Group 9: MagFormer: Parameter-Efficient Modular Approach for Person Re-Identification

Susim Mukul Roy*
Dept. of CSE
susimmuk
susimmuk@buffalo.edu

Harshvardhan Bhosale*
Dept. of CSE
hbhosale
hbhosale@buffalo.edu

Senan Demel*
Dept. of CSE
senandem
senandem@buffalo.edu

Abstract

Person re-identification across diverse environments and camera views requires robust, discriminative features, but existing methods often ignore relational cues between images, leading to noisy or unstable representations. In addition, a large network needs to be trained every time a new sample is introduced. To address this, we propose MagFormer, a modular transformer-based model that incorporates image-to-image interactions during training. It introduces three efficient components: (1) Magnitude-Aware Landmark Agent Attention (MALAA) to approximate dense relations using compact magnitude-aware landmarks, (2) Reciprocal Neighbor Softmax(RNS) to sparsify attention by focusing on contextually relevant samples, and (3) Differential Noise Canceler(DiffAttn) to suppress residual noise and enhance identity consistency. MagFormer is scalable, interpretable, and consistently outperforms baselines on benchmarks. Code: https://github.com/SusimRoy/MagFormer

1 Introduction

Person re-identification (re-ID) aims to match a person with the same identity as a given query across disjoint camera views and different timestamps. Thanks to the discriminative features learned from deep neural networks, significant achievements have been made in this task. However, one of the main challenges in Re-ID is that any individual typically undergoes significant variations in their appearance due to extrinsic factors, like different camera settings, lighting, viewpoints, occlusions, or intrinsic factors like dress changing, to name a few examples. As a result, there are high intra-identity along with high inter-identity variations in the representations corresponding to a specific individual, leading to unstable matching and sensitivity to outliers.

2 Related Work

2.1 Feature Representation Learning Methods

Most person Re-ID methods[3][8] learn features from individual images: they either use part-based modeling (via human-part detectors or horizontal stripes) or embed attention (pixel-level, channel-wise, background suppression) to focus on discriminative regions. Complementary approaches enrich training data—by generating adversarial occlusions or using GANs to synthesize auxiliary views—so that, even from single images, the model learns more robust, discriminative representations

2.2 Transformer and Related Applications

Transformers[5] have gained popularity in various computer vision tasks due to their ability to model long-range dependencies and capture global context. In the context of person Re-ID, transformers

*Equal Contribution

Figure 1: The above picture demonstrates different samples from our dataset where our task is to identify the same individual across different environmental settings.

have been utilized to model relationships between different images, aiming to suppress outlier features and achieve more robust representations. However, modeling interactions between a large number of images poses computational challenges. To address this, methods like the Magnitude-Aware Landmark Agent Attention (MALAA) and Reciprocal Neighbor Softmax (RNS) have been proposed to efficiently model relations and achieve sparse attention to relevant neighbors only, thereby alleviating interference from irrelevant representations and reducing computational burden.

3 Magnitude Transformer Network

Re-ID is usually treated as an image retrieval task, where the goal is to match images of the same person. Let the training dataset be $T = \{(x_i, y_i)\}_{i=1}^{N^T}$, where x_i is the i-th image and $y_i \in S_T$ is its identity label. The set S_T contains all identities in the training data.

We train a model $f()$ that extracts features from images. For each input image x_i, the model outputs a feature vector $z_i = f(x_i)$. These feature vectors help the model tell apart different people by learning to group similar identities and separate different ones. At test time, we are given a query set $U = \{x_i\}_{i=1}^{N^U}$, which includes images of new people the model hasn't seen during training. We also have a gallery set $G = \{x_j\}_{j=1}^{N^G}$, which acts like a database. The goal is to find images in G that match the people in U by comparing their feature vectors. An important point is that the identities in the test set are not part of the training set. This means the identity set for the queries, S_U, and the training set, S_T, are completely separate: $S_U \cap S_T = \emptyset$.

3.1 Magnitude-Aware Landmark Agent Attention

A major challenge in Re-ID is effectively modeling relations between a large number of image representations, where noise, distractions, and identity variations can deteriorate learned embeddings. Magnitude-Aware Landmark Agent Attention (MALAA) is an efficient and interpretable attention mechanism that addresses this by approximating global interactions across identities using a compact set of landmark agents.

MALAA uses a low-rank factorization strategy instead of computing a full pairwise affinity matrix that scales with the number of images and can include noisy interactions. In high-dimensional embedding spaces typical for deep Re-ID pipelines, manually computing full pairwise attention across all samples scales poorly with the number of input images N resulting in $\mathcal{O}(N^2 d)$ time and memory complexity. MALAA mitigates this by compressing the relational structure among embeddings using a small set of learnable landmark agents $L = \{l_1, l_2, \ldots, l_p\} \in \mathbb{R}^{p \times d}$, where $p \ll d$ thereby $\mathcal{O}(N^2 p)$. Each landmark acts as a representative anchor in the feature space for a subset of samples with similar feature magnitude and semantic structure. However, each landmark might not be of the same sample. Hence, we want to achieve different gradient decrease directions for each sample. To do this, we decouple the magnitude from the direction of the each landmark in the query and the key matrics.

As shown in Figure 3, the query, key and value matrices $\mathbf{q}, \mathbf{k}, \mathbf{v} \in \mathbb{R}^{N \times d}$ are obtained by three separate linear projections $\varphi_q(\cdot), \varphi_k(\cdot), \varphi_v(\cdot)$ using representation vectors $\mathbf{z} \in \mathbb{R}^{N \times d}$ as input.

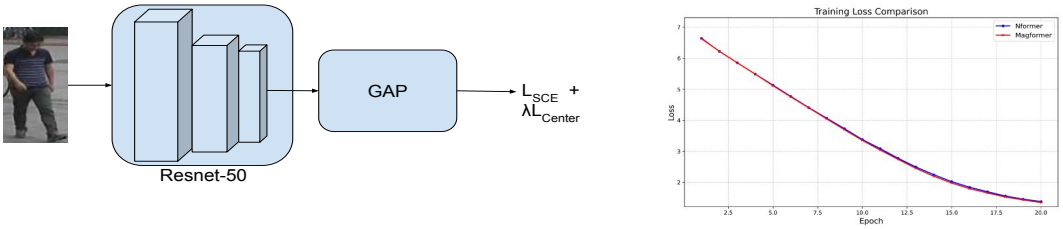

Figure 2: The figure on the left is the Stage-1 of our algorithm where we train our backbone. The figure on the right shows the training loss comparison between our algorithm and the compared baseline.

Specifically, we randomly sample p representations $\mathbf{z}_p \in \mathbb{R}^{p\times d}$ from $\mathbf{z}$ as landmark agents, and then obtain the $\mathbf{q}_p$ and $\mathbf{k}_p$ matrices with $\varphi_q(\cdot)$ and $\varphi_k(\cdot)$. Thus we could map the original query and key matrices $\mathbf{q}, \mathbf{k} \in \mathbb{R}^{N\times d}$ to a p-dimensional space by

$$\tilde{\mathbf{q}} = \frac{\mathbf{q}\mathbf{k}_p^\top}{\|k_p^T\|_2}, \quad \tilde{\mathbf{k}} = \frac{\mathbf{k}\mathbf{q}_p^\top}{\|q_p^T\|_2},$$

where $\tilde{\mathbf{q}}, \tilde{\mathbf{k}} \in \mathbb{R}^{N\times p}$. $\tilde{q}_{ij}, \tilde{k}_{ij}$ indicate the similarity between representation vector $i \in \{1, \ldots, N\}$ and landmark agent $j \in \{1, \ldots, p\}$. Also, we have the learnable magnitude vectors $m_1 \in N^{p\times 1}$ and $m_2 \in N^{1\times p}$ for the query and the key matrices respectively. Therefore, we get the final affinity matrix representation as:

$$\tilde{A}_{ij} = \frac{(m1 \cdot \tilde{q})_i (m2 \cdot \tilde{k})_j}{\sqrt{d}} \tag{1}$$

In this way, we decompose the computation of the large affinity map $A \in R^{N\times N}$ into a multiplication of two low- rank matrices q, k. We also observe that this procedure stabilizes the training process, as shown in Figure 2 which leads to better numerical results.

3.2 Reciprocal Neighbor Softmax

Once we compute the approximate affinity matrix $\mathbf{A}$, we usually apply a softmax function s to convert affinities into attention weights (i.e., probabilities), which can be written as two parts:

$$u_i = \sum_{j:A_{ij}\le\rho} s(\tilde{A})_{ij}\varphi_v(z_j) + \sum_{j:A_{ij}>\rho} s(\tilde{A}_{ij})\varphi_v(z_j) \tag{2}$$

Here, ρ is a small threshold. The first term adds up elements with small attention values, whereas the second term sums those with large attention values. Even though each weight in the first sum (where $\tilde{A}_{ij} \le \rho$) is small, as the number of samples N increases, the total value of this sum becomes large. This makes it comparable to the second term, and as a result, irrelevant samples can significantly influence the final output vector $\mathbf{u}_i$. In addition to hurting performance, the cost of this aggregation step is $O(N^2 d)$, which can be very expensive for large input sizes N.

To solve these problems, we introduce Reciprocal Neighbor Softmax (RNS), which adds sparsity by focusing attention only on a few relevant neighbors using a reciprocal neighbor mask. The idea is if two images are top-k neighbors of each other in the feature space, then they are likely to be related. To create this mask, we first define a top-k neighbor mask $\mathbf{M}^k$ from the affinity matrix $\tilde{\mathbf{A}}$ as follows:

$$\mathbf{M}_{ij}^k = \begin{cases} 1, & \text{if } j \in \text{topk}(\widetilde{\mathbf{A}}_{i,:}) \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Next, we calculate the reciprocal neighbor mask $\mathbf{M}$ by multiplying $\mathbf{M}^k$ with its transpose element-wise (Hadamard product):

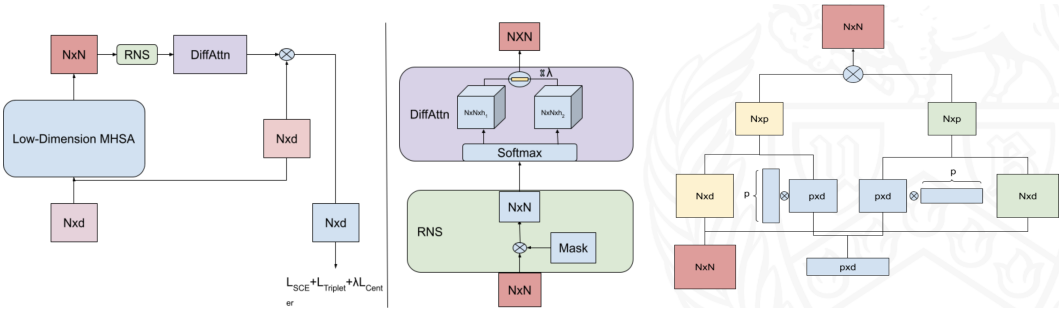

Figure 3: The above figure shows the Stage-2 of our pipeline. We obtain the $N \times d$ matrix from the trained Resnet-50 of our stage-1 and thereby produce a better representation of the feature vectors through the use of Low-Dimension MHSA, RNS and DiffAttn modules.

$$\mathbf{M}_{ij} = \mathbf{M}_{ij}^{k} \cdot \mathbf{M}_{ji}^{k} = \begin{cases} 1, & \text{if } j \in \text{topk}(\widetilde{\mathbf{A}}_{i,:}) \text{ and } i \in \text{topk}(\widetilde{\mathbf{A}}_{:,j}) \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

This mask sets $\mathbf{M}_{ij} = 1$ only when i and j are among each other's top-k neighbors. Otherwise, it is 0. By applying this mask to the softmax, attention is only given to relevant neighbors. The RNS function is then finally defined as:

$$\text{RNS}(\widetilde{\mathbf{A}})_{ij} = \frac{\mathbf{M}_{ij} \cdot \exp(-\widetilde{\mathbf{A}}_{ij})}{\sum_{k} \mathbf{M}_{ik} \cdot \exp(-\widetilde{\mathbf{A}}_{ik})} \tag{5}$$

As shown in Figure 4, this leads to many zero attention values, forcing the network to only consider useful neighbors. This makes the aggregation in equation (2) more accurate and robust. Since we skip any summation over zero-weighted samples, the complexity drops from $O(N^2 d)$ to $O(Nkd)$.

3.3 Differential Attention

The differential attention mechanism maps query, key, and value vectors to outputs. We use query and key vectors to compute attention scores, and then compute a weighted sum of value vectors. The critical design is that we use a pair of softmax functions to cancel the noise of attention scores.

Specifically, given input $X \in \mathbb{R}^{N \times d_{\text{model}}}$, we first project it to query, key, and value $Q_1, Q_2, K_1, K_2 \in R^{N \times d}, V \in R^{N \times 2d}$.Thus, the operator is as follows:

$$[Q_1; Q_2] = XW_Q, \quad [K_1; K_2] = XW_K, \quad V = XW_V$$

where $W_Q, W_K, W_V \in \mathbb{R}^{d_{\text{model}} \times 2d}$ are learnable parameters and λ is a learnable parameter. In order to synchronize the learning dynamics, we re-parameterize the scalar λ as:

Then, the differential attention operator $\text{DiffAttn}(\cdot)$ computes the output as:

$$\text{DiffAttn}(X) = \left(\text{softmax}\left(\frac{Q_1 K_2^{\top}}{\sqrt{d}} \right) - \lambda \cdot \text{softmax}\left(\frac{Q_2 K^{\top}}{\sqrt{d}} \right) \right) V$$

The scalar λ is re-parameterized to synchronize the learning dynamics:

$$\lambda = \exp(\lambda_{q1} \cdot \lambda_{k1}) - \exp(\lambda_{q2} \cdot \lambda_{k2}) + \lambda_{\text{init}}$$

where $\lambda_{q1}, \lambda_{k1}, \lambda_{q2}, \lambda_{k2} \in \mathbb{R}^d$ are learnable vectors, and $\lambda_{\text{init}} \in (0, 1)$ is a constant used for initialization. We empirically find that the setting $\lambda_{init} = 0.8 - 0.6 \times exp(-0.3 \times (l-1))$ works well in practice, where $l \in [1, L]$ represents layer index. It is used as the default strategy in our

experiments.
Differential attention takes the difference between two softmax attention functions to eliminate attention noise. The idea is analogous to differential amplifiers[1] proposed in electrical engineering, where the difference between two signals is used as output, so that we can null out the common-mode noise of the input. In addition, the design of noise-canceling headphones is based on a similar idea

4 Experimental Settings

4.1 Model Description

We use ResNet-50 as our feature extractor, initialized with ImageNet-pretrained weights and fine-tuned on each dataset. To preserve spatial information, we replace the stride convolutions in the final stage with dilated convolutions, which resulted in a downsampling factor of 16. After the backbone, A fully connected layer follows to reduce the feature dimension from 2048 to 512 for better efficiency. In the MALAA module, we use 5 landmark agents, and for RNS, we set $k = 20$ to balance accuracy and speed based on experiments.

To ensure fair inference, query images do not interact. We train ResNet-50 and MagFormer separately for 160 epochs each. ResNet-50 uses a batch size of 128, and MagFormer is trained with a batch size of 2048, enabled by freezing the ResNet-50 parameters. All experiments are conducted in PyTorch on a single NVIDIA H100 GPU.

4.2 Dataset and Protocols

We tested our method on three popular large-scale person Re-ID datasets: Market1501 [7] with 750 training identities and 12,936 training images and 751 testing identities, DukeMTMC-reID [4] with 702 training and testing identities which makes up the 16,522 training images, and CUHK03-L [2] with 767 training identities and 7,368 training images and 700 testing identities. These datasets include multiple images per identity from various cameras, making them ideal benchmarks. To ensure fair comparison, we followed standard protocols. During training, images were resized to 256×128, and augmented with random horizontal flips, padding, random crops, and random erasing. For validation and testing, we only resized images to 256×128 to maintain consistency without extra augmentation.

4.3 Loss Functions

In training the first stage of our network, we used a combination of the Smooth Cross-Entropy Loss and the Center Loss. This is because the former allows inter-class separation, ensuring different identities occupy distinct regions of the feature space while the latter drives intra-class compactness, tightening each identity's cluster and reducing "within-class scatter". The *center loss* addresses this by maintaining a learnable prototype (or "center") $\mathbf{c}_k \in \mathbb{R}^d$ for each class k, and penalizing the Euclidean distance between each feature $\mathbf{f}_i = f(\mathbf{x}_i) \in \mathbb{R}^d$ and its corresponding class center:

$$L_{\text{center}} \;=\; \frac{1}{2} \sum_{i=1}^{N} \big\| \mathbf{f}_i \;-\; \mathbf{c}_{y_i} \big\|_2^2.$$

By jointly optimizing this term with a standard softmax (or cross-entropy) loss, networks learn embeddings that are not only discriminative across classes but also exhibit reduced intra-class variance. In the second stage of our network, we use the same loss function as above and also combined triplet loss with it. This is because we need to ensure that every positive pair sits closer than every negative pair and triplet loss helps in alleviating this problem. The *triplet loss* encourages an embedding function $f : \mathcal{X} \to \mathbb{R}^d$ to map samples of the same identity closer together than those of different identities by at least a margin $m > 0$. Given an anchor sample $\mathbf{x}_a$, a positive sample $\mathbf{x}_p$ (same class), and a negative sample $\mathbf{x}_n$ (different class), the loss for one triplet is defined as

$$L_{\text{triplet}} \;=\; \max\big\{ \, \|f(\mathbf{x}_a) - f(\mathbf{x}_p)\|_2^2 \;-\; \|f(\mathbf{x}_a) - f(\mathbf{x}_n)\|_2^2 \;+\; m,\; 0 \big\}.$$

By minimizing this over many triplets, the model learns to enforce a margin between intra-class and inter-class distances. The three loss functions are weighted by 1, 1, 0.0005 respectively.

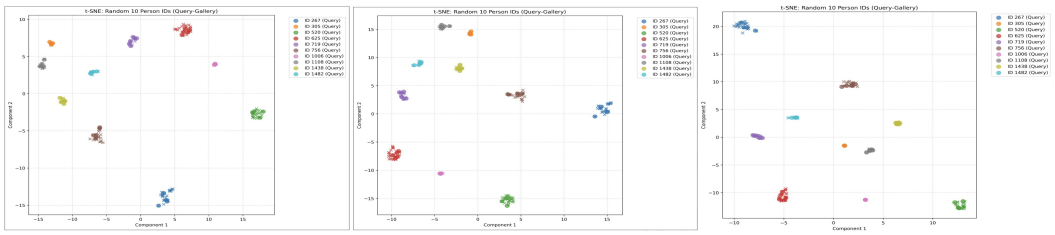

Figure 4: The above three figures shows the t-SNE visualization of the feature space. The plot on the left, middle and right shows the clusters obtained by the baseline, our midterm and our final approach respectively.

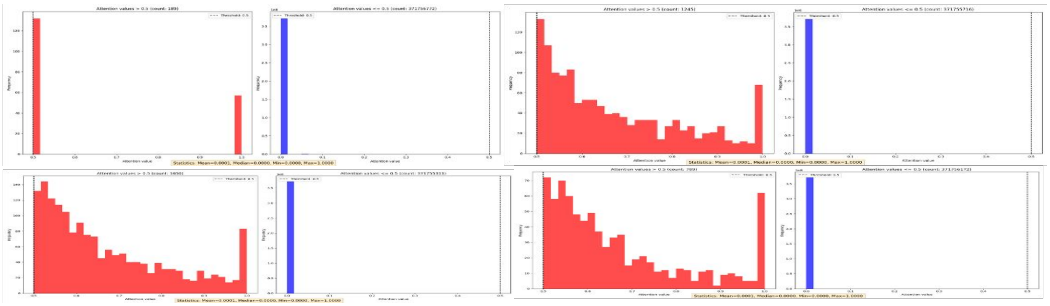

Figure 5: Four figures showing attention scores of the last layer of our encoder module. Red bars show frequency of attention scores between $0.5 - 1$ and blue bars show frequency of attention scores between $0 - 0.5$.

4.4 Optimization Algorithm

For training the first stage of our pipeline, we used the Adam optimizer for the optimizing the network weights while the SGD optimizer for optimizng the parameters of the Center Loss function. In the second stage, we used the AdamW optimizer to train the network parameters while SGD was used to train the center loss function parameters. The learning rate for AdamW was $1.5 \times 1e - 4$, betas were $0.9, 0.98$ and weight decay was 0.05. In case of SGD, in both the cases we used a learning rate of 0.5. Finally, for Adam optimizer we used the default hyperparameters. We also tried other optimizers like RMSProp etc. along with different sets of hyperparameters. However, we found empiricially that the mentioned setting worked best.

5 Experimental Results

5.1 Metrics

To evaluate performance, we follow standard practices using two widely adopted metrics: the Cumulative Matching Characteristic (CMC) curve and mean Average Precision (mAP). The CMC curve evaluates how well the system ranks the correct identity. Specifically, it reports the accuracy at different rank levels (i.e. rank-1 or rank-5) by checking whether the correct match appears within the top-K candidates in the ranking list. This provides insight into the model's ability to rank the correct identity near the top. However, mAP offers a more comprehensive measure of retrieval performance. It computes the average precision over all queries and then takes the mean. This captures the area under the precision-recall curve. Unlike CMC, which focuses on the top matches, mAP evaluates the model's effectiveness over the full ranking list, reflecting how well it retrieves all relevant identities in the gallery set.

5.2 Qualitative Results

. In this section, we discuss the ablation studies we did which led to the use of our different components. In Figure 5, the four figures show the attention scores of the layer-wise affinity matrix

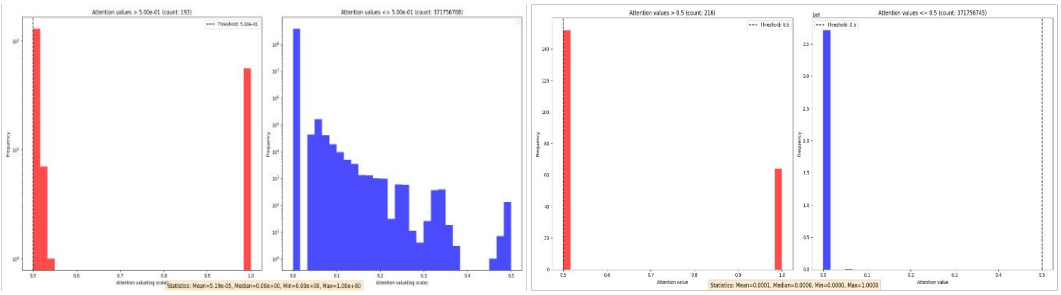

Figure 6: The left and the right figure are the attention score graph of the first head of the last layer obtained without and with DiffAttn respectively.

Table 1: Comparison of various ReID methods on three common benchmarks.

Method	Market-1501		duke-reID		CUHK03-L	
	T-1	mAP	T-1	mAP	T-1	mAP
ResNet-50	93.2	83.5	86.1	76.1	74.7	73.8
+ Nformer[6]	94.7	91.1	89.4	83.5	77.2	78.0
+ MagFormer(Midterm)	94.8	91.7	**89.7**	85.1	78.2	79.6
+ MagFormer(Final)	**94.9**	**92.6**	89.4	**86.1**	**78.2**	**80.5**

obtained after applying RNS but without DiffAttn. We observe that although we have removed the low-attention scores, we are still left with a range of attention scores in 0.5-1. However, we want the model to be more confident in it's predictions. There might be a case where our model is overconfident about some tokens while it has given low-attention value to the actual relevant sample token. Thus, we try to solve this problem with DiffAttn. Therefore, in Figure 6 we observe the change in the attention scores of the first head in the last layer of the encoder module. We see that after applying DiffAttn, our model is either completely sure or says that there is a 50-50 chance(as shown by the left figure). Finally, in Figure 4 we plot the t-SNE of the feature vectors obtained. In the leftmost figure, we observe a lot of dispersion in the clusters especially in the orange one. However, as we move towards the right we observe that they are more centered with the leftmost figure having a almost near perfect centered cluster. Thus, we can visually observe that our features vectors are clustered in a better way in the feature space as per the sample ids.

5.3 Quantitative Results

In Table 1, we observe that our method has surpassed the baseline model in terms of the T-1 and mAP metric on all datasets. Specifically, even with a simple feature extractor Res50, the mAP of MagFormer outperforms the 2022 SOTA in person re-identification. Instead of Resnet-50, if we had used CLIP or ABD-Net, we could have reached the current SOTA too while being parameter efficient by a large margin. This indicates that the relation modeling among all input images both during training and test leads to better representations.

6 Conclusion

In summary, while multi-head self-attention can be powerful, simply increasing the number of heads or layers often leads to attention collapse rather than better discrimination. In contrast, augmenting each identity with more images consistently boosts performance—indeed, datasets like CUHK03-L, which have fewer samples per identity, typically yield lower accuracy. Equally, excessively large feature-vector dimensions may harm both accuracy and efficiency, inflating GFLOPS without gains, whereas modestly shrinking the rank of the query/key projections has minimal impact. Our modular, parameter-efficient attention block sidesteps these issues: it plugs into any backbone, keeps computational overhead low, and excels in large-scale, real-world surveillance scenarios.

References

[1] Phillip A. Laplante, editor. *Comprehensive Dictionary of Electrical Engineering*. CRC Press, Boca Raton, FL, 2nd edition, 2005.

[2] Wei Li, Rui Zhao, Tong Xiao, and Xiaogang Wang. Deepreid: Deep filter pairing neural network for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[3] Wei Li, Xiatian Zhu, and Shaogang Gong. Harmonious attention network for person re-identification, 2018.

[4] Ergys Ristani, Francesco Solera, Roger S. Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *ECCV Workshops*, 2016.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[6] Haochen Wang, Jiayi Shen, Yongtuo Liu, Yan Gao, and Efstratios Gavves. Nformer: Robust person re-identification with neighbor transformer. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7287–7297, 2022.

[7] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1116–1124, 2015.

[8] Zhedong Zheng, Liang Zheng, and Yi Yang. Unlabeled samples generated by gan improve the person re-identification baseline in vitro. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3774–3782, 2017.