GIF: Generative Inspiration for Face Recognition at Scale

Supplementary Material

6. CE Derivative

Considering layer-peeled model to make a tractable analysis [15, 68], the gradient of Equation 1 w.r.t. the $\mathbf{w}_j$ is:

$$\frac{\partial L_{CE}}{\partial \mathbf{w}_j} = \sum_{i=1}^{n} \left[-(1 - p_j(\mathbf{z}_i))\mathbf{z}_i \delta(j, y_i) + p_j(\mathbf{z}_i)\mathbf{z}_i(1 - \delta(j, y_i)) \right], \tag{11}$$

here $p_j(\mathbf{z})$ is the predicted probability that $\mathbf{z} = F_\theta(\mathbf{x})$ belongs to the j-th class and $\delta(i, j)$ is one if i is equal to j and 0 otherwise. We can reformulate the Equation 11 to the following form:

$$-\frac{\partial L_{CE}}{\partial \mathbf{w}_j} = \mathbf{f}_{\text{pull}} + \mathbf{f}_{\text{push}}, \tag{12}$$

where $\mathbf{f}_{\text{pull}}^{(\mathbf{w}_j)} = \sum_{i=1}^{n^+} [(1 - p_j(\mathbf{z}_i))\, \mathbf{z}_i]$, $\mathbf{f}_{\text{push}}^{(\mathbf{w}_j)} = -\sum_{i=1}^{n^-} p_j [(\mathbf{z}_i)\, \mathbf{z}_i]$, n^+ represents samples belonging to the j-th class, *i.e.*, positives, and n^- denotes the samples from other classes, *i.e.*, negatives. Equation 12 reveals that CE pulls $\mathbf{w}_j$ toward the positive instances, *i.e.*, n^+, while pushing $\mathbf{w}_j$ away from negative ones, *i.e.*, n^-.

Large-scale FR benchmarks follow an imbalanced distribution where some identities have plenty of instances, while others only contain a few, *i.e.*, $n^+ \ll n^-$ [12, 50, 76]. Consequently, the optimization of $\mathbf{w}_j$ of minority classes is predominantly influenced by $\mathbf{f}_{\text{push}}$. Additionally, $\mathbf{f}_{\text{push}}$ is approximately uniform across all minority classes and forces them to the same subspace [12, 68]. Thus, centroids of minority classes merge, *i.e.*, dubbed '*minority collapse*', lowering inter-class discrimination and metric-space exploitation. Despite the progress of available methods [2, 32], the '*minority collapse*' issue remains unsolved as long as identity centroids' optimization remains dependent on the number of per-identity instances.

7. Ablation on λ

Here, we examine the impact of varying each λ_j on the training process. Each λ_j quantifies the relative importance of the j-th token during training, where a higher λ_j indicates greater impact of the corresponding token, and a lower λ_j suggests less importance. We conducted a series of ablation studies using the IJB-B and IJB-C datasets with the WebFace4M training set and a ResNet-100 backbone. Our objective was to isolate the influence of the length of the codes; hence, we utilized only L_C as the training signal. We explored four distinct patterns in the distribution of λ_i: increasing, decreasing, Gaussian, and uniform, as depicted in Figure 6a. For each configuration, we normalized the λ_i values so that their sum equals one. The results,

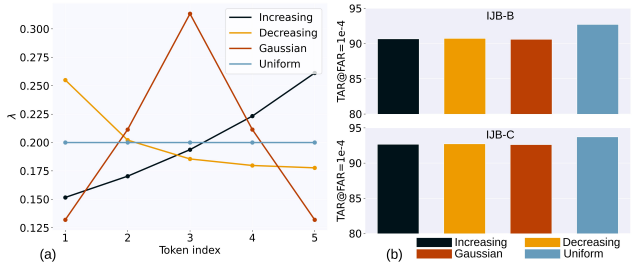

Figure 6. a) Showing the value of the λ for each token index in different scenarios. b) GIF performance is the best when the balancing factor of tokens, *i.e.*, λ, is uniform across tokens.

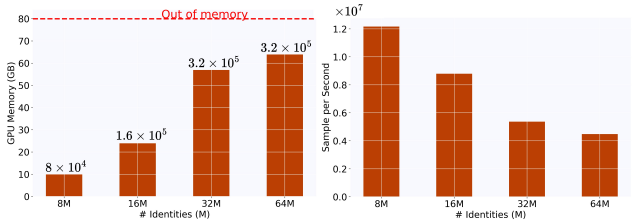

Figure 7. GPU memory consumption (a) and Training speed (b) needed for optimization of code vectors.

shown in Figure 6b, indicate that a uniform of λ across token indices yields superior performance compared to non-uniform. This finding aligns with our identity tokenization strategy, wherein the search space is sequentially narrowed with each correctly predicted token $c_j^{y_i}$ of the identity code $\mathbf{c}^{y_i}$. Based on these findings, we employed a uniform $\lambda = \frac{1}{l}$ in our experiments.

8. Code Vector Optimization Cost

Here, we investigate the GPU memory consumption associated with the optimization of code vectors. As demonstrated in Figure 7a, even with the number of identities reaching 64 million, the GPU memory usage remains significantly lower than the OOM threshold. Moreover, Figure 7b shows the remarkable speed of the optimization in this optimization. This substantial reduction in memory consumption, coupled with improvements in processing speed and batch size, can be attributed to the fact that code vector optimization does not depend on the dataset or backbone architecture. Consequently, the optimization in Equation 6 bypasses the time-intensive tasks of image loading and executing feedforward and backward passes through the backbone.

9. Replacing CLIP with DINO

In this study, we explore the sensitivity of GIF to changes in the model used for initializing code vectors. We con-

Method	Train Set	LFW	CPLFW	CALFW	CFP-FP	Age-DB	IJB-B	IJB-C
GIF (DINO)	MS1MV2	**99.85**	**94.47**	96.75	98.75	**98.67**	94.98	**96.80**
GIF (CLIP)	MS1MV2	**99.85**	94.45	**96.94**	**98.80**	98.58	**95.05**	96.77
GIF (DINO)	WebFace4M	99.83	94.97	**96.92**	**98.41**	**98.63**	**96.95**	97.76
GIF (CLIP)	WebFace4M	**99.85**	**95.03**	96.85	98.36	**98.55**	96.90	**97.83**

Table 4. Performance comparison to when we substitute CLIP with DINO for initializing the code vectors. Verification accuracy (%) is reported for LFW, CFP-FP, and AgeDB. TAR@FAR= $1e-4$ is reported for IJB-B and IJB-C.

duct experiments employing the DINO [7] representation, adjusting our embedding dimension to $d = 718$ to accommodate this model. Results presented in Table 4 confirm our expectations: GIF demonstrates robustness to the specific pretrained model used to initialize the code vectors. Models trained on large datasets with the objective of developing a generalized representation, such as CLIP or DINO, prove adequate for initializing the code vectors since the proposed method solely needs the meaningful order of similarity from initialized code vectors.