# Meta-Learning of Structured Representation by Proximal Mapping

**Mao Li, Yingyi Ma, Xinhua Zhang**
Department of Computer Science, University of Illinois at Chicago
Chicago, IL 60607
{mli206,yma36,zhangx}@uic.edu

## Abstract

Underpinning the success of deep learning are the effective regularization techniques that allow a broad range of structures in data to be compactly modeled in a deep architecture. Examples include transformation invariances 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 based on meta-learning, which allows prior structures to be encoded *explicitly* in the hidden layers, and to be trained efficiently in conjunction with the supervised learning task. It extends meta-learning to unsupervised base learners. The resulting technique is applied to generalize dropout and invariant kernel warping, and to develop novel algorithms for multiview modeling and robust temporal learning.

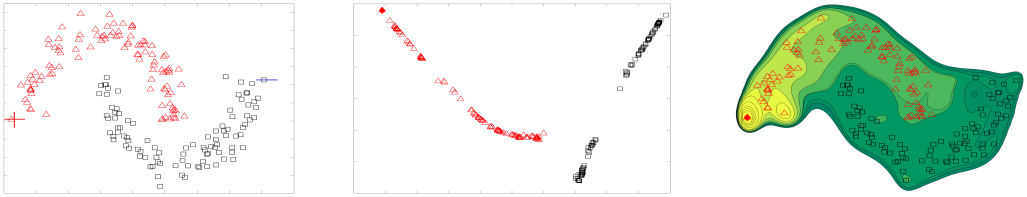
## 1 Introduction

The success of deep learning relies on massive neural networks that often considerably out-scale the training dataset, defying the conventional learning theory [1, 2]. Regularizations have been shown to be essential and a variety of forms are available. Standard invariances to transformations such as rotation and translation [3] have been extended beyond group-based diffeomorphisms to indecipherable transformations only exemplified by pairs of views [4], e.g., sentences uttered by the same person. Structural regularities are also commonly present a) *within* layers of neural networks, such as sparsity [5], spatial invariance in convolutional nets, structured gradient that accounts for data covariance [6], and manifold smoothness characterized by a graph [7]; and b) *between* layers of representation, such as stability under dropout and adversarial perturbations of preceding layers [8], contractivity between layers [9], and correlations in hidden layers among multiple views [10, 11].

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 [12–14]. 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 [15, 16]. 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 finding the separating boundary is easy for graph-based semi-supervised learning [17], it does not uncover any manifold-aware representation of the data.

This limitation has been well addressed by auto-encoders that extract the most salient features to reconstruct the input, while also encoding a variety of latent structures such as layer-wise contractivity

3rd Workshop on Meta-Learning at NeurIPS 2019, Vancouver, Canada.



(a) Original two-moon dataset (b) Representation after MetaProx (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 the proximal map based on unsupervised gradient flatness (Equation 8 with Gaussian kernel); (c) contour of distance to the leftmost point ♦, based on the representation from proximal mapping.

[9], noise resilience [18–20], correlations between views [10], sparsity in latent code [5], on-manifold robustness [21], and graphical interconnections in node embeddings [22]. However most of these methods are not amenable to end-to-end learning, and are used as an initialization for the subsequent 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 *meta-learning* based on proximal mapping (hence the name MetaProx), which has been extensively used in optimization to enforce structured solutions (e.g., sparsity) at each iteration [23], but surprisingly not yet in modeling deep neural networks. Given a closed convex set $C \subseteq \mathbb{R}^n$ 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 $L_2$. $f$ and $C$ can be nonconvex, making $\mathsf{P}_f(z)$ set valued [24–26], and we will only need differentiation at one element. In essence, $f$ and $C$ promote the result of the mapping to capture the desired structure, while staying close to the original $z$. For example, Figures 1b and 1c show the resulting representation and distance metric of the two-moon data where $f$ accounts for the underlying manifold, making the classification problem trivial (see Section 2). In practice, this proximal layer can be conveniently inserted into a deep network, proffering the favorable decoupling of regularization and 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.

Motivated by meta-learning, $z$ in (1) may consist of the embeddings of input objects in *mini-batches*, where each mini-batch can be considered as a "task" (or dataset, episode, etc) in the standard meta-learning terminology, and the structure-inducing transformation it undergoes corresponds to the task-specific base learner inside each iteration of the meta learner. For example, [27–29] used simple metric-based nearest neighbor, [30, 31] optimized standard learning algorithms iteratively, and [32, 33] leveraged closed-form solutions for base learners. Explicit learning of learner's update rule was investigated in [34–36]. In this sense, MetaProx extends meta-learning to *unsupervised* base learners. As a concrete example, we will instantiate MetaProx in the context of multiview learning in Section 3, where $f$ takes the form of the canonical correlation analysis objective. Efficient algorithms for solving the resulting proximal mapping will be designed, and superior empirical performance will be demonstrated in Section 4.

## 2 Proximal Mapping as a Primitive Construct in Deep Networks

Proximal map is highly general, encompassing most primitive operations in deep learning [23, 37]. For example, it is easy to verify that for any activation function $\sigma$ 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 $\mathbb{R}^n \ni x \mapsto (e^{x_1}, \ldots, e^{x_n})^\top / \sum_i e^{x_i}$ 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 $\mu$ and $\sigma$ 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 $x \neq 0$.

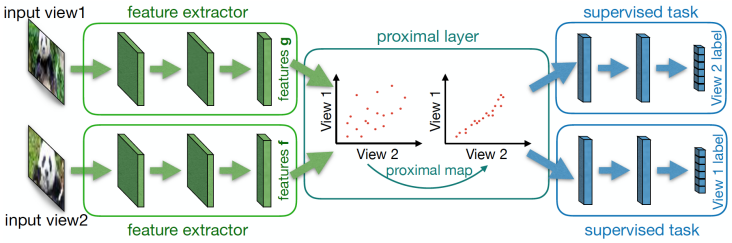



Figure 2: Multi-view learning with a proximal layer for CCA based meta-learning.

Essentially, MetaProx falls in the differentiable optimization framework laid by OptNet [38] along with [32, 33, 39–48], which provides recipes for differentiating through an optimization layer. However, our focus is not on optimization, but on using MetaProx to model the prior structures in the data, which typically involves an (inner) unsupervised learning task such as CCA. Detailed discussions on the relationship between MetaProx and OptNet or related works are available in Appendix C.

**Kernelized deep neural networks.** Clearly MetaProx can be extended to reproducing kernel Hilbert spaces (RKHS), allowing non-vectorial data to be encoded [49], and some *prior* invariances to be hard wired [50–52]. Assume the RKHS $\mathcal{H}$ is induced by a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ on input space $\mathcal{X}$. Given a convex functional $L : \mathcal{H} \to \mathbb{R}$, the proximal map $\mathsf{P}_L : \mathcal{H} \to \mathcal{H}$ is defined as [53]

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

An example of $L$ that captures data manifold is the graph Laplacian $L(f) := \sum_{ij} w_{ij}(f(x_i) - f(x_j))^2$. Parameterizing images as $I(\alpha)$ where $\alpha$ is the degree of rotation, translation, etc, transformation invariance favors a small absolute value of $L(f) := \frac{\partial}{\partial \alpha}|_{\alpha=0} f(I(\alpha))$. Figure 1b was obtained simply with $\mathsf{P}_L(k(x, \cdot))$ where $L(f) = \sum_i \|\nabla f(x_i)\|^2$, followed by kernel PCA. More details are given in Appendix A, where MetaProx recovers the kernel warping for invariance modeling in [54].

## 3 MetaProx 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, 11], 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 $H = I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$, CCA finds $U = (u_1, \ldots, u_k)$ and $V = (v_1, \ldots, v_k)$ that maximize the correlation:

$$\min_{U,V} -\operatorname{tr}(U^\top X Y^\top V), \;\; \text{s.t} \;\; U^\top X X^\top U = I, \; V^\top Y Y^\top V = I, \; u_i^\top X Y^\top v_j = 0, \forall\, i \neq j. \tag{3}$$

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, 10] 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, omitting regularizers, can be written as $\min_{f,g,h} \sum_i \ell(h(p_i, q_i), c_i)$ with

$$\{(p_i, q_i)\}_{i=1}^n = \mathsf{P}_L(X, Y) := \arg\min_{P,Q} \; \tfrac{\lambda}{2} \|P - X\|_F^2 + \tfrac{\lambda}{2} \|Q - Y\|_F^2 + L(P, Q), \tag{4}$$

where $\|\cdot\|_F$ stands for the Frobenious norm. The proximal mapping is performed in mini-batches, and the entire framework is illustrated in Figure 2.

**Optimization and BP.** Although efficient closed-form solution is available to the CCA objective in (3), none exists for the proximal mapping in (4). 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, then [11] shows that $L(X, Y) = -\sum_{i=1}^k \sigma_i(T)$, where $\sigma_i$ 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 constant to enable inversion. Then (4) can be easily solved by gradient descent or L-BFGS. The gradient of $\sum_{i=1}^k \sigma_i(T(P, Q))$ is available from [11], 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 SVD and computing $T$ is small.

Table 1: 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 |
| MetaProx | **75.38** | 69.19 | 78.28 | **75.40** | **70.97** | **66.81** | **39.99** | **44.23** |
| CL-DEPEMB | - | - | - | - | - | - | 35.60 | 30.60 |

BP 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 [38], but the structure of our problem is amenable to a simpler solution in [39]. With small $\epsilon > 0$, $\left(\frac{\partial J}{\partial X}, \frac{\partial J}{\partial Y}\right) \approx \frac{1}{\epsilon}\big(\mathsf{P}_L(X+\epsilon\frac{\partial J}{\partial P}, Y+\epsilon\frac{\partial J}{\partial Q}) - \mathsf{P}_L(X,Y)\big)$.

# 4 Experiments

We evaluated the empirical performance of MetaProx for multiview learning on cross-lingual word embedding. Details on dataset preprocessing, experiment setting, optimization parameters, and additional results are available in Appendix E. Here we highlight the major results and experiment setup for crosslingual/multilingual word embedding, where the goal 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.

**Datasets.** We obtained 36K English-German word pairs (training examples) from the parallel news commentary corpora [WMT 2012-2018, 55] using the word alignment method from [56, 57]. Based on the corpora we also built a bilingual dictionary, where each English word x is matched with the (unique) German word that has been most frequently aligned to x. The raw word embedding ($x_i$ and $y_i$) used the pretrained monolingual 300-dimensional word vectors from fastText [58, 59].

The evaluation was conducted on two commonly used datasets [60, 61]. 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. The second dataset is multilingual SimLex999 which consists of 999 English word pairs and their translations.

**Algorithms.** We compared MetaProx with DCCA and DCCAE, and all of them used multilayer perceptrons with ReLU activation. MetaProx used the CCA value as the ultimate objective. 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 $\lambda$, and the depth and layer width from 1 to 4 and $\{256, 512, 1024, 2048\}$, respectively. We also compared with linear CCA [62]. At test time, the (English, German) word pairs from the test set were fed to the four multivew 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 1 the Spearman's correlation between the model's ranking and human's ranking.

**Results.** Clearly, MetaProx 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 [63], and the paper only provided the results for SimLex999 while no code was made available. It can be observed from Table 1 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, it is not surprising that more improvement was achieved on German.

**Extensions and conclusion.** In Appendix F, one more instantiation of MetaProx will be demonstrated for adversarial recurrent learning. Appendix B further shows dropout can also be cast as proximal map. To summarize, motivated by meta-learning with unsupervised base learners, we proposed using proximal mapping as a new primitive in deep learning, which explicitly encodes desired structure. The new models for multiview and adversarial recurrent learning are shown effective.

## References


[1] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *In International Conference on Learning Representations (ICLR)*. 2017.

[2] D. Arpitz, S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Lacoste-Julien. A closer look at memorization in deep networks. *In International Conference on Machine Learning (ICML)*. 2017.

[3] P. Y. Simard, Y. A. LeCun, J. S. Denker, and B. Victorri. Transformation invariance in pattern recognition – tangent distance and tangent propagation. *In* G. Montavon, G. B. Orr, and K.-R. Müller, eds., *Neural Networks: Tricks of the Trade: Second Edition*, pp. 235–269. 2012.

[4] D. K. Pal, A. A. Kannan, G. Arakalgud, and M. Savvides. Max-margin invariant features from transformed unlabeled data. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[5] A. Makhzani and B. Frey. $k$-Sparse autoencoders. *In International Conference on Learning Representations (ICLR)*. 2014.

[6] K. Roth, A. Lucchi, S. Nowozin, and T. Hofmann. Adversarially robust training through structured gradient regularization, 2018. ArXiv:1805.08736.

[7] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *In International Conference on Learning Representations (ICLR)*. 2017.

[8] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[9] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. *In International Conference on Machine Learning (ICML)*. 2011.

[10] W. Wang, R. Arora, K. Livescu, and J. A. Bilmes. On deep multi-view representation learning. *In International Conference on Machine Learning (ICML)*. 2015.

[11] G. Andrew, R. Arora, J. A. Bilmes, and K. Livescu. Deep canonical correlation analysis. *In International Conference on Machine Learning (ICML)*. 2013.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *In Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1097–1105. 2012.

[13] 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.

[14] 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.

[15] T. Miyato, S. ichi Maeda, M. Koyama, K. Nakae, and S. Ishii. Distributional smoothing with virtual adversarial training. *In International Conference on Learning Representations (ICLR)*. 2016.

[16] T. Miyato, A. M. Dai, and I. Goodfellow. Adversarial training methods for semi-supervised text classification. *In International Conference on Learning Representations (ICLR)*. 2017.

[17] 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.

[18] 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.

[19] J. Sietsma and R. J. F. Dow. Creating artificial neural networks that generalize. *Neural Networks*, 4(1):67–79, 1991.

[20] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. *In International Conference on Machine Learning (ICML)*. 2008.

[21] D. Stutz, M. Hein, and B. Schiele. Disentangling adversarial robustness and generalization. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[22] T. N. Kipf and M. Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[23] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

[24] W. Hare and C. Sagastizábal. Computing proximal points of nonconvex functions. *Mathematical Programming*, 116(1):221–258, 2009.

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

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

[27] G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. *In International Conference on Machine Learning (ICML)*. 2015.

[28] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[29] J. Snell, K. Swersky, and R. S. Zemel. Prototypical networks for few-shot learning. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[30] C. Finn and S. Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. 2018.

[31] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *In International Conference on Machine Learning (ICML)*. 2017.

[32] L. Bertinetto, J. F. Henriques, P. H. Torr, and A. Vedaldi. Meta-learning with differentiable closed-form solvers. *In International Conference on Learning Representations (ICLR)*. 2019.

[33] K. Lee, S. Maji, A. Ravichandran, and S. Soatto. Meta-learning with differentiable convex optimization. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[34] S. Hochreiter, A. S. Younger, and P. R. Conwell. Learning to learn using gradient descent. *In International Conference on Artificial Neural Networks*. 2001.

[35] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. de Freitas. Learning to learn by gradient descent by gradient descent. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[36] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. 2017.

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

[38] B. Amos and J. Z. Kolter. OptNet: Differentiable optimization as a layer in neural networks. *In International Conference on Machine Learning (ICML)*. 2017.

[39] J. Domke. Implicit differentiation by perturbation. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2010.

[40] J. Domke. Generic methods for optimization-based modeling. *In International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2012.

[41] D. Belanger, B. Yang, and A. McCallum. End-to-end learning for structured prediction energy networks. *In International Conference on Machine Learning (ICML)*. 2017.

[42] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: A survey. *Journal of Machine Learning Research*, 18(1):5595–5637, 2017.

[43] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. Unrolled generative adversarial networks. *In International Conference on Learning Representations (ICLR)*. 2017.

[44] P. Brakel, D. Stroobandt, and B. Schrauwen. Training energy-based models for time-series imputation. *Journal of Machine Learning Research*, 14:2771–2797, 2013.

[45] 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.

[46] V. Stoyanov, A. Ropson, and J. 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.

[47] I. Goodfellow, M. Mirza, A. Courville, and Y. Bengio. Multi-prediction deep boltzmann machines. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2013.

[48] S. Gould, B. Fernando, A. Cherian, P. Anderson, R. S. Cruz, and E. Guo. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv:1607.05447*, 2016.

[49] P. Laforgue, S. Clémençon, and F. d'Alché-Buc. Autoencoding any data through kernel autoencoders. *In International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2019.

[50] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional kernel networks. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2014.

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

[52] A. Bietti and J. Mairal. Group Invariance, Stability to Deformations, and Complexity of Deep Convolutional Representations. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

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

[54] Y. Ma, V. Ganapaman, and X. Zhang. Learning invariant representation with kernel warping. *In International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2019.

[55] News commentary corpus. `http://www.statmt.org/wmt18`.

[56] C. Dyer, V. Chahuneau, and N. A. Smith. A simple, fast, and effective reparameterization of ibm model 2. *In HLT-NAACL*. 2013.

[57] Fast align toolbox. `https://github.com/clab/fast_align`.

[58] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov. Learning word vectors for 157 languages. *In Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*. 2018.

[59] Pretrained fasttext word vectors. `https://fasttext.cc/docs/en/crawl-vectors.html`.

[60] I. Leviant and R. Reichart. Separated by an un-common language: Towards judgment language informed vector space modeling. *arXiv:1508.00106*, 2015.

[61] Multilingual simlex999 and wordsim353 datasets. `http://leviants.com/ira.leviant/MultilingualVSMdata.html`.

[62] M. Faruqui and C. Dyer. Improving vector space word representations using multilingual correlation. *In EACL*. 2014.

[63] I. Vulić. Cross-lingual syntactically informed distributed word representations. *In European Chapter of the Association for Computational Linguistics*. 2017.

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

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

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

[67] 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.

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

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

[70] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012.

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

[72] J. A. Bagnell and D. M. Bradley. Differentiable sparse coding. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2008.

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

[74] S. Wang, S. Fidler, and R. Urtasun. Proximal deep structured models. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2016.

[75] J. T. Zhou, K. Di, J. Du, X. Peng, H. Yang, S. J. Pan, I. W.-H. Tsang, Y. Liu, Z. Qin, and R. S. M. Goh. Sc2net: Sparse lstms for sparse coding. *In National Conference of Artificial Intelligence (AAAI)*. 2018.

[76] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *In European Conference on Computer Vision (ECCV)*, pp. 630–645. 2016.

[77] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[78] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng. Dual path networks. *In Advances in Neural Information Processing Systems (NeurIPS)*. 2017.

[79] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang. Random erasing data augmentation. *arXiv:1708.04896*, 2017.

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

[81] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[82] A. Lu, W. Wang, M. Bansal, K. Gimpel, and K. 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.

[83] M. Shimbo, M. Kudo, and J. Toyama. Multidimensional curve classification using passing-through regions. *Pattern Recognition Letters*, 20:1103, 1999.

[84] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. *In European Symposium on Artificial Neural Networks*. 2013.

[85] N. Hammami and M. Bedda. Improved tree model for arabic speech recognition. *In International Conference on Computer Science and Information Technology*. 2010.

# A Generalizing Multi-layer Invariant Kernel Warping by Proximal Mapping

We will review in the next two sections two existing techniques in deep learning, revealing new insights by connecting them to proximal mapping. [50, 51, 64] introduced multi-layer transformations of convolutional kernel descriptors, and [54] proposed warping kernels to incorporate a broad variety of data-dependent invariances that supersede transformation, if the invariances can be represented by a bounded linear functional. Our first new insight is that kernel warping is indeed a proximal mapping, which, as a result, allows invariances to be significantly extended to *nonlinear* functionals.

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 [49], and some *prior* invariances to be hard wired [50–52]. It is different from the conventional motivation of using kernels for enriching the feature space, which is already facilitated by deep nets. [54] 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.

Let us 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. The RKHS induced by $k$ is denoted as $\mathcal{H}$, and more details on RKHS can be found in [65, 66]. 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 [53]: given a convex functional $L : \mathcal{H} \to \mathbb{R}$, the proximal map $\mathsf{P}_L : \mathcal{H} \to \mathcal{H}$ is defined as in (2), which we copy here for convenience:

$$\mathcal{H} \ni h \mapsto \mathsf{P}_L(h) := \arg\min_{f\in C} L(f) + \tfrac{\lambda}{2}\|f - h\|^2_{\mathcal{H}}, \quad \text{where} \quad C \subseteq \mathcal{H} \text{ is convex.} \tag{5}$$

**Finite approximation (FA).** Although this optimization can often be solved when favorable structures are available (e.g., representer theorem), Backpropagation through it can still be hard. Therefore we resort to Nyström approximations of functions in $\mathcal{H}$ [67]. 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 $f, h \in \mathcal{H}$:

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

Therefore, 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 an FA of $C$, e.g., $||\tilde{f}|| \leq 1$.

**Kernel warping for invariance modeling.** 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 bounded linear functional. Parameterizing images as $I(\alpha)$ where $\alpha$ is the degree of rotation, tran slation, etc, transformation invariance favors a small absolute value of $\frac{\partial}{\partial\alpha}|_{\alpha=0} f(I(\alpha))$, which is a bounded linear functional. Local averaging compares $f(x_i)$ with the neighborhood: $\int f(\tau)\rho(x_i - \tau)\,\mathrm{d}\tau$, where $\rho$ is a distribution peaked at zero. The difference is also a bounded linear functional, which, by Riesz representation theorem, 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, [54] proposed a warped RKHS $\mathcal{H}^\circ$ consisting of the same functions in the original RKHS $\mathcal{H}$, but redefining the norm and the corresponding kernel by

$$\|f\|^2_{\mathcal{H}^\circ} := \|f\|^2_{\mathcal{H}} + \sum\nolimits_{i=1}^m \langle z_i, f\rangle^2_{\mathcal{H}} \quad \Leftrightarrow \quad k^\circ(x_1, x_2) = k(x_1, x_2) - z(x_1)^\top K_Z z(x_2), \tag{6}$$

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 of the network, e.g., deep convolutional kernel network [CKNs, 50, 64], instilling invariance 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. In addition, the change of RKHS creates conceptional and practical complications, and it will be much more convenient if we retain $\mathcal{H}$ and just change $k(x, \cdot)$.

**Kernel warping as MetaProx.** Interestingly, both of these desirabilities can be achieved by simply reformulating kernel warping into a proximal mapping. Since $\mathcal{H}$ and $\mathcal{H}^\circ$ share the same function

space but endow different meanings, we will avoid ambiguity by considering the FA of $k^\circ(x,\cdot)$ in $\mathcal{H}^\circ$ [54], which can be written as:

$$(I+\tilde{Z}\tilde{Z}^\top)^{-1/2}\tilde{\varphi}(x), \quad \text{where} \quad \tilde{Z}=(\tilde{z}_1,\ldots,\tilde{z}_m), \text{ and } \tilde{\varphi}(x) \text{ is the FA of } k(x,\cdot) \text{ in } \mathcal{H}. \tag{7}$$

Simply setting $L(f) := \frac{1}{2}\sum_{i=1}^m \langle z_i, f\rangle_{\mathcal{H}}^2$ in the proximal map (2) to measure invariance, we derive

$$\mathsf{P}_L(k(x,\cdot)) = \arg\min_{f\in\mathcal{H}} \ \tfrac{1}{2}\sum\nolimits_{i=1}^m \langle z_i, f\rangle_{\mathcal{H}}^2 + \tfrac{1}{2}\left\|f-k(x,\cdot)\right\|_{\mathcal{H}}^2. \tag{8}$$

Using FA $\tilde{z}_i$ for $z_i$ and $\tilde{\varphi}(x)$ for $k(x,\cdot)$, we obtain the FA of $\mathsf{P}_L(\varphi(x))$ as $(I+\tilde{Z}\tilde{Z}^\top)^{-1}\tilde{\varphi}(x)$, which is almost the same as that from kernel warping in (7), 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 gradient flatness invariance is shown in Figure 1. Trivially, now the regularizer $L$ in (8) can be extended to nonlinear (convex) functionals, such as total variation $f\mapsto\int|f'(x)|\,\mathrm{d}x$. In hindsight, (8) is very natural: shift $k(x,\cdot)$ into a new representation $f\in\mathcal{H}$ with small deviation while better respecting the invariances. It broadens the invariances encodable in multi-layer CKNs [54].

## B Dropout Training as a Proximal Mapping

Another interesting but unheeded instance of proximal map is the well-known dropout. Although connection with adaptive regularization has been unraveled by [68, 69] for a single layer, extensions to multiple layers have been left unclear. We will achieve this by proximal mapping.

The underlying rationale of dropout is to introduce a blackout noise $\epsilon$ to a hidden layer $x\in\mathbb{R}^h$ [70], so that the subsequent layer's output $f$ is robust to it, as quantified by a metric $D$. For better generality, we let $f$ lie in an RKHS as in the previous section. Denote the noised $x$ as $x_\epsilon$, which can be i) $x+\epsilon$ for additive Gaussian noise with $\epsilon_i\sim\mathcal{N}(\mu,\sigma^2)$, and ii) $x\odot\epsilon$ for dropout noise, where $\odot$ stands for elementwise multiplication, and $\epsilon_i=0$ with probability $\delta$, and $\epsilon_i=(1-\delta)^{-1}$ otherwise.

In [68], a target variable $y$ is endowed with a conditional distribution $p(y|x)=\exp(yf(x)-A(f(x)))$, and the metric $D$ is defined using the sensitivity of $p(y|x)$ with respect to $x$. Here $A$ is the log-partition function, which equals $\log(e^z+e^{-z})$ for logistic regression. Formally,

$$D(f,x) = \mathbb{E}_\epsilon[\log p(y|x) - \log p(y|x_\epsilon)] = \mathbb{E}_\epsilon[A(f(x_\epsilon))] - A(f(x)). \tag{9}$$

Taking expectation over the empirical distribution $\tilde{p}$ yields $\tilde{D}(f) := \mathbb{E}_{x\sim\tilde{p}}D(f,x)$. By specializing to a linear function $f_\beta: x\mapsto\beta^\top x$ and applying a Taylor expansion on $A$ about $f(x)$, [68] showed that when $y|x$ is logistic, $\tilde{D}(f_\beta)$ is proportional to the following terms with $p_x := (1+\exp(-f(x)))^{-1}$

$$\left\|\beta\right\|^2\mathbb{E}_{x\sim\tilde{p}}[p_x(1-p_x)] \text{ (additive noise)}, \quad \text{and} \quad \sum\nolimits_j a_j\beta_j^2 \text{ (dropout noise)}. \tag{10}$$

Here $a_j=\mathbb{E}_{x\sim\tilde{p}}[p_x(1-p_x)x_j^2]$. So the key advantage of dropout lies in permitting $\beta_j$ to take a larger magnitude if $x_j$ is generally small (so is $a_j$), hence favoring rare but discriminative features that are common in text data. It also encourages more committed $p_x$, with a lower value of $p_x(1-p_x)$.

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

$$\mathsf{P}_L(k(x,\cdot)) = \arg\min_{f\in\mathcal{H}} \ \tfrac{1}{2}\tilde{D}(f) + \tfrac{\lambda}{2}\left\|f-k(x,\cdot)\right\|_{\mathcal{H}}^2. \tag{11}$$

If we hypothetically suppress the dependency of $p_x$ in (10) on $\beta$, $\mathsf{P}_L(k(x,\cdot))$ admits a closed form under dropout noise and linear kernel with $f(z)=f^\top z$: $\mathsf{P}_L(x)=b\odot x$ where $b_j+\lambda(\lambda+a_j)^{-1}$. Letting the output layer be $x\mapsto\alpha^\top x$, the regularized risk for loss $\ell$ can be written as

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

Thus, as long as $\lambda$ is small and $c$ is set proportionally to $\lambda^2$, the regularizer on $\beta$ almost recovers $\sum_j a_j\beta_j^2$ in (10), 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 (11). Fortunately, our simulations show that this does not result in a significant difference.

The proximal map in (11) can be readily extended to nonlinear kernels, calling for Nyström approximation of $\tilde{D}(f)$. The robustness measure $\tilde{D}$ and $D$ may enjoy even more flexibility. The current assignment in (9) 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 [71]. To summarize Sections A and B, proximal mapping provides a compact framework that allows us to efficiently model the invariance to perturbations in both input and hidden layer output. We will next leverage this tool to develop two novel algorithms.

# C 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, [41] used it to train a structured prediction energy network, and [44] used it to train an energy-based model for time-series imputation. Other works include [46, 47], etc. A number of implicit and auto-differentiation algorithms have been proposed for it, e.g., [38–40, 42, 48].

Other uses of such differentiable optimization have been found in meta-learning to differentiate through the learning algorithm in the middle [32, 33], or to train the generator in a generative adversarial model by optimizing out the discriminator [43], or for end-to-end planning and control [45]. 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 [38] 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, 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 [72, 73]. 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 F).

This is achieved by MetaProx. Although MetaProx 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 MetaProx 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.

We note that despite the similarity in titles, [74] 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 [75]. We instead use proximal map as a first-class construct/layer of meta-learning in a deep network.

# 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

$$\frac{\partial J}{\partial W} = \sum_{t=1}^{T} \left\{ \underbrace{\frac{\partial J}{\partial h_t}}_{\text{by (13)}} \underbrace{\frac{\partial}{\partial W} h_t(c_{t-1}, h_{t-1}, x_t)}_{\text{standard LSTM}} + \underbrace{\frac{\partial J}{\partial c_t}}_{\text{by (15)}} \underbrace{\frac{\partial}{\partial W} c_t(c_{t-1}, h_{t-1}, x_t)}_{\text{standard LSTM}} \right\}, \tag{12}$$

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,

$$\frac{\partial J}{\partial h_{t-1}} = \underbrace{\frac{\partial J}{\partial h_t}}_{\text{by recursion}} \underbrace{\frac{\partial h_t}{\partial h_{t-1}}}_{\text{std LSTM}} + \underbrace{\frac{\partial J}{\partial G_t} \frac{\partial G_t}{\partial h_{t-1}}}_{\text{by (14)}} + \underbrace{\frac{\partial J}{\partial s_t}}_{\text{by (20)}} \underbrace{\frac{\partial s_t}{\partial h_{t-1}}}_{\text{std LSTM}}. \tag{13}$$

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 (29):

$$\frac{\partial J}{\partial G_t} \frac{\partial G_t}{\partial h_{t-1}} = \frac{\partial J}{\partial G_t} \frac{\partial^2}{\partial x_t \partial h_{t-1}} s_t(c_{t-1}, h_{t-1}, x_t) = \frac{\partial}{\partial h_{t-1}} \left\langle \underbrace{\frac{\partial J}{\partial G_t}}_{\text{by (29)}}, \frac{\partial}{\partial x_t} s_t(c_{t-1}, h_{t-1}, x_t) \right\rangle. \tag{14}$$

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

$$\frac{\partial J}{\partial c_{t-1}} = \underbrace{\frac{\partial J}{\partial h_t}}_{\text{by recursion}} \underbrace{\frac{\partial h_t}{\partial c_{t-1}}}_{\text{std LSTM}} + \underbrace{\frac{\partial J}{\partial G_t} \frac{\partial G_t}{\partial c_{t-1}}}_{\text{by (16)}} + \underbrace{\frac{\partial J}{\partial s_t}}_{\text{by (20)}} \underbrace{\frac{\partial s_t}{\partial c_{t-1}}}_{\text{std LSTM}}. \tag{15}$$

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:

$$\frac{\partial J}{\partial G_t} \frac{\partial G_t}{\partial c_{t-1}} = \frac{\partial J}{\partial G_t} \frac{\partial^2}{\partial x_t \partial c_{t-1}} s_t(c_{t-1}, h_{t-1}, x_t) = \frac{\partial}{\partial c_{t-1}} \left\langle \underbrace{\frac{\partial J}{\partial G_t}}_{\text{by (29)}}, \frac{\partial}{\partial x_t} s_t(c_{t-1}, h_{t-1}, x_t) \right\rangle. \tag{16}$$

### 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

$$J = f(c), \quad \text{where} \quad c := c(G, s) := (I + GG^\top)^{-1} s. \tag{17}$$

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

$$\Delta J := f(c(G, s + \Delta s)) - f(c(G, s)) = \nabla f(c)^\top (c(G, s + \Delta s) - c(G, s)) + o(\|\Delta s\|) \tag{18}$$

$$= \nabla f(c)^\top (I + GG^\top)^{-1} \Delta s + o(\|\Delta s\|). \tag{19}$$

Therefore,

$$\frac{\partial J}{\partial s} = \nabla f(c)^\top (I + GG^\top)^{-1}. \tag{20}$$

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

$$\Delta J := f(c(G + \Delta G, s)) - f(c(G, s)) = \nabla f(c)^\top (c(G + \Delta G, s) - c(G, s)) + o(\|\Delta G\|). \tag{21}$$

Since

$$c(G+\Delta G,s)=(I+(G+\Delta G)(G+\Delta G)^\top)^{-1}s \tag{22}$$

$$=\left[(I+GG^\top)^{\frac{1}{2}}\left(I+(I+GG^\top)^{-\frac{1}{2}}(\Delta GG^\top+G\Delta G^\top)(I+GG^\top)^{-\frac{1}{2}}\right)(I+GG^\top)^{\frac{1}{2}}\right]^{-1}s \tag{23}$$

$$=(I+GG^\top)^{-\frac{1}{2}}\left(I-(I+GG^\top)^{-\frac{1}{2}}(\Delta GG^\top+G\Delta G^\top)(I+GG^\top)^{-\frac{1}{2}}+o(\|\Delta G\|)\right)(I+GG^\top)^{-\frac{1}{2}}s \tag{24}$$

$$=c(G,s)-(I+GG^\top)^{-1}(\Delta GG^\top+G\Delta G^\top)(I+GG^\top)^{-1}s+o(\|\Delta G\|), \tag{25}$$

we can finally obtain

$$\Delta J=-\nabla f(c)^\top(I+GG^\top)^{-1}(\Delta GG^\top+G\Delta G^\top)(I+GG^\top)^{-1}s+o(\|\Delta G\|) \tag{26}$$

$$=-\operatorname{tr}\left(\Delta G^\top(I+GG^\top)^{-1}\left(\nabla f(c)s^\top+s\nabla f(c)^\top\right)(I+GG^\top)^{-1}G\right)+o(\|\Delta G\|). \tag{27}$$

So in conclusion,

$$\frac{\partial J}{\partial G}=-(I+GG^\top)^{-1}\left(\nabla f(c)s^\top+s\nabla f(c)^\top\right)(I+GG^\top)^{-1}G \tag{28}$$

$$=-(ac^\top+ca^\top)G,\quad \text{where}\quad a=(I+GG^\top)^{-1}\nabla f(c). \tag{29}$$

# E Detailed and Self-contained Experimental Result

## E.1 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 55] using the word alignment method from [56, 57]. 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 [10, 62], where word embeddings are trained via Latent Semantic Analysis (LSA) using parallel corpora, we used the pretrained monolingual 300-dimensional word embedding from fastText [58, 59] as the raw word embeddings ($x_i$ and $y_i$).

To evaluate the quality of learned word representation, we experimented on two different benchmarks [60, 61] that have been widely used to measure word similarity. 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 MetaProx 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 [62], 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 [82], which, instead of learning linear transformations with CCA, learned nonlinear transformations of two languages' embedding that are highly correlated.
- DCCAE [10], 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.

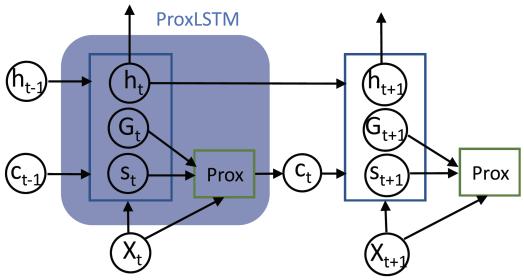



Figure 3: A proximal LSTM layer

- CL-DEPEMB [63], 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 $\ell_2$ 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 rather than 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. Note line 279 mentioned that the ultimate optimization objective is the CCA value. This was indeed used in a previous versions of the experiment, and later on we discovered that using the input reconstruction error as the ultimate objective is more effective, and the results reported in the paper are all based on it. We will update line 279 in later revisions.

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 $\lambda$ 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.

## F MetaProx for Adversarial Learning in Recurrent Neural Networks

Our second novel instance of MetaProx tries to invariantize LSTM to perturbations on inputs $x_t$ at each step. Adversarial training has been proposed in this context [16], 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 $h_t = g(c_{t-1}, h_{t-1}, x_t)$. Our goal is to encourage that the hidden state $c_t$ stays invariant, when each $x_t$ is perturbed by $\delta_t$ whose norm is bounded by $\delta$. To this end, we introduce an intermediate step $s_t = s_t(c_{t-1}, h_{t-1}, x_t)$ that computes the original LSTM hidden state, and then applies proximal mapping so that the next state $c_t$ remains in the proximity of $s_t$, while also favoring the *null space* of the variation of $s_t$ under the perturbations on $x_t$. Formally,

Table 2: Summary of datasets for adversarial LSTM training

| Dataset | Training | Validation | Test | Length | Attributes | Classes |
|---|---|---|---|---|---|---|
| JV | 225 | 45 | 370 | 7-26 | 12 | 9 |
| HAR | 6127 | 1225 | 2974 | 128 | 9 | 6 |
| AD | 5500 | 1100 | 2200 | 4-93 | 13 | 10 |

$$
\begin{aligned}
c_t &:= \arg\min_c \tfrac{\lambda}{2}\left\|c - s_t\right\|^2 + \tfrac{1}{2}\max_{\delta_t:\|\delta_t\|\le\delta} \langle c, s_t(c_{t-1}, h_{t-1}, x_t) - s_t(c_{t-1}, h_{t-1}, x_t + \delta_t)\rangle^2 \\
&\approx \arg\min_c \frac{\lambda}{2}\left\|c - s_t\right\|^2 + \frac{1}{2}\max_{\delta_t:\|\delta_t\|\le\delta} \left\langle c, \frac{\partial}{\partial x_t} s_t(c_{t-1}, h_{t-1}, x_t)\delta_t \right\rangle^2 \qquad (30) \\
&= \arg\min_c \tfrac{\lambda}{2}\left\|c - s_t\right\|^2 + \tfrac{\delta^2}{2}\left\|c^\top G_t\right\|_*^2, \quad \text{where} \quad G_t := \tfrac{\partial}{\partial x_t} s_t(c_{t-1}, h_{t-1}, x_t), \qquad (31)
\end{aligned}
$$

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.

**Datasets.** To demonstrate the effectiveness of using proximal mapping, we tested on 3 different sequence datasets. Janpanese Vowels (JV) dataset [83]contains time series data where 9 male speakers uttered Japanese Vowels successively, the task is to classify speakers. Human Activity Recognition (HAR) dataset [84] is used to classify what a person is doing (sitting, walking, etc.) based on a trace of their movement using sensors. Arabic Digits dataset (AD) [85] contains times series corresponding to spoken Arabic digits by native speakers, the task is to classify digits. Details of each dataset were summarized in Table 2.

**Algorithms.** We compared ProxLSTM with two baselines: vanilla LSTM and the adversarial training of LSTM [16], which we will refer to as AdvLSTM. All these 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. All settings were evaluated 10 times to report mean and standard deviation.

**Preprocessing.** Normalization was the only preprocessing applied on all datasets. For those datasets who contain various length sequences, sequences were zero padded to the same size of the longest sequence in batch. To reduce the effect of padding, we first sorted all sequences by length, so that sequences with similar length were assigned to the same batch.

**Results.** From Table 3, it is clear that adversarial training improves the test accuracy, and ProxLSTM promotes the performance even more than AdvLSTM. Since the accuracy gap is lowest on the HAR set, we also plotted the t-SNE embedding of the features from the last time step for the HAR set. As Figure 4 shows (best viewed in color), the representations learned from ProxLSTM (left) appear better clustered than that of AdvLSTM (right), especially the yellow class. This further indicates that ProxLSTM can learn better latent representations than AdvLSTM by applying proximal mapping.

**Baseline models:** To show the impact of applying proximal mapping on LSTM, we compared our method with 2 baselines. The basic model structure is composed of 1 CNN layer followed by a LSTM layer, then a fully connected layer is applied. CNN layer is constructed with kernel size 3,8,3 and contains 32, 64, 64 filters for JV, HAR, AD dataset respectively. For LSTM layer, the number of hidden units used in these three datasets are 64, 128, 64 respectively. This basic structure is denoted as LSTM in Table 3. On top of this basic structure, we compareed 2 different adversarial training methods. AdvLSTM is the adversarial training method in [16], we reimplemented their method in Pytorch. Following their work, perturbation is added to the input of LSTM layer. ProxLSTM denote our method described in section F, here we replace LSTM cell in basic structure to ProxLSTM cell.

| | JV | HAR | AD |
|---|---|---|---|
| LSTM | 94.02 ±0.72 | 89.75 ±0.59 | 96.32 ±0.55 |
| AdvLSTM | 94.96 ±0.44 | 92.01 ±0.18 | 97.45 ±0.38 |
| ProxLSTM | **95.52** ±0.63 | **92.08** ±0.23 | **97.99** ±0.29 |

Table 3: Test accuracy for sequence classification

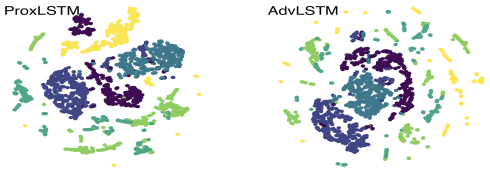



Figure 4: t-SNE embedding of the HAR dataset

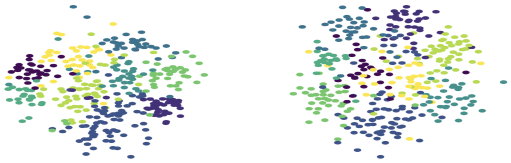


Figure 5: t-SNE embedding of the JV dataset

Figure 6: t-SNE embedding of the AD dataset

**Training:** While training, we first train baseline LSTM to convergence, then apply AdvLSTM and ProxLSTM as fine tunning. We use Adam optimizer with learning rate $10^{-3}$, weight decay $10^{-4}$. All settings were evaluated 10 times to report mean and standard deviation. The results were all summmarized in Table 3. Figure 4 shows the t-SNE embedding of extracted features from the last time step's hidden state of HAR test set. ProxLSTM on the left, AdvLSTM on the right.

**Results:** According to table 3, it's clear that applying adversarial training improves the performance. ProxLSTM even promotes the performance more than AdvLSTM. Though this gap is subtle on some datasets, Figure 4 ilustrates embedded features from ProxLSTM gather more compact than that of AdvLSTM (e.g. yellow class). t-SNE plot of other datasets are available in Figures 5 and 6. This further indicates that ProxLSTM can learn better latent representation than AdvLSTM by applying proximal mapping.