# Reidentification of Objects From Aerial Photos With Hybrid Siamese Neural Networks

Alessio Devoto, Indro Spinelli, *Student Member, IEEE*, Francesca Murabito, Fabrizio Chiovoloni, Riccardo Musmeci, and Simone Scardapane

***Abstract*—In this article, we consider the task of reidentifying the same object in different photos taken from separate positions and angles during aerial reconnaissance, which is a crucial task for the maintenance and surveillance of critical large-scale infrastructure. To effectively hybridize deep neural networks with available domain expertise for a given scenario, we propose a customized pipeline, wherein a domain-dependent object detector is trained to extract the assets (i.e., subcomponents) present on the objects, and a siamese neural network learns to reidentify the objects, exploiting both visual features (i.e., the image crops corresponding to the assets) and the graphs describing the relations among their constituting assets. We describe a real-world application concerning the reidentification of electric poles in the Italian energy grid, showing our pipeline to significantly outperform siamese networks trained from visual information alone. We also provide a series of ablation studies of our framework to underline the effect of including topological asset information in the pipeline, learnable positional embeddings in the graphs, and the effect of different types of graph neural networks on the final accuracy.**

***Index Terms*—Energy grids, graph neural networks (GNNs), object detection, object reidentification, siamese networks.**

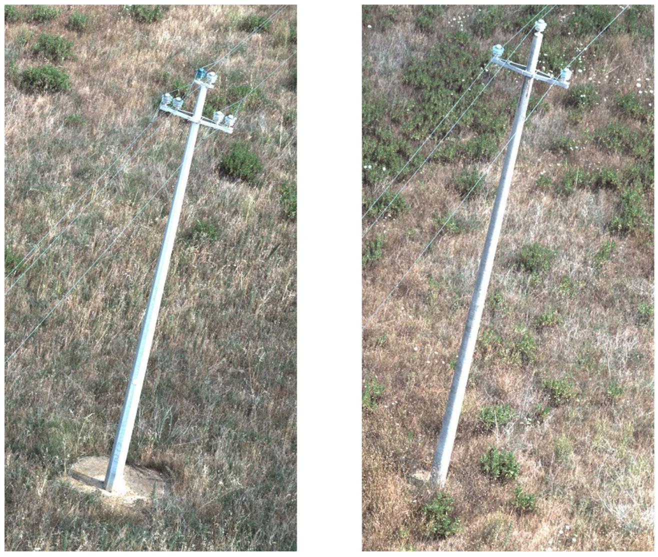


Fig. 1. Example of a difficult pair taken from the dataset: the two poles have a very similar structure and are set against similar background. However, they can be distinguished by carefully looking at their assets (in particular, their isolators).

## I. Introduction

The maintenance of transmission and distribution networks is a fundamental problem faced by dozens of companies around the world, and it is a critical task for ensuring their constant reliability and performance. However, maintenance is far from trivial since large-scale power grids are composed of millions of interacting components, which are spread over large distances and continuously exposed to wind, rain, extreme weather events (e.g., earthquakes), and standard wear of their components. As a representative example, the Italian energy grid is composed of over 74 000 km of lines spanning the entire country, alongside thousands of assets including high-voltage lines and transformer stations.

In this article, we focus on the task of mapping and surveying electric poles across the grid which, because of their number (typically thousands) and characteristics, constitute an immense task requiring careful automation. In order to identify possible issues, cyclical visual inspections are performed, which are done by repeatedly mapping out all the poles of the network with planned aerial flights [1], [2], remote sensing images [3], or even on-foot patrols. However, individual poles in aerial images are identifiable only up to a certain precision, due to pictures being taken from different angles and in different operating conditions (e.g., weathers, occlusions). An example of this is shown in Fig. 1. Because of this, algorithms to automatically reidentify the same object from different pictures are required [4].

In the literature, object reidentification tasks are commonly solved by the use of deep learning combined with metric learning algorithms (e.g., siamese networks), wherein different images are processed via the same convolutional architecture to obtain a fixed-dimensional embedding, and the networks are trained

Manuscript received 21 January 2022; revised 23 May 2022; accepted 13 June 2022. Date of publication 21 June 2022; date of current version 3 March 2023. Paper no. TII-22-0330. *(Corresponding author: Simone Scardapane.)*

Alessio Devoto, Indro Spinelli, and Simone Scardapane are with the Department of Information Engineering, Electronics and Telecommunications (DIET), Sapienza University of Rome, 00185 Roma, Italy (e-mail: devoto.1701081@studenti.uniroma1.it; indro.spinelli@uniroma1.it; simone.scardapane@uniroma1.it).

Francesca Murabito, Fabrizio Chiovoloni, and Riccardo Musmeci are with Enel S.p.A, 00198 Rome, Italy (e-mail: francesca.murabito@enel.com; fabrizio.chiovoloni@enel.com; riccardo.musmeci@enel.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2022.3184407.

Digital Object Identifier 10.1109/TII.2022.3184407

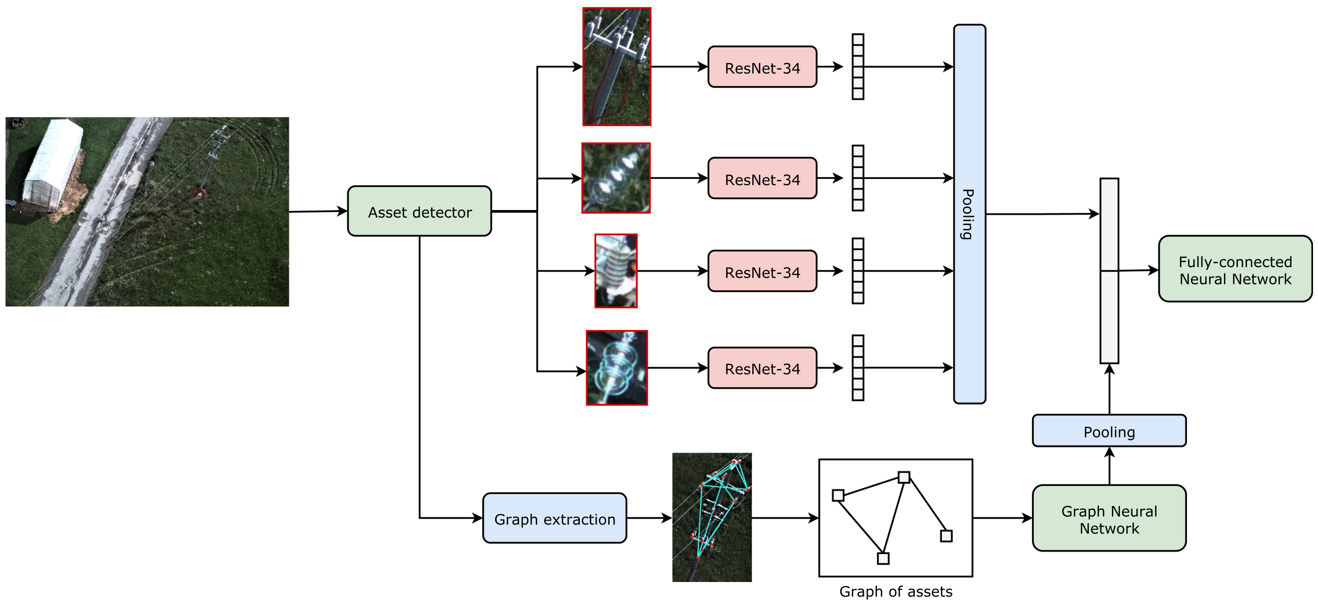


Fig. 2. Schematic diagram of the feature extraction pipeline. A customized object detection model simultaneously detects the bounding box of the pole and the bounding boxes of its composing assets (e.g., transformers). Then, a graph is built where each node corresponds to a detected asset and edges encode their relation (as described in the main text).The image is represented by combining the embedding from a visual branch (top) and from a graph branch (below). Only a subset of the assets is shown for clarity.

so that embeddings belonging to the same object (e.g., poles) are closer than embeddings belonging to different objects [5]. However, as we show in the experimental section, pure reidentification based on raw visual images is insufficient in our scenario, where the object of interest only occupies a small part of the full image with large overlaps with its background (see again Fig. 1).

This article is motivated by the following observation: electric poles, like many other components of the energy grids, are in themselves complex objects, composed of a variety of assets including several types of isolators, cabinets, transformers, and so on (we describe more in-depth the assets we consider in our work in Section IV-A). These assets provide valuable information that is crucial for identifying different poles from similar photos, and which is not considered explicitly in standard approaches working on the entire image as a whole. Handling them inside a neural pipeline is the major aim of this article.

*Contribution of This Article:* Inspired by recent works on graph deep learning [6], [7] and object-centric models [8], [9] (as described more in-depth in Section II), in this article, we propose and empirically validate a novel framework for object reidentification in aerial images aided by an object detection model that is based on what we call a *graph of visual assets*. As shown in Fig. 2, we first train a customized object detection model to extract all the assets belonging to a single pole. From these, we build a graph where each node corresponds to an asset, and edges describe the connection between different assets (in a way which is robust to changes in orientation and view). Using both types of information, we train a siamese neural network on the combination of the following two different embeddings: one in the original image domain (to identify poles based on the appearance of the assets), and one in the graph domain (to identify poles based instead on the relations between assets),

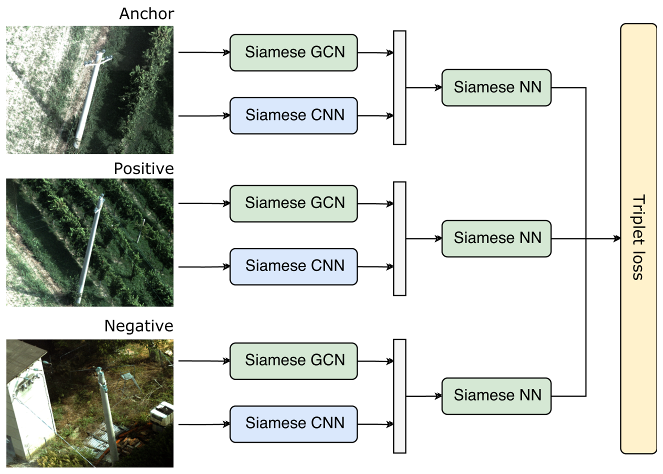


Fig. 3. During training, we randomly sample a triplet of objects, including a positive pair (two photos of the same pole), and a negative pair (two photos of different poles). The images are processed according to the pipeline in Fig. 2, on top of which we train with a proper metric learning loss. The GCN, CNN, and neural network (NN) blocks correspond, respectively, to the lower branch, upper branch, and rightmost block of Fig. 2.

as shown, respectively, in the upper and lower parts of Figs. 2 and 3. To build the embedding in the graph domain, we use graph neural networks (GNNs), a powerful family of neural networks that are equivariant to permutation of the assets in the image [6], and are flexible enough to handle a combination of object detection features, trainable positional features [10], and relational features.

We empirically validate our framework on a realistic use case collected from the Italian energy grid (described in

Section III), showing that it provides significant increases in the performance if compared to a standard object reidentification approach working on the visual features alone, even when the latter is combined with the trained object detector. Although GNNs have already been shown to provide strong performance in problems ranging from bioinformatics to social networks, the framework proposed in this article, to the best of authors' knowledge, shows for the first time that a more sophisticated *composite* approach exploiting both the visual features and the relational information coming from the assets can outperform both convolutional neural networks (CNNs) and GNNs trained individually. Since graphs are pervasive in energy grids, we hypothesize these hybrid approaches can become a significant component of the next generation of artificial intelligence algorithms applied to these fields, allowing to inject domain expertise in a more explicit fashion inside the predictive models provided by the deep networks.

The rest of this article is organized as follows. In Section II, we briefly overview recent works related to the proposed framework. The use case we consider is described in Section III, while the framework itself is provided in Section IV. We perform a thorough evaluation in Section V. Finally, Section VI concludes this article.

## II. Related Works

*1) Object Reidentification:* Object reidentification is the task of recognizing the same object appearing in different photos, which is a common task for, e.g., security [11] and object tracking [12]. In deep learning, a common solution is to consider a *siamese* model, where two photos are processed with the same neural network, and the resulting vectorial embeddings are trained to provide small distances for the same objects, and large distances for different objects [4]. This is achieved by using loss functions designed for metric learning [13], such as the triplet loss or the InfoNCE loss, as opposed to classical NN pipelines for classification exploiting cross-entropy losses [14]. Recently, contrastive learning [8] has popularized the use of these losses also for unsupervised and self-supervised learning of models, by using augmentations of the same image (or sample) to create different views of the same object.

*2) Electric Pole Maintenance:* Maintenance and control of energy grids (and their assets) has become a widespread problem over the last decades due to the aging of components, the heterogeneity of the grids (e.g., multiple types of renewable sources), and the interlocking of the grids inside smart cities [15]. In energy grids, deep learning has found widespread use, ranging from optimal flow analysis [16] to anomaly detection [17] and energy forecasting [18]. Concerning electric poles in particular, a lot of attention has gone into ways of mapping them periodically, including manned [1] and unmanned [2] aerial flights, and remote sensing pipelines [3] (we refer to [2] for a more in-depth overview). Concerning deep learning instead, deep neural networks have been used to predict possible failures [19], identifying specific poles from images [20], or finding vegetation or icing on the poles [2]. State-of-the-art methods are generally framed as an object detection problem, where the task is to find the proper bounding box surrounding a pole from an aerial image [20]. In this article, we consider an intermediate problem, where we assume that poles have been successfully identified in multiple photos (taken from successive aerial routes, see Section III), but we need to *reidentify* the same pole from different images to plan potential maintenance activities.

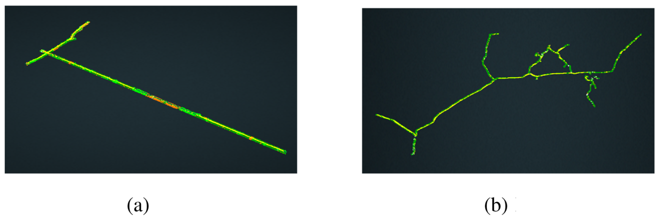


Fig. 4. Sample of two lines considered when building the dataset. (a) Simple morphology. (b) Nested morphology.

*3) Graph Neural Networks:* GNNs are neural networks that can process graph-based data, such as road transportation networks, without embedding them first into vectors. Research into this class of models increased rapidly after the original definition of convolutions over graphs [6], and today multiple architectures exist ranging from graph convolutional networks (GCNs) [21] to Chebyshev GCNs [22], graph attention networks [23], and more [6]. Siamese GNNs have also been proposed, although in most cases the graph is assumed to be given [24], or the result of a segmentation across the original image [7]. Interestingly, GNNs have been shown to be powerful architectures for processing spatio-temporal data [25], despite begin originally defined for data having no metric properties. Different methods have been proposed to encode spatial information about each node inside the GNN processing [10], which we leverage in our formulation. Finally, this article is connected to several recent works that combine CNNs and GNNs into hybrid models that represent images in terms of generic "objects," which are then processed as a graph [8].

## III. Data

Our dataset was captured in eight different electrical lines in Italy, where each line contains several dozen poles situated according to the line's so-called *morphology*. Specifically, images of the poles are taken from a helicopter that flies back and forth around the line while taking pictures. Photos are high-resolution, with an acquisition size of $6567 \times 4384$ pixels, and the bounding boxes occupy from 3% to 11% of the full picture on average. The eight lines are chosen in order to deliver a high heterogeneity in terms of morphology, and consequently high complexity in the dataset, varying from a straightforward line (seen on the left side of Fig. 4), to a nested morphology (seen on the right side of Fig. 4). Because of variability in the helicopter's flight, weather, etc., the outcome of the aerial monitoring is a set of photos in which the background and orientation for the same pole can vary by a large margin, making their classification a nontrivial issue.

The final dataset consists of 6948 training images and 2121 test images, split such that photos of a certain pole can only be found either in the training or in the test part. The training

TABLE I
RESULTS FOR THE OBJECT DETECTION MODEL

| Asset | Train samples | F1-score |
|---|---|---|
| Breaker switch | 563 | 0.98 |
| Tripolar horizontal switch | 2333 | 0.98 |
| Ceiling | 5959 | 0.97 |
| Transformer | 4728 | 0.97 |
| Steel cross arm | 18282 | 0.96 |
| Tripolar vertical switch | 1648 | 0.95 |
| Rigid insulator | 15928 | 0.94 |
| Suspended insulator | 26983 | 0.94 |
| Concrete cross arm | 4098 | 0.94 |
| Bushing | 4713 | 0.92 |
| Mixed surge arrester | 3741 | 0.90 |
| Jumper | 10009 | 0.89 |
| Unipolar switch | 1114 | 0.86 |
| STD surge arrester | 7271 | 0.83 |
| Wire to cable transition | 4864 | 0.76 |
| Horn gap surger arrester | 1251 | 0.76 |
| Nest | 2161 | 0.72 |
| Terminal connector | 5471 | 0.73 |
| Window | 2323 | 0.70 |
| Wall bushing | 3876 | 0.71 |
| Elicord cross arm | 814 | 0.73 |

TABLE II
MAIN HYPERPARAMETERS OF THE FRAMEWORK, AND THE VALUES USED IN OUR EXPERIMENTAL EVALUATION

| Hyperparameters | Values |
|---|---|
| Object detection model | EfficientDet-D2 |
| Backbone model | ResNet-34 |
| Data augmentation | Random crop and flip |
| Embedding size for the asset type | 8 |
| Thresholds on asset distance | $> 0.01$ and $< 0.06$ |
| GNN | Two Chebyshev layers |
| Size of the GNN layers | 100 |
| Graph positional embeddings | Three lowest eigenvectors |
| Batch size | 64 |
| Optimizer | Adam |
| Learning rate | $1e{-3}$ |
| Epochs | 80 |

set images are distributed among 713 distinct groups, while the validation has 268 groups. Each group contains images of the same pole, taken from different angles. Each cluster in the training and test set contains on average around nine images of the pole. In Fig. 1 two photos from different clusters are shown.

## IV. METHODS

In this section, we describe our proposed pipeline for object reidentification. Given an image $x$ containing a pole, we first train a customized object detection model to extract its composing assets, as described in Section IV-A. We then build two embeddings for each image, a vector $\mathbf{x}_v$ of visual features, as described in Section IV-B and shown in the upper part of Fig. 2, and a vector $\mathbf{x}_g$ of *graph* features, as described in Section IV-C and shown in the lower part of Fig. 2. The two embeddings are then concatenated and fed to a standard triplet loss to perform reidentification (Section IV-D and Fig. 3). To simplify reading, we also summarize the main hyperparameters of the framework and their empirically chosen values in Table II, and we provide

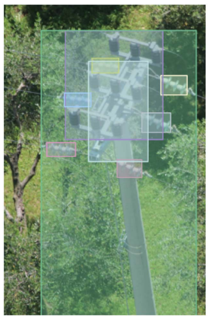


Fig. 5. Example of output from the object detection model. Note how the entire pole itself is always included as an asset in the output.

a readable pseudocode of the feature extraction pipeline in Algorithm 1.

### A. Object Detection Model

The object detection step consists of two main phases. In both phases, the object detection model is an EfficientDet-D2 [26], trained ad-hoc on a separate object detection dataset customized for our application and labeled by several domain experts, whose size is listed in Table I. We keep the object detection dataset separate from the main dataset described in Section III to avoid data leakage and to ensure generalization to unseen poles. With the help of the domain experts, we identify 21 assets, comprising all visible assets in the majority of the poles, listed in Table I.

First, an image is analyzed by an EfficientDet-D2 that identifies the presence and the type of the pole in the image with close to 100% accuracy. Second, we crop the pole from the image and run a second object detection model to identify every asset of interest among the full list of 21 assets. The number of samples listed in Table I refers to the number of crops extracted from the original pole images and used to train the object detector. We fine-tune the original EfficientDet-D2 model [26] for 50 epochs, achieving an F1-score ranging from 70% to 98% for the different assets, shown again in Table I. A sample of the output for the object detector is shown in Fig. 5. Because the focus of this article is on the proposed pipeline, we consider the object detection model as given in Algorithm 1, where we denote with $c_i$ the $i$th asset cropped from the original image.

### B. Visual Features

We now describe the pipeline for extracting the vector of visual features $\mathbf{x}_v$, as shown on the upper part of Fig. 2. Given the output of the object detector, we first extract all crops of $x$ corresponding to a detected asset. As described in Section IV-A, one of the crops always corresponds to the entire pole, which is crucial to process the background of the pole and its surrounding environment. Each crop $c_i$ is then passed to a standard ResNet-34 model pretrained on the ImageNet dataset [27] to provide a 1000-D embedding of the asset $\tilde{c}_i = \text{ResNet}(c_i)$. The weights of the ResNet-34 model are fixed during the training of the reidentification framework. To regularize training, we also perform simple data augmentation directly on the crops, by random rotations and flipping, in order to make the network more robust to changes in the orientation of the images. Finally, we perform min-pooling and max-pooling on the set of embeddings, and the resulting two 1000-D vectors for each image are concatenated to obtain the visual embedding vector $\mathbf{x}_v$ (denoted as $\mathrm{MinMaxPool}(\{\tilde{c}_i\})$ in line 5 of Algorithm 1). We also experimented with taking the embeddings from additional intermediate layers of the ResNet-34 network, as long as with different pooling strategies, and we empirically found the approach described here to be the more robust in our experiments.

## C. Graph Features

The visual pipeline described in Section IV-B exploits a pretrained CNN model, which is common when performing object reidentification (see also the discussion in Section II). In fact, pretrained CNN features have been shown to perform well in a number of computer vision scenarios and have become a standard baseline for many tasks. However, some information about the assets is lost in this preprocessing step, including the class of each asset (which is recognized by the object detection model), but also the relative positioning of the assets on the image which, as we discussed in Section I, can provide useful information to distinguish different poles. Since GNNs are powerful models to perform reasoning on sets of objects [8], we introduce a second pipeline in the framework by building a graph describing the assets, and applying a trainable GNN on top of it, as described in the following.

*1) Construction of the Graph:* We represent the assets of the image as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes (each node representing an asset) and $\mathcal{E}$ the set of edges (wherein we connect two assets depending on their relative distance on the image, as described in the following). First, to each asset $i$ of the image we associate a vector of node features $\mathbf{x}_i$ by concatenating the following four sets of features.

1) A scalar value describing the relative size (in percentage) of the bounding box containing the asset, varying between 0 and 1.
2) The confidence (in $[0, 1]$) of the object detection model.
3) A trainable 8-D embedding representing the category of the asset.
4) A positional embedding describing the position of the node inside the overall graph.

For point (4), we experimented with using the coordinates of the bounding boxes as positional features, but we found it to provide no gain in accuracy, possibly because the absolute coordinates do not provide enough invariance to a change in perspective. For this reason, we adapt the trainable positional embedding strategy introduced in [10], as described later in Section IV-C3. We also experimented with larger embedding sizes in point (3), with no significant changes.

For each pair of assets $i$ and $j$, we compute their distance $d_{ij}$ as the minimum distance between their corresponding bounding boxes, and we add the edge $(i, j)$ in $\mathcal{E}$ if the distance is contained in a prespecified range $[d_{\text{min}}, d_{\text{max}}]$, where we empirically set $d_{\text{min}} = 0.01$ and $d_{\text{max}} = 0.06$. We use the relative distance with respect to the original image size. For example, $d_{ij} = 0.05$ means that the two bounding boxes are separated by 5% the maximum possible distance in the image. We remove edges when the assets are either too far away or too close, as we found those in this range to provide the most interesting information to the GNN.

**Algorithm 1:** Pseudocode of the Feature Extraction Pipeline.

```
Input: Image $x$, pre-trained object detection model
    (ObjDet) and backbone model (ResNet)
output: Feature embedding to be used for re-identification
 1:  $\{c_i\} = \mathrm{ObjDet}(x)$ ▷Object detection, Sec. IV-A
 2:  for each asset $c_i$ do
 3:    $\tilde{c}_i = \mathrm{ResNet}(c_i)$ ▷Visual features, Sec. IV-B
 4:  end for
 5:  $x_v = \mathrm{MinMaxPool}(\{\tilde{c}_i\})$
 6:  $\mathbf{X}, \mathbf{A} = \mathrm{BuildGraph}(x)$ ▷Graph extraction,
     Sec. IV-C1
 7:  $x_g = \mathrm{GNN}(\mathbf{X}, \mathbf{A})$ ▷Graph network, Sec. IV-C2
 8:  return $\mathbf{V}[x_v \parallel x_g]$
```

*2) Graph Neural Network:* GNNs have become a standard tool to analyze graph-based data in a trainable fashion. A generic GNN is composed by a number of message-passing layers [6] defined by a node-wise operation, followed by an aggregator over each neighborhood of the node:

$$\mathbf{h}_i = \phi\left(\sum_{j\in\mathcal{N}_i} \eta(i,j)\pi(\mathbf{x}_j)\right) \tag{1}$$

where $\mathbf{h}_i$ is the new embedding for node $i$, $\phi$ is an activation function (e.g., an ReLU $\phi(s) = \max(0, s)$), $\pi(\cdot)$ is a generic fully connected network applied to each node embedding, $\mathcal{N}_i = i \bigcup \{j | (i,j) \in \mathcal{E}\}$ is the neighborhood of node $i$, and $\eta(\cdot,\cdot)$ is a symmetric weighting function. In particular, GCNs (see [21]) are a common instantiation of (1) using a single fully connected layer for $\pi$ and fixed weights for $\eta$

$$\mathbf{h}_i = \phi\left(\sum_{j\in\mathcal{N}_i} A_{ij}\mathbf{W}\mathbf{x}_j\right) \tag{2}$$

where $\mathbf{W}$ is a trainable matrix, and $\mathbf{A}$ is a generic matrix satisfying $A_{ij} = 0$ if $(i,j) \notin \mathcal{E}$. In our experimental comparison, we compare a standard GCN to more sophisticated choices, including graph attention layers (GATs) [23] and Chebyshev convolutional layers [22]. We build our complete GNN by stacking two layers of the form (1), both of size 100, each followed by a batch normalization operation. The outputs of the two convolutional layer for each node are then stacked, and the node embeddings are processed in a similar way as in Section IV-B: we perform min-pooling and max-pooling with respect to all nodes, and stack the two representations to obtain the graph vector embedding $\mathbf{x}_g$ of size 200. To obtain the final representation for the image, we concatenate $\mathbf{x}_v$ and $\mathbf{x}_g$, and perform a final trainable projection with output size 2000. In Algorithm 1, we denote by $\mathbf{X}$ the stack of all node features $\mathbf{x}_i$, and we summarize the output of the GNN block in line 7. The final linear projection is denoted by $\mathbf{V}[x_v \parallel x_g]$, where $\mathbf{V}$ is a trainable matrix.

*3) Positional Embeddings:* In their standard form, GNNs do not encode any positional information on the nodes, since their output is equivariant to any permutation of their ordering [6]. While in our case it would be possible to encode positional information by using the coordinates of the assets as node features, we found this to provide a poor empirical performance, possibly because the coordinates vary too much depending on the orientation of the aerial photo, and they are not invariant to a change of pose. To improve the GNN model, instead, we consider trainable positional embeddings following the work on spectral attention from [10]. For brevity, we only briefly describe the technique here, and we refer the interested reader to [10] for a more in-depth explanation.

Denoting by $\mathbf{A}$ the adjacency matrix of the graph (as built in Section IV-C), by $\mathbf{D}$ the diagonal degree matrix $D_{ii} = \sum_j A_{ij}$, and by $\mathbf{L} = \mathbf{D} - \mathbf{A}$ the Laplacian matrix of the graph, consider the eigendecomposition:

$$\mathbf{L} = \mathbf{U}\mathbf{V}\mathbf{U}^\top \tag{3}$$

where $\mathbf{U}$ is a square matrix containing the eigenvectors over the columns, and $\mathbf{V}$ is a diagonal matrix of eigenvalues. Equation (3) can be understood as a Fourier transform over the graph [6], with the difference that different graphs cannot be compared directly in terms of eigenvalues because they do not share the same eigenvectors. To solve this issue, similar to [10], we select the $m$ lowest eigenvectors (with $m = 3$ in the experiments), where the eigenvectors are normalized to unity norm, and padding is applied is the graph has less than $m$ nodes. The output is then processed with a trainable neural network as described in [10], except that we use a simple sum-pooling operation instead of the final Transformer block. The positional embeddings for each asset are concatenated to the other features as described in the previous section.

## D. Training Details

Given the feature encoding steps, we train the framework using a classic metric learning formulation, shown in Fig. 3. Given a triplet $(x_a, x_p, x_n)$, with $x_a$ an image of a pole (the anchor), $x_p$ another image of the same pole (positive), and $x_n$ an image of a different pole (negative), we apply the feature extraction step described earlier to obtain embeddings $(\mathbf{h}_a, \mathbf{h}_p, \mathbf{h}_n)$. Denoting by $d_{ap}$ and $d_{an}$ the Euclidean distance between anchor and positive, and anchor and negative, respectively, we minimize the triplet loss

$$l(x_a, x_p, x_n) = \max(d_{ap} - d_{an} + 0.05, 0)$$

which encourages the distance to be small for images of the same pole, and vice versa. To sample the triplets when training, for each iteration, we choose up to eight images from each pole. Then, we build 64 triplets (the batch size) by randomly sampling an anchor, and then selecting the easy positive pair and the semihard negative pair according to the strategies described in [28]. To stabilize training, we also add the center invariant regularization from [29]. We keep the ResNet-34 model from Section IV-B fixed, while we train the categorical embeddings from Section IV-A, the final projection layer from Section IV-B, and the GNN from Section IV-C, including the block to extract node positional embeddings. We train with the Adam optimization algorithm with default hypeparameters using the train/test split described in Section III. The models leverage PyTorch Geometric[1] for the GNN component and PyTorch Metric Learning[2] for the training. All hyperparameters are summarized in Table II.

## V. Experimental Results

### A. Experimental Setup

We evaluate the proposed framework using a classical metric learning setup. We first compute the embeddings generated by the trained model for all the images inside the test set (see Section III). Next, for each image, we compute the $k$ closest embeddings in the set (with $k = 30$), and we evaluate the ranking using the mean average precision (MAP), and the cumulative matching curve (CMC) computed at 1, 3, 5, and 10 [30], which we denote as CMC@1, CMC@3, CMC@5, and CMC@10 [30]. The MAP approximates the average precision of the ranking, while CMC@$p$ describes the probability that at least one image from the same pole was ranked among the top $p$ positions. We average the values for the entire test dataset to compute the final metrics.

### B. Benchmark Comparison

We start by comparing the proposed pipeline to two benchmark models, which are all trained using the same data and losses described in Section IV-D. The first benchmark is a standard siamese ResNet-34 model trained with a triplet loss on the crop of the entire electric pole. Second, we consider a model where we combine the information of the trained object detection model with the CNN, which is obtained by removing the lower branch of Fig. 2. For the proposed framework, we use Chebyshev convolutional layers (for brevity, Cheb) because of their good empirical performance, although we compare other alternatives in the following section.

We report the results of the experiments in Table III. There are several interesting observations to make from the results. First, the inclusion of the graph information significantly improves all metrics, in some cases by a very large margin (e.g., CMC@1 has a relative improvement of 6 percentage points over a standard CNN, and similar improvements are found across the other metrics). Second, simply including the information of all the assets inside the CNN is not enough to improve the performance, and in fact may even hurt certain metrics (e.g., CMC@5 drops of 1 percentage point in the second row of Table III). Overall, the results show that including the domain expertise provided by the object detection model is essential to obtaining good results, but so is the need to process this information with a model that can efficiently reason on the relations between the assets, as provided

[1][Online]. Available: https://github.com/pyg-team/pytorch_geometric

[2][Online]. Available: https://github.com/KevinMusgrave/pytorch-metric-learning

TABLE III
COMPARISON OF THE PROPOSED FRAMEWORK TO STATE-OF-THE-ART MODELS

| Model | CMC@1 | CMC@3 | CMC@5 | CMC@10 | MAP |
|---|---|---|---|---|---|
| **Proposed framework** | **0.9306** | **0.9658** | **0.9753** | **0.9829** | **0.788** |
| Siamese CNN (all assets) | 0.885 | 0.926 | 0.935 | 0.952 | 0.753 |
| Siamese CNN (pole) | 0.876 | 0.926 | 0.943 | 0.958 | 0.776 |

Best results are highlighted in bold.

TABLE IV
PERFORMANCE EVALUATION OF THE PROPOSED FRAMEWORK WHEN CHANGING THE TYPE OF GRAPH CONVOLUTIONAL LAYER

| Layer | CMC@1 | CMC@3 | CMC@5 | CMC@10 | MAP |
|---|---|---|---|---|---|
| CHEB | **0.61** $\pm0.008$ | **0.72** $\pm0.007$ | **0.78** $\pm0.006$ | **0.84** $\pm0.005$ | **0.53** $\pm0.005$ |
| GCN | 0.50 $\pm0.1$ | 0.63 $\pm0.11$ | 0.68 $\pm0.09$ | 0.79 $\pm0.03$ | 0.45 $\pm0.09$ |
| GAT | 0.50 $\pm0.05$ | 0.65 $\pm0.06$ | 0.72 $\pm0.01$ | 0.80 $\pm0.01$ | 0.46 $\pm0.01$ |

Best results are highlighted in bold.

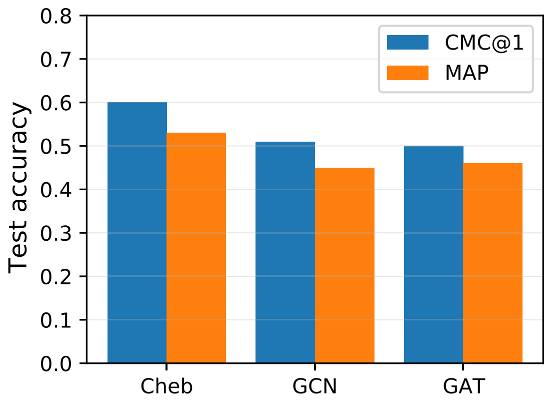


Fig. 6. Comparison of MAP and CMC@1 using different types of graph neural layers.

by the GNN. As an additional performance metric, we train a standard DBScan clustering algorithm on the obtained vectorial embeddings, and we compute the v-measure score [31] using the ID of the poles as ground truth. In this case, the baseline Siamese CNN obtains a v-measure of 82%, which increases to 85% with the proposed framework.

## C. Ablation Studies

We now perform several ablation studies to validate all the components of the pipeline. First, the results in the previous section show that the visual branch alone is insufficient to achieve optimal results. We now train the graph branch alone to isolate its effects, and also evaluate different types of graph neural layers. The results shown in Fig. 6 are obtained by using a Chebyshev GNN [22], which is a second-order polynomial graph layer. We experiment here also with simpler GCN layers [21] and GATs [6]. We provide a comparison of the results in Table IV, and a visual evaluation in Fig. 6. As expected, the graph information alone is insufficient to achieve a good accuracy, and the best performing model obtains a CMC@1 of 61%, vastly inferior also to a simple CNN siamese network. More interestingly, the result is highly dependent on the type of message passing operation, with the Cheb layer being 22%

TABLE V
ABLATION STUDIES WITH RESPECT TO THE POSITIONAL EMBEDDINGS (PES) AND THE PRESENCE OF THE EDGES

| | PEs ✗ | Edges ✓ | PEs ✗ | Edges ✗ |
|---|---|---|---|---|
| CMC@1 | 0.848 $\pm0.05$ | | 0.85 $\pm0.02$ | |
| CMC@3 | 0.92 $\pm0.02$ | | 0.91 $\pm0.02$ | |
| CMC@5 | 0.94 $\pm0.01$ | | 0.94 $\pm0.02$ | |
| CMC@10 | 0.96 $\pm0.01$ | | 0.96 $\pm0.02$ | |
| MAP | 0.72 $\pm0.04$ | | 0.72 $\pm0.02$ | |

better than the GCN and the GAT layer. We conjecture that the larger expressive power afforded by the Cheb layer is especially suited to our scenario, where graphs are composed of only a few nodes. We leave however a more comprehensive evaluation of GNN variants for a future work.

Second, we evaluate the impact of the procedure for constructing edges described in Section IV-C, and of the positional embeddings described in Section IV-C3. To this end, we train two variants of our framework. In the first, we remove all edges by setting the adjacency matrix to the identity $\mathbf{A} = \mathbf{I}$. In this case, the graph component of the pipeline becomes a set-based model that reasons only on the list of assets. For the second ablation, we train a model but we do not concatenate positional embeddings to the features of the nodes. Results for this second study are shown in Table V. Interestingly, performances vastly deteriorates in both cases, showing that it is essential for the graph component to be able to reason on the relative positioning of the assets using a suitably rich representation.

## D. Further Results

We also evaluated the results of the models visually, by comparing the hardest triplets after training. For example, in Fig. 7, we show a triplet which is mistaken by all models except the proposed one, where the two poles can only be distinguished by carefully looking at the relative orientation of the insulators (a similar example is given in Fig. 1). A small number of triplets are mistaken by all models (e.g., Fig. 8), although these are

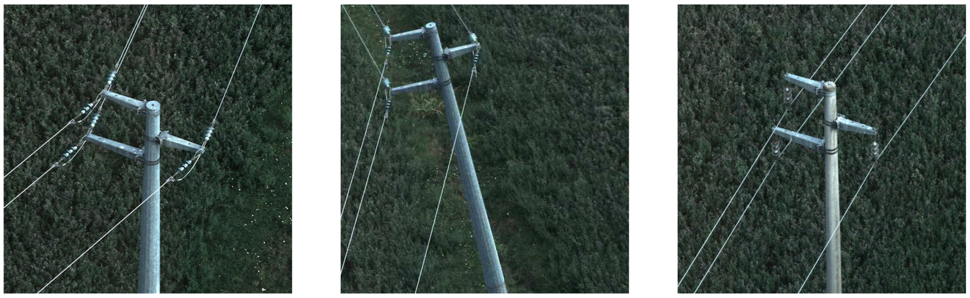


Fig. 7. Hard triplet which is recognized by the framework. Photos are anchor, positive, and negative respectively, from left to right.

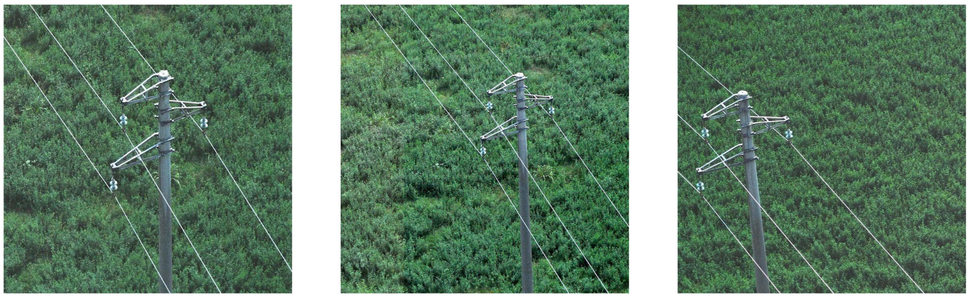


Fig. 8. Hard triplet which is not recognized by the framework. Photos are anchor, positive and negative, respectively, from left to right.

almost-impossible to distinguish also for a human operator, leaving room for possible improvements to the pipeline.

## VI. Conclusion

In this article, we proposed a new framework to perform image reidentification in the presence of complex objects composed of multiple assets, which is a common scenario in industrial applications. The proposed framework exploits a customized object detection model to extract the assets, and then feeds this information to a siamese architecture composed of a visual branch (a standard CNN) and a relational branch (a GNN). We experimentally evaluated our method on a challenging problem of electric pole reidentification, showing significant gains in performance thanks to the combined pipeline. Overall, our framework provides a viable method to incorporate multiple types of domain expertise (in this case, the assets) inside a standard deep learning solution, thus creating a hybrid pipeline that simultaneously exploits raw visual imagery and relational information. Future work will consider extending this framework to other use cases, as long as evaluating the interpretability of the pipeline, the robustness with respect to the performance of the object detection model, and more refined solutions to build the graph of assets, possibly with a joint training of the entire architecture.

## References

[1] C. Whitworth, A. Duller, D. Jones, and G. Earp, “Aerial video inspection of overhead power lines,” *Power Eng. J.*, vol. 15, no. 1, pp. 25–32, 2001.

[2] R. Jenssen et al., “Automatic autonomous vision-based power line inspection: A review of current status and the potential role of deep learning,” *Int. J. Elect. Power Energy Syst.*, vol. 99, pp. 107–120, 2018.

[3] L. Matikainen et al., “Remote sensing methods for power line corridor surveys,” *ISPRS J. Photogrammetry Remote Sens.*, vol. 119, pp. 10–31, 2016.

[4] H. Zhang, Z. Tang, Y. Xie, H. Yuan, Q. Chen, and W. Gui, “Siamese time series and difference networks for performance monitoring in the froth flotation process,” *IEEE Trans. Ind. Informat.*, vol. 18, no. 4, pp. 2539–2549, Apr. 2022.

[5] M. Zheng, S. Karanam, Z. Wu, and R. J. Radke, “Re-identification with consistent attentive siamese networks,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5735–5744.

[6] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, “Geometric deep learning: Going beyond euclidean data,” *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, Jul. 2017.

[7] U. Chaudhuri, B. Banerjee, and A. Bhattacharya, “Siamese graph convolutional network for content based remote sensing image retrieval,” *Comput. Vis. Image Understanding*, vol. 184, pp. 22–30, 2019.

[8] T. Kipf, E. van der Pol, and M. Welling, “Contrastive learning of structured world models,” in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–13.

[9] F. Locatello et al., “Object-centric learning with slot attention,” in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 11525–11538.

[10] D. Kreuzer, D. Beaini, W. L. Hamilton, V. Létourneau, and P. Tossou, “Rethinking graph transformers with spectral attention,” in *Adv. Neural Inf. Process. Syst.*, 2021. [Online]. Available: https://www.scopus.com/record/display.uri?eid=2-s2.0-85131819697&origin=resultslist

[11] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, “Scalable person re-identification: A benchmark,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1116–1124.

[12] X. Li and C. C. Loy, “Video object segmentation with joint re-identification and attention-aware mask propagation,” in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 90–105.

[13] M. Kaya and H. Ş. Bilge, “Deep metric learning: A survey,” *Symmetry*, vol. 11, no. 9, 2019, Art. no. 1066.

[14] P. K. Upadhyay and C. Nagpal, “Wavelet based performance analysis of SVM and RBF kernel for classifying stress conditions of sleep EEG,” *Sci. Technol.*, vol. 23, no. 3, pp. 292–310, 2020.

[15] F. Kennel, D. Görges, and S. Liu, “Energy management for smart grids with electric vehicles based on hierarchical MPC,” *IEEE Trans. Ind. Informat.*, vol. 9, no. 3, pp. 1528–1537, Aug. 2013.

[16] J. B. Hansen, S. N. Anfinsen, and F. M. Bianchi, “Power flow balancing with decentralized graph neural networks,” 2021, *arXiv:2111.02169*.

[17] S. Zhai, Y. Cheng, W. Lu, and Z. Zhang, “Deep structured energy based models for anomaly detection,” in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1100–1109.

[18] K. Amber, R. Ahmad, M. Aslam, A. Kousar, M. Usman, and M. Khan, “Intelligent techniques for forecasting electricity consumption of buildings,” *Energy*, vol. 157, pp. 886–893, 2018.

[19] L. Liu, T. Zhang, K. Zhao, A. Wiliem, K. Astin-Walmsley, and B. Lovell, “Deep inspection: An electrical distribution pole parts study via deep neural networks,” in *Proc. IEEE Int. Conf. Image Process.*, 2019, pp. 4170–4174.

[20] M. Gomes et al., “Mapping utility poles in aerial orthoimages using ATSS deep learning method,” *Sensors*, vol. 20, no. 21, 2020, Art. no. 6070.

[21] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–10.

[22] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” *Adv. Neural Inf. Process. Syst.*, vol. 29, pp. 3844–3852, 2016.

[23] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” in *Proc. Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://www.scopus.com/record/display.uri?eid=2-s2.0-85083953221&origin=resultslist&sort=cp-f

[24] S. I. Ktena et al., “Distance metric learning using graph convolutional networks: Application to functional brain networks,” in *Proc. Int. Conf. Med. Image Comput. Comput.- Assist. Intervention*, 2017, pp. 469–477.

[25] T. Sofianos, A. Sampieri, L. Franco, and F. Galasso, “Space-time-separable graph convolutional network for pose forecasting,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 11209–11218.

[26] M. Tan, R. Pang, and Q. V. Le, “Efficientdet: Scalable and efficient object detection,” in *Proc. IEEE/CVF Conf. Comput. Vis. pattern Recognit.*, 2020, pp. 10781–10790.

[27] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[28] H. Xuan, A. Stylianou, and R. Pless, “Improved embeddings with easy positive triplet mining,” in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, 2020, pp. 2474–2482.

[29] Y. Wu, H. Liu, J. Li, and Y. Fu, “Deep face recognition with center invariant loss,” in *Proc. Thematic Workshops ACM Multimedia*, 2017, pp. 408–414.

[30] S. Ceri, A. Bozzon, M. Brambilla, E. Della Valle, P. Fraternali, and S. Quarteroni, “An introduction to information retrieval,” in *Web Information Retrieval*, Berlin, Germany: Springer, 2013, pp. 3–11.

[31] I.-D. Borlea, R.-E. Precup, A.-B. Borlea, and D. Iercan, “A unified form of fuzzy c-means and k-means algorithms and its partitional implementation,” *Knowl.-Based Syst.*, vol. 214, 2021, Art. no. 106731.

**Alessio Devoto** received the master’s degree in computer engineering from the Sapienza University of Rome, Rome, Italy, in 2022.

He is currently with the research team, ISPAMM lab, Rome, Italy, where he is currently a Research Fellow on several topics in Deep Learning. His research interests include passionate about machine learning and deep learning with focus on graph neural networks, natural language processing, and AI explainability.

**Indro Spinelli** (Senior Member, IEEE) received the master’s degree in artificial intelligence and robotics in 2019 from the Sapienza University of Rome, Rome, Italy, where he is currently working toward the Ph.D. degree in information and communication technologies with the Department of Information Engineering, Electronics, and Telecommunications.

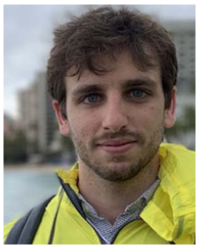
He is a member of the “Intelligent Signal Processing and Multimedia” (ISPAMM) group and his main research interests include graph deep learning and trustworthy machine learning for graph-structured data.

**Francesca Murabito** received the master’s degree in computer engineering and the Ph.D. degrees from the University of Catania, Catania, Italy, 2015 and 2018, respectively.

She is currently a Data Scientist with Enel agile room ODIN, Rome, Italy, specialized in Computer Vision, where she is responsible for the satellite imagery analysis. Her research interests include the integration of high-level knowledge into models for fine-grained visual recognition.

**Fabrizio Chiovoloni** received the graduate degree in statistics from the Sapienza University of Rome, Rome, Italy, in 2016.

After some projects in the Big Data field, he began to delve into Deep Learning and started working on Computer Vision projects, such as OCR in the wild and action recognition from video. Recently, he contributed to a reidentification algorithm based on SimCLR and Triplet Networks with the Computer Vision team, Enel, in which he is currently working on a image-based safety project.



**Riccardo Musmeci** received the graduate degree in computer engineering from Università degli Studi di Palermo, Palermo, Italy, in 2018.

He is currently an Applied Scientist in Computer Vision with Enel, Rome, Italy. He worked in several CV projects for different industries, from utilities to media. He worked mostly on CV tasks, such as reidentification, classification, object detection, and image quality assessment. He spent one year as a Research Assistant with the University of Kentucky, Lexington, KY, USA.

**Simone Scardapane** received the master’s degree in artificial intelligence and robotics and the Ph.D. degree in information and communication technology from the Sapienza University of Rome, Rome, Italy, in 2011 and 2016, respectively.

He is currently a Tenure-Track Assistant Professor with the Sapienza University of Rome, where he works on deep learning applied to audio, video, and graphs, and their application in distributed and decentralized environments. He has authored more than 70 articles in the fields of machine and deep learning. He is the Chair of the Statistical Pattern Recognition Techniques TC of the International Association for Pattern Recognition.