Deep Metric Learning via Lifted Structured Feature Embedding

Hyun Oh Song
Stanford University
hsong@cs.stanford.edu

Yu Xiang
Stanford University
yuxiang@cs.stanford.edu

Stefanie Jegelka
MIT
stefje@csail.mit.edu

Silvio Savarese
Stanford University
ssilvio@stanford.edu

Abstract

Learning the distance metric between pairs of examples is of great importance for learning and visual recognition. With the remarkable success from the state of the art convolutional neural networks, recent works [1, 31] have shown promising results on discriminatively training the networks to learn semantic feature embeddings where similar examples are mapped close to each other and dissimilar examples are mapped farther apart. In this paper, we describe an algorithm for taking full advantage of the training batches in the neural network training by lifting the vector of pairwise distances within the batch to the matrix of pairwise distances. This step enables the algorithm to learn the state of the art feature embedding by optimizing a novel structured prediction objective on the lifted problem. Additionally, we collected Stanford Online Products dataset: 120k images of 23k classes of online products for metric learning. Our experiments on the CUB-200-2011 [37], CARS196 [19], and Stanford Online Products datasets demonstrate significant improvement over existing deep feature embedding methods on all experimented embedding sizes with the GoogLeNet [33] network. The source code and the dataset are available at: https://github.com/rksltnl/Deep-Metric-Learning-CVPR16.

1. Introduction

Comparing and measuring similarities between pairs of examples is a core requirement for learning and visual competence. Being able to first measure how similar a given pair of examples are makes the following learning problems a lot simpler. Given such a similarity function, classification tasks could be simply reduced to the nearest neighbor problem with the given similarity measure, and clustering tasks would be made easier given the similarity matrix. In this regard, metric learning [13, 39, 34] and dimensionality reduction [18, 7, 29, 2] techniques aim at learning semantic distance measures and embeddings such that similar input objects are mapped to nearby points on a manifold and dissimilar objects are mapped apart from each other.

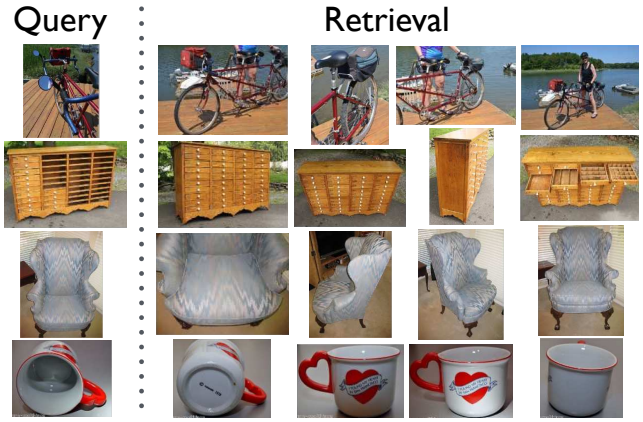


Figure 1: Example retrieval results on our *Stanford Online Products* dataset using the proposed embedding. The images in the first column are the query images.

Furthermore, the problem of *extreme classification* [6, 26] with enormous number of categories has recently attracted a lot of attention in the learning community. In this setting, two major problems arise which renders conventional classification approaches practically obsolete. First, algorithms with the learning and inference complexity linear in the number of classes become impractical. Second, the availability of training data per class becomes very scarce. In contrast to conventional classification approaches, metric learning becomes a very appealing technique in this regime because of its ability to learn the general concept of distance metrics (as opposed to category specific concepts) and its compatibility with efficient nearest neighbor inference on the learned metric space.

With the remarkable success from the state of the art convolutional neural networks [20, 33], recent works [1, 31] discriminatively train neural network to directly learn the the non-linear mapping function from the input image to a lower dimensional embedding given the input label annotations. In high level, these embeddings are optimized to pull examples with different class labels apart from each other and push examples from the same classes closer to each other. One of the main advantages of these discriminatively

trained network models is that the network jointly learns the feature representation and semantically meaningful embeddings which are robust against intra-class variations.

However, the existing approaches [1, 31] cannot take full advantage of the training batches used during the mini batch stochastic gradient descent training of the networks [20, 33]. The existing approaches first take randomly sampled pairs or triplets to construct the training batches and compute the loss on the individual pairs or triplets within the batch. Our proposed method *lifts* the *vector* of pairwise distances ($O(m)$) within the batch to the *matrix* of pairwise distances ($O(m^2)$). Then we design a novel structured loss objective on the lifted problem. Our experiments show that the proposed method of learning the embedding with the structured loss objective on the lifted problem significantly outperforms existing methods on all the experimented embedding dimensions with the GoogLeNet [33] network.

We evaluate our methods on the CUB200-2011 [37], CARS196 [19], and *Stanford Online Products* dataset we collected. The *Stanford Online Products* has approximately $120k$ images and $23k$ classes of product photos from online e-commerce websites. To the best of our knowledge, the dataset is one of the largest publicly available dataset in terms of the number and the variety of classes. We plan to maintain and grow the dataset for the research community.

In similar spirit of general metric learning where the task is to learn a generic concept of similarity/distance, we construct our train and test split such that there is no intersection between the set of classes used for training versus testing. We show that the clustering quality (in terms of standard F_1 and NMI metrics [23]) and retrieval quality (in terms of standard Recall@K score) on images from previously unseen classes are significantly better when using the proposed embedding. Figure 1 shows some example retrieval results with the *Stanford Online Products* dataset using the proposed embedding. Although we experiment on clustering and retrieval tasks, the conceptual contribution of this paper - lifting a batch of examples into a dense pairwise matrix and defining a structured learning problem - is generally applicable to a variety of learning and recognition tasks where feature embedding is employed.

2. Related works

Our work is related to three lines of active research: (1) Deep metric learning for recognition, (2) Deep feature embedding with convolutional neural networks, and (3) Zero shot learning and ranking.

Deep metric learning: Bromley *et al*. [3] paved the way on deep metric learning and trained Siamese networks for signature verification. Chopra *et al*. [5] trained the network discriminatively for face verification. Chechik *et al*. [4] learn ranking function using triplet [39] loss. Qian *et al*. [27] uses precomputed [20] activation features and learns a feature embedding via distance metric for classification.

Deep feature embedding with state of the art convolutional neural networks: Bell *et al*. [1] learn embedding for visual search in interior design using contrastive [14] embedding, FaceNet [31] uses triplet [39] embedding to learn embedding on faces for face verification and recognition. Li *et al*. [22] learn a joint embedding shared by both 3D shapes and 2D images of objects. In contrast to the existing approaches above, our method computes a novel structured loss and the gradient on the lifted dense pairwise distance matrix to take full advantage of batches in SGD.

Zero shot learning and ranking: Frome *et al*., Socher *et al*., and Weston *et al*. [12, 32, 40] leverage text data to train visual ranking models and to constrain the visual predictions for zero shot learning. Wang *et al*. [38] learns to rank input triplet of data given human rater's rank ratings on each triplets and also released a triplet ranking dataset with 5,033 triplet examples [8]. However, the approach is not scalable with the size of the training data because it's very costly to obtain ranking annotations in contrast to multiclass labels (i.e., product name) and because the approach is limited to ranking the data in triplet form. Lampert *et al*. [21] does zero shot learning but with attributes (such as objects's color or shape) provided for both the train and the test data. On a related note, [24, 25, 28] do zero-shot learning for visual recognition but rely on the WordNet hierarchy for semantic information of the labels.

The paper is organized as follows. In section 3, we start with a brief review of recent state of the art deep learning based embedding methods [14, 31]. In section 4, we describe how we lift the problem and define a novel structured loss. In section 5 and 6, we describe the implementation details and the evaluation metrics. We present the experimental results and visualizations in section 7.

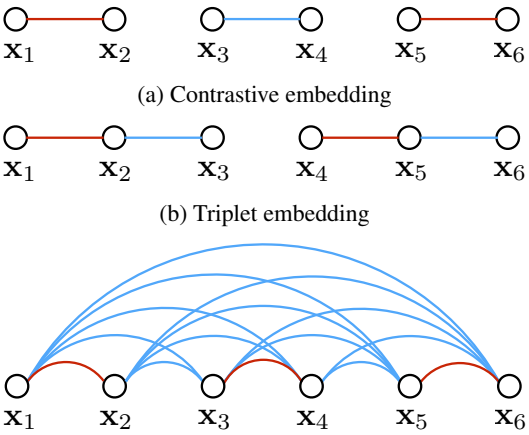
3. Review

In this section, we briefly review recent works on discriminatively training networks to learn semantic embeddings.

Contrastive embedding [14] is trained on the paired data $\{(\mathbf{x}_i, \mathbf{x}_j, y_{ij})\}$. The contrastive training minimizes the distance between a pair of examples with the same class label and penalizes the negative pair distances for being smaller than the margin α. The cost function is defined as,

$$J = \frac{1}{m} \sum_{(i,j)}^{m/2} y_{i,j} D_{i,j}^2 + (1 - y_{i,j}) \left[\alpha - D_{i,j}\right]_+^2 , \quad (1)$$

where m stands for the number of images in the batch, $f(\cdot)$ is the feature embedding output from the network,



(a) Contrastive embedding

(b) Triplet embedding

(c) Lifted structured embedding

Figure 2: Illustration for a training batch with six examples. Red edges and blue edges represent similar and dissimilar examples respectively. In contrast, our method explicitly takes into account all pair wise edges within the batch.

$D_{i,j} = ||f(\mathbf{x}_i) - f(\mathbf{x}_j)||_2$, and the label $y_{i,j} \in \{0, 1\}$ indicates whether a pair $(\mathbf{x}_i, \mathbf{x}_j)$ is from the same class or not. The $[\cdot]_+$ operation indicates the hinge function $\max(0, \cdot)$. Please refer to [14] for more details.

Triplet embedding [39, 31] is trained on the triplet data $\left\{\left(\mathbf{x}_a^{(i)}, \mathbf{x}_p^{(i)}, \mathbf{x}_n^{(i)}\right)\right\}$ where $\left(\mathbf{x}_a^{(i)}, \mathbf{x}_p^{(i)}\right)$ have the same class labels and $\left(\mathbf{x}_a^{(i)}, \mathbf{x}_n^{(i)}\right)$ have different class labels. The $\mathbf{x}_a^{(i)}$ term is referred to as an *anchor* of a triplet. Intuitively, the training process encourages the network to find an embedding where the distance between $\mathbf{x}_a^{(i)}$ and $\mathbf{x}_n^{(i)}$ is larger than the distance between $\mathbf{x}_a^{(i)}$ and $\mathbf{x}_p^{(i)}$ plus some margin α. The cost function is defined as,

$$J = \frac{3}{2m} \sum_{i}^{m/3} \left[D_{ia,ip}^2 - D_{ia,in}^2 + \alpha\right]_+, \tag{2}$$

where $D_{ia,ip} = ||f(\mathbf{x}_i^a) - f(\mathbf{x}_i^p)||$ and $D_{ia,in} = ||f(\mathbf{x}_i^a) - f(\mathbf{x}_i^n)||$. Please refer to [31, 39] for the complete details.

4. Deep metric learning via lifted structured feature embedding

We define a structured loss function based on all positive and negative pairs of samples in the training set:

$$J = \frac{1}{2|\widehat{\mathcal{P}}|} \sum_{(i,j)\in\widehat{\mathcal{P}}} \max\left(0,\ J_{i,j}\right)^2, \tag{3}$$

$$J_{i,j} = \max\left(\max_{(i,k)\in\widehat{\mathcal{N}}} \alpha - D_{i,k},\ \max_{(j,l)\in\widehat{\mathcal{N}}} \alpha - D_{j,l}\right) + D_{i,j}$$

where $\widehat{\mathcal{P}}$ is the set of positive pairs and $\widehat{\mathcal{N}}$ is the set of negative pairs in the training set. This function poses two computational challenges: (1) it is non-smooth, and (2) both evaluating it and computing the subgradient requires mining all pairs of examples several times.

We address these challenges in two ways: First, we optimize a smooth upper bound on the function instead. Second, as is common for large data sets, we use a stochastic approach. However, while previous work implements a stochastic gradient descent by drawing pairs or triplets of points uniformly at random [14, 1, 22], our approach deviates from those methods in two ways: (1) it biases the sample towards including "difficult" pairs, just like a subgradient of $J_{i,j}$ would use the close negative pairs [1]; (2) it makes use of the full information of the mini-batch that is sampled at a time, and not only the individual pairs.

Figures 2a and 2b illustrate a sample batch of size $m = 6$ for the contrastive and triplet embedding. Red edges in the illustration represent positive pairs (same class) and the blue edges represent negative pairs (different class) in the batch. In this illustration, it is important to note that adding extra vertices to the graph is a lot more costly than adding extra edges because adding vertices to the graph incurs extra I/O time and/or storage overhead.

To make full use of the batch, one key idea is to enhance the mini-batch optimization to use all $O(m^2)$ pairs in the batch, instead of $O(m)$ separate pairs. Figure 2c illustrates the concept of of transforming a training batch of examples to a fully connected dense matrix of pairwise distances. Given a batch of c-dimensional embedded features $X \in \mathbb{R}^{m\times c}$ and the column vector of squared norm of individual batch elements $\tilde{\mathbf{x}} = \left[||f(\mathbf{x}_1)||_2^2, \ldots, ||f(\mathbf{x}_m)||_2^2\right]^\intercal$, the dense pairwise squared distance matrix can be efficiently constructed by computing, $D^2 = \tilde{\mathbf{x}}\mathbf{1}^\intercal + \mathbf{1}\tilde{\mathbf{x}}^\intercal - 2XX^\intercal$, where $D_{ij}^2 = ||f(\mathbf{x}_i) - f(\mathbf{x}_j)||_2^2$. However, it is important to note that the negative edges induced between randomly sampled pairs carry limited information. Most likely, they are different from the much sharper, close ("difficult") neighbors that a full subgradient method would focus on.

Hence, we change our batch to be not completely random, but integrate elements of importance sampling. We sample a few positive pairs at random, and then actively add their difficult neighbors to the training mini-batch. This augmentation adds relevant information that a subgradient would use. Figure 3 illustrates the mining process for one positive pair in the batch, where for each image in a posi-

[1] Strictly speaking, this would be a subgradient replacing the nested max by a plus.

tive pair we find its close (hard) negative images. Note that our method allows mining the hard negatives from both the left and right image of a pair in contrast to the rigid triplet structure [31] where the negative is defined only with respect to the predefined anchor point. Indeed, the procedure of mining hard negative edges is equivalent to computing the loss augmented inference in structured prediction setting [35, 17]. Our loss augmented inference can be efficiently processed by first precomputing the pairwise batch squared distance matrix D^2.

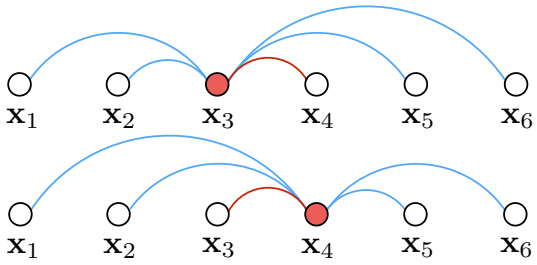


Figure 3: Hard negative edge is mined with respect to each left and right example per each positive pairs. In this illustration with 6 examples in the batch, both $\mathbf{x}_3$ and $\mathbf{x}_4$ independently compares against all other negative edges and mines the hardest negative edge.

Furthermore, mining the single hardest negative with nested max functions (eqn. 4) in practice causes the network to converge to a bad local optimum. Hence we optimize the following smooth upper bound $\tilde{J}(D(f(\mathbf{x})))$. Concretely, our loss function per each batch is defined as,

$$\tilde{J}_{i,j} = \log\left(\sum_{(i,k)\in\mathcal{N}} \exp\{\alpha - D_{i,k}\} + \sum_{(j,l)\in\mathcal{N}} \exp\{\alpha - D_{j,l}\}\right) + D_{i,j}$$

$$\tilde{J} = \frac{1}{2|\mathcal{P}|} \sum_{(i,j)\in\mathcal{P}} \max\left(0,\ \tilde{J}_{i,j}\right)^2, \tag{4}$$

where $\mathcal{P}$ denotes the set of positive pairs in the batch and $\mathcal{N}$ denotes the set of negative pairs in the batch. The back propagation gradients for the input feature embeddings can be derived as shown in algorithm 1, where the gradients with respect to the distances are,

$$\frac{\partial \tilde{J}}{\partial D_{i,j}} = \frac{1}{|\mathcal{P}|} \tilde{J}_{i,j}\ \mathbb{1}[\tilde{J}_{i,j} > 0] \tag{5}$$

$$\frac{\partial \tilde{J}}{\partial D_{i,k}} = \frac{1}{|\mathcal{P}|} \tilde{J}_{i,j}\ \mathbb{1}[\tilde{J}_{i,j} > 0]\ \frac{-\exp\{\alpha - D_{i,k}\}}{\exp\{\tilde{J}_{i,j} - D_{i,j}\}} \tag{6}$$

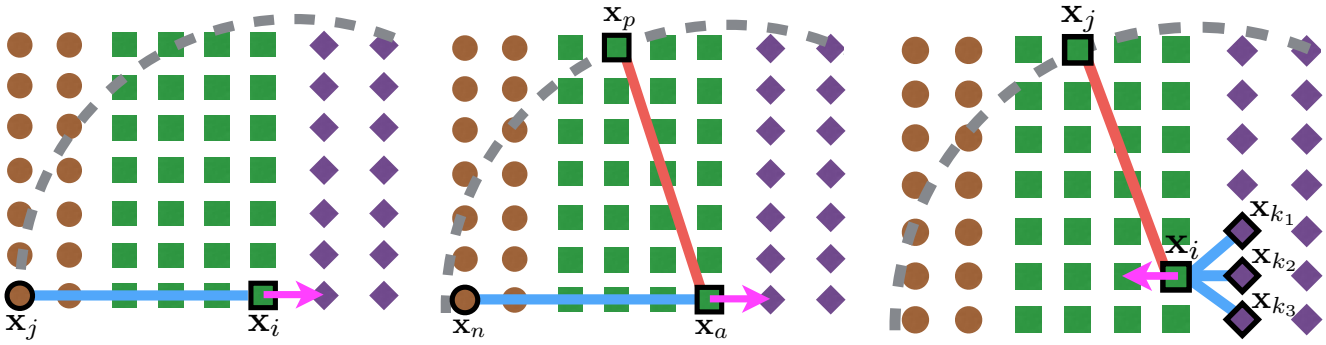
$$\frac{\partial \tilde{J}}{\partial D_{j,l}} = \frac{1}{|\mathcal{P}|} \tilde{J}_{i,j}\ \mathbb{1}[\tilde{J}_{i,j} > 0]\ \frac{-\exp\{\alpha - D_{j,l}\}}{\exp\{\tilde{J}_{i,j} - D_{i,j}\}}, \tag{7}$$

where $\mathbb{1}[\cdot]$ is the indicator function which outputs 1 if the expression evaluates to true and outputs 0 otherwise. As shown in algorithm 1 and equations 5, 6, and 7, our method provides informative gradient signals for all negative pairs as long as they are within the margin of any positive pairs (in contrast to only updating the hardest negative) which makes the optimization much more stable.

input : D, α
output: $\partial\tilde{J}/\partial f(\mathbf{x}_i),\ \ \forall i \in [1, m]$
Initialize: $\partial\tilde{J}/\partial f(\mathbf{x}_i) = \mathbf{0},\ \ \forall i \in [1, m]$
for $i = 1, \ldots, m$ **do**
  for $j = i+1, \ldots, m,\ \ s.t.\ (i,j) \in \mathcal{P}$ **do**
    $\partial\tilde{J}/\partial f(\mathbf{x}_i) \leftarrow \partial\tilde{J}/\partial f(\mathbf{x}_i) + \partial\tilde{J}/\partial D_{i,j}\,\partial D_{i,j}/\partial f(\mathbf{x}_i)$
    $\partial\tilde{J}/\partial f(\mathbf{x}_j) \leftarrow \partial\tilde{J}/\partial f(\mathbf{x}_j) + \partial\tilde{J}/\partial D_{i,j}\,\partial D_{i,j}/\partial f(\mathbf{x}_j)$
    for $k = 1, \ldots, m,\ \ s.t.\ (i,k) \in \mathcal{N}$ **do**
      $\partial\tilde{J}/\partial f(\mathbf{x}_i) \leftarrow \partial\tilde{J}/\partial f(\mathbf{x}_i) + \partial\tilde{J}/\partial D_{i,k}\,\partial D_{i,k}/\partial f(\mathbf{x}_i)$
      $\partial\tilde{J}/\partial f(\mathbf{x}_k) \leftarrow \partial\tilde{J}/\partial f(\mathbf{x}_k) + \partial\tilde{J}/\partial D_{i,k}\,\partial D_{i,k}/\partial f(\mathbf{x}_k)$
    end
    for $l = 1, \ldots, m,\ \ s.t.\ (j,l) \in \mathcal{N}$ **do**
      $\partial\tilde{J}/\partial f(\mathbf{x}_j) \leftarrow \partial\tilde{J}/\partial f(\mathbf{x}_j) + \partial\tilde{J}/\partial D_{j,l}\,\partial D_{j,l}/\partial f(\mathbf{x}_j)$
      $\partial\tilde{J}/\partial f(\mathbf{x}_l) \leftarrow \partial\tilde{J}/\partial f(\mathbf{x}_l) + \partial\tilde{J}/\partial D_{j,l}\,\partial D_{j,l}/\partial f(\mathbf{x}_l)$
    end
  end
end

Algorithm 1: Backpropagation gradient

Having stated the formal objective, we now illustrate and discuss some of the failure modes of the contrastive [14] and triplet [31, 39] embedding in which the proposed embedding learns successfully. Figure 4 illustrates the failure cases in 2D with examples from three different classes. Contrastive embedding (Fig. 4a) can fail if the randomly sampled negative ($\mathbf{x}_j$) is collinear with the examples from another class (purple examples in the figure). Triplet embedding (Fig. 4b) can also fail if such sampled negative ($\mathbf{x}_n$) is within the margin bound with respect to the sampled the positive example ($\mathbf{x}_p$) and the anchor ($\mathbf{x}_a$). In this case, both contrastive and triplet embedding incorrectly pushes the positive ($\mathbf{x}_i$/$\mathbf{x}_a$) towards the cluster of examples from the third class. However, in the proposed embedding (Fig. 4c), given sufficiently large random samples m, the hard negative examples ($\mathbf{x}_k$'s in Fig. 4c) within the margin bound pushes the positive $\mathbf{x}_i$ towards the correct direction.



(a) Contrastive embedding (b) Triplet embedding (c) Lifted structured similarity

Figure 4: Illustration of failure modes of contrastive and triplet loss with randomly sampled training batch. Brown circles, green squares, and purple diamonds represent three different classes. Dotted gray arcs indicate the margin bound (where the loss becomes zero out of the bound) in the hinge loss. Magenta arrows denote the negative gradient direction for the positives.

5. Implementation details

We used the Caffe [16] package for training and testing the embedding with contrastive [14], triplet [31, 39], and ours. Maximum training iteration was set to $20,000$ for all the experiments. The margin parameter α was set to 1.0. The batch size was set to 128 for contrastive and our method and to 120 for triplet. For training, all the convolutional layers were initialized from the network pretrained on ImageNet ILSVRC [30] dataset and the fully connected layer (the last layer) was initialized with random weights. We also multiplied the learning rate for the randomly initialized fully connected layers by 10.0 for faster convergence. All the train and test images are normalized to 256 by 256. For training data augmentation, all images are randomly cropped at 227 by 227 and randomly mirrored horizontally. For training, we exhaustively use all the positive pairs of examples and randomly subsample approximately equal number of negative pairs of examples as positives.

6. Evaluation

In this section, we briefly introduce the evaluation metrics used in the experiments. For the clustering task, we use the F_1 and NMI metrics. F_1 metric computes the harmonic mean of precision and recall. $F_1 = \frac{2PR}{P+R}$. The normalized mutual information (NMI) metric take as input a set of clusters $\Omega = \{\omega_1, \ldots, \omega_K\}$ and a set of ground truth classes $\mathbb{C} = \{c_1, \ldots, c_K\}$. ω_i indicates the set of examples with cluster assignment i. c_j indicates the set of examples with the ground truth class label j. NMI is defined by the ratio of mutual information and the average entropy of clusters and the entropy of labels. $\text{NMI}(\Omega, \mathbb{C}) = \frac{I(\Omega;\mathbb{C})}{2(H(\Omega)+H(\mathbb{C}))}$. We direct interested readers to refer [23] for complete details. For the retrieval task, we use the Recall@K [15] metric. Each test image (query) first retrieves K nearest neighbors from the test set and receives score 1 if an image of the same class is retrieved among the K nearest neighbors and 0 otherwise.

7. Experiments

We show experiments on CUB200-2011 [37], CARS196 [19], and our *Stanford Online Products* datasets where we use the first half of classes for training and the rest half classes for testing. For testing, we first compute the embedding on all the test images at varying embedding sizes $\{64, 128, 256, 512\}$ following the practice in [1, 31]. For clustering evaluation, we run affinity propagation clustering [11] with bisection method [10] for the desired number of clusters set equal to the number of classes in the test set. The clustering quality is measured in the standard F_1 and NMI metrics. For the retrieval evaluation, we report the result on the standard Recall@K metric [15] in log space of K. The experiments are performed with GoogLeNet [33].

7.1. Ablation study: effect of the batch size m

	F_1	NMI	R@1
$m = 32$	18.4	53.2	42.4
$m = 48$	19.1	53.8	42.1
$m = 64$	19.9	53.8	42.4
$m = 128$	19.7	54.1	42.8

Table 1: CUB200

	F_1	NMI	R@1
$m = 32$	20.5	55.6	46.9
$m = 48$	21.2	55.9	49.4
$m = 64$	22.7	56.6	50.3
$m = 128$	22.8	56.7	49.5

Table 2: Cars196

Tables 1 and 2 show the effect of batch size (m) for CUB-200-211 and CARS196 datasets in terms of F1, NMI, and R@1. On GoogLeNet, the maximum batch size is limited to 128 due to GPU (NVIDIA K80) memory constraint. The minimum batch size where the training doesn't diverge due to unstable gradient is around 32. Computing the proposed smooth structured estimation provides stability in terms of

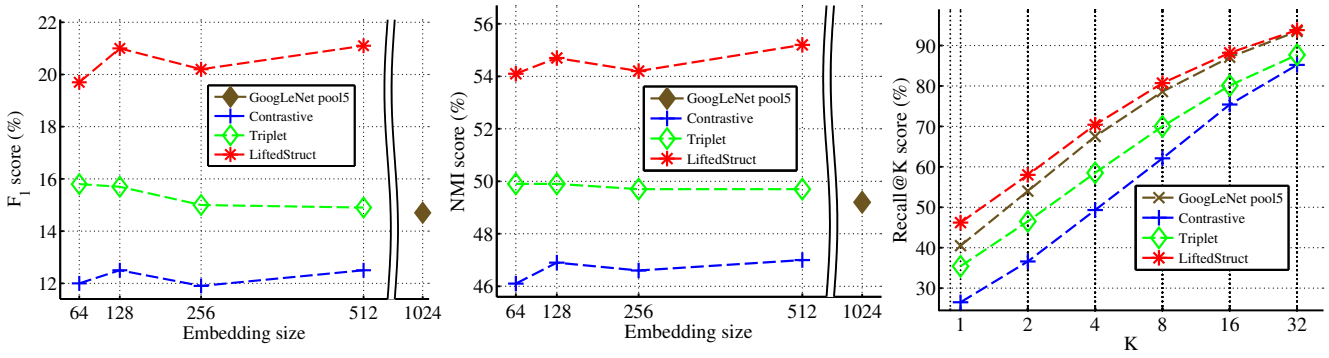


Figure 5: F_1, NMI, and Recall@K score metrics on the test split of CUB200-2011 with GoogLeNet [33].

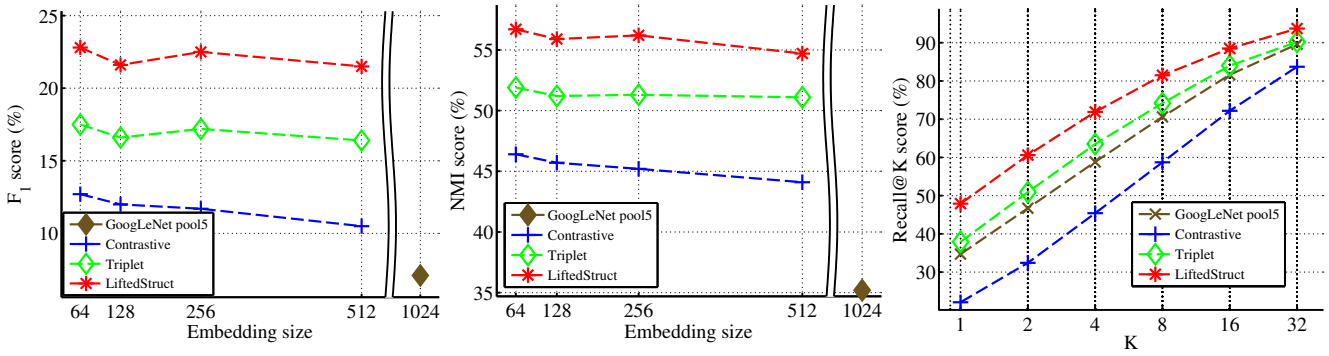


Figure 6: F_1, NMI, and Recall@K score metrics on the test split of CARS196 with GoogLeNet [33].

the batch size as shown in tables 1 and 2.

7.2. CUB-200-2011

The CUB-200-2011 dataset [37] has 200 classes of birds with 11,788 images. We split the first 100 classes for training (5,864 images) and the rest of the classes for testing (5,924 images). Figure 5 shows the quantitative clustering quality for the contrastive [14], triplet [39, 31], and using pool5 activation from the pretrained GoogLeNet [33] network on ImageNet [30], and our method on both F_1, NMI, and Recall@K metrics. Our embedding shows significant performance margin both on the standard F_1, NMI, and Recall@K metrics on all the embedding sizes. Please refer to the supplementary material for qualitative retrieval results on the test split of CUB200-2011 [37] dataset. Figure 7 shows the Barnes-Hut t-SNE visualization [36] on our 64 dimensional embedding. Although t-SNE embedding does not directly translate to the high dimensional embedding, it is clear that similar types of birds are quite clustered together and are apart from other species.

7.3. CARS196

The CARS196 data set [19] has 198 classes of cars with 16,185 images. We split the first 98 classes for training (8,054 images) and the other 98 classes for testing (8,131 images). Figure 6 shows the quantitative clustering quality for the contrastive [14], triplet [39, 31], and using pool5 activation from pretrained GoogLeNet [33] network on ImageNet [30]. Our embedding shows significant margin in terms of the standard F_1, NMI, and Recall@K metrics on all the embedding sizes. Please refer to the supplementary material for qualitative retrieval results on the test split of Cars196 [19] dataset. Figure 8 shows the Barnes-Hut t-SNE visualization [36] on our 64 dimensional embedding. We can observe that the embedding clusters the images from the same brand of cars despite the significant pose variations and the changes in the body paint.

7.4. Stanford Online Products dataset

We used the web crawling API from eBay.com [9] to download images and filtered duplicate and irrelevant images (i.e. photos of contact phone numbers, logos, etc). The preprocessed dataset has 120,053 images of 22,634 online products (classes) from eBay.com. Each product has approximately 5.3 images. For the experiments, we split 59,551 images of 11,318 classes for training and 60,502 images of 11,316 classes for testing. Figure 9 shows the quantitative clustering and retrieval results on F_1, NMI, and Recall@K metric with GoogLeNet. Figures 10 and 11

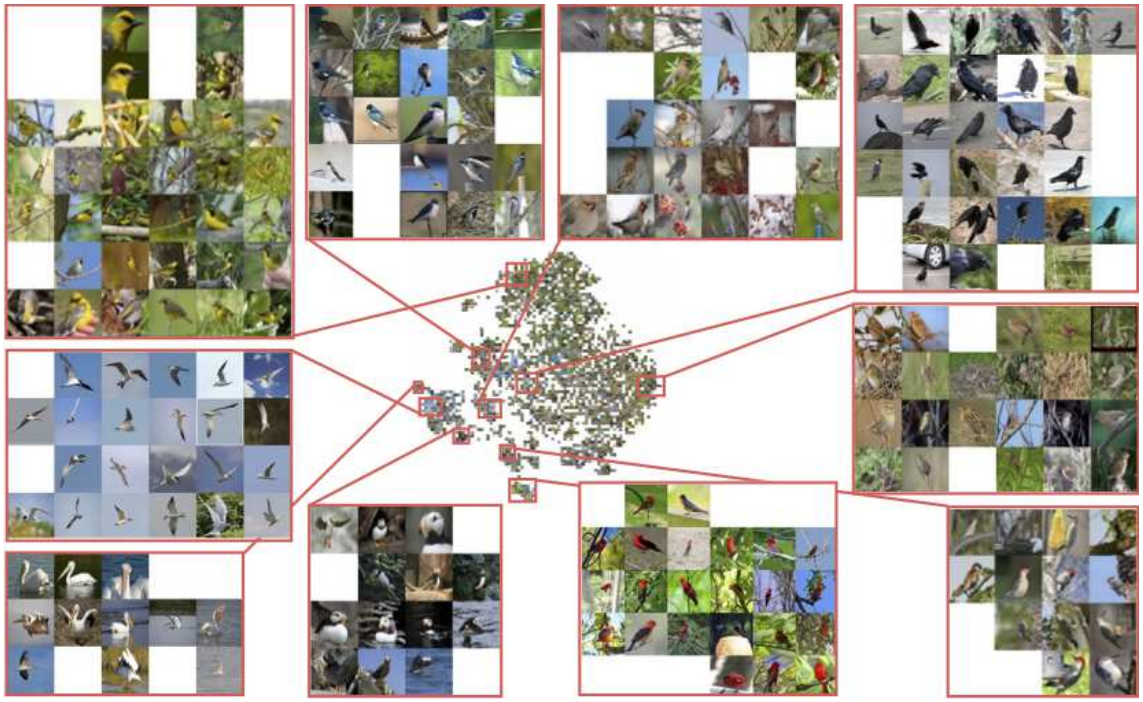


Figure 7: Barnes-Hut t-SNE visualization [36] of our embedding on the test split (class 101 to 200; 5,924 images) of CUB-200-2011. Best viewed on a monitor when zoomed in.

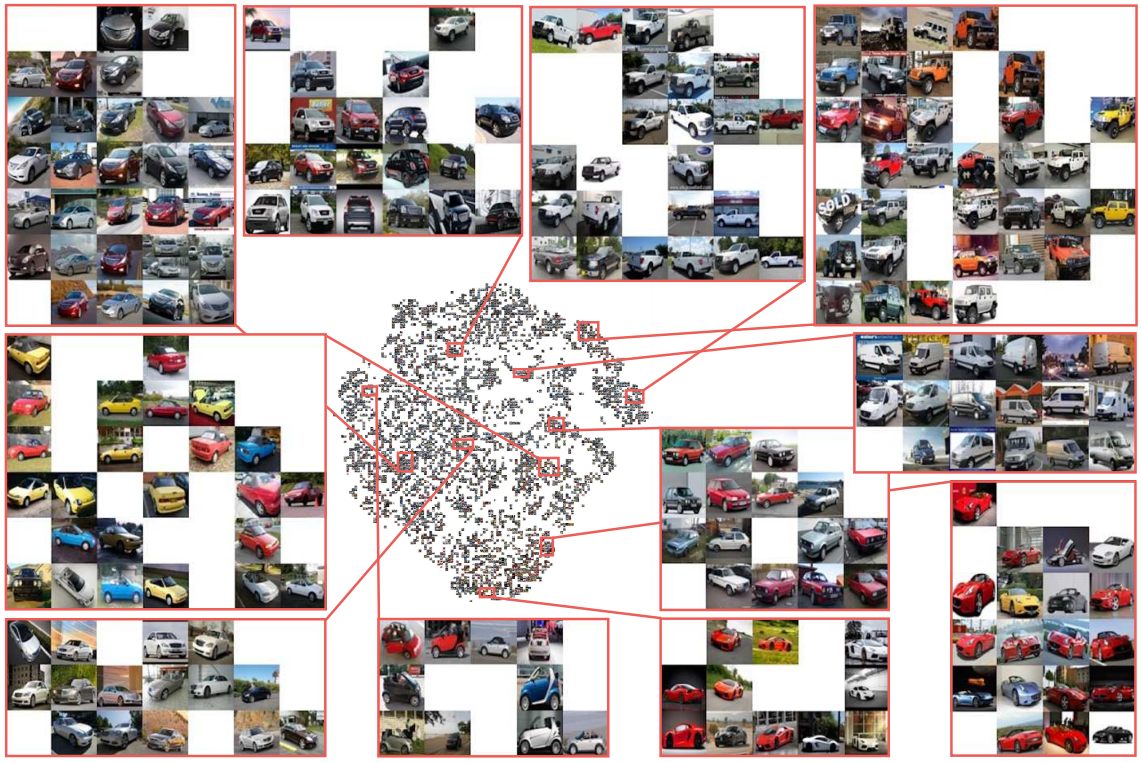


Figure 8: Barnes-Hut t-SNE visualization [36] of our embedding on the test split (class 99 to 196; 8,131 images) of CARS196. Best viewed on a monitor when zoomed in.

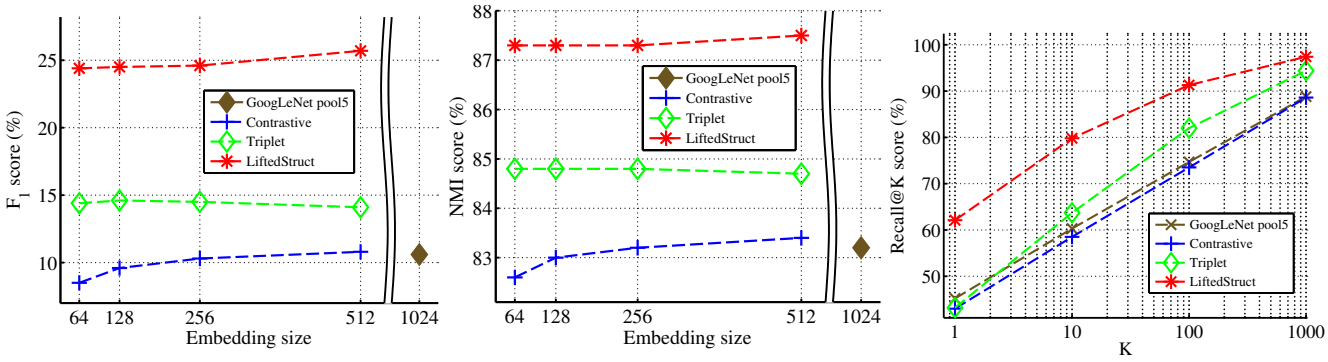


Figure 9: F_1, NMI, and Recall@K score metrics on the test split of *Stanford Online Products* with GoogLeNet [33].

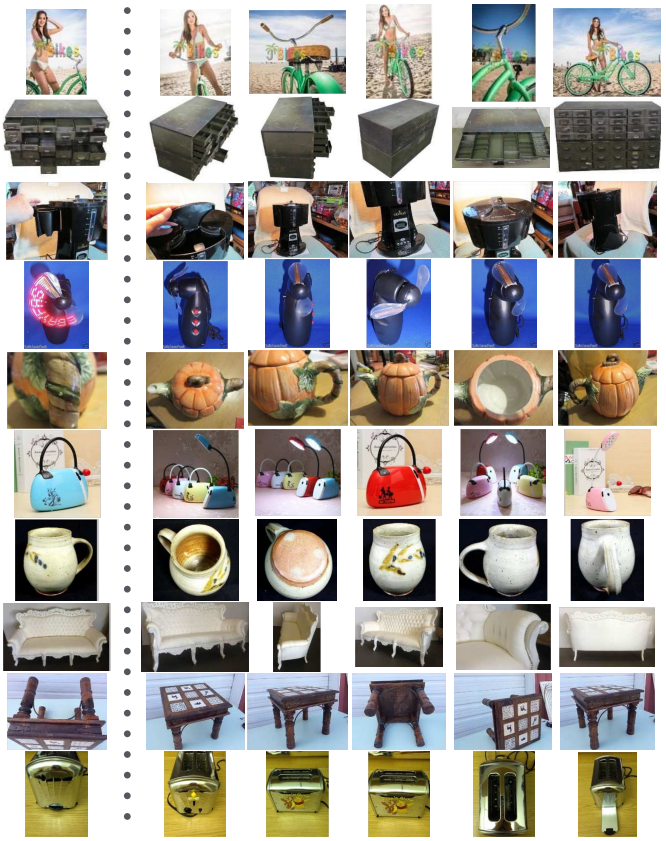


Figure 10: Examples of successful queries on our *Stanford Online Products* dataset using our embedding (size 512). Images in the first column are query images and the rest are five nearest neighbors.

show some example queries and nearest neighbors on the dataset for both successful and failure cases. Despite the huge changes in the viewpoint, configuration, and illumination, our method can successfully retrieve examples from the same class and most retrieval failures come from fine grained subtle differences among similar products. Please refer to the supplementary material for the t-SNE visualization of the learned embedding on our *Stanford Online Products* dataset.

Figure 11: Examples of failure queries on *Stanford Online Products* dataset. Most failures are fine grained subtle differences among similar products. Images in the first column are query images and the rest are five nearest neighbors.

8. Conclusion

We described a deep feature embedding and metric learning algorithm which defines a novel structured prediction objective on the lifted pairwise distance matrix within the batch during the neural network training. The experimental results on CUB-200-2011 [37], CARS196 [19], and *Stanford Online Products* datasets show state of the art performance on all the experimented embedding dimensions.

Acknowledgments

We acknowledge the support of ONR grant #N00014-13-1-0761 and grant #122282 from the Stanford AI Lab-Toyota Center for Artificial Intelligence Research.

References

[1] S. Bell and K. Bala. Learning visual similarity for product design with convolutional neural networks. In *SIGGRAPH*, 2015. 1, 2, 3, 5

[2] Y. Bengio, J. Paiement, and P. Vincent. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In *NIPS*, 2004. 1

[3] J. Bromley, I. Guyon, Y. Lecun, E. SŁckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. In *NIPS*, 1994. 2

[4] G. Chechik, V. Sharma, U. Shalit, and S. Bengio. Large scale online learning of image similarity through ranking. *JMLR*, 11, 2010. 2

[5] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, June 2005. 2

[6] A. Choromanska, A. Agarwal, and J. Langford. Extreme multi class classification. In *NIPS*, 2013. 1

[7] T. Cox and M. Cox. Multidimensional scaling. In *London: Chapman and Hill*, 1994. 1

[8] I. S. Data. https://sites.google.com/site/imagesimilaritydata/, 2014. 2

[9] eBay Developers Program. http://go.developer.ebay.com/what-ebay-api, 2015. 6

[10] B. J. Frey and D. Dueck. apclusterk.m. http://www.psi.toronto.edu/affinitypropagation/apclusterK.m, 2007. 5

[11] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 2007. 5

[12] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov. Devise: A deep visual-semantic embedding model. In *NIPS*, 2013. 2

[13] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood component analysis. In *NIPS*, 2004. 1

[14] R. Hadsell, S. Chopra, and Y. Lecun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006. 2, 3, 4, 5, 6

[15] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. In *PAMI*, 2011. 5

[16] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 5

[17] T. Joachims, T. Finley, and C.-N. Yu. Cutting-plane training of structural svms. *JMLR*, 2009. 4

[18] T. I. Jolliffe. Principal component analysis. In *New York: Springer-Verlag*, 1986. 1

[19] J. Krause, M. Stark, J. Deng, and F.-F. Li. 3d object representations for fine-grained categorization. *ICCV 3dRR-13*, 2013. 1, 2, 5, 6, 8

[20] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2

[21] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. In *TPAMI*, 2014. 2

[22] Y. Li, H. Su, C. Qi, N. Fish, D. Cohen-Or, and L. Guibas. Joint embeddings of shapes and images via cnn image purification. In *SIGGRAPH Asia*, 2015. 2, 3

[23] C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge university press, 2008. 2, 5

[24] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurk. Metric learning for large scale image classification: Generalizaing to new classes at near-zero cost. In *ECCV*, 2012. 2

[25] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell. Zero-shot learning with semantic output codes. In *NIPS*, 2009. 2

[26] Y. Prabhu and M. Varma. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *SIGKDD*, 2014. 1

[27] Q. Qian, R. Jin, S. Zhu, and Y. Lin. Fine-grained visual categorization via multi-stage metric learning. In *CVPR*, 2015. 2

[28] M. Rohrbach, M. Stark, and B. Schiel. Evaluating knowledge transfer and zero-shot learn- ing in a large-scale setting. In *CVPR*, 2011. 2

[29] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. In *Science*, 290. 1

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. 5, 6

[31] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015. 1, 2, 3, 4, 5, 6

[32] R. Socher, C. D. M. M. Ganjoo H. Sridhar, O. Bastani, and A. Y. Ng. Zero-shot learning through cross-modal transfer. In *ICLR*, 2013. 2

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1, 2, 5, 6, 8

[34] G. Taylor, R. Fergus, G. Williams, I. Spiro, and C. Bregler. Pose-sensitive embedding by nonlinear nca regression. In *NIPS*, 2010. 1

[35] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004. 4

[36] L. van der maaten. Accelerating t-sne using tree-based algorithms. In *JMLR*, 2014. 6, 7

[37] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 1, 2, 5, 6, 8

[38] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *CVPR*, 2014. 2

[39] K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. In *NIPS*, 2006. 1, 2, 3, 4, 5, 6

[40] J. Weston, S. Bengio, and N. Usurer. Wsabi: Scaling up to large vocabulary image annotation. In *IJCAI*, 2011. 2