Warping Layer: Representation Learning for Label Structures in Weakly Supervised Learning

Yingyi Ma **Xinhua Zhang**
Department of Computer Science, University of Illinois at Chicago, Chicago IL 60607

Abstract

Many learning tasks only receive weak supervision, such as semi-supervised learning and few-shot learning. With limited labeled data, prior structures become especially important, and prominent examples include hierarchies and mutual exclusions in the class space. However, most existing approaches only learn the representations *separately* in the feature space and the label space, and do not explicitly enforce the logical relationships. In this paper, we propose a novel warping layer that *jointly* learns representations in *both* spaces, and thanks to the modularity and differentiability, it can be directly embedded into generative models to leverage the prior hierarchical structure and unlabeled data. The effectiveness of the warping layer is demonstrated on both few-shot and semi-supervised learning, outperforming the state of the art in practice.

1 INTRODUCTION

Weakly supervised learning represents a very common scenario in practice, where labeled examples are not available in a large supply. For example, in few-shot learning (Finn et al., 2017; Ravi & Larochelle, 2017; Snell et al., 2017; Vinyals et al., 2016), classifiers need to adapt rapidly to new classes when only a few examples are demonstrated. In semi-supervised learning (Kingma et al., 2014; Narayanaswamy et al., 2017), a large amount of unlabeled data is available but few are labeled.

The low-data regime makes it crucial to leverage prior inductive bias. In the context of multiple classes, an important, but under-studied, prior is the logical relationship between labels. For example, if an image shows a *cat*, then it must also be relevant to the category of *animal*. Whenever such implication relationships form a tree in the label space, we call it a hierarchical structure. But such relationships can be more general, such as exclusions where no two labels can co-exist (e.g., an animal cannot be a *dog* and a *cat* at the same time), or a co-occurrence relationship which is effectively a bidirectional mutual implication. The most primitive models leverage such structures by re-engineering the loss function to penalize the first mistake along a path from root to a node in a tree hierarchy (Cesa-Bianchi et al., 2006; Cai & Hofmann, 2004). Inspired by it, Rousu et al. (2006) designed a kernel on the input-output space jointly to factor in the hierarchy.

Unfortunately, these methods neither learn feature representations for each data instance that account for the label structure, nor learn the representation for each class. These are the central objects underlying many low-data applications. For example, few-shot learning are commonly based on distance metrics between data embeddings (Vinyals et al., 2016; Snell et al., 2017), and such distances (or equivalently, embeddings) are the key entities that demand awareness of label structure. Zero-shot learning relies intrinsically on class prototypes (Lampert et al., 2009; Xian et al., 2016; Xie et al., 2019; Yu et al., 2018). Similarly, semi-supervised learning often capitalizes on class-conditioned generative models to tap into unlabeled data. As a result, leveraging logical structures in low-data regime calls for learning *structure-aware* representations.

However, such awareness still lacks a concrete definition. Recent attempts include Chen et al. (2019); Zhou et al. (2020) which build a graph where each node corresponds to a class, and a pair of nodes are connected by a directed edge if an implication relationship is present. Then the node representation, i.e., the class representation, is computed through a graph convolutional network (GCN), whose parameters are learned

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151. Copyright 2022 by the author(s).

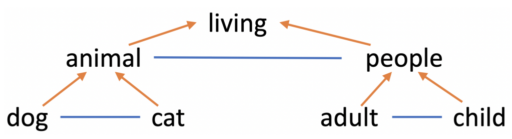

Figure 1: implication ($\rightarrow$) and exclusion ($-$) relations between classes, derived from a taxonomy hierarchy

by back-propagation. Despite the improvement in performance, it is unclear how the label space relations is upheld by the GCN, because such logic is used only to define its edges. Without any constraint on the graph convolution parameters (e.g., positivity), there is no mechanism that explicitly aligns the output class-wise logits with the logical relationships.

To address this issue, Mirzazadeh et al. (2015) recently enforced the logics exhaustively on the training data. Using $f(x, y)$ to quantify the compatibility between input x and label y, $f(x, y_1)$ is constrained to be no greater than $f(x, y_2)$ for all x of interest if class y_1 implies y_2 (y_2 subsumes y_1). Such constraints lead to semi-definite programs which are expensive, and they still do not produce class representations. Most relevant to our work is Ma et al. (2020), which penalized the violation of the logics and used it to warp the norm of a reproducing kernel Hilbert space (RKHS). This results in a semi-inner-product (s.i.p.) space, which preserves the key element in an RKHS—a reproducing kernel representer for each (example, label) pair. Training a vanilla linear classifier based on it, the discriminant value turns out to respect the logical relations even without requiring additional constraints, confirming that such a warped joint representation does capture the label structure.

In practice, however, such a joint feature map is very expensive in computation, because a constrained nonlinear optimization problem needs to be solved for *each pair* of example and label. It also does not provide a separate representations for output classes and for input features. Our first contribution, therefore, is to *factorize* the joint representation into features specific to each example and features specific to each class. As a result, the optimization is accelerated by orders of magnitude, and feature representations on both sides can be learned jointly via efficient implicit differentiation. Remarkably, it preserves the logic relations explicitly in the label space. We call this operation a *warping layer*, which can be flexibly embedded into a general neural network to incorporate structures.

As an important example, our second contribution integrates the structure-aware class representation into generative modeling, facilitating semi-supervised learning that utilizes unlabeled data. Our key observation is that the class representation can naturally serve as the basis (first layer) of the class-specific part of the decoder in a variational auto-encoder (VAE). In general, partially tying the encoder and decoder requires a grain of salt, because features useful for discrimination might not necessarily benefit reconstruction. However, in the low-data regime, such an inductive bias can be helpful, and interestingly, our ablation study reveals significant boost in performance.

In Section 2, we first recap the kernel warping technique, and extend it to factored structure-aware representation learning which can be encapsulated into a generic warping layer. Section 3 and 4 then apply it to few-shot learning and semi-supervised VAE, resp. Experimental results showing superior performance over the state of the art are presented in Section 5.

2 REPRESENTATION LEARNING WITH LABEL STRUCTURE BY KERNEL WARPING

Assume we are given a full taxonomy like WordNet (Miller, 1995). In practice, it is often easy to infer some useful logics between classes as a priori. Following Mirzazadeh et al. (2015); Deng et al. (2012), the taxonomy can be represented by a graph $\mathcal{G} = (Y, E_i, E_e)$ as illustrated in Figure 1. Here $Y = (y_1, ..., y_m)$ is a set of nodes representing all the classes (leaf and non-leaf). E_i is a set of directed edges between classes indicating **implication**. For example, if an object is a dog, then it must be an animal, and we denote it as "dog $\rightarrow$ animal". Similarly, if an object is a dog, then it cannot be a cat at the same time. We denote such **exclusion** relations as "dog $\leftrightsquigarrow$ cat", represented by undirected edges that collectively form the set E_e.

To incorporate structured label priors, a natural model resorts to joint prediction functions $f(x, y)$, which characterizes the likelihood for each example x to be relevant to an individual class y. As shown in the sequel, it also allows representations to be learned for both input and output. For implication relations $y_1 \rightarrow y_2$, y_2 is a more general class and should achieve higher likelihood than y_1. So Ma et al. (2020) proposed enforcing $f(x, y_2) \geq f(x, y_1)$ for all x by penalizing

$$[f(x, y_1) - f(x, y_2)]_+, \quad \text{where } [z]_+ = \max\{0, z\}. \quad (1)$$

To model exclusion relations where two classes should not receive high likelihood at the same time, one can impose $f(x, y_1) + f(x, y_2) \leq 0$. Intuitively, a high likelihood attached to one class will demote the likelihood of the other one. We can implement this constraint by penalizing

$$[f(x, y_1) + f(x, y_2)]_+. \quad (2)$$

Since there is often a large number of implication and exclusion relations in each taxonomy graph, we consider their maximum violation instead of their sum, amounting to a regularizer R that accounts for the structured label prior:

$$R(f)^2 := \mathop{\mathbb{E}}_{x\sim\tilde{p}}\Big[\max_{y_1\to y_2\in E_i}[f(x,y_1)-f(x,y_2)]_+^2 \quad (3)$$

$$+\max_{y_1\leftrightsquigarrow y_2\in E_e}[f(x,y_1)+f(x,y_2)]_+^2\Big]. \quad (4)$$

Here, $\tilde{p}$ is the empirical distribution of input x.

2.1 From kernel warping to warping layer

Although regularization such as data augmentation is commonly used to incorporate priors, recent works have shown that another effective approach is to encode them directly in representation (Ma et al., 2019, 2020). Kernel warping is a typical method of such, which also provides significant convenience in representation learning. It warps an RKHS $\mathcal{H}$ with desired priors to construct a new semi-inner-product (s.i.p.) space when R is a semi-norm. In particular, it does not change the set of functions in the RKHS, but endows a new norm on them as $\|f\|_{\mathcal{B}}^2 = \|f\|_{\mathcal{H}}^2 + R(f)^2$.

Although the new norm $\|f\|_{\mathcal{B}}^2$ could be used directly for regularized risk minimization, it is computationally inconvenient. To work around it, Ma et al. (2020) pointed out another solution path noting that the s.i.p. space inherits a kernel representer G_x for any input x under the semi-inner-product associated with the norm $\|f\|_{\mathcal{B}}$, and in general it differs from the original RKHS representer $k_x := k(x,\cdot)$.

Using G_x in lieu of k_x, a classifier f in $\mathcal{H}$ can be sought to minimize the empirical risk, where the prediction for x is made by the original RKHS inner product $\langle u, f\rangle_{\mathcal{H}}$. Extensions can also be easily made to joint kernels, resulting in representers $G_{x,y}$. Interestingly, it was shown that the predicted discriminant values (logits) faithfully align with the prior logic relationships, even though the search for f (under pre-computed $G_{x,y}$) is oblivious to them. This indicates that the representations $G_{x,y}$ have already incorporated the prior label structure.

To restore computational tractability from the infinite dimensionality of the s.i.p. space, Ma et al. (2020) showed that G_x can be embedded into a Euclidean space (akin to the Nyström approximation) by solving

$$\tilde{G}_x := \mathop{\mathrm{argmin}}_{v\in\mathbb{R}^d}\{\delta\|v\|^2+\tilde{R}(v)^2\}, \quad s.t.\ \langle v,\tilde{k}_x\rangle = 1, \quad (5)$$

where $\delta > 0$ is a trade-off parameter, and tilde denotes the Euclidean embedding of an object from the original RKHS. For example, $\tilde{k}_x \in \mathbb{R}^d$ is the Nyström approximation k_x (Williams & Seeger, 2001). Since $R(f)$ is a semi-norm, it can be written as the support function $\sup_{u\in C}\langle u,f\rangle_{\mathcal{H}}$ over a set $C\subseteq\mathcal{H}$. As a result, we take $\tilde{R}(v) := \sup_{u\in C}\tilde{u}^\top v$.

2.2 Kernel warping under label structure

To incorporate the prior structure defined in (3), we resort to joint kernels. Given a taxonomy graph with m classes in total, define the joint kernel $k^{xy}((x,y),(x',y')) := k^x(x,x')k^y(y,y')$, and for simplicity, let us just use a linear kernel for k^y, *i.e.*, $k^y(y,y') = e_y^\top e_{y'}$, where e_y is the one-hot vector for label y. Then the Euclidean embedding for representer $k_{x,y}$ can be written as $\tilde{k}_{x,y} = \tilde{k}_x e_y^\top \in \mathbb{R}^{d\times m}$, where $\tilde{k}_x \in \mathbb{R}^d$ is the embedding of k_x. Instantiating the general recipe (5), the warped representation with logical label relationship can be obtained by solving:

$$\tilde{G}_{x,y} := \mathrm{argmin}_{V\in\mathbb{R}^{d\times m}}\ \{\delta\|V\|_F^2+\tilde{R}^2(V)\} \quad (6)$$

$$s.t.\ \left\langle V,\tilde{k}_x e_y^\top\right\rangle = 1, \quad (7)$$

where $\|V\|_F$ is the Frobenious norm of V, and $\tilde{R}$ is the finite approximation of R from (3):

$$\tilde{R}(V)^2 := \mathop{\mathbb{E}}_{x\sim\tilde{p}}\Big[\max_{y_1\leftrightsquigarrow y_2}\Big[\left\langle V,\tilde{k}_x e_{y_1}^\top\right\rangle+\left\langle V,\tilde{k}_x e_{y_2}^\top\right\rangle\Big]_+^2$$

$$+\max_{y_1\to y_2}\Big[\left\langle V,\tilde{k}_x e_{y_1}^\top\right\rangle-\left\langle V,\tilde{k}_x e_{y_2}^\top\right\rangle\Big]_+^2\Big]. \quad (8)$$

However, this framework requires solving $\tilde{G}_{x,y}$ for each x and y, which is computationally demanding. Motivated by the fact that all examples share the same label structure relation, we propose restricting the representation to a rank one matrix:

$$\tilde{G}_{x,y} = r_x S_y^\top, \quad \text{where} \quad r_x\in\mathbb{R}^d,\ S_y\in\mathbb{R}^m. \quad (9)$$

As a result, $\tilde{G}_{x,y}$ is decomposed into the feature representation r_x that is specific to each example x, and the class representation S_y that is specific for each class y. This significantly simplifies the model and accelerates the learning process. Now the constraint (7) becomes $(r_x^\top\tilde{k}_x)S_{yy} = 1$, implying that S_{yy} (the y-th entry of S_y) is independent of both x and y. So we assemble all S_y into a class representation matrix as $S = [S_1,\ldots,S_m]\in\mathbb{R}^{m\times m}$, and let $\mathcal{S} := \{S : S_{ii} = S_{jj}, \forall i\neq j\}$. Finally, given a mini-batch $\mathcal{D}$, $\tilde{G}_{x,y}$ can be computed for all $x\in\mathcal{B}$ and y by solving

$$\mathop{\mathrm{argmin}}_{S\in\mathcal{S},\ \{r_x\}}\sum_{x\in\mathcal{B}}\Big[\delta\,\|S\|_F^2\,\|r_x\|^2+\sum\nolimits_y\tilde{R}(r_x S_y^\top)^2\Big] \quad (10)$$

$$s.t.\quad S_{yy}\left\langle r_x,\tilde{k}_x\right\rangle = 1, \quad \forall\, y,\, x\in\mathcal{D}. \quad (11)$$

A discussion of efficient optimization strategy is deferred to Appendix B.1. It is noteworthy that different from (6) where each $\tilde{G}_{x,y}$ is computed individually, here we compute them in mini-batches and for all y. This is natural because the class representation matrix S is shared by the examples. Indeed, the $\tilde{R}$ in (8) can also be changed to measure violations on the mini-batch instead of the entire dataset $\tilde{p}$. Ideally one may want to solve (10) over the entire dataset, but this is overly expensive and does not conform with the standard practice of mini-batch based training. In experiment, both the r_x and S computed from $\mathcal{B}$ perform well, and such a middle ground appears effective.

2.3 From warping optimization to warping layer

Inspired by the ability of warping in inducing structure-aware representations, we would naturally like to encapsulate it as a generic layer, so that it can be inserted flexibly into a neural network to promote desired structures. This turns out quite viable thanks to the "conditional separability" in (10). In particular, once S is fixed, the optimization over all r_x becomes independent. Therefore, we can regard the shared $S \in \mathcal{S}$ as a trainable model parameter associated with a layer (like convolution weights), and then consider r_x as the output of the layer in response to the input $\tilde{k}_x$:

$$W_S : \mathbb{R}^d \to \mathbb{R}^d, \text{ with} \quad (12)$$
$$h \mapsto \operatorname*{argmin}_{r_x : S_{11}\langle r_x, h\rangle = 1} \delta \|S\|_F^2 \|r_x\|^2 + \sum_y \tilde{R}(r_x S_y^\top)^2.$$

Clearly this layer creates a bi-level optimization for the overall training objective. Given the gradient with respect to the layer's output $W_S(h)$, we need to compute the gradient with respect to both S and the input h. Leveraging implicit differentiation and following Amos & Kolter (2017), we can derive the exact formula for backpropagation, and the details are relegated to Appendix B.2. It is noteworthy that in the rank-one setting, the onus of incorporating the prior structure is much more on the feature representation r_x which must be optimized *with respect to a given* S_y. However, although it appears different from Eq 10 by not directly optimizing over S, S is still judiciously optimized through other pursuits such as regularization, and the task specific objective, etc.

Remarks. Compared with (10) which computes the warped representation $\tilde{G}_{x,y}$ by jointly optimizing S and r_x, the warping layer in (12) only optimizes r_x, while keeping the class representation S as a modulating parameter. By lifting S out of the inner level of the bi-level optimization, we allow S to be optimized in a broader context, *e.g.*, when it also participates in a generative model instead of only a discriminative model (Section 4).

We will next demonstrate the flexibility and effectiveness of the warping layer via two low-data tasks.

3 WARPING LAYER FOR FEW-SHOT LEARNING

Few-shot learning is a classic learning scenario in the low-data regime, where structured label prior can be significantly beneficial. In this section, we will briefly review few-shot learning settings and explain how to incorporate such priors through the warping layer.

3.1 Few-shot learning background

Few-shot learning aims at inferring a classifier that adapts to unseen classes after observing only a few examples from each class. The problem is usually formulated as meta-learning where the goal is to learn a base learner and a meta learner (Finn et al., 2017; Ravi & Larochelle, 2017; Snell et al., 2017; Vinyals et al., 2016; Lee et al., 2019). Generally, the meta learner focuses on learning a embedding network that maps the input to a feature space, while the base leaner aims at learning a classifier that can well separate the representations in the feature space. The overall meta-learning goal is to learn embedding representations so that the base learner can generalize well across tasks.

Few-shot learning is typically formulated as K-way, N-shot classification tasks. A collection of tasks (episodes) $\mathcal{T}$ are sampled on the fly from the dataset. In each task $\mathcal{T}^{(i)} = (P^{(i)}, Q^{(i)})$, K different classes are chosen with N samples for each class, and these samples form a support set $P^{(i)} = \{(x_n, y_n) : n = 1, ..., K \times N\}$. Typically, N is a small value in $[1, 5]$. The same K classes with M different samples are sampled in addition, forming a query set $Q^{(i)} = \{(x_n, y_n) : n = 1, ..., K \times M\}$. The overall goal of learning is to minimize the prediction error across tasks on the query sets given the support sets. To summarize, the meta objective can be formulated as:

$$\min_\phi \mathop{\mathbb{E}}_{\mathcal{T}} \mathop{\mathbb{E}}_{(x,y)\in Q^{(i)}} \mathcal{L}^{meta}(g(f_\phi(x)), y), \quad (13)$$

where f_ϕ is the embedding network and ϕ is the meta parameter. g is the base learner, for which various choices are available. A typical one is the nearest neighbor classifier, such as in matching network (Vinyals et al., 2016) and prototypical network (Snell et al., 2017). They learn the representation of each class in the feature space by the examples in the support set. Then the query examples are matched to their nearest neighbor classes based on a certain distance metric. The base learner g is defined as the posterior probability of each class k in the current task:

$$g_k(f_\phi(x)) = \frac{\exp(-d(f_\phi(x), c_k))}{\Sigma_{k'} \exp(-d(f_\phi(x), c_{k'}))}, \quad (14)$$

where c_k is the center of class k computed from the corres-ponding examples in the support set $P^{(i)}$ (denoted as $P_k^{(i)}$):

$$c_k = |P_k^{(i)}|^{-1} \cdot \Sigma_{(x_n,y_n)\in P_k^{(i)}} f_\phi(x_n). \tag{15}$$

Another choice of base learner is a discriminatively trained linear classifier, e.g., SVM (Lee et al., 2019) or ridge regression (Bertinetto et al., 2019). In this case, the base parameter θ for g is learned through the following objective:

$$\underset{\theta}{\operatorname{argmin}} \underset{(x,y)\in P^{(i)}}{\mathbb{E}} \mathcal{L}^{base}(g_\theta(f_\phi(x)), y). \tag{16}$$

3.2 Incorporating label structures by warping layer

To leverage the structured label prior through the warping layer W_S as constructed in (12), we can encode the prior into the embeddings learned by meta learner. As a result, the meta learner's objective becomes

$$\min_{\phi, S\in\mathcal{S}} \underset{\mathcal{T}}{\mathbb{E}} \underset{(x,y)\in Q^{(i)}}{\mathbb{E}} \mathcal{L}^{meta}(g(W_S(f_\phi(x))), y). \tag{17}$$

The parameter S which captures the property of classes will be learned along with other meta parameters like ϕ. The behavior of the base learner remains intact. The new embeddings $W_S(f_\phi(x))$ are thus encoded with desired prior without sacrificing the freedom of choosing the base learner.

We remark in passing that the effective application of warping requires the availability of logical relationships covering the classes in the latest task. This is realistic in practice because the training data in the support set do carry the label, and many commonly used datasets are derived from ImageNet which carries a label hierarchy in the first place.

4 WARPING LAYER FOR SEMI-SUPERVISED VAE

Semi-supervised learning is another common scenario in learning with low data, and the key challenge is to make full use of unlabeled examples. One approach is the VAE model. It encompasses a discriminative learning branch (encoder) for inference, and a generative decoder, which allows unlabeled examples to improve the learning of latent representation. We will first briefly review the components in semi-supervised VAE, and then introduce how to apply the warping layer to instill the label structure. Specifically, we will explain how the prior is incorporated in *both* the inference/encoding *and* the generation/decoding process.

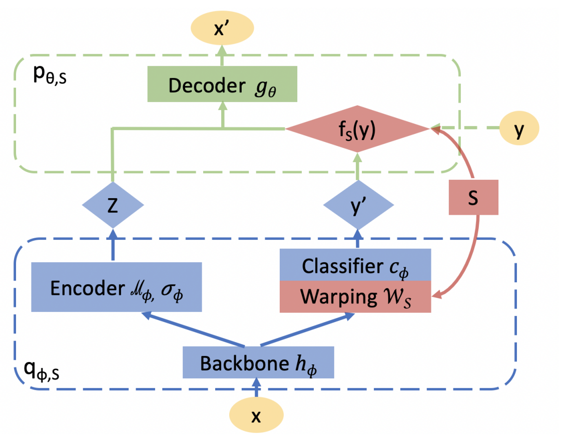

Figure 2: Apply warping to semi-supervised VAE model. Variables in rectangles indicate trainable parameters or networks, and diamond shaped variables indicate latent variables. The encoder and decoder data flows are shown as **blue** and **green** arrows, respectively, while the maroon arrows indicate the parameters concerning the warping layer.

4.1 Semi-supervised VAE background

Semi-supervised VAE (Narayanaswamy et al., 2017; Kingma et al., 2014) builds a probabilistic model to make use of unlabeled data. It consists of a generation process and an inference process parameterized by θ and ϕ respectively. In the generation process, the data is generated by a latent class variable y and a latent continuous variable z:

$$p_\theta(x|y,z) = \mathcal{N}(x|g_\theta(y,z)) \tag{18}$$
$$p(z) = \mathcal{N}(z|0,I) \tag{19}$$
$$p(y) = \mathrm{Cat}(y|\pi) \tag{20}$$
$$p(\pi) = \mathrm{SymDir}(\gamma), \tag{21}$$

where $\mathrm{Cat}(y|\pi)$ is the multinomial distribution of the class variable. SymDir is the symmetric Dirichlet distribution with hyper-parameter γ.

The inference process learns disentangled representation and assumes z and y are conditionally independent given x. Thus the approximate posterior $q_\phi(y,z|x)$ can be factorized as follows:

$$q_\phi(y,z|x) = q_\phi(z|x)q_\phi(y|x) \tag{22}$$
$$\text{where} \quad q_\phi(z|x) = \mathcal{N}(z|\mu_\phi(x), \sigma_\phi^2(x)) \tag{23}$$
$$q_\phi(y|x) = \mathrm{Cat}(y|c_\phi(h_\phi(x))). \tag{24}$$

Here $\mu_\phi, \sigma_\phi, h_\phi, c_\phi$ are all defined as neural networks. h_ϕ can be regarded as the backbone network for learning hidden representations, and c_ϕ as the classifier network which outputs probability logits.

4.2 Incorporating label structures by warping layer

Our first step is to incorporate the structured label prior into the semi-supervised VAE by embedding the warping layer of (12) in the *inference* process. This is analogous to warping the few-shot model in (17), changing the posterior distribution from (24) to

$$q_{\phi,S}(y|x) = \text{Cat}(y|c_\phi(W_S(h_\phi(x)))). \tag{25}$$

What makes warping particularly interesting and effective in VAE is the possibility of further utilizing the class representation S in the *generation* process. Noting that S captures the relationship between classes, we inject it into the generation process by replacing the posterior distribution of x in (18) as follows, while leaving the prior distributions $p(y), p(z), p(\pi)$ unchanged:

$$p_{\theta,S}(x|y,z) = \mathcal{N}(x|g_\theta(f_S(y), z)) \quad \text{where} \quad f_S(y) = S_y.$$

<u>*Remarks*</u>. The final model is illustrated in Figure 2. Integrating S_y into the decoder is not an adventitious reuse of model, but carries subtle and significant consequences. Intuitively, S_y captures salient features for identifying class y and promotes structure-aware representations. But whether it thus provides better clue for reconstructing the input x remains debatable. However, in the low-data regime, S_y appears more informative than the one-hot vector e_y in providing a basis for generative modeling. Therefore, as also corroborated by empirical performance, such a parameter sharing is significantly beneficial for semi-supervised VAE.

Now we are ready to derive the warped semi-supervised VAE objectives. Following Kingma et al. (2014), the variational lower bound on labeled data is:

$$\begin{aligned}\log p_{\theta,S}(x,y) &\geq \mathop{\mathbb{E}}_{q_{\phi,S}(z|x,y)}[\log p_{\theta,S}(x|y,z) + \log p(y) \\ &\quad + \log p(z) - \log q_{\phi,S}(z|x,y)] \\ &= \mathop{\mathbb{E}}_{q_{\phi,S}(z|x,y)}[\log p_{\theta,S}(x|y,z) + \log p(y)] \\ &\quad - \text{KL}[q_{\phi,S}(z|x,y)||p(z)] \\ &=: \text{ELBO}_{\mathcal{L}}(x,y).\end{aligned}$$

For unlabeled data, y is treated as a latent variable which is then marginalized out over all classes. The variational lower bound on unlabeled data is:

$$\begin{aligned}\log p_{\theta,S}(x) &\geq \mathop{\mathbb{E}}_{q_{\phi,S}(z,y|x)}[\log p_{\theta,S}(x|z,y) + \log p(y) \\ &\quad + \log p(z) - \log q_{\phi,S}(y,z|x)] \\ &= \mathop{\mathbb{E}}_{q_{\phi,S}(y|x)}[\mathop{\mathbb{E}}_{q_{\phi,S}(z|x)}[\log p_{\theta,S}(x|y,z) + \log p(y)] \\ &\quad - \text{KL}[q_{\phi,S}(z|x)||p(z)] + \log q_{\phi,S}(y|x)] \\ &= \Sigma_y q_{\phi,S}(y|x)[\text{ELBO}_{\mathcal{L}}(x,y) + \log q_{\phi,S}(y|x)] \\ &=: \text{ELBO}_{\mathcal{U}}(x).\end{aligned}$$

The classification loss is $\mathcal{L}_c = -\log q_{\phi,S}(y|x)$. Combined with the VAE objectives for both labeled and unlabeled data, the final objective for the warped semi-supervised VAE can be written as

$$\mathcal{L} = \mathcal{L}_c - \lambda \mathop{\mathbb{E}}_{(x,y)\sim D_L} \text{ELBO}_{\mathcal{L}}(x,y) - \lambda \mathop{\mathbb{E}}_{x\sim D_U} \text{ELBO}_{\mathcal{U}}(x),$$

where D_L and D_U stand for the labeled and unlabeled datasets, respectively. The hyperparameter λ controls the trade-off between ELBO and the discriminative loss.

5 EXPERIMENT

We now demonstrate that the warping layer empirically improves the performance in low-data regimes by incorporating the hierarchical prior. We first present the evaluation results and analysis on few-shot learning, and then move on to semi-supervised learning with warped VAE. The implementation code for both tasks are available in Github[1]

5.1 Experiment on few-shot learning

We tested on three datasets for few-shot learning, and their corresponding hierarchical prior is summarized as follows.

miniImageNet: The miniImageNet dataset (Vinyals et al., 2016) consists of 100 classes, each having 600 images with resolution 84×84. In few-shot learning, 16 and 20 classes were reserved for validation and testing, and the remaining 64 classes were used for training. Since miniImageNet is a subset sampled from ImageNet which carries a hierarchical label structure derived from WordNet (Miller, 1995), it inherits the hierarchy. There are 100 observed classes and 43 hidden/non-leaf classes. The hierarchy trivially specifies the implication constraints, and we extracted exclusion constraints from siblings of the same parent. As a result, we derived 138 implication and 202 exclusions relations that were used in the warping operator.

CIFAR-FS: The CIFAR-FS dataset (Bertinetto et al., 2019) contains all the 100 classes form CIFAR100. In each class, there are 600 images of size 32×32. The dataset was split into 64, 16, 20 classes for training, validation, and testing, respectively. CIFAR100 comes with super-classes. However, there is only one level of super-class and it is too shallow to serve as a useful prior. So instead, we followed the hierarchy of CIFAR100 used in Barz & Denzler (2019) to derive implication and exclusion relations. This amounted to 100 observed classes and 50 hidden

[1]https://github.com/myy920213/WarpingLayer

Table 1: Test accuracy on 5-way 1-shot and 5-way 5-shot tasks (#shot refers to support set at testing)

	mini-imagenet		CIFAR-FS		FC100	
Methods	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
ProtoNet	53.38±0.52	70.43±0.42	56.27±0.64	72.60±0.60	35.42±0.58	48.65±0.49
ProtoNet+W	**54.19**±0.59	**70.96**±0.51	**56.92**±0.69	**73.34**±0.58	**35.90**±0.56	**49.12**±0.52
MetaSVM	52.75±0.59	68.65±0.40	56.65±0.68	73.14±0.59	34.88±0.58	48.22±0.55
MetaSVM+W	**54.03**±0.60	**69.42**±0.42	**58.02**±0.71	**74.89**±0.68	**35.68**±0.62	**48.98**±0.60
R2D2	52.12±0.54	68.42±0.40	56.50±0.62	72.76±0.62	34.76±0.59	47.78±0.56
R2D2+W	**53.44**±0.58	**69.02**±0.45	**57.49**±0.70	**73.49**±0.65	**35.22**±0.59	**48.25**±0.54

classes, along with 149 implication and 182 exclusion relations.

FC100: FC100 (Oreshkin et al., 2018) is another dataset derived from CIFAR100. It shares the same number and size of images as CIFAR-FS, but differs in the dataset partitioning. As was mentioned above, CIFAR100 comes with super-classes. FC100 was split along super-classes, and the training set consisted of 60 randomly selected classes out of 12 super-classes. 20 classes from 4 super-classes were sampled for validation, and another 20 classes from another 4 super-classes for testing. The implication and exclusion relations used for warping were the same as those in CIFAR-FS.

Implementation details To pre-process the data, we applied horizontal flip, random crop, and color jitter following the practice in Lee et al. (2019). We used the standard 4-layer convolution network in Snell et al. (2017) as the embedding network for the meta learner. The embedding dimension is $d = 1600$ for miniImagenet, and $d = 256$ for both CIFAR-FS and FC100. The warping operator was implemented by following Section 3.2, and we adopted linear kernel in practice.

The regularizer $\tilde{R}$ in (8) requires computing the expectation over the entire training set. To save computation, we selected five examples from each class as anchor points to represent the whole training set. Empirically, it appeared effective to choose the five points lying closest to the center of each class in the input image space. We also tried selecting anchor points based on the embedding space, but observed no significant difference in the result. The hyperparameter δ was selected from $[50, 20, 10, 5, 1, 0.5, 0.1]$ based on the performance on the validation set.

Baselines We evaluated the effectiveness of warping by applying it to three different base learners. The first is the soft distance based classifier used by Prototypical Network (ProtoNet, Snell et al., 2017). The other two are linear classifiers, namely SVM (MetaSVM, Lee et al., 2019) and ridge regression (R2D2, Bertinetto et al., 2019).

We **emphasize** that our aim is to demonstrate warping as a general performance booster flexibly applicable to a range of base learners. Hence, the performance difference resulting from the base learners themselves is not of our concern.

Setup We first trained each baseline model to their best by following the strategy in Lee et al. (2019), including the learning rate schedule, and each model was trained for 60 epochs by SGD. Afterwards, we inserted the warping layer, and continued training for 20 more epochs by SGD.

We set training episodes to 5-way 5-shot for all base learners and datasets, except that 10 shots were used for FC100 when training MetaSVM and R2D2 to obtain competitive performance. This remains a fair comparison for warping, and the same shot setting was used after applying warping. The query sets contained 5 examples at training time and 15 examples at test time for all methods and datasets.

Results Table 1 shows the test accuracy under 5-way 1-shot and 5-way 5-shot. ProtoNet+W, MetaSVM+W, and R2D2+W correspond to applying the warping layer to ProtoNet, MetaSVM, and R2D2, respectively. Each setting was evaluated over 2000 episodes to report the mean and standard deviation. Clearly, warping can significantly improve the performance across all datasets and base learners. The improvement is more significant in 1 shot, indicating the hierarchy prior is more informative in lower data settings. To the best of our knowledge, there is no existing few-shot learning algorithm that can leverage such a prior.

5.2 Experiment on Semi-supervised VAE

In this section, we will investigate the effectiveness of applying the warping layer to semi-supervised VAE.

Datasets We conducted semi-supervised learning on two datasets: CIFAR100 and miniImageNet. The statistics of the datasets, along with the implication and exclusion relations, are the same as in Section 5.1 for few-shot learning. On both datasets, 40k examples

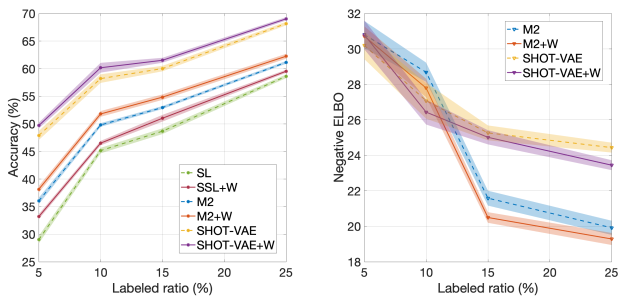

Figure 3: Accuracy and negative ELBO for CIFAR100

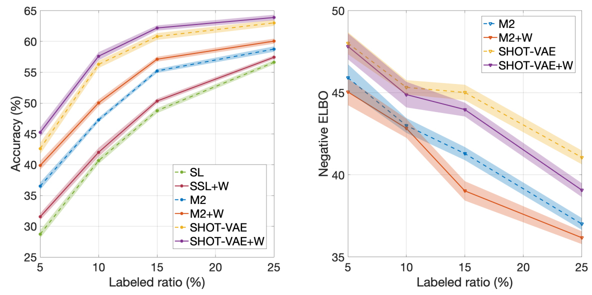

Figure 4: Accuracy and negative ELBO for miniImageNet

were used for training, 10k for validation, and 10k for testing.

Baselines We selected two standard VAE models for semi-supervised learning as baselines, and then applied the warping layer to them respectively for comparison. The first baseline is the M2 model proposed in Kingma et al. (2014), and its structure was introduced in Section 4.1. The other one is SHOT-VAE (Feng et al., 2020), which learns disentangled hidden representations to utilize unlabeled data for improving inference accuracy. It also applies mixup and the label smoothing technique to further improve the efficiency of using unlabeled examples. To apply the warping layer efficiently, we selected anchor points from *labeled* dataset to represent the whole training set, same as in the few-shot learning. The resulting models are referred to as M2+W and SHOT-VAE+W.

For reference, we also included a supervised learning method (denoted as SL), which is simply the classification branch of the M2 model (Y in Figure 2 but without using $\mathcal{W}_S$). Although this is no longer a generative model, warping can still provide an effective mechanism to tap into unlabeled data. In particular, we incorporated W_S to SL as in Figure 2 (no decoder included), and let the regularizer $\tilde{R}$ take expectation from the union of anchor points and a random subset of *unlabeled* data. We call the model SSL+W.

Implementation details We used WideResNet-28-2 as the backbone network and the embedding dimension was $d = 512$ for both datasets. The hyperparamter δ was selected from $[10, 50, 100, 150, 200, 500]$ based on the validation set. The decoder architecture was directly adopted from Feng et al. (2020). The images in miniImageNet are sized 84×84 which is different from CIFAR100. According to our observation, reconstructing larger dimensional images hardly improves the inference accuracy. Thus, when training models on miniImageNet, we fed the original images as input to backbone network, but used down-sampled images (32×32) for reconstruction to save computation.

We followed the strategy in Feng et al. (2020) to train each baseline, including the learning rate schedule. Each model was trained for 700 epochs with SGD. To apply warping, we first trained each baseline for 400 epochs, and then added warping before resuming the training process up to 700 epochs.

Results Figures 3 and 4 demonstrate the test performance attained by all learners on CIFAR100 and miniImageNet, respectively. The numerical results are listed in Table 2 and 3 in Appendix A for reference. We evaluated each method five times by randomly splitting labeled and unlabeled examples to report the mean and standard deviation. We also varied the ratio of labeled examples on each dataset. For VAE based methods, we reported both inference accuracy and negative ELBO value to illustrate the impact on both the inference process and the generation process. For non-VAE based methods, only inference accuracy was reported.

Our focus is again on the effect of warping on the original models. Based on the two figures, it is clear that the warping layer always promotes the performance of inference accuracy. For VAE based methods, the warping layer can lower the negative ELBO value, indicating improved generation process as well. On both datasets, semi-supervised VAE clearly outperforms SL (supervised learning), and SHOT-VAE outperforms M2. Warping boosts the accuracy slightly more significantly on M2 than on SHOT-VAE. SSL-W does predict more accurately than SL thanks to the warping layer which incorporates both unlabeled data and the hierarchical structure. But overall it still inferior to generative approaches. In almost all cases, warping is more effective when the labeled set is smaller.

Ablation study To investigate the impact of warping layer in more depth, we set up two ablation experiments. Firstly, recall that the hierarchical label prior is incorporated by not only inserting the warping layer in the inference process, but also sharing S with the decoder. To analyze the source of gain, we stud-

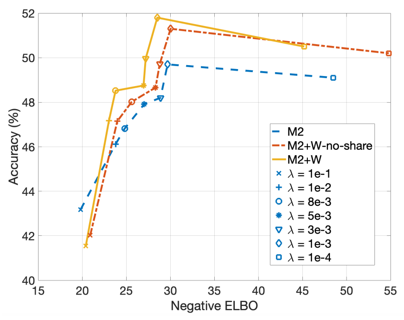

Figure 5: Accuracy v.s. negative ELBO on various trade off parameter λ

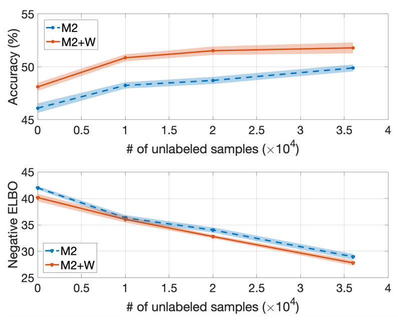

Figure 6: Accuracy and negative ELBO v.s. # unlabeled examples

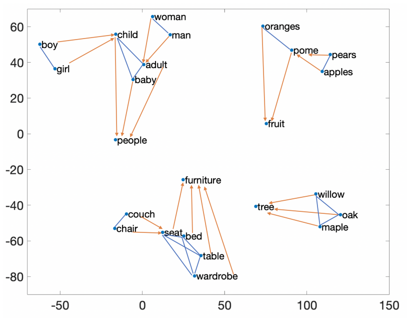

Figure 7: t-SNE embedding of class representations

ied the consequence of *not* sharing S with the decoder (M2+W-no-share). The experiment was conducted on CIFAR100 with 10% labeled examples. Moreover, we varied the trade off weight $\lambda \in$ [1e-4, 1e-1] to better illustrate the impact on both inference accuracy and ELBO. This range ensured that both labeled and unlabeled data were given appropriate weights.

Figure 5 illustrates the trade off between accuracy and negative ELBO under varied values of λ (indicated by +,*, etc.). The upper left corner represents the ideal result. As warping was only applied in the classification branch, the best accuracy achieved by M2+W-no-share is 0.4% below that of M2+W. In addition, even under other values of λ where M2+W-no-share achieves similar accuracy as M2+W, it is still inferior in negative ELBO (higher), indicating a sacrifice of performance in the generation process. These two observations corroborate the importance of sharing S, indicating that the learned S can efficiently characterize the prior, hence benefiting both the inference and the generation process. More results under other percentages of labeled data (5, 15, 25%) are available in Appendix A.

Secondly, we would like to study whether warping suppresses the efficiency of utilizing unlabeled examples. So we tested M2 and M2+W on CIFAR100 with 10% (4k) labeled examples, but varied the number of unlabeled examples. As can be read from Figure 6, warping improves the accuracy and ELBO under all possible sizes of unlabeled set. Thus, warping does not suppress the efficiency of using unlabeled data; conversely, it offers additional promotion.

Explainable parameters The semi-supervised VAE with a warping layer learns the representation of classes as encoded by S. Different from most neural network layers where the learned parameters are hard to explain, the value of S turns out explainable. To verify that S_y represents the classes, we plot in Figure 7 the t-SNE embeddings of S_y for some typical classes in the hierarchy. We added directed and undirected edge to illustrate implication and exclusion relations, respectively. Clearly, classes that share similar relations with other classes trend to gather closer. For example, *girl* and *boy* share the same relations with higher level classes like *child* and *people*. Their representations are close but not too close to overlap because of the exclusion relations between them. Therefore, the learned class representations well reflect the enforced prior.

Computation complexity To demonstrate the efficiency of warping layer, we here provide the computation complexity analysis. The forward pass of warping layer requires solving the optimization in Eq 12, whose bottleneck is the inner product between all r_x in the current batch and all k_x in the anchor point set. So the computation cost is (# anchor point) × (# hidden dimension) × (mini-batch size) × (# BFGS iterations). In our experiment, all datasets used 500 anchor points, and the hidden dimension was in $[256, 1024]$. 10 steps of BFGS gave sufficiently good results. These led to just one second for each forward pass. Backpropagation given in §B.2 costs the same order of magnitude.

6 CONCLUSION

We developed a novel warping layer that allows label structure priors to be effectively incorporated into feature and class representation learning. Improved performance was achieved in few-shot learning and semi-supervised learning. Future work will extend the technique to a broader range of priors, such as smoothness in label space (mixup, Zhang et al., 2018) and relational learning with more involved logic (Getoor & Taskar, 2007).

Acknowledgements

We thank the reviewers for their constructive comments. This work is supported by NSF grant RI:1910146.

References

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

Barz, Björn and Denzler, Joachim. Hierarchy-based image embeddings for semantic image retrieval. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

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

Cai, Lijuan and Hofmann, T. Hierarchical document categorization with support vector machines. In *the Thirteenth ACM conference on Information and knowledge management*, pp. 78–87, New York, NY, USA, 2004. ACM Press.

Cesa-Bianchi, Nicolò, Gentile, Claudio, and Zaniboni, Luca. Incremental algorithms for hierarchical classification. *Journal of Machine Learning Research (JMLR)*, 7:31–54, 2006.

Chen, Zhao-Min, Wei, Xiu-Shen, Wang, Peng, and Guo, Yanwen. Multi-label image recognition with graph convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Deng, Jia, Ding, Nan, Jia, Yangqing, Frome, Andrea, Murphy, Kevin, Bengio, Samy, Li, Yuan, Neven, Hartmut, and Adam, Hartwig. Large-scale object classification using label relation graphs. In *European Conference on Computer Vision (ECCV)*, 2012.

Feng, Hao-Zhe, Kong, Kezhi, Chen, Minghao, Zhang, Tianye, Zhu, Minfeng, and Chen, Wei. SHOT-VAE: Semi-supervised deep generative models with label-aware ELBO approximations. In *National Conference of Artificial Intelligence (AAAI)*, 2020.

Finn, Chelsea, Abbeel, Pieter, and Levine, Sergey. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, 2017.

Getoor, Lise and Taskar, Ben. *Introduction to Statistical Relational Learning.* The MIT Press, 2007.

Kingma, Diederik P, Mohamed, Shakir, Rezende, Danilo Jimenez, and Welling, Max. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

Lampert, Christoph H, Nickisch, Hannes, and Harmeling, Stefan. Learning to detect unseen object classes by between-class attribute transfer. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

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

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

Ma, Yingyi, Ganapathiraman, Vignesh, Yu, Yaoliang, and Zhang, Xinhua. Convex representation learning for generalized invariance in semi-inner-product space. In *International Conference on Machine Learning (ICML)*, 2020.

Miller, George A. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

Mirzazadeh, Farzaneh, Ravanbakhsh, Siamak, Ding, Nan, and Schuurmans, Dale. Embedding inference for structured multilabel prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Narayanaswamy, Siddharth, Paige, Timothy, Van de Meent, Jan-Willem, Desmaison, Alban, Goodman, Noah, Kohli, Pushmeet, Wood, Frank, and Torr, Philip. Learning disentangled representations with semi-supervised deep generative models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Oreshkin, Boris N, Rodriguez, Pau, and Lacoste, Alexandre. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Ravi, Sachin and Larochelle, Hugo. Optimization as a model for few-shot learning. In *International Conference on Learning Representations (ICLR)*, 2017.

Rousu, Juho, Saunders, Craig, Szedmak, Sandor, and Shawe-Taylor, John. Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research (JMLR)*, 7(Jul):1601–1626, 2006.

Snell, Jake, Swersky, Kevin, and Zemel, Richard S. Prototypical networks for few-shot learning. In *Ad-*

vances in Neural Information Processing Systems (NeurIPS), 2017.

Vinyals, Oriol, Blundell, Charles, Lillicrap, Timothy, Wierstra, Daan, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Williams, Christoper K. I. and Seeger, Matthias. Using the Nystrom method to speed up kernel machines. In Leen, T. K., Dietterich, T. G., and Tresp, V. (eds.), *Advances in Neural Information Processing Systems 13*, pp. 682–688, Cambridge, MA, 2001. MIT Press.

Xian, Yongqin, Akata, Zeynep, Sharma, Gaurav, Nguyen, Quynh, Hein, Matthias, and Schiele, Bernt. Latent embeddings for zero-shot classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Xie, Guo-Sen, Liu, Li, Jin, Xiaobo, Zhu, Fan, Zhang, Zheng, Qin, Jie, Yao, Yazhou, and Shao, Ling. Attentive region embedding network for zero-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Yu, Yunlong, Ji, Zhong, Fu, Yanwei, Guo, Jichang, Pang, Yanwei, and Zhang, Zhongfei. Stacked semantic-guided attention model for fine-grained zero-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Zhang, Hongyi, Cisse, Moustapha, Dauphin, Yann N, and Lopez-Paz, David. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations (ICLR)*, 2018.

Zhou, Jie, Ma, Chunping, Long, Dingkun, Xu, Guangwei, Ding, Ning, Zhang, Haoyu, Xie, Pengjun, and Liu, Gongshen. Hierarchy-aware global model for hierarchical text classification. In *Association for Computational Linguistics (ACL)*, 2020.

Supplementary Material: Warping Layer: Representation Learning for Label Structures in Weakly Supervised Learning

A Additional Experimental Results

A.1 Numerical results on CIFAR100 and miniImageNet

Section 5.2 has shown the results for semi-supervised learning in Figure 3 and 4. For a clearer and more in-depth comparison, we here list the numerical results in Table 2 and 3, respectively. The improvement is more significant when there are fewer labeled examples.

Table 2: Accuracy and ELBO loss results on CIFAR100

Label ratio	5%		10%		15%		25%	
	Acc.	Neg. ELBO	Acc.	Neg. ELBO	Acc.	Neg. ELBO	Acc.	Neg. ELBO
SL	29.02±0.68	-	45.16±0.51	-	48.67±0.32	-	58.62±0.33	-
SSL+W	33.19±0.45	-	46.51±0.40	-	51.02±0.42	-	59.53±0.28	-
M2	36.04±0.58	30.81±0.77	49.40±0.33	28.69±0.54	52.96±0.27	21.59±0.42	61.14±0.25	19.92±0.40
M2+W	**38.15**±0.60	**30.72**±0.56	**51.80**±0.52	**27.80**±0.52	**54.84**±0.50	**20.50**±0.30	**62.25**±0.44	**19.29**±0.34
SHOT-VAE	47.91±0.82	**30.20**±0.77	58.20±0.78	27.08±0.91	60.02±0.50	25.27±0.42	68.17±0.32	24.44±0.38
SHOT-VAE+W	**49.75**±0.52	30.80±0.82	**60.17**±0.89	**26.44**±0.70	**61.52**±0.45	**25.02**±0.48	**69.03**±0.34	**23.45**±0.28

Table 3: Accuracy and ELBO loss results on miniImageNet

Label ratio	5%		10%		15%		25%	
	Acc.	Neg. ELBO	Acc.	Neg. ELBO	Acc.	Neg. ELBO	Acc.	Neg. ELBO
SL	28.71±0.685	-	40.65±0.52	-	48.76±0.40	-	56.62±0.34	-
SSL+W	31.54±0.56	-	42.01±0.72	-	50.33±0.38	-	57.43±0.31	-
M2	36.52±0.61	45.92±0.82	47.27±0.35	43.01±0.45	55.22±0.32	41.30±0.40	58.76±0.44	37.02±0.36
M2+Warping	**39.87**±0.72	**45.05**±0.80	**50.03**±0.56	**42.84**±0.58	**57.12**±0.40	**39.03**±0.56	**60.05**±0.41	**36.18**±0.39
SHOT-VAE	42.61±0.92	48.02±0.67	56.28±0.60	45.33±0.44	60.81±0.55	45.02±0.47	63.02±0.50	41.07±0.42
SHOT-VAE+Warping	**45.24**±0.78	**47.83**±0.79	**57.61**±0.65	**44.89**±0.78	**62.19**±0.42	**43.98**±0.42	**63.89**±0.45	**39.08**±0.42

A.2 More results on ablation study with varied proportion of labeled examples

In Section 5.2, we showed the effect of not sharing S with the generation process under the setting of 10% labeled examples. We now provide in Figures 8 to 10 the results of such effect under additional percentages of labeled examples. In all figures, the inference accuracy first grows as the value of λ decreases, until it reaches the peak. Then the accuracy decays if λ is reduced further. The peak point specifies the most efficient trade-off value of λ for utilizing unlabeled data. An overly high value of λ will distract the learning process from the classification task, while an overly low value suppresses the benefit of unlabeled data. When similar accuracy is achieved under different values of λ, M2+W-no-share usually obtains lower (better) negative ELBO value than M2+W. This corroborates the benefit of sharing the class representation S.

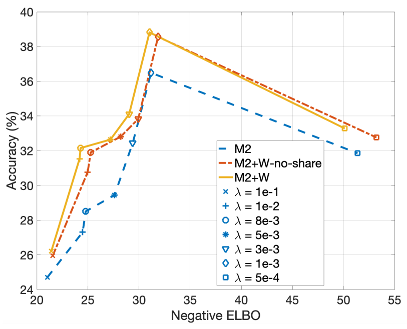

Figure 8: Accuracy v.s. negative ELBO on 5% labeled examples

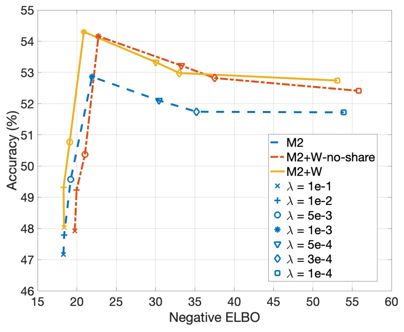

Figure 9: Accuracy v.s. negative ELBO on 15% labeled exmaples

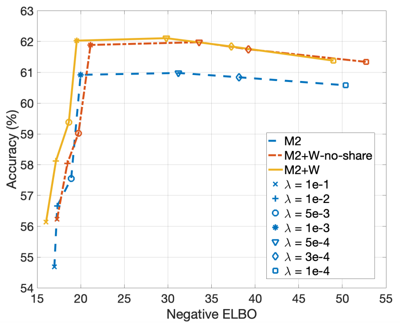

Figure 10: Accuracy v.s. negative ELBO on 25% labeled exmaples

B Optimization Related to Warping Operation

B.1 Efficient optimization of (10)

We first duplicate (10) here for convenience:

$$\min_{S\in\mathcal{S},\ \{r_x\}} \sum_{x\in\mathcal{B}} \left[\delta \left\|S\right\|_F^2 \left\|r_x\right\|^2 + \sum_y \tilde{R}(r_x S_y^\top)^2\right] \tag{26}$$

$$s.t. \quad S_{yy}\left\langle r_x, \tilde{k}_x\right\rangle = 1, \quad \forall\, y,\, x \in \mathcal{B}. \tag{27}$$

The main difficulty lies in the nonconvex constraint, making it hard to project to the feasible domain. We resolve this issue by first noticing that, by the constraint, there is $\alpha \in \mathbb{R}$, such that $S_{yy} = \alpha$, and $\left\langle r_x, \tilde{k}_x\right\rangle = \frac{1}{\alpha}$. So we can reparameterize S_y by

$$S_y = \alpha e_y + \boldsymbol{\beta}_y, \quad \text{where} \quad \beta_{yy} = 0. \tag{28}$$

This allows us to rewrite the objective as

$$\min_{\alpha} \min_{\{\boldsymbol{\beta}_y\}:\beta_{yy}=0} \min_{\{r_x\}} \sum_{x\in\mathcal{B}} \left[\delta\left(m\alpha^2 + \sum_y \left\|\boldsymbol{\beta}_y\right\|^2\right)\left\|r_x\right\|^2 + \sum_y \tilde{R}(r_x(\alpha e_y + \boldsymbol{\beta}_y)^\top)^2\right] \tag{29}$$

$$s.t. \quad \left\langle r_x, \tilde{k}_x\right\rangle = 1/\alpha, \quad \forall\, x \in \mathcal{B}. \tag{30}$$

So given α, the projection to the constraint set is just projecting to a hyperplane, which admits a closed-form solution. Furthermore, given α and $\{\boldsymbol{\beta}_y\}$, the optimization over $\{r_x\}$ is convex. Similarly, given α and $\{r_x\}$, the optimization over $\{\boldsymbol{\beta}_y\}$ is also convex. So we can alternate in optimizing $\{\boldsymbol{\beta}_y\}$ and $\{r_x\}$. Finally, the outer-loop optimization over α is simply a 1-D line search, and with the gradient in α, the optimization is generally completed in about 10 iterations.

B.2 Details on implicit differentiation

As $R(v)$ is sublinear in v, we denoted $\tilde{R}$ as $\tilde{R}(v) := \sup_{u\in C} \tilde{u}^\top v$ at the end of Section 2.1. Given x and the corresponding optimal v, let $u := \operatorname{argmax}_{u\in C} \tilde{u}^\top v$ and $z := \tilde{u}$. So $\tilde{R}^2$ can be approximated about v by $(z^\top v)^2$, and we further define $Q = 2(\delta I + zz^\top)$. We regretfully note that there was a typo in Equations (5), (6), (10), (12) where all R and $\tilde{R}$ should be squared, to be consistent with Ma et al. (2020). Our implementation did follow the squared formulation. Then the Lagrangian of (5) can be derived as:

$$G(v, \lambda) = \frac{1}{2} v^\top Q v + \lambda(v^\top \tilde{k}_x - a) \tag{31}$$

where λ is the dual variable for equality constraint. The KKT conditions of the above optimization are:

$$Qv^* + \lambda^* \tilde{k}_x = 0 \qquad\qquad \tilde{k}_x^\top v^* - a = 0 \tag{32}$$

Taking the differentials of these conditions we can get:

$$\underbrace{\begin{bmatrix} Q & \tilde{k}_x \\ \tilde{k}_x^\top & 0 \end{bmatrix}}_{:=A} \begin{bmatrix} \mathrm{d}v \\ \mathrm{d}\lambda \end{bmatrix} = \underbrace{\begin{bmatrix} \mathrm{d}Qv^* + \lambda^* \mathrm{d}\tilde{k}_x \\ v^{*\top}\mathrm{d}\tilde{k}_x - \mathrm{d}a \end{bmatrix}}_{:=B}. \tag{33}$$

We can derive the derivatives of different parameters by taking the matrix vector product between the Jacobian matrix and backward pass vector $\frac{\partial L}{\partial v}$. For example: $\frac{\partial L}{\partial \tilde{k}_x} = \frac{\partial v^*}{\partial \tilde{k}_x}^\top \frac{\partial L}{\partial v}$. The Jacobian $\frac{\partial v^*}{\partial \tilde{k}_x}$ can be obtained by setting $\mathrm{d}\tilde{k}_x = I$ and other parameters to 0, followed by solving for $\mathrm{d}v$. However, we never need to have the actual Jacobian for backpropogation:

$$\begin{bmatrix} \frac{\partial v^*}{\partial \tilde{k}_x} \\ \frac{\partial \lambda^*}{\partial \tilde{k}_x} \end{bmatrix} = A^{-1} \underbrace{\begin{bmatrix} \lambda^* I \\ v^{*\top} \end{bmatrix}}_{:=C}, \qquad \frac{\partial L}{\partial \tilde{k}_x} = C^\top (A^\top)^{-1} \begin{bmatrix} \frac{\partial L}{\partial v^*} \\ 0 \end{bmatrix}. \tag{34}$$

Notice that we can first compute:

$$\begin{bmatrix} d_v \\ d_\lambda \end{bmatrix} = (A^\top)^{-1} \begin{bmatrix} \frac{\partial L}{\partial v} \\ 0 \end{bmatrix}. \tag{35}$$

The matrix inversion can be efficiently addressed via the conjugate gradient method. Then the gradients with respect to $\tilde{k}_x$ and all other parameters can be derived by

$$\nabla L(\tilde{k}_x) = \lambda^* d_v + d_\lambda v^* \tag{36}$$

$$\nabla L(a) = -d_\lambda \tag{37}$$

$$\nabla L(Q) = d_v v^{*\top} + v^* d_v^\top. \tag{38}$$

And the gradients for other parameters in Q or a can be derived through the chain rule.