Re-identification of objects from aerial photos with hybrid siamese neural networks

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

***Abstract*—In this paper, we consider the task of re-identifying 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., sub-components) present on the objects, and a siamese neural network learns to re-identify 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 re-identification 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*—Object re-identification, graph neural networks, siamese networks, object detection, energy grids**

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 kilometres of lines spanning the entire country, alongside thousands of assets including high-voltage lines and transformer stations.

In this paper, 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 *re-identify* the same object from different pictures are required [4].

In the literature, object re-identification 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 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 re-identification 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 paper is motivated by the following observation: electric poles, like many other components of the energy grids, are

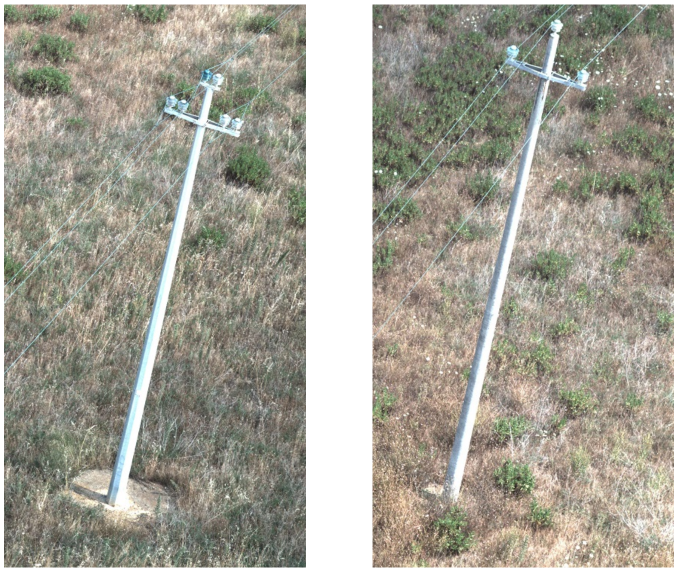


Fig. 1: An example of a difficult pair taken from the dataset: the two poles have a very similar structure and are set against similar background. However, they can be distinguished by carefully looking at their assets (in particular, their isolators).

A. Devoto, I. Spinelli, and S. Scardapane are with the Department of Information Engineering, Electronics and Telecommunications (DIET), Sapienza University of Rome, Italy (e-mails: indro.spinelli@uniroma1.it, simone.scardapane@uniroma1.it). F. Murabito, F. Chiovoloni, and R. Musmeci are with Enel S.p.A.

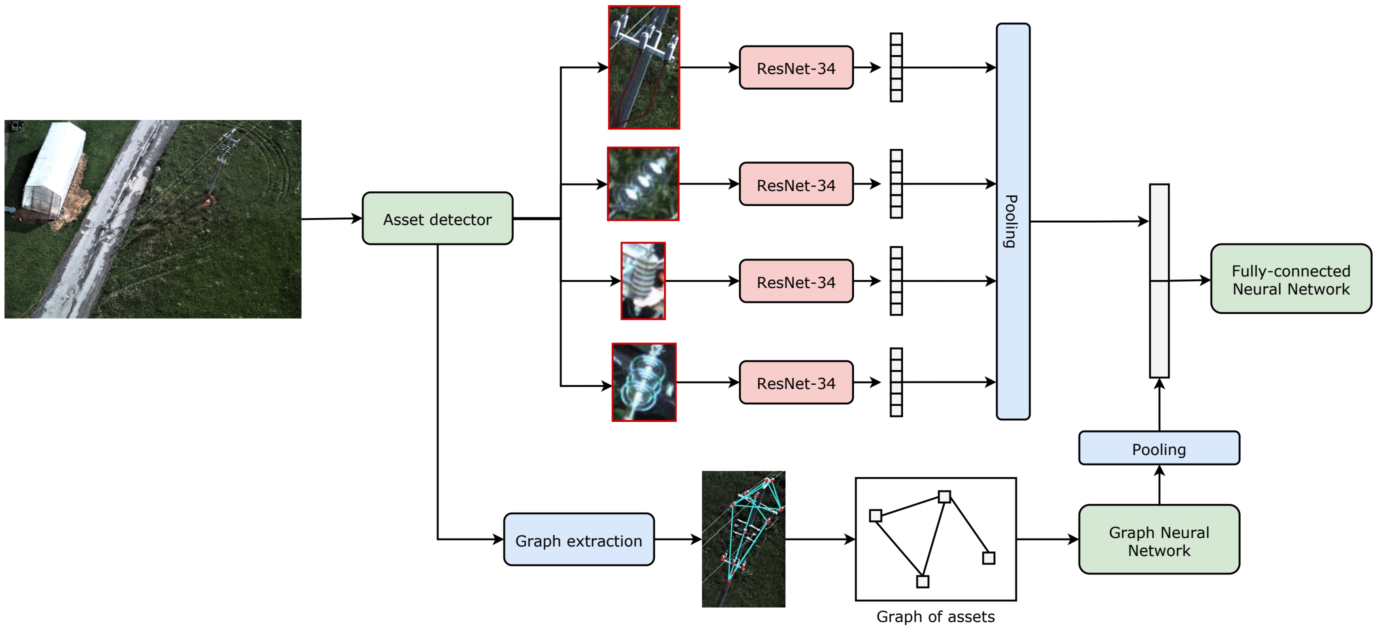


Fig. 2: Schematic diagram of the feature extraction pipeline. A customized object detection model simultaneously detects the bounding box of the pole and the bounding boxes of its composing assets (e.g., transformers). Then, a graph is built where each node corresponds to a detected asset and edges encode 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.

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 paper.

Contribution of the paper: 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 paper we propose and empirically validate a novel framework for object re-identification 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 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), as shown respectively in the upper and lower parts of Fig. 2 and in Fig. 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 performance if compared to a standard object re-identification 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 bio-informatics to social networks, the framework proposed in this paper, to the best of our 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 the paper is structured 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, before describing some concluding remarks in Section VI.

II. Related works

1) Object re-identification: Object re-identification 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 paper 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 *re-identify* the same pole from different images to plan potential maintenance activities.

3) Graph neural networks: Graph neural networks 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 paper is connected to several recent works that combine convolutional neural networks and GNNs into hybrid models that represent images in terms of generic 'objects' which are then processed as a graph [8].

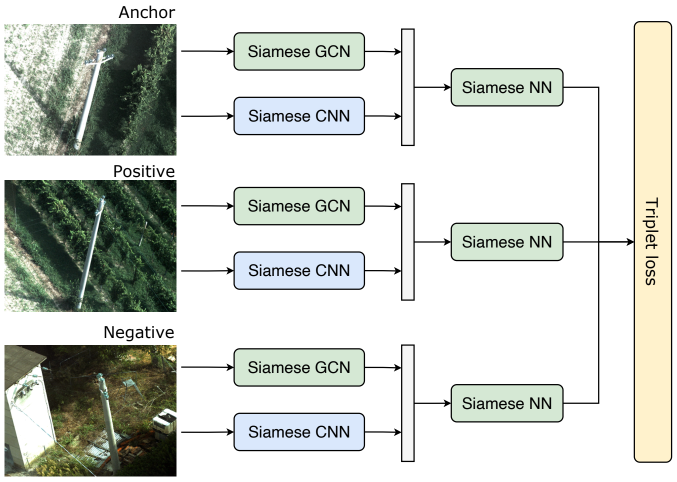
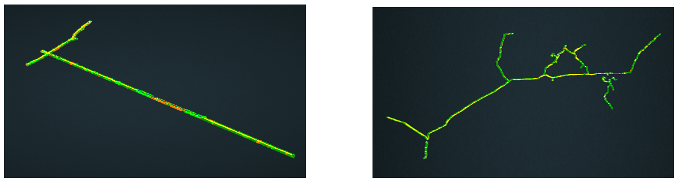


Fig. 3: During training, we randomly sample a triplet of objects, including a positive pair (two photos of the same pole), and a negative pair (two photos of different poles). The images are processed according to the pipeline in Fig. 2, on top of which we train with a proper metric learning loss. The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



(a) Simple morphology (b) Nested morphology

Fig. 4: A sample of two lines considered when building the dataset.

III. Data

Our dataset was captured in 8 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 6567×4384 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 non-trivial 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 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

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

the training and test set contains on average around 9 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 re-identification. 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 re-identification (Section IV-D and Fig. 3). To simplify reading, we also summarize the main hyper-parameters of the framework and their empirically chosen values in Tab. II, and we provide 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 Tab. 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.

Firstly, 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. Secondly, 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 Tab. 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 Tab. I. A sample of the output for the object detector is shown in Fig. 5. Because the focus of the paper 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 pre-trained on the ImageNet dataset [27] to provide a 1000-dimensional 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 re-identification 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-dimensional vectors for each image are concatenated to obtain the visual embedding vector $\mathbf{x}_v$ (denoted as $\text{MinMaxPool}(\{\tilde{c}_i\})$ 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 pre-trained CNN model, which is common when performing object re-identification (see also the discussion in Section II). In fact, pre-trained 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 pre-processing 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 next.

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 below). First, to each asset i of the image

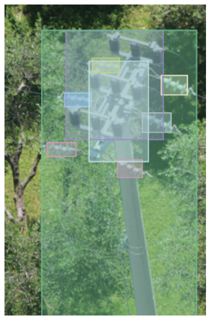


Fig. 5: Example of output from the object detection model. Note how the entire pole itself is always included as an asset in the output.

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: $\{c_i\} = \text{ObjDet}(x)$          ▷ Object detection, Sec. IV-A
2: for each asset $c_i$ do
3:     $\tilde{c}_i = \text{ResNet}(c_i)$          ▷ Visual features, Sec. IV-B
4: end for
5: $x_v = \text{MinMaxPool}(\{\tilde{c}_i\})$
6: $\mathbf{X}, \mathbf{A} = \text{BuildGraph}(x)$   ▷ Graph extraction, Sec. IV-C.1
7: $x_g = \text{GNN}(\mathbf{X}, \mathbf{A})$          ▷ Graph network, Sec. IV-C.2
8: return $\mathbf{V}\left[x_v \parallel x_g\right]$
```

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-dimensional 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-C.3. 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 pre-specified 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.

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, ϕ is an activation function (e.g., a 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, graph convolutional networks (GCN, [21]) are a common instantiation of (1) using a single fully connected layer for π and fixed weights for η:

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

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 [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 $\mathbf{V}\left[x_v \parallel x_g\right]$, 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 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. (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 8 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 semi-hard 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 graph neural network from Section IV-C, including the block to extract node positional embeddings. We train with the Adam optimization algorithm with default hyper-parameters 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 hyper-parameters are summarized in Tab. II.

TABLE II: Main hyper-parameters 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
Graph neural network	2 Chebyshev layers
Size of the GNN layers	100
Graph positional embeddings	3 lowest eigenvectors
Batch size	64
Optimizer	Adam
Learning rate	$1e-3$
Epochs	80

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.

[1]https://github.com/pyg-team/pytorch_geometric
[2]https://github.com/KevinMusgrave/pytorch-metric-learning

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 next section.

We report the results of the experiments in Tab. 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 performance, and in fact may even hurt certain metrics (e.g., CMC@5 drops of 1 percentage point in the second row of Tab. 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 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.

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

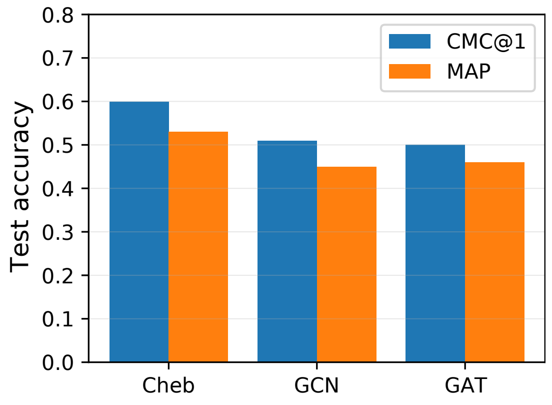


Fig. 6: Comparison of MAP and CMC@1 using different types of graph neural layers.

C. Ablation studies

We now perform several ablation studies to validate all the components of the pipeline. Firstly, 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 graph neural network [22], which is a second-order polynomial graph layer. We experiment here also with simpler GCN layers [21] and graph attention layers (GATs) [6]. We provide a comparison of the results in Tab. 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% 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 graph neural network variants for a future work.

Secondly, we evaluate the impact of the procedure for constructing edges described in Section IV-C, and of the positional embeddings described in Section IV-C.3. 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 Tab. 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 almost-impossible to distinguish also for a human operator, leaving room for possible improvements to the pipeline.

VI. Future works

In this paper, we proposed a new framework to perform image re-identification 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 convolutional neural network) and a relational branch (a graph neural network). We experimentally evaluated our method on a challenging problem of electric pole re-identification, 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 Engineering Journal*, vol. 15, no. 1, pp. 25–32, 2001.

[2] R. Jenssen, D. Roverso, *et al.*, "Automatic autonomous vision-based power line inspection: A review of current status and the potential role of deep learning," *International Journal of Electrical Power & Energy Systems*, vol. 99, pp. 107–120, 2018.

[3] L. Matikainen, M. Lehtomäki, E. Ahokas, J. Hyyppä, M. Karjalainen, A. Jaakkola, A. Kukko, and T. Heinonen, "Remote sensing methods for power line corridor surveys," *ISPRS Journal of Photogrammetry and Remote sensing*, vol. 119, pp. 10–31, 2016.

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** ±0.008	**0.72** ±0.007	**0.78** ±0.006	**0.84** ±0.005	**0.53** ±0.005
GCN	0.50 ±0.1	0.63 ±0.11	0.68 ±0.09	0.79 ±0.03	0.45 ±0.09
GAT	0.50 ±0.05	0.65 ±0.06	0.72 ±0.01	0.80 ±0.01	0.46 ±0.01

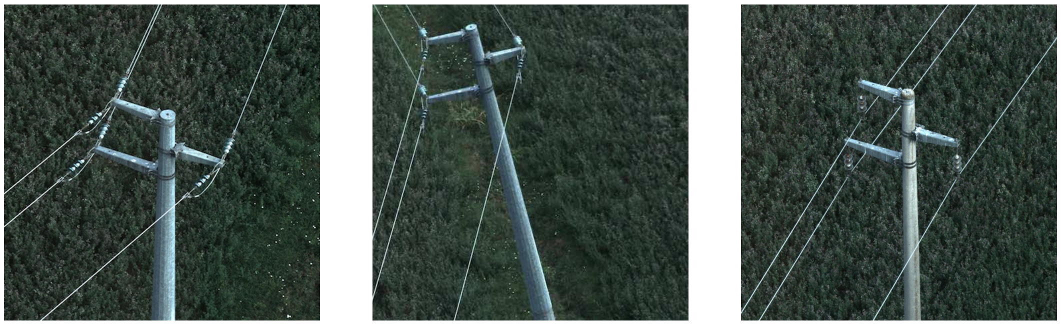


Fig. 7: A hard triplet which is recognized by the framework. Photos are anchor, positive, and negative respectively, from left to right.

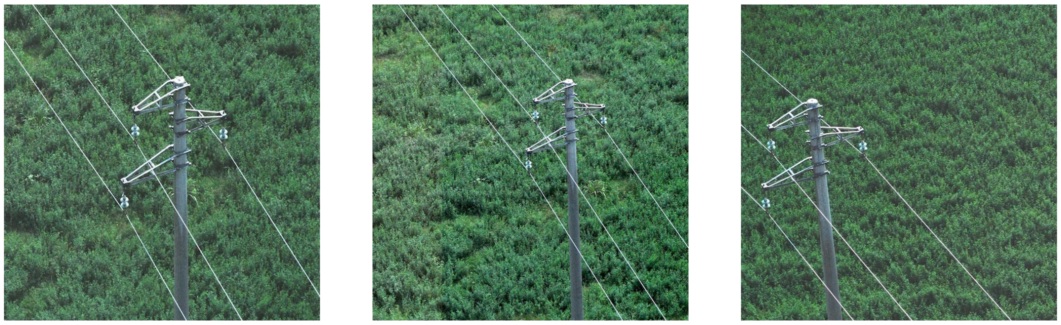


Fig. 8: A hard triplet which is not recognized by the framework. Photos are anchor, positive and negative respectively, from left to right.

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 ±0.05	0.85 ±0.02
CMC@3	0.92 ±0.02	0.91 ±0.02
CMC@5	0.94 ±0.01	0.94 ±0.02
CMC@10	0.96 ±0.01	0.96 ±0.02
MAP	0.72 ±0.04	0.72 ±0.02

[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 Transactions on Industrial Informatics*, 2021.

[5] M. Zheng, S. Karanam, Z. Wu, and R. J. Radke, "Re-identification with consistent attentive siamese networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5735–5744, 2019.

[6] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[7] U. Chaudhuri, B. Banerjee, and A. Bhattacharya, "Siamese graph convolutional network for content based remote sensing image retrieval," *Computer Vision and 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 *International Conference on Learning Representations (ICLR)*, 2020.

[9] F. Locatello, D. Weissenborn, T. Unterthiner, A. Mahendran, G. Heigold, J. Uszkoreit, A. Dosovitskiy, and T. Kipf, "Object-centric learning with slot attention," vol. 33, pp. 11525–11538, 2020.

[10] D. Kreuzer, D. Beaini, W. L. Hamilton, V. Létourneau, and P. Tossou, "Rethinking graph transformers with spectral attention," *Advances in Neural Information Processing Systems*, 2021.

[11] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable person re-identification: A benchmark," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1116–1124, 2015.

[12] X. Li and C. C. Loy, "Video object segmentation with joint re-identification and attention-aware mask propagation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 90–105, 2018.

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

[14] P. K. Upadhyay and C. Nagpal, "Wavelet based performance analysis of svm and rbf kernel for classifying stress conditions of sleep eeg," *Science and Technology*, 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 Transactions on Industrial Informatics*, vol. 9, no. 3, pp. 1528–1537, 2012.

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

[17] S. Zhai, Y. Cheng, W. Lu, and Z. Zhang, "Deep structured energy based models for anomaly detection," in *International Conference on Machine Learning (ICML)*, pp. 1100–1109, PMLR, 2016.

[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 *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 4170–4174, IEEE, 2019.

[20] M. Gomes, J. Silva, D. Gonçalves, P. Zamboni, J. Perez, E. Batista, A. Ramos, L. Osco, E. Matsubara, and J. Li, "Mapping utility poles in aerial orthoimages using atss deep learning method," *Sensors*, vol. 20, no. 21, p. 6070, 2020.

[21] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[22] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in Neural Information Processing Systems*, vol. 29, pp. 3844–3852, 2016.

[23] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations (ICLR)*, 2018.

[24] S. I. Ktena, S. Parisot, E. Ferrante, M. Rajchl, M. Lee, B. Glocker, and D. Rueckert, "Distance metric learning using graph convolutional networks: Application to functional brain networks," in *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 469–477, Springer, 2017.

[25] T. Sofianos, A. Sampieri, L. Franco, and F. Galasso, "Space-time-separable graph convolutional network for pose forecasting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11209–11218, 2021.

[26] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10781–10790, 2020.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[28] H. Xuan, A. Stylianou, and R. Pless, "Improved embeddings with easy positive triplet mining," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2474–2482, 2020.

[29] Y. Wu, H. Liu, J. Li, and Y. Fu, "Deep face recognition with center invariant loss," in *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, pp. 408–414, 2017.

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

[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," *Knowledge-Based Systems*, vol. 214, p. 106731, 2021.

Alessio Devoto is a recently graduated engineer, passionate about Machine Learning and Deep Learning. After earning his Master's degree in computer engineering at Sapienza University of Rome, he joined the research team at the ISPAMM lab where he works as research fellow on several topics in Deep Learning. His research focus is on graph neural networks, natural language processing, and AI explainability.

Indro Spinelli (SM'19) received a master's degree in artificial intelligence and robotics in 2019 from Sapienza University of Rome, Italy, where he is currently working towards a PhD in 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, trustworthy machine learning for graph-structured data.

Francesca Murabito received her Computer Engineering degree from the University of Catania, where she also obtained a PhD with a thesis on Fine-Grained Visual Categorization. Currently, she is a Data Scientist part of the Enel agile room ODIN, specialized in Computer Vision, where she is responsible for the Satellite Imagery Analysis. Her research activity mainly concerns the integration of high-level knowledge into models for fine-grained visual recognition.

Fabrizio Chiovoloni graduated in statistics at Sapienza University of Rome. 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 re-identification algorithm based on SimCLR and Triplet Networks in the Computer Vision team at Enel, in which he is currently working on a image-based safety project.

Riccardo Musmeci is an Applied Scientist in Computer Vision for Enel. He worked in several CV projects for different industries, from utilities to media. He worked mostly on CV tasks such as re-identification, classification, object detection, and image quality assessment. In 2018 he graduated in Computer Engineering at Università degli Studi di Palermo, and spent 1 year as a research assistant at the University of Kentucky.

Simone Scardapane is currently a Tenure-Track Assistant Professor with the Sapienza University of Rome, Italy, 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 Chair of the Statistical Pattern Recognition Techniques TC of the International Association for Pattern Recognition.