Explaining Visual Models with Logic and Concept-Part Networks

Mathilde Bonin[1,*], Sergio Mover[1]

[1]LIX, CNRS, École polytechnique, Institut Polytechnique de Paris, Palaiseau, France

Abstract

Neural networks for vision are widely used but do not provide explanations for their decisions. A recent approach (i.e., ConSpec [1]) uses concepts to bridge the gap between pixel-like features and human-understandable text, and a propositional logic to express specifications of the model's behavior. However, such approach remains limited and cannot express simple properties involving multiple concepts, predicate on their strength (a proxy for the presence of a concept in the input), or predicate on multiple input images. This limitation is mostly due to the non-linear relationship between concepts' strengths and the latent space representation of the model, which makes satisfiability checking difficult and restricts the application of the logic to explain the network's decisions. We propose Concept Part Network (ConPnet), a novel architecture that integrates the concepts' strengths computation directly into a classifier and then uses these strengths for prediction. Such architecture allows us to extend ConSpec to support arbitrary linear combinations of concepts' strengths (ConSpec$_{\mathbb{R}}$), which are necessary to express a wider range of specifications, and efficiently check them with an SMT solver. Our initial experiments show that ConPnet retains competitive classification accuracy and supports effective explanation of predictions through ConSpec$_{\mathbb{R}}$ formulas.

Keywords

Vision-Language Models, Explainability, Concepts Specifications, Formal Specifications, SMT

1. Introduction

Visual models are the standard tools to solve visual tasks like classification, segmentation, and object recognition. Such neural networks are complex black box models and require more automatic techniques to explain their decisions. Since the input of neural networks for visual tasks are low-level features (e.g., pixels), they also do not allow a user to provide a "semantic" specification to express the expected behaviors of the model or understand "why" the model took a given decision. For such reasons, concepts [2] and formal logics based on concepts, such as ConSpec [1], have been proposed to provide a high-level, human-understandable specification to describe the model's behavior. For example, in an image classification task, concepts used to motivate a classification result could be "has wheels" or "has wings", and ConSpec can express properties like "if the neural network thinks there are more wheels than wings in the image, then the model classifies an object (and not an animal)".

ConSpec includes concepts with the $strength(x, \mathrm{con})$ function that expresses how much a concept con is present or not in the input x. More technically, $strength(x, \mathrm{con})$ is computed as the cosine similarity between a vector computed from the input image x (its representation in the model's latent space) and a constant vector of concepts, expressed in the range $[-1, 1]$ (-1 means the concept is absent, 1 means the concept is present). For such reason, checking the satisfiability of formulas using the strength function generally requires to use non-linear contraints, which is a non-trivial task (e.g., it would require Non-Linear Real Arithmetic theory in SMT, or non-linear optimization of a non-convex problem, to encode the Euclidian norm of a high-dimensional vector of about 500 elements that are common in modern architectures). In [1], the ConSpec syntax was limited to only allow a comparison of the strength of two concepts (e.g., $strength(x, \mathrm{con}_i) > strength(y, \mathrm{con}_j)$), which simplifies to linear constraints (see Equation (1)). However, such limitation restricts the usefulness of ConSpec when

8th International Workshop on Artificial Intelligence and fOrmal VERification, Logic, Automata, and sYnthesis. OVERLAY 2026 @ FLoC, July 18–19, Lisbon, Portugal

*Corresponding author.

mathilde.bonin@polytechnique.edu (M. Bonin); sergio.mover@polytechnique.edu (S. Mover)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-09

CEUR-WS.org/Vol-4324/paper5.pdf

trying to understand the neural network's behavior. For example, even expressing "the concept con_i is present when x is classified as c?" is hard in ConSpec (in fact, in [1] the "has concept" predicate is defined comparing the strengths of different concepts, assuming a concept is present only if its strength is higher than the strengths of other concepts, which is an arbitrary assumption and not true in general). As another example, after classifying an image x as an object with concepts' strengths has wings $= 0.5$ and has wheels $= 0.7$, one would like to quantify how much the strength of the "has wings" concept can change for the model to be able to still recognize the image as an object:

$$(|strength(x, \text{has wings}) - 0.5| < \epsilon \wedge strength(x, \text{has wheels}) = 0.7) \rightarrow predict(x, \text{object}),$$

where $\epsilon \geq 0$ is a parameter. Such properties, including the ones predicating on multiple input images, cannot be expressed and checked efficiently in ConSpec.

Here, we observe that if we focus on the problem of checking formulas to explain the decisions of a model we don't need to verify the entire neural network, but rather check formulas on the decision head of the model to explain its behavior. Note that this task is different and computationally easier than verifying the full neural network. Motivated by such observation, in this paper we propose the use of a different neural network architecture, Concept Part Network (ConPNet) (see Fig. 1b). ConPNet takes inspiration from Prototypical Part Network (ProtoPNet) [3], a class of interpretable neural networks that use prototypes (i.e., parts of representative examples of each class) to interpret their predictions. The main intuition of ConPNet is to integrate the computation of the concepts' strengths in the visual model and directly use the strengths for classification. This approach was briefly considered in the appendix of [4] but not used for checking formulas on the model's head. A consequence of adopting ConPNet as vision model is that each decision is interpretable, as when the network makes a prediction it also provides the strength of each concept. Another consequence is that, since the concepts' strengths are directly computed in the model, we obtain a simpler satisfiability problem for the specification (i.e., it can be expressed in Linear Real Arithmetic), where variables are directly the concepts' strengths and not a higher-dimensional vector in the latent space that requires to compute a non-linear norm.

The contributions of our paper are the following: (i) We use ConPNet, an architecture for visual models that integrates the computation of concepts' strengths in the model and directly uses them for classification. (ii) We extend ConSpec to express arbitrary linear combination of concepts' strengths (ConSpec$_{\mathbb{R}}$), which allows us to express a wider range of specifications and allows us to check such specifications automatically (i.e., with an SMT solver). (iii) We show an initial experimental evaluation, demonstrating that ConPNet is an efficient architecture for classification and that we can use ConSpec$_{\mathbb{R}}$ formulas to explain the model's behavior.

In Section 2 we discuss the related work, in Section 3 we introduce visual models and concepts, and in Section 4 we introduce ConSpec$_{\mathbb{R}}$ and ConPNet, and we show how ConPNet simplifies checking ConSpec$_{\mathbb{R}}$ formulas. We present the experimental evaluation in Section 5 and conclude in Section 6.

2. Related works

Concepts-based interpretability and explainability. Existing concept-based interpretable networks, such as ProtoPNet defined in [3] or the ante-hoc explainable model in [5], rely on learnt concepts rather than concepts chosen and defined by humans. The learnt concepts offer an insight on the internal encoding space of the model, but they need further methods, such as [6], to produce formal explanations of their behaviours. Even so, the interpretation of such explanations demands that the learnt concepts be translated into human-understandable visualisations, either using generative models such as [7] or projecting the learnt concepts back on the training data as in the original ProtoPNet architecture.

Logic specifications and analysis for vision models. Here, we focus on works (e.g., [8, 9, 1]) tackling the problem of specifying semantic properties of vision models and applying them to formal verification. Some approaches are fundamentally different for the definition of concepts and for how such concepts are obtained. For example, [9] proposes to train additional outputs from the latent space

of the vision model and use that as proxy for more semantic specifications. Importantly, training such output requires additional labeled datasets. The work of [8] proposes a fuzzy logic that predicates on similar concepts as ours for specifying semantic adversarial attacks. However, this approach learns the concepts as new outputs of the vision model (via linear layers) with additional labeled datasets for the concepts, as opposed to using a pre-trained VLM. The closest work to ours is ConSpec [1], which we build upon, as we use the same notion of concepts, how to extract them from a VLM, and, in practice, almost the same logic. As explained earlier, we differ from ConSpec as we allow to express more complex specifications involving linear combinations of concepts' strengths while avoiding the issue of non-linearity thanks to the different ConPnet architecture. Our work also focuses on the application of the specifications to explain the model's decisions, while ConSpec focuses on the application of the specifications to verify the model's behavior. As we will explain later in Section 4, ConSpec's verification approach assumes an oracle to compute a set of feasible encodings from a set of input images, which is not feasible, and in practice defines an approximation of such set. Under a similar, but unrealistic, assumption, we could tackle the neural network verification problem for $\text{ConSpec}_{\mathbb{R}}$ specifications and ConPnet visual models.

Neural network verification. While our main goal is not to verify the entire vision model, techniques for the formal verification of neural networks (e.g., [10, 11, 12]) are relevant to the satisfiability checking of the specifications we propose. In our settings, we employ a generic SMT solver, which would not scale in verifying neural networks, as all the constraints we have are linear, given the head of ConPnet is a single linear layer (i.e., there is no activation function), and the size of the input is small (i.e., the number of concepts), as opposed to the size of the embedding vector of the vision model. State-of-the-art neural network verification techniques will be necessary to check if the whole network satisfies a ConSpec specifications (i.e., in case we solve the verification problem).

3. Background on concept extraction from visual models using VLMs

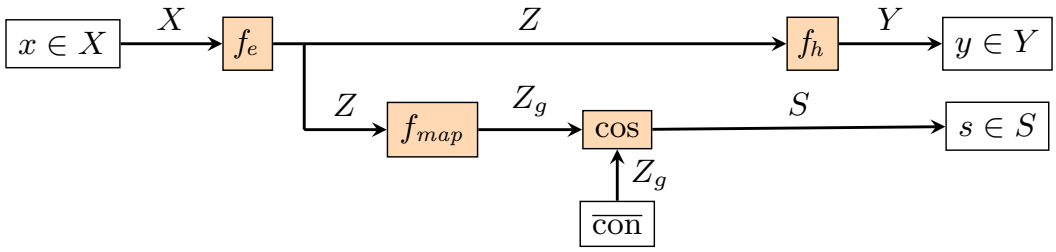
We revise the setting of ConSpec [1] and we refer to Fig. 1a for the original architecture of the vision model and the extraction of concepts using a Visual Language Model. We will introduce the architecture shown in Fig. 1b later in Section 4.

A *vision model* $f : X \mapsto Y$ maps an input image $x \in X \subseteq \mathbb{R}^d$ to a class $y \in Y$.

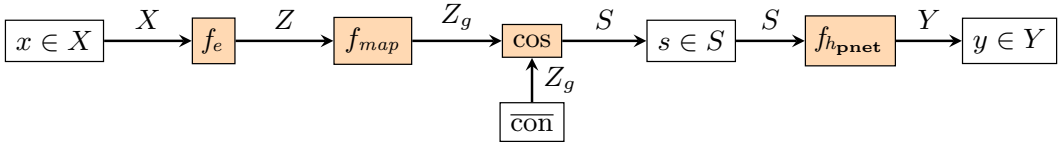
In Fig. 1b, $f : X \mapsto Y$ is a neural network composed as $f = f_h \circ f_e$, where $f_e : X \mapsto Z$ is a feature extractor (*encoder*) that maps an input image to a latent space Z, and $f_h : Z \mapsto Y$ is a classifier (*head*) that maps the latent representation to a class. The feature extractor f_e is often a pre-trained model and the head f_h is a simpler network (e.g., a linear layer) trained to accomplish a specific task.

A Visual Language Model (VLM) g maps both an image $x \in X_g$ and a text description t to the same latent space Z_g via two encoders $g_e : X \mapsto Z_g$ and $g_t : T \mapsto Z_g$, and is trained simultaneously from text and images. Intuitively, the VLM g learns to encode images and text in the same latent space Z_g so that an image encoding will be similar to the encoding of its text description. This property is leveraged to compute concepts: given a word of interest in natural language $w \in T$ (the name of the concept such as "wheels"), a concept vector con_w is computed by first injecting w into a set of templates $t \in T$, e.g. "a picture of the w". Each instantiated template is then encoded with the language part of the VLM, $E_w = \{g_t(t(w)) \mid t \in T\}$. The concept vector con_w is defined as the mean vector of E_w. Crucially, given an image $x \in X_g$ and its encoding $g_e(x)$, the *strength* of a concept w in the image is the cosine similarity between the vectors $g_e(x)$ and $\text{con}_w \in \mathbb{R}^{Z_g}$ (i.e., $\cos(g_e(x), \text{con}_w)$). For example, if the concept "has_legs" has a strength of 1 for an image of a dog, it means that the two vectors are aligned (i.e., the "has_legs" is a concept present in the image), -1 means the two vectors are in opposite directions, and 0 means they are orthogonal. There exists several VLMs, for example CLIP [13], that have been trained on large datasets of images and their associated text descriptions.

The approach of [4] bridges a vision model and a VLM, as their latent spaces Z and Z_g are different, and allows us to extract concepts from the vision model. In Figure 1, the matrix $\overline{\text{con}} \in \mathbb{R}^{|\text{Concepts}| \times Z_g}$ is



(a) Original ConSpec architecture from [1].



(b) Concept Part Network (ConPnet) architecture.

Figure 1: Comparison of concept extraction architectures for vision models (orange colored rectangle represents functions and arrows represent the flow of data and are labeled with the data's domain).Both architectures classify one image $x \in X$ to a class $y \in Y$ and produce a strength vector $s \in S$.f_{map} maps the latent space Z of the vision model to the latent space Z_g of a VLM. cos is the cosine similarity between the image encoding in the VLM latent space and the concept encoding $\overline{\text{con}}$, a constant vector computed once using a VLM not shown in the picture). The original architecture at the top computes the concept strengths independently from the classification, while the ConPnet architecture at the bottom classifies the input using the concepts' strength.

the matrix of the concept vectors. Note that $\overline{\text{con}}$ is constant and computed only once with the VLM (which is not shown in the figure). The idea is to train a mapping $f_{map} : Z \mapsto Z_g$ from the latent space Z of the vision model to the VLM latent space Z_g. As shown in Fig. 1a, the input image $x \in X$ is fed to the encoder (i.e., $f_e(x)$) and then to the mapping function (i.e., $f_{map}(f_e(x))$). The result after applying f_{map} is a vector in Z_g that can be compared with th encoding of each concept vector $\overline{\text{con}}_i \in \overline{\text{con}}$ via cosine similarity (i.e., $\cos(f_{map}(f_e(x)), \overline{\text{con}}_i)$), resulting in the strength of the concept in the image x. The strength of the concept determines "how much" a concept is present in the image, and can be used to explain in a human understandable way the prediction of the vision model f.

4. ConSpec$_{\mathbb{R}}$: concept-based specification over the reals

CONcept-based SPECifications (ConSpec) [1] is a logic that uses standard propositional logic operators and predicates on the predicted classes and the strengths of the concepts. We extend ConSpec to express linear combinations of the concepts' strengths and the classes' outputs (shown in blue) as follows:

$$\varphi ::= \mathit{predict}(x, \mathrm{c}) \;\Big|\; \sum_i a_i \cdot \mathit{out}(x_i, \mathrm{c}_i) \bowtie b \;\Big|\; \sum_i a_i \cdot \mathit{strength}(x_i, \mathrm{con}_i) \bowtie b \;\Big|\; \varphi \wedge \varphi \;\Big|\; \varphi \vee \varphi \;\Big|\; \neg\varphi,$$

where $x_i, x \in \text{Vars}$ are image variables, $\mathrm{con}_i \in \text{Concepts}$ is a set of constants for the concepts, $\mathrm{c}, \mathrm{c}_i \in \text{Classes}$ is a set of constants for the classes, $\bowtie = \{<, >, \geq, \leq\}$, and the coefficients a_i, and b are real numbers.The predicate $\mathit{predict}(x, \mathrm{c})$ evaluates to true if the neural network classifies the image x as class c, the function $\mathit{out}(x, \mathrm{c})$ is the output of the network on the image x for class c, and the function $\mathit{strength}(x, \mathrm{con})$ is the strength of the concept con in the image x. To avoid ambiguity with the original ConSpec, we refer to this extension as ConSpec$_{\mathbb{R}}$.

The semantic of ConSpec$_{\mathbb{R}}$ is parametric in the visual model $\overline{f}$ used to produce the network's outputs and the networks $\overline{f}_{\mathit{strength}}$ used to compute the concepts' strengths. Here, we assume a generic function $\overline{f}_{\mathit{strength}} : X \mapsto [-1, 1]$ that computes the strength of the concept con_i in the image x, and we will

later discuss the limitations when instantiating it as in [1]. We write $\mu : \text{Vars} \mapsto \mathbb{R}^d$ for an assignment from the variables Vars to images. We write $\mu \models_{\overline{f},\overline{f}_{strength}} \varphi$ if the model μ satisfies the formula φ with respect to $\overline{f}$ and $\overline{f}_{strength}$. The semantic $\mu \models_{\overline{f},\overline{f}_{strength}} \varphi$ is defined as follows:

$$\begin{aligned}
\mu \models_{\overline{f},\overline{f}_{strength}} predict(x, \mathrm{c}) \quad &\text{iff} \quad \arg\max_{i \in d} \overline{f}(\mu(x))_i = \mathrm{c} \\
\mu \models_{\overline{f},\overline{f}_{strength}} \sum_i a_i \cdot out(x_i, \mathrm{c}_i) \bowtie b \quad &\text{iff} \quad \sum_i a_i \cdot \overline{f}(\mu(x))_i \bowtie b \\
\mu \models_{\overline{f},\overline{f}_{strength}} \sum_i a_i \cdot strength(x_i, \mathrm{con}_i) \bowtie b \quad &\text{iff} \quad \sum_i a_i \cdot \overline{f}_{strength}(\mu(x), \mathrm{con}_i) \bowtie b \\
\mu \models_{\overline{f},\overline{f}_{strength}} (\varphi_1 \wedge \varphi_2) \quad &\text{iff} \quad \mu \models_{\overline{f},\overline{f}_{strength}} \varphi_1 \text{ and } \mu \models_{\overline{f},\overline{f}_{strength}} \varphi_2 \\
\mu \models_{\overline{f},\overline{f}_{strength}} (\varphi_1 \vee \varphi_2) \quad &\text{iff} \quad \mu \models_{\overline{f},\overline{f}_{strength}} \varphi_1 \text{ or } \mu \models_{\overline{f},\overline{f}_{strength}} \varphi_2 \\
\mu \models_{\overline{f},\overline{f}_{strength}} \neg\varphi \quad &\text{iff} \quad \mu \not\models_{\overline{f},\overline{f}_{strength}} \varphi.
\end{aligned}$$

Satisfiability check for the network f from [1]. In the original ConSpec settings [1] and the architecture from Fig. 1a, the function $\overline{f}$ is instantiated as the vision model $f(x) := f_h(f_e(x))$, and the strength function $\overline{f}_{strength}$ is instantiated as $f_{strength}(x, \mathrm{con}_i) := \cos(\overline{\mathrm{con}}_i, f_{map}(f_e(x)))$. Here, $\overline{\mathrm{con}} \in \mathbb{R}^{|\text{Concepts}| \times Z_g}$ are the *constant* vectors that contains the mean representation of each concept in the latent space Z_g, and $\overline{\mathrm{con}}_i$ is the i-th vector for the concept $\mathrm{con}_i \in \text{Concepts}$. Checking the satisfiability of φ requires to solve a difficult neural network verification problem as f is a visual model and $f_{strength}$ requires to compute the cosine similarity, which is a non-linear function. The approach from [1] avoid such issues adopting two simplifications.

First, it restricts the input space with a set $B \subseteq X$ and assumes one can compute $B_e := \{f_e(x) \mid x \in B\}$ (i.e., the exact output of the encoder f_e on the set B). Under such assumption, one can "forget" about the encoder network f_e and directly considers inputs $x_e \in B_e$. With such simplification, the function $\overline{f}$ is only the output of the head part of the network, $f_h(x_e)$, and the strength function $\overline{f}_{strength}$ is the function $f^o_{strength}(x_e, \mathrm{con}_i) := \cos(\overline{\mathrm{con}}_i, f_{map}(x_e))$ (See Theorem 1 from [1]):

$$\mu \models_{f, f_{strength}} \varphi \quad \text{iff} \quad \exists\, \mu, \mu_Z \text{ such that if } f_e(\mu) = \mu_Z \text{ , then } \mu_Z \models_{f_h, f^o_{strength}} \varphi.$$

In the above formula, we write μ_Z for an assignment to a set of variables Vars_Z where each variable is a vector in the space Z. We abuse the notation and write $f_e(\mu) = \mu_Z$ to mean that μ_Z is the assignment of the variables in Vars_Z such that $f_e(\mu(x)) = \mu_Z(z_x)$ for all $x \in \text{Vars}$.

Then, the second restriction limits the logic to compare strength predicates with each other and always over the same input (i.e., $strength(x, \mathrm{con}_i) > strength(x, \mathrm{con}_j)$). This restriction simplifies the computation of the strengths to a linear constraint, instead of a non-linear one, as one can simplify the norm $\|v\| = \sqrt{\sum_k v_k^2}$ from the following inequality (i.e., the denominator $\|f_{map}(x)\|$):

$$strength(x, \mathrm{con}_i) > strength(x, \mathrm{con}_j) := \sum_k \frac{f_{map}(x)_k}{\|f_{map}(x)\|} \cdot \overline{\mathrm{con}}_i > \sum_k \frac{f_{map}(x)_k}{\|f_{map}(x)\|} \cdot \overline{\mathrm{con}}_j. \tag{1}$$

Given both assumptions and since f_{map} and f_h are linear, one can encode the satisfiability of φ with linear constraints that use *the latent space* variables Vars_Z([1] implements such check via linear programming and further restrict the grammar avoiding negation and disjunction). As in ConSpec$_{\mathbb{R}}$ we can compare the strength of a concept with constants (i.e., $strength(x, \mathrm{con}_i) > b$) and, more generally, write an arbitrary linear combination of strength predicates, *we cannot encode the cosine similarity using a the linear encoding* in Equation (1).

Satisfiability check for ConPnet. We depart from the neural network architecture from Fig. 1a and use an approach similar to *prototypical part networks* [3], where the classification is done using

the similarity of the input image with a set of prototypes, i.e., latent representations of common parts of the images in a dataset. Note that prototypes do not correspond to semantic meaningful words, as opposed to concepts, so they are not easy to use in a specification language.

The Concept Part Network (ConPnet) shown in Fig. 1b uses the similarity of the input image with a set of concepts to directly classify the image. The network is defined as $f_{\mathbf{pnet}}(x) = f_{h_{\mathbf{pnet}}}(\cos(\overline{\text{con}}, f_{map}(f_e(x))))$, where we extend the cosine similarity cos to a matrix (i.e., $\cos(\overline{\text{con}}, f_{map}(f_e(x))) = (\cos(\overline{\text{con}}_1, f_{map}(f_e(x)), \ldots, \cos(\overline{\text{con}}_{|\text{Concepts}|}, f_{map}(f_e(x)))$, and the other blocks are as before, except for the network's head $f_{h_{\mathbf{pnet}}} : S \mapsto Y$.

If we restrict the input space with a set $B \subseteq X$ and we assume we can compute $B_{\text{S}} := \{\cos(\overline{\text{con}}, f_{map}(f_e(x))) \mid x \in B\}$, we obtain a similar result for verifying ConSpec$_{\mathbb{R}}$ formulas as [1]:

$$\begin{aligned} \mu \models_{f_{\mathbf{pnet}}, f_{strength}} \varphi \quad &\text{iff} \quad \exists\, \mu \text{ such that } \cos(\overline{\text{con}}, f_{map}(f_e(\mu))) \models_{f_{h_{\mathbf{pnet}}}, id} \varphi, \\ &\text{iff} \quad \exists\, \mu, \mu_{\text{S}} \text{ such that if } \cos(\overline{\text{con}}, f_{map}(f_e(\mu))) = \mu_S \text{ , then } \mu_S \models_{f_{h_{\mathbf{pnet}}}, id} \varphi, \end{aligned}$$

where $id : \text{S} \mapsto \text{S}$ is the identity function, and μ_S is an assignment from a set of variables Vars_S to the strength domain S. Under such assumption, one can "forget" the network before the concepts' strengths, so the strength function in the semantic is just the identity id. In the satisfiability check we introduce a set of real-valued variables containing the strength $\text{Vars}_S := \{s_{\text{con}} \in [0, 1] \mid \text{con} \in \text{Concepts}\}$ and, as $f_{h_{\mathbf{pnet}}}$ is linear,the output of the network $out(x, \text{c})$ is a linear combination of the strength variables Vars_S. In this way, we decide the satisfiability of φ by directly finding an assignment to the strength variables Vars_S with an SMT solver, as the resulting encoding is a Linear Real Arithmetic formula.

A weakness of the approach is that the computation of B_{S} would require us to compute the non-linear cosine similarity function for a set of inputs, which is a hard problem. Since our main goal is to use the logic to explain the neural network decisions, and not that a specification holds for all the inputs of the vision model, we ignore the computation of B_{S} (i.e., in practice, we ignore the encoder f_e).

5. Experiments

We have two main research questions:

RQ1 Are concepts a good abstraction for classic computer vision tasks? Here, we restrict to: can we use concepts as features in a classification task?

RQ2 Can we use ConSpec$_{\mathbb{R}}$ to express specifications useful for explaining the model's behaviour?

We first show that our architecture has a satisfying accuracy, comparable to the original networks trained in the same conditions. We also show that chosen concepts are relevant by showing a loss of accuracy on restricted subsets of concepts. Finally, we show that concepts can be used to compute easy-to-interpret explanations of the model, and give insight on the classification decision.

5.1. conPNet for classification

Setup. We choose to use the 18 concepts defined by [14] and used by [1] for their logic (the list of concepts is visible in Table 3). These concepts were chosen for the classes of the CIFAR10 dataset but evaluated on a subset of the ImageNet dataset. We choose to use the same concepts, with the expectation that they have been well-chosen for the classes and not just the specific ImageNet synsets. We build a ConPnet model using a beheaded ResNet50 pretrained on ImageNet as the encoder (i.e., we used the encoder of ResNet50 as f_e). We then trained the head $f_{h_{\mathbf{pnet}}}$ as a unique linear layer that takes as input the strengths of the concepts, i.e. the cosine similarity values between the chosen concepts and the encodings of the input images, as shown in Fig.1b.

In practice, the accuracy of such a network depends on the set of concepts used. When considering a classification task for all the 10 different classes of CIFAR10, we observe that the accuracy of the model is rather low (cf Table 1), and the best model has an accuracy of 65%. We suppose that this

Table 1
Mean accuracies of ConPNet models on 3 classification tasks. They are compared to ResNet50 baselines on the same tasks, results directly copied from the sources cited for RIVAL10 and CIFAR10. Accuracy for Binary CIFAR is the best accuracy obtained for 100 epochs of training, using a ResNet50 model pretrained on ImageNet.

		Dataset		
		Binary CIFAR	CIFAR-10	RIVAL-10
ResNet50	Accuracy	99.07	93.03[15]	99.10[14]
conPNet	Accuracy	93.78	62.34	87.39
	Concepts/class	9	1.8	1.8
	f_e pretrained on target	no	no	yes

low accuracy is due to the low ratio of concepts per class: to provide a measure for comparison, the prototype-part network defined by [3] uses at least 10 prototypes (and not concepts) per class to achieve SotA accuracy. In order to have a comparable ratio of concepts per class, we define Binary CIFAR: using the CIFAR10 dataset, we map each of the original 10 classes to either "animal" or "object". We obtain a slightly unbalanced dataset with 6000 images in the $animal$ class, and 4000 images in the $object$ class.

Considering the accuracies achieved on different datasets, cf Table 1, we make the following hypotheses:

- the higher accuracy on Binary CIFAR compared to CIFAR10 makes us hypothise that a higher ratio of concepts per class is preferable. This hypothesis is confirmed by the results of Table 2.
- the higher accuracy on RIVAL10 compared to CIFAR10 makes us hypothise that pretraining the encoder f_e on the target dataset's images and using higher quality images may offset the low ratio of concepts per class. This difference in accuracy could also be due to the concepts having been chosen specifically for RIVAL10.

To verify the hypothesis that the network's accuracy depends on the number of concepts and their quality, we build different classifiers for Binary CIFAR using different random subsets of the original set of concepts with different sizes. We built the subset so that each concept appears in at least one subset. Finally, we trained 10 different ConPNet networks on 10 different subsets of concepts of size 5, and 10 networks on 10 different subsets of size 10. We compare their final accuracy, and we look at the subset of concepts of the worse and best networks for each category. We also retrain one linear layer directly on the encodings of the network for comparison.

Table 2
Accuracies of models on Binary CIFAR classification task. The parentheses indicate the number of concepts sampled from the original 18 concepts used for the subset to train the model. For 5 and 10 concepts, we randomly sample the concepts 10 times then report the mean and standard deviation of the accuracy over the trained ConPNet models.

Model	ConPNet (18)	ConPNet (5)	conPNet (10)	ResNet50 finetuned
Accuracy	91.24	76.48 ± 3.50	81.94 ± 1.91	96.93

Results. From the accuracy results in Table 2, the models trained on concepts that are not significantly more represented in one class over the other (such as patterned, which appears in at least 20% of the samples of the original 10 classes) seem to have a slightly lower accuracy. Surprisingly, the best model on 5 concepts was trained on the subset $\{long\ snout, wings, text, wet, tail\}$, where only long snout and text are representative of only one of the two classes. This model scores 82% accuracy, which outperforms 4 of the 10 models trained with 10 concepts. Models trained on concepts represented in only one class performed worse than the ones trained on concepts with more variety (71 to 74% accuracy).

Interpretation. The result presented in Table 2 validate our hypotheses: the number of concepts chosen is the main limiting factor to the accuracy. With enough concepts, we achieve similar results to classic machine learning models. Furthermore, we can observe that more concepts allow for more flexibility, but a smaller set of well chosen concepts can have comparable accuracy to a bigger set of loosely picked ones. In practice, we expect experts in the field to choose the concepts relevant for the task.

5.2. Explanability through specification

In this section, we want to show that ConSpec$_{\mathbb{R}}$ can be used to produce explanations with concepts for a ConPnet model.

Setup. Using the best ConPnet model with a ResNet50 backbone and a linear head (18,10) trained with the Cross Entropy Loss that we obtained from 5.1, we evaluate a robustness query on 3000 inputs from the test partition of RIVAL10. In the following, let $\vec{s}$ be the vector of the concepts' strengths for an input x with $\vec{s}_i$ the strength of con_i, c be the class predicted by $f_{\mathbf{pnet}}(x)$, and $\epsilon \geq 0$, be a positive real number. We express if the concept con_i is relevant for the classification of x as c with the following specification:

$$\varphi_{\vec{s},\mathrm{c},\mathrm{con}_i,\epsilon} := \left(|\mathit{strength}(x,\mathrm{con}_i) - \vec{s}_i| < \epsilon \wedge \bigwedge_{\substack{\mathrm{con}_j \in \mathrm{Concepts} \\ \mathrm{con}_j \neq \mathrm{con}_i}} \mathit{strength}(x,\mathrm{con}_j) = \vec{s}_j \right) \implies \mathit{predict}(x,\mathrm{c}). \quad (2)$$

The formula $\varphi_{\vec{s},\mathrm{c},\mathrm{con}_i,\epsilon}$ is satisfiable when the strength of all concepts but con_i are unchanged, and we modify the strength of con_i by a value smaller than ϵ, then the head $f_{h_{\mathbf{pnet}}}$ still predicts c. In practice, we check the validity of $\varphi_{\vec{s},\mathrm{c},\mathrm{con}_i,\epsilon}$ for a fixed set of $\epsilon \in \{0.05, 0.1, 0.3, 1, 2\}$ and an input x. If the formula is not valid, it means that con_i is relevant for the classification of x as c. We report the percentage of the 3000 inputs for which $\varphi_{\vec{s},\mathrm{c},\mathrm{con}_i,\epsilon}$ is not valid for each concept con_i and each ϵ. We implemented the validity check using PySMT [16] and the Z3 SMT solver [17] and checking the validity of each formula is almost instantaneous.

Results. The highest the percentage of failure, the more relevant the concept is for the classification task: if all inputs are falsely classified when the concept's strength is modified, it means that this concept is critical for the task and a modification leads to failure of classification. All concepts used are very sensible to high perturbation (100% failure rate for $\epsilon \geq 1$). When considering $\epsilon = 0.1$, we observe in Table 3 that $rectangular$ has the least impact on classification. This allow us to deduce that $rectangular$ is the least relevant concept for the task: perturbating its value to up to 0.1 (i.e. 10% of its maximal value, 5% of its theoretical amplitude) causes misclassifications one third of the time. Perturbation of the concept $beak$ leads to misclassification in 93% of the cases: it is a critical concept for the classification task.

Interpretation. Different specifications can be checked on the model using concepts. We chose to use the formula (2) to illustrate the importance of each concept for the classification, and their robustness to perturbations. In this case, we can conclude from Table 3 that the subset of concepts most critical for the classification is $\{beak, wheels, wings, mane, colored\ eyes, floppy\ ears, long\ snout, tail, horns\}$, as they have the highest rate of misclassification, all perturbations considered. On the other hand, $rectangular$ has a more limited role.

Table 3
Percentage of datapoints (over a total of 3000) where $\varphi_{\vec{s},\mathrm{c},\mathrm{con}_i,\epsilon}$ is not valid for different values of ϵ, on RIVAL10.

Concept \ ϵ	0.05	0.1	0.3	1	2
beak	34.8%	93.2%	100.0%	100.0%	100.0%
wheels	43.9%	87.0%	100.0%	100.0%	100.0%
wings	41.3%	80.3%	100.0%	100.0%	100.0%
mane	45.3%	77.8%	100.0%	100.0%	100.0%
colored eyes	34.4%	73.5%	100.0%	100.0%	100.0%
floppy ears	34.7%	72.4%	100.0%	100.0%	100.0%
long snout	38.8%	71.2%	100.0%	100.0%	100.0%
tail	38.9%	69.8%	93.8%	100.0%	100.0%
horns	35.7%	66.6%	100.0%	100.0%	100.0%
patterned	30.7%	60.8%	92.2%	100.0%	100.0%
metallic	28.3%	58.3%	95.4%	100.0%	100.0%
hairy	28.6%	56.6%	98.4%	100.0%	100.0%
tall	24.9%	50.2%	99.2%	100.0%	100.0%
wet	29.3%	47.5%	92.8%	100.0%	100.0%
ears	25.0%	46.9%	99.7%	100.0%	100.0%
text	22.6%	45.6%	83.3%	100.0%	100.0%
long	20.6%	38.3%	93.9%	100.0%	100.0%
rectangular	17.3%	30.5%	91.0%	100.0%	100.0%

6. Conclusion

In this paper, we extended the ConSpec approach for explaining the decision of visual models. The main technical contribution is using the Concept Part Network (ConPnet), which allows us to express more complex specifications on involving the concepts' strengths, as checking when a concept has been relevant or not for classification. Our initial experiments show that ConPnet is as efficient as a standard architecture for classification, as long as we provide enough concepts, and that ConSpec$_{\mathbb{R}}$ formulas can be used to explain the model's behavior.

As close future work, we plan to extend the generality of our experiments to include more complex datasets (e.g., Rival10), which may require more concepts, test more diverse specifications, and experiment with a robust semantic. We also plan to explore the computation of concept-based abductive explanations, which requires to compute explanations in the latent space similarly to [6] but using concepts instead of prototypes, and explanation via MUSes extraction [18], using the abstraction we can define from the concepts' strengths (e.g., one can define a predicate abstraction with a set of finite predicates settings threshold on the concepts' strengths).

As a longer-term goal, we plan to extend our work to more complex visual models for object recognition, segmentation, and video classification, which would require to extend the specification language to express properties over sets of objects and their associated concepts, and properties over sequences of concepts (e.g., a temporal logic).

Acknowledgments

This work was partially supported by the SAIF project, funded by the "France 2030" government investment plan managed by the French National Research Agency, under the reference ANR-23-PEIA-0006.

Declaration on Generative AI

During the preparation of this work, the authors used Github Copilot (using Claude Sonnet 4.6 and GPT-5.4 models) to paraphrase and reword paragrahs. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] R. Mangal, N. Narodytska, D. Gopinath, B. C. Hu, A. Roy, S. Jha, C. S. Păsăreanu, Concept-based analysis of neural networks via vision-language models, in: International Symposium on AI Verification, Springer, 2024, pp. 49–77.

[2] E. Poeta, G. Ciravegna, E. Pastor, T. Cerquitelli, E. Baralis, Concept-based explainable artificial intelligence: A survey, ACM Computing Surveys (2023).

[3] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, J. K. Su, This looks like that: deep learning for interpretable image recognition, Advances in neural information processing systems 32 (2019).

[4] M. Moayeri, K. Rezaei, M. Sanjabi, S. Feizi, Text-to-concept (and back) via cross-model alignment, in: International Conference on Machine Learning, PMLR, 2023, pp. 25037–25060.

[5] A. Sarkar, D. Vijaykeerthy, A. Sarkar, V. N. Balasubramanian, A framework for learning ante-hoc explainable models via concepts, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 10286–10295.

[6] J. Soria, Z. Chihani, J. Girard-Satabin, A. Grastien, R. Xu-Darme, D. Cancila, Formal abductive latent explanations for prototype-based networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, 30, 2026, pp. 25590–25598.

[7] J. Parekh, Q. Bouniot, P. Mozharovskyi, A. Newson, F. d'Alché Buc, Restyling unsupervised concept based interpretable networks with generative models, arXiv preprint arXiv:2407.01331 (2024).

[8] R. Dankworth, G. Schwalbe, Attack logics, not outputs: Towards efficient robustification of deep neural networks by falsifying concept-based properties, in: OVERLAY, CEUR Workshop Proceedings, CEUR-WS.org, 2025, pp. 153–165.

[9] C. Cheng, C. Huang, T. Brunner, V. Hashemi, Towards safety verification of direct perception neural networks, in: 2020 Design, Automation & Test in Europe Conference & Exhibition, DATE 2020, Grenoble, France, March 9-13, 2020, IEEE, 2020, pp. 1640–1643. URL: https://doi.org/10.23919/DATE48585.2020.9116205. doi:10.23919/DATE48585.2020.9116205.

[10] G. Katz, D. A. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljic, D. L. Dill, M. J. Kochenderfer, C. W. Barrett, The marabou framework for verification and analysis of deep neural networks, in: CAV (1), Lecture Notes in Computer Science, Springer, 2019, pp. 443–452.

[11] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, M. Vechev, Ai2: Safety and robustness certification of neural networks with abstract interpretation, in: 2018 IEEE Symposium on Security and Privacy (SP), 2018, pp. 3–18. doi:10.1109/SP.2018.00058.

[12] S. Wang, H. Zhang, K. Xu, X. Lin, S. Jana, C. Hsieh, J. Z. Kolter, Beta-crown: Efficient bound propagation with per-neuron split constraints for neural network robustness verification, in: NeurIPS, 2021, pp. 29909–29921.

[13] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, I. Sutskever, Learning transferable visual models from natural language supervision, in: ICML, Proceedings of Machine Learning Research, PMLR, 2021, pp. 8748–8763.

[14] M. Moayeri, P. Pope, Y. Balaji, S. Feizi, A comprehensive study of image classification model sensitivity to foregrounds, backgrounds, and visual attributes, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022.
[15] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
[16] M. Gario, A. Micheli, Pysmt: a solver-agnostic library for fast prototyping of smt-based algorithms, in: SMT Workshop 2015, 2015.
[17] L. M. de Moura, N. S. Bjørner, Z3: an efficient SMT solver, in: TACAS, Lecture Notes in Computer Science, Springer, 2008, pp. 337–340.
[18] A. Ignatiev, N. Narodytska, J. Marques-Silva, Abduction-based explanations for machine learning models, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, 2019, pp. 1511–1519.