Reasoning for Complex Data through Ensemble-based Self-Supervised Learning

Gabriel Bertocco, Antonio Theophilo, Fernanda Andaló, Member, IEEE, and Anderson Rocha, Senior Member, IEEE

Abstract—Self-supervised learning deals with problems that have little or no available labeled data. Recent work has shown impressive results when underlying classes have significant semantic differences. One important dataset in which this technique thrives is ImageNet, as intra-class distances are substantially lower than inter-class distances. However, this is not the case for several critical tasks, and general self-supervised learning methods fail to learn discriminative features when classes have closer semantics, thus requiring more robust strategies. We propose a strategy to tackle this problem, and to enable learning from unlabeled data even when samples from different classes are not prominently diverse. We approach the problem by leveraging a novel ensemble-based clustering strategy where clusters derived from different configurations are combined to generate a better grouping for the data samples in a fully-unsupervised way. This strategy allows clusters with different densities and higher variability to emerge, which in turn reduces intra-class discrepancies, without requiring the burden of finding an optimal configuration per dataset. We also consider different Convolutional Neural Networks to compute distances between samples. We refine these distances by performing contrastive analysis and group them to capture complementary information. We consider two applications to validate our pipeline: Person Re-Identification and Text Authorship Verification. These are challenging applications considering that classes are semantically close to each other and that training and test sets have disjoint identities. Our method is robust across different modalities and outperforms state-of-the-art results with a fully-unsupervised solution without any labeling or human intervention.

Index Terms—Self-Supervised Learning, Deep Learning, Person Re-Identification, Authorship Verification, Model Ensemble.

I. INTRODUCTION

SELF-supervised learning has been gaining increasing attention due to its capacity to learn patterns from unlabeled data. Techniques for self-supervised learning usually rely on contrastive strategies [1], [2], [3], [4], where augmented views of the same image are arranged to be closer in an embedding space but further apart from other images.

It is common practice to consider ImageNet [5] as the dataset of choice for experimentation, as its high number of classes usually translates to general features, providing a good weight initialization for transfer learning. Although ImageNet contains some closer classes (e.g., different dog breeds), most of them have very different semantics (e.g., cars, airplanes, fruits, and flowers) and are easier to distinguish. This enables the learning process to rely more on coarse rather than fine-grained details, and the mere use of well-known augmentation strategies (such as cropping, blurring, erasing, flipping) provides sufficient variation for optimization [1], [2], [3]. Nonetheless, coarse features are insufficient for several critical tasks, requiring fine-grained focus and other strategies to enable effective learning in a self-supervised way. One such example is the Person Re-Identification (ReID) task.

Person ReID aims to retrieve the same person seen in one camera from all the other cameras present in a system. In this scenario, all classes are semantically identical as they represent people in the same environment. Therefore, the model must overcome point-of-view changes, occlusions, different lighting, resolutions, and backgrounds. Moreover, people in a specific place tend to dress similarly. DukeMTMC-ReID [6], a ReID dataset composed of videos from eight cameras at a university campus during winter, presents most of the identities wearing pants, dark coats, backpacks, boots, and other attributes that people usually wear on cold weather or universities. Therefore, most identities (classes) tend to be similar, which means a high inter-class similarity. In contrast, a person might be recorded from different angles by different cameras, which results in a higher intra-class disparity.

Moreover, Person ReID is intrinsically an open-world problem: identities in the test set (query and gallery sets) are disjoint from those in the training set, while ImageNet training and test sets have the same classes. Aside from this, ReID datasets can be as large as ImageNet: Market1501 [7] has 751 identities in the training set, DukeMTMC-ReID [6] has 702, and MSMT17 [8] even exceeds ImageNet with 1041 classes in the training set. This clearly shows that Person ReID is a more demanding task than ImageNet classification, and general state-of-the-art self-supervised learning methods are not suitable for the task. These methods result in much less accurate models when compared to prior unsupervised learning methods tailored specifically to ReID, even with ImageNet weight initialization [9], [10].

In this context, we aim to design a novel self-supervised learning pipeline to handle tasks that require a robust distance measure and fine-grained analysis. We start by considering a common approach: clustering steps to propose pseudo-labels to unlabeled samples and optimization steps to update
the model supervised by those pseudo-labels [10], [11], [12]. However, prior methods that consider this approach often overlook two aspects: the quality of the features and the choice of hyper-parameters for the clustering algorithm. If the features are not too descriptive, samples from different classes might end up closer in the feature space, leading them to be clustered together, increasing the number of false positives and ultimately hindering model updates. Even when the features are adequate, a bad choice of hyper-parameters for the clustering process might yield suboptimal groups.

To address these problems, we take inspiration from re-ranking techniques [13] to filter out false-positive samples. Our proposed pipeline starts by calculating pairwise distances for unlabeled samples, considering features extracted by M Deep Neural Networks (we refer to them as backbones). Those distances are normalized by considering the mutual neighbors in each of the M feature spaces. As a second distance refinement, we average the M distances between two samples, as each backbone can provide a complementary description. Recent works consider ensemble techniques for Unsupervised Domain Adaptation [14], [15]. These studies, however, apply mutual-learning by leveraging complex loss functions with one backbone supervising the other, which brings complexity to the training process. Our method, in turn, ensembles models by only taking the average distances, allowing the amalgamation of complementary information from each manifold, but with a much more straightforward setup.

The second aspect is the choice of hyper-parameters for the clustering process. We take DBSCAN as our clustering algorithm and, instead of fixing a value for the $\varepsilon$ parameter, we scan different clustering densities — the lower the value, the denser the cluster. If a sample is identified as an outlier in any of these levels, it is marked as an outlier in the other levels. By tracking the state of each sample (inliers and outliers) through clustering runs, we can produce clusters that connect different dense regions while disregarding noisy samples.

Our pipeline is designed for general self-supervised learning and, to demonstrate this capability, we consider two critical applications that operate with different types of data: person re-identification and authorship verification from social media texts. The second task is to group tweets (Twitter short messages) of the same author in a fully-unsupervised manner, only considering raw texts as input. To the best of our knowledge, we are the first ones to apply the same self-supervised learning considering raw texts as input. To the best of our knowledge, sages) of the same author in a fully-unsupervised manner, only texts. The second task is to group tweets (Twitter short mes-

In this section, we describe related methods for self-supervised learning, with a more detailed exploration on person re-identification and text analysis.

A. Self-Supervised Learning

Self-Supervised learning is usually carried out by generating two or more views of the same sample through augmentation techniques. A contrastive loss is minimized to pull together different views of the same image while pushing original images of different classes apart.

MoCo [2] generates two random augmented versions for each image of a batch. One is fed to a key encoder and the other to a query encoder for feature extraction. The features from the key encoder are added to a dictionary that stores features from previous batches. The method minimizes a contrastive loss to pull together both augmented versions while keeping other features in the dictionary apart. SimCLR [3] also adopts two augmented versions of each image, but without a dictionary of features. Instead, it minimizes a contrastive loss considering both augmented images as a positive pair and the other images (and their augmented versions) as negative pairs.

SWaV [1] applies a multi-cropping strategy by considering two standard resolution crops and several low-resolution crops for optimization. Like SWaV, Dino [4] adopts different levels of cropping to optimize a teacher-student loss function. They feed the global crops to the teacher network to get a final probability distribution used to supervise the student network, which is fed with local crops.

These methods obtain competitive performance when compared to their supervised learning equivalents. However, as they are generally tested on ImageNet, they tend to fail on problems with high intra-class dissimilarity and inter-class similarity, such as person ReID and authorship verification of short messages.

B. Person Re-Identification

To tackle unsupervised person re-identification, some methods rely on pre-training a model on a source ReID dataset to acquire prior knowledge of the problem. This model is then adapted to the unlabeled target domain.

ECN-GPP [16] is a pipeline that handles example-, camera-, and neighborhood-invariance to enforce images to be closer to their closest samples, by minimizing all three aspects in a loss function. DG-Net++ [17] proposes a pipeline to disentangle ID-related features by generating images on target and source domains with semantic features across both domains. They also consider a clustering algorithm to propose pseudo-labels to unlabeled data to regularize training.

MMT [14] and MEB-Net [15] are ensemble-based methods that leverage two and three backbones, respectively, one supervising the other in a teacher-student regime. They propose soft and hard pseudo-labels to the unlabeled samples to perform optimization. ABMT [11] leverages a teacher-student model, where the global average pooling branch supervises the global max pooling branch, and vice-versa. In a previous work [12]
of ours, we considered ensembles only during evaluation. We generated cross-camera triplets using camera information of samples in the generated clusters. We also proposed a self-ensembling strategy in which the training of each backbone is summarized by weight averaging the checkpoints.

Instead of considering pre-training in the ReID domain, another set of methods relies on other pieces of information, such as camera labels. IICS [18] leverages intra-camera training by dividing samples into sets according to their camera labels and performing clustering on each one. A backbone is trained in a multi-task manner (one task per camera) and clustering is run for the whole dataset, grouping samples of the same identity seen from different cameras. CAP [19] performs global clustering by assigning pseudo-labels for each sample of the dataset. They obtain camera proxies on each cluster for intra- and inter-camera training. ICE [20] has two versions: camera-aware and camera-agnostic. The first one considers the camera proxy features similar to CAP. The second considers only the cluster proxy, which is obtained by averaging features regardless of the camera label. They use a proxy-based loss, along with a hard- and a soft-instance loss.

Instead of camera labels, some works rely on tracklets. CycAs [21] aims to identify the same person on frames of a video for intra-sampling. They also find the same person in other videos by considering the overlapping field-of-view between two videos with inter-sampling. With both intra- and inter-sampling, they optimize a backbone to match the same person from different points-of-view. UGA [22] averages the features of the same person in the same tracklet and performs cross-camera feature association creating a Cross-View-Graph (CVG) to encourage the matching of tracklets of the same person from different points-of-view.

Other works assume that only person bounding boxes are available. These are considered fully unsupervised. ABMT [11] relies on source pre-training, but the authors also present results when no prior knowledge is considered. SpCL [10] proposes a self-paced strategy that introduces metrics to measure cluster reliability: cluster independence and cluster compactness. If both are higher than predefined thresholds, the cluster is kept within the feature space. They also minimize the loss function considering cluster centroids and feature samples stored in feature memory. RLCC [9] refines clusters by a consensus among iterations. Pseudo-labels on a certain iteration are created by considering the ones generated on a previous iteration, keeping the training stable.

Compared to such methods, our pipeline also does not rely on any extra information, requiring only the bounding boxes of the people in the dataset; therefore, it best fits this last category of methods. Nonetheless, prior art often relies on clustering methods with manually chosen optimal hyper-parameter values, which might be impractical when working with unlabeled datasets. Our method differs from the rest by proposing a clustering criterion, which alleviates the burden of choosing optimal hyper-parameters.

C. Unsupervised Text Analysis

Text Analysis is another application that can be explored with unsupervised learning methods.

The Natural Language Processing (NLP) community witnessed a significant development with the introduction of models based on Attention and the Transformer architecture [23]. BERT [24] is one of the most successful of these models, applying an encoder-only architecture to solve many NLP tasks. The authors propose a self-supervised pre-training regime using masked language modeling and next sequence prediction tasks, followed by a fine-tuning step using supervised data. Several works followed BERT, proposing variations using more targeted data. One example is BERTweet [25], in which the authors propose an extension to deal with tweets (short messages from Twitter).

T5 [26] takes a step forward and proposes a single architecture to solve any NLP problem that can be modeled as a text-to-text task. They apply the architecture presented by Vaswani et al. [23] with small changes in the normalization, dropout, and embeddings layers.

A text analysis task that is very similar to Person ReID is authorship verification, which aims to recover, from a gallery, tweets from the same author of a query tweet. It is a genuinely open-set setup since the model may have never seen the actual author, like in ReID. Despite current methods achieving good results for lengthy texts, authorship attribution is still challenging for small texts [27].

In this work, unlike BERT and T5, we target the authorship verification task in a fully unsupervised way, considering a dataset of tweets. We consider a challenging and less explored setup, in which the authors from the training set are unlabeled and disjoint from those in the test set. As our baseline, we consider AdHominem [28], an attention- and LSTM-based model for authorship verification originally trained in a supervised manner using social media posts.

III. Proposed Method

The training pipeline is composed of seven steps: feature extraction and neighborhood-based distance computation, ensemble-based clustering, learning rate update, proxy selection, batch creation, optimization, and Mean Teacher averaging. Figure 1 depicts an overview of these steps.

In the first step, features are extracted for each sample, considering different backbones. We compute pairwise distances between samples based on their neighborhood for each backbone, and average them across backbones to obtain a more refined and unique distance matrix. The second step is the application of our ensemble-based clustering technique to obtain pseudo-labels for the samples. We perform the learning rate calculation in the third step. In the fourth step, a proxy feature vector is selected for each cluster and, in the fifth step, sample batches are created. This information is used during the sixth step, which is the optimization of each backbone, independently. In the last step, a Mean Teacher technique is used to combine the weights of a backbone over training steps in a momentum model, which is used later for inference.

We use the term iteration to refer to one complete iteration of the pipeline (blue flow in Figure 1), and epoch to refer to when the proposed clusters are used for optimization in the current iteration (green flow in Figure 1). We perform $K_1$ iterations and $K_2$ epochs per iteration.
A. Step 1: Feature extraction and neighborhood-based distance computation

Consider a set \( X = \{x_1\}_{i=1}^N \) of unlabeled data points in the target domain, consisting of \( N \) samples, and \( M \) backbones that generate feature representations for these samples.

In prior art for image representation, the output of the last global max or average pooling layer is commonly used as the final feature vector. However, global max and average pooling operations produce distinct and complementary descriptions and, when used together, they can increase the quality of the representation [11]. Following this idea, we perform both global max and average pooling after the last feature map and add the resulting vectors element-wise for the final feature vector (Figure 2). It is important to note that this is only done for images. For text representation, the output of the last layer is directly used as the final representation.

![Fig. 2. The final feature vector (FV) is obtained by extracting both global max pooling (GMP) and global average pooling (GAP) of the previous layer’s output and then adding the two resulting vectors element-wise.](image)

After extracting features for all samples, we L2-normalize them so that they are projected onto a unit hyper-sphere. Therefore, we have a set \( F^m \in \mathbb{R}^{N \times d_m} \) of normalized feature vectors extracted by the \( m \)-th backbone, where \( d_m \) represents the dimension of features in this set.

For each \( F^m \), we calculate pairwise distances for all samples in the set. Inspired by re-ranking techniques, we refine these distances by considering the neighborhood of the samples; i.e., we normalize the distance between two samples based exclusively on the number of neighbors in common: 0 (no neighbors in common) and 1 (all neighbors in common). Consequently, we have \( M \) refined distance matrices \( \{D_1, D_2, \ldots, D_M\} \), one for each backbone.

Each distance matrix \( D_m \) is a general representation of the knowledge obtained with each backbone over the input data, as it is calculated based on the samples’ feature vectors. To explore potential complementary knowledge, we propose averaging all distance matrices:

\[
\overline{D} = \frac{1}{M} \sum_{m=1}^{M} D_m. \tag{1}
\]

The final distance matrix \( \overline{D} \) is used as input to the proposed ensemble-based clustering technique.

B. Step 2: Ensemble-based Clustering

DBSCAN [30] clustering is the basis for our ensemble-based clustering. It relies on two hyper-parameters: \( \text{minPts} \) – the minimum number of samples on a point’s neighborhood to consider it as a core point – and \( \varepsilon \) – the size of the neighborhood. Two data points \( p \) and \( q \) are considered neighbors if the distance between them is less than \( \varepsilon \). A data point \( p \) is a core point if it has at least \( \text{minPts} \) neighbors. If it has less than \( \text{minPts} \) but is neighbor to a core point, then \( p \) is a border point. Otherwise, it is considered as an outlier. Two points \( p \) and \( q \) are within the same cluster if a path exists \( P = \{p_0, p_1, \ldots, p_n\} \), where \( \forall_{1 \leq i \leq n-1} p_i \) is a core point, \( p = p_0 \) and \( q = p_n \).

The performance impact of the two hyper-parameters has been studied [31], [11], and the conclusion is that DBSCAN is more sensitive to \( \varepsilon \) than to \( \text{minPts} \). A wrong choice of \( \varepsilon \) can substantially hinder the performance, requiring domain knowledge to select its optimal value. A dataset with high intra-class variability might yield non-convex and sparse clusters, rendering the intra-class data points far away from each other while inter-class samples are closer. To account for this,
a higher \( \varepsilon \) would be needed to group sparse samples in the same cluster. In turn, datasets with lower intra-class variability might require a lower \( \varepsilon \), as a larger value could introduce false positives in the same cluster.

The described problem is common to several Person ReID benchmarks. Market1501 [7] is a dataset that comprises 751 identities recorded from six different cameras in the training set, while MSMT17 [8] has 1041 identities recorded from fifteen different cameras. Identities on MSMT17 have larger intra-class variability than on Market1501 as the number of different views of an identity is prone to be higher. Thus, different datasets require different values of \( \varepsilon \), and this has also been pointed out in prior art. In [20], the authors use a lower value for Market1501 (\( \varepsilon = 0.5 \)) and a larger value for MSMT17 (\( \varepsilon = 0.6 \)) to account for dataset complexity. In [11], the authors adjust the value of \( \varepsilon \) to obtain better results. However, in an unsupervised scenario, it is not possible to select an optimal value for \( \varepsilon \) as there is no prior knowledge of the target data. Therefore, it is paramount to develop a clustering algorithm that does not depend on hyper-parameter tuning.

We propose an ensemble-based clustering algorithm. As different values of \( \varepsilon \) yield different clusters, we run DBSCAN with different \( \varepsilon \) values and combine their results into a single final result. The proposed method effectively deals with noisy cases, allowing different but closer dense regions to be assigned to the same cluster, avoiding false positives and alleviating the burden of choosing the proper value for \( \varepsilon \).

Considering the feature space defined by the refined distance matrix \( \overline{D} \) from Step 1, we perform DBSCAN with five \( \varepsilon \) values: 0.5, 0.55, 0.6, 0.65, and 0.7. As the neighborhood increases, more samples are assigned to a cluster. This does not mean that all samples are true positives and we need a way to detect false positives. If a sample has been detected as an outlier with \( \varepsilon = 0.5 \), it is kept as an outlier on further runs. We then can better filter out false positive samples while grouping closer dense true positive regions in the same cluster.

Our ensemble-based clustering is illustrated and compared with the normal DBSCAN in Figure 3, where different colors represent different clusters (black represents outliers) and the circles highlight important regions to be analyzed. In the ground-truth, the pink and red circles present noisy data with two identities mixed, which would be hard to split with a single \( \varepsilon \) value. With \( \varepsilon = 0.5 \), DBSCAN yields the highest true positive rate, but it is not able to group sparser identities such as the one within the green circle, which is subdivided into two clusters (blue and yellow). The same happens within the blue circle: most samples are from the same identity but DBSCAN subdivides it into five clusters. In turn, our method is able to group results from lower \( \varepsilon \) values and create more robust clusters, as shown in Figure 3, \( \varepsilon = 0.7 \).

One optimization is the reduction of DBSCAN runs. One must keep in mind that if a point is detected as an outlier for \( \varepsilon = 0.5 \), it is kept as an outlier in subsequent runs with larger values. The same happens if a sample is an inlier for \( \varepsilon = 0.5 \); it will be an inlier in further runs as the neighborhood always increases. However, inlier samples assigned to different clusters can be put together in the same cluster due to the increasing neighborhood. For this reason, we say that the results for the \( j \)-th run contain the results for the \( i \)-th run, with \( i \leq j \), as samples that are inliers or outliers keep their status in subsequent runs. It becomes clear that it is enough to run DBSCAN with \( \varepsilon = 0.5 \) and \( \varepsilon = 0.7 \) only, as the last run implicitly combines all intermediate results.

C. Step 3: Learning rate update

To update the learning rate, we consider a warmup strategy [29], which is effective mainly in the first training iterations, where the number of samples is lower than on further iterations. In the first iterations, there is still little data available for training because the outliers detected by the clustering algorithm are in greater quantity and are discarded as noisy samples. In this context, the model is more prone to overfitting and a lower learning rate can aid the training in such cases.

The warmup strategy consists of starting the training process with a small learning rate value and gradually increasing it along the first iterations. Based on [29], we define the learning rate at iteration \( t \) as

\[
lr_t = \begin{cases} 
\frac{lr_{base}}{t^{\alpha}}, & 0 \leq t \leq 10, \\
lr_{base}, & 10 < t \leq K_1, 
\end{cases}
\]

where \( lr_{base} \) is a base value for the learning rate usually set to 0.00035, and \( K_1 = 30 \). The learning rate linearly increases in the first ten iterations and is constant for the remaining ones.

D. Step 4: Proxy selection

Once pseudo-labels are assigned to unlabeled data samples, based on the clustering results, and the learning rate is adjusted, the backbones can be updated.

This process starts in Step 4, with the selection of cluster proxies, which are prototypes that represent the clusters. For each cluster, its proxy is the feature vector of a randomly selected sample within that cluster.

Although clusters tend to become more reliable as more iterations are performed, there is still a chance they might contain false positive samples. Assuming that the majority of samples are true positives, we hypothesize that a random selection is more likely to return a robust proxy than, for instance, computing the mean vector from all samples (true and false positives). We verify the impact of selecting a random or mean proxy in Section V.

E. Step 5: Batch creation

The next step is batch creation. We consider the PK strategy [33], where we randomly select \( P \) out of \( C \) clusters generated in the current iteration, and \( K \) samples per cluster, creating batches of size \( P * K \).

It is important that a cluster appears only once per epoch, exposing the optimization to more diversity. For this, we create \( \left\lfloor \frac{C}{P} \right\rfloor \) batches, which can leave some clusters out of the current epoch because of the rounding. However, as we perform \( K_2 \) epochs per iteration, the clusters that were not used in the current epoch will likely be selected in the next epochs.
F. Step 6: Optimization

The created batches are forwarded to the backbones for the optimization. The loss function $L$ to be minimized is composed of two other loss functions: $L_{proxy}$ and $L_{hard}$.

For the $m$-th backbone, loss function $L_{proxy}$ is based on cluster proxies $p_{jm}$, $1 \leq j \leq C$ and is defined as

$$L_{proxy}(B; \theta_m) = -\frac{1}{|B|} \sum_{i=1}^{|B|} \log \left[ \frac{\exp(f_i^m f_{jm}^m / \tau)}{\sum_{j=1}^C \exp(f_i^m f_{jm}^m / \tau)} \right],$$

where $B$ is the batch, $|B|$ is the batch size, $\theta_m$ are the weights of the $m$-th backbone in the current iteration, $f_i^m$ is the feature vector of the $i$-th sample in $B$ extracted by the $m$-th backbone, $p_{jm}^m$ is the proxy of the same cluster as the $i$-th sample in $B$; and $\tau$ is a temperature hyper-parameter to regulate the sharpness of the distribution of distances from the $i$-th sample to all proxies. The rationale is to approximate each sample in the batch from its respective proxy and keep it apart from the other proxies.

As hard sample mining has shown promising results in prior art [11, 20], we considered it by utilizing a hard instance-based softmax-triplet loss defined as

$$L_{hard}(B; \theta_m) = -\frac{1}{|B|} \sum_{i=1}^{|B|} \log \left[ \frac{\exp(f_i^m f_{hm}^m / \tau)}{\exp(f_i^m f_{hm}^m / \tau) + \exp(f_i^m f_{jn}^m / \tau)} \right],$$

where $f_{hm}^m$ is the hardest positive sample in comparison to the $i$-th sample in $B$, i.e., it is the most distant feature from $f_i^m$ within the same cluster in the current batch. Analogously, $f_{jn}^m$ is the hardest negative sample in comparison to $f_i^m$ in the current batch, i.e., it is the sample closest to $f_i^m$ but from another cluster.

The $L_{triplet}$ loss provides a local view as it only considers samples in the current batch, while $L_{proxy}$ is a global loss since it considers all proxies from all clusters. The final loss combines them into a single function:

$$L(B; \theta_m) = L_{proxy}(B; \theta_m) + \lambda L_{hard}(B; \theta_m),$$

where $\lambda$ is a hyper-parameter to control the impact of $L_{hard}$.

Each backbone is trained independently, but with the same set of pseudo-labels and learning rate obtained in previous steps. That is, the green flow in Figure 1 is performed for each one of the $M$ backbones, once at a time.

G. Step 7: Mean Teacher average

In this step, we leverage the Mean Teacher strategy [34], which averages model weights over training steps to produce a final more accurate model. It computes a teacher model as the average of consecutive student models.

For each backbone (student model), we keep a teacher (or momentum) model with the same architecture. After each optimization step, the weights of a backbone are used to update the respective momentum model by Exponential Moving Average:

$$\Theta_m^{(t)} := \beta \Theta_m^{(t-1)} + (1 - \beta) \theta_m^{(t)},$$

where $\beta$ controls the inertia of the momentum weights over training, $t$ represents the iteration, and $\Theta_m$ are the weights of the momentum model that corresponds to the $m$-th backbone.

H. Inference

After the training pipeline, inference is done by ranking all gallery samples based on the distance to a query sample. We extract feature vectors for all gallery and query sets using the momentum models from each backbone, which we denote as $F_q^m$ and $F_g^m$, respectively, with $1 \leq m \leq M$. For each $m$, we calculate pairwise distances between samples of $F_q^m$ and $F_g^m$, resulting in a distance matrix $D_{qg}^m \in \mathbb{R}^{|Q| \times |G|}$, where $|Q|$ and $|G|$ are the number of samples in the query and gallery sets.

A final distance matrix $\overline{D}_{qg}$ is obtained by averaging all matrices element-wise:

$$\overline{D}_{qg} = \frac{1}{M} \sum_{m=1}^M D_{qg}^m.$$

Each row of $\overline{D}_{qg}$ holds the distances from a query to the gallery samples. We sort these distance to infer the closest class to the query sample.
IV. EXPERIMENTS

We perform experiments to validate our self-supervised learning pipeline, considering two applications: Person ReID and Authorship Verification from short text messages.

A. Datasets

For Person ReID, we use well-known large-scale datasets:

- Market1501 [7]: With a total of 12,936 images of 751 identities in the training set and 19,732 images in the test set. The test set is divided into 3,368 images for the query set and 15,913 images for the gallery set. We removed “junk” images from the gallery set as done by all previous works, so 451 images were discarded. This dataset has a total of six non-overlapping cameras and each identity is captured by at least two.

- DukeMTMC-ReID [6]: With 16,522 images of 702 identities in the training set and 19,889 images in the test set. The test set is divided into 2,228 query images and 17,661 gallery images of other 702 identities. It has eight cameras and each identity is captured by at least two.

- MSMT17 [8]: The most challenging ReID dataset. It comprises 32,621 images of 1,401 identities in the training set and 93,820 images of 3,060 identities in the test set. The test set is divided into 11,659 images for the query set and 82,161 images for the gallery set. It has a total of 15 cameras recording in three day periods (morning, afternoon, and night) on four different days. Out of the 15 cameras, 12 are outdoor cameras, and 3 are indoor. Each identity is captured by at least two cameras.

As done in previous ReID works, we remove images from the gallery that have the same identity and camera of the query to assess performance in a true cross-camera scenario. For evaluation, we calculate the Cumulative Matching Curve (CMC), from which we report Rank-1 (R1), Rank-5 (R5), and Rank-10 (R10), respectively, in the Market dataset; and by 2.0 p.p. in R1.

B. Implementation Details

We adopt \( M = 3 \) backbones. For person ReID, the backbones are well-known Deep Convolutional Neural Network (DCNN) architectures: ResNet50 [36], OSNet [37], and DenseNet121 [38], all of them previously trained over the ImageNet dataset [5]. For authorship verification, we consider BERT [24], BERTweet [25], and T5 [26] architectures.

For optimization, we consider the Adam [39] optimizer with weight decay 0.00035. The learning rate is set following the behavior in Equation 2 with \( lr_{base} = 0.00035 \). We implement the neighborhood-based distance with re-ranking [13], which relies on two parameters: \( k_1 \) which defines the k-reciprocal neighborhood size, and \( k_2 \), which defines the neighborhood size to average the distance representation. Following prior art, we set them to \( k_1 = 30 \) and \( k_2 = 6 \). For batch creation using the PK technique, we set \( P = 16 \) and \( K = 12 \), totaling 192 samples per batch.

The values for \( \varepsilon \) in the proposed ensemble-based clustering are 0.5, 0.55, 0.6, 0.65, and 0.7. We keep \( minPts = 4 \) in all DBSCAN runs as done in prior art. For the loss function, we set \( \tau = 0.04 \) and \( \lambda = 0.5 \) in Equations 3, 4, and 5.

The pipeline (blue flow in Figure 1) is executed for \( K_1 = 30 \) iterations and the green flow is executed for \( K_2 = 7 \) epochs, for each set of proposed clusters, and \( \beta = 0.999 \) in Equation 6.

The training pipeline is implemented using PyTorch [40]. The evaluation part and the OSNet backbone are implemented on Torchreid [41]. We perform all experiments on three RTX5000 GPUs, each with 16 GB of RAM. One of them is used to perform re-ranking while the other two are used to execute the whole training pipeline. The code will be made available upon publication of this manuscript.

C. Person ReID

We compare our pipeline applied to the Unsupervised Person Re-Identification problem with relevant methods in the literature. The results are shown in Table I.

We outperform all methods in the fully-unsupervised setup, which is the most challenging, in all datasets regarding mAP and R1. More specifically, we outperform the recent ICE method by 3.9 and 0.9 percentage points (p.p.) in mAP and R1, respectively, in the Market dataset; and by 12.8 and 9.2 p.p. on MSMT17, the most challenging unsupervised ReID dataset. We also surpass the state-of-the-art method in the Duke dataset, RLCC, by 5.5 and 0.7 p.p. in mAP and R1, respectively. This shows the effectiveness of our model on the fully-unsupervised scenario without using any annotation regarding identity nor any other side information.

Other works assume metadata, such as camera labels and tracklets, but no identity information. In Table I, we can see that the most helpful metadata is camera information. Person ReID is naturally a cross-camera retrieval task: a method must be able to retrieve (from the gallery) images of the same identity used as query, but seen from other cameras. In this sense, camera information provides a significant impact if it is leveraged during training. This is evinced when we compare our results with the camera-based ICE. It considers camera proxies per cluster and pulls images from different cameras closer to overcome differences brought by different points-of-view. This seems especially beneficial in more complex datasets (MSMT17), in which the camera-based ICE outperforms our method by 2.0 p.p. in R1.
TABLE I
COMPARISON WITH RELEVANT PERSON ReID METHODS CONSIDERING THREE SETUPS: CAMERA-BASED, TRACKLET-BASED, AND FULLY-UNSUPERVISED METHODS. OUR WORK FITS IN THE LAST CATEGORY, WHICH IS THE MOST CHALLENGING. WE HIGHLIGHT THE THREE BEST RESULTS IN THE FULLY-UNSUPERVISED SCENARIO: THE BEST ONE IN BLUE, THE SECOND BEST IN GREEN, AND THE THIRD IN ORANGE. FOR THE OTHER CATEGORIES, WE ONLY HIGHLIGHT THE BEST RESULT. BEST VIEWED IN COLOR.

| Dataset | Camera-Based | Tracklet-Based | Fully Unsupervised |
|---------|--------------|----------------|--------------------|
| Market  |              |                |                    |
| SSL [42]| CVPR’20      | 37.8           | R4                 |
| CCSE [43]| TIP’20      | 38.0           | R5                 |
| MPRD [44][*]| ICCV’21 | 51.1           | R6                 |
| DSCE-MC [45]| CVPR’21 | 61.7           | R7                 |
| JVTC [46]| ECCV’20       | 47.5           | R8                 |
| JCGL [47]| CVPR’21       | 66.8           | R9                 |
| ICS [18]| CVPR’21      | 72.9           | R10                |
| CAP [19]| AAAI’20     | 79.2           | R11                |
| CCTSE [12]| TIFS’21   | 67.7           | R12                |
| ICE [20]| ICCV’21 | 82.3           | R13                |
|            |              |                |                    |
| Duke      |              |                |                    |
| GPUFL [51]| ICIP’21       | 42.3           | R14                |
| MV-ReID [52]| SPL’21 | 45.6           | R15                |
| MMC [53]| CVPR’20 | 45.5           | R16                |
| HCT [54]| CVPR’20 | 56.4           | R17                |
| ABMT [11]| VACW’20 | 65.1           | R18                |
| SpCL [10]| NeuIPS’20 | 73.1           | R19                |
| SpCL-IBN [10]| NeuIPS’20 | 73.8    | R20                |
| RLCC [9]| CVPR’21 | 77.7           | R21                |
| ICE [20]| ICCV’21 | 79.5           | R22                |
|            |              | 83.4           | R23                |
|            |              |                |                    |
| MSMT17    |              |                |                    |
| SSL [42]| CVPR’20      | 37.8           | R4                 |
| CCSE [43]| TIP’20      | 38.0           | R5                 |
| MPRD [44][*]| ICCV’21 | 51.1           | R6                 |
| DSCE-MC [45]| CVPR’21 | 61.7           | R7                 |
| JVTC [46]| ECCV’20 | 47.5           | R8                 |
| JCGL [47]| CVPR’21 | 66.8           | R9                 |
| ICS [18]| CVPR’21 | 72.9           | R10                |
| CAP [19]| AAAI’20 | 79.2           | R11                |
| CCTSE [12]| TIFS’21 | 67.7           | R12                |
| ICE [20]| ICCV’21 | 82.3           | R13                |

Table: Execution time. We report the average time taken by some steps as well as the total time to execute the pipeline, for the three ReID datasets. The time format is HH:MM:SS.

| Dataset | Step 1    | Step 2    | Steps 3 to 7 | Total Time | Inference Time (ms) |
|---------|-----------|-----------|--------------|------------|---------------------|
| Market  | 00:13:27  | 00:13:19  | 00:13:19     | 00:49:45   | 41.4                |
| Duke    | 00:13:53  | 00:04:00  | 00:14:13     | 00:31:18   | 59.1                |
| MSMT17  | 00:04:25  | 00:00:38  | 00:04:25     | 00:08:45   | 80.2                |

One should also note that, in general, our assumptions are even more relaxed than other methods. Our clustering algorithm, for instance, does not require hyper-parameter tuning, while ICE explicitly fixes a value for ε depending on the target dataset.

Finally, our method is able to place most true positive samples from gallery closer to the query in all scenarios, which is represented by the greatest mAP for the three datasets. Our method is thus able to achieve the best performance among all methods even over those assuming strong camera information as metadata. One should also note that, in general, our assumptions are even more relaxed than other methods. Our clustering algorithm, for instance, does not require hyper-parameter tuning, while ICE explicitly fixes a value for ε depending on the target dataset.

In the Supplementary Material, we also compare our pipeline to UDA methods that require a source domain to provide an initial task-related knowledge. We outperform all these methods in mAP, and obtain the best or second best ranking values considering other metrics.

1) Training time: We also analyze our pipeline in terms of execution time for training (Table II). Clustering (Step 2) time is negligible due to its optimized implementation. The fine-tuning process (Steps 3 to 7) takes longer as it requires optimization using the generated clusters for K2 = 7 epochs. The total time for the pipeline is in the order of a few hours, and it depends on the size of the dataset. For MSMT17, the largest one with 32,621 images in the training set and 1,041 identities (greater than the number of classes on ImageNet), the method presents a reasonable time around 19 hours.

Inference time also increases with the size of the gallery sets, as there are more samples to compare to the query. Inference for a query on Market, the smallest one, takes 41.4 milliseconds, while for MSMT17, the largest dataset, it takes 80.2 milliseconds. All scenarios present reasonable inference time, under one second.

2) Qualitative Analysis: In terms of qualitative results, we provide some examples of success and some for failure, for DukeMTMCReID and Market datasets, illustrated in Figure 4. We do not show examples for MSMT17 as reproducing this dataset’s images is not allowed in any format.

From the successful examples (Figures 4a and 4c), we observe that our method can identify several fine-grained details throughout the image in order to retrieve the correct identity. This means that it overcomes differences in point-of-view, pose, illumination, and background. In one of the failure examples (Figure 4b), there are two people in the same bounding box, which is ambiguous. In the second failure
example (Figure 4d), the query presents an identity in low resolution and with clothing colors similar to the background. Despite being failure cases, the model can still find similar and fine-grained details, which could help in cases where the goal is to find people dressing similarly. We present more success and failure cases in the Supplementary Material.

D. Authorship verification

We now consider a second task — authorship verification — with minor adjustments mainly related to the nature of the problem. For the backbones, we consider BERT [24], BERTweet [25], and T5 [26] as they were developed to deal with text. We apply augmentation on tweets with more than 5 tokens, by masking from 10% to 20% of the tokens with a “mask” token on BERT and BERTweet, and with an “unknown” token on T5. Even the base version of BERT is too complex for short signals (text tweets), making the training more prone to overfitting. Hence, we freeze the first ten attention blocks of BERT and BERTweet, leaving only the 11th block to be updated. For the same reason, we set \( K_1 = 15 \) as the number of iterations — half of the value used for the ReID experiments —, to alleviate the impact of over-training. For batch creation, we set \( P = 8 \) and \( K = 8 \). All other parameters are the same as used for ReID. The only difference in the pipeline is that we do not apply the optimization shown in Figure 2, as it is only for image representations.

We run comparative experiments considering the AdHominem method [28]. It employs an attention-based model for authorship verification on social media text. AdHominem performs supervised training using Siamese networks to answer if two tweets were written by the same author or not. To perform the ranking task, we take the Euclidean distance between two tweets returned by the model and rank the gallery tweets given a query. The results are shown in Table III.

Our method outperforms AdHominem in both subsets. More specifically, we outperform AdHominem by 7.0 and 24.5 p.p. in mAP and R1, respectively, in the first subset. In the second one, we outperform it by 2.6 and 11.7 p.p.

One must note that AdHominem is trained in a supervised manner considering the identity of each tweet, i.e., the method knows a priori “who” wrote the tweets to supervise the training. However, our method relaxes this constraint by taking only the raw tweet text without any labeling. Moreover, our training and test data are totally disjoint on the identities and, since AdHominem is trained for a closed set of authors, it generalizes poorly to unseen authors.

Despite the fact that our method utilizes pre-initialized weights, these weights were trained for other tasks, such as question answering, next sequence prediction, predict missing words and so on, instead of authorship verification.

In Figure 5, we provide the evolution of Rank-1 during training to show the merits of employing our pipeline. In the first subset (50 authors), we verify a Rank-1 oscillation after the 10th iteration and a slight decrease until the last iteration. BERTweet provides the greatest gain, followed by BERT and T5. In the second subset (500 authors), the training process is more stable though it is numerically inferior as there are more authors. The feature space is denser, securing more stability for the convergence of the models.

V. ABLATION STUDY

We validate each part of our pipeline by checking how these influence the performance, considering Market and Duke...
datasets. If not specified, we assume ensemble-based clustering, and \( \tau = 0.04 \) and \( \lambda = 0.5 \) in Equations 3, 4, and 5.

A. Step 1: impact of distance averaging

In the first step of the pipeline, we average the distance matrices obtained with each backbone separately, computing a combined distance matrix \( \bar{D} \) (Equation 1). This allows complementary knowledge to be grouped together for training. To measure how this impacts the final performance, we train each backbone separately, as expected, but feed each distance matrix directly to Step 2, instead of averaging them together. We present the results in Figure 6, showing the impact of averaging the distances in Step 1 for each backbone separately and for the combined result (considering Equation 7).

Distance averaging in Step 1 is important for training and it positively impacts each backbone individually, as well as their final combination, which allows for a better grouping in the clustering step. Considering the combination of backbones (Equation 7), the gains are also considerable for both datasets. In Market, we achieve an improvement of 2.4 p.p. and 1.1 p.p., in mAP and R1, respectively, and 2.1 p.p. and 1.6 p.p. on Duke. These results show the effectiveness of our proposed approach without requiring mutual training [14], [15] or co-teaching [55], which in turn promotes a simpler training.

B. Step 2: impact of the ensemble-based clustering

We verify the effectiveness of our proposed ensemble-based clustering method. We replace it by the standard DBSCAN algorithm in the second step of our pipeline, and keep the remaining parts unchanged. As we combine the results of DBSCAN runs with different \( \varepsilon \) values into a single result, in Table IV, we present the separate results for each \( \varepsilon \) value.

This experiment shows that the proposed ensemble-based clustering effectively combines DBSCAN intermediate results into a final one, suggesting more robust clusters, and outperforming all results in both datasets. Note that, as motivated in Section III-B, if a single \( \varepsilon \) is employed for clustering, each dataset has different optimal values. In Table IV, we see that the model achieves the best performance in mAP and R1 with \( \varepsilon = 0.55 \) for Market and \( \varepsilon = 0.7 \) for Duke, which shows that the optimal values can change significantly from a dataset to another. In contrast, our proposed ensemble-based clustering strategy obtains the best performance by grouping DBSCAN results using lower \( \varepsilon \) values (denser clusters, lower false positive rate) and higher \( \varepsilon \) values (more diverse clusters, lower false negative rate), alleviating the burden of choosing a proper unique value for this hyper-parameter.

C. Step 4: impact of proxy selection

In Step 4, we select a random sample per cluster as a proxy to aid the optimization as in [56]. We validate this choice by replacing the random selection by the mean feature vector of each cluster. The results are shown in Table V.

| TABLE V |
| IMPACT OF THE PROXY SELECTION. WE REPLACE THE RANDOM SELECTION OF SAMPLES TO SERVE AS CLUSTER PROXIES BY THE MEAN FEATURE VECTOR OF THE CLUSTER. THE BEST RESULTS ARE IN BLUE. |

| Proxy Selection | Market | Duke |
|-----------------|--------|------|
| Mean            | 82.0   | R1   |
| (random)        | 83.4   | R5   |
| R5              | 96.2   | R10  |
| R10             | 97.5   | R5   |
| Ours (random)   | 97.8   | R10  |
| Ours            | 91.0   | 93.0 |

Random selection improves the performance for all metrics, but mainly for mAP and R1. More specifically, we obtain a gain of 1.4 and 1.1 p.p. in mAP and R1, respectively, in the Market dataset, and 2.0 and 2.1 p.p. in the Duke dataset. This validates our assumption that using a mean vector as proxy hinders cluster representation and further training, as it is affected by the false positive samples of the cluster.

D. Step 6: impact of loss function hyper-parameters

We vary hyper-parameters \( \tau \) and \( \lambda \) in the loss functions (Equations 3, 4, and 5) to check how they impact the pipeline. The results with varying \( \tau \) are shown in Figures 7a and 7b. For both datasets, the best value is 0.04 as it provides the best mAP and R1, and top results for R5 and R10. A lower value (\( \tau = 0.03 \)) provides slightly lower performance, and greater values (\( \tau = 0.07 \)) increases the loss and the gradient magnitude, which leads to suboptimal optimization.

Figures 7c and 7d show the results with varying \( \lambda \) (Equation 5). This hyper-parameter regulates the influence of the hard instance-based softmax-triplet loss (\( L_{hard} \)) on the final loss. This loss brings a local view by considering triplets at the batch level. When \( \lambda = 0.0 \) (i.e., \( L_{hard} \) is not considered for optimization), we obtain one of the worst performances, showing that a more local view of the data is also important for model training. As we increase its value, the performance increases. But for higher values (from 0.75 to 1.0), we verify a performance drop as the local view dominates the global view. We keep \( \lambda = 0.5 \) in all experiments, as it yields the best mAP and R1 results for both datasets.

VI. Conclusion

In this work, we proposed a novel self-supervised learning pipeline for scenarios with high intra-class semantic disparity and inter-class similarity. General methods do not account for this problem as they are usually devised for less complex datasets, such as Imagenet.

Our pipeline starts from a common concept — clustering steps to propose pseudo-labels for unlabeled samples and optimization steps to update backbones supervised by the
pseudo-labels —, but we incorporate novel techniques to effectively address more critical tasks.

We propose the use of a neighborhood-based distance refinement followed by distance averaging to amalgamate complementary knowledge learned by different backbones. We showed that this is highly effective when compared to using distances obtained from each backbone directly. We provide a better distance measurement between samples, even without task-related initialization, due to the joint contribution of neighborhood-based distances and distance matrices ensemble.

Our second contribution is an ensemble-based clustering algorithm to provide pseudo-labels for optimization. The advantages are twofold: our solution creates dense but diverse clusters, and it does not need hyper-parameter tuning.

To show the generalizing ability of our pipeline, we applied it to two tasks that are highly different in terms of input data: person re-identification and authorship verification from short text messages. To the best of our knowledge, this is the first self-supervised learning method that can be applied to different modalities with only minor adjustments.

For Person ReID, our method yields state-of-the-art performance in terms of mAP and Rank-1 in all benchmarks. For Authorship Verification, we obtained competitive results when compared to a prominent method that was trained in a supervised manner. Therefore the ensemble-based clustering has strong potential to find satisfactory clusters for model training on the fully-unsupervised scenario.

We conclude that it is possible to learn from complex fully-unlabeled data in different modalities, but the model requires a robust distance measurement (brought by the ensemble of distances matrices) along with a clustering strategy that tackles the unknown feature distribution from different datasets. When both strategies are put together, the method finds robust clusters for optimization.

One important aspect of the method that still needs optimization is memory usage. Currently, it requires quadratic memory $O(N^2)$, where $N$ is the total number of samples available for training, due to the pairwise distance matrices. Nonetheless, all prior art also face the same issue.

Finally, our proposed method is applicable to myriad of tasks, such as vehicle and place re-identification [57], [58]. In a future study, we aim to explore these new applications in an attempt to understand how the pipeline would adapt to new requirements. We would also like to consider different modalities simultaneously to propose possible connections among the main elements in a scene in a self-supervised way, contributing to scene or event understanding.

ACKNOWLEDGMENTS

We thank the financial support of the São Paulo Research Foundation (FAPESP), grants DejáVu #2017/12646-3, #2019/15825-1, and #2018/10204-6, and the support of Google through LARA (Latin America Research Awards).

REFERENCES

[1] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, “Unsupervised learning of visual features by contrasting cluster assignments,” arXiv preprint, vol. arXiv:2006.09882, 2020.
[2] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning,” in Conf. Comput. Vis. Pattern Recog., 2020, pp. 9729–9738.
[3] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” in Int. Conf. Mach. Learn., 2020, pp. 1597–1607.
[4] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, “Emerging properties in self-supervised vision transformers,” arXiv preprint, vol. arXiv:2104.14294, 2021.
[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in Conf. Comput. Vis. Pattern Recog., 2009, pp. 248–255.
[6] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, “Performance measures and a data set for multi-target, multi-camera tracking,” in Eur. Conf. Comput. Vis., 2016, pp. 17–35.
[7] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, “Scalable person re-identification: A benchmark,” in Int. Conf. Comput. Vis., 2015, pp. 1116–1124.
[8] L. Wei, S. Zhang, W. Gao, and Q. Tian, “Person transfer GAN to bridge domain gap for person re-identification,” in Conf. Comput. Vis. Pattern Recog., 2018, pp. 79–88.
Supplementary Material

Reasoning for Complex Data through Ensemble-based Self-Supervised Learning

Gabriel Bertocco, Antonio Theophilo, Fernanda Andaló, Member, IEEE, and Anderson Rocha, Senior Member, IEEE

I. SUPPLEMENTARY MATERIAL

A. Comparison with Unsupervised Domain Adaptation

Here we extend our comparison to prior art considering the works that employ a supervised pre-training on a source domain and then leverage their pipelines on the target domain. Our pipeline, as explained in the main article, does not require pre-training on task-related datasets and is applied directly over the target domain from weights initialized over ImageNet. Therefore, our model operates under a more challenging scenario with less constraints. The results are shown in Tables I and II.

Despite not training on source domain, we can see in Table I that our pipeline yields the best mAP on both scenarios and the first or at least the second best value for all other metrics. On the challenging MSMT17 (Table II), we present even better performance by obtaining the best values for all metrics, surpassing the second best method by a margin in all metrics. This shows the strong ability of our model to learn in a fully unsupervised scenario without requiring any meta-information, source domain or manual definition of hyper-parameter clustering.

B. Further Qualitative Analysis

In the main article, we show the qualitative analysis considering one success and one failure cases for Market and Duke datasets. We extend this analysis here and show the activated regions in the gallery images given a query from each camera in each dataset, considering the ResNet50 backbone. Blue frames indicate the query image, green indicates the true positive samples, and red the false positive samples. We adapted the method from [?] to visualize the results.

Figure 1 shows one success case for each one of the eight cameras present on the Duke dataset. We see that the model is able to mine fine-grained details on all images, regardless of the camera, and retrieve true positive samples in the top-10 images. The only exception is for camera 8 (Figure 1h), where the last six images are false positive samples. However, for all samples, the model still focuses on fine-grained details, and the errors occur due to similar clothing of the identities. The same happens on the examples in Figure 2. Despite being error cases, the model can focus on fine-grained details. The errors occur when there are two or more identities in the query image (Figure 2f), the background color is similar to an object held by the identity (in Figure 2c the backpack is the same color of the background), or due to very similar clothing of different identities. Since Duke was recorded at a university campus during winter, it is natural to find individuals wearing similar (and dark) clothing.

Figure 3 shows one success case for each camera on the Market dataset. Following the same previous conclusions, the model is able to identify fine-grained features in the image to correctly retrieve the true positive samples among the top-10 images. The failure cases (Figure 4) are mainly due to the identities being similar (similar T-shirts, hair, and skin tone), which is the case in Figures 4b, 4d, and 4f.

To summarize, the model is able to identify fine-grained details to match the same identity, allowing us to outperform prior art.
### TABLE I
RESULTS ON UNSUPERVISED DOMAIN ADAPTATION. OUR MODELS DO NOT REQUIRE PRE-TRAINING ON SOURCE DOMAIN. HERE WE PRESENT MARKET1501 TO DUKE-MSMT-ReID AND DUKE-MSMT-ReID TO MARKET1501 ADAPTATION SCENARIOS. WE HIGHLIGHT THE THREE BEST MODELS WITH BLUE, GREEN, AND ORANGE, RESPECTIVELY.

| Method          | reference | Duke → Market | Market → Duke |
|-----------------|-----------|---------------|---------------|
|                 |           | mAP | R1 | R5 | R10 | mAP | R1 | R5 | R10 |
| PTGAN [?]       | CVPR'18   | -  | 38.6 | -  | 66.1 | -  | 27.4 | -  | 50.7 |
| PUL [?]         | TOMM'18   | 20.5 | 45.5 | 60.7 | 66.7 | 16.4 | 30.0 | 43.4 | 48.5 |
| MMFA [?]        | ArXiv'18  | 27.4 | 56.7 | 75.0 | 81.8 | 24.7 | 45.3 | 59.8 | 66.3 |
| SPGAN [?]       | CVPR’18   | 22.8 | 51.5 | 70.1 | 76.8 | 22.3 | 41.1 | 56.6 | 63.0 |
| TJ-AIDL [?]     | CVPR’18   | 26.5 | 58.2 | 74.8 | 81.1 | 23.0 | 44.3 | 59.6 | 65.0 |
| SPGAN+LMP [?]   | CVPR’18   | 26.7 | 57.7 | 75.8 | 82.4 | 26.2 | 46.4 | 62.3 | 68.0 |
| HHL [?]         | ECCV’18   | 31.4 | 62.2 | 78.8 | 84.0 | 27.2 | 46.9 | 61.0 | 66.7 |
| ATNet [?]       | CVPR’19   | 25.6 | 55.7 | 73.2 | 79.4 | 24.9 | 45.1 | 59.5 | 64.2 |
| CamStyle [?]    | TIP’19    | 27.4 | 58.8 | 78.2 | 84.3 | 25.1 | 48.4 | 62.5 | 68.9 |
| MAR [?]         | CVPR’19   | 40.0 | 67.7 | 81.9 | -    | 48.0 | 67.1 | 79.8 | -    |
| PAUL [?]        | CVPR’19   | 40.1 | 68.5 | 82.4 | 87.4 | 53.2 | 72.0 | 82.7 | 86.0 |
| ECN [?]         | CVPR’19   | 43.0 | 75.1 | 87.6 | 91.6 | 40.4 | 63.3 | 75.8 | 80.4 |
| ISSDA-ReID [?]  | CVPR’19   | 63.1 | 81.3 | 92.4 | 95.2 | 54.1 | 72.8 | 82.9 | 85.9 |
| PDA-Net [?]     | ICC’19    | 47.6 | 75.2 | 86.3 | 90.2 | 45.1 | 63.2 | 77.0 | 82.5 |
| CR-GAN [?]      | ICC’19    | 54.0 | 77.7 | 89.7 | 92.7 | 48.6 | 68.9 | 80.2 | 84.7 |
| PCB-PAST [?]    | ICC’19    | 54.6 | 78.4 | -    | -    | 54.3 | 72.4 | -    | -    |
| UCDA [?]        | ICC’19    | 30.9 | 60.4 | -    | -    | 31.0 | 47.7 | -    | -    |
| SSG [?]         | ICC’19    | 58.3 | 80.0 | 90.0 | 92.4 | 53.4 | 73.0 | 80.6 | 83.2 |
| CASCL [?]       | ICC’19    | 35.5 | 65.4 | 80.6 | 86.2 | 37.8 | 59.3 | 73.2 | 77.8 |
| UDAP [?]        | PR’20     | 53.7 | 75.8 | 89.5 | 93.2 | 49.0 | 68.4 | 80.1 | 83.5 |
| MMCL [?]        | CVPR’19   | 60.4 | 84.4 | 92.8 | 95.0 | 51.4 | 72.4 | 82.9 | 85.0 |
| ACT [?]         | AAAT’20   | 60.6 | 80.5 | -    | -    | 54.5 | 72.4 | -    | -    |
| ECN-GPP [?]     | TPAMI’20  | 63.8 | 84.1 | 92.8 | 95.4 | 54.4 | 74.0 | 83.7 | 87.4 |
| SNR [?]         | CVPR’20   | 61.7 | 82.8 | -    | -    | 58.1 | 76.3 | -    | -    |
| AD-Cluster [?]  | CVPR’20   | 68.3 | 86.7 | 94.4 | 96.5 | 54.1 | 72.6 | 82.5 | 85.5 |
| MMT [?]         | ICLR’20   | 71.2 | 87.7 | 94.9 | 96.9 | 65.1 | 78.0 | 88.8 | 92.5 |
| DG-Net++ [?]    | ECCV’20   | 61.7 | 82.1 | 90.2 | 92.7 | 63.8 | 78.9 | 87.8 | 90.4 |
| MEB-Net [?]     | ECCV’20   | 76.0 | 89.9 | 96.0 | 97.5 | 66.1 | 79.6 | 88.3 | 92.2 |
| ABMT [?]        | WACV’20   | 80.4 | 93.0 | -    | -    | 70.8 | 83.3 | -    | -    |
| CCTSE [?]       | TIFS’21   | 78.4 | 92.9 | 96.9 | 97.8 | 72.6 | 85.0 | 92.1 | 93.9 |
| Ours            | This work | 83.4 | 92.9 | 97.1 | 97.8 | 72.7 | 83.9 | 91.0 | 93.0 |

### TABLE II
RESULTS ON UNSUPERVISED DOMAIN ADAPTATION. WE PRESENT MARKET1501 TO MSMT17 AND DUKE-MSMT-CRe-ID TO MSMT17 ADAPTATION SCENARIOS. OUR MODELS DO NOT REQUIRE PRE-TRAINING ON SOURCE DOMAIN SO WE REPLICATE OUR RESULTS FOR BOTH CONSIDERED SCENARIOS. WE HIGHLIGHT THE THREE BEST MODELS WITH BLUE, GREEN, AND ORANGE, RESPECTIVELY.

| Method          | reference | Duke → MSMT17 | Market → MSMT17 |
|-----------------|-----------|---------------|-----------------|
|                 |           | mAP | R1 | R5 | R10 | mAP | R1 | R5 | R10 |
| PTGAN [?]       | CVPR 2018 | 3.3  | 11.8 | -  | 27.4 | 3.9  | 10.2 | -  | 24.4 |
| ECN [?]         | CVPR 2019 | 10.2 | 30.2 | 41.5 | 46.8 | 8.5  | 25.3 | 36.3 | 42.1 |
| SSG [?]         | ICCV 2019 | 13.3 | 32.2 | -  | 51.2 | 13.2 | 31.6 | -  | 49.6 |
| ECN-GPP [?]     | TPAMI 2020 | 16.0 | 42.5 | 55.9 | 61.5 | 15.2 | 40.4 | 53.1 | 58.7 |
| MMCL [?]        | CVPR 2020 | 16.2 | 43.6 | 54.3 | 58.9 | 15.1 | 40.8 | 51.8 | 56.7 |
| MMT [?]         | ICLR 2020 | 23.3 | 50.1 | 63.9 | 69.8 | 22.9 | 49.2 | 63.1 | 68.8 |
| ABMT [?]        | WACV 2020 | 33.0 | 61.8 | -  | -   | 27.8 | 55.5 | -  | -   |
| SpCL [?]        | NeurIPS 2020 | -  | -  | -  | -   | 31.0 | 58.1 | 69.6 | 74.1 |
| CCTSE [?]       | TIFS’21 | 34.5 | 63.9 | 75.3 | 79.6 | 33.2 | 62.3 | 74.1 | 78.5 |
| Ours            | This work | 42.6 | 68.2 | 77.9 | 81.4 | 42.6 | 68.2 | 77.9 | 81.4 |
Fig. 1. Success cases considering one query from each camera on Duke. These results are obtained with the ResNet50 backbone after training on Duke.

Fig. 2. Failure cases considering one query from each camera on Duke. This results are obtained with the ResNet50 backbone after training on Duke.
Fig. 3. Success cases considering one query from each camera on Market. These results are obtained with the ResNet50 backbone after training on Market.

Fig. 4. Failure cases considering one query from each camera on Market. These results are obtained with the ResNet50 backbone after training on Market.