# Supplementary Materials
# RankMI: A Mutual Information Maximizing Ranking Loss

Mete Kemertas, Leila Pishdad, Konstantinos G. Derpanis, and Afsaneh Fazly
Samsung AI Centre Toronto
{mete.kemertas,leila.p,k.derpanis,a.fazly}@samsung.com

## 1. Approximating the product of marginals

In (5) of the main manuscript, we presented an approximation to the product of marginals via negative pairs. Here, we elaborate and justify this approximation.

First, we rewrite the product of marginals as follows:

$$\begin{aligned} p(x_i)p(x_j) &= \sum_{k\in\mathcal{C}} \Bigg( p(x_i, c{=}k)p(x_j, c{=}k) \\ &\quad + \sum_{\substack{k'\in\mathcal{C} \\ k'\neq k}} p(x_i, c{=}k)p(x_j, c{=}k') \Bigg) \\ &= p(x_i, x_j) + \sum_{k\in\mathcal{C}} \sum_{\substack{k'\in\mathcal{C} \\ k'\neq k}} p(x_i, c{=}k)p(x_j, c{=}k'). \end{aligned} \tag{1}$$

We consider a dataset of $N$ images equally partitioned between $||\mathcal{C}||$ classes, i.e., $p(c{=}k) = 1/||\mathcal{C}||$ is uniformly distributed and $p(x_i|c{=}k) = ||\mathcal{C}||/N$. We observe that the first term corresponding to the joint distribution grows linearly with $||\mathcal{C}||$, while the second term corresponding to the distribution of negative pairs grows quadratically:

$$p(x_i)p(x_j) = \frac{||\mathcal{C}||}{N^2} + \frac{||\mathcal{C}||^2 - ||\mathcal{C}||}{N^2}. \tag{2}$$

Since $||\mathcal{C}||^2 - ||\mathcal{C}|| \gg ||\mathcal{C}||$, for large $||\mathcal{C}||$:

$$p(x_i)p(x_j) \approx \frac{||\mathcal{C}||^2 - ||\mathcal{C}||}{N^2}. \tag{3}$$

For the datasets used in our experiments, CUB-200-2011 [17], CARS-196 [4] and Stanford Online Products [13], the number of classes in the training set are 100, 98, and 11318, respectively.

## 2. MI estimation with pairwise distances

In (6) and (7) of the main manuscript, we defined $T_\phi(z_i, z_j)$ as a function of the distance $d_{ij}$ between $(z_i, z_j)$. Here, we discuss the conditions under which this definition can provide tight estimates of JSD and provide a motivating example.

Recall that the dual representation of JSD is defined as:

$$\hat{D}_{\text{JSD}}(\mathbb{J} \parallel \mathbb{M}) = \sup_{T\in\mathcal{F}} \Big\{ \mathbb{E}_{\mathbb{J}}[T(z_i, z_j)] + \mathbb{E}_{\mathbb{M}}[\log(2 - e^{T(z_i, z_j)})] \Big\}, \tag{4}$$

where $\mathcal{F}$ contains all functions $T : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that the expectations are finite. This representation has been previously used to establish a lower bound [8]:

$$D_{\text{JSD}}(\mathbb{J} \parallel \mathbb{M}) \geq \hat{D}_{\text{JSD}}(\mathbb{J} \parallel \mathbb{M}). \tag{5}$$

**Lemma 1** Given an optimal quantizer $f : \mathbb{R}^n \to \mathbb{R}^d$, there exists a class of functions, $\tilde{\mathcal{F}} \subset \mathcal{F}$ of the following form, such that the tightest lower bound on JSD can be attained:

$$\tilde{\mathcal{F}} = \{T(z_i, z_j) = \log(2) - \log(1 + e^{-V(d_{ij})})\}, \tag{6}$$

where $d_{ij} = D(f(x_i), f(x_j))$ for some distance function $D : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$.

*Proof.* Let $f : \mathbb{R}^n \to \mathbb{S}^1$, an optimal quantizer of $x_i$, be a function mapping each image to one of $||C||$ equally spaced points on the unit circle $\mathbb{S}^1$ that corresponds to its class label $c_i{=}k_i$, where $k_i \in \{0, 1, ..., ||\mathcal{C}||{-}1\}$:[1]

$$z_i = f(x_i) = \begin{bmatrix} \cos(2\pi \frac{k_i}{||C||}) & \sin(2\pi \frac{k_i}{||C||}) \end{bmatrix}. \tag{7}$$

Let the lower bound on the JSD estimation error, $\epsilon$, be defined as follows:

$$\epsilon = \Big| D_{\text{JSD}}(\mathbb{J} \parallel \mathbb{M}) - \hat{D}_{\text{JSD}}(\mathbb{J} \parallel \mathbb{M}) \Big|. \tag{8}$$

Any subset of $\mathcal{F}$ will induce an error $\eta$ such that $0 \leq \epsilon \leq \eta$. Let the function $V : \mathbb{R}_+ \to \mathbb{R}$ in (6) be defined as follows:

$$V(d_{ij}) = \begin{cases} M & d_{ij} < \tau \\ -M & d_{ij} \geq \tau \end{cases}, \tag{9}$$

[1] Here, we focus on 2D embeddings lying on the unit circle; however, this proof can be extended to any convex subset of $\mathbb{R}^n$.

where $\tau$ is the minimum distance between any pair $(z_i, z_j)$ s.t. $c_i \neq c_j$. Observe that, as $M \to \infty$, $\eta \to \epsilon$, yielding the tightest possible lower bound.

**Lemma 2** Given an optimal quantizer of $x_i$, $f(x_i)$, there exists a neural network with parameters $\phi$, which can estimate JSD with arbitrarily small added error.

*Proof.* By the universal approximation theorem for neural networks [2], a neural network $V_\phi$ can approximate the supremum over $\tilde{\mathcal{F}}$, and therefore estimate the JSD with an arbitrarily small, finite error $\eta \in [\epsilon, \infty)$.

**Conjecture** By the universal approximation theorem for neural networks, a neural network $f_\theta$ can be trained to approximate the optimal quantizer, $f$, such that when trained in conjunction with $V_\phi$ (see Lemma 2), mutual information between $(z_i, z_j)$ can be estimated as a function of their distances $d_{ij}$.

**Summary** In Lemma 1, we showed that given a well-behaved function, $f$, i.e., an optimal quantizer, the JSD between the joint distribution and the product of marginals can be estimated tightly as a function of pairwise distances. In Lemma 2, we showed that this estimation can be performed with a neural network, $V_\phi$, that induces an arbitrarily small added error. Finally, we claimed that given a neural network, $f_\theta$, that approximates an optimal quantizer, a joint training process can successfully improve the tightness of mutual information estimates, as well as the quantization. Our experiments in the main manuscript confirm this intuition.

## 3. Monotonic $V_\phi$

Even though we did not observe cases where the requirement $\partial V_\phi / \partial d_{ij} < 0$, as described in (9) of the main text, is violated, a non-monotonic $V_\phi$ could also maximize the divergence between two $1D$ distributions in principle. While we do not adopt this approach for our experiments, here we describe an added loss term to $\mathcal{L}_{RankMI}$ to provide a mechanism for incorporating this behaviour into the statistics network.

We can penalize positive gradients at sampled points for $V_\phi$ with a loss term we define as order loss:

$$\mathcal{L}_{\text{order}} = \frac{1}{||\mathcal{P}||} \sum_{(z_i,z_j)\in\mathcal{P}} \max(0, \partial V_\phi / \partial d_{ij}) + \frac{1}{||\mathcal{N}||} \sum_{(z_i,z_j)\in\mathcal{N}} \max(0, \partial V_\phi / \partial d_{ij}). \quad (10)$$

Then, we can reformulate our total loss as follows:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{RankMI}} + \lambda \mathcal{L}_{\text{order}}. \quad (11)$$

With this loss, we can softly constrain the parameter space $\Phi$ to corresponding functions $V_\phi$ that are non-increasing.

**Algorithm 1** Newton's method for finding $\beta$.

**Require:** $\phi_t$: Statistics network parameters at timestep $t$.
**Require:** $\beta_t^{(0)}$: Initialization of $\beta$ for Newton's method.
**Require:** $\epsilon$: Stopping criterion.
**Require:** $maxIter$: Maximum number of iterations.

$\epsilon^{(0)} \leftarrow \infty$
$n \leftarrow 0$
**while** $|\epsilon^{(n)}| > \epsilon$ and $n < maxIter$ **do**
  $T \leftarrow T_{\phi_t}(\beta_t^{(n)})$
  $g \leftarrow \partial T / \partial \beta_t^{(n)}$
  $\epsilon^{(n)} \leftarrow T / g$
  $\beta_t^{(n+1)} \leftarrow \beta_t^{(n)} - \epsilon^{(n)}$
  $n \leftarrow n + 1$
**return** $\beta_t^{(n)}$

## 4. Esimating $\beta$ with Newton's method

Algorithm 1 outlines our simple implementation of Newton's method that is used to estimate the $\beta$ value described in Section 3.3 of the main manuscript. For all experiments, we used the following settings: $\epsilon = 10^{-6}$ and $maxIter = 100$.

## 5. Expanded quantitative results

An extended table of results for CUB200-2011 [17] and CARS-196 [4] is shown in Tables 1 and 2, respectively.

## 6. Ablation study

In this section we present an ablation study that analyzes the sensitivity of RankMI to its hyperparameters, i.e., statistics network depth/width, alternating gradient descent ratio (AGDR), batch size, and embedding dimensionality. The study is based on CUB-200 [17]. Unless otherwise stated, the statistics network depth/width is set to $5/128$, AGDR is 1, embedding dimensionality is 128, batch size is 120, and distance weighted sampling is used.

Table 3 analyzes the contribution of the depth and width of the statistics network. As can be seen, changes in the network size yield negligible differences in performance. Similarly in Table 4, varying the AGDR does not impact performance. In Table 5, varying the batch size, $B$, from 120, 180 to 240 results in a performance loss of up to $4\%$ with increasing $B$. Further evaluations suggest that this can be mitigated if the number of classes on each GPU is fixed to a constant value, three for our eight GPU evaluation setup. This may be related to PyTorch's batchnorm implementation, which uses different running statistics on each GPU. Finally in Table 6, increasing the embedding dimension to 512 yields a $1\%$ increase in R@1. Overall, RankMI is relatively robust to hyperparameter settings across various axes with the exception of batch size, which requires adjustments

| Methods | | Recall@$k$ | | | | | NMI |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 4 | 8 | 16 | |
| Triplet Semi-hard [10] 128O | | 42.6 | 55.0 | 66.4 | 77.2 | - | 55.4 |
| LiftedStruct [13, 12] 64B | | 43.6 | 56.6 | 68.6 | 79.6 | - | 56.5 |
| StructClustering [12] 64B | | 48.2 | 61.4 | 71.8 | 81.9 | - | 59.2 |
| Proxy NCA [7] 64B | | 49.2 | 61.9 | 67.9 | 72.4 | - | 59.5 |
| Binomial Deviance [14] 512G | | 50.3 | 61.9 | 72.6 | 82.4 | 88.8 | - |
| N-pairs [11] 64G | | 51.0 | 63.3 | 74.3 | 83.2 | - | 60.4 |
| DVML + Triplet$_2$ + DWS [5] 512G | | 52.7 | 65.1 | 75.5 | 84.3 | - | 61.4 |
| Histogram [14] 512G | | 52.8 | 64.4 | 74.7 | 83.9 | 90.4 | - |
| Angular Loss [16] 512G | | 53.6 | 65.0 | 75.3 | 83.7 | - | 61.0 |
| HDML + N-pairs [19] 512G | | 53.7 | 65.7 | 76.7 | 85.7 | - | 62.6 |
| HTL [1] 512G | | 57.1 | 68.8 | 78.7 | 86.5 | 92.5 | - |
| Margin [6] 128R | | 63.6 | 74.4 | 83.1 | 90.0 | 94.2 | 69.0 |
| Ensemble | HDC [18] 384G | 53.6 | 65.7 | 77.0 | 85.6 | 91.5 | - |
| | BIER [9] 512G | 55.3 | 67.2 | 76.9 | 85.1 | 91.7 | - |
| | ABE-8 [3] 512G | 60.6 | 71.5 | 79.8 | 87.4 | - | - |
| RankMI (Ours) 128R | | 66.5 | 77.3 | 85.5 | 91.1 | 95.1 | 72.8 |

Table 1. Recall@$k$ and NMI on CUB200-2011 [17]. Baseline results are taken from the respective papers. The number after each citation denotes the embedding dimensionality. The letter after each embedding dimension indicates the embedding network used. The letters R, G, B, and O denote ResNet-50, GoogLeNet, BN-Inception and Other, respectively.

| Methods | | Recall@$k$ | | | | | NMI |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 4 | 8 | 16 | |
| Triplet Semi-hard [10] 128O | | 51.5 | 63.8 | 73.5 | 82.4 | - | 53.4 |
| LiftedStruct [13, 12] 64B | | 53.0 | 65.7 | 76.0 | 84.3 | - | 56.9 |
| StructClustering [12] 64B | | 58.1 | 70.6 | 80.3 | 87.8 | - | 59.0 |
| Angular Loss [16] 512G | | 71.3 | 80.7 | 87.0 | 91.8 | - | 62.4 |
| N-pairs [11] 64G | | 71.1 | 79.7 | 86.5 | 91.6 | - | 64.0 |
| Proxy NCA [7] 64B | | 73.2 | 82.4 | 86.4 | 88.7 | - | 64.9 |
| Margin [6] 128R | | 79.6 | 86.5 | 91.9 | 95.1 | 97.3 | 69.1 |
| HTL [1] 512G | | 81.4 | 88.0 | 92.7 | 95.7 | 97.4 | - |
| DVML + Triplet$_2$ + DWS [5] 512G | | 82.0 | 88.4 | 93.3 | 96.3 | - | 67.6 |
| Ensemble | HDC [18] 384G | 73.7 | 83.2 | 89.5 | 93.8 | 96.7 | - |
| | BIER [9] 512G | 78.0 | 85.8 | 91.1 | 95.1 | 97.3 | - |
| | ABE-8 [3] 512G | 85.2 | 90.5 | 94.0 | 96.1 | - | - |
| RankMI (Ours) 128R | | 83.3 | 89.8 | 93.8 | 96.1 | 97.7 | 69.0 |

Table 2. Recall@$k$ and NMI on CARS-196 [4]. Baseline results are taken from the respective papers. The number after each citation denotes the embedding dimensionality. The letter after each embedding dimension indicates the embedding network used. The letters R, G, B, and O denote ResNet-50, GoogLeNet, BN-Inception and Other, respectively.

to sampling.

## 7. Training complexity

We measure the overhead of the alternating gradient descent (AGD) by timing the full iteration when the embedding network, $\theta$, and statistics network, $\phi$, are updated. Training steps for the statistics network are inexpensive and take $\sim$0.35s, while updates to the embedding network take $\sim$1.28s on an NVIDIA Tesla M40 GPU. Thus, a single update to the embedding network takes $\sim 1.27\times$ wall-clock time with an AGD ratio of 1 ($1.3\times$ with 8 GPUs). Since the number of parameters $||\phi|| \ll ||\theta||$ (50k vs. 25.8m), additional memory cost is negligible. Hyperparameter $\beta_0$ is not sensitive, since it is updated before training starts in line 2 of Algorithm 1. The cost of Newton's algorithm for estimating $\beta_1$ from $\beta_0$ is negligible.

## 8. Qualitative results

Figures 1 and 2 show t-SNE [15] visualizations of the test set embeddings after training with RankMI on CUB-

| depth | width | Recall@$k$ | | | | |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 4 | 8 | 16 |
| 5 | 128 | 66.4 | 76.7 | 85.5 | 91.2 | 94.9 |
| 3 | 128 | 66.4 | 77.2 | 85.4 | 91.2 | 94.8 |
| 7 | 128 | 65.7 | 77.2 | 85.6 | 91.1 | 95.0 |
| 5 | 32 | 65.7 | 77.1 | 85.4 | 91.4 | 94.9 |
| 5 | 512 | 66.6 | 77.4 | 85.4 | 91.2 | 94.8 |

Table 3. Ablation study of the statistics network depth and width on CUB-200 [17].

| AGDR | Recall@$k$ | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 4 | 8 | 16 |
| 1 | 66.4 | 76.7 | 85.5 | 91.2 | 94.9 |
| 2 | 66.6 | 77.1 | 85.8 | 91.5 | 95.1 |
| 4 | 65.9 | 76.0 | 84.9 | 91.0 | 94.8 |

Table 4. Ablation study of the alternating gradient descent ratio (AGDR) on CUB-200 [17].

| batch size | Recall@$k$ | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 4 | 8 | 16 |
| 120 | 66.4 | 76.7 | 85.5 | 91.2 | 94.9 |
| 180 | 65.2 | 75.8 | 84.8 | 90.9 | 95.1 |
| 240 | 62.1 | 74.1 | 83.5 | 89.7 | 94.4 |

Table 5. Ablation study of the training batch size on CUB-200 [17].

| embedding dim | Recall@$k$ | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 4 | 8 | 16 |
| 128 | 66.4 | 76.7 | 85.5 | 91.2 | 94.9 |
| 512 | 67.2 | 77.0 | 86.0 | 91.8 | 95.3 |

Table 6. Ablation study of the embedding dimensionality on CUB-200 [17].

200-2011 [17] and CARS-196 [4] training sets, respectively. Despite the subtle inter-class and large-class variations contained in the datasets, the embeddings form tight clusters around semantic classes.

## References


[1] Weifeng Ge, Weilin Huang, Dengke Dong, and Matthew R. Scott. Deep metric learning with hierarchical triplet loss. In *European Conference on Computer Vision*, 2018. 3

[2] Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. 2

[3] Wonsik Kim, Bhavya Goyal, Kunal Chawla, Jungmin Lee, and Keunjoo Kwon. Attention-based ensemble for deep metric learning. In *European Conference on Computer Vision*, pages 760–777, 2018. 3

[4] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D object representations for fine-grained categorization. In *International Conference on Computer Vision Workshops*, pages 554–561, 2013. 1, 2, 3, 4, 5

[5] Xudong Lin, Yueqi Duan, Qiyuan Dong, Jiwen Lu, and Jie Zhou. Deep variational metric learning. In *European Conference on Computer Vision*, pages 714–729, 2018. 3

[6] R. Manmatha, Chao-Yuan Wu, Alexander J. Smola, and Philipp Krähenbühl. Sampling matters in deep embedding learning. In *International Conference on Computer Vision*, pages 2859–2867, 2017. 3

[7] Yair Movshovitz-Attias, Alexander Toshev, Thomas K. Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *International Conference on Computer Vision*, pages 360–368, 2017. 3

[8] XuanLong Nguyen, Martin J Wainwright, and Michael I. Jordan. Estimating divergence functionals and the likelihood ratio by penalized convex risk minimization. In *Conference on Neural Information Processing Systems*, 2008. 1

[9] Michael Opitz, Georg Waltner, Horst Possegger, and Horst Bischof. Deep metric learning with BIER: boosting independent embeddings robustly. *CoRR*, abs/1801.04815, 2018. 3

[10] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. In *Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015. 3

[11] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Conference on Neural Information Processing Systems*, pages 1849–1857, 2016. 3

[12] Hyun Oh Song, Stefanie Jegelka, Vivek Rathod, and Kevin Murphy. Learnable structured clustering framework for deep metric learning. *CoRR*, abs/1612.01213, 2016. 3

[13] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Conference on Computer Vision and Pattern Recognition*, pages 4004–4012, 2016. 1, 3

[14] Evgeniya Ustinova and Victor Lempitsky. Learning deep embeddings with histogram loss. In *Conference on Neural Information Processing Systems*, 2016. 3

[15] Laurens van der Maaten and Geoffrey Hinton. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 9: 2579–2605, 2008. 3, 5

[16] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. Deep metric learning with angular loss. In *International Conference on Computer Vision*, 2017. 3

[17] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010. 1, 2, 3, 4, 5

[18] Yuhui Yuan, Kuiyuan Yang, and Chao Zhang. Hard-aware deeply cascaded embedding. In *International Conference on Computer Vision*, pages 814–823, 2017. 3

[19] Wenzhao Zheng, Zhaodong Chen, Jiwen Lu, and Jie Zhou. Hardness-aware deep metric learning. In *Conference on Computer Vision and Pattern Recognition*, 2019. 3

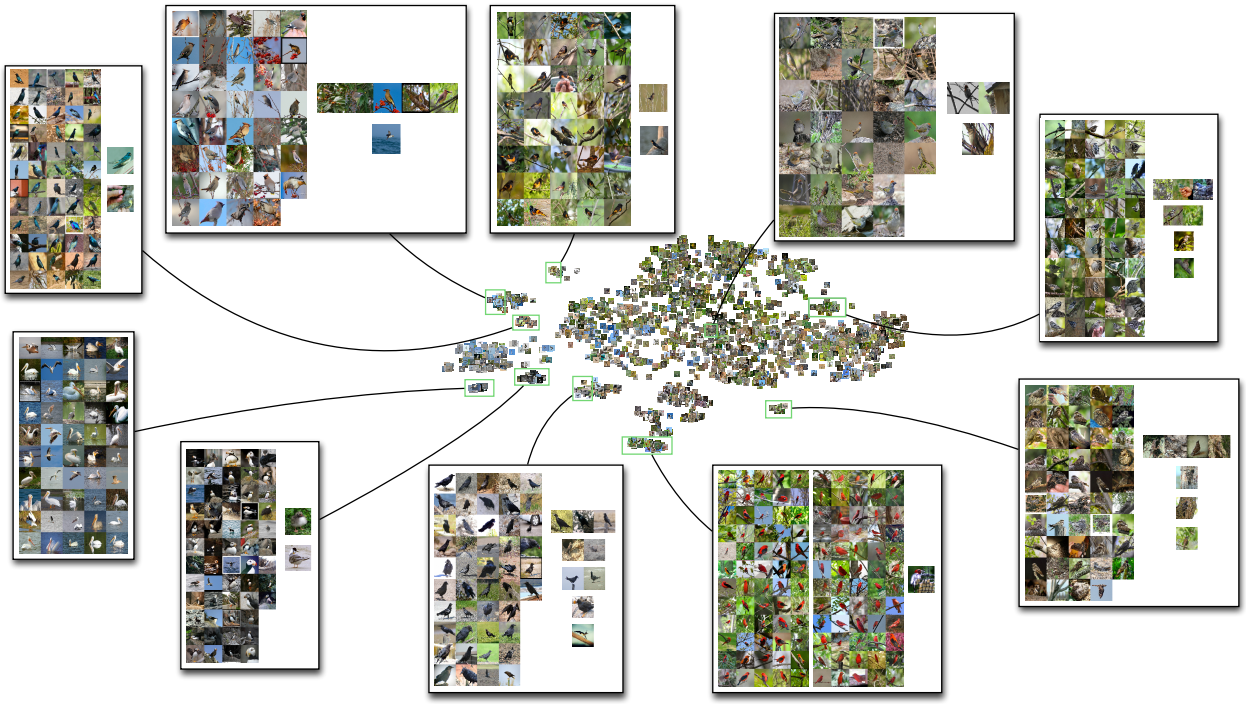

Figure 1. t-SNE visualizations [15] of the CUB-200-2011 [17] evaluation set embeddings after training with RankMI on the training set. Image blocks in the zoomed-in panels correspond to a single class.

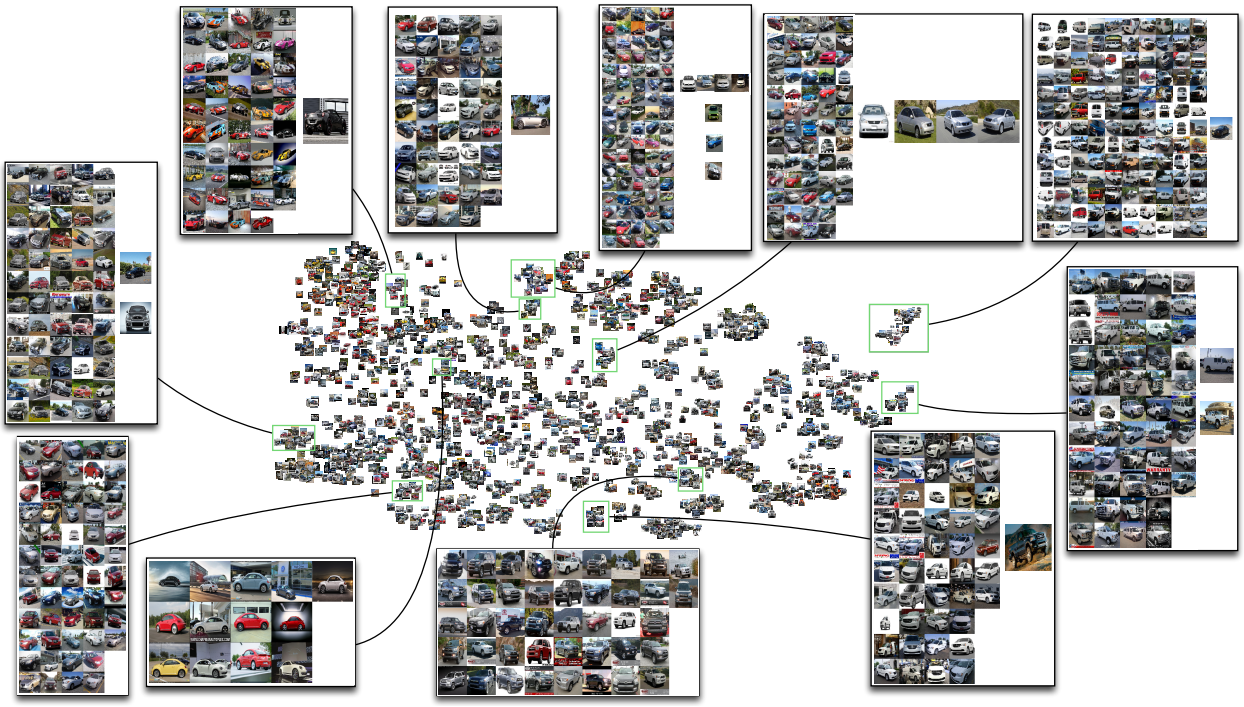

Figure 2. t-SNE visualizations [15] of the CARS-196 [4] evaluation set embeddings after training with RankMI on the training set. Image blocks in the zoomed-in panels correspond to a single class.