Stochastic Prototype Embeddings

Tyler R. Scott[12], **Karl Ridgeway**[1], **Michael C. Mozer**[13]
[1]University of Colorado, Boulder
[2]Sensory Inc.
[3]Google Research
tysc7237@colorado.edu, karl.ridgeway@gmail.com, mcmozer@google.com

Abstract

Supervised deep-embedding methods project inputs of a domain to a representational space where same-class instances lie near one another and different-class instances lie far apart. We propose a probabilistic method that treats embeddings as random variables. Based on a state-of-the-art deterministic method, Prototypical Networks (Snell et al., 2017), our approach supposes the existence of a class prototype around which class instances are Gaussian distributed. The prototype posterior is a product distribution over labeled instances, and query instances are classified by marginalizing relative prototype proximity over embedding uncertainty. We describe an efficient sampler for approximate inference that allows us to train the model at roughly the same space and time cost as its deterministic sibling. Incorporating uncertainty improves performance on few-shot learning and gracefully handles label noise and out-of-distribution inputs. Compared to a recent stochastic alternative, Hedged Instance Embeddings (Oh et al., 2019), we achieve superior large- and open-set classification accuracy.

1 Introduction

Supervised deep-embedding methods map instances from an input space to a latent embedding space in which same-label pairs are near and different-label pairs are far. The embedding thus captures semantic relationships without discarding inter-class structure. In contrast, consider a standard neural network classifier with a softmax output layer trained with a cross-entropy loss. Although its penultimate layer might be treated as an embedding, the classifier's training objective attempts to orthogonalize all classes and thereby eliminate any information about inter-class structure.

Embedding methods are popular in the few-shot learning literature [4, 5, 10, 14, 16, 19, 20, 23, 25] where the goal is to classify query instances based on one or a small number of labeled exemplars of novel classes. These methods operate by embedding the queries and exemplars using a pre-trained network, and classifying each query according to its proximity to the exemplars. Embedding methods are also critical in open-set recognition domains such as face recognition and person re-identification [3, 12, 13, 18, 22, 24, 26–28].

Loss functions used to obtain embeddings can be characterized according to the number of instances used to specify a loss. To describe these losses, we will use the notation z_α for an embedding of class α. *Pairwise* losses attempt to minimize within-class distances, $||z_\alpha - z'_\alpha||$, and maximize between-class distances, $||z_\alpha - z_\beta||$ [3, 7, 27]. *Triplet* losses attempt to ensure within-class instances are closer than between-class instances, $||z_\alpha - z'_\alpha|| < ||z_\alpha - z_\beta||$ [18, 22, 26]. *Quadruplet* losses attempt to ensure every within-class pair is closer than every between-class pair, $||z_\alpha - z'_\alpha|| < ||z''_\alpha - z_\beta||$ [24]. Finally, *cluster-based losses* attempt to use all instances of a class [16, 17, 20, 21]; for example, *Prototypical Networks (PN)* compute the mean of a set of instances of a class, $\bar{z}_\alpha$, and ensure that additional instances of that class, z_α, satisfy a proximity constraint such as $||z_\alpha - \bar{z}_\alpha|| < ||z_\alpha - \bar{z}_\beta||$.

Preprint. Under review.

Nearly all methods previously proposed for deep embeddings are deterministic: an instance projects to a single point in the embedding space. Deterministic embeddings fail to capture uncertainty due either to out-of-distribution inputs (e.g., data corruption) or label ambiguity (e.g., overlapping classes). Representing uncertainty is important for many reasons, including robust classification and decision making, informing downstream models, interpreting representations, and detecting out-of-distribution samples.

In this article, we propose a method for discovering *stochastic* embeddings, where z_α is a random variable whose distribution reflects the uncertainty in the embedding space. We are aware of two prior methods to obtain stochastic embeddings for few-shot and open-set classification: the *Hedged Instance Embedding (HIB)* [15] and the *Oracle-Prioritized Belief Network (OPBN)* [8]. The HIB and OPBN are trained using pairwise and triplet losses, respectively. Cluster-based methods have proven to be superior to pairwise and triplet methods for deterministic embeddings [19]. This ranking motivated us to explore a cluster-based method for stochastic embeddings.

Our proposed method, the *Stochastic Prototype Embedding (SPE)*, is a variant of the PN [20], described above. PNs have yet to receive a proper probabilistic treatment, although authors have begun to consider small steps in this direction. For example, Fort [6] incorporated a Mahalanobis distance with covariance specified by the distribution of instances forming the prototype. As in the PN, our SPE assumes each class can be characterized by a prototype in the embedding space and an instance is classified based on its proximity to a prototype. In the case of the SPE, the embeddings and prototypes are Gaussian random variables, each class instance is assumed to be a Gaussian perturbation of the prototype, and a query instance is classified by marginalizing out over the embedding uncertainty.

Using a synthetic data set, we demonstrate that the embedding uncertainty is related to both input and label noise. On a few-shot learning task, we show that the SPE outperforms its state-of-the-art deterministic sibling, the PN. And on a challenging classification task, we find that the SPE outperforms HIB, the state-of-the-art stochastic embedding method.

2 The Model

The SPE assumes that the latent representation, z, is a Gaussian conditioned on the input, x:

$$p(z|x) = \mathcal{N}(z; \boldsymbol{\mu}_x, \boldsymbol{\sigma}_x^2 \mathbf{I}) \tag{1}$$

with mean, $\boldsymbol{\mu}_x$, and variance, $\boldsymbol{\sigma}_x^2$, computed by a deep neural network, similar to a Variational Autoencoder [9]. (We discuss the possibility of using a full covariance matrix in Appendix D.) The classification, y, in turn is conditioned on z, with $p(y|z)$ taking the same form as in the original PN [20], to be described shortly. Given an input, a class prediction is made by marginalizing over the embedding uncertainty:

$$p(y|x) = \int_z p(y|z)p(z|x)dz, \tag{2}$$

Figure 1 depicts the relationship between the input, latent, and class representations. We train the SPE using the standard few-shot learning paradigm, consisting of a sequence of *episodes*, each with

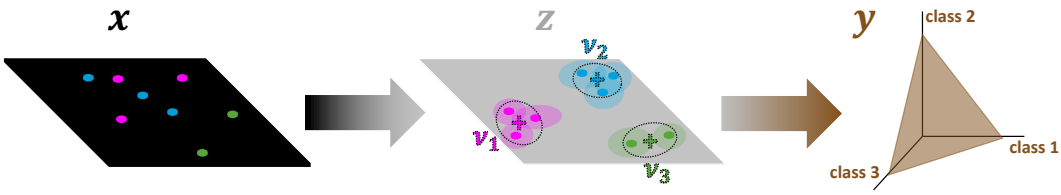

Figure 1: Illustration of the stochastic prototype embedding. The model learns a mapping from input space, x, to embedding space, z, in which same-class instances are near and different-class instances are far. Embeddings are represented as Gaussian random variables. Prototypes, noted as $+$ symbols in the embedding, are formed via a confidence-weighted average of the embeddings of instances known to belong to a class (support instances). Prototype uncertainty is depicted with the dotted ovals. Given the prototypes, a prediction of class y is made for a query instance by marginalizing a softmax prediction over the embedding space.

m instances of n classes. We split the $m \times n$ instances into $k \times n$ *support* examples, defining a set S, and $(m-k) \times n$ *query* examples. The support instances for each class c, $S_c \in S$, are used to determine the class prototype, ν_c, and the query instances are evaluated to predict class label (Equation 2).

2.1 Forming class prototypes

In the SPE, each class y has an associated prototype, ν_y, in the embedding space, and each instance i of class y, denoted x_i, projects to an embedding, z_i, in the neighborhood of ν_y:

$$\nu_y = z_i + \epsilon, \text{ where } \epsilon \sim \mathcal{N}(0, \boldsymbol{\sigma}_\epsilon^2 \mathbf{I}). \tag{3}$$

We assume that the prototype is consistent with all support instances, allowing us to express the likelihood of ν_y as a product distribution:

$$p(\nu_y | S_y) = \frac{\prod_{i \in S_y} p(\nu_y | x_i)}{\int_\nu \prod_{i \in S_y} p(\nu_y | x_i) d\nu}. \tag{4}$$

Because $p(\nu_y | x_i)$ is Gaussian, the resulting product is too:

$$\nu_y | S_y \sim \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\sigma}_y^2 \mathbf{I}) \text{ with}$$
$$\boldsymbol{\sigma}_y^2 = \left(\sum_{i \in S_y} \hat{\boldsymbol{\sigma}}_{x_i}^{-2} \right)^{-1} \text{ and } \boldsymbol{\mu}_y = \boldsymbol{\sigma}_y^2 \circ \left(\sum_{i \in S_y} \hat{\boldsymbol{\sigma}}_{x_i}^{-2} \circ \boldsymbol{\mu}_{x_i} \right),$$

where $\hat{\boldsymbol{\sigma}}_{x_i}^2 = \boldsymbol{\sigma}_{x_i}^2 + \boldsymbol{\sigma}_\epsilon^2$ and $\circ$ denotes the Hadamard product. Essentially, the prototype is a confidence-weighted average of the support instances.

2.2 Prediction and approximate inference

We assume a softmax prediction for a query embedding, z:

$$p(y | z, S) \propto \mathcal{N}(z; \boldsymbol{\mu}_y, \hat{\boldsymbol{\sigma}}_y^2 \mathbf{I}) \tag{5}$$

with $\hat{\boldsymbol{\sigma}}_y^2 = \boldsymbol{\sigma}_y^2 + \boldsymbol{\sigma}_\epsilon^2$ as before, yielding the class posterior for query x:

$$p(y | x, S) = \int_z \mathcal{N}(z; \boldsymbol{\mu}_x, \boldsymbol{\sigma}_x^2 \mathbf{I}) \, \frac{\mathcal{N}(z; \boldsymbol{\mu}_y, \hat{\boldsymbol{\sigma}}_y^2 \mathbf{I})}{\sum_c \mathcal{N}(z; \boldsymbol{\mu}_c, \hat{\boldsymbol{\sigma}}_c^2 \mathbf{I})} \, dz. \tag{6}$$

The class distribution is equivalent to that produced by the deterministic PN as $\boldsymbol{\sigma}_x^2 \to \mathbf{0}$ when $\boldsymbol{\sigma}_y^2 = \boldsymbol{\sigma}_{y'}^2$ for all class pairs (y, y'). However, in the general case, the integral has no closed form solution; thus, we must sample to approximate $p(y|x, S)$, both for training and evaluation. We employ two samplers, which we refer to as *naïve* and *intersection*.

2.2.1 Naïve sampling

A direct approach to approximating the class posterior is to express Equation 2 as an expectation, $\mathbb{E}_{z \sim p(z|x)} [p(y|z, S)]$, and to replace the expectation with the average over a set of samples. We utilize the reparameterization trick of Kingma and Welling [9] to train the model. Although this is the simplest approach, it is sample-inefficient during training, and when the number of samples is reduced, performance is impacted.

2.2.2 Intersection sampling

In Equation 6, the product of Gaussian densities in the numerator can be rewritten:

$$\mathcal{N}\left(z; \boldsymbol{\mu}_x, \boldsymbol{\sigma}_x^2 \mathbf{I}\right) \, \mathcal{N}\left(z; \boldsymbol{\mu}_y, \hat{\boldsymbol{\sigma}}_y^2 \mathbf{I}\right) = \mathcal{N}\left(z; \boldsymbol{\mu}_{xy}, \boldsymbol{\sigma}_{xy}^2 \mathbf{I}\right) \, \mathcal{N}\left(\boldsymbol{\mu}_x; \boldsymbol{\mu}_y, (\boldsymbol{\sigma}_x^2 + \hat{\boldsymbol{\sigma}}_y^2) \mathbf{I}\right), \tag{7}$$

where

$$\boldsymbol{\sigma}_{xy}^2 = (\boldsymbol{\sigma}_x^{-2} + \hat{\boldsymbol{\sigma}}_y^{-2})^{-1} \text{ and}$$
$$\boldsymbol{\mu}_{xy} = \boldsymbol{\sigma}_{xy}^2 \circ (\boldsymbol{\sigma}_x^{-2} \circ \boldsymbol{\mu}_x + \hat{\boldsymbol{\sigma}}_y^{-2} \circ \boldsymbol{\mu}_y).$$

Substituting Equation 7 into Equation 6,

$$p(y|x,S) = \mathcal{N}\left(\boldsymbol{\mu}_x; \boldsymbol{\mu}_y, (\boldsymbol{\sigma}_x^2 + \hat{\boldsymbol{\sigma}}_y^2)\mathbf{I}\right) \mathbb{E}_{z\sim\mathcal{N}\left(\boldsymbol{\mu}_{xy}, \boldsymbol{\sigma}_{xy}^2\mathbf{I}\right)} \left[\sum_c \mathcal{N}(z; \boldsymbol{\mu}_c, \hat{\boldsymbol{\sigma}}_c^2\mathbf{I})\right]^{-1}. \tag{8}$$

Approximating the expectation with samples from $\mathcal{N}\left(\boldsymbol{\mu}_{xy}, \boldsymbol{\sigma}_{xy}^2\mathbf{I}\right)$, we obtain a sampler that focuses on the intersection of the input distribution and a given class distribution. During training with a cross entropy loss, we need only sample for the known (target) class y. Compared to naïve sampling, we found this scheme to be more robust and significantly more sample efficient, requiring only a *single* sample to train effectively.

3 Experimental Results

We assess the SPE when trained on either the naïve or intersection sampler; in both cases, we evaluate using the naïve sampler with 200 samples. For details regarding network architectures and hyperparameters, see Appendix A, and for simulation details, including the choice of initialization for $\boldsymbol{\sigma}_c^2$, see Appendix B.

3.1 Synthetic color-orientation data set

Using a synthetic image data set, we demonstrate that the SPE can capture the generative structure of a domain and provide sensible estimates of uncertainty. The data set consists of 64×64 pixel images with a well-defined class structure. The four classes in our domain are distinguished by orientation, color, or both (Figure 2). Class-conditional generative parameters—orientation and color—are sampled from an isotropic Gaussian distribution. (The isotropy of these qualitatively different dimensions comes from the fact that both can be mapped as directional quantities.) Class overlap on color and orientation dimensions is symmetric. We tuned the variance such that there would be significant overlap between classes in order to elicit embeddings with increased uncertainty in overlapping regions. Full details of the synthetic data set can be found in Appendix A.2.

We trained a two-dimensional, intersection-sampling SPE on samples from this domain, using two instances per class to form prototypes. Classification accuracy of held-out samples is approximately 86%. Accounting for class overlap, a Bayes optimal classifier has an accuracy of approximately 87%. For visualization, we selected 25 examples by interpolating between the class means. The examples and their embeddings are shown in the left and right panels of Figure 3. The network has captured the structure of the domain by disentangling the two factors of variation. To identify the input array and the embedding space, the input array must be mirrored along the horizontal axis. The embedding variance encodes label ambiguity. Instances half way between two classes on one dimension have maximal variance along that dimension. Label ambiguity is one type of uncertainty. An equally important source of uncertainty comes from noisy or out-of-distribution (OOD) inputs. We examined OOD inputs generated in two different ways. In the left panel of Figure 4, we show the consequence of adding pixel hue noise to the four class centroids. Only one of these centroids is shown along the

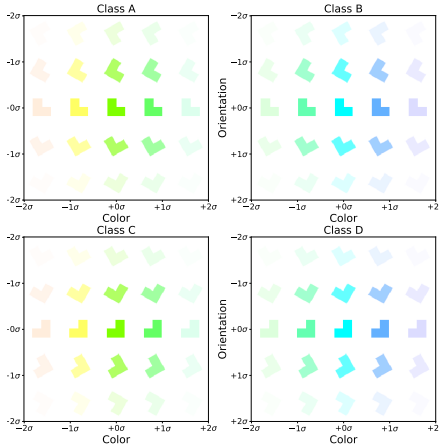

Figure 2: Samples from the four classes in our synthetic data set. In each plot, class means are shown at the center, along with samples spanning ± 2 standard deviations in both orientation and color. A sample's transparency is set according to its class-conditional likelihood. Both dimensions can be coded as directional variables. The class centroids on each dimension are 90° apart with standard deviation 22.5°.

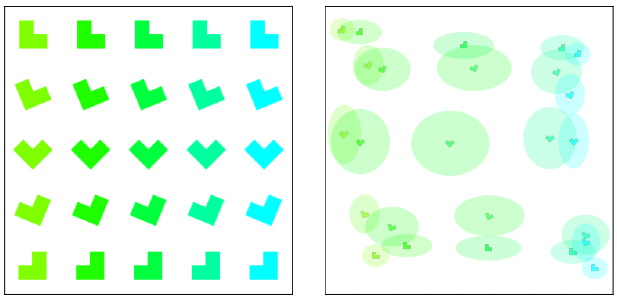

Figure 3: Synthetic data set: (left) A set of examples, with the four class centroids located in the corners and examples between formed via linear interpolation in the generative space. (right) The 2D stochastic prototype embedding for the examples. The shape is plotted at the mean of $p(z|x)$, and the outlines of the ovals represent equi-probability contours at 0.4 standard deviations.

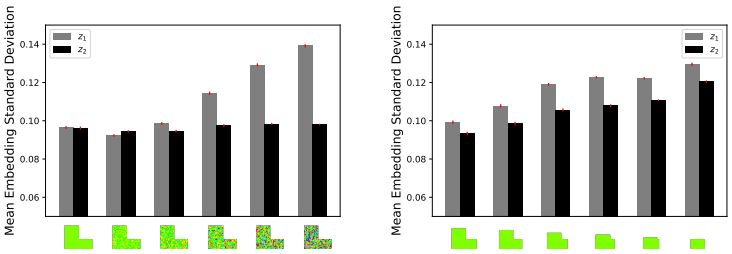

Figure 4: Synthetic data set: (left) uncertainty on the two embedding dimensions as the hue becomes more difficult to discern; (right) uncertainty on the two embedding dimensions as the orientation becomes more difficult to discern.

abscissa, but all four are used to make the graph, with many samples per noise level. The grey and black bars in the graph indicate variance on the vertical and horizontal dimensions of the embedding space, respectively. As pixel hue noise increases, uncertainty in the color grows but uncertainty in the orientation does not. In the right panel of Figure 4, we show the consequence of shortening the length of the legs of the shape. Shortening the legs removes cues that can be used both for determining both color and orientation. As a result, the uncertainty grows on both dimensions.

3.2 Quantitative evaluation

We evaluate the SPE on two tasks: few-shot learning classification using Omniglot [11] and a many-classes problem with multi-digit MNIST images [15].

3.2.1 Omniglot

The Omniglot data set contains images of labeled, handwritten characters from diverse alphabets. Omniglot is one of the standard data sets for comparing methods in the few-shot learning literature. The data set contains 1623 unique characters, each with 20 instances. Following Snell et al. [20], we augment the original classes with all $90°$ rotations, resulting in 6492 total classes, and we use the same train, validation, and test splits. Each grayscale image is resized from 32×32 to 28×28. We trained and evaluated deterministic Prototypical Networks, naïve-sampling SPEs, and intersection-sampling SPEs. Each is trained episodically, where a training episode contains 60 randomly sampled classes and 5 query instances per class, and test episodes contain 15 query instances per class.

We first compare the effectiveness of using naïve and intersection samplers during SPE training of Omniglot. We vary both the sampler and the number of samples drawn per training query, denoted

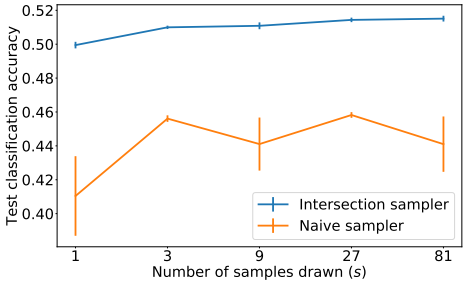

Figure 5: Test classification accuracy as a function of number of training samples per query instance for a naïve-sampling and intersection-sampling 2D SPE on a 1-shot, 20-class Omniglot task. Performance is a mean over 5 replications of running the model, showing ± 1 standard error of the mean.

Table 1: Test classification accuracy (%) on Omniglot with both 2D and 64D embeddings comparing the PN and the SPE with intersection sampling (1 sample per trial). Performance for PN and SPE is a mean over 1000 random test episodes, showing ± 1 standard error of the mean.

2D	1-SHOT, 5-CLASS	5-SHOT, 5-CLASS	1-SHOT, 20-CLASS	5-SHOT, 20-CLASS
PN	75.7 ± 0.4	$\mathbf{82.6 \pm 0.3}$	45.0 ± 0.2	$\mathbf{55.9 \pm 0.2}$
SPE	$\mathbf{76.9 \pm 0.4}$	$\mathbf{82.3 \pm 0.3}$	$\mathbf{49.7 \pm 0.2}$	55.3 ± 0.2
64D	1-SHOT, 5-CLASS	5-SHOT, 5-CLASS	1-SHOT, 20-CLASS	5-SHOT, 20-CLASS
PN	$\mathbf{98.46 \pm 0.09}$	$\mathbf{99.55 \pm 0.03}$	$\mathbf{94.87 \pm 0.08}$	$\mathbf{98.63 \pm 0.03}$
SPE	$\mathbf{98.53 \pm 0.08}$	$\mathbf{99.52 \pm 0.03}$	$\mathbf{94.93 \pm 0.08}$	98.56 ± 0.02

by s. We evaluate in a 1-shot 20-class setting, where shot refers to the number of support examples used to compute a prototype, and class refers to the number of candidate test classes per episode. Figure 5 shows test classification accuracy as the number of samples drawn per training trial (s) increases. As we previously claimed, the intersection-sampling SPE is much more sample-efficient and systematically outperforms the naive-sampling SPE. Furthermore, the intersection-sampling SPE is more robust and less susceptible to poor initialization or uninformative samples. The trend in Figure 5 is consistent across simulations; thus, we present only intersection-sampling SPE results henceforth, and all SPEs are trained with a single sample ($s = 1$) per query. This decision causes the SPE to be on par with the PN in time and space requirements, even though using more samples may have boosted classification accuracy.

Table 1 presents results for simulations with Omniglot comparing the deterministic PN to the SPE. For both 2- and 64-dimension embeddings, the SPE either matches or outperforms the PN, a state-of-the-art algorithm in few-shot learning. In the 64D embedding, both methods are close to ceiling. In the 2D embedding, the SPE particularly shines in the 1-shot tests, perhaps because, relative to the 5-shot tests, performance is further from ceiling. However, another possibility is that the role of uncertainty is greater when only one instance is used to form the prototype. When 5 instances are used, the uncertainty shrinks significantly, causing the SPE to behave more like its deterministic sibling. To emphasize, the improved performance of the SPE does not incur much of a computational cost in training. And by providing an estimate of uncertainty associated with embedded instances, the SPE offers the possibility of detecting OOD samples and informing downstream systems that operate on the embedding.

3.2.2 N-digit MNIST

The N-digit MNIST data set was proposed to evaluate HIB [15]; it is formed by horizontal concatenation of N MNIST digit images. The resulting images are $28 \times 28N$. To compare with HIB, we study 2- and 3-digit MNIST, and use a network architecture identical to that in [15] (see Appendix A.3). Oh et al. [15] split the data into a training set with 70% of the total classes, a *seen* test set, and an *unseen* test set. For 2-digit MNIST, the seen test set has the same 70 of 100 classes as the training set and the unseen test set has the remaining 30 classes. For 3-digit MNIST, the training set has 700 classes, the seen and unseen test sets each have a sample of 100 of the 700 seen or 300 unseen classes, respectively. We use the same train and test data splits as [15], but we further divide the training split to include a validation set for early stopping. Following [15], each digit in each of the training and validation images is randomly occluded, independently, with a probability of 0.2. In evaluating the model, two versions of the test data are used: *clean* images, which contain no occlusions, and *corrupt* images, which contain random occlusions in every digit of the image. Appendix C describes the procedure for synthesizing occlusions, and Figure 6 shows sample images in which each digit has an occlusion.

Figure 7 is a 2D embedding learned by the SPE on the 2-digit MNIST test set. Each number in the embedding represents the class label, for example, 71, located in the lower left of the embedding, is the class in which the first of the two MNIST digits is a 7 and the second is a 1. The location of a class in the space corresponds to the mean of its prototype. The SPE learned an incredibly robust

Figure 6: Examples of occluded 2-digit sequences. Occlusion is based on random rectangles that black out portions of each digit.

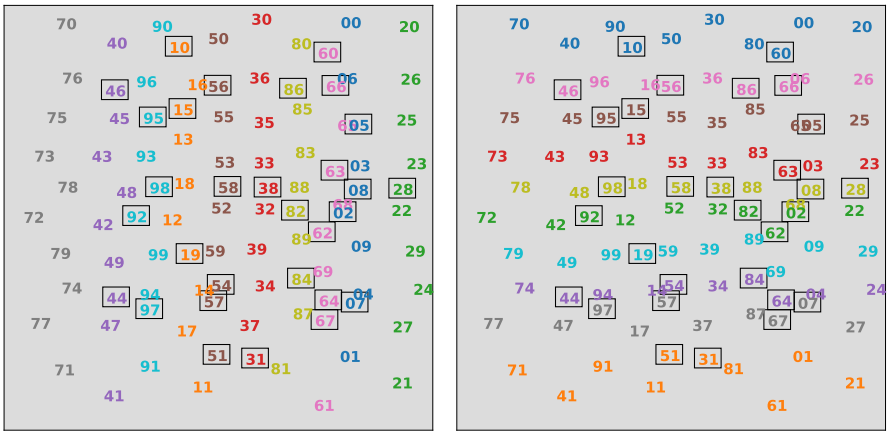

Figure 7: Two-dimensional embedding learned by the SPE on the 2-digit MNIST test set. A class is specified by a two-digit number. In both figures, the location of the class corresponds to the mean of the prototype in the test set using 140 support instances. The digits surrounded by a black border are classes that were not seen during training. (left) The prototypes are colored according to the first digit of the class. (right) The prototypes are colored according to the second digit of the class.

factorial representation, in which the horizontal dimension represents the first digit of a class and the vertical dimension represents the second digit. In the left plot, each class is colored according to the first digit. In the right plot is the same embedding, but each prototype is colored according to the second digit. A black bounding box indicates the unseen test classes, classes not presented during training. Impressively, the unseen test classes are embedded in exactly the positions where they belong, indicating that the SPE is able to discover inter-class structure that generalizes to classes it has never seen during training. Furthermore, the embedding has captured inter-class similarity structure by placing visually similar digits close to one another. For example, on both the vertical and horizontal bands, nines (teal) and fours (purple) are adjacent, and fives (brown) and threes (red) are adjacent. The adjacency relationships vary a bit from one dimension of the mapping to the other; for example, sixes (pink) are adjacent to eights (yellow) and zeros (blue) in the vertical bands, but adjacent to fives (brown) and zeros in the horizontal bands. Although the HIB is able to discover a similar structure along one dimension, the second dimension is more entangled, providing qualitative evidence that the SPE learns a more robust representation. Additionally, embeddings for the unseen class are not shown for HIB. The ability to sensibly embed novel classes is essential for any model that will be used for open-set recognition or few-shot learning.

Table 2 compares 2- and 3-digit MNIST classification performance for the contrastive-pair loss (the deterministic form of HIB), HIB, PN (the deterministic form of SPE), and the SPE. Following Oh et al., we tested on clean query instances but considered support sets made either entirely of clean or corrupted digits, and on either seen or unseen classes. The contrastive-pair and HIB results are from [15]. (Results on unseen classes were not published, but a spreadsheet provided by the authors includes results for HIB on unseen classes.)

Clean data results. Comparing the two deterministic models, PN outperforms contrastive-pairs (mean across conditions 87.1% versus 79.1%, with PN best on 4 of 4 conditions), consistent with prior work suggesting that cluster-based losses yield superior embeddings to those based on pairwise constraints. Turning to the stochastic models, SPE reliably outperforms HIB (mean 89.1% vs. 79.8% on seen classes, 87.0% vs. 74.9% on unseen classes) and also PN (89.1% vs. 87.1% on seen, 87.0% vs. 82.8% on unseen). Whereas SPE is a discriminative model with a specified classification procedure, Oh et al. [15] had freedom to design one. They use all available data—roughly 140 examples per class—and perform leave-one-out 5-nearest-neighbor classification. To be consistent with our episodic testing procedure, the SPE uses only 50 support instances per class to form prototypes for 2-digit MNIST and 20 support instances for 3-digit MNIST. It is therefore impressive that the SPE, based on a single stored prototype and less than 1/3 the labeled data, can outperform a memory-based nonparametric method that is able to model arbitrary distributions in latent space.

Corrupted data results. For experiments with corrupted test data, Oh et al. [15] focus on the case where the support set contains only occluded imagery and the query set contains only clean. (The

Table 2: Test classification accuracy (%) on clean 2- and 3-digit MNIST for clean (top) and corrupted (bottom) support sets. N: number of digits in each image; D: dimensionality of the embedding. **Boldface** indicates best result in a column. HIB and Contrastive-Pairs results from [15]. PN is our implementation of Prototypical Networks [20]. SPE is our model.

CLEAN	SEEN TEST CLASSES					UNSEEN TEST CLASSES				
	N=2		N=3			N=2		N=3		
	D=2	D=3	D=2	D=3	MEAN	D=2	D=3	D=2	D=3	MEAN
CONTRASTIVE	88.2	95.0	65.8	87.3	84.1	85.5	84.8	59.0	85.5	78.7
HIB	87.9	**95.2**	65.0	87.3	83.8	87.3	**91.0**	64.4	**88.2**	82.7
PN	91.3	95.1	71.8	**90.4**	87.1	83.1	89.2	70.8	**88.2**	82.8
SPE (OURS)	**92.4**	93.8	**81.7**	88.5	**89.1**	**88.7**	90.2	**81.0**	**88.2**	**87.0**

CORRUPT	SEEN TEST CLASSES					UNSEEN TEST CLASSES				
	N=2		N=3			N=2		N=3		
	D=2	D=3	D=2	D=3	MEAN	D=2	D=3	D=2	D=3	MEAN
CONTRASTIVE	76.2	92.2	49.5	77.6	73.9	76.5	73.3	42.6	73.2	66.4
HIB	81.6	**94.3**	54.0	81.2	77.8	80.8	86.7	53.9	81.2	75.7
PN	75.2	93.8	46.2	79.7	73.7	72.0	86.3	43.2	77.3	69.7
SPE (OURS)	**91.1**	93.9	**77.5**	**87.5**	**87.5**	**86.4**	**87.1**	**76.0**	**85.5**	**83.8**

experiments are described in their article as having a clean support and a corrupt query, but this description is incorrect, as we verified from code provided by the authors.) In this situation, SPE will attempt to infer prototypes from widely dispersed instances due to occlusions. Assuming successful prototype reconstruction in the embedding space, clean query instances should be classified with high accuracy. The SPE outperforms contrastive-pairs, HIB, and PN across all experimental configurations, for both the seen and unseen test classes. The mean improvements are substantive: 35% and 36% reductions in error rate for SPE vs. HIB on seen and unseen classes, respectively. We attribute SPE's exceptional performance to formation of class prototypes via a confidence-weighted average, expressed as a product distribution on the densities. As a result, corrupted instance will have a smaller influence on the prototype.

4 Discussion and Conclusions

Our Stochastic Prototype Embedding (SPE) method outperforms a state-of-the-art deterministic method, the Prototypical Net (PN), as well as a recent state-of-the-art stochastic method, the Hedged Instance Embedding (HIB). Beyond its performance, SPE has no hand tuned parameters, whereas HIB has constant β that determines characteristics of an information bottleneck (i.e., how much of the input entropy is retained in the embedding). Although one could simply set $\beta = 0$, doing so would encourage the net to perform like a softmax classifier and discard all information about inter-class similarity. Such similarities are essential in order to obtain beautiful embeddings like Figures 3 and 7.

We proposed an intersection sampler to train the SPE efficiently, which makes the SPE as time and space efficient for training as its deterministic sibling, and more efficient for training than HIB which relies on about 8 samples per item. (Our evaluation method for SPE presently involves drawing 200 samples from the naive sampler, though this conservative decision was arbitrary and not tuned.) An unexpected virtue of our method, as well as other methods with uncertainty represented as a diagonal covariance matrix, is that it has a bias toward disentangling, i.e., forcing dimensions of variation to align with the principle axes in latent space.

Stochastic embedding methods are an active field of research. In recent months, several new articles have appeared that make tangential contact with our work. Infinite Mixture Prototypes [1] use Bayesian nonparametric methods to determine a mixture distribution for the prototype, though they do not examine uncertainty in embedding space or leverage the embedding to handle noisy inputs and noisy labels. Deep Variational Transfer [2] is a generative form of our discriminative model, which has the drawback that it needs to model the input distribution. Authors of this work used their approach for covariate shift, a somewhat different problem than SPE and HIB have tackled.

References

[1] Allen, K. R., Shelhamer, E., Shin, H., and Tenenbaum, J. B. (2019). Infinite Mixture Prototypes for Few-Shot Learning. *arXiv e-prints 1902.04552 cs.LG*.

[2] Belhaj, M., Protopapas, P., and Pan, W. (2018). Deep Variational Transfer: Transfer Learning through Semi-supervised Deep Generative Models. *arXiv e-prints 1812.03123 cs.LG*.

[3] Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a Similarity Metric Discriminatively, with Application to Face Verification. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[4] Edwards, H. and Storkey, A. (2017). Towards a Neural Statistician. In *International Conference on Learning Representations*.

[5] Finn, C., Abbeel, P., and Levine, S. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135.

[6] Fort, S. (2017). Gaussian Prototypical Networks for Few-Shot Learning on Omniglot. In *Second Workshop on Bayesian Deep Learning (NIPS 2017)*.

[7] Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality Reduction by Learning an Invariant Mapping. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1735–1742.

[8] Karaletsos, T., Belongie, S., and Rätsch, G. (2016). Bayesian Representation Learning with Oracle Constraints. In *International Conference on Learning Representations*.

[9] Kingma, D. P. and Welling, M. (2014). Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*.

[10] Koch, G., Zemel, R., and Salakhutdinov, R. (2015). Siamese Neural Networks for One-Shot Image Recognition. In *ICML Deep Learning Workshop*, volume 2.

[11] Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-Level Concept Learning through Probabilistic Program Induction. *Science*, 350(6266):1332–1338.

[12] Li, W., Zhao, R., Xiao, T., and Wang, X. (2014). DeepReID: Deep Filter Pairing Neural Network for Person Re-identification. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[13] Liu, J., Deng, Y., Bai, T., Wei, Z., and Huang, C. (2015). Targeting Ultimate Accuracy: Face Recognition via Deep Embedding. *arXiv e-prints 1506.07310 cs.CV*.

[14] Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. (2018). A Simple Neural Attentive Meta-Learner. In *International Conference on Learning Representations*.

[15] Oh, S. J., Gallagher, A. C., Murphy, K. P., Schroff, F., Pan, J., and Roth, J. (2019). Modeling Uncertainty with Hedged Instance Embeddings. In *International Conference on Learning Representations*.

[16] Ridgeway, K. and Mozer, M. C. (2018). Learning Deep Disentangled Embeddings With the F-Statistic Loss. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 185–194. Curran Associates, Inc.

[17] Rippel, O., Paluri, M., Dollar, P., and Bourdev, L. (2016). Metric Learning with Adaptive Density Discrimination. In *International Conference on Learning Representations*.

[18] Schroff, F., Kalenichenko, D., and Philbin, J. (2015). FaceNet: A Unified Embedding for Face Recognition and Clustering. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[19] Scott, T., Ridgeway, K., and Mozer, M. C. (2018). Adapted Deep Embeddings: A Synthesis of Methods for k-Shot Inductive Transfer Learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 76–85. Curran Associates, Inc.

[20] Snell, J., Swersky, K., and Zemel, R. (2017). Prototypical Networks for Few-shot Learning. In *Advances in Neural Information Processing Systems 31*, pages 4077–4087.

[21] Song, H. O., Jegelka, S., Rathod, V., and Murphy, K. (2017). Deep Metric Learning via Facility Location. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2206–2214.

[22] Song, H. O., Xiang, Y., Jegelka, S., and Savarese, S. (2016). Deep Metric Learning via Lifted Structured Feature Embedding. In *IEEE Conference on Computer Vision and Pattern Recognition*.

[23] Triantafillou, E., Zemel, R., and Urtasun, R. (2017). Few-Shot Learning Through an Information Retrieval Lens. In *Advances in Neural Information Processing Systems 31*, pages 2255–2265.

[24] Ustinova, E. and Lempitsky, V. (2016). Learning Deep Embeddings with Histogram Loss. In *Advances in Neural Information Processing Systems 30*, pages 4170–4178.

[25] Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. (2016). Matching Networks for One Shot Learning. In *Advances in Neural Information Processing Systems 30*, pages 3630–3638.

[26] Wang, J., Zhou, F., Wen, S., Liu, X., and Lin, Y. (2017). Deep Metric Learning with Angular Loss. In *IEEE International Conference on Computer Vision*, pages 2612–2620.

[27] Yi, D., Lei, Z., Liao, S., and Li, S. Z. (2014). Deep Metric Learning for Person Re-identification. In *International Conference on Pattern Recognition*, pages 34–39.

[28] Zheng, L., Shen, L., Tian, L., Wang, S., Wang, J., and Tian, Q. (2015). Scalable Person Re-identification: A Benchmark. In *IEEE International Conference on Computer Vision*.

A Network Architectures and Hyperparameters

A.1 Omniglot

For all Omniglot experiments, the network consisted of four convolutional blocks. The first three blocks had a convolutional layer with 64 filters, a 3×3 kernel, zero-padding of length 1, and a stride of 1, followed by a batch normalization layer, ReLU activation, and 2×2 max-pooling. The fourth and final block had a convolutional layer with $2d$ filters, a 3×3 kernel, zero-padding of length 1, and a stride of 1, followed by 2×2 max-pooling, where d represents the dimensionality of the embedding space. The flattened output of the network is a vector of length $2d$, where the first d elements were considered the mean of the Gaussian distribution and the remaining d elements were the diagonal covariance entries. The weights were initialized using He initialization and the biases with the following uniform distribution: $\mathcal{U}(-\frac{1}{\sqrt{\text{fan in}}}, \frac{1}{\sqrt{\text{fan in}}})$.

All Omniglot models were trained with an initial learning rate of 0.001 which was cut in half every 50 epochs. The models were stopped early using a patience parameter when performance on the validation set no longer increased.

A.2 Synthetic data

The images in the synthetic data set are 64×64 pixels in size. For orientation, we chose class centers at $90°$ and $180°$, with a standard deviation of $30°$. For color, we manipulated the hue and kept value and saturation constant. Like orientation, hue is a circular quantity. If hue ranges from 0 to 360 degrees, we chose color class centers and standard deviation in the same way as orientation. Additionally, we add noise to a minority (15%) of the images used to train the model. For these, we add Gaussian noise to the hue of each pixel inside the shape. The standard deviation of the hue noise was chosen uniformly between $18°$ and $54°$. We also added noise to the leg lengths of the L shapes. The leg length was chosen uniformly between 10% and 98% of its original length. See Figure 4 for some examples.

The network followed an architecture similar to the one we used for Omniglot, except that we added two additional blocks of convolution, batch normalization, ReLU, and max-pooling because the images are larger. We used 2 instances per class to form prototypes and 8 samples per query instance during training. We used a learning rate of 0.0001 and the models were stopped early using a patience parameter when performance on the validation set no longer increased.

A.3 N-Digit MNIST

For all N-digit MNIST experiments, we constructed an architecture which we believe to be identical to that used for HIB MNIST experiments [15], based on code provided by the authors of [15]. The network consisted of two convolutional blocks followed by two fully-connected layers. The convolutional blocks each contained a convolutional layer, followed by an ReLU activation, and 2×2 max-pooling. The first convolutional layer had 6 filters, a 5×5 kernel, zero-padding of length 2, and a stride of 1. The second convolutional layer was identical to the first, but had 16 filters instead of 6. The output of the second convolutional block was flattened, passed through a fully-connected layer with 120 units, an ReLU activation, and a final fully-connected layer with $2d$ units, where d represents the dimensionality of the embedding space. Like the Omniglot architectures, the first d entries in the output vector are treated as the mean and the remaining d elements as the diagonal covariance entries. The weights were initialized using a Xavier uniform initialization and biases were initialized to zero.

The PN and SPE are trained episodically with all performance results in the main article measured as the mean over 1000 random test episodes.

All N-digit MNIST models were trained with an initial learning rate of 0.001 which was cut in half every 50 epochs. The models were stopped early using a patience parameter when performance on the validation set no longer increased. For 2-digit MNIST, each episode in training and evaluation contained all 70 classes and 50 support instances per class. For 3-digit MNIST, each episode in training and evaluation contained 100 classes and 20 support instances per class.

B Simulation Details

For all SPE models,

$$\boldsymbol{\sigma}_{\epsilon}^2 = \mathrm{softplus}\left(\boldsymbol{\epsilon}\right),$$

where $\boldsymbol{\epsilon}$ is a trainable parameter. We initialize $\boldsymbol{\epsilon}$ using the following prescription:

$$\boldsymbol{\epsilon} = |S|\boldsymbol{\epsilon}_0^{2/d},$$

where $|S|$ is the number of support examples per episode during training and d is the dimensionality of the embedding. We chose this prescription for two reasons: (1) as the number of support examples increases, the variance of the prototype distribution approaches zero, so scaling linearly by $|S|$ tends to provide a stronger training signal early on, and (2) the amount of noise in the projection of an embedding should scale with the dimensionality of the embedding space as to maintain unit-volume. All models used $\boldsymbol{\epsilon}_0 = 0.01$.

It should also be noted that there is no constraint for $\boldsymbol{\sigma}_x^2 > 0$, since $\boldsymbol{\sigma}_x^2$ is a direct output of a deep network. To ensure a positive value, $\boldsymbol{\sigma}_x^2 \leftarrow \mathrm{softplus}(\boldsymbol{\sigma}_x^2)$.

C N-Digit MNIST Corruption

The algorithm for applying corruption was identical to the scheme used in [15]. A random rectangular-sized occlusion of black pixels was determined by first sampling a patch width, L_x, and patch height, L_y, from a uniform distribution, $L_x, L_y \sim \mathcal{U}(0, 28)$, and then sampling the top-left corner coordinates, $TL_x \sim \mathcal{U}(0, 28 - L_x)$, $TL_y \sim \mathcal{U}(0, 28 - L_y)$. This resulted in an occlusion of area $L_x \times L_y$. Note that if $L_x = 0$ or $L_y = 0$, the image was left unoccluded. The training and validation sets for N-digit MNIST corrupted each digit of each image independently with a probability of 0.2. The test sets (both seen and unseen) were either clean, in which all digits were unoccluded, or corrupted, in which all digits were randomly occluded. During training and validation of the model, both the support and query sets drawn in an episode could contain corrupted images. During testing, the query set always contained clean imagery.

D SPE Variants

We assumed only diagonal covariance matrices in this work. Switching to a full covariance matrix would require matrix inversion, which is ordinarily infeasible, but because one purpose of deep embeddings is visualization, there may be interesting cases involving 2D embeddings where the cost of inversion is trivial.