0.19**	**1.67±0.08**	**1.34±0.05**	**17.95±0.61**	**11.01±0.36**	**9.21±0.13**
	Handwritten			FashionMnist			SVHN		
	1k	5k	10k	1k	5k	10k	1k	5k	10k
CNN	11.23±0.52	4.55±0.14	3.34±0.23	19.34±0.67	14.13±0.36	12.24±0.45	37.18±1.56	23.66±0.48	19.02±0.55
CKN	8.87±0.54	3.76±0.13	2.96±0.20	15.72±0.43	11.42±0.19	10.16±0.23	36.98±0.99	23.04±0.64	19.27±0.45
CKNW	**7.55±0.51**	**3.31±0.17**	**2.32±0.21**	**14.63±0.41**	**11.15±0.17**	**10.01±0.15**	**35.26±0.69**	**22.43±0.49**	**18.84±0.43**

bases were used in Fourier FA and 5000 prototypes were used in projection FA.

For baseline methods, 1-NN is a one-hidden-layer neural network with 500 hidden neurons. SVM_{rbf} is a Gaussian kernel SVM without any finite approximation. Virtual sample SVM (VirSVM) was trained with additional instances generated by applying four different transformations to all training examples: 2-pixel shifts to left and up, rotation by 10 degree, and scaling by 0.1 unit. All settings were evaluated 10 times for randomly sampled training data, and the final output classifier was trained using LIBSVM [42].

We experimented on six datasets: USPS [43], MNIST [44], Handwritten, EmnistLetter [45], FashionMnist [46] and SVHN (greyscaled) [47], which contain 10, 10, 26, 26, 10, and 10 classes respectively. Since USPS is a smaller dataset, we used 1k examples for training and 10k for testing. For all the other datasets, we randomly drew 10k samples for training, and 5k additional *unlabeled* samples for constructing the invariance matrix $\tilde{Z}$ in FourierW and ProjectionW. Finally, the test error was evaluated on the remaining images in each dataset.

As shown in Table 1, applying the warping operator generally leads to lower error than without warping. ProjectionW and VirSVM outperform other methods on most datasets. However, by involving additional instances, VirSVM requires larger storage and longer training time.

Convolution Kernel Network. We next present experiments that illustrate the impact of warping in CKN (§4). Here the kernel mapping operator M_k maps all patches centered at $u \in \Omega$ to a feature in the corresponding RKHS $\mathcal{H}_k$, and these features were approximated by projection FA.

In Table 2, The baseline CNN model has 2 convolutional layers, each one is constructed on 5×5 patches, followed by a 2×2 average pooling. The first and second convolutional layers contains 16 and 32 filters, respectively. CKN stands for a one-layer CKN with patch size 5×5 and 300 filters, followed by a pooling layer with stride 2. CKNW has the same setting as CKN, but with warping applied. The warping operator here used the same four transformations as in the above one-hidden-layer network. The SVD in (12) was approximated with the top 100 singular vectors. We compared CKNW with CKN on the same six datasets as in Table 1. To show the effect of applying warping on different sizes of training data, we randomly selected 100, 500, 1k training examples from the USPS dataset, and 1k, 5k, 10k training examples for other datasets. The performance was evaluated on all the rest images in each dataset, and all settings were evaluated 10 times. It is clear that CKN with warping always delivers lower error than vanilla CKN, and the improvement is more significant when the training set is smaller, i.e., when invariance is more informative.

Conclusion. In this paper we proposed kernel warping to induce feature representations that respect invariances that reach beyond transformation. The framework is efficient and flexible when applied to CKN. For future work, we will further study the end-to-end supervised training for the latter setting, and use sparse greedy approximation to extend to global invariance.

References

[1] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. *In KDD*. 2004.

[2] M. Ferraro and T. M. Caelli. Lie transformation groups, integral transforms, and invariant pattern recognition. *Spatial Vision*, 8:33–44, 1994.

[3] P. Simard, Y. LeCun, J. S. Denker, and B. Victorri. Transformation invariance in pattern recognition-tangent distance and tangent propagation. *In Neural Networks: Tricks of the Trade*. 1996.

[4] O. Bousquet, O. Chapelle, and M. Hein. Measure based regularization. *In NIPS*. 2003.

[5] T. Joachims. Transductive inference for text classification using support vector machines. *In ICML*. 1999.

[6] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. *In ICML*. 2001.

[7] M. Belkin, P. Niyogi, and V. Sindhwani. On manifold regularization. *In AISTATS*. 2005.

[8] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. *In ICML*. 2003.

[9] D. Zhou and B. Schölkopf. Discrete regularization. *In Semi-Supervised Learning*, pp. 221–232. MIT Press, 2006.

[10] D. DeCoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 46:161–190, 2002.

[11] A. Ben-Tal, L. E. Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton University Press, 2008.

[12] C. Bhattacharyya, K. S. Pannagadatta, and A. J. Smola. A second order cone programming formulation for classifying missing data. *In NIPS*. 2005.

[13] T. Graepel and R. Herbrich. Invariant pattern recognition by semidefinite programming machines. *In NIPS*. 2004.

[14] C. H. Teo, A. Globerson, S. Roweis, and A. Smola. Convex learning with invariances. *In NIPS*. 2007.

[15] X. Zhang, W. S. Lee, and Y. W. Teh. Learning with Invariances via Linear Functionals on Reproducing Kernel Hilbert Space. *In NIPS*. 2013.

[16] A. J. Smola and B. Schölkopf. On a kernel-based method for pattern recognition, regression, approximation and operator inversion. *Algorithmica*, 22:211–231, 1998.

[17] J. Bruna and S. Mallat. Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1872–1886, Aug 2013. ISSN 0162-8828. doi:10.1109/TPAMI.2012.230.

[18] S. Mallat. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398, 2012.

[19] C. Walder, O. Chapelle, and P. M. Learning with Transformation Invariant Kernels. *In NIPS*. 2008.

[20] C. J. C. Burges. Geometry and invariance in kernel based methods. *In* B. Schölkopf, C. J. C. Burges, and A. J. Smola, eds., *Advances in Kernel Methods — Support Vector Learning*, pp. 89–116. MIT Press, Cambridge, MA, 1999.

[21] A. Raj, A. Kumar, Y. Mroueh, P. Thomas Fletcher, and B. Schoelkopf. Local group invariant representations via orbit embeddings. *In AISTATS*. 2017.

[22] Y. Mroueh, S. Voinea, and T. Poggio. Learning with group invariant features: A kernel perspective. *In NIPS*. 2015.

[23] B. Haasdonk and H. Burkhardt. Invariant kernel functions for pattern analysis and machine learning. *Machine Learning*, 68(1):35–61, 2007.

[24] T. A. Poggio and F. Anselmi. *Visual Cortex and Deep Networks: Learning Invariant Representations*. The MIT Press, Cambridge, MA, USA, 2016.

[25] C. K. I. Williams and M. Seeger. Using the Nystrom method to speed up kernel machines. *In NIPS*. 2000.

[26] A. Rahimi and B. Recht. Random features for large-scale kernel machines. *In NIPS*. 2008.

[27] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional kernel networks. *In NIPS*. 2014.

[28] J. Mairal. End-to-end kernel learning with supervised convolutional kernel networks. *In NIPS*. 2016.

[29] A. Bietti and J. Mairal. Group Invariance, Stability to Deformations, and Complexity of Deep Convolutional Representations. *In NIPS*. 2017.

[30] S. Ravanbakhsh, J. Schneider, and B. Poczos. Equivariance Through Parameter-Sharing. *In ICML*. 2017.

[31] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[32] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other*

Kernel-based Learning Methods. Cambridge University Press, Cambridge, UK, 2000.

[33] I. Steinwart and A. Christmann. *Support Vector Machines*. Information Science and Statistics. 2008.

[34] E. Kreyszig. *Introductory Functional Analysis with Applications*. Wiley, 1989.

[35] A. J. Smola and B. Schölkopf. On a kernel-based method for pattern recognition, regression, approximation and operator inversion. *Tech. Rep. 1064*, GMD, 1997.

[36] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[37] V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: from transductive to semi-supervised learning. *In ICML*. 2005.

[38] M. A. Woodbury. Inverting modified matrices. *Tech. Rep. Memorandum Rept. 42*, Statistical Research Group, Princeton University, Princeton, NJ, 1950.

[39] D. S. Rosenberg and P. L. Bartlett. The rademacher complexity of co-regularized kernel classes. *In AISTATS*. 2007.

[40] W. Rudin. *Fourier analysis on groups*. Interscience Publishers, New York, 1962.

[41] K. Zhang, I. W. Tsang, and J. T. Kwok. Improved nystr om low-rank approximation and error analysis. *In ICML*. 2008.

[42] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[43] USPS. USPS dataset. `https://cs.nyu.edu/~roweis/data.html`.

[44] Y. LeCun. MNIST handwritten digit database, 1998. URL `http://yann.lecun.com/exdb/mnist/`.

[45] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik. Emnist: an extension of mnist to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017.

[46] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[47] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. *In NIPS workshop on deep learning and unsupervised feature learning*, vol. 2011, p. 5. 2011.