PROXNET: END-TO-END LEARNING OF STRUCTURED REPRESENTATION BY PROXIMAL MAPPING

Anonymous authors
Paper under double-blind review

ABSTRACT

Underpinning the success of deep learning is the effective regularization that allows a broad range of structures in data to be compactly modeled in a deep architecture. Examples include transformation invariances, robustness to adversarial/random perturbations, and correlations between multiple modalities. However, most existing methods incorporate such priors either by auto-encoders, whose result is used to initialize supervised learning, or by augmenting the data with exemplifications of the transformations which, despite the improved performance of supervised learning, leaves it unclear whether the learned latent representation does encode the desired regularities. To address these issues, this work proposes an *end-to-end* representation learning framework that allows prior structures to be encoded *explicitly* in the hidden layers, and to be trained efficiently in conjunction with the supervised target. Our approach is based on proximal mapping in a reproducing kernel Hilbert space, and leverages differentiable optimization. The resulting technique is applied to generalize dropout and invariant kernel warping, and to develop novel algorithms for multiview modeling and robust temporal learning.

1 INTRODUCTION

The success of deep learning relies on massive neural networks that often considerably out-scale the training dataset, defying the conventional learning theory (Zhang et al., 2017; Arpitz et al., 2017). Regularization has been shown essential and a variety of forms are available for it. Standard invariances to transformations such as rotation and translation (Simard et al., 2012) have been extended beyond group-based diffeomorphisms to indecipherable transformations that are only exemplified by pairs of views (Pal et al., 2017), e.g., sentences uttered by the same person. Structural regularities are also commonly present **a)** *within* layers of neural networks, such as sparsity (Makhzani & Frey, 2014), spatial invariance in convolutional nets, structured gradient that accounts for data covariance (Roth et al., 2018), and manifold smoothness characterized by a graph (Kipf & Welling, 2017); and **b)** *between* layers of representation, such as stability under dropout and adversarial perturbations of preceding layers (Srivastava et al., 2014), contractivity between layers (Rifai et al., 2011), and correlations in hidden layers among multiple views (Wang et al., 2015; Andrew et al., 2013).

Structural regularities can be captured in the context of both supervised and unsupervised learning. Dataset augmentation, probably the most prevalent approach, generates new data to better exemplify the classes in a supervised task (Krizhevsky et al., 2012; Poggio & Vetter, 1992; Niyogi et al., 1998). Adversarial learning finds the attack that maximally impairs classification, and virtual adversarial learning, despite the capability of using unlabeled data, relies on the predicted class distribution to select adversarial examples (Miyato et al., 2016; 2017). Although these methods boost prediction performance, it is unclear whether this is due to the learned representation or due to the improved classifier. For example, Figure 1a shows the two-moon dataset with only two labeled examples and many unlabeled ones. Although graph-based semi-supervised learning can easily find the separating boundary (Zhu et al., 2003), it does not uncover any manifold-aware representation of the data.

This limitation has been tackled by auto-encoders that extract salient features to reconstruct the input, while also encoding a variety of latent structures such as layer-wise contractivity (Rifai et al., 2011), noise resilience (Jaitly & Hinton, 2011; Vincent et al., 2008), correlations between views (Wang et al., 2015), sparsity in latent code (Makhzani & Frey, 2014), on-manifold robustness (Stutz et al., 2019), and graphical interconnections in node embeddings (Kipf & Welling, 2016). However most of these methods are not amenable to end-to-end learning, and are used as an initialization for the subsequent

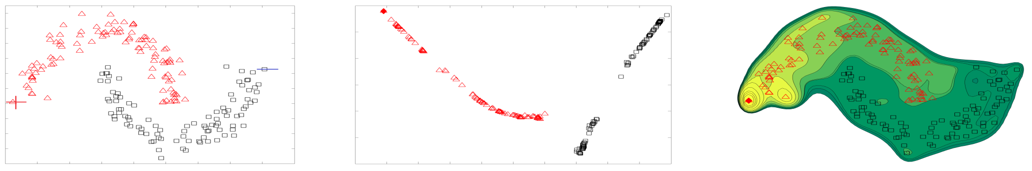

(a) Original two-moon dataset (b) Representation after prox-map (c) Contour to ♦ after prox-map

Figure 1: (a) The two-moon dataset with only two labeled examples '+' and '−' (left and right), but abundant unlabeled examples that reveal the inherent structure; (b) Representation inferred from top-2 kernel PCA based on the proximal mapping with gradient flatness and Gaussian kernel (see §3); (c) contour of distance to the leftmost point ♦, based on the representation from proximal mapping.

supervised task. Impeding the synergy of these two objectives is the *contention* between accurate prediction and faithful input reconstruction, compelling model weights to make compromises.

The goal of this paper, therefore, is to learn representations that *explicitly* encode structural priors in an *end-to-end* fashion. Our tool is proximal mapping, which has been extensively used in optimization to enforce structured solutions (e.g., sparsity) at each iteration (Parikh & Boyd, 2014), but surprisingly underutilized in modeling deep neural networks. Our inspiration stems from the deeply supervised network (Lee et al., 2015), but instead of promoting the *discriminative* power of hidden features at each layer for the final supervised task, we promote their desired regularities in an unsupervised manner by directly projecting them to the subspace of such structured representations, or moving towards it. Using the prior that gradients are flat at all examples, Figures 1b and 1c show the resulting representation and distance metric of the two-moon data that accounts for the underlying manifold, making the classification problem trivial.

Why does it help to insert proximal mappings as layers into a deep network (hence the name ProxNet)? First of all, it provides the modularity by decoupling regularization from supervised learning — the structural regularization is encapsulated *within* the proximal layer and the supervised learning signal originating from downstream layers can be backpropagated *through* it in an end-to-end manner. This frees weight optimization from simultaneously catering for unsupervised structures and supervised performance metrics, as in the conventional regularized risk minimization. We will take a further step of kernelization by treating the hidden layer outputs as functions in a reproducing kernel Hilbert space, enriching the flexibility of regularization that is not attainable by vectors.

Although the *general* framework of embedding an optimization layer in a deep architecture has been established in OptNet (Amos & Kolter, 2017) and (Domke, 2010; 2012; Belanger et al., 2017; Baydin et al., 2017; Metz et al., 2017; Brakel et al., 2013; Bertinetto et al., 2019; Lee et al., 2019; Amos et al., 2018; Stoyanov et al., 2011; Goodfellow et al., 2013; Gould et al., 2016), we instead focus on *concrete* designs to effectively and efficiently model the prior regularities in data. So in fact more closely related to our work is task-driven sparse coding (Mairal et al., 2012; Bagnell & Bradley, 2008), where a dictionary is learned by backpropagating the supervised error incurred by the code. However, besides being shallow, these models require overhauling existing primitives in deep learning, e.g. LSTM blocks, while significant composability and extensibility will be made available if they can be reused by simply transforming their output through a new layer, or extending their functionality by using them as a black box. This is exactly accomplished by ProxNet.

To warm up, we will first review the existing building blocks of deep networks through the lens of proximal mapping (§2), and then we will illustrate how ProxNet is non-trivially connected with kernel warping (robustness to noise in the input, §3) and dropout (noise in the preceding layer, §4). Afterwards, two novel ProxNets will be introduced that achieve end-to-end multiview learning (§5) and invariant recurrent modeling (§6). Extensive experiments (§7) show that ProxNet not only achieves state-of-the-art prediction performance, but also learns superior data representations.

2 Proximal Mapping as a Primitive Construct in Deep Networks

Proximal mapping has been used extensively in nonsmooth and constrained optimization; see a comprehensive review in Parikh & Boyd (2014). Given a closed convex set C in $\mathbb{R}^n$ and a convex function $f : \mathbb{R}^n \to \mathbb{R}$, the proximal mapping $\mathsf{P}_f : \mathbb{R}^n \to \mathbb{R}^n$ is defined as

$$\mathbb{R}^n \ni z \mapsto \mathsf{P}_f(z) := \arg\min_{x \in C} f(x) + \tfrac{\lambda}{2} \|x - z\|^2, \tag{1}$$

where the norm is L_2. To lighten the notation, we will set $\lambda = 1$. In general, C and f can be nonconvex, making $\mathsf{P}_f(z)$ set valued (Hare & Sagastizábal, 2009; Bernard & Thibault, 2004; Poliquin & Rockafellar, 1996); we will only need differentiation at one element.

Proximal map is highly general, encompassing most primitive operations in deep learning (Combettes & Pesquet, 2018). For example, any activation function σ with $\sigma'(x) \in (0, 1]$ (e.g., sigmoid), $x \mapsto \sigma(x)$ is indeed a proximal map with $C = \mathbb{R}^n$ and $f(x) = \int \sigma^{-1}(x)\,\mathrm{d}x - \frac{1}{2}x^2$, which is convex. The ReLu and hard tanh activations can be recovered by $f = 0$, with $C = [0, \infty)$ and $C = [-1, 1]$, respectively. Soft-max transfer $\mathbb{R}^n \ni x \mapsto (e^{x_1}, \dots, e^{x_n})^\top / \sum_i e^{x_i}$ corresponds to $C = \{x \in \mathbb{R}^n_+ : \mathbf{1}^\top x = 1\}$ and $f(x) = \sum_i x_i \log x_i - \frac{1}{2}x_i^2$, which are convex. Batch normalization maps $x \in \mathbb{R}^n$ to $(x - \mu\mathbf{1})/\sigma$, where $\mathbf{1}$ is a vector of all ones, and μ and σ are the mean and standard deviation of the elements in x, respectively. This mapping can be recovered by $f = 0$ and $C = \{x : \|x\| = \sqrt{n}, \mathbf{1}^\top x = 0\}$. Although C is not convex, this $\mathsf{P}_f(x)$ must be a singleton for $x \neq 0$.

Backpropagation (BP) through proximal mapping. In essence, ProxNet falls in the framework of differentiable optimization network (OptNet) laid by Amos & Kolter (2017), which provides recipes for differentiating through an optimization layer. However, different from OptNet, our focus is not on optimization, but on using ProxNet to model the prior structures in the data. Detailed discussions on the relationship between ProxNet and OptNet or related works are available in Appendix A.

Kernelized deep neural networks. Analogous to the multi-layer transformation of *vectors* in deep nets, it has been proposed that the latent representations can be extended to functions in a reproducing kernel Hilbert space (RKHS). This allows non-vectorial data to be encoded (Laforgue et al., 2019), and some *prior* invariances to be hard wired (Mairal et al., 2014; Mairal, 2016; Bietti & Mairal, 2017). It is different from the conventional motivation of using kernels for enriching the feature space, which is already facilitated by deep neural networks. Ma et al. (2019) further developed kernel warping to incorporate *data-dependent* invariances at each layer, leveraging the feasibility of differentiation or integration of function representations that is not available to simple vectors.

In this paper, we assume that the input space $\mathcal{X}$ is a compact metric space, and a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a continuous and positive definite function (Kreyszig, 1989; Steinwart & Christmann, 2008). The RKHS induced by k is denoted as $\mathcal{H}$. Recall that a linear functional L from a normed space $\mathcal{V}$ to $\mathbb{R}$ is bounded, if $\|L\| := \sup_{f \in \mathcal{V} : \|f\|_{\mathcal{V}} = 1} |L(f)|$ is finite. Clearly proximal maps can be extended to RKHS for any convex functional $L : \mathcal{H} \to \mathbb{R}$ (Bauschke & Combettes, 2011):

$$\mathcal{H} \ni h \mapsto \mathsf{P}_L(h) := \arg\min\nolimits_{f \in C} L(f) + \tfrac{\lambda}{2} \|f - h\|^2_{\mathcal{H}}, \quad \text{where} \quad C \subseteq \mathcal{H} \text{ is convex.} \tag{2}$$

Kernel linearization. Although this optimization can often be solved when favorable structures are available (e.g., representer theorem), BP through it can still be hard. So we resort to Nyström approximations of functions in $\mathcal{H}$ (Williams & Seeger, 2000). Using p samples $W := \{\omega_i\}_{i=1}^p$ drawn i.i.d. from $\mathcal{X}$, we derive an FA of f as follows, ensuring that $\langle \tilde{f}, \tilde{h} \rangle \approx \langle f, h \rangle_{\mathcal{H}}$ for all $f, h \in \mathcal{H}$:

$$\tilde{f} := K_W^{-1/2} f_W, \quad \text{where} \quad K_W := (k(\omega_i, \omega_j))_{ij} \in \mathbb{R}^{p \times p}, \quad f_W := (f(\omega_1), \dots, f(\omega_p))^\top \in \mathbb{R}^p.$$

Thus, if $L(f)$ can be represented as a $g(\{\langle z_i, f \rangle_{\mathcal{H}}\}_{i=1}^m)$ where $m \leq \infty$ and $z_i \in \mathcal{H}$, then $\mathsf{P}_L(f)$ can be approximated by minimizing $g(\{\tilde{z}_i^\top \tilde{f}\}_{i=1}^m) + \frac{\lambda}{2}||\tilde{f} - \tilde{h}||^2$ over a linearization of C, e.g., $||\tilde{f}||_2 \leq 1$.

3 Generalizing Multi-layer Invariant Kernel Warping by ProxNet

We will review in the next two sections two existing techniques in deep learning, shedding new insights from the perspective of proximal mapping. Mairal et al. (2014) and Mairal (2016) introduced multi-layer transformations of convolutional kernel descriptors, and Ma et al. (2019) proposed warping kernels to incorporate a variety of data-dependent invariances beyond transformation, if the invariance can be represented by a bounded linear functional (BLF). Our first new insight shows that kernel warping is a bona fide proximal mapping, freeing it from being restricted to linear invariances.

The graph Laplacian on a function f is $\sum_{ij} w_{ij}(f(x_i) - f(x_j))^2$, where $f(x_i) - f(x_j)$ is a BLF of f. Parameterizing an image as $I(\alpha)$ where α is the degree of rotation/translation/etc, transformation invariance favors a small magnitude of $\frac{\partial}{\partial \alpha}|_{\alpha=0} f(I(\alpha))$, again a BLF. Local averaging compares $f(x_i)$ with its neighborhood $\int f(\tau)\rho(x_i - \tau)\,\mathrm{d}\tau$, where ρ is a zero-centered distribution. The difference is also a BLF. By Riesz representation theorem, a BLF can be written as $\langle z_i, f \rangle_{\mathcal{H}}$ for some $z_i \in \mathcal{H}$.

In order to respect the desired invariances, Ma et al. (2019) proposed a warped RKHS $\mathcal{H}^\circ$ consisting of the same functions in the original $\mathcal{H}$, but redefining the norm and the corresponding kernel by

$$\|f\|^2_{\mathcal{H}^\circ} := \|f\|^2_{\mathcal{H}} + \sum\nolimits_{i=1}^m \langle z_i, f\rangle^2_{\mathcal{H}} \quad \Leftrightarrow \quad k^\circ(x_1, x_2) = k(x_1, x_2) - z(x_1)^\top K_Z z(x_2), \tag{3}$$

where $z(x) = (z_1(x), \ldots, z_m(x))^\top$. Then replacing $k(x, \cdot)$ by $k^\circ(x, \cdot)$ results in a new invariant representation. Such a warping can be applied to all layers in, e.g., deep convolutional kernel networks (CKNs, Bietti & Mairal, 2019), instilling invariance with respect to the preceding layer's output.

The major limitation of this method, however, is that the invariances have to be modeled by $\langle z_i, f\rangle^2_{\mathcal{H}}$ in order to make $\|f\|^2_{\mathcal{H}} + \sum_{i=1}^m \langle z_i, f\rangle^2_{\mathcal{H}}$ a norm square, precluding many interesting invariances such as total variation $f \mapsto \int |f'(x)| \, \mathrm{d}x$. In addition, the change of RKHS creates conceptional and practical complications, and it will be much more convenient if we retain $\mathcal{H}$.

Interestingly, both of the desirabilities can be achieved by simply reformulating kernel warping into a proximal mapping. Ma et al. (2019) showed that the linearization of $k^\circ(x, \cdot)$ in $\mathcal{H}^\circ$ can be written as

$$(I + \tilde{Z}\tilde{Z}^\top)^{-1/2}\tilde{\varphi}(x), \quad \text{where } \tilde{Z} = (\tilde{z}_1, \ldots, \tilde{z}_m), \text{and } \tilde{\varphi}(x) \text{ is the linearization of } k(x, \cdot) \text{ in } \mathcal{H}. \tag{4}$$

Simply setting $L(f) := \frac{1}{2}\sum_{i=1}^m \langle z_i, f\rangle^2_{\mathcal{H}}$ in the proximal map (2) to measure invariance, we derive that the linearization of $\mathsf{P}_L(k(x, \cdot))$ is $(I + \tilde{Z}\tilde{Z}^\top)^{-1}\tilde{\varphi}(x)$, which is almost the same as that from kernel warping in (4), except for the exponent on $I + \tilde{Z}\tilde{Z}^\top$. In practice, we observed that it led to little difference, and the result of proximal mapping using Gaussian kernel and flat-gradient invariance is shown in Figure 1. Trivially, CKNs can now leverage nonlinear invariances such as total variation by using a nonlinear (convex) regularizer L in (2). In hindsight, $\mathsf{P}_L(k(x, \cdot))$ is very intuitive: shift $k(x, \cdot)$ to a new representation in $\mathcal{H}$ with small displacement while improving the invariances.

4 Dropout Training as a Proximal Mapping

Another interesting but unheeded instance of proximal mapping is the well-known dropout. Although connection with adaptive regularization has been unraveled by Wager et al. (2013) and Wang & Manning (2013) for a single layer, extensions to multiple layers remain unclear.

We will achieve it by ProxNet. The underlying rationale of dropout is to introduce a blackout noise ϵ to a hidden layer $x \in \mathbb{R}^h$ (Hinton et al., 2012), so that the subsequent layer's output f is robust to the noise. For better generality, we let f lie in an RKHS. Denote the perturbed x as x_ϵ, which can be i) $x + \epsilon$ for additive Gaussian noise with $\epsilon_i \sim \mathcal{N}(\mu, \sigma^2)$, or ii) $x \odot \epsilon$ for dropout noise, where $\odot$ stands for elementwise multiplication, and $\epsilon_i = 0$ with probability δ, and $\epsilon_i = (1 - \delta)^{-1}$ otherwise.

In Wager et al. (2013), the target variable y is concerned about a conditional distribution $p(y|x) = \exp(yf(x) - A(f(x)))$ for $y = \pm 1$, where A is the log-partition function which equals $\log(e^z + e^{-z})$ for logistic regression. Then the sensitivity of $p(y|x)$ with respect to x can be naturally measured by

$$D(f, x) := \mathbb{E}_\epsilon[\log p(y|x) - \log p(y|x_\epsilon)] = \mathbb{E}_\epsilon[A(f(x_\epsilon))] - A(f(x)). \tag{5}$$

Taking expectation over the empirical distribution $\tilde{p}$ yields $\tilde{D}(f) := \mathbb{E}_{x\sim\tilde{p}} D(f, x)$. By specializing f to a linear function $f_\beta : x \mapsto \beta^\top x$ and applying a Taylor expansion on A about $f(x)$, Wager et al. (2013) showed that when $y|x$ is logistic, $\tilde{D}(f_\beta)$ is proportional to the following terms

$$\tilde{D}(f_\beta) \quad \propto \quad \|\beta\|^2 \, \mathbb{E}_{x\sim\tilde{p}}[p_x(1 - p_x)] \text{ (additive noise)}, \quad \text{and} \quad \sum\nolimits_j a_j \beta_j^2 \text{ (dropout noise)}, \tag{6}$$

where $p_x := (1 + \exp(-f(x)))^{-1}$ and $a_j = \mathbb{E}_{x\sim\tilde{p}}[p_x(1 - p_x)x_j^2]$. So the advantage of dropout lies in permitting β_j to take a larger magnitude if a) feature x_j contributes to a confident prediction with small $p_x(1 - p_x)$ whenever it is active; or b) x_j is small in general (hence is a_j), i.e., allowing rare but discriminative features to receive higher weights. These effects are especially useful in text data.

We now show that a similar regularization can be achieved in a nontrivial fashion through proximal mapping. Naturally we instantiate the L in (2) by $\tilde{D}(f)$ to induce invariance to dropout, amounting to

$$\mathsf{P}_L(k(x, \cdot)) = \arg\min\nolimits_{f\in\mathcal{H}} \ \tfrac{1}{2}\tilde{D}(f) + \tfrac{\lambda}{2}\|f - k(x, \cdot)\|^2_{\mathcal{H}}. \tag{7}$$

If we hypothetically suppress the dependency of p_x on β, $\mathsf{P}_L(k(x, \cdot))$ admits a *closed form* under dropout noise and linear kernel (so that $k(x, \cdot) = x$ and $f(x) = f^\top x$): $\mathsf{P}_L(x) = b \odot x$ where $b_j = \lambda(a_j + \lambda)^{-1}$. Letting the output layer be $x \mapsto \alpha^\top x$, the regularized risk for loss ℓ can be written as

$$\min_\alpha \mathbb{E}_{(x,y)\sim\tilde{p}}[\ell(\alpha^\top \mathsf{P}_L(x), y)] + c\,\|\alpha\|^2 \quad \Leftrightarrow \quad \min_\beta \mathbb{E}_{(x,y)\sim\tilde{p}}[\ell(\beta^\top x, y)] + \frac{c}{\lambda^2}\sum\nolimits_j (a_j + \lambda)^2 \beta_j^2.$$

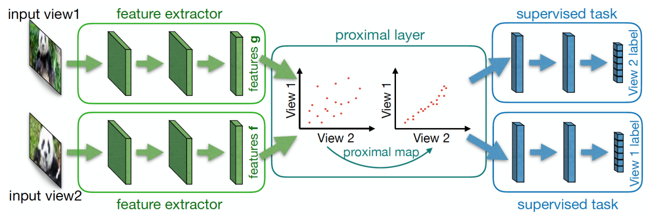


Figure 2: Multiview learning with a proximal CCA layer

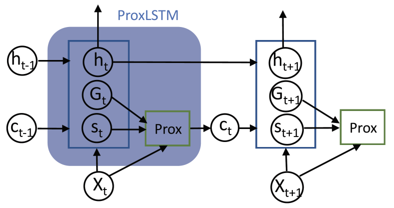


Figure 3: A proximal LSTM layer

Thus, as long as λ is small and c is set proportionally to λ^2, the regularizer on β almost recovers $\sum_j a_j \beta_j^2$ in (6), modulo the squaring of a_j. However, one should be mindful that the above expression of $P_L(x)$ takes p_x in a_j as a constant, while in practice it depends on f in the optimization of (7). Fortunately, our simulations in Appendix B show that this does not result in a significant difference.

The proximal mapping in (7) can be readily extended to nonlinear kernels, calling for Nyström approximation of $\tilde{D}(f)$. The robustness measures $\tilde{D}$ and D may enjoy even more flexibility. The current assignment in (5) essentially uses the curvature of $A \circ f$ to quantify the impact of noise. Other choices that measure the discrepancy between the distribution of $A(f(x_\epsilon))$ and the singleton $A(f(x))$ can also be used, e.g., Wasserstein distance (Kantorovitch, 1958). To summarize Sections 3 and 4, proximal mapping provides a compact framework that allows us to flexibly model the invariance to perturbation in both input and hidden layers. We will next leverage it to develop two novel algorithms.

5 ProxNet for Multiview Learning

In multiview learning, observations are available from a pair of views: $\{(x_i, y_i)\}_{i=1}^n$, and each pair is associated with a label c_i. In the deep canonical correlation analysis model (DCCA, Andrew et al., 2013), the x-view is passed through a multi-layer neural network or kernel machine, leading to a hidden representation $f(x_i)$. Similarly the y-views are transformed into $g(y_i)$. CCA aims to maximize the correlation of these two views after projecting into a common k-dimensional subspace, through $\{u_i\}_{i=1}^k$ and $\{v_i\}_{i=1}^k$ respectively. Denoting $X = (f(x_1), \ldots, f(x_n))H$ and $Y = (g(y_1), \ldots, g(y_n))H$ where $H = I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$ is the centering matrix, CCA finds $U = (u_1, \ldots, u_k)$ and $V = (v_1, \ldots, v_k)$ that maximize the correlation:

$$\min_{U,V} -\operatorname{tr}(U^\top XY^\top V), \quad \text{s.t} \quad U^\top XX^\top U = I, \; V^\top YY^\top V = I, \; u_i^\top XY^\top v_j = 0, \forall i \neq j. \quad (8)$$

Denote the optimal objective value as $L(X, Y)$. DCCA directly optimizes it with respect to the parameters in f and g, while DCCA autoencoder (DCCAE, Wang et al., 2015) further reconstructs the input. They both use the result to initialize a finer tuning of f and g, in conjunction with subsequent layers h for a supervised target c_i. We aim to improve this two-stage process with an end-to-end approach based on proximal mapping, which can be written as $\min_{f,g,h} \sum_i \ell(h(p_i, q_i), c_i)$ with

$$\{(p_i, q_i)\}_{i=1}^n = \mathsf{P}_L(X, Y) := \arg\min_{P,Q} \tfrac{\lambda}{2} \|P - X\|_F^2 + \tfrac{\lambda}{2} \|Q - Y\|_F^2 + L(P, Q), \quad (9)$$

where $\|\cdot\|_F$ stands for the Frobenius norm, $P = (p_1, \ldots, p_n)$, and $Q = (q_1, \ldots, q_n)$. The proximal mapping can also be performed in *mini-batch* at both *training* and *testing*, a nontrivial setting which resembles the "tasks" in meta-learning. The entire framework is illustrated in Figure 2. Kernelization is straightforward using the above techniques, and we leave it for future work.

Optimization and BP. Although efficient closed-form solution is available for the CCA objective in (8), none exists for the proximal mapping in (9). However, it is natural to take advantage of this closed-form solution. In particular, assuming $f(x_i)$ and $g(y_i)$ have the same dimensionality, Andrew et al. (2013) showed that $L(X, Y) = -\sum_{i=1}^k \sigma_i(T)$, where σ_i stands for the i-th largest singular value, and $T(X, Y) = (XX^\top + \epsilon I)^{-1/2}(XY^\top)(YY^\top + \epsilon I)^{-1/2}$. Here $\epsilon > 0$ is a small stabilizing constant. Then (9) can be solved by gradient descent or L-BFGS. The gradient of $\sum_{i=1}^k \sigma_i(T(P, Q))$ is available from Andrew et al. (2013), which relies on SVD. But since the dimension of f and g is low in practice (10 in our experiment and DCCA), the cost of computing T and SVD is low. Details on backpropagation and extension to more than two views are relegated to Appendix C.

6 ProxNet for Adversarial Learning in Recurrent Neural Nets

Our second novel instance of ProxNet tries to invariantize LSTM to perturbations on inputs x_t *at each step*. Adversarial training has been proposed in this context (Miyato et al., 2017), but not for representation learning. The dynamics of hidden states c_t in an LSTM can be compactly represented by $c_t = f(c_{t-1}, h_{t-1}, x_t)$, with outputs h_t updated by $h_t = g(c_{t-1}, h_{t-1}, x_t)$. We aim to encourage that the hidden state c_t stays invariant, when each x_t is perturbed by δ_t whose norm is bounded by δ. To this end, we introduce an intermediate step $s_t = s_t(c_{t-1}, h_{t-1}, x_t)$ that computes the original hidden state, and then apply proximal mapping so that the next state c_t remains close to s_t, while also moving towards the *null space* of the variation of s_t under the perturbations on x_t. Formally,

$$c_t := \arg\min_c \tfrac{\lambda}{2} \|c - s_t\|^2 + \tfrac{1}{2} \max_{\delta_t : \|\delta_t\| \le \delta} \langle c, s_t(c_{t-1}, h_{t-1}, x_t) - s_t(c_{t-1}, h_{t-1}, x_t + \delta_t)\rangle^2$$

$$\approx \arg\min_c \frac{\lambda}{2} \|c - s_t\|^2 + \frac{1}{2} \max_{\delta_t : \|\delta_t\| \le \delta} \left\langle c, \frac{\partial}{\partial x_t} s_t(c_{t-1}, h_{t-1}, x_t)\delta_t \right\rangle^2 \tag{10}$$

$$= \arg\min_c \tfrac{\lambda}{2} \|c - s_t\|^2 + \tfrac{\delta^2}{2} \left\|c^\top G_t\right\|_*^2, \quad \text{where} \quad G_t := \tfrac{\partial}{\partial x_t} s_t(c_{t-1}, h_{t-1}, x_t), \tag{11}$$

and $\|\cdot\|_*$ is the dual norm. The diagram is shown in Figure 3. Using the L_2 norm, we obtain a closed-form solution for c_t: $(I + \lambda^{-1}\delta^2 G_t G_t^\top)^{-1} s_t$, and BP can be reduced to second-order derivatives (see Appendix D). A key advantage of this framework is the generality and ease in inserting proximal layers into the framework, almost oblivious to the underlying building blocks, be it LSTM or GRU. The implementation only needs to directly invoke their second-order derivatives as a black box.

7 Experiments

We evaluated the empirical performance of ProxNet for multiview learning on supervised learning (three tasks) and unsupervised learning (crosslingual word embedding). ProxLSTM was evaluated on sequence classification. Details on data preprocessing, experiment setting, optimization, and additional results are given in Appendix E. Here we highlight the major results and experiment setup.

7.1 Multiview Supervised Learning

Dataset. We used three datasets. **Caltech101-20** consists of 2,386 images of 20 classes. Following Zhao et al. (2017), six sets of handcrafted features (views) were extracted: wavelet moments, Gabor, CENTRIST, HOG, GIST, and LBP features. **Reuters** dataset contains 18,758 documents of 6 classes, each with English, French, German, Spanish, and Italian versions (views). **NUS-WIDE-OBJ** has 30,000 images of 31 classes with five views: color histogram/correlation/moments, edge distribution, and wavelet texture. In view that the datasets are highly imbalanced across classes, we reweighted all examples inverse proportionally to the size of its class. Since the existing packages do not take nonuniform instance weights, we manually resampled the training examples to achieve the same effect. This led to 1198 / 7980 / 1188, 9379 / 15426 / 9379, 14270 / 58590 / 9683 examples for original-training/resampled-training/test in Caltech101-20, Reuters, and NUS-WIDE-OBJ, respectively.

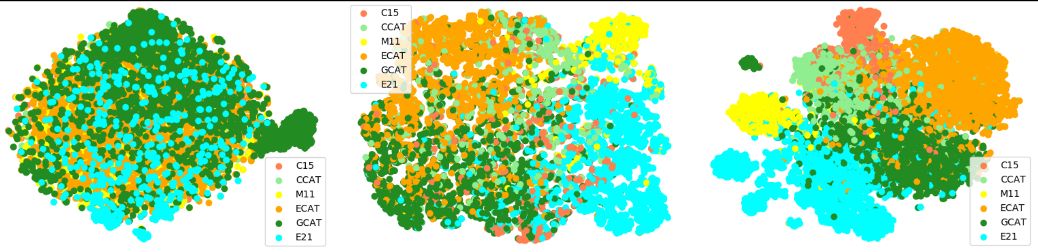
Algorithms. We compared ProxNet with state-of-the-art multiview learning methods. DCCA and DCCAE only take two views, so we partitioned the views in each dataset into two groups, and the concatenation of their corresponding features formed two views. Multiple partitionings were tried, and the best performance was reported. We also tested on DGCCA which extended DCCA to more than two views (Benton et al., 2019).

We trained ProxNet using three fully-connected layers as the feature map, then the proximal layer, followed by three fully-connected layers each having 512 sigmoid units. The final linear output layer has softmax units corresponding to the classes. The proximal layer maps from $\mathbb{R}^{10}$ to $\mathbb{R}^{10}$, and the mini-batch size was set to 200 — a larger value did not improve the performance much. As a result, the proximal mapping was solved very efficiently. Test examples were also grouped into mini-batches of 200, fed into ProxNet for proximal mapping. λ in (9) was chosen from $\{0.1, 1, 10, 100\}$ based on a validation set. All experiments were run five times to produce mean and standard deviation.

Results. As shown in Table 1, ProxNet consistently delivers significantly higher test accuracy and F1-score than all other methods. F1-score is useful because the classes are imbalanced. With no surprise, DGCCA outperforms DCCA and DCCAE because it can finely model more than two views. The comparative performance between DCCA and DCCAE depends on the data. DGCCA achieved 55.51%, 70.29%, 15.08% F1-score on Caltech101-20, Reuters, NUS-WIDE-OBJ, respectively, and ProxNet improved them considerably by 13.35%, 13.03%, 1.35% (in absolute value). It achieved higher correlation between the views (Eq (17)) after proximal mapping, compared with DGCCA.

Table 1: Comparison of test accuracy (%) and F1-score with state-of-the-art multiview classifiers

	Caltech101-20 (20 classes)		**Reuters** (6 classes)		**NUS-WIDE-OBJ** (31 classes)	
	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score
DCCA	62.37 ± 0.43	46.40 ± 0.42	47.62 ± 0.57	46.49 ± 0.44	17.21 ± 0.19	13.11 ± 0.28
DGCCA	75.18 ± 0.23	55.51 ± 0.78	73.12 ± 0.28	70.29 ± 0.59	21.76 ± 0.22	15.08 ± 0.30
DCCAE	58.43 ± 0.31	43.36 ± 0.52	50.28 ± 0.43	49.55 ± 0.38	20.07 ± 0.20	14.93 ± 0.13
ProxNet	**86.56** ± 0.32	**69.87** ± 0.65	**85.14** ± 0.15	**83.32** ± 0.29	**23.88** ± 0.25	**16.43** ± 0.32



(a) Representation from DGCCA (b) Representation before prox-map (c) Representation after prox-map

Figure 4: t-SNE embedding of the extracted features from test set of Reuters (best viewed in color)

Visualization of representation. To further understand the superior performance of ProxNet and to examine the quality of the learned representation, we plotted in Figure 4 the t-SNE embedding (van der Maaten & Hinton, 2008) of the extracted features for Reuters before and after the proximal mapping, using test set images. This allowed us to examine the effect of proximal mapping.

The raw input, as shown in Figures 7 in Appendix E.1, has heavy overlap between different classes/clusters. Figure 4b shows that the features before proximal mapping exhibit good separation and clustering structures, but there are still some overlap. Interestingly, after proximal mapping, the representations form clusters with smaller intra-class distance and larger inter-class distance, as demonstrated in Figures 4c. Therefore, proximal mapping pushes different classes apart which simplifies the classification task. The representation from the code learned by DGCCA does not have good structure as shown in Figure 4a.

7.2 Multiview Unsupervised Learning: Crosslingual Word Embedding

The goal here is to learn word representations that reflect word similarities, and the multiview approach tries to train on pairs of (English, German) words, so as to transfer in the latent subspace.

We obtained 36K English-German word pairs (training examples) from the parallel news commentary corpora (WMT 2012-2018, Bojar et al., 2018) using the word alignment method from Dyer et al. (2013) and FastAlign. Based on the corpora we also built a bilingual dictionary, where each English word is matched with the (unique) German word that has been most frequently aligned to it. The raw word embedding (x_i and y_i) used the pretrained monolingual 300-dimensional word vectors from fastText (Grave et al., 2018; FastText).

The evaluation was conducted on two commonly used datasets (Leviant & Reichart, 2015; 2019): **a)** multilingual WS353 contains 353 pairs of English words, and their translations to German, Italian and Russian, that have been assigned similarity ratings by humans. It was further split into multilingual WS-SIM and multilingual WS-REL which measure similarity and relatedness between word pairs, respectively; **b)** multilingual SimLex999 consists of 999 English word pairs and their translations.

Algorithms. We compared ProxNet with DCCA and DCCAE, and all of them used multilayer perceptrons with ReLU activation. ProxNet used the input reconstruction error as the ultimate objective, hence *making DCCAE exactly the variant of ProxNet that induces the structure (high CCA correlation) via a regularizer* added to the reconstruction error. A validation set was employed to select the hidden dimension h for f and g from $\{0.1, 0.3, 0.5, 0.7, 0.9\} \times 300$, the regularization parameter λ, and the depth and layer width from 1 to 4 and $\{256, 512, 1024, 2048\}$, respectively. We also compared with linear CCA (Faruqui & Dyer, 2014). At test time, the (English, German) word pairs from the test set were fed to the four multiview based models, extracting the English and German word representations. Then the cosine similarity can be computed between all pairs of monolingual words in the test set (English and German), and we reported in Table 2 the Spearman's correlation between the model's ranking and human's ranking.

Table 2: Spearman's correlation for word similarity

	WS-353		WS-SIM		WS-REL		SimLex999	
	EN	DE	EN	DE	EN	DE	EN	DE
Baseline	73.35	52.68	77.84	63.34	67.66	44.24	37.15	29.09
linearCCA	73.79	68.45	76.06	73.02	67.01	62.95	37.84	43.34
DCCA	73.86	69.09	**78.69**	74.13	66.57	64.66	38.78	43.29
DCCAE	72.39	**69.67**	75.74	74.65	65.96	64.20	36.72	41.81
ProxNet	**75.38**	69.19	78.28	**75.40**	**70.97**	**66.81**	**39.99**	**44.23**
CL-DEPEMB	-	-	-	-	-	-	35.60	30.60

	#train	length	LSTM	AdvLSTM	ProxLSTM
JV	225	15	94.02 ±0.72	94.96 ±0.44	**95.52** ±0.63
HAR	6.1k	128	89.75 ±0.59	**92.01** ±0.18	**92.08** ±0.23
AD	5.5k	39	96.32 ±0.55	97.45 ±0.38	**97.99** ±0.29
IMDB	25k	239	92.65 ±0.04	93.65 ±0.03	**94.16** ±0.11

Table 3: Test accuracy for sequence classification

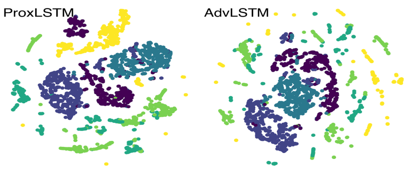


Figure 5: t-SNE embedding of the HAR dataset (best viewed in color)

Results. Clearly, ProxNet always achieves the highest or close to highest Spearman's correlation on all test sets and for both English (EN) and German (DE). We also included a baseline which only uses the monolingual word vectors. CL-DEPEMB is from Vulić (2017), and the paper only provided the results for SimLex999 with no code made available. It can be observed from Table 2 that multiview based methods achieved more significant improvement over the baseline on German data than on English data. This is not surprising, because the presence of multiple views offers an opportunity to transfer useful information from other views/languages. Since the performance on English is generally better than that of German, more improvement is expected on German.

7.3 Adversarial Training in Recurrent Neural Network

We now present the experimental results of adversarial training for LSTMs as described in Section 6.

Datasets. We tested on four sequence datasets: Japanese vowels (JV, Shimbo et al., 1999) which contains time series data for speaker recognition based on uttered vowels; Human Activity Recognition (HAR, Anguita et al., 2013) which classifies activity; Arabic Digits (AD, Hammami & Bedda, 2010) which recognizes digits from speeches; and IMDB (Maas et al., 2011), a large movie review dataset for sentiment classification. Table 3 presents the training set size and *median* sequence length.

Algorithms. We compared ProxLSTM with two baselines: vanilla LSTM and the adversarial training of LSTM (Miyato et al., 2017), which we will refer to as AdvLSTM. For JV, HAR, AD datasets, the base models are preceded by a CNN layer, and succeeded by a fully connected layer. The CNN layer consists of kernels sized 3, 8, 3 and contains 32, 64, 64 filters for the JV, HAR, AD datasets, respectively. LSTM used 64, 128, 64 hidden units for these three datasets, respectively. All these parameters were tuned to optimize the performance of vanilla LSTM, and then shared with ProxLSTM and AdvLSTM for a fair comparison. We first trained the vanilla LSTM to convergence, and used the resulting model to initialize AdvLSTM and ProxLSTM. For IMDB, we first trained AdvLSTM by following the settings in Miyato et al. (2017), and then used the result to initialize the weights of ProxLSTM. All settings were evaluated 10 times to report mean and standard deviation.

Results. From Table 3, it is clear that adversarial training improves test accuracy, and ProxLSTM promotes the performance even more than AdvLSTM. Since the accuracy gap is lowest on the HAR dataset, we also plotted the t-SNE embedding of the features from the *last time step* for HAR. As Figure 5 shows, the representation learned by ProxLSTM is better clustered than that of AdvLSTM, especially the yellow class. This further indicates that ProxLSTM learns better latent representations than AdvLSTM by applying proximal mapping. Plots for other datasets are in Appendix E.3.

Conclusion. In this paper, we proposed using proximal mapping as a new primitive in deep learning, which explicitly encodes the desired structure. Connection to existing constructs in deep learning are shown, and new models for multiview and adversarial recurrent learning are demonstrated effective. Future work will extend it to meta-learning, and reinforcement learning with knowledge transfer.

REFERENCES

B. Amos, I. Rodriguez, J. Sacks, B. Boots, and J. Kolter. Differentiable mpc for end-to-end planning and control. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Brandon Amos and J. Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

Galen Andrew, Raman Arora, Jeff A. Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *International Conference on Machine Learning (ICML)*, 2013.

Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *European Symposium on Artificial Neural Networks*, 2013.

Devansh Arpitz, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In *International Conference on Machine Learning (ICML)*, 2017.

J. A. Bagnell and David M. Bradley. Differentiable sparse coding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2008.

H. H. Bauschke and P. L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, 2011.

Baydin G. Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: A survey. *Journal of Machine Learning Research*, 18(1):5595–5637, 2017.

David Belanger, Bishan Yang, and Andrew McCallum. End-to-end learning for structured prediction energy networks. In *International Conference on Machine Learning (ICML)*, 2017.

Adrian Benton, Huda Khayrallah, Biman Gujral, Dee Ann Reisinger, Sheng Zhang, and Raman Arora. Deep generalized canonical correlation analysis. In *Workshop on Representation Learning for NLP*, 2019.

Frédéric Bernard and Lionel Thibault. Prox-regularity of functions and sets in banach spaces. *Set-Valued Analysis*, 12(1):25–47, Mar 2004.

Luca Bertinetto, Joao F. Henriques, Philip H.S. Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations (ICLR)*, 2019.

Alberto Bietti and Julien Mairal. Group Invariance, Stability to Deformations, and Complexity of Deep Convolutional Representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Alberto Bietti and Julien Mairal. Group invariance, stability to deformations, and complexity of deep convolutional representations. *Journal of Machine Learning Research*, 20(25):1–49, 2019.

Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Matt Post, Lucia Specia, Marco Turchi, Karin Verspoor, and Mark Fishel. News commentary corpus. 2018. `http://www.statmt.org/wmt18`.

Philémon Brakel, Dirk Stroobandt, and Benjamin Schrauwen. Training energy-based models for time-series imputation. *Journal of Machine Learning Research*, 14:2771–2797, 2013.

Caltech101-20. Caltech 101 dataset. `http://www.vision.caltech.edu/Image_Datasets/Caltech101/Caltech101.html`.

P. L. Combettes and J. C. Pesquet. Deep neural network structures solving variational inequalities. *arXiv:1808.07526*, 2018.

Justin Domke. Implicit differentiation by perturbation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2010.

Justin Domke. Generic methods for optimization-based modeling. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. A simple, fast, and effective reparameterization of ibm model 2. In *HLT-NAACL*, 2013.

Manaal Faruqui and Chris Dyer. Improving vector space word representations using multilingual correlation. In *EACL*, 2014.

FastAlign. Fast align toolbox. `https://github.com/clab/fast_align`.

FastText. Pretrained fasttext word vectors. `https://fasttext.cc/docs/en/crawl-vectors.html`.

Ian Goodfellow, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Multi-prediction deep boltzmann machines. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv:1607.05447*, 2016.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

Nacereddine Hammami and Mouldi Bedda. Improved tree model for arabic speech recognition. In *International Conference on Computer Science and Information Technology*, 2010.

Warren Hare and Claudia Sagastizábal. Computing proximal points of nonconvex functions. *Mathematical Programming*, 116(1):221–258, 2009.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012.

Paul Horst. Generalized canonical correlations and their applications to experimental data. *Journalof Clinical Psychology*, 17(4), 1961.

N. Jaitly and G. Hinton. Learning a better representation of speech soundwaves using restricted boltzmann machines. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011.

L. Kantorovitch. On the translocation of masses. *Management Science*, 5(1):1–4, 1958.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

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

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1097–1105, 2012.

Pierre Laforgue, Stéphan Clémençon, and Florence d'Alché-Buc. Autoencoding any data through kernel autoencoders. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-Supervised Nets. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Ira Leviant and Roi Reichart. Separated by an un-common language: Towards judgment language informed vector space modeling. *arXiv:1508.00106*, 2015.

Ira Leviant and Roi Reichart. Multilingual simlex999 and wordsim353 datasets. `http://leviants.com/ira.leviant/MultilingualVSMdata.html`, 2019.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Deep multilingual correlation for improved word embeddings. In *HLT: Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2015.

Yingyi Ma, Vignesh Ganapaman, and Xinhua Zhang. Learning invariant representation with kernel warping. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Association for Computational Linguistics (ACL)*, pp. 142–150. Association for Computational Linguistics, 2011.

J. Mairal, F. Bach, and J. Ponce. Task-driven dictionary learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):791–804, 2012.

Julien Mairal. End-to-end kernel learning with supervised convolutional kernel networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional kernel networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

Alireza Makhzani and Brendan Frey. k-Sparse autoencoders. In *International Conference on Learning Representations (ICLR)*, 2014.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Takeru Miyato, Shin ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing with virtual adversarial training. In *International Conference on Learning Representations (ICLR)*, 2016.

Takeru Miyato, Andrew M. Dai, and Ian Goodfellow. Adversarial training methods for semi-supervised text classification. In *International Conference on Learning Representations (ICLR)*, 2017.

Vlad Niculae and Mathieu Blondel. A regularized framework for sparse and structured neural attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

P. Niyogi, F. Girosi, and T. Poggio. Incorporating prior knowledge in machine learning by creating virtual examples. *Proceedings of the IEEE*, 86(11):2196–2209, November 1998.

NUS-WIDE-OBJ. Nus-wide-obj dataset. `http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm`.

Dipan K. Pal, Ashwin A. Kannan, Gautam Arakalgud, and Marios Savvides. Max-margin invariant features from transformed unlabeled data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3): 127–239, 2014.

T. Poggio and T. Vetter. Recognition and structure from one 2D model view: observations on prototypes, object classes and symmetries. A.I. Memo No. 1347, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1992.

R. A. Poliquin and R. T. Rockafellar. Prox-regular functions in variational analysis. *Transactions of the American Mathematical Society*, 348(5):1805–1838, 1996.

Aravind Rajeswaran, Chelsea Finn, Sham Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Pushpendre Rastogi, Benjamin Van Durme, and Raman Arora. Multiview LSA: Representation learning via generalized CCA. In *HLT: Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, 2015.

Reuters. Reuters dataset. `https://archive.ics.uci.edu/ml/datasets.html`.

Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *International Conference on Machine Learning (ICML)*, 2011.

Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Adversarially robust training through structured gradient regularization. arXiv:1805.08736, 2018.

Masaru Shimbo, Mineichi Kudo, and Jun Toyama. Multidimensional curve classification using passing-through regions. *Pattern Recognition Letters*, 20:1103, 1999.

Patrice Y. Simard, Yann A. LeCun, John S. Denker, and Bernard Victorri. Transformation invariance in pattern recognition – tangent distance and tangent propagation. In Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller (eds.), *Neural Networks: Tricks of the Trade: Second Edition*, pp. 235–269. 2012.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. Information Science and Statistics. 2008.

Veselin Stoyanov, Alexander Ropson, and Jason Eisner. Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

David Stutz, Matthias Hein, and Bernt Schiele. Disentangling adversarial robustness and generalization. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Laurens van der Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, (9):2579–2605, 2008.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning (ICML)*, 2008.

Ivan Vulić. Cross-lingual syntactically informed distributed word representations. In *European Chapter of the Association for Computational Linguistics*, 2017.

S. Wager, S. I. Wang, and P. Liang. Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

S. I. Wang and C. D. Manning. Fast dropout training. In *International Conference on Machine Learning (ICML)*, 2013.

Shenlong Wang, Sanja Fidler, and Raquel Urtasun. Proximal deep structured models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Weiran Wang, Raman Arora, Karen Livescu, and Jeff A. Bilmes. On deep multi-view representation learning. In *International Conference on Machine Learning (ICML)*, 2015.

C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2000.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017.

Handong Zhao, Zhengming Ding, and Yun Fu. Multi-view clustering via deep matrix factorization. In *National Conference of Artificial Intelligence (AAAI)*, 2017.

Joey Tianyi Zhou, Kai Di, Jiawei Du, Xi Peng, Hao Yang, Sinno Jialin Pan, Ivor Wai-Hung Tsang, Yong Liu, Zheng Qin, and Rick Siow Mong Goh. Sc2net: Sparse lstms for sparse coding. In *National Conference of Artificial Intelligence (AAAI)*, 2018.

X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *International Conference on Machine Learning (ICML)*, pp. 912–919, 2003.

A Relationship with OptNet and Implicit Differentiation Based Learning

Given a prediction model such as linear model, energy-based model, kernel function, deep neural network, etc, a loss function is needed to measure the quality of its prediction against the given ground truth. Although surrogate losses had been popular in making the loss convex, recently it is often observed that directly comparing the prediction of the model, typically computed through an argmin optimization (or argmax), against the ground truth under the true loss of interest can be much more effective. The error signal is originated from the *last step* through the argmin, and then backpropagated through the model itself for training. For example, Amos et al used it to train input convex neural networks at ICML 2017, Belanger et al. (2017) used it to train a structured prediction energy network, and Brakel et al. (2013) used it to train an energy-based model for time-series imputation. Other works include Stoyanov et al. (2011); Goodfellow et al. (2013), etc. A number of implicit and auto-differentiation algorithms have been proposed for it, e.g., Domke (2010; 2012); Baydin et al. (2017); Amos & Kolter (2017); Gould et al. (2016).

Other uses of such differentiable optimization have been found in learning attention models (Niculae & Blondel, 2017), meta-learning to differentiate through the base learning algorithm (Bertinetto et al., 2019; Lee et al., 2019), or to train the generator in a generative adversarial model by optimizing out the discriminator (Metz et al., 2017), or for end-to-end planning and control (Amos et al., 2018). In all these cases, differentiable optimization is used as an algorithm to train a *given* component within a multi-component learning paradigm. But each component itself has its own pre-fixed model and parameterization.

To the best of our knowledge, OptNet (Amos & Kolter, 2017) proposed for the first time using optimization as a *layer* of the deep neural network, hence extending the model itself. However, it focused on efficient algorithms for differentiation[1], and the general framework of optimization layer was demonstrated by using standard operations such as total variation denoising, which bears resemblance to task-driven dictionary learning (Bagnell & Bradley, 2008; Mairal et al., 2012). It remains unclear how to leverage the general framework of OptNet to flexibly model a broad range of structures, while reusing the existing primitives in deep learning (like our extension of LSTM in Section 6).

This is achieved by ProxNet. Although ProxNet also inserts a new layer, it provides *concrete and novel* ways to model structured priors in data through proximal mapping. Most aforementioned works use differentiable optimization as a learning algorithm for a *given* model, while ProxNet uses it as a first-class modeling construct within a deep network. Designing the potential function f in (1) can be highly nontrivial, as we have demonstrated in the examples of dropout, kernel warping, multiview learning, and LSTM.

Rajeswaran et al. (2019) used proximal mapping for the inner-level optimization of meta-learning, which constitutes a bi-level optimization. Their focus is to streamline the optimization using implicit gradient, while our goal, in contrast, is to use proximal mapping to learn structured data representations.

We note that despite the similarity in titles, Wang et al. (2016) differs from our work as it applies proximal mapping in a solver to perform inference in a graphical model, whose cliques are neural networks. The optimization process *happens to* be analogous to a recurrent net, interlaced with proximal maps, and similar analogy has been drawn between the ISTA optimization algorithm and LSTM (Zhou et al., 2018). We instead use proximal map as a first-class construct/layer in a deep network.

B Simulations for Reinterpreting Dropout as Proximal Mapping

We next use the two-moon dataset to verify that the slight approximation in Section 4 only leads to small differences in dropout and proximal mapping. Suppose the i-th training examples is $x_i \in \mathbb{R}^d$ with label $y_i \in \{-1, 1\}$. The j-th feature of x_i is denoted as x_{ij}. Employing logistic loss, the

[1]Although the original paper only detailed on quadratic optimization mainly for the efficient GPU implementation, it is conceptually applicable to general nonlinear optimization.

adaptive regularization view of dropout by Wager et al. (2013) can be written as

$$\beta_* := \min_{\beta \in \mathbb{R}^d} \left\{ \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-y_i \beta^\top x_i) + \mu \sum_j a_j \beta_j^2 \right\}, \tag{12}$$

$$\text{where} \quad a_j = \frac{1}{n} \sum_{i=1}^{n} p_i(1-p_i) x_{ij}^2, \quad p_i = (1 + \exp(-\beta^\top x_i))^{-1}. \tag{13}$$

Our proximal map is defined as

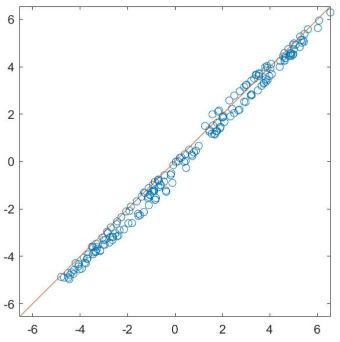
$$\mathsf{P}_L(x) = \arg\min_{z \in \mathbb{R}^d} \left\{ \frac{\lambda}{2} \|z - x\|_2^2 + \sum_j b_j z_j^2 \right\}, \tag{14}$$

$$\text{where} \quad b_j = \frac{1}{n} \sum_{i=1}^{n} q_i(1-q_i) x_{ij}^2, \quad q_i = (1 + \exp(-z^\top x_i))^{-1}. \tag{15}$$

And the output layer is trained by

$$\alpha_* := \min_{\alpha \in \mathbb{R}^d} \left\{ \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-y_i \alpha^\top \mathsf{P}_L(x_i)) + c \|\alpha\|^2 \right\}. \tag{16}$$

To demonstrate that the two methods yield similar discriminant values, we produce a scatter plot of $\alpha_*^\top P_L(x_i)$ (for proximal mapping) versus $\beta_*^\top x_i$ (for dropout). Figure 6 shows the result for two example settings. Clearly, the two methods produce similar discriminant values for all training examples. The Matlab code is also available on GitHub.


(a) $\lambda = 0.5$, $\mu = 0.1$, and $c = 0.2\lambda^2\mu$

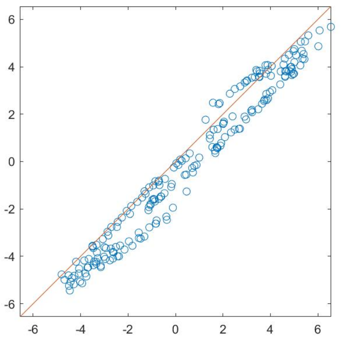

(b) $\mu = 0.1$, $\lambda = 0.1$, and $c = 15\lambda^2\mu$

Figure 6: Scatter plot of $\alpha_*^\top P_L(x_i)$ (y-axis for proximal mapping) versus $\beta_*^\top x_i$ (x-axis for dropout)

C ProxNet for Multiview Learning

Most multiview learning algorithms are based on CCA, which most commonly involves only two views. It is in fact not hard to extend it to more than two views. For example, Horst (1961) proposed that given J centered views $X_j \in \mathbb{R}^{N \times d_j}$ for $j \in [J]$, where N is the number of training examples and d_j is the dimensionality of the j-th view, the generalized CCA (GCCA) can be written as the following optimization problem

$$L(\{X_j\}_{j=1}^J) := \min \left\{ \sum_{j=1}^{J} \|G - X_j U_j\|_F^2 : G \in \mathbb{R}^{N \times r},\ U_j \in \mathbb{R}^{d_j \times r},\ G^\top G = I \right\}. \tag{17}$$

Intuitively, it finds a linear transformation U_j for each view, so that all views can be transformed to a similar core G. Furthermore, G needs to be orthonormal, to avoid mode collapse. The optimal value, denoted as $L(\{X_j\})$, will be used as the L function in (9).

Furthermore, given $\{X_j\}$, (17) can be optimized efficiently in closed form based on generalized eigenvalues (Rastogi et al., 2015; Horst, 1961; Benton et al., 2019). Based on the optimal solution of G and $\{U_j\}$, the derivative of $L(\{X_j\})$ in $\{X_j\}$ can be directly computed by Danskin's theorem.

Backpropagation through the proximal mapping in (9) requires that given $\frac{\partial J}{\partial P}$ and $\frac{\partial J}{\partial Q}$ where J is the ultimate objective value, compute $\frac{\partial J}{\partial X}$ and $\frac{\partial J}{\partial Y}$. The most general solution has been provided by OptNet (Amos & Kolter, 2017), but the structure of our problem is amenable to a simpler solution in Domke (2010). With small $\epsilon > 0$, $\big(\frac{\partial J}{\partial X}, \frac{\partial J}{\partial Y}\big) \approx \frac{1}{\epsilon}\big(\mathsf{P}_L(X + \epsilon\frac{\partial J}{\partial P}, Y + \epsilon\frac{\partial J}{\partial Q}) - \mathsf{P}_L(X, Y)\big)$. More accurate recipes can be found in Domke (2010).

D Backpropagation Through Time for Adversarial LSTM

To concentrate on backpropagation, we assume that the ultimate objective J only depends only on the output of the last time step T, i.e., h_T. Extension can be easily made to the case where each step also contributes to the overall loss. From the final layer, we get $\frac{\partial J}{\partial h_T}$. Then we can get $\frac{\partial J}{\partial h_{T-1}}$ and $\frac{\partial J}{\partial c_{T-1}}$ as in the standard LSTM (G_T in the final layer can be ignored and $\frac{\partial J}{\partial c_T} = 0$). In order to compute the derivatives with respect to the weights W in the LSTMs, we need to recursively compute $\frac{\partial J}{\partial h_{t-1}}$ and $\frac{\partial J}{\partial c_{t-1}}$, given $\frac{\partial J}{\partial h_t}$ and $\frac{\partial J}{\partial c_t}$. Once they are available, then

$$\frac{\partial J}{\partial W} = \sum_{t=1}^{T} \left\{ \underbrace{\frac{\partial J}{\partial h_t}}_{\text{by (19)}} \underbrace{\frac{\partial}{\partial W} h_t(c_{t-1}, h_{t-1}, x_t)}_{\text{standard LSTM}} + \underbrace{\frac{\partial J}{\partial c_t}}_{\text{by (21)}} \underbrace{\frac{\partial}{\partial W} c_t(c_{t-1}, h_{t-1}, x_t)}_{\text{standard LSTM}} \right\}, \tag{18}$$

where the two $\frac{\partial}{\partial W}$ on the right-hand side are identical to the standard operations in LSTMs. Here we use the Jacobian matrix arrangement for partial derivatives, i.e., if f maps from $\mathbb{R}^n$ to $\mathbb{R}^m$, then $\frac{\partial f(x)}{\partial x} \in \mathbb{R}^{m \times n}$.

Given $\frac{\partial J}{\partial c_t}$, we can first compute $\frac{\partial J}{\partial s_t}$ and $\frac{\partial J}{\partial G_t}$ based on the proximal map, and the details will be provided in Section D.1. Given their values, we now compute $\frac{\partial J}{\partial h_{t-1}}$ and $\frac{\partial J}{\partial c_{t-1}}$. Firstly,

$$\frac{\partial J}{\partial h_{t-1}} = \underbrace{\frac{\partial J}{\partial h_t}}_{\text{by recursion}} \underbrace{\frac{\partial h_t}{\partial h_{t-1}}}_{\text{std LSTM}} + \underbrace{\frac{\partial J}{\partial G_t} \frac{\partial G_t}{\partial h_{t-1}}}_{\text{by (20)}} + \underbrace{\frac{\partial J}{\partial s_t}}_{\text{by (26)}} \underbrace{\frac{\partial s_t}{\partial h_{t-1}}}_{\text{std LSTM}}. \tag{19}$$

The terms $\frac{\partial h_t}{\partial h_{t-1}}$ and $\frac{\partial s_t}{\partial h_{t-1}}$ are identical to the operations in the standard LSTM. The only remaining term is in fact a directional second-order derivative, where the direction $\frac{\partial J}{\partial G_t}$ can be computed from from (35):

$$\frac{\partial J}{\partial G_t} \frac{\partial G_t}{\partial h_{t-1}} = \frac{\partial J}{\partial G_t} \frac{\partial^2}{\partial x_t \partial h_{t-1}} s_t(c_{t-1}, h_{t-1}, x_t) = \frac{\partial}{\partial h_{t-1}} \left\langle \underbrace{\frac{\partial J}{\partial G_t}}_{\text{by (35)}}, \frac{\partial}{\partial x_t} s_t(c_{t-1}, h_{t-1}, x_t) \right\rangle. \tag{20}$$

Such computations are well supported in most deep learning packages, such as PyTorch. Secondly,

$$\frac{\partial J}{\partial c_{t-1}} = \underbrace{\frac{\partial J}{\partial h_t}}_{\text{by recursion}} \underbrace{\frac{\partial h_t}{\partial c_{t-1}}}_{\text{std LSTM}} + \underbrace{\frac{\partial J}{\partial G_t} \frac{\partial G_t}{\partial c_{t-1}}}_{\text{by (22)}} + \underbrace{\frac{\partial J}{\partial s_t}}_{\text{by (26)}} \underbrace{\frac{\partial s_t}{\partial c_{t-1}}}_{\text{std LSTM}}. \tag{21}$$

The terms $\frac{\partial h_t}{\partial c_{t-1}}$ and $\frac{\partial s_t}{\partial c_{t-1}}$ are identical to the operations in the standard LSTM. The only remaining term is in fact a directional second-order derivative:

$$\frac{\partial J}{\partial G_t}\frac{\partial G_t}{\partial c_{t-1}} = \frac{\partial J}{\partial G_t}\frac{\partial^2}{\partial x_t \partial c_{t-1}} s_t(c_{t-1}, h_{t-1}, x_t) = \frac{\partial}{\partial c_{t-1}} \left\langle \underbrace{\frac{\partial J}{\partial G_t}}_{\text{by (35)}}, \frac{\partial}{\partial x_t} s_t(c_{t-1}, h_{t-1}, x_t) \right\rangle. \quad (22)$$

D.1 Gradient Derivation for the Proximal Map

We now compute the derivatives involved in the proximal operator, namely $\frac{\partial J}{\partial s_t}$ and $\frac{\partial J}{\partial G_t}$. For clarify, let us omit the step index t, set $\delta = \sqrt{\lambda}$ without loss of generality, and denote

$$J = f(c), \quad \text{where} \quad c := c(G, s) := (I + GG^\top)^{-1} s. \quad (23)$$

We first compute $\partial J / \partial s$ which is easier.

$$\Delta J := f(c(G, s + \Delta s)) - f(c(G, s)) = \nabla f(c)^\top (c(G, s + \Delta s) - c(G, s)) + o(\|\Delta s\|) \quad (24)$$

$$= \nabla f(c)^\top (I + GG^\top)^{-1} \Delta s + o(\|\Delta s\|). \quad (25)$$

Therefore,

$$\frac{\partial J}{\partial s} = \nabla f(c)^\top (I + GG^\top)^{-1}. \quad (26)$$

We now move on to $\partial J / \partial G$. Notice

$$\Delta J := f(c(G + \Delta G, s)) - f(c(G, s)) = \nabla f(c)^\top (c(G + \Delta G, s) - c(G, s)) + o(\|\Delta G\|). \quad (27)$$

Since

$$c(G + \Delta G, s) = (I + (G + \Delta G)(G + \Delta G)^\top)^{-1} s \quad (28)$$

$$= \left[(I + GG^\top)^{\frac{1}{2}} \left(I + (I + GG^\top)^{-\frac{1}{2}} (\Delta G G^\top + G \Delta G^\top)(I + GG^\top)^{-\frac{1}{2}} \right) (I + GG^\top)^{\frac{1}{2}} \right]^{-1} s \quad (29)$$

$$= (I + GG^\top)^{-\frac{1}{2}} \left(I - (I + GG^\top)^{-\frac{1}{2}} (\Delta G G^\top + G \Delta G^\top)(I + GG^\top)^{-\frac{1}{2}} + o(\|\Delta G\|) \right) (I + GG^\top)^{-\frac{1}{2}} s \quad (30)$$

$$= c(G, s) - (I + GG^\top)^{-1} (\Delta G G^\top + G \Delta G^\top)(I + GG^\top)^{-1} s + o(\|\Delta G\|), \quad (31)$$

we can finally obtain

$$\Delta J = -\nabla f(c)^\top (I + GG^\top)^{-1} (\Delta G G^\top + G \Delta G^\top)(I + GG^\top)^{-1} s + o(\|\Delta G\|) \quad (32)$$

$$= -\operatorname{tr}\left(\Delta G^\top (I + GG^\top)^{-1} \left(\nabla f(c) s^\top + s \nabla f(c)^\top \right) (I + GG^\top)^{-1} G \right) + o(\|\Delta G\|). \quad (33)$$

So in conclusion,

$$\frac{\partial J}{\partial G} = -(I + GG^\top)^{-1} \left(\nabla f(c) s^\top + s \nabla f(c)^\top \right) (I + GG^\top)^{-1} G \quad (34)$$

$$= -(ac^\top + ca^\top) G, \quad \text{where} \quad a = (I + GG^\top)^{-1} \nabla f(c). \quad (35)$$

E Detailed Experimental Result

E.1 Multiview Experiment for Supervised Learning

Dataset. Caltech101-20 has 6 views, whose dimensions are 48, 40, 254, 1984, 512, 928. Reuters has 5 views, whose dimensions are 21531, 24892, 34251, 15506, 11547. NUS-WIDE-OBJ has five views, whose dimensions are 65, 226, 145, 74, 129.

Competing algorithms. In this task, our experiments were designed to show that ProxNet outperforms state-of-the-art algorithms for supervised multiview learning. So we chose four algorithms for comparison:

- DCCA (Andrew et al., 2013), which learns a nonlinear transformation of the two views to optimize the CCA score;
- DCCAE (Wang et al., 2015), which demonstrates that correlation-based representation learning is an efficient way to improve the performance;
- DGCCA (Benton et al., 2019), a generalization of DCCA which allows more than two views.

Implementation Details. Our implementation was based on PyTorch and all experiments were conducted on two NIVIDIA GeForce 1080 Ti GPU. We trained ProxNet using three fully-connected layers as the feature map. Then the proximal layer maps from $\mathbb{R}^{10}$ to $\mathbb{R}^{10}$, followed with three fully-connected layers each having 512 sigmoid units. The final output layer has softmax units that correspond to the output classes. We empirically used the mini-batch size 200, the subspace dimensionality $k = 10$ and the trade-off parameter λ via grid search over $\{0.001, 0.01, 0.1, 1, 10\}$, allowing a trade-off between correlation term and Frobenius norm term.

In training, for all datasets, we used Adam with a weight decay of 10^{-4} and learning rate of 0.001, and divided it by 10 after 50/200/300 epochs for Reuters/Caltech101-20/NUS-WIDE-OBJ, respectively.

Visualization of representation. The test examples from Reuters, as shown in Figure 7, exhibit heavy overlap between different classes/clusters which also result in a challenging classification task.

We did not plot for Caltech101-20 and NUS-WIDE-OBJ because they have too many classes for visualization (20 and 31, respectively), and their classes are very imbalanced.

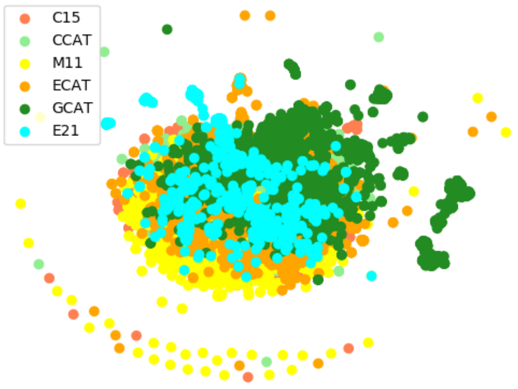


Figure 7: t-SNE embedding of the inputs of Reuters

E.2 CROSSLINGUAL/MULTILINGUAL WORD EMBEDDING

In this task, we learned representation of English and German words from the paired (English, German) word embeddings for improved semantic similarity.

Dataset. We first built a parallel vocabulary of English and German from the parallel news commentary corpora (WMT 2012-2018 Bojar et al., 2018) using the word alignment method from FastAlign; Dyer et al. (2013). Then we selected 36K English-German word pairs, in descending order of frequency, for training. Based on the vocabulary we also built a bilingual dictionary for testing, where each English word x_i is matched with the (unique) German word y_i that has been most frequently aligned to x_i. Unlike the setup in Faruqui & Dyer (2014) and Wang et al. (2015), where word embeddings are trained via Latent Semantic Analysis (LSA) using parallel corpora, we used the pretrained monolingual 300-dimensional word embedding from FastText and Grave et al. (2018) as the raw word embeddings (x_i and y_i).

To evaluate the quality of learned word representation, we experimented on two different benchmarks that have been widely used to measure word similarity (Leviant & Reichart, 2019; 2015). Multilingual WS353 contains 353 pairs of English words, and their translations to German, Italian and Russian, that have been assigned similarity ratings by humans. It was further split into Multilingual WS-SIM and Multilingual WS-REL which measure the similarity and relatedness between word pairs respectively. Multilingual SimLex999 is a similarity-focused dataset consisting of 666 noun pairs, 222 verb pairs, 111 adjective pairs, and their translations from English to German, Italian and Russian.

Baselines. We compared our method with the monolingual word embedding (baseline method) from fastText to show that ProxNet learned a good word representation through the proximal layer. Since our method is mainly based on CCA, we also chose three competitive CCA-based models for comparison, including:

- linearCCA (Faruqui & Dyer, 2014), which applied a linear projection on the two languages' word embedding and then projected them into a common vector space such that aligned word pairs should be maximally correlated.
- DCCA (Lu et al., 2015), which, instead of learning linear transformations with CCA, learned nonlinear transformations of two languages' embedding that are highly correlated.
- DCCAE (Wang et al., 2015), which noted that there is useful information in the original inputs that is not correlated across views. Therefore, they not only projected the original embedding into subspace, but also reconstructed the inputs from the latent representation.
- CL-DEPEMB (Vulić, 2017), a novel cross-lingual word representation model which injects syntactic information through dependency-based contexts into a shared cross-lingual word vector space.

Implementation details. We first used the fastText model to embed the 36K English-German word pairs into vectors. Then we normalized each vector to unit ℓ_2 norm and removed the per-dimension mean and standard deviation of the training pairs.

To build an end-to-end model, we followed the same intuition as DCCAE but instead of using the latent representation from the encoder to reconstruct the inputs, we used the outputs of proximal layer, which is a proximal approximation of latent representation from the encoder, to do the reconstruction. That is, the input reconstruction error was used as the ultimate objective.

We implemented the encoder (feature mapping f and g) by using multilayer perceptrons with ReLU activation and the decoder by using a symmetric architecture of encoder. We tuned the hidden dimension h for f and g among $\{0.1, 0.3, 0.5, 0.7, 0.9\} \times 300$, the regularization parameter λ from $\{0.001, 0.01, 0.1, 1, 10\}$, and the depth and layer width from 1 to 4 and $\{256, 512, 1024, 2048\}$, respectively. For optimization, we used SGD with momentum 0.99, a weight decay of 0.0005, and a learning rate 0.1 which was divided by 10 after 100 and 200 epochs.

At test time, for numerical stability, we combined the word vectors from bilingual dictionary and the test set to build paired vocabulary for each language. We applied the same data preprocessing (normalize to unit norm, remove the mean/standard deviation of the training set) on test vocabularies (English and German word vectors). Then we fed paired test vocabularies into the models and obtained the word representation of the test data. We projected the output of the proximal layer to the subspace where each paired word representation was maximally correlated. The projection matrices were calculated from the 36K training set through the standard CCA method. We computed the cosine similarity between the final word vectors in each pair, ordered the pairs by similarity, and computed the Spearman's correlation between the model's ranking and human's ranking.

E.3 Adversarial Training in Recurrent Neural Network

Here we include more details on the experiment of adversarial training in recurrent neural network as described in Section 7.3.

Datasets. To demonstrate the effectiveness of using proximal mapping, we tested on four different sequence datasets. The Janpanese Vowels dataset (JV Shimbo et al., 1999) contains time series data where nine male speakers uttered Japanese Vowels successively, and the task is to classify speakers.

Table 4: Summary of datasets for adversarial LSTM training

Dataset	Training	Test	Median length	Attributes	Classes
JV	225	370	15	12	9
HAR	6,127	2,974	128	9	6
AD	5,500	2,200	39	13	10
IMDB	25,000	25,000	239	-	2

The Human Activity Recognition dataset (HAR Anguita et al., 2013) is used to classify a person's activity (sitting, walking, etc.) based on a trace of their movement using sensors. The Arabic Digits dataset (AD, Hammami & Bedda, 2010) contains time series corresponding to spoken Arabic digits by native speakers, and the task is to classify digits. IMDB (Maas et al., 2011) is a standard movie review dataset for sentiment classification. Details of the datasets are summarized in Table 4. The - is because IMDB is a text dataset, for which a 256-dimensional word embedding is learned.

Preprocessing. Normalization was the only preprocessing applied to all datasets. For those datasets that contain variable-length sequences, zero-padding was used to make all sequences have the same length as the longest sequence in a mini-batch. To reduce the effect of padding, we first sorted all sequences by length (except the IMDB dataset), so that sequences with similar length were assigned to the same mini-batch.

Baseline models. To show the impact of applying proximal mapping on LSTM, we compared our method with two baselines. For JV, HAR and AD datasets, the base model structure was composed of a CNN layer, followed by an LSTM layer and a fully-connected layer. The CNN layer was constructed with kernel size 3, 8, 3 and contained 32, 64, 64 filters for JV, HAR, AD respectively. For the LSTM layer, the number of hidden units used in these three datasets are 64, 128, 64, respectively. This architecture was denoted as LSTM in Table 3. For IMDB, following Miyato et al. (2017), the basic model consisted of a word embedding layer with dimension 256, a single-layer LSTM with 1024 hidden units, and a hidden dense layer of dimension 30.

On top of this basic LSTM structure, we compared two different adversarial training methods. AdvLSTM is the adversarial training method in Miyato et al. (2017), which we reimplemented in PyTorch, and perturbation was added to the input of each LSTM layer. ProxLSTM denotes our method described in Section 6, where the LSTM cell in the basic structure was replaced by our ProxLSTM cell. LSTM and AdvLSTM here correspond to "Baseline" and "Adversarial" in Miyato et al. (2017) respectively.

Training. For the JV, HAR, AD datasets, we first trained the baseline LSTM to convergence, and then applied AdvLSTM and ProxLSTM as fine tunning, where ADAM was used with learning rate 10^{-3} and weight decay 10^{-4}. For IMDB, we first trained LSTM and AdvLSTM by following the settings in Miyato et al. (2017), with an ADAM optimizer of learning rate $5 \cdot 10^{-4}$ and exponential decay 0.9998. Then the result of AdvLSTM was used to initialize the weights of ProxLSTM. All settings were evaluated 10 times to report the mean and standard deviation.

Results. The test accuracies were summarized in Table 3. Clearly, adversarial training improves the performance, and ProxLSTM even promotes the performance more than AdvLSTM. Figure 5 illustrates the t-SNE embedding of extracted features from the last time step's hidden state of HAR test set. Although ProxLSTM only improves upon AdvLSTM marginally in test accuracy, Figure 5 shows the embedded features from ProxLSTM cluster more compactly than those of AdvLSTM (e.g. the yellow class). The t-SNE plot of other datasets are available in Figures 8, 9 and 10. This further indicates that ProxLSTM can learn better latent representation than AdvLSTM by applying proximal mapping.

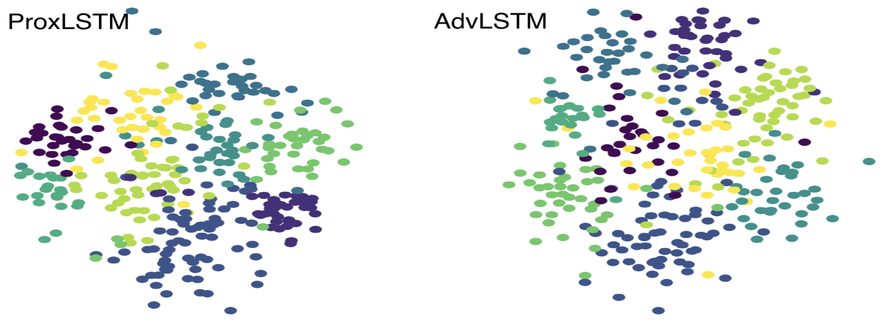


Figure 8: t-SNE embedding of the JV dataset

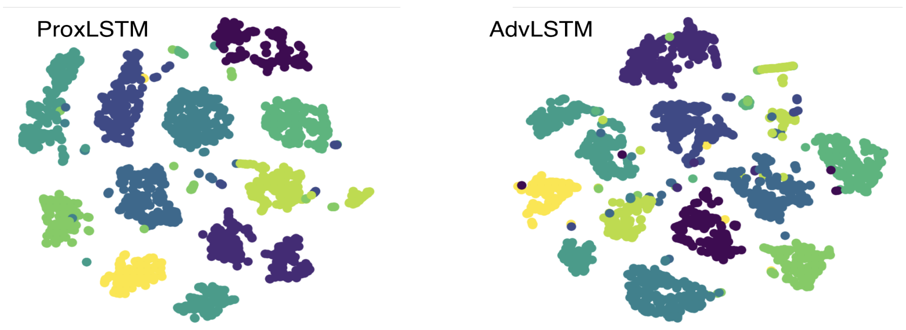


Figure 9: t-SNE embedding of the AD dataset

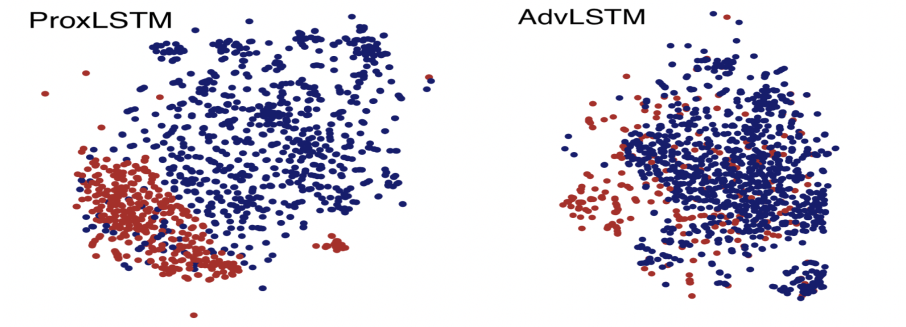


Figure 10: t-SNE embedding of the IMDB dataset