POLITECNICO DI TORINO
Repository ISTITUZIONALE

Boosting zero-shot learning through neuro-symbolic integration

Original
Boosting zero-shot learning through neuro-symbolic integration / Manigrasso, F., Lamberti, F., Morra, L.. - In: PATTERN RECOGNITION. - ISSN 0031-3203. - 170:(2026). [10.1016/j.patcog.2025.111869]

Availability:
This version is available at: 11583/3001317 since: 2025-06-26T16:35:29Z

Publisher:
Elsevier

Published
DOI:10.1016/j.patcog.2025.111869

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

19 November 2026

Pattern Recognition 170 (2026) 111869



Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr





Boosting zero-shot learning through neuro-symbolic integration

Francesco Manigrasso, Fabrizio Lamberti, Lia Morra *

Politecnico di Torino, Corso Duca degli Abruzzi 24, Turin, 10129, Italy

ARTICLE INFO

Keywords:
Neuro-symbolic AI
Logic tensor networks
Zero shot learning

ABSTRACT

Zero-shot learning (ZSL) aims to train deep neural networks to recognize objects from unseen classes, starting from a semantic description of the concepts. Neuro-symbolic (NeSy) integration refers to a class of techniques that incorporate symbolic knowledge representation and reasoning with the learning capabilities of deep neural networks. However, to date, few studies have explored how to leverage NeSy techniques to inject prior knowledge during the training process to boost ZSL capabilities. Here, we present Fuzzy Logic Prototypical Network (FLPN) that formulates the classification task as prototype matching in a visual-semantic embedding space, which is trained by optimizing a NeSy loss. Specifically, FLPN exploits the Logic Tensor Network (LTN) framework to incorporate background knowledge in the form of logical axioms by grounding a first-order logic language as differentiable operations between real tensors. This prior knowledge includes class hierarchies (classes and macroclasses) along with robust high-level inductive biases. The latter allow, for instance, to handle exceptions in class-level attributes and to enforce similarity between images of the same class, preventing premature overfitting to seen classes and improving overall performance. Both class-level and attribute-level prototypes through an attention mechanism specialized for either convolutional- or transformer-based backbones. FLPN achieves state-of-the-art performance on the GZSL benchmarks AWA2 and SUN, matching or exceeding the performance of competing algorithms with minimal computational overhead. The code is available at https://github.com/FrancescoManigrass/FLPN.

1. Introduction

The field of computer vision stands to benefit from methods that promote the combination of learning and reasoning [1–3]. Neuro-symbolic (NeSy) artificial intelligence aims to integrate symbolic representation and reasoning with perception and machine learning within a unified framework, enhancing the explainability, robustness, and flexibility of deep learning models. We focus here in particular on a class of NeSy techniques that aim to incorporate previous knowledge expressed in symbolic form into the training objectives of deep neural networks, compensating for imperfect or limited supervision provided by the labeled examples [4–9]. This is particularly relevant in Zero-Shot Learning (ZSL), which aims to recognize novel objects by linking both known and unknown categories with some kind of external information, usually in the form of class attributes or descriptions [10]. In this paper, we introduce the Fuzzy Logic Prototypical Network (FLPN), the latest addition to a series of NeSy approaches for ZSL image recognition that includes Proto-LTN [8] and FLVN [9].

The effectiveness of neural networks stems from their ability to learn distributed representations that, unlike symbolic representations, induce a rich similarity space, in which semantically close concepts (or inputs) are close in distance [11]. FLPN builds on Logic Tensor Networks (LTNs) [2], which can be classified as tightly-coupled, distributed NeSy architectures (classified as Level 5 according to Kautz's taxonomy [12]). In FLPN, the NeSy component acts as a soft constraint or regularizer, guiding the network towards learning a more effective representation, which is exploited at inference time to classify new samples. FLPN thus seeks to surpass the dichotomy between symbolic and distributed representations by associating symbols (e.g., class labels) with prototypes (a process denoted as *grounding*). This approach offers several advantages: the ability to exploit semantic relationships between symbols through distance, easier manipulation of abstract concepts in vectorized form, and improved interpretability of prototypes over simple labels through visualization techniques like t-SNE [8].

Briefly, the core idea of LTN is that each logical axiom (e.g., class hierarchies, attribute constraints) becomes a differentiable constraint incorporated into the loss function. This ensures that the neural network not only learns from the supervising signal (i.e., the seen classes), but also must satisfy the symbolic constraints, which enforce a proper organization of the prototype and feature space. Specifically, in FLPN, class hierarchies (macro-classes) provide a coherent semantic structure, guiding the network to "place" unseen classes appropriately

* Corresponding author.
E-mail addresses: francesco.manigrasso@polito.it (F. Manigrasso), fabrizio.lamberti@polito.it (F. Lamberti), lia.morra@polito.it (L. Morra).

https://doi.org/10.1016/j.patcog.2025.111869
Received 7 November 2024; Received in revised form 2 April 2025; Accepted 14 May 2025
Available online 25 June 2025
0031-3203/© 2025 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

when they share characteristics with seen classes. The attribute axioms and the masking mechanism prevent the network from over-relying or misrepresenting attributes that do not appear in all examples of a class, thereby improving the separability of different attributes in the embedding space. Similarity axioms promote consistent clustering of images within the feature space. Overall, these symbolic constraints act as an inductive bias, restricting the solution space and reducing the risk of overfitting to the training (seen) classes.

In summary, our contributions are the following ones:

- we introduce the FLPN architecture, a novel NeSy approach to ZSL recognition that combines the flexibility of prototypical networks with a semantics-enhanced training procedure empowered by the LTN framework. FLPN combines elements from our previous works [8,9] and extends them with the extraction of attribute-level information and a more comprehensive set of axioms;
- through experiments on three different benchmarks, we show that FLPN outperforms or closely matches existing ZSL techniques with minimal overhead at inference time and without the need to generate additional samples at training time;
- we conduct extensive experiments and ablation studies to understand the role of different backbones (CNNs and transformers), LTN semantics, and knowledge base formulations.

The rest of the paper is organized as follows. Section 2 presents the foundational concepts of the LTN framework and related studies. Section 3 describes the proposed architecture. The performance of the model in the ZSL and GZSL contexts over established benchmarks is evaluated in Sections 4 and 5. Finally, Section 6 discusses the results and potential future research directions.

2. Related work

2.1. Zero-shot learning

ZSL is a machine learning paradigm in which a model is trained to recognize and classify objects from unseen categories by leveraging auxiliary information, such as semantic attributes, that relates unseen categories to seen training categories. Generalized Zero-Shot Learning (GZSL) extends the ZSL paradigm to recognize and classify objects from seen and unseen categories. Unlike ZSL, GZSL requires the model to effectively handle a more complicated scenario by having to distinguish seen and unseen classes in the same space. Several methods have been developed for ZSL, which we will roughly categorize in the following as embedding-based (either with fixed embeddings or trained end-to-end) and generative methods.

In **embedding-based** methods, semantic attributes and visual data are integrated by mapping them into a shared space. Some techniques project images into the attribute space through an embedding function, treating semantic attributes as the common space [13,14]. FLPN projects instead the semantic attributes into the image embedding space using a projection function and uses the image embedding space as the common reference space, similarly to many methods dating back to DEM, in order to address the *hubness* problem [8,15,16]. There are also methods [17–19] that employ a separate shared space, distinct from both the image and attribute domains. To mitigate overfitting to known categories, these approaches often use pseudo-labeling techniques or operate in a transductive manner [18], accessing unlabeled images from new categories during the training phase.

More recently, embedding-based methods were extended to enable end-to-end training of the backbone [9,13,20,21]. These methods, which include FLPN and will be denoted in the following as **end-to-end embedding-based** methods, are capable of identifying the essential areas of the image for class categorization and thus improving the embedding space during training [13,21,22]. Previous research has exploited attributes to develop class-level representations using regularization methods [18] or contrastive learning [13] to prevent overfitting of the seen classes. Building on previous research [22–27], FLPN enhances its performance by combining attribute and class representations and leveraging the logical reasoning capability of a NeSy framework.

Compared to existing methods that employ a visual-semantic space, FLPN differs in the way it promotes alignment of the visual and semantic information. Techniques such as CoAR-ZSL [23] and APN [28] define an auxiliary loss in the semantic space, after performing a visual-to-semantic mapping. Other techniques [29] target the alignment between a visual and a semantic embedding space, where the latter is defined based on the text embedding of the attributes, a textual description of the class, or other sources of textual information.

DCA-VAE [26] employs a cross-aligned VAE to map visual and semantic features into a shared latent space, addressing the hubness problem via a discriminative projection. Meanwhile, ViFR [27] refines visual features both before and after semantic mapping – using Pre- and Post-Feature refinement modules plus specialized losses – to mitigate cross-dataset bias from pre-trained models.

Our proposed NeSy approach, on the other hand, exploits structured knowledge (e.g., extracted from knowledge graphs) to regularize the learning process. On the one hand, it is a more flexible approach as the LTN knowledge base can encode different sources of information and constraints, but requires more tuning as it introduces additional hyper-parameters (such as the choice of grounding and the connective semantics).

Previous approaches such as ProtoLTN [8], FLVN [9] and CSNL [7] have successfully integrated symbolic knowledge with neural embeddings for ZSL, but there remain limitations in capturing attribute-level information and enforcing comprehensive logical constraints. FLPN addresses these gaps through several key innovations. FLPN introduces attribute-level prototypes in addition to class and macroclass prototypes, enabling finer recognition and better distinction between subtle visual differences in unseen classes. Furthermore, FLPN integrates a more diverse set of logical axioms within the LTN framework, including class hierarchies, exception handling, and similarity constraints. This comprehensive knowledge base provides stronger inductive biases and improves generalization to unseen classes by ensuring logical consistency and semantic coherence across different levels of abstraction. Additionally, FLPN employs specialized attention mechanisms tailored for both convolutional and transformer-based backbones, facilitating accurate attribute localization within images. By embedding prior knowledge directly into the feature representation and utilizing a unified embedding space, FLPN achieves state-of-the-art performance with reduced computational overhead compared to generative NeSy approaches. Lastly, FLPN is compatible with both CNN and transformer backbones, leveraging the advanced feature extraction capabilities and attention mechanisms of transformer models to achieve substantial performance gains on complex datasets.

Generative techniques utilize auxiliary models such as Generative Adversarial Networks (GANs) to create synthetic instances that represent unobserved classes by learning a conditional probability distribution for each class [30–35]. Feature generation models have recently been combined with embedding-based models in a contrastive framework [33]. Generative approaches require prior knowledge of unseen classes to generate training data. In contrast, the FLPN training approach relies on choosing a set of seen classes and only requires knowledge of the complete class hierarchy during the training phase, facilitating the inclusion of new unseen classes in the training process. FLPN is also compatible with generative techniques and can, in principle, integrate synthetic samples during training.

Prototypical Networks map inputs into a shared space, where each class is represented by a "prototype" (the average embedded sample). In zero-shot settings, ProtoNets can embed textual or attribute-based descriptions of novel classes, enabling classification of unseen categories. Recent extensions enhance ProtoNets for ZSL/GZSL. APN [36]

adds an attribute-centric prototype layer for fine-grained localization, while generative approaches like E-PGN [37] produce synthetic prototypes for novel classes via episodic training. DPPN [38] refines both attribute and category prototypes progressively, DSP [39] iteratively adjusts predefined prototypes to better match real data, and PPN [40] links local image regions with distinct semantic components. FLPN simultaneously extracts class, macroclass, and attribute prototypes within one embedding space, guided by a fuzzy knowledge base to maintain hierarchical consistency and accommodate partially visible attributes. Through continuous re-evaluation of prototype relationships in the LTN framework and features like attribute masking, FLPN aims for robust performance in ZSL and GZSL—even when dealing with missing or non-visible attributes.

Finally, more recently **transformers-based** models, that is, techniques that integrate transformers as their backbone, have shown enhanced robustness compared to convolutional neural networks, or CNNs. According to Du et al. [23], it is feasible not only to use global features for embedding spaces, but also to apply Visual Transformer (ViT) embedding patches as an attention mechanism to identify attribute-level features. Recent transformer-based models with dual attention pathways improve visual-semantic alignment, addressing the semantic gap in zero-shot learning [41]. Incorporating topological structure enhances the classification of unseen categories [42], while linking sample clusters to class prototypes mitigates domain shift, reinforcing generalized zero-shot learning [35]. Both Hou et al. [43] and Ran et al. [44] refine visual-semantic alignment by combining attention mechanisms with structured feature representations, leading to improved classification of unseen categories.

2.2. NeSy AI for semantic image interpretation

This research is part of a larger effort within the NeSy domain to incorporate symbolic information from the outset of training [3]. In particular, LTNs are a modular framework designed to incorporate FOL constraints [1,2,6]. They have been effectively applied in various computer vision tasks, such as object detection [4,45], ZSL [8,9] and visual reasoning tasks such as Sudoku puzzle classification [46]. Several studies highlight the potential to compensate for limited supervision by incorporating prior knowledge [4,5,9].

Within the LTN framework, grounding $\mathcal{G}$ is the process of assigning real-valued semantics to logical symbols. In this framework, individuals are represented as vectors in a high-dimensional space and each predicate symbol, denoted by $p \in \mathcal{P}$, is mapped to a function $\mathcal{G}(p) \rightarrow [0, 1]$. An example is the `isOfClass` predicate commonly used in NeSy architectures, which measures the probability that a given term belongs to a specific class c [4,8]. Predicates are typically implemented using neural tensor networks (multi-layer perceptrons, or MLPs), but they can also be interpreted as measures of distance between terms (or better, between their vectorized groundings) and class-defining attribute matrices [9] or prototypes [8]. Complex expressions are constructed using a combination of logical connectives ($\wedge$, $\vee$, $\rightarrow$, $\neg$) and quantifiers ($\forall$, $\exists$), which are grounded through mathematical operations and neural network architectures. The training objective is established by defining a knowledge base $\mathcal{K}$ of the FOL axioms and then optimizing for the best satisfiability (sat) problem by maximizing the truth values of all formulas $\phi \in \mathcal{K}$ $\Theta^* = \text{argmax}_\Theta \left(\hat{\mathcal{G}}_\theta \left(\bigwedge_{\phi \in \mathcal{K}} \phi \right) - \lambda \|\Theta\|_2^2 \right)$.

3. Methodology

The architecture of FLPN is depicted in Fig. 1. Its main components include the backbone, which is responsible for extracting the features from images (detailed in Section 3.2), and the prototypical network (detailed in Section 3.3), which generates prototypes for different classes, macroclasses, and attributes (as illustrated in Section 3.3). The architecture was tested with two distinct backbones: a CNN (ResNet-101) and a transformer (ViT). Sections 3.4 and 3.5 detail the process of integrating prototypes and images into a unified embedding space, which serves as the core of the knowledge base $\mathcal{K}$, continually refined throughout the training phase [6]. Section 3.6 illustrates how optimizing the satisfiability of the grounded knowledge base relates to losses commonly used in prototype and contrastive learning.

3.1. ZSL and GZSL settings

The ZSL task entails recognizing objects from previously unseen classes by exploiting some form of auxiliary knowledge, usually attribute-based, and learning a mapping on seen classes. GZSL extends the ZSL setting by assuming that both seen and unseen classes are present at test time. Formally, the training dataset is defined as $S = \{(x_n, y_n) \mid n = 1, \dots, N\}$, with the training categories denoted by $y_n \in Y^{tr}$. In the GZSL setting, a test image can be classified into seen or unseen classes, represented as $Y^{ts+tu} \subset Y$, whereas in ZSL we assume that seen classes are not present at test time ($Y^{tu} \subset Y$).

Let us define C attribute vectors $\{a^1, a^2, a^3, \dots, a^C\}$, where each vector $a^c \in \mathbb{R}^A$ denotes attributes typically associated with a class $c \in C$. The entries in the attribute matrix $a = \left[a^1, \dots, a^C\right]$ are non-binary: a zero value indicates an undefined attribute for the class, a negative value indicates that the attribute is not present in the class, and a positive value indicates the attribute is present with a relevance proportional to the value [10]. Similarly to previous methods [8,15,23], we map the input images and the attribute matrix onto a *common embedding space* by introducing functions f_θ and g_θ, defined in Section 3.2, to map images from their native domain, and a collection of functions (defined in Section 3.3) that transform attribute vectors into *prototypes* into the common embedding space.

3.2. Image feature extraction

Two feature extraction methods are presented, based either on a CNN or a ViT. These modules gather detailed image information, producing feature embeddings at the class, macroclass, and attribute levels to minimize distances to the representative prototypes.

3.2.1. Global features representation

Similarly to PROTO-LTN [8], images are mapped into a shared embedding space in which both classes and macroclasses are represented through function:

$$f_\theta : \mathbb{R}^{Ch \times W \times H} \rightarrow \mathbb{R}^M \tag{1}$$

where $D = Ch \times W \times H$ is the input image domain, W, H and Ch are the image width, height and number of channels, M is the dimensionality of the feature space, and θ are the model parameters.

CNN-based. The global features are extracted directly from the last layer of the Resnet-101 backbone.

ViT-based. The images are divided into square patches and passed through the transformer encoder to extract the feature tensors. In contrast to CNN-based architectures, the class-level feature is represented by adding a learnable classification token [CLS] as described in [23].

3.2.2. Attribute features representation

An additional embedding function converts each image into suitable attribute embeddings through an attention mechanism:

$$g_\theta : \mathbb{R}^{Ch \times W \times H} \rightarrow \mathbb{R}^{A \times M} \tag{2}$$

where $D = Ch \times W \times H$ is the input image space, M the dimensionality of the shared feature space, A the number of attributes, and θ a set of trainable weights. Similarly to previous work [23], g_θ incorporates an attention mechanism on top of the shared feature extraction module used by f_θ, so that the backbone parameters are shared among the two functions.

CNN-based module. We added four convolutional layers after each stage to generate feature tensors. These tensors are combined to form

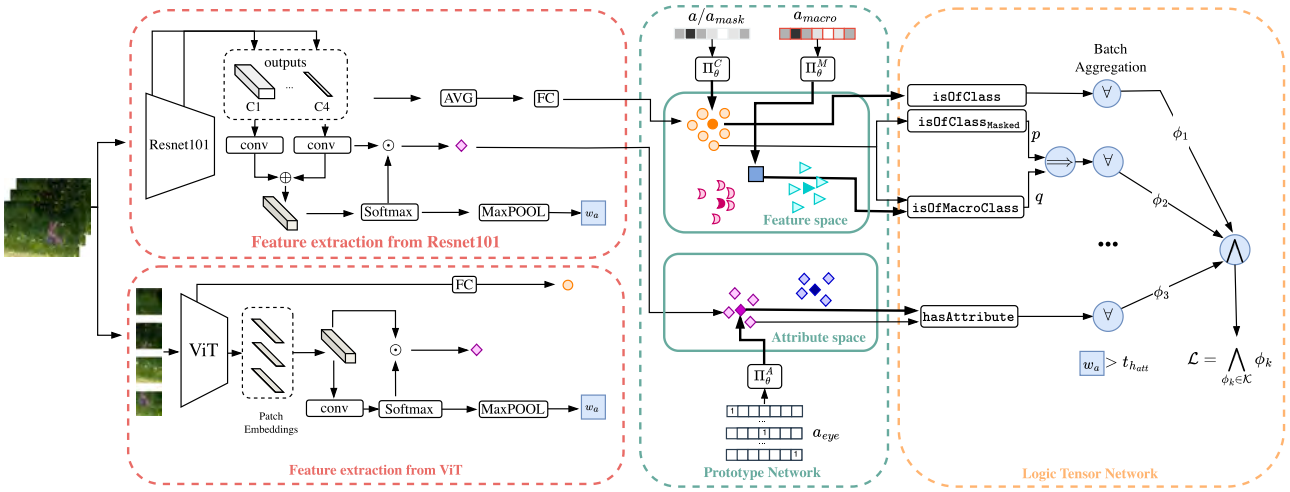


Fig. 1. The Fuzzy Logic Prototype Network (FLPN) architecture is composed of two main components: the feature extractor and the prototypical network. The feature extractor can be either a CNN (Resnet-101 in our experiments) or a Visual Transformer (ViT). The prototypical networks (Π_θ^C, Π_θ^M, Π_θ^A) integrate input images and the class (a), macroclass (a_{macro}), and attribute (a_{eye}) labels into a single embedding space, which in LTN terminology grounds the symbols in the embedding space. Building on this embedding space, various predicates, such as `isOfClass`, $\texttt{isOfClass}_{\text{masked}}$, and `isOfMacro` are defined as class membership functions. In addition, the `hasAttribute` predicate is used to detect specific attributes in images. These predicates constitute the foundational elements of the knowledge base, denoted $\mathcal{K}$, within the LTN framework. The training objective (loss function) is formulated to optimize the satisfiability or truth value of $\mathcal{K}$. The symbols $\oplus$ and $\odot$ denote the addition and multiplication operations, respectively.

an attention map, which is normalized using the Softmax function. Each feature map in the attention map acts as a soft mask to highlight probable attribute locations. Attribute-specific features are extracted through bilinear pooling between the attention map and the image feature tensor, involving pixel-wise multiplication and average pooling to localize attributes, as detailed in [23].

ViT-based module. An attention-based attribute localization method, analogously to the CNN-based architecture, is defined. Features extracted from the backbone are passed through a convolutional layer to generate the feature tensor. A soft mask for each attribute localization is created using a max-pooling function. Each attribute feature is then bi-linearly pooled with the image feature to obtain the attribute-level feature.

3.2.3. Attribute features weight

In GZSL and ZSL, attributes are often defined at the class level, and individual annotations for each image are not available. An attribute weight vector $W_a \in \mathbb{R}^A$ acts as a data-driven gating mechanism to identify attributes that are not visually present, despite being associated with the class, by applying a threshold th_{att}. Although the weight vector W_a may seem conceptually redundant, it substantially reduces the computational cost of computing $\mathcal{K}$. Further details on this efficiency gain will be elaborated later in the paper.

3.3. Prototypes for semantic mapping

Each entity, be it a class, a macroclass, or an attribute, is represented by a prototype embedded in a shared embedding space of size M. These prototypes serve as reference points to classify the input images by comparing them with their respective embeddings. Different parameterized functions, collectively denoted as Π_θ are employed to learn the prototypes for classes, macroclasses, and attributes. We employ here the term macroclass to indicate categorization at different semantic levels, as induced by a hypernym-hyponym relationship: in our datasets, we consider a "tiger" to be a class and "feline" to be the corresponding macroclass. Different groundings of the class-membership function may be empirically more effective depending on the type of categorization and the desired level of mutual exclusivity.

- **Class Prototypes**: Π_θ^C transforms the attribute matrix (denoted by a, of dimension $C \times A$, where C is the number of classes and A the number of attributes) into Class Prototypes in the shared space M.
- **Macroclass Prototypes**: similarly to class prototypes, Π_θ^M maps the attribute matrix a_{macro} (dimension $C_{\text{macro}} \times A$, where C_{macro} is the number of macroclasses) into Macroclass Prototypes within the shared space M. These prototypes merge the C classes into a smaller set of C_{macro} macroclasses based on attributes similarities. We employ a trainable matrix $a_{macro} \in \mathbb{R}^{C_{macro} \times A}$ to identify the attributes relevant to each macroclass, as they are not available in the ground truth.
- **Attribute Prototypes**: individual attributes are mapped into a shared space by a mapping function Π_θ^A which takes as input the eye matrix, which represents the one-hot encoded attribute labels.

The initial layers of the network modules that implement the three prototype functions (Π_θ^C, Π_θ^M, and Π_θ^A) share the same weights.

3.4. Logic tensor network

The LTN reformulates the learning objective by optimizing the satisfiability of a knowledge base ($\mathcal{K}$). For each training batch, $\mathcal{K}$ is calculated starting from axioms based on labeled training instances (ϕ_{class}) and prior domain knowledge ($\phi_{\text{macro}}, \phi_{\text{masked}}, \phi_{\text{attr}}, \phi_{\text{simil}}, \phi_{\text{dissimil}}$). A detailed overview of the $\mathcal{K}$ construction is provided in the remainder of this section.

The maximum satisfiability loss is then defined based on the aggregation of all axioms as follows:

$$\mathcal{L}^{\text{ep}} = 1 - \left(\bigwedge_{\phi \in \mathcal{K}} \phi \right) = 1 - \mathcal{G}(\phi) \tag{3}$$

3.4.1. Variables

The following variables are defined: $\mathcal{G}(x) = \mathbb{R}^{Ch \times H \times W}$; $\mathcal{G}(l) = \mathbb{N}^C$; $\mathcal{G}(l_m) = \mathbb{N}^{C_{macro}}$; $\mathcal{G}(l_a) = \mathbb{N}^{C \times A}$; $\mathcal{G}(w_a) \in \mathbb{R}^A$; $\mathcal{G}(a) = \mathcal{G}(a_{\text{mask}}) = \mathbb{R}^{C \times A}$; and $\mathcal{G}(a_{\text{macro}}) = \mathbb{R}^{C_{macro} \times A}$. Here, x denotes an RGB image with size $Ch \times H \times W$, l, l_m and l_a represent class, macroclass, and attribute labels, C, C_{macro} and A are the total number of classes, macroclasses, and attributes, and M is the dimensionality of the embedding space.

3.4.2. Predicates and functions

We define four key predicates. First, the $\texttt{isOfClass}(x, l)$ predicate evaluates the probability that an image x belongs to class l. This is achieved by computing the cosine similarity between the global features of the image, extracted by f_θ, and the class prototype computed by $\Pi_\theta^C(a_l)$, where a_l denotes the attribute vector for class l. The probability is computed using a Softmax function as:

$$p(x, l) = \frac{\exp\left(\delta \cos(f_\theta(x), \Pi_\theta^C(a_l))\right)}{\sum_{s=1}^{S} \exp\left(\delta \cos(f_\theta(x), \Pi_\theta^C(a_s))\right)} \tag{4}$$

where δ is a constant value that acts as a temperature parameter.

The prediction score for an instance x with respect to class c (that is, the degree of truthiness of the predicate $\texttt{isOfClass}$ for a specific input x and class c) is computed by taking the dot product of the one-hot encoded class label l_c^T (where T denotes transpose) with the output of the function p as:

$$\mathcal{G}(\texttt{isOfClass})(x, l_c) = l_c^T p(x, l_c) \tag{5}$$

We further define a masked version $\texttt{isOfClass}_{\text{masked}}$. Its grounding is the same as that of the $\texttt{isOfClass}$ predicate, except that a masked attribute vector a_c^{masked} is used. This predicate was specifically introduced, as in FLVN [9], to address potential discrepancies between image-level and class-level attributes (for example, a front-facing photo of a bird might not show its tail). The masked attribute vector $a_{mask} \in \mathbb{R}^{C \times A}$ has the same content of the attribute vector a_c, except that k elements are randomly masked (i.e., set to 0). Unlike FLVN [9], FLPN randomly drops only attributes with scores in a_c lower than the class mean, that is attributes which are less frequently observed within the class.

The $\texttt{isOfMacro}(x, l_m)$ predicate computes the likelihood that an image x belongs to the macroclass m. It is grounded as a function of the distance between the embedding of the image x and the prototype embedding representing the macroclass l_m:

$$\mathcal{G}(\texttt{isOfMacro}) : x, l_m = \mathrm{e}^{-\alpha\left(\mathrm{d}\left(f_\theta(x), \Pi_\theta^M(a_m)\right)\right)} \tag{6}$$

where x is an input image, l_m is the macroclass label, a_m is the corresponding attribute matrix, α is a hyperparameter and $\mathrm{d}(\cdot, \cdot)$ is a measure of distance. In our experiments, $\mathrm{d}(\cdot, \cdot)$ is the cosine distance (1 minus the cosine similarity), while PROTO-LTN [8] and DEM [15] used the Euclidean distance. $\mathcal{G}(\texttt{isOfMacro})$ takes the value of 1 when the distance from the macroclass prototype is zero.

The $\texttt{hasAttribute}$ predicate computes the probability that an image x is associated with the attribute label l_a. It is again computed as a function of the cosine distance between an embedding and an attribute prototype:

$$\mathcal{G}(\texttt{hasAttribute}) : x, l_a = l_a^T \cdot \mathrm{e}^{-\beta\left(\mathrm{d}\left(g_\theta(x), \Pi_\theta^A(a_{eye})\right)\right)} \tag{7}$$

Experimentally, we observed better results when using different distance measures for the predicates $\texttt{isOfClass}$, $\texttt{isOfMacro}$, and $\texttt{hasAttribute}$. We postulate that the effectiveness of the $\texttt{isOfClass}$ grounding (Eq. (5)) stems from promoting class exclusivity, while the grounding of the predicates $\texttt{isOfMacro}$ and $\texttt{hasAttribute}$ allows greater overlap of class attributes.

3.5. Knowledge base axioms

Learning from labeled examples. Axiom ϕ_{class} promotes predictions that are consistent with the data labels (in natural language, "this image is a zebra")[1]:

$$\phi_{\text{class}} = \forall \text{Diag}(x, l_c).\texttt{isOfClass}(x, l_c) \tag{8}$$

[1] Diagonal Quantification quantifies over pairs of instances, e.g., images and their labels. A more formal definition can be found in [2].

Prior knowledge about class hierarchies (e.g., "zebras belong to the class of ungulates") is encoded by axiom ϕ_{macro}:

$$\phi_{\text{macro}} = \forall \text{Diag}(x, l_c, l^c_{macro}).\ \texttt{isOfClass}(x, l_c) \implies \texttt{isOfMacro}(x, l^c_{macro}) \tag{9}$$

where l^c_{macro} is the macroclass associated with class c.

Learning with exceptions. The axiom ϕ_{masked} captures the notion that not all class attributes are present in every sample (e.g.,"there exists a zebra that is not agile" or "there is an image of a bird in which the tail is not visible"). The axiom is based on the $\texttt{isOfClass}_{\text{masked}}$ predicate:

$$\phi_{\text{masked}} = \forall l^{\text{seen}}(\exists x.\texttt{isOfClass}_{\text{masked}}(x, l^{\text{seen}})) \tag{10}$$

where l^{seen} denotes the list of seen classes. Let us remind that the $\texttt{isOfClass}_{\text{masked}}$ predicate randomly masks k attributes for each class from the attribute vector when computing the class prototypes. Specifically, we calculate the average of all non-zero entries per class in the attribute vector a_c and select k attributes with values below this average to be masked. Other strategies are possible, e.g., random selection regardless of the attribute matrix a, but we empirically found them to be less effective.

Learning with attention. The axiom ϕ_{attr} encourages images with similar attributes to reside close together within the embedding space, regardless of their associated class:

$$\phi_{\text{attr}} = \forall \text{Diag}(x, w_a, l^c_a) : w_a[l^c_a] > th_{atl}.\texttt{hasAttribute}(x, l^c_a) \tag{11}$$

where x represents an input image, l^c_a identifies the non-zero attributes linked to its class, and w_a stands for the attribute weight vector. For greater computational efficiency, we focus on attributes l^c_a that are both relevant to the class (based on the attribute matrix a) and likely present in the image x, by considering only attributes l with weights w^l_a exceeding a threshold th_{atl}. In practice, since the knowledge base $\mathcal{K}$ is updated every training batch, we leverage current attention values to enforce this criterion.

Learning better feature representations. Axioms ϕ_{simil} and ϕ_{dissimil} encourage image embeddings of similar images to be clustered together within the embedding space, while maintaining the separation between distinct classes. Specifically, to group similar image representations belonging to the same class, we introduce the axiom ϕ_{simil}:

$$\phi_{\text{simil}} = \forall l^{\text{seen}}, \forall x, \forall y : \cos(f_\theta(x), f_\theta(y)) > th_{sg}.\ \texttt{isOfClass}(x, l^{\text{seen}}) \Leftrightarrow \texttt{isOfClass}(y, l^{\text{seen}}) \tag{12}$$

where $f_\theta(x)$ and $f_\theta(y)$ represent the visual embeddings of images x and y. The axiom simply states that when two images x and y are similar, based on their respective cosine similarity $\cos(f_\theta(x), f_\theta(y))$, then they should be classified as belonging to the same class. To keep the size of $\mathcal{K}$ within manageable limits, we restrict the axiom to image pairs (x, y) with cosine similarity exceeding a threshold th_{sg}.

Likewise, axiom ϕ_{dissimil} increases the distance between images belonging to different classes, based on the cosine similarity between their respective features:

$$\phi_{\text{dissimil}} = \forall l^{\text{seen}}, \forall x, \forall y : \cos(f_\theta(x), f_\theta(y)) < th_{sl}.\ \neg\left(\texttt{isOfClass}(x, l^{\text{seen}}) \wedge \texttt{isOfClass}(y, l^{\text{seen}})\right) \tag{13}$$

Again, $\mathcal{K}$ includes only pairs of images with cosine similarity lower than a threshold th_{sl}.

3.5.1. Grounding logical connectives and aggregators

We experimented with two different operator semantics, denoted in the following as LTN [2] and logLTN [6], obtaining two versions of our architecture which we denote for brevity FLPN and Log-FLPN. LTN operates in the probability space and logLTN in the log probability space [6]. Table 1 outlines the main differences between the logical connectives and aggregation operators used in logLTN versus the standard LTN formulation. Further details on the roles of $\exists$ and $\forall$ can be found in [2,6].

Table 1
Grounding of logical connectives and aggregators ($p \geq 1$). In the existential quantifier, $C = \max(\alpha \log(\sigma(x)_i))$. σ is the activation function.

Connective	LTN	logLTN
$\neg$	$1 - \sigma(a)$	$\log(1 - e^{\log(\sigma(a))})$
$\wedge$	$\sigma(a) * \sigma(b)$	$\log(\sigma(a)) + \log(\sigma(b))$
$\vee$	$\sigma(a) + \sigma(b) - \sigma(a) * \sigma(b)$	$\frac{1}{\alpha}C + \log\left(\frac{1}{n}\sum_{i=1}^{n} e^{\log(\alpha\sigma(a_i))-C}\right)$
$\rightarrow$	$1 - \sigma(a) + \sigma(a) * \sigma(b)$	$a \rightarrow b \Leftrightarrow \neg a \vee b$
$\forall$	$1 - \left(\frac{1}{n}\sum_{i=1}^{n}\left(1 - \sigma(a)_i\right)^p\right)^{\frac{1}{p}}$	$\frac{1}{n}\sum_{i=1}^{n}\log(\sigma(a)_i)$
$\exists$	$\left(\frac{1}{n}\sum_{i=1}^{n}\sigma(a)_i^p\right)^{\frac{1}{p}}$	$\frac{1}{\alpha}C + \log\left(\frac{1}{n}\sum_{i=1}^{n} e^{\log(\alpha\sigma(a_i))-C}\right)$
$\Leftrightarrow$	$(1 - \sigma(a) + \sigma(a) * \sigma(b)) * (1 - \sigma(b) + \sigma(a) * \sigma(b))$	$(\neg a \vee b) \wedge (\neg b \vee a)$

Table 2
The details of five datasets. The dimensionality of semantic attributes is denoted by $\mathcal{A}$, the number of seen and unseen categories is indicated by $\mathcal{N}_s$ and $\mathcal{N}_u$, and the number of training samples, testing seen samples, and testing unseen samples is indicated by $\mathcal{N}^{tr}$, $\mathcal{N}_s^t$, and $\mathcal{N}_u^t$.

Dataset	$\mathcal{A}$	$\mathcal{N}_s$	$\mathcal{N}_u$	$\mathcal{N}^{tr}$	$\mathcal{N}_s^t$	$\mathcal{N}_u^t$
FLO	1024	82	20	5631	1403	1115
CUB	1024	150	50	7057	1764	2967
AwA2	85	40	10	23,527	5882	7913
SUN	102	645	72	10,320	2580	1440

3.5.2. Querying the knowledge base

During inference, the class with the maximum score is chosen as the predicted class. The Calibrated Stacking method, with parameter δ was used to reduce the bias towards classes seen at training time by lowering their scores [47,48].

3.6. Relationship to contrastive learning

After grounding the axioms, interesting connections can be drawn between their mathematical formulation and loss functions commonly used in deep learning. In the following, unless otherwise noted, we will use the logLTN grounding for the logical connectives.

Let us consider axiom ϕ_{attr} defined in Eq. (11). After grounding the universal quantifier as defined in Table 1 and substituting the grounding of the `hasAttribute` predicate defined in Eq. (7), the satisfiability of the axiom is computed as follows:

$$\phi_{\text{attr}} = \sum_{i=1}^{N_b}\sum_{j=1}^{\tilde{K}_i}\left(\log(e^{-\beta \mathrm{d}(g_\theta(x), \Pi_\theta^A(a_{eye}))})\right) \tag{14}$$

where N_b is the number of examples in the current batch, $\tilde{K}_i$ the list of attributes that are relevant for the specific image, and $d(\cdot)$ is a distance function. This loss is, up to a scaling constant, equivalent to existing losses for learning a deep embedding model for zero-shot learning [8,15].

Following the same approach, for the ϕ_{class} axiom defined in Eq. (8), we obtain the following grounded formulation:

$$\phi_{\text{class}} = \sum_{i=1}^{N_b}\log\frac{\exp\left(\delta\cos(f_\theta(x), \Pi_\theta^C(a_l))\right)}{\sum_{s=1}^{S}\exp\left(\delta\cos(f_\theta(x), \Pi_\theta^C(a_s))\right)} \tag{15}$$

Maximizing Eq. (15) corresponds thus to minimizing the loss defined in the original prototype network [52]. Moreover, as suggested in [53], Eq. (15) can be seen as a variant of the InfoNCE loss used by many popular contrastive learning methods [54]:

$$\mathcal{L}_{\text{InfoNCE}} = \sum_{i=1}^{N_b} -\log\frac{\exp(\mathbf{z}_i \cdot \mathbf{z}_i'/\tau)}{\sum_{j=1}^{R}\exp(\mathbf{z}_i \cdot \mathbf{z}_j'/\tau)} \tag{16}$$

where N_b is the number of samples in a batch, containing one positive embedding and R negative embeddings. As suggested in [53], the two formulations coincide if δ replaces the temperature parameter τ, the augmented positive instance $\mathbf{z}_i'$ is replaced by the class prototype $\Pi_\theta^C(a_l)$ and the augmented negative instances $\mathbf{z}_j'$ are replaced by the other class prototypes $\Pi_\theta^C(a_s)$. The view that ϕ_{class} introduces a contrastive element to the training procedure is reinforced by the way in which the training batch is sampled (detailed in Section 4), ensuring a certain ratio of positive to negative examples, which we experimentally found to be crucial in experiments with smaller batch sizes (i.e., when using a ViT backbone). This observation suggests a potential connection with the performance of our NeSy approach and the batch size, which we leave for future work.

Similarly, ϕ_{simil} and ϕ_{dissimil} introduce a form of contrastive learning that encourages similar images to cluster around their corresponding prototype. Let us consider for instance axiom ϕ_{dissimil} defined in Eq. (13), and define $P = \{(x_i, y_i) \mid i = 1, 2, \ldots, K\}$ as the list of dissimilar image pairs selected by the guarded universal quantifier. After grounding, ϕ_{dissimil} can be expressed as:

$$\begin{aligned}\phi_{\text{dissimil}} = \sum_{l=i}^{L}\sum_{i=1}^{P}\log\left(1 - \frac{\exp\left(\delta\cos(f_\theta(x_i), \Pi_\theta^C(a_l))\right)}{\sum_{s=1}^{S}\exp\left(\delta\cos(f_\theta(x_i), \Pi_\theta^C(a_s))\right)}\right) \\ \cdot \log\frac{\exp\left(\delta\cos(f_\theta(y_i), \Pi_\theta^C(a_l))\right)}{\sum_{s=1}^{S}\exp\left(\delta\cos(f_\theta(y_i), \Pi_\theta^C(a_s))\right)}\end{aligned} \tag{17}$$

which encourages a prototype space in which, for each class label l, the corresponding prototype $\Pi_\theta^C(a_l)$ is either pushed towards x_i or y_i, but not both.

It must be stressed that the exact mathematical formulation also depends on the chosen grounding (or semantics) of the logical connectives. When choosing the alternative LTN formulation in Table 1, the universal aggregator becomes the p-mean error $1 - \left(\frac{1}{n}\sum_{i=1}^{n}\left(1 - \sigma(a)_i\right)^{p_\forall}\right)^{\frac{1}{p_\forall}}$, with $p_\forall = 2$ in our experiments, which replaces the outer sum in Eqs. (14), (15) and (17). Mathematically, with $p_\forall > 1$ the loss can no longer be decomposed into the sum of independent components, one for each training instance. This adjustable hyper-parameter places more or less emphasis on outliers (see [2,6] for additional discussions on the role of this hyper-parameter, and Section 4 for the exact scheduling employed in our experiments). Furthermore, the satisfiability of the axiom and the training convergence depend on the batch size and sampling strategy.

4. Experiments

In this section, we present the datasets used in our experiments, the knowledge base adopted for each dataset, and the list of hyperparameters.

Dataset. The experiments were performed on AWA2 [10], CUB [55], SUN [56] and FLO [57] benchmarks. The statistics of each dataset, including the number of seen and unseen categories, training samples, and test samples, are reported in Table 2. The GZSL evaluation metrics followed the standards set by previous research [10].

Knowledge base. The composition of the knowledge base $\mathcal{K}$ is elaborated in Section 3.5. For constructing the class hierarchy, we associate classes from the AWA2 and CUB dataset with their respective synsets in WordNet, following the approach used in [7,9].

Following the approach outlined in previous works [7,9], we constructed a semantic hierarchy by grouping the classes from the AWA2

Table 3
We compare our approach to the current state of the art on the AWA2, CUB, and SUN datasets for both GZSL and ZSL settings. We separately analyze CNN-based and transformer-based backbones to provide a head-to-head evaluation of different methods. The results are categorized into four groups: embedding-based, end-to-end embedding-based (separately for CNN and ViT backbones), and generative. We highlight the **best** and second-best method for each metric, separately for CNN and ViT-based architectures. For FLVN and FLPN, we report the mean, standard deviation and best results (in parenthesis) over three repetitions.

Model	AWA2				CUB				SUN				FLO			
	T1	U	S	H	T1	U	S	H	T1	U	S	H	T1	U	S	H
Embedding-based methods																
Proto-LTN [8][a]	67.6	32.0	83.7	46.2	48.8	20.8	54.3	30.0	60.4	20.4	36.8	26.2	–	–	–	–
DEM [15]	67.1	30.5	86.4	45.1	51.7	19.6	57.9	29.2	61.9	20.5	34.3	25.6	–	–	–	–
VSE [16]	**84.4**	45.6	**88.7**	60.2	71.9	39.5	68.9	50.2	–	–	–	–	–	–	–	–
TCN [19]	71.2	61.2	65.8	63.4	59.5	52.6	52.0	52.3	61.5	31.2	37.3	34.0	–	–	–	–
End-to-end Embedding-based methods (CNN backbone)																
AMGML [20]	71.7	56.0	74.6	64.0	70.0	58.2	55.7	56.9	59.7	42.0	35.1	38.3	–	–	–	–
CC-ZSL [13]	68.8	62.2	83.1	71.1	74.3	66.1	73.2	69.5	62.4	44.4	36.9	40.3	–	–	–	–
CoAR-ZSL [23]	74.1	68.1	79.1	73.2	79.2	70.9	77.3	74.0	66.7	50.6	38.0	43.4	–	–	–	–
FLVN [9] [a]	69.8	65.8	82.3	73.1	71.2	62.6	83.1	71.5	61.7	48.4	32.7	39.0	–	–	–	–
	±0.8	±0.9	±0.3	±0.6	±0.2	±0.5	±0.3	±0.2	±0.18	±0.57	±0.2	±0.1	–	–	–	–
	(71.0)	(67.1)	(82.8)	(74.1)	(71.4)	(63.2)	(**83.4**)	(71.7)	(61.9)	(48.9)	(32.9)	(39.1)	–	–	–	–
PRZSL [25]	73.6	65.8	77.8	71.3	77.1	68.8	63.7	66.2	64.2	53.6	37.7	44.3	–	–	–	–
AS-ZSL [14]	68.9	66.5	78.38	71.9	78.5	65.8	78.2	71.5	62.2	39.7	37.2	38.3	–	–	–	–
HRT [41]	67.3	**78.7**	58.9	67.4	71.7	63.5	62.1	62.8	63.9	26.9	**53.2**	35.7	–	–	–	–
CSDR [34]	76.7	68.0	82.7	74.6	80.3	**73.1**	78.9	**75.9**	64.5	43.9	44.6	44.2	90.1	82.4	**94.5**	88.0
CTPM [42]	74.1	67.2	79.6	72.9	73.4	64.7	73.2	68.7	66.0	44.0	43.2	43.6	–	–	–	–
MSDN [49]	70.1	62.0	74.5	67.7	76.1	68.7	67.5	68.1	65.8	52.2	34.2	41.3	–	–	–	–
GNDAN [22]	71.0	60.2	80.0	69.0	75.1	69.2	69.6	69.4	65.3	50.0	34.7	41.0	–	–	–	–
DCA-VAE [26]	73.1	58.7	76.9	66.6	64.1	52.9	56.0	54.4	64.0	49.2	36.7	42.0	–	–	–	–
ViFR [27]	77.8	68.2	78.9	72.2	74.5	63.9	72.0	67.7	69.2	51.3	40.0	44.7	–	–	–	–
FLPN[a]	71.43	67.80	85.93	74.53	72.73	71.00	78.50	74.53	66.73	53.53	38.50	44.83	61.30	58.30	89.80	70.70
	±3.44	±2.99	±2.28	±0.91	±0.48	±0.65	±2.76	±1.21	±0.50	±0.56	±0.36	±0.17	±1.60	±2.62	±1.80	±1.30
	(74.87)	(70.79)	(88.21)	(**76.61**)	(75.01)	(71.65)	(81.26)	(75.75)	(67.23)	(54.09)	(38.86)	(**45.00**)	62.90	60.90	91.60	72.06
Log-FLPN[a]	66.53	64.17	82.00	71.90	73.87	71.93	73.67	72.73	61.60	52.77	37.93	44.10	64.00	58.80	89.90	70.73
	±4.97	±5.15	±1.50	±3.18	±0.42	±0.54	±1.64	±0.62	±7.64	±2.18	±0.66	±0.33	±0.79	±1.24	±1.17	±0.36
	(71.50)	(69.32)	(83.50)	(75.08)	(74.29)	(72.48)	(75.31)	(73.35)	(**69.24**)	(**54.94**)	(38.59)	(44.43)	64.75	60.08	93.91	72.53
End-to-end embedding-based methods (ViT backbone)																
CoAR-ZSL [23]	78.7	76.5	88.2	82.0	**79.9**	72.5	76.3	74.4	**79.4**	**68.3**	55.0	**61.0**	–	–	–	–
ZeroMamba [43]	80.0	72.1	76.4	74.2	71.9	67.9	**87.6**	**76.5**	72.4	56.5	41.4	47.7	–	–	–	–
ZSLViT [44]	**90.1**	52.4	64.3	57.6	56.9	64.6	52.8	43.8	57.4	43.2	53.2	47.3	–	–	–	–
TransZero++[50]	78.3	67.5	73.6	70.4	72.6	64.6	82.7	72.5	67.6	48.6	37.8	42.5	–	–	–	–
FLPN[a]	80.97	80.73	83.17	81.87	76.43	70.10	70.30	70.00	74.77	58.60	53.90	56.13	59.10	57.46	85.89	68.80
	±1.97	±1.82	±3.24	±2.19	±1.16	±3.59	±5.03	±2.14	±0.56	±1.79	±1.21	±0.88	±3.20	±2.90	±2.90	±2.72
	(82.94)	(**82.55**)	(86.41)	(84.06)	(77.59)	(**73.69**)	(75.33)	(72.14)	(75.32)	(60.39)	(**55.11**)	(57.01)	62.30	60.36	88.80	71.52
Log-FLPN[a]	78.23	77.10	86.23	81.40	75.20	69.63	63.10	65.93	73.37	58.03	51.43	54.53	61.23	56.96	87.36	68.60
	±2.60	±2.12	±5.11	±3.34	±2.86	±3.65	±6.64	±3.64	±0.45	±1.11	±0.46	±0.69	±0.46	±4.09	±6.00	±1.41
	(80.83)	(79.22)	(**91.34**)	(**84.74**)	(78.06)	(73.29)	(69.74)	(69.57)	(73.82)	(59.15)	(51.90)	(55.23)	61.69	61.06	93.37	70.01
Generative methods																
Cycle-CLSWGAN [32]	–	–	–	–	58.4	45.7	61.0	52.3	60.0	49.4	33.6	40.0	70.1	59.2	72.5	65.1
TGMZ [30]	78.4	64.1	77.3	70.1	66.1	60.3	56.8	58.5	–	–	–	–	–	–	–	–
CEGZSL [33]	70.4	63.1	78.6	70.0	77.5	63.9	66.8	65.3	63.3	48.8	38.6	43.1	70.6	69.0	78.7	73.5
DFCA-GZSL [51]	74.7	66.5	81.5	73.3	**80.0**	70.9	63.1	66.8	62.6	48.9	38.8	43.3	**91.0**	**88.0**	90.3	**89.2**
LisGan-CPM [35]	77.6	60.4	73.7	66.4	61.9	45.9	55.2	50.1	65.8	45.8	37.8	41.4	–	–	–	–

[a] Models are explicitly based on a NeSy approach.

and CUB datasets into 9 and 49 macroclasses, respectively. Specifically, the macroclasses for the AWA2 and CUB datasets were formed by finding the root synset whose subtree includes the dataset classes, then assigning its direct descendants (via the WordNet hierarchy) as classes. It was not possible to define macroclasses for the SUN dataset, and hence the related axiom was excluded from the knowledge base.

For all experiments, within the predicate $\texttt{isofClass}_{masked}$, 15 attributes ($K = 15$) were omitted at random. A consistent threshold value of $th_{att} = 9.0$ was applied in all experiments, with the value of δ uniformly set to 50 for every dataset. The thresholds th_{sg} and th_{sl} were set at 0.5 and 0.3 for AWA2 and to 0.7 and 0.6 for CUB, SUN and FLO, respectively. To manage outliers, the initial settings for the parameters of the aggregation function in the standard implementation of LTN were set at $p_{\exists} = 2$ and $p_{\forall} = 2$ (see LTN in Table 5). These parameters were increased by 2 after each 4 epoch for all datasets, adhering to the protocol recommended in [2].

Hyperparameter selection. The embedding functions f_θ and g_θ convert an image into a vector in a 2048-dimensional space for ResNet and a 1024-dimensional space for ViT. We trained the model end-to-end, using the SGD optimizer with an initial learning rate of $1e{-}3$. During the fine-tuning phase, the learning rate was subsequently decreased by a factor of $1e{-}7\iota$, where $\iota = 0.8^{epoch//10}$, for a total of 300 epochs for both the ResNet-101 and ViT models. When training the ResNet-101 backbone, samples were chosen at random from a batch of 64 samples, while for ViT, training batches consisted of a ratio of 4 to 8 for positive and negative examples. We set the scaling factor γ to 0.7 at inference time to calibrate the scores for the seen classes. As data augmentation, we used a random flip with 0.5 chance and a random crop in each experiment. Using the LTNtorch library [58], we developed the architectures in PyTorch 2.1.1 and trained it on a single Nvidia 3090 Ti GPU. Each experiment was repeated three times to report the mean and standard deviation.

Table 4
Ablation study: effect of different axioms in the $\mathcal{K}$ on the performance, for both CNN (top rows) and ViT (bottom rows) backbones, on Awa2 and CUB.

Architecture	Knowledge base						AWA2				CUB			
	ϕ_{class}	ϕ_{macro}	ϕ_{masked}	ϕ_{attr}	ϕ_{simil}	ϕ_{dissimil}	T1	U	S	H	T1	U	S	H
Resnet101 + Π_θ^C	✓	–	–	–	–	–	64.3	60.2	82.5	68.8	72.9	67.9	72.2	70.0
Resnet101 + Π_θ^C	✓	–	✓	–	–	–	66.3	62.2	84.7	71.7	74.4	70.0	81.0	75.1
Resnet101 + $\Pi_\theta^C + \Pi_\theta^A$	✓	–	✓	✓	–	–	69.8	66.4	80.2	72.7	74.3	70.1	80.3	74.9
Resnet101 + $\Pi_\theta^C + \Pi_\theta^A + \Pi_\theta^M$	✓	✓	✓	✓	–	–	70.9	67.7	79.3	73.1	75.2	71.6	80.6	75.8
Resnet101 + $\Pi_\theta^C + \Pi_\theta^A + \Pi_\theta^M$	✓	✓	✓	✓	✓	✓	72.9	71.0	78.4	74.5	75.2	71.7	78.9	75.1
ViT + Π_θ^C	✓	–	–	–	–	–	72.6	72.3	86.2	78.6	72.7	65.3	76.2	70.3
ViT + Π_θ^C	✓	–	✓	–	–	–	79.8	78.9	89.6	83.9	73.3	66.5	78.3	71.9
ViT + $\Pi_\theta^C + \Pi_\theta^A$	✓	–	✓	✓	–	–	80.9	80.3	89.2	84.5	76.2	72.4	73.5	72.9
ViT + $\Pi_\theta^C + \Pi_\theta^A + \Pi_\theta^M$	✓	✓	✓	✓	–	–	83.1	82.6	85.2	83.9	77.5	71.4	74.3	72.8
ViT + $\Pi_\theta^C + \Pi_\theta^A + \Pi_\theta^M$	✓	✓	✓	✓	✓	✓	83.0	82.4	87.6	84.9	77.6	73.7	75.3	72.1

5. Results

In Table 3, we experimentally compare the proposed method against previously published techniques on the AWA2, CUB and SUN datasets. We compare against *embedding-based* methods such as PROTO-LTN [8], DEM [15] and VSE [16], DCA-VAE [26], ViFR [27], recent embedding-based methods trained *end-to-end* such as AMGML [20], CC-ZSL [13], CoAR-ZSL [23], FLVN [9], PRZSL [25], AS-ZSL [14], HRT [41], CSDR [34], and CTPM [42], and *generative* methods such as Cycle-CLSWGAN [32], TGMZ [30], CEGZSL [33], DFCA-GZSL [51] and LisGan-CPM [35]. We further differentiate between methods that employ CNN and transformer backbones. The results of state-of-the-art methods are derived from the respective publications. For FLVN and FLPN, we report the mean, standard deviation, and best results over three repetitions.

FLPN obtains better performance compared to our previous NeSy frameworks PROTO-LTN [52] and FLVN [9], reaching state-of-the-art results on the AWA2 and SUN datasets. Using the standard LTN grounding, FLPN outperforms FLVN [9] with an absolute improvement of +2.5% and +5.9% on AWA2 (H = 76.61 vs. 74.1) and SUN (H = 45.00 vs. 39.1), and a +3.7% and +5.96% on unseen classes (AWA2: U = 70.79 vs. 67.1; SUN: U = 38.86 vs. 32.9). The greatest difference between FLPN and FLVN is observed in SUN, which consists of more complex scene images, whereas CUB and AWA2 are object-centric approaches. A more detailed comparison of the three architectures is provided in Section 5.4. Log-FLPN, which is trained in logarithmic space, achieves a similar performance boost over FLPN (AWA2: H = 75.08, U = 69.32; SUN: H = 44.43, U = 54.94). However, when comparing FLPN and Log-FLPN, we do not observe a consistent performance benefit associated with operating in the logarithmic space. Lastly, switching from a CNN- to a ViT-based backbone is associated with a substantial absolute performance benefit of roughly +12% on both AWA2 and SUN, but not on CUB (−0.36%). These trends are consistent with those observed in [23], and may be partly attributed to the smaller size of the dataset compared to AWA2.

Among architectures based on CNN backbones FLPN achieves performance that is better or similar to existing state-of-the-art embedding-based models. The closest methods, in terms of performance, include CC-ZSL [13], CoAR-ZSL [23], PRZSL [25], CSDR [34], AS-ZSL [14], DCA-VAE [26] and ViFR [27]. Generally speaking, FLPN achieves the best performance on AWA2 and SUN, but lags behind methods such as CoAR-ZSL [23] and CSDR [34] on CUB.

Overall, FLPN shows greater performance benefits in combination with CNN than ViT, but still remains competitive among transformer-based architectures. For the transformer-based category, ZeroMamba [43], TransZero++ [50] and ZSLViT [44] achieve strong ZSL accuracy on AWA2 (T1 = 80.0 and 90.1, respectively), yet exhibit a more pronounced drop in H for GZSL (74.2 and 57.6, respectively), confirming that FLPN provides a more balanced seen/unseen classification.

When operating in a GZSL setting, it is crucial to balance performance on seen vs. unseen classes. On AWA2, FLPN (U = 70.79) outperforms most other methods on unseen classes, including CC-ZSL (U = 62.2), PRZSL (U = 65.8), AS-ZSL (U = 66.5) and CoAR-ZSL (U = 68.1). While methods such as CoAR-ZSL (U = 68.1) and HRT (U = 78.7) achieve comparable or better results on unseen classes, their performance suffers on seen classes compared to FLPN (S = 88.21 vs. 79.1 vs. 58.9). In a GZSL setting, relying on a fixed alignments (as in PRZSL) and subspace embeddings (as in AS-ZSL) limit their ability to balance seen and unseen class representations, where FLPN's adaptive prototypes and fine-grained loss provide a distinct advantage.

Although similar general trends are observed on SUN, on the CUB dataset CSDR [34] and CoAR-ZSL [23] achieve the best performance in both ZSL (CSDR: T1 = 80.2; CoAR-ZSL: T1 = 79.2; FLPN: T1 = 75.01) and GZSL settings (CSDR: H = 75.9; CoAR-ZSL: H = 74.0; FLPN: H = 75.75), despite FLPN's strong performance on unseen classes (CSDR: U = 73.1; CoAR-ZSL: U = 70.9; FLPN: U = 71.65). This discrepancy could be due to the different methods each model uses to extract image attributes. CoAR-ZSL and CDSR employ a contrastive loss, whereas FLPN focuses on classifying attributes via the `hasAttribute` predicate. This emphasis on classification might limit FLPN's ability to accurately capture image attributes. CoAR-ZSL's contrastive loss enhances feature separation and attribute discrimination in the embedding space, beneficial for fine-grained datasets like CUB. It uses a dual attention pathway to better align global and local features with semantic attributes, surpassing FLPN's single-path attention.

Compared to generative-based methods, FLPN and Log-FLPN outperform Cycle-CLSWGAN [32], LisGan [31], and LisGan-CPM [35] for unseen categories. TGMZ [30] performs well in AWA2 (T1 = 78.4) in the ZSL setting, but drops in the CUB and SUN datasets and the GZSL setting in general. CEGZSL [33] and DFCA-GZSL [51] are competitive but less robust than FLPN.

Generative approaches can be effective in bridging the gap between seen and unseen classes, as they artificially enrich the dataset. However, generating class-attribute distributions and synthetic data is computationally costly and resource-intensive, whereas FLPN through Nesy integration effectively embeds prior knowledge in the feature representation with minimal overhead.

Specifically, on the FLO dataset, FLPN (H = 70.70) and Log-FLPN (H = 70.73) outperform Cycle-CLSWGAN (H = 65.1) and come close to CEGZSL (H = 73.5). These results remain below the best-performing DFCA-GZSL (H = 89.2), which leverages a generative paradigm to synthesize additional training samples, an approach that can be particularly advantageous for specialized, fine-grained datasets like FLO. However, FLPN and Log-FLPN strike a favorable balance on FLO between computational efficiency and performance, again underscoring the value of neurosymbolic integration in GZSL. Notably, from Table 2, FLO has significantly fewer images than other benchmarks, making it inherently challenging. Methods such as DFCA-GZSL [51] (which must learn to generate highly similar features across different flower classes) or CSDR [34] (which focuses on careful separation of fine-grained cues) can be advantageous in producing discriminative features for visually similar flower classes. In contrast, FLPN, which does not have a generative component, may struggle to separate visually similar categories if the class-level attributes are not sufficiently discriminative.

Table 5
Comparison of FLPN, FLVN and Proto-LTN NeSy architectures based on Res101 backbone.

Model	Knowledge base	Embedding space	End-to-end training	`isOfClass` grounding	Batch sampling	Aggregator
Proto-LTN	$\{\phi_{\text{class}}\}$	M	–	Euclidean	Random	LogProduct
FLVN	$\{\phi_{\text{class}}, \phi_{\text{macro}}, \phi_{\text{masked}}, \phi_{\text{simil}}, \phi_{\text{dissimil}}\}$	A	✓	Softmax	Shots and Ways	Mean-Error
FLPN (ours)	$\{\phi_{\text{class}}, \phi_{\text{macro}}, \phi_{\text{masked}}, \phi_{\text{attr}}, \phi_{\text{simil}}, \phi_{\text{dissimil}}\}$	M	✓	Softmax	Random	Mean-Error

In terms of computational performance, FLPN has an inference time of 0.55 s/im, which is higher than FLVN (0.22 s/im) and Proto-LTN (0.09 s/im), yet lower than those obtained by CC-ZSL (0.85 s/im) and CoAR-ZSL (0.98 s/im). These results demonstrate how FLPN manages to embed a rich set of knowledge – made possible by the additional axioms – in a compact embedding space, achieving competitive performance both in terms of accuracy and inference time.

5.1. Ablation experiments

The ablation experiments presented in Table 4 were performed to elucidate the effect of individual axioms on FLVN performance. Starting from a baseline $\mathcal{K}$ that included only axioms ϕ_{class}, we progressively expand it until the whole $\mathcal{K}$ is obtained. In general, all axioms in combination contribute to a boost in performance in unseen classes, with performance increasing from 60.2 to 71.0 with the CNN backbone (+6.7) and from 72.3 to 82.4 with the ViT backbone (+10.1). Each individual axiom includes different predicates, which are grounded by separate components of the architecture, also depending on the backbone used to extract features. Thus, the knowledge base and its predicates are intrinsically linked to specific sections of the FLPN architecture. As a result, ablating an axiom from the knowledge base is equivalent to ablating the parts of the architecture that ground all the predicates that are no longer included in knowledge base, and as a consequence in the loss. In Table 4, the model column specifies which parts of the network are included in the training of each ablated model.

Role of ϕ_{class}. The base architecture includes only axioms designed to learn from labeled examples. Each class has inherent characteristics that are always present: for instance, a zebra will always have stripes, even in a cropped image. As a result, the network prioritizes seen classes.

Role of ϕ_{masked}. This axiom accounts for exceptions by randomly masking attributes (ϕ_{masked}). It improves on both seen and unseen classes (S = 84.7 for AWA2 and S = 81.0 for CUB in Table 4). The introduction of axiom ϕ_{masked} helps the model to handle exceptions due to the use of image-level attribute labels during training. In preliminary experiments, we found that ϕ_{masked} is particularly effective when the k elements are chosen among attributes with values lower than the class average.

Role of ϕ_{attr}. Using the attribute matrix and the attribute weight w_a for a class, we introduce the attribute classification with the axiom ϕ_{attr}. The predicate `hasAttribute` in this axiom is based on the cosine distance between an attribute-level embedding and an attribute prototype. Despite a nearly 1-point loss in the GZSL setting, Table 4 shows improvement in unseen class classification in the ZSL (T1 = 66.3 to 69.8 for AWA2) and GZSL (U = 62.2 to 66.4 for AWA2 and U = 70.1 to 70.1 for CUB) settings, respectively.

Role of ϕ_{macro}. The inclusion of axiom ϕ_{macro} slightly improves the harmonic mean (H) from 72.7 to 73.1 for AWA2 and 74.9 to 75.8 for CUB, indicating that the addition of semantically relevant constraints on macroclasses aggregation provides a minor boost to the architecture's performance.Proto-LTN [8] has previously shown that pre-trained embeddings, even without the imposition of specific constraints on the class hierarchy, naturally tend to cluster related classes in feature space.

While previous observations were based on t-Distributed Stochastic Neighbor (t-SNE) visualization, the present ablation study paints a slightly more nuanced picture: explicitly introducing constraints on macroclasses negatively affects performance on seen classes S (−0.9% with CNN backbone, −4% with ViT backbone on the AWA2 dataset) but promotes generalization to unseen classes U (+1.3% with CNN backbone, +2.3% with ViT backbone also on AWA2). On CUB we observe a slight increase for seen classes with both CNN (+0.3%) and ViT (+0.8%), while generalization on unseen classes improves with CNN (+1.5%) but slightly decreases with ViT (−1.0%). This phenomenon can be attributed to the aggregation of related classes in a more compact feature space, which benefits unseen samples.

Role of ϕ_{simil} and ϕ_{dissimil}. These two axioms introduce constraints that force similar images (i.e., images that are close in the embedding space) to be classified as belonging to the same class, while dissimilar images (i.e., those farther away in the feature space) must be classified into different classes. Both axioms act essentially as a contrastive component to the loss. From Table 4, we observe that these axioms in combination further boost the harmonic mean (H) on AWA2 (+1.4% with the CNN backbone, +1% with the ViT backbone), with a performance increase attributed mostly to unseen classes for the CNN backbone (+3.3%) and to seen classes for the ViT backbone (+2.4%). On the CUB dataset, these axioms achieve a more limited performance gain. We hypothesize that this may be due to the fine-grained nature of the CUB classification task.

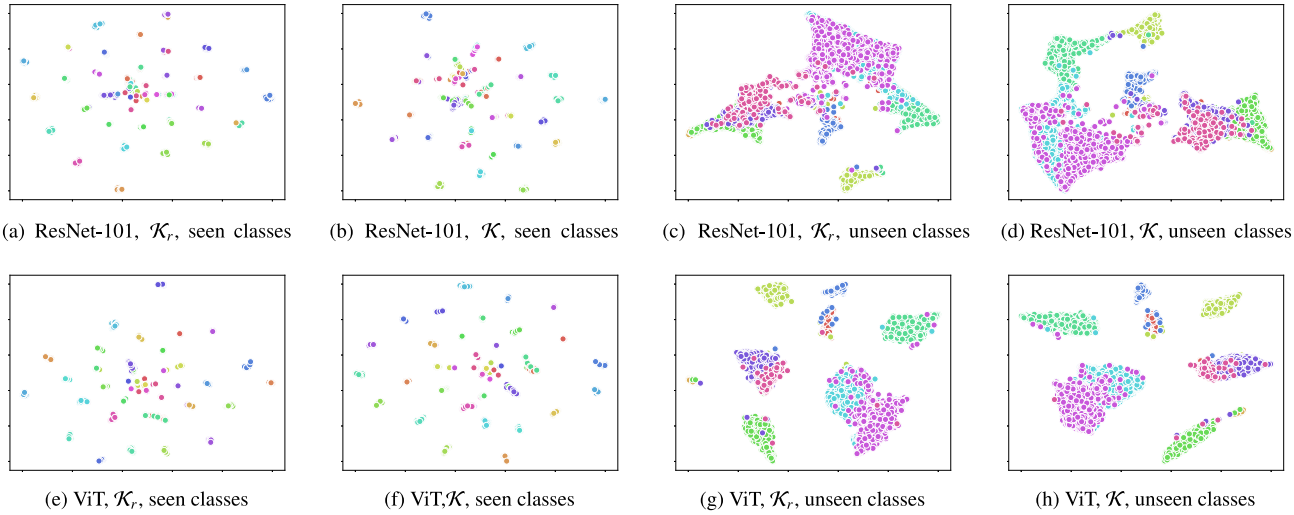
5.2. Class feature visualization using T-SNE

We examined whether the extracted attributes accurately represent the characteristics of the dataset using t-SNE visualization. We visualized class separation on the AWA2 and CUB datasets using attribute scores, as shown in Fig. 2. Focusing on the GZSL scenario, we compared the embeddings for two knowledge bases: one with a minimal set of axioms (ϕ_{class} and ϕ_{masked}), and the final one with all proposed axioms. We analyzed the embedding space for both seen and unseen classes separately. For clarity, we included only 500 samples per class, each with a distinct color. Although Figs. 2(b) and 2(f) show a slight improvement in the distribution of unseen classes compared to Figs. 2(c) and 2(d) (supporting the results in Table 3), both the seen and unseen classes are clearly separated in the feature space.

Fig. 3 presents the t-SNE visualizations of the attribute-level embeddings for each image. We analyze both the AWA2 (Figs. 3(a), 3(b)) and the CUB datasets (Figs. 3(c), 3(d)), focusing on attributes of seen and unseen classes. The attention mechanism helps the model prioritize relevant attributes for distinguishing classes. Key attributes, especially in unseen class images, are distinctly grouped in the feature space. This result suggests that the extracted information effectively captures valuable characteristics for accurate class discrimination.

5.3. Attribute attention maps

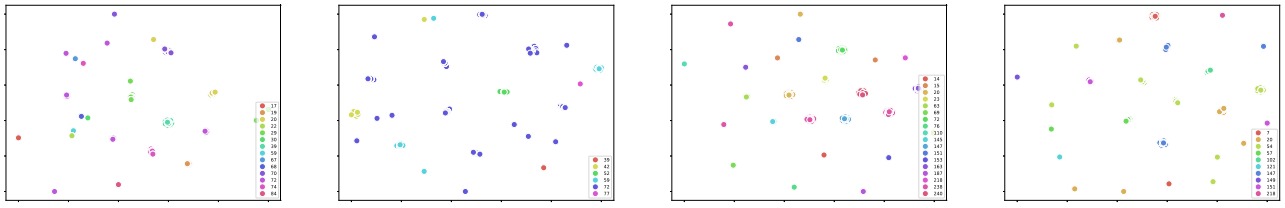
Introducing axioms based on the `hasAttribute` predicate, which enable FLPN to produce attribute-level classification, demonstrably improves overall image classification results (as shown in Table 3 by comparing against FLVN architecture). This section explores how the attention mechanism helps to identify specific attributes in images for both CNN- and ViT-based architectures (Fig. 4). Both the examples showcase the model's ability to pinpoint image regions with specific properties. Qualitatively, the ViT-based architecture seems to outperform the CNN-based one. In both the cases, limitations persist due to the lack of annotations at the attribute level, which, as also noted in prior works [23], can introduce noise into the attention maps.



(a) ResNet-101, $\mathcal{K}_r$, seen classes (b) ResNet-101, $\mathcal{K}$, seen classes (c) ResNet-101, $\mathcal{K}_r$, unseen classes (d) ResNet-101, $\mathcal{K}$, unseen classes

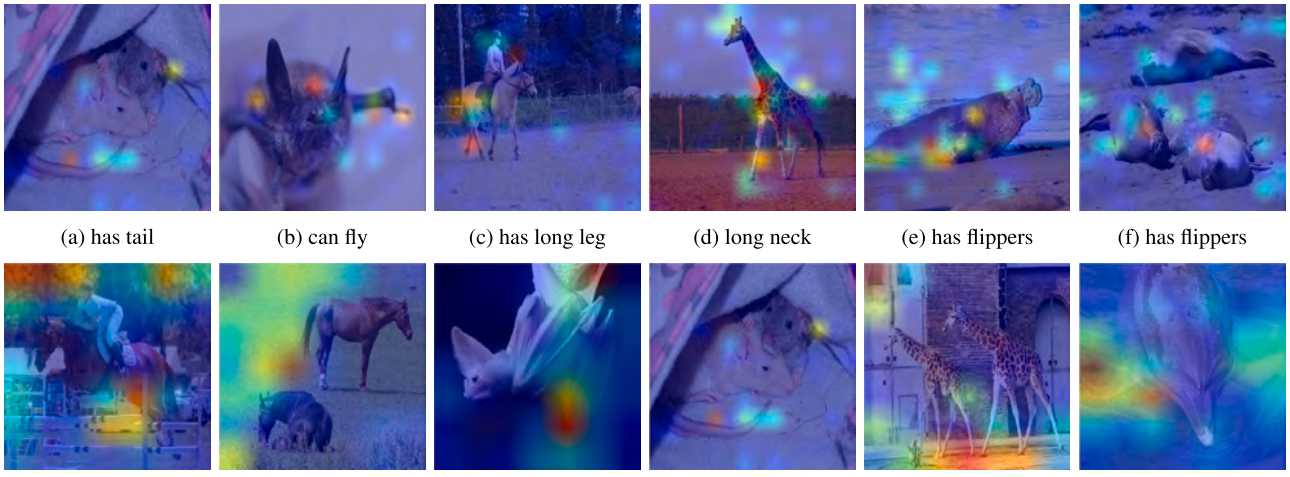
(e) ViT, $\mathcal{K}_r$, seen classes (f) ViT,$\mathcal{K}$, seen classes (g) ViT, $\mathcal{K}_r$, unseen classes (h) ViT, $\mathcal{K}$, unseen classes

Fig. 2. The class-level feature distribution on AWA2 dataset for FLPN with the ResNet-101 (a–d) and ViT (e–h) backbones is represented using t-SNE. The models in plots (a), (c), (e), and (g) contain a knowledge base $\mathcal{K}_r$ composed only of ϕ_{class} and ϕ_{masked}, whereas the knowledge base in the remaining plots includes all the axioms. The likelihood scores generated by the model for every class were utilized to produce these representations.



(a) seen classes; ResNet-101 backbone (b) unseen classes; ResNet-101 backbone (c) seen classes; ViT backbone (d) unseen classes; ViT backbone

Fig. 3. Attribute-level feature distribution for FLPN on AWA2 (a–b) and CUB (c–d) datasets represented using t-SNE.



(a) has tail (b) can fly (c) has long leg (d) long neck (e) has flippers (f) has flippers

(g) quadruped (h) has tail (i) can fly (j) has tail (k) has long leg (l) has flippers

Fig. 4. Attention maps for randomly selected examples and attributes from AWA2 with ViT (a–f) and ResNet101 (g–l) backbones.

5.4. Prototypical LTNs family: architectural comparison

The proposed architecture is an evolution and improvement of the previous architectures Proto-LTN [8] and FLVN [9], both based on similar NeSy principles. Table 5 details differences among the three architectures concerning the knowledge base $\mathcal{K}$ used for training the models, the *embedding space*, that is the shared embedding space used to compute the similarity between the image embeddings and the class prototypes, the *grounding* of the `isOfClass` predicate, how the *Batch* is constructed by selecting examples for the training set, the *Aggregator* which combines all axioms to compute the final satisfiability, and lastly the *backbone*.

Proto-LTN vs. FLPN Proto-LTN [8] differs both in the knowledge base and in the implementation of the `isOfClass` predicate. Proto-LTN leverages the image space as a shared embedding, with training involving random sample selection. It employs the Euclidean distance to reinforce similarity between images and prototypes belonging to the same class. In contrast, FLPN adopts a similarity metric that multiplies

image and prototype features, followed by a Softmax activation function. This formulation, incorporating mutual exclusivity constraints, facilitates end-to-end training without features collapsing or overfitting to the seen classes. Unlike Euclidean distance-based measures, classifying an image into one class decreases the likelihood that the same embedding will be assigned to other classes.

FLVN vs. FLPN In addition to their shared embedding space and backbone architecture, FLVN and FLPN exhibit fundamental design similarities. Some constraints inherent to FLVN structure preclude the implementation of a prototypical network. While FLVN incorporates attribute information by multiplying image features (projected into the attribute space) with the semantic attribute matrix, it does not explicitly generate class prototypes. As a result, directly incorporating a prototypical network for attributes within the FLVN architecture may not be feasible. In contrast, FLPN includes multiple prototypical networks, which however share initial layers to reduce the effective parameter count and prevent overfitting.

6. Conclusion

Building on prior works in ZSL [13,23] and embedding them within a NeSy framework [8,9], we developed a new NeSy architecture for ZSL and GZSL tasks named FLPN. Like FLVN, FLPN integrates axioms combining class-specific knowledge (e.g., class hierarchies) with inductive biases to handle dataset exceptions (e.g., "a non-ferocious lion") and establish image relationships (e.g., similar images belong to the same class). These axioms serve as semantic priors, addressing annotation gaps by considering class-level attributes and weighting image-level attributes, strengthening the NeSy foundation for GZSL tasks. Experimental results show that FLPN meets or exceeds existing methods on standard GZSL benchmarks with respect to both performance and explainability. FLPN does not require multiple backbone networks and maintains a parameter count comparable to traditional embedding techniques. The proposed approach can be integrated into other techniques, e.g., by modifying the predicate grounding accordingly. We see several expansion avenues: exploring different formulations for the `isOfClass` predicate (e.g., based on hyperbolic embeddings [59]), incorporating alternative attribute detection mechanisms, introducing contrastive loss as a logical constraint, expanding the knowledge base to include class-related information beyond class hierarchies, incorporating information from sources such as language models, or testing FLPN in a transductive ZSL context.

CRediT authorship contribution statement

Francesco Manigrasso: Writing – review & editing, Writing – original draft, Software, Methodology, Data curation, Conceptualization. **Fabrizio Lamberti:** Writing – review & editing, Supervision, Funding acquisition. **Lia Morra:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This study was carried out within the FAIR - Future Artificial Intelligence Research and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 – D.D. 1555 11/10/2022, PE00000013). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

Data availability

Public datasets were used for the experiments.

References

[1] D. Yu, B. Yang, D. Liu, H. Wang, S. Pan, A survey on neural-symbolic learning systems, Neural Netw. (2023).
[2] S. Badreddine, A.d. Garcez, L. Serafini, M. Spranger, Logic tensor networks, Artificial Intelligence 303 (2022) 103649.
[3] M. van Bekkum, M. de Boer, F. van Harmelen, A. Meyer-Vitali, A.t. Teije, Modular design patterns for hybrid learning and reasoning systems: a taxonomy, patterns and use cases, Appl. Intell. 51 (9) (2021) 6528–6546.
[4] F. Manigrasso, F.D. Miro, L. Morra, F. Lamberti, Faster-LTN: a neuro-symbolic, end-to-end object detection architecture, in: Proceedings of the 30th International Conference on Artificial Neural Networks, ICANN, Springer, 2021, pp. 40–52.
[5] I. Donadello, L. Serafini, Compensating supervision incompleteness with prior knowledge in semantic image interpretation, in: 2019 International Joint Conference on Neural Networks, IJCNN, 2019.
[6] S. Badreddine, L. Serafini, M. Spranger, logLTN: Differentiable fuzzy logic in the logarithm space, 2023, arXiv, arXiv:2306.14546.
[7] K. Sikka, J. Huang, A. Silberfarb, P. Nayak, L. Rohrer, P. Sahu, J. Byrnes, A. Divakaran, R. Rohwer, Zero-shot learning with knowledge enhanced visual semantic embeddings, 2020, arXiv:2011.10889.
[8] S. Martone, F. Manigrasso, F. Lamberti, L. Morra, PROTOtypical logic tensor networks (PROTO-LTN) for zero shot learning, in: 26th International Conference on Pattern Recognition, ICPR, 2022, pp. 4427–4433.
[9] F. Manigrasso, L. Morra, F. Lamberti, Fuzzy Logic Visual Network (FLVN): A neuro-symbolic approach for visual features matching, in: International Conference on Image Analysis and Processing, Springer, 2023, pp. 456–467.
[10] Y. Xian, C.H. Lampert, B. Schiele, Z. Akata, Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly, IEEE Trans. Pattern Anal. Mach. Intell. 41 (9) (2018) 2251–2265.
[11] I. Goodfellow, Deep learning, 2016.
[12] A.d. Garcez, L.C. Lamb, Neurosymbolic AI: The 3 rd wave, Artif. Intell. Rev. 56 (11) (2023) 12387–12406.
[13] D. Cheng, G. Wang, N. Wang, D. Zhang, Q. Zhang, X. Gao, Discriminative and robust attribute alignment for zero-shot learning, IEEE Trans. Circuits Syst. Video Technol. (2023).
[14] L. Zhou, Y. Liu, X. Bai, N. Li, X. Yu, J. Zhou, E.R. Hancock, Attribute subspaces for zero-shot learning, Pattern Recognit. 144 (2023) 109869.
[15] L. Zhang, T. Xiang, S. Gong, Learning a deep embedding model for zero-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2021–2030.
[16] P. Zhu, H. Wang, V. Saligrama, Generalized zero-shot recognition based on visually semantic embedding, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 2995–3003.
[17] L. Chen, H. Zhang, J. Xiao, W. Liu, S.-F. Chang, Zero-shot visual recognition using semantics-preserving adversarial embedding networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 1043–1052.
[18] L. Zhang, P. Wang, L. Liu, C. Shen, W. Wei, et al., Towards effective deep embedding for zero-shot learning, IEEE Trans. Circuits Syst. Video Technol. 30 (9) (2020) 2843–2852.
[19] H. Jiang, R. Wang, S. Shan, X. Chen, Transferable contrastive network for generalized zero-shot learning, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 9765–9774.
[20] Y. Li, Z. Liu, L. Yao, X. Chang, Attribute-modulated generative meta learning for zero-shot learning, IEEE Trans. Multimed. 25 (2021) 1600–1610.
[21] G.-S. Xie, L. Liu, X. Jin, F. Zhu, Z. Zhang, J. Qin, Y. Yao, L. Shao, Attentive region embedding network for zero-shot learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 9384–9393.
[22] S. Chen, Z. Hong, G. Xie, Q. Peng, X. You, W. Ding, L. Shao, GNDAN: Graph navigated dual attention network for zero-shot learning, IEEE Trans. Neural Netw. Learn. Syst. 35 (4) (2022) 4516–4529.
[23] Y. Du, M. Shi, F. Wei, G. Li, Boosting zero-shot learning via contrastive optimization of attribute representations, IEEE Trans. Neural Netw. Learn. Syst. (2023) 1–14.
[24] W. Xu, Y. Xian, J. Wang, B. Schiele, Z. Akata, VGSE: Visually-grounded semantic embeddings for zero-shot learning, in: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2022.
[25] Y. Yi, G. Zeng, B. Ren, L.T. Yang, B. Chai, Y. Li, Prototype rectification for zero-shot learning, Pattern Recognit. 156 (2024) 110750.
[26] Y. Liu, X. Gao, J. Han, L. Shao, A discriminative cross-aligned variational autoencoder for zero-shot learning, IEEE Trans. Cybern. 53 (6) (2023) 3794–3805, http://dx.doi.org/10.1109/TCYB.2022.3164142.
[27] S. Chen, Z. Hong, X. You, L. Shao, Semantics-conditioned generative zero-shot learning via feature refinement, Int. J. Comput. Vis. (2025) 1–18.

[28] Z. Akata, F. Perronnin, Z. Harchaoui, C. Schmid, Label-embedding for attribute-based classification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2013.
[29] S. Chen, G. Xie, Y. Liu, Q. Peng, B. Sun, H. Li, X. You, L. Shao, Hsva: Hierarchical semantic-visual adaptation for zero-shot learning, Adv. Neural Inf. Process. Syst. 34 (2021) 16622–16634.
[30] Z. Liu, Y. Li, L. Yao, X. Wang, G. Long, Task aligned generative meta-learning for zero-shot learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 2021, pp. 8723–8731.
[31] J. Li, M. Jing, K. Lu, Z. Ding, L. Zhu, Z. Huang, Leveraging the invariant side of generative zero-shot learning, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2019.
[32] R. Felix, I. Reid, G. Carneiro, et al., Multi-modal cycle-consistent generalized zero-shot learning, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 21–37.
[33] Z. Han, Z. Fu, S. Chen, J. Yang, Contrastive embedding for generalized zero-shot learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 2371–2381.
[34] Y. Gao, W. Feng, R. Xiao, L. He, Z. He, J. Lv, C. Tang, Improving generalized zero-shot learning via cluster-based semantic disentangling representation, Pattern Recognit. 150 (2024) 110320.
[35] J. Zhang, Q. Li, Y.-a. Geng, W. Wang, W. Sun, C. Shi, Z. Ding, A zero-shot learning framework via cluster-prototype matching, Pattern Recognit. 124 (2022) 108469.
[36] W. Xu, Y. Xian, J. Wang, B. Schiele, Z. Akata, Attribute prototype network for zero-shot learning, Adv. Neural Inf. Process. Syst. 33 (2020) 21969–21980.
[37] Y. Yu, Z. Ji, J. Han, Z. Zhang, Episode-based prototype generating network for zero-shot learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 14035–14044.
[38] C. Wang, S. Min, X. Chen, X. Sun, H. Li, Dual progressive prototype network for generalized zero-shot learning, in: Neural Information Processing Systems, 2021.
[39] S. Chen, Y. Li, J. Wang, Z. Zhang, Evolving semantic prototype improves generative zero-shot learning, in: Proceedings of the 40th International Conference on Machine Learning, ICML, 2023, DSP reference.
[40] J. Feinglass, A. Smith, R. Kumar, Part prototype network for generalized zero-shot learning, in: CVPR Workshops, 2024, PPN reference.
[41] D. Cheng, G. Wang, B. Wang, Q. Zhang, J. Han, D. Zhang, Hybrid routing transformer for zero-shot learning, Pattern Recognit. 137 (2023) 109270.
[42] Z. Chen, Y. Gao, C. Lang, L. Wei, Y. Li, H. Liu, F. Liu, Integrating topology beyond descriptions for zero-shot learning, Pattern Recognit. 143 (2023) 109738.
[43] W. Hou, D. Fu, K. Li, S. Chen, H. Fan, Y. Yang, ZeroMamba: exploring visual state space model for zero-shot learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, 39, (4) 2025, pp. 3527–3535.
[44] R. Ran, Q. Hu, T. Gao, S. Dong, Zero-shot learning based on vision transformer, in: 2023 International Conference on Pattern Recognition, Machine Vision and Intelligent Algorithms, PRMVIA, IEEE, 2023, pp. 24–28.
[45] I. Donadello, L. Serafini, A.D. Garcez, Logic tensor networks for semantic image interpretation, in: 26th International Joint Conference on Artificial Intelligence, 2017, pp. 1596–1602.
[46] L. Morra, A. Azzari, L. Bergamasco, et al., Designing logic tensor networks for visual sudoku puzzle classification, in: 17th International Workshop on Neural-Symbolic Learning and Reasoning, 2023, pp. 223–232.
[47] S. Chen, Z. Hong, G.-S. Xie, W. Yang, Q. Peng, et al., MSDN: Mutually semantic distillation network for zero-shot learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 7612–7621.
[48] W. Wang, Q. Li, Generalized zero-shot activity recognition with embedding-based method, ACM Trans. Sens. Netw. 19 (3) (2023) 1–25.
[49] S. Chen, Z. Hong, G.-S. Xie, W. Yang, Q. Peng, K. Wang, J. Zhao, X. You, Msdn: Mutually semantic distillation network for zero-shot learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 7612–7621.
[50] S. Chen, Z. Hong, W. Hou, G.-S. Xie, Y. Song, J. Zhao, X. You, S. Yan, L. Shao, TransZero++: Cross attribute-guided transformer for zero-shot learning, IEEE Trans. Pattern Anal. Mach. Intell. 45 (11) (2022) 12844–12861.
[51] H. Su, J. Li, K. Lu, L. Zhu, H.T. Shen, Dual-aligned feature confusion alleviation for generalized zero-shot learning, IEEE Trans. Circuits Syst. Video Technol. 33 (8) (2023) 3774–3785.
[52] J. Snell, K. Swersky, R. Zemel, Prototypical networks for few-shot learning, in: 31st International Conference on Neural Information Processing Systems, 2017, pp. 4080–4090.
[53] J. Li, P. Zhou, C. Xiong, S. Hoi, Prototypical contrastive learning of unsupervised representations, in: International Conference on Learning Representations, 2021.
[54] K. He, H. Fan, Y. Wu, S. Xie, R. Girshick, Momentum contrast for unsupervised visual representation learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 9729–9738.
[55] C. Wah, S. Branson, P. Perona, S. Belongie, Multiclass recognition and part localization with humans in the loop, in: 2011 International Conference on Computer Vision, IEEE, 2011, pp. 2524–2531.
[56] J. Xiao, J. Hays, K.A. Ehinger, A. Oliva, A. Torralba, SUN database: Large-scale scene recognition from abbey to zoo, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, 2010, pp. 3485–3492.
[57] M.-E. Nilsback, A. Zisserman, Automated flower classification over a large number of classes, Proc. the Indian Conf. Comput. Vis. Graph. Image Process. (2008) 722–729.
[58] T. Carraro, LTNtorch, 2022, URL https://doi.org/10.5281/zenodo.6394282.
[59] V. Khrulkov, L. Mirvakhabova, E. Ustinova, et al., Hyperbolic image embeddings, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 6418–6428.