Supplementary material to “Uncertainty-guided Metric Learning without Labels”

Dhanunjaya Varma Devalraju and C Chandra Sekhar
Department of Computer Science and Engineering
Indian Institute of Technology Madras, India
cs21d006@smail.iitm.ac.in, chandra@cse.iitm.ac.in

This supplementary material provides more details on the General Pair Weighting (GPW) framework [11] used to analyze the modified loss function and more ablation studies to understand the proposed framework further. The complete information on the GPW framework and the analysis of the vanilla multi-similarity loss are already presented in [11]. We present the same material here for completeness and to understand the modified loss function.

1. Modified loss function Analysis

The vanilla multi-similarity loss proposed in [11]:

$$\begin{aligned}\mathcal{L}_{MS} = \frac{1}{m}\sum_{i=1}^{m}\Big(\frac{1}{\alpha}\log(1+\sum_{l\in P_i^+} e^{-\alpha(S_{il}-\lambda)})\\ +\frac{1}{\beta}\log(1+\sum_{l\in N_i^-} e^{\beta(S_{il}-\lambda)})\Big)\end{aligned} \tag{1}$$

The modified loss function that penalize the highly uncertain and less confident pairs by incorporating weight (w_{il}):

$$\begin{aligned}\widehat{\mathcal{L}}_{MS} = \frac{1}{m}\sum_{i=1}^{m}\Big(\frac{1}{\alpha}\log(1+\sum_{l\in P_i^+} (w_{il}\cdot e^{-\alpha(S_{il}-\lambda)}))\\ +\frac{1}{\beta}\log(1+\sum_{l\in N_i^-} (w_{il}\cdot e^{\beta(S_{il}-\lambda)}))\Big)\end{aligned} \tag{2}$$

To analyse the modified loss function, we use the General Pair Weighting (GPW) framework proposed in [11]. In general, given a neural network parameterized by θ and a pair-based loss function $\mathcal{L}$ defined in terms of similarity matrix $\mathbf{S}$ and labels y, the model parameters (θ) are optimized by computing the gradient with respect to θ as given below.

$$\begin{aligned}\frac{\partial\mathcal{L}(\mathbf{S},y)}{\partial\theta} &= \frac{\partial\mathcal{L}(\mathbf{S},y)}{\partial\mathbf{S}}\frac{\partial\mathbf{S}}{\partial\theta}\\ &= \sum_{i=1}^{m}\sum_{j=1}^{m}\frac{\partial\mathcal{L}(\mathbf{S},y)}{\partial S_{ij}}\frac{\partial S_{ij}}{\partial\theta}\end{aligned} \tag{3}$$

The GPW framework [11] presents a new formulation, whose gradient with respect to θ is determined exactly as Eq. 3. Consider the function defined below.

$$\begin{aligned}\mathcal{F} &= \sum_{i=1}^{m}\sum_{j=1}^{m}\frac{\partial\mathcal{L}(\mathbf{S},y)}{\partial S_{ij}}S_{ij}\\ &= \sum_{i=1}^{m}\left(\sum_{y_j\neq y_i}^{m}\frac{\partial\mathcal{L}(\mathbf{S},y)}{\partial S_{ij}}S_{ij} + \sum_{y_j=y_i}^{m}\frac{\partial\mathcal{L}(\mathbf{S},y)}{\partial S_{ij}}S_{ij}\right)\end{aligned} \tag{4}$$

Here $\frac{\partial\mathcal{L}(\mathbf{S},y)}{\partial S_{ij}}$ is considered as weight term since it does not participate in the gradient computation of $\mathcal{F}$ w.r.t θ [11]. Further, in [11], it was assumed that $\frac{\partial\mathcal{L}(\mathbf{S},y)}{\partial S_{ij}} \geq 0$ for a negative pair and $\frac{\partial\mathcal{L}(\mathbf{S},y)}{\partial S_{ij}} \leq 0$ for a positive pair and the equation in 4 is rewritten as:

$$\mathcal{F} = \sum_{i=1}^{m}\left(\sum_{y_j\neq y_i}^{m}\left|\frac{\partial\mathcal{L}(\mathbf{S},y)}{\partial S_{ij}}\right|S_{ij} - \sum_{y_j=y_i}^{m}\left|\frac{\partial\mathcal{L}(\mathbf{S},y)}{\partial S_{ij}}\right|S_{ij}\right) \tag{5}$$

A good pair-based loss function, when realized in GPW framework, is expected to assign higher weights ($\left|\frac{\partial\mathcal{L}(\mathbf{S},y)}{\partial S_{ij}}\right|$) to the informative pairs. When the vanilla multi-similarity loss (Eq. 1) is realized in the GPW framework, the positive and negative pair weights are obtained by computing gradient w.r.t S_{ij} [11].

The weight of a positive pair $\{\mathbf{x}_i,\mathbf{x}_j\}\in P_i^+$ is given by:

$$\left|\frac{\partial\mathcal{L}_{MS}}{\partial S_{ij}}\right|^{+} = \frac{1}{e^{-\alpha(\lambda-S_{ij})}+\sum_{l\in P_i^+}e^{-\alpha(S_{il}-S_{ij})}} \tag{6}$$

and the weight of a negative pair $\{\mathbf{x}_i,\mathbf{x}_j\}\in N_i^-$ is given by:

$$\left|\frac{\partial\mathcal{L}_{MS}}{\partial S_{ij}}\right|^{-} = \frac{1}{e^{\beta(\lambda-S_{ij})}+\sum_{l\in N_i^-}e^{\beta(S_{il}-S_{ij})}} \tag{7}$$

From Eq. 6, it can be observed that self-similarity (i.e., $e^{-\alpha(\lambda - S_{ij})}$) and relative similarity with other positive pairs (i.e., $e^{-\alpha(S_{il} - S_{ij})}$) are jointly used to compute the weight of a positive pair [11]. Similar procedures apply for computing a negative pair weight, as in Eq. 7. These pair weights assist the loss function by assigning higher weights to the informative pairs that violate self-similarities and relative similarities.

Similarly, the positive and negative pairs weights for the modified multi-similarity loss defined in Eq. 2 obtained by computing gradient w.r.t S_{ij} are given in Eq. 8 and Eq. 9.

$$\left| \frac{\partial \widehat{\mathcal{L}}_{MS}}{\partial S_{ij}} \right|^{+} = \frac{w_{ij}}{e^{-\alpha(\lambda - S_{ij})} + \sum_{l \in P_i^+} w_{il} \cdot e^{-\alpha(S_{il} - S_{ij})}} \tag{8}$$

$$\left| \frac{\partial \widehat{\mathcal{L}}_{MS}}{\partial S_{ij}} \right|^{-} = \frac{w_{ij}}{e^{\beta(\lambda - S_{ij})} + \sum_{l \in N_i^-} w_{il} \cdot e^{\beta(S_{il} - S_{ij})}} \tag{9}$$

These pair weights employ pair-wise confidence and uncertainty to weigh the violation of its self-similarity and relative similarity. This aids the loss function in assigning higher weights to the more confident and less uncertain informative pairs.

2. Implementation Details

For the classification network, we use Inception-V1 [9] pre-trained on ImageNet [8] as CNN base (f_{cls}), a fully connected layer (g_{cls}) of length 512 and a classification layer (h_{cls}) of size equal to the number of clusters (C). The number clusters used for datasets CUB, Cars, and SOP is 100, 100, and 10000, respectively, as commonly done in the unsupervised metric learning [1, 3, 5]. We added dropout to the third and fourth inception blocks of the Inception-V1 network. To train the classification network, we use the categorical cross-entropy loss, Adam optimizer with an initial learning rate of $1e^{-4}$, and cosine annealing [7] for learning rate decay.

For the embedding network, for a fair comparison, we use Inception-V1 [9] pre-trained on ImageNet [8] as CNN base (f_{emb}) and a 128/512 dimensional embedding layer (g_{emb}). We use Adam optimizer with an initial learning rate of $5e^{-5}$, $3e^{-5}$, and $1e^{-5}$ for the datasets CUB, Cars, and SOP, respectively, with cosine annealing decay to train the embedding network. For data augmentation, the training images are randomly cropped to 227×227 and are horizontally flipped. The testing images are resized to 256×256 followed by a single center crop. For all the experiments, the values of τ, k and T are set to 3, 5 and 15.

Batch construction for training the classifier and the embedding network: We use a mini-batch of size 120 and followed the sampling strategy given in [11]. The mini-batch

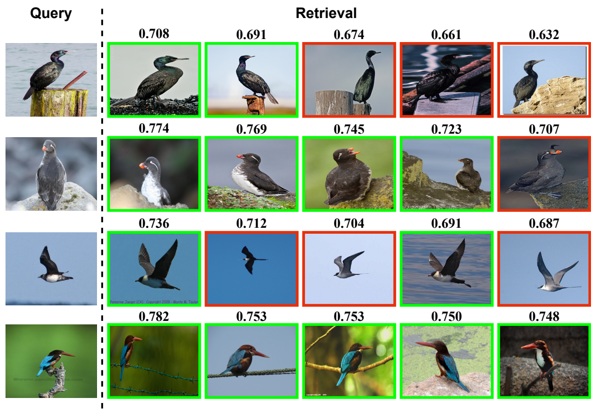

Figure 1. Retrieval results along with the cosine similarity of some randomly selected examples from the CUB [10] dataset with the proposed framework. The green and red frames indicate the positive and negative retrieved results.

is constructed by randomly choosing a certain number of clusters and sampling M examples from each cluster. For all the experiments, the value of M is set to 4 for CUB and Cars datasets and 2 for the SOP dataset.

3. More Ablation Studies

This section provides qualitative results on the CUB [10] dataset to support our framework (UGML). We also analyze the impact of hyperparameters dropout rate, number of stochastic forward passes (T), number of nearest neighbors, Gaussian kernel width, and the number of mini-batches used for neighborhood aggregation on the proposed framework.

3.1. Qualitative Results

In Fig. 1, we show the retrieval results of few randomly selected queries from the CUB [10] dataset. The results show that most top-ranked retrieval results are from the same category as the query example. Further, the negative retrieved results look visually similar to the query, though they are from different classes. This shows the effectiveness of the proposed framework in learning discriminative embeddings. We compare the retrieval results of UGML against spatial assembly network (SAN) [6] for the CUB dataset. As shown in Fig. 2, UGML retrieval results are comparable to those of SAN.

3.2. Impact of the number of stochastic forward passes (T)

To analyze the impact of the number of stochastic forward passes (T), we experimented with different values of T by fixing the dropout probability at 0.2. As shown in Fig. 3, T has very little to no influence on the performance of UGML.

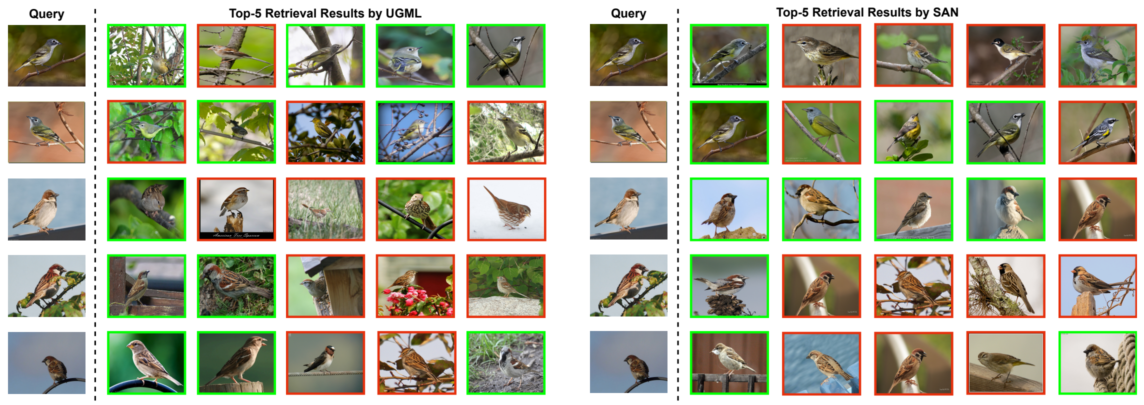
Query
Top-5 Retrieval Results by UGML
Query
Top-5 Retrieval Results by SAN

Figure 2. Comparison of UGML retrieval results with SAN [6] in CUB [10] dataset. The green and red boundaries indicate the positive and negative retrieved results.

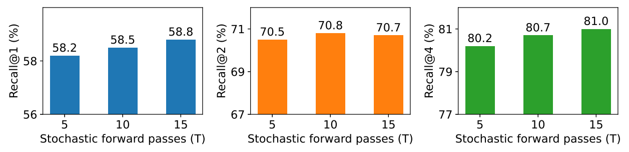

Figure 3. Impact of the number of stochastic forward passes (T) on the performance of UGML with the CUB [10] dataset.

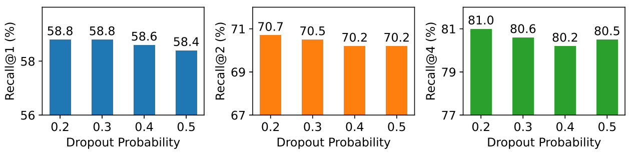

Figure 4. Impact of dropout on the performance of UGML with CUB [10] dataset.

3.3. Impact of the dropout

Fig. 4 illustrates the impact of dropout probability on the performance of the UGML using SCDA features on the CUB dataset. All the results are obtained by considering 15 stochastic forward passes, i.e., ($T = 15$). The proposed framework achieves the best results with a dropout probability of 0.2. Further, there is a drop in the performance of UGML for higher values of dropout probability.

3.4. Impact of the number of nearest neighbors (k) and Gaussian kernel width (τ)

The number of nearest neighbors (k) and Gaussian kernel width (τ) are hyperparameters used in neighborhood aggregation. To analyze the impact of k, τ on the performance of UGML, we experimented with various choices for k and τ on the CUB dataset with SCDA features. From Fig. 5, UGML performs reasonably well with various choices for k and τ while achieving best when k and τ are set to 5 and 3, respectively.

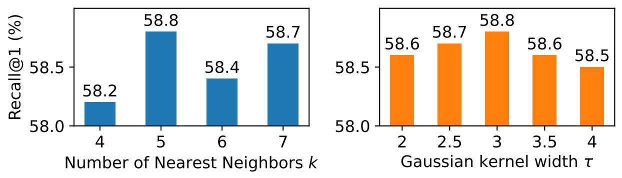

Figure 5. Impact of the number of nearest neighbors k and Gaussian kernel width τ on the performance of UGML with CUB dataset.

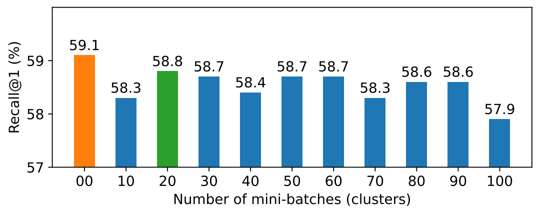

Figure 6. Impact of the number of mini-batches (clusters) used for neighborhood aggregation on the performance of UGML.

3.5. Impact of the number of mini-batches used for neighborhood aggregation

As mentioned in the main paper, using the entire dataset is the ideal choice for neighborhood aggregation to refine the pseudo-labels. However, using the full dataset does not scale for large datasets like SOP. To overcome this, we have proposed a new approach to construct a mini-batch by clustering classes that are close to each other in the feature space of the classification model. To analyze the influence of the number of clusters (or mini-batches) used on the performance of the UGML, we experimented on the CUB dataset with various choices for the hyperparameter number of clusters (or mini-batches). From Fig. 6, as expected, the best result is achieved when the entire dataset is used for neighborhood aggregation (orange bar). Further, the best result with the proposed batch construction with 20 mini-batches

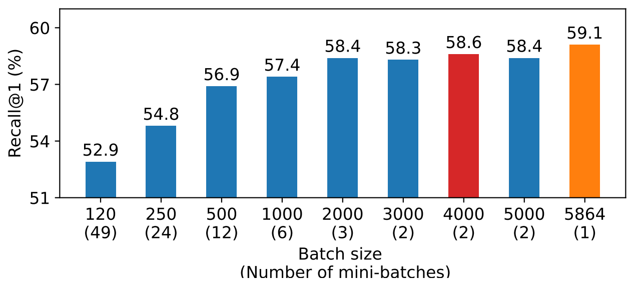

Figure 7. Impact of random sampling with various batch sizes on the performance of UGML.

(green bar) is very close to the result with the full dataset (orange bar). This shows the effectiveness of the proposed batch construction method, as it facilitates scalability while maintaining performance. Also, the performance variation with different choices of the number of mini-batches is minimal, showing the robustness of the UGML.

To further assess the importance of our proposed batch construction strategy, we evaluated UGML using random sampling for batch construction. Given N training examples and batch size q, we form $p = \lceil \frac{N}{q} \rceil$ mini-batches. The first $p - 1$ batches contain q randomly sampled examples, while the remaining examples form the last batch. In Fig. 7, the orange bar represents results with neighborhood aggregation using the full dataset, while the red bar shows the best result with random sampling. As shown in Fig. 7, UGML performs poorly with small batch sizes when using random sampling. Achieving results comparable to those obtained with our proposed batch construction method requires significantly larger batch sizes (≥ 2000 examples per mini-batch) with random sampling, which we are trying to avoid. In other words, random sampling requires fewer mini-batches (≤ 3) with a larger number of examples per batch to match the performance of our proposed batch construction strategy, which achieves better results using more mini-batches (20) with fewer examples per batch (see Fig. 6). This demonstrates the efficiency of the proposed batch construction strategy for neighborhood aggregation, as it optimizes memory usage while maintaining performance.

3.6. Impact of different embedding sizes:

The results on the CUB dataset using SCDA features with different embedding dimensions are given in Table 1. The results show that performance improves as the embedding dimension grows from 64 to 1024.

3.7. Performance with different network architectures as a base

In the paper, the performance of the proposed method (UGML) is evaluated using Inception-V1 [9] as the base. However, different network architectures can be used as a base for UGML. To demonstrate this, we have evaluated the UGML using Inception-V2 [2] as the base, and the results on the CUB dataset using SCDA features are given in Table 2. The results show that the Recall@1 of the UGML is improved by 6.5% compared to 58.8% with Inception-V1 as the base.

Table 1. Performance on CUB dataset with Inception-V1 backbone for different embedding sizes.

	CUB		
Embedding Size	R@1	R@2	R@4
64	51.8	64.8	75.6
128	55.7	67.8	77.8
256	58.0	69.4	79.7
512	58.8	70.7	81.0
1024	59.8	71.2	81.4

Table 2. Performance on CUB dataset using Inception-V2 (Inception with BatchNorm) [2] as the base.

	CUB		
Method	R@1	R@2	R@4
UDML-SS [1]	63.7	75.0	83.8
STML [4]	**68.0**	**78.8**	**86.4**
UGML(SCDA)	65.3	76.5	84.7

References

[1] Xuefei Cao, Bor-Chun Chen, and Ser-Nam Lim. Unsupervised deep metric learning via auxiliary rotation loss. *arXiv preprint arXiv:1911.07072*, 2019. 2, 4

[2] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 448–456. PMLR, 2015. 4

[3] Shichao Kan, Yigang Cen, Yang Li, Vladimir Mladenovic, and Zhihai He. Relative order analysis and optimization for unsupervised deep metric learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13994–14003, 2021. 2

[4] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. Self-taught metric learning without labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7431–7441, 2022. 4

[5] Yang Li, Shichao Kan, and Zhihai He. Unsupervised deep metric learning with transformed attention consistency and contrastive clustering loss. In *European Conference on Computer Vision*, pages 141–157. Springer, 2020. 2

[6] Yang Li, Shichao Kan, Jianhe Yuan, Wenming Cao, and Zhihai He. Spatial assembly networks for image representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13876–13885, 2021. 2, 3

[7] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 2

[8] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy,

Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015. 2

[9] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 2, 4

[10] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 2, 3

[11] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5022–5030, 2019. 1, 2