LOGICDEF: An Interpretable Defense Framework Against Adversarial Examples via Inductive Scene Graph Reasoning

Yuan Yang, James C Kerce and Faramarz Fekri

Georgia Institute of Technology
{yyang754@, clayton.kerce@gtri., faramarz.fekri@ece.}gatech.edu

Abstract

Deep vision models are successfully employed in many applications. However, recent studies have shown that they are vulnerable to adversarial sample attacks. Existing adversarial defense methods are either limited to specific types of attacks or are too complex to be applied to practical vision models. More importantly, these methods rely on techniques that are not interpretable to humans. In this work, we argue that an effective defense should produce an explanation as to why the system is attacked, and by using a representation that is easily readable by a human user, e.g. a logic formalism. To this end, we propose *logic adversarial defense* (LOGICDEF), a defense framework that utilizes the *scene graph* of the image to provide a contextual structure for detecting and explaining object classification. Our framework first mines inductive logic rules from the graph to predict the objects, and then uses these rules to construct a defense model that alerts the user when the vision model violates those rules. The defense model is interpretable and its robustness can be further enhanced by incorporating commonsense knowledge from ConceptNet. Moreover, we propose a curriculum learning of the defense model using object taxonomy, yielding additional improvements in performance.

Introduction

Deep learning models have achieved great success in many visual tasks, such as object classification and detection. However, a wealth of research has shown that these models are not robust in situations where an attacker can modulate the input with only small perturbations, since such data modulation can significantly alter the model predictions (Szegedy et al. 2013; Biggio et al. 2013; Brown et al. 2017).

Multiple mechanisms have been proposed to defend against perturbation attacks. Some methods rely on empirical techniques, such as training with adversarial examples (Goodfellow, Shlens, and Szegedy 2014; Tramèr et al. 2017). This defense is straightforward to apply but is limited to the attack types seen in the training phase. Subsequent studies showed this defense to be vulnerable to carefully designed adversarial attacks (Carlini and Wagner 2017; Athalye and Carlini 2018). Other methods such as certifiable defenses seek to protect the classifier by verifying its prediction is within a prescribed ℓ_p ball around the input (Raghunathan, Steinhardt, and Liang 2018; Chiang et al. 2020; Cohen, Rosenfeld, and Kolter 2019). However, these methods are developed specifically for ℓ_p-norm attacks; furthermore, their computational expense and the architectural limitations they impose limit their practical application.

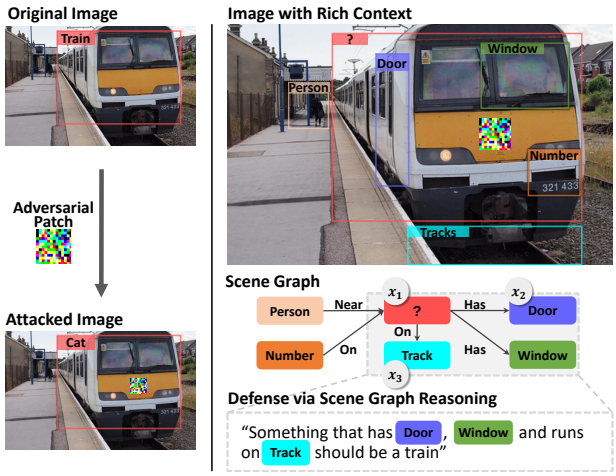

Figure 1: Defending the adversarial patch by reasoning over the scene graph.

Despite their sophistication, we note that most of these attacks can be easily detected by humans. In contrast to machine learning models, humans recognize and describe objects using a component hierarchy and by correlating co-occurring ancillary scene features, properties, and entities. This association with relevant entities suggests it as a natural defense mechanism against such attacks. For example, consider an image of a train as shown in Figure 1. The "train" object in the image is classified as a "cat" by the deep object classifier due to the adversarial attack. How would a human detect the object is misclassified? One can say that "a cat is an animal with paws and a tail" and these details are absent from the image. Furthermore, the bounded object has components "door" and "window", and is above "tracks", and these relationships are more likely associated with "train".

In this case, one detects the attack by reasoning over the relations of all the scene objects using commonsense knowl-

edge. This process is beneficial for two reasons: first, commonsense reasoning is general and does not rely on visual features, which makes it robust against arbitrary attacks on the visual domain when contextual information is preserved. Secondly, in contrast to other defenses, the reasoning process is self-explanatory (As shown in Figure 1) and can be easily understood by human users.

Inspired by this observation, we raise a challenging question: "*Can we propose a method that utilizes human-like reasoning to defend against the adversarial examples?*"

To this end, we propose *logic adversarial defense* (LOGICDEF), an interpretable defense framework for the adversarial examples. Our framework considers the scenario where, apart from the attacked object, it can access the rich context of the image in the form of a *scene graph* which represents the visual objects and their relations as nodes and edges of a directed graph. Given a scene graph, LOGICDEF mines commonsense knowledge that is predictive to the objects and represents them into logic rules through *inductive logic programming* (ILP). These rules are interpretable and serve as the framework's belief on how objects are associated in the real world. We then construct a defense model that integrates the rule knowledge as constraints into the deep classifier in a principled manner, i.e. *posterior regularization* (PR). Additionally, as the knowledge is stored in symbolic form, LOGICDEF can incorporate existing commonsense knowledge base such as ConceptNet[1] (Speer, Chin, and Havasi 2017) for zero-shot defense. It can also utilize the natural object taxonomy for curriculum learning to achieve improved performance in defenses with fewer labeled data.

In summary, our contributions are as follows:

- We propose LOGICDEF, an interpretable defense framework that performs inductive reasoning over the scene graph to defend the adversarial examples.
- LOGICDEF relies on commonsense knowledge that is mined from the data or imported from ConceptNet, making it highly interpretable and robust against the attacks in the visual domain.
- LOGICDEF is tested against the adversarial patch attack on the Visual Genome dataset (Krishna et al. 2016). The defended model achieves 97% of the accuracy obtainable when no attack is present and can achieve 86% accuracy with zero-shot defense. We show that LOGICDEF can incorporate knowledge and taxonomy from ConceptNet to obtain additional improvements with curriculum learning.

Related Work

Adversarial examples Adversarial example methods generate additive perturbations to the input to fool the deep models. Some methods generate global perturbations over the whole image (Szegedy et al. 2013; Goodfellow, Shlens, and Szegedy 2014; Kurakin et al. 2016). However, perturbing the whole image is usually impractical for physically realizable attacks (Chiang et al. 2020; Liu et al. 2018). Recent methods focus on generating localized perturbations that change a small part of the image. Adversarial patch (Brown et al. 2017), for example, learns to generate an image patch that can fool the classifier when put next to the visual object.

Note that, while we use the scene graphs to identify multiple image objects and their relations, we do not seek to defend against object detection attacks where the global structures (i.e. scene graphs) change significantly, as in Dense Adversary Generation (Xie et al. 2017) or DPatch (Liu et al. 2018). We instead aim to defend against adversarial patch attacks on an object classifier, that is, the attack targets the classification of a single object in the scene (e.g. the "train" in Figure 1). As the main motivation of this work is to explore how human-like reasoning could lead to interpretable defense, we treat the scene graph as background context that can be influenced by the attack but is not adversarially targeted.

Adversarial defenses Many defense methods have been proposed to mitigate adversarial attacks. *Adversarial training* (Goodfellow, Shlens, and Szegedy 2014; Tramèr et al. 2017) achieves robustness by building adversarial attacks into the training phase. On the other hand, *certified defense* methods (Raghunathan, Steinhardt, and Liang 2018; Chiang et al. 2020) study the principled defenses by verifying the model's prediction is within a bounded ℓ_p distance from the input. Such certified methods are computationally expensive and do not usually scale to large vision models, although the recent work of *random smoothing* (Cohen, Rosenfeld, and Kolter 2019) does so within the limitations of ℓ_2 norm attacks.

Our framework is different from most of the existing defense methods, as LOGICDEF achieves robustness through human-like commonsense reasoning. Some work shares the same motivation as our framework. For example, the *attacks meet interpretability* (Tao et al. 2018) technique detects attacks on face recognition by extracting face attributes that are perceptible by humans. The LOGICDEF approach differs in that it mines inductive logic rules from explicit scene graph data to detect attacks across arbitrary classes.

Inductive logic programming (ILP) LOGICDEF performs inductive reasoning on the scene graph to learn logic rules for adversarial defenses. This process is referred to as *Inductive logic programming* (ILP) and is studied extensively in the literature of inductive reasoning (Guu, Miller, and Liang 2015; Lao and Cohen 2010; Das et al. 2016) and formal logic. Classical ILP methods (Lavrac and Dzeroski 1994; Galárraga et al. 2015) rely on search-based techniques and are not tolerant to typical scene graph labeling errors. In this work, we utilize a differentiable ILP method for rule mining. Such methods fall into two categories: forward-chaining methods (Evans and Grefenstette 2018; Payani and Fekri 2019; Campero et al. 2018) propose rules from templates and verify them through repetitive deductive reasoning. Backward-chaining methods (Yang and Song 2020; Yang, Yang, and Cohen 2017) construct the rules on the fly by searching for the patterns that answer a given query. LOGICDEF uses backward-chaining due to its scalability.

Reasoning on graphs Defending the attacks with a scene graph can be also viewed as a *graph completion* problem which seeks to infer missing components, such as attributes or links in a given graph. This problem can be addressed

[1] https://conceptnet.io/

with a variety of approaches including embedding-based methods (Bordes et al. 2013; Sun et al. 2019), graph neural networks (Yang et al. 2018; Zhang et al. 2020), or RNN architectures (Zellers et al. 2018). Compared to LOGICDEF, these methods can solve the problem directly, although using black-box models that do not provide human-readable explanations as to the decisions are made. Therefore, it is difficult to use general graph completion models for defenses due to this lack of interpretability. Similarly, these models are also not capable of utilizing prior knowledge such as ConceptNet.

Problem Setup

Attacks in context-rich environment In this work, we aim to create an environment where the model is tasked to defend the attack by reasoning over relevant objects and their relations in the scene.

Specifically, we consider the attack scenario where: 1) the scene contains multiple objects; 2) an object classifier infers the class label for each object, and 3) a localized attack such as the adversarial patch (Brown et al. 2017) is applied to one object in the scene to fool the classifier. We refer to this scenario as *attacks in context-rich environment*.

Formally, let I denotes the image and $\mathcal{X} = \{x_1, x_2, ...\}$ be the set of objects in the image. Let I_x be the localized region that contains the object x. We define the object classifier as $f : \mathcal{X} \mapsto \mathbb{C}$ where $\mathbb{C}$ denotes the set of all class labels. It takes in the image region I_x and predicts its class label as $C = f(I_x; \theta) \in \mathbb{C}$, where θ represents the trained deep model parameter set. Let $A(\delta, I_x, l)$ be the "apply" function which adds perturbation δ to I_x at location l, we define the patch attack on the object x in the scene as

$$\hat{\delta} = \arg\min_{\delta} \mathbb{E}_{I\sim\mathbb{I}, l\sim\mathbb{L}_x} \mathbb{1}[f(A(\delta, I_x, l); \theta) = C], \quad (1)$$

where C is the ground-truth class label of x, and $\mathbb{1}[\cdot]$ is the indicator function that returns 1 if the equation inside is true. And $\mathbb{I}$ and $\mathbb{L}_x$ denote the set of all images and the set of all locations within the region of I_x. Note that Eq.(1) is adopted from adversarial patch formulation (Brown et al. 2017). However, it is different that the attack is optimized with respect to a single object x instead of the whole image I (e.g. the "train" in Figure 1).

Nevertheless, the $\hat{\delta}$ is still applied to the original image, as $I_x \subseteq I$. As a result, the perturbation will partially alter the scene information. But since $\hat{\delta}$ is local, most of the contextual information is retained. We will discuss its impact on the scene graph in the next section.

Note that this scenario not only provides a rich context for defenses but is also practical in real-world applications, as it resembles the physical-world attacks (Chen et al. 2018; Eykholt et al. 2018) in many aspects except that the perturbations are added in a post hoc manner.

Defense via inductive scene graph reasoning Our framework uses the contextual information in the form a scene graph to construct the defense model. Formally, a scene graph is represented as a collection of relations and attributes of the objects $\mathcal{G} = \{\mathcal{X}, \mathcal{T}, \mathcal{C}\}$. A triple $P(x, x') \in \mathcal{T}$ indicates that a pair of entities $x, x' \in \mathcal{X}$ are connected by relation (or binary predicate) P. For example, `Has`, `Near` are binary predicates. Similarly, we view class label, denoted as $C(x) \in \mathcal{C}$, as the object's attribute, and $C \in \mathbb{C}$ is an object class (or unary predicate) such as `Train` and `Person`.

Because LOGICDEF operates without prior knowledge of any scene ground truth, we assume that the scene graph is obtained after the attack is applied to the image. We denote this potentially imperfect scene graph estimate as $\hat{\mathcal{G}}$. It is collected in two steps: 1) the class labels are obtained from the classifier $\mathcal{C} = \{f(I_x; \theta) | x \in \mathcal{X}\}$ and, therefore, contain misclassified label; 2) relations $\mathcal{T}$ are obtained either from a graph generator such as Graph R-CNN (GRCNN) (Yang et al. 2018) or from ground-truth labels such as those provided in Visual Genome dataset. Formally, we define the scene graph-based defense as follows:

Def 1 (Defense via inductive scene graph reasoning) *Given the object classifier f, find an interpretable defense model $\mathbf{M}_{def} : \mathcal{X} \mapsto \mathbb{C}$ such that the following holds*

$$\mathbf{M}_{def}(I_x + \hat{\delta}; \hat{\mathcal{G}}, \mathcal{R}) = f(I_x; \theta)$$

for all images $I \in \mathbb{I}$ and objects $x \in \mathcal{X}$, where $I_x + \hat{\delta}$ denotes the perturbed image region of object x, and $\mathcal{R}$ denotes a set of inductive logic rules.

Proposed Method

To construct the defense model $\mathbf{M}_{\text{def}}$, one needs to address three challenges: 1) **Scene graph generation**: given the perturbed image $I + \hat{\delta}$ and class labels $\mathcal{C}$, how to generate the scene graph $\hat{\mathcal{G}}$ over which one can reason with commonsense? 2) **Rule mining and defense**: how to collect a set of logic rules $\mathcal{R}$? And how to use them for defense? 3) **Knowledge integration**: how to incorporate the rule knowledge into the classifier f in a principled manner such that the inference is robust against attack? We discuss each item in more detail.

Scene Graph Generation

In the context-rich environment, we assume the object bounding boxes are given together with the image I, that is, the object detection $\mathcal{X}$ is ground-truth. The classifier then infers the class labels $\mathcal{C}$ on the perturbed image $I + \hat{\delta}$. Since $\hat{\delta}$ is generated with respect to one object in the scene, one label in $\mathcal{C}$ is corrupted intentionally. This can also lead to misclassification on a few other objects if their bounding boxes overlap. Apart from the class labels, a complete scene graph $\hat{\mathcal{G}}$ should also include object relations $\mathcal{T}$. We propose to collect them in two ways.

Ground-truth graphs We collect the ground-truth graphs from the Visual Genome (VG) dataset to get relations $\mathcal{T}$. The original VG is noisy and contains labeling errors (Zellers et al. 2018). We follow the same protocol as in (Yang et al. 2018) to pre-process the data, keeping the top 50 relation types.

Graph generator We also consider a more practical scenario where no ground-truth relations are provided. In this case, we obtain the relations with a pre-trained scene graph

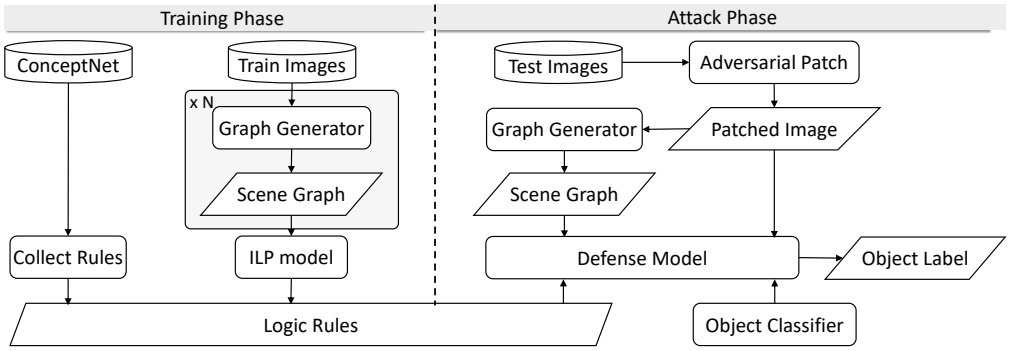

Figure 2: Overview of the LOGICDEF model construction and the evaluation framework.

generator, that is Graph R-CNN (GRCNN) (Yang et al. 2018). This setting is more challenging, as there are potentially more errors in relations compared to the ground truth. In this way, we evaluate the robustness of LOGICDEF as it needs to perform correct inference in the presence of partial errors in the scene graph developed from the image under evaluation.

Rule Mining and Defenses

First-order logic Our framework utilizes the framework of restricted first-order logic (FOL) to represent the defense. A FOL system consists of (i) a set of predicates, (ii) a set of logical variables, and (iii) logical operations $\{\wedge, \vee, \neg\}$. For example,

$$\texttt{Train}(X) \leftarrow \texttt{Has}(X, Y) \wedge \texttt{Door}(Y) \wedge \texttt{On}(X, Z) \wedge \texttt{Tracks}(Z) \quad (2)$$

involves unary predicates (or class label) `Train`, `Door` and `Tracks`, and binary predicates (or relations) `Has` and `On`. Here, X, Y and Z are logical variables and "$\leftarrow$" denotes the *logic implication* where $p \leftarrow q$ is equivalent to $p \vee \neg q$. Components such as $\texttt{Has}(X, Y)$ are called *atoms*. One can view atoms as boolean functions that take logical variables as inputs. To evaluate an atom, one *instantiates* the logical variables into the specific objects. For example, let $\mathcal{X} = \{x_1, x_2, x_3, ...\}$ as shown in Figure 1, we evaluate $\texttt{Has}(x_1/X, x_2/Y)$ by instantiating X and Y into x_1 and x_2 respectively. This yields `True` because "x_1 has x_2 as its component" is `True`. A logic rule such as Eq.(2) consists of a set of atoms and can be evaluated in a similar manner. One instantiates all variables in the rule, obtains the binary outputs of all atoms, then executes the prescribed logical operations to get the final output.

By using the logical variables, the rule encodes the *inductive knowledge* as it does not depend on the specific data. Such representations are interpretable by human users.

Logic rules for adversarial defense For defense purpose, we aim to learn logic rules similar to Eq.(2) which entails the existence of a `Train` by checking if it has `Door` and is on `Tracks`. Formally, we define *logic entailment rules* by

$$R(X) : C(X) \leftarrow P_1(X, Y_1) \wedge P_2(Y_1, Y_2) ... \wedge C_n(Z). \quad (3)$$

Logic rules with this form encodes the knowledge that "X belongs to class C if and only if the condition $P_1(X, Y_1) \wedge P_2(Y_1, Y_2) ... \wedge C_n(Z)$ is `True`". Note that Z is a free variable and can be instantiated into any class instance in the graph (e.g. $\texttt{Tracks}(z/Z)$), which we refer to as the *anchor object*. In other words, $R : \mathcal{X} \mapsto \{0, 1\}$ can be seen as a binary classifier that infers if an object x/X belongs to class C by verifying whether the condition holds in the given scene graph. Therefore, if we collect a set of such rules $\mathcal{R}$ for all classes, we can apply them to the scene graph $\hat{\mathcal{G}}$ and use their inference results to detect the misclassified labels in $\mathcal{C}$.

Mining rules with ILP The defense rule set $\mathcal{R}$ is collected in two ways: rule mining and human-generated prior knowledge. For rule mining, we apply backward-chaining inductive logic programming (ILP) (Yang and Song 2020; Yang, Yang, and Cohen 2017) to mine the inductive patterns in scene graphs and represent them into logic entailment rules.

Formally, a backward-chaining ILP model is defined as $\mathbf{M}_{\text{ILP}} : \mathcal{X} \mapsto \Omega$, where $R \in \Omega$ denotes the space of rules in Eq.(3). Given a scene graph $\mathcal{G}$ and a *query* $x \in \mathcal{X}$, the ILP model searches for the rule in $\mathcal{G}$ that answers the query $R \leftarrow \mathbf{M}_{\text{ILP}}(x; \mathcal{G})$ such that the query score is maximized:

$$\text{score}(x; R) = \boldsymbol{x}^\top \boldsymbol{M}_1 \cdot \boldsymbol{M}_2 ... \cdot \boldsymbol{M}_{n-1} \cdot \boldsymbol{z},$$

where $\boldsymbol{x}, \boldsymbol{z} \in \{0, 1\}^{|\mathcal{X}|}$ are the one-hot vectors of query object x and anchor object z, and $\boldsymbol{M}_1, ..., \boldsymbol{M}_{n-1} \in \{0, 1\}^{|\mathcal{X}| \times |\mathcal{X}|}$ are the adjacency matrices of relations $P_1, ..., P_{n-1}$ of logic rule R (see derivations in Appendix).

Mining such rule can be formulated as a learning problem with the *differentiable ILP* paradigm, where the hard search is parameterized by attention mechanisms in differentiable models. Let $\mathcal{D}_{\text{train}} = \{\langle I_i, \mathcal{G}_i \rangle\}_{i=1}^N$ be the set of images and scene graphs provided in the training phase (as shown in Figure 2). In this phase, we learn a differentiable ILP model such that

$$\max_{\omega} \sum_{\mathcal{G} \in \mathcal{D}_{train}} \sum_{x \in \mathcal{X}} \text{score}(x; \mathbf{M}_{\text{ILP}}(x; \omega, \mathcal{G})), \quad (4)$$

where ω denotes the parameters of the ILP model. After training, we collect all rules learned from $\mathcal{D}_{train}$ as the defense rule set $\mathcal{R} = \{\mathbf{M}_{\text{ILP}}(x; \omega, \mathcal{G}) | x \in \mathcal{X}, \mathcal{G} \in \mathcal{D}_{\text{train}}\}$.

Rule confidence As a simple binary classifier, a logic rule can be inaccurate and ambiguous. We want to quantify the quality of a logic rule such that our defense model favors

the high-quality ones. As such, we follow the common practice and use the rule *precision* (See definition in Appendix Eq.(12)) as its confidence score which we denote as λ.

Commonsense knowledge and curriculum learning LOGICDEF is capable of incorporating prior knowledge from existing sources like ConceptNet (Speer, Chin, and Havasi 2017) which is a semantic network that contains crowdsourced commonsense knowledge. Objects in ConceptNet are represented with a set of attributes and their relations to other objects.

ConceptNet contains numerous relation types. In this work, we focus on the `HasA` and `IsA` relations. The `HasA` associates objects with their natural properties such as components. For example, `Cat` $\xrightarrow{\texttt{HasA}}$ `Tail`. We collect all the $\texttt{HasA}(X, Y)$ triples in the ConceptNet that involves entities in our class vocabulary $\mathbb{C}$. We convert each triple into a pair of symmetric entailment rules

$$C_1(X) \leftarrow \texttt{HasA}(X, Y) \wedge C_2(Y),$$
$$C_2(Y) \leftarrow \texttt{PartOf}(Y, X) \wedge C_1(X),$$

where $C_1, C_2 \in \mathbb{C}$ are class labels of objects X and Y. We obtain total of 297 rules from ConceptNet, examples shown in Appen. Table 3. In experiments, we show that this rule set is powerful by itself: our defense model achieves 86% of the best performance by solely relying on the ConceptNet rules.

On the other hand, `IsA` associates the objects with their taxonomic superclasses, for example `Cat` $\xrightarrow{\texttt{IsA}}$ `Animal` and `Train` $\xrightarrow{\texttt{IsA}}$ `Vehicle`. We collect objects in $\mathbb{C}$ that are linked by this relation. We organize them into a taxonomy tree: we create an abstract class `Object` as the root of the tree. We put most of the classes as leave nodes and some classes such as `Person` as branch nodes as they subsume other classes such as `Boy`. This tree (see Appendix Figure 4) provides a natural curriculum for learning the logic rules for the objects: the rules learned for the superclass can be inherited into its subclasses. In the experiments, we demonstrate that this taxonomy-based curriculum learning helps LOGICDEF to achieve better performance with fewer data.

Knowledge Integration

There are many ways the rule set $\mathcal{R}$ can be used to infer the class label of an object in the scene graph. In LOGICDEF, we need to construct the defense model $\mathbf{M}_{\text{def}}$ that incorporates the rule knowledge into the object classifier in a principled manner. To this end, we propose to utilize the *posterior regularization* (PR) (Ganchev et al. 2010) technique which converts rule knowledge into constraints on the posterior distributions of the classifier model.

Formally, let $\mathbf{M}_{\text{def}}(I_x; \mathcal{R})$ be the defense model we aim to construct and use for the final robust prediction (we omit $\hat{\delta}$ and $\hat{\mathcal{G}}$ for notational simplicity). Through PR, we obtain the model by solving the following convex optimization problem

$$\min_{\mathbf{M}, \xi_1 \ldots \xi_{|\mathcal{R}|}} \text{KL}\left(\mathbf{M}_{\text{def}}(I_x; \mathcal{R}) \,\|\, f(I_x; \theta)\right) + \pi \sum_{j=1}^{|\mathcal{R}|} \xi_j, \quad (5)$$

$$s.t. - \lambda_j \mathbb{E}_x\left[R_j(x)\right] \leq \xi_j,\ j = 1, ..., |\mathcal{R}|. \quad (6)$$

Here, $\xi_1, ..., \xi_{|\mathcal{R}|}$ denotes the slack variables and π is the regularization coefficient. The inequality constraint of Eq.(6) specifies that, for each rule R_j, its negative weighted expectation should be less than the variable ξ_j. And the optimization of Eq.(5) specifies that the defense model $\mathbf{M}_{\text{def}}$ should stay close to the classifier f such that the inference is faithful to the original output. This problem is solved analytically

$$\mathbf{M}_{\text{def}}(I_x; \mathcal{R}) \propto f(I_x; \theta) \exp\left(\pi \sum_{j=1}^{|\mathcal{R}|} \lambda_j R_j(x)\right). \quad (7)$$

We note that a logic rule like Eq.(3) is a binary classifier associated with a single class. In Eq.(7), we want to aggregate the results of all classes into a probabilistic vector. To do this, we parameterize the output of a rule into a one-hot vector $\boldsymbol{y} \in \mathbb{R}^{|\mathcal{C}|}$, that is if the head predicate is C_i then its i-th element y_i is either 1 or 0 and the rest are always 0. In this way, the output $\bar{\boldsymbol{y}} \in \mathbb{R}^{|\mathcal{C}|}$ is the product of $f(I_x; \theta)$ and the exponential of the sum of one-hot vectors from $\mathcal{R}$ weighted by their confidence scores. This PR integration gives a principled way to balance between rule constraints and faithfulness and has been proven effective in similar applications with logic rules (Guo et al. 2018; Hu et al. 2016).

Candidate filtering In Eq.(7), aggregating inference results from all classes can lead to decreased accuracy. This is because most of the scene graph data available in the literature are sparse and not exhaustive. For example, one can identify both `Plane` and `Bird` by checking if the object has `Tail` and `Wing`. However, to distinguish between these two classes, one needs more evidence such as the existence of an `Engine`. And if the evidence is absent (which is true in many scene graphs), it can lead to false positives.

To address this issue, we note that, in practice, the adversarial attacks are typically bounded for how much change they can impose on the image. For example, a ℓ_p-norm attack bounds the perturbation to be within the ℓ_p ball of the input. This means the perturbed object is close to the original one in the latent space of the object classifier. Therefore, one can filter out irrelevant classes by measuring the distances from the perturbed object to the centroids of each class in the latent space. The intuition is that the visual latent space is different from that of the scene graph, such that objects that are confusing in the scene graph (e.g. `Plane` and `Bird`) are usually distinct in the visual domain.

Specifically, for each class C, we collect embeddings of the training samples from the last convolution layer of the object classifier. We compute and store the class mean $\boldsymbol{u}_C$ and covariance matrix $\boldsymbol{S}_C$. During the attack phase, given the embedding of the perturbed object $\boldsymbol{h}$, we compute the Mahalanobis distance between $\boldsymbol{h}$ and all classes as $d_C = \sqrt{(\boldsymbol{h} - \boldsymbol{u}_C)^\top \boldsymbol{S}_C^{-1} (\boldsymbol{h} - \boldsymbol{u}_C)}, C \in \mathbb{C}$. We rank the classes in increasing order and keep only the top-K closest classes for Eq.(7) inference.

Training Figure 2 illustrates the overview of LOGICDEF. The training process is summarized in Appendix Algorithm 1.

Table 1: Defending against adversarial patch attacks on VG dataset with Adv-Train and LOGICDEF methods. The object classifier has the best performance when defended by LOGICDEF-HYB, achieving 97% of the accuracy obtainable when no adversarial attack is present (clean accuracy). On the other hand, LOGICDEF-CN achieves 86% accuracy as a zero-shot defense.

Classes	Accuracy of ResNet-101 classifier							
		Imp. with GRCNN Scene Graph			Imp. with GT Scene Graph			
	Adv Train	LOGICDEF CN	LOGICDEF AUTO	LOGICDEF HYB	LOGICDEF CN	LOGICDEF AUTO	LOGICDEF HYB	Clean Accuracy
Bed	0.362	0.342	0.341	0.341	0.398	0.465	**0.465**	0.466
Building	0.061	0.730	0.650	0.694	0.699	0.803	**0.804**	0.639
Cat	0.271	0.372	0.556	0.571	0.495	0.592	**0.593**	0.772
Man	0.056	0.442	0.357	0.408	0.472	**0.606**	**0.606**	0.638
Sign	0.138	0.711	0.705	0.711	0.708	0.713	**0.714**	0.721
Tree	0.002	0.616	0.525	0.614	0.728	0.708	**0.731**	0.601
Overall	0.152	0.295	0.314	0.316	0.313	0.336	**0.349**	0.361

Experiments

Dataset We evaluate LOGICDEF on the Visual Genome (VG) dataset (Krishna et al. 2016). Compared to common benchmarks such as MNIST and CIFAR-10, whose images typically contain only a single labeled object, VG contains around 100K images annotated with scene graphs, providing the rich context required for evaluating models targeted towards real-world application. We use the pre-processing module provided in (Yang et al. 2018) to generate and split the data. The resulting dataset contains 56K images in the training set $\mathcal{D}_{\text{train}}$ and 26K images in the test set $\mathcal{D}_{\text{test}}$ for 150 object classes and 50 relations. We evaluate LOGICDEF in two settings: **GRCNN Scene Graph** where relations are generated by GRCNN, and **GT Scene Graph** with given ground-truth relations.

Models For the object classifier f, we use the pre-trained ResNet-101 provided in (Yang et al. 2018). We freeze the weights of convolutional layers and only finetune the fully connected layer. For each image $I \in \mathcal{D}_{\text{train}}$, we collect I_x for all $x \in \mathcal{X}$ and resize them to 224×224 patches and then feed them to f for training. For the ILP model, we train NLIL (Yang and Song 2020) on scene graphs in $\mathcal{D}_{\text{train}}$ and store the learned rules as rule set $\mathcal{R}$. The scene graph generator is the pre-trained GRCNN model (Yang et al. 2018).

We evaluate LOGICDEF in 3 modes: **CN**, **AUTO** and **HYB**. Specifically, LOGICDEF-**CN** only uses logic rules extracted from ConceptNet, making it a zero-shot mode as it does not involve any training; LOGICDEF-**AUTO** uses $\mathbf{M}_{\text{ILP}}$ to mine logic rules from the VG scene graphs; and LOGICDEF-**HYB** is the hybrid mode which combines the rule sets from the previous two modes for better performance.

Attack, defense, and evaluation We test LOGICDEF with the adversarial patch attack. For each I_x in the test set, we generate one attack instance by applying a square patch with a maximum patch size fraction of 0.2 (i.e. 20% of the size of an I_x). Since this work focuses on defenses, the patch does not need to be transferable. We generate a patch for each I_x with 50 iterations individually. We also compare LOGICDEF with adversarial training (Adv-Train) (Tramèr et al. 2017; Madry et al. 2017), a defense method that increases robustness by injecting adversarial samples during training. For adversarial patch and Adv-Train, we use the implementations provided here[2]. For hyper-parameters, we run grid search over a held-out validation set, where we set $\pi = 20$ and $K = 15$. Computations were done on a desktop computer with an Intel i7-8700K CPU and a GTX1080Ti.

Results

Table 1 shows the overall accuracy of the object classifier together with class-wise accuracy of 6 exemplar classes. The ResNet-101 classifier achieves clean accuracy of 0.361, that is the model accuracy in the absence of the attack. This classification task is significantly harder than benchmarks such as MNIST and CIFAR, due to both the bounding box and labeling errors in VG (Zellers et al. 2018) and the 150-way classification. As a reference, the reported mean average precision at 0.5 intersections over union (mAP@0.5) of object detection on VG is 24.8 (Yang et al. 2018; Xu et al. 2017).

The classifier achieves the best accuracy with LOGICDEF-HYB on the GT scene graph which is 97% of the clean accuracy. Adv-Train method achieves 42% of the clean accuracy. This is because the adversarial patch is inherently difficult to defend with only visual features and Adv-Train is shown to be sensitive to the randomness of the adversaries (Carlini et al. 2019).

GRCNN vs GT Scene Graph Scene graphs generated with GRCNN have a higher error rate than the GT scene graphs. As a reference, the reported recall@50 for relation generation of GRCNN is 58.1. However, LOGICDEF is insensitive to the noises in the relations. For all 3 modes, our framework maintains at least 90% of the accuracy as those with GT scene graphs.

Zero-shot performance We also note that LOGICDEF-CN is the training-free version that only relies on commonsense knowledge extracted from ConceptNet. Yet it achieves 86%

[2]https://adversarial-robustness-toolbox.readthedocs.io/en/latest/index.html

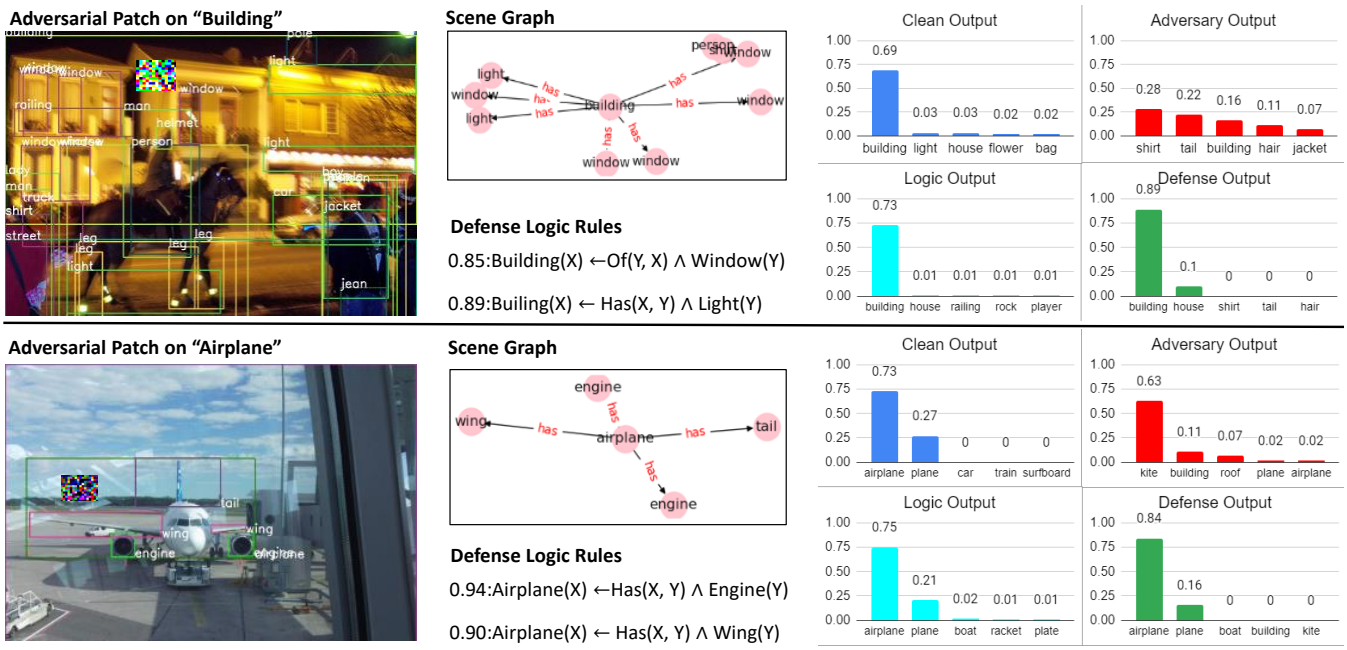

Figure 3: Examples of defending the adversarial patch attack with LOGICDEF-AUTO. Attacks are detected with interpretable logic rules and the decisions are self-explanatory.

of the clean accuracy. This suggests that human-generated prior knowledge is a powerful data source.

Interpreting the defense model One of the main advantages of LOGICDEF is its interpretability. We show defense examples by LOGICDEF-AUTO in Figure 3. The left column shows the patched images and the objects that are attacked. The middle column shows the scene graphs and the logic rules that are used for defenses. For the right column, the first row shows the probability output of the classifier on a clean and patched image respectively and the second row shows the probability output of the logic rules and the final defense model, respectively. We show more mined rules in Appendix Table 4.

Curriculum Learning

The taxonomy tree (Appendix Figure 4) we extracted from the ConceptNet provides a natural curriculum of learning objects. We conduct top-down learning of $\mathbf{M}_{\text{ILP}}$ on the object classes: for each branch node (e.g. `Animal`), the model learns from its own training data and also 10% randomly sampled data from its subclasses. When learning the leave nodes (e.g. `Cat`), the model is initialized with the weights of its superclass model. The intuition is that the rules learned for the superclasses should also transfer to the subclasses. To validate this, we test LOGICDEF by training it on the subset of $\mathcal{D}_{\text{train}}$, where 50, 100, and 500 random samples are provided for the leaf nodes. The results (see Table 2) suggest that curriculum learning leads to better performance with fewer data.

Table 2: LOGICDEF-AUTO-CL achieves better accuracy with taxonomy-based curriculum learning.

K-shots	GRCNN Scene Graph		GT Scene Graph	
	LOGICDEF AUTO	LOGICDEF AUTO-CL	LOGICDEF CN	LOGICDEF AUTO-CL
50	0.195	**0.231**	0.257	**0.289**
100	0.212	**0.234**	0.264	**0.303**
500	0.248	**0.254**	0.315	**0.319**

Ablation Study

In the ablation study, we investigate the effect of PR and candidate filtering. For PR, we compare LOGICDEF-AUTO with a version where Eq.(7) is replaced with plain summation. The results (Appendix Table 5) suggest that PR leads to 30% better accuracy and 50% lower false positive rate. For candidate filtering, we evaluate LOGICDEF with different K values from 1 to 150. The results (Appendix Figure 5) suggest that a small K indeed increases model accuracy and the best range is 10∼15. See detailed discussion in Appendix.

Conclusion

We propose LOGICDEF, an interpretable framework that defends the adversarial patch attack via inductive scene graph reasoning. The framework mines logic rules about objects from scene graphs and the commonsense knowledge from ConceptNet, and uses them to construct a defense model with posterior regularization. Experiments show that LOGICDEF is highly interpretable and can achieve 94% of the clean accuracy in hybrid mode and 86% in zero-shot mode.

References

Athalye, A.; and Carlini, N. 2018. On the robustness of the cvpr 2018 white-box adversarial example defenses. *arXiv preprint arXiv:1804.03286*.

Biggio, B.; Corona, I.; Maiorca, D.; Nelson, B.; Šrndić, N.; Laskov, P.; Giacinto, G.; and Roli, F. 2013. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, 387–402. Springer.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, 2787–2795.

Brown, T. B.; Mané, D.; Roy, A.; Abadi, M.; and Gilmer, J. 2017. Adversarial patch. *arXiv preprint arXiv:1712.09665*.

Campero, A.; Pareja, A.; Klinger, T.; Tenenbaum, J.; and Riedel, S. 2018. Logical Rule Induction and Theory Learning Using Neural Theorem Proving. *arXiv preprint arXiv:1809.02193*.

Carlini, N.; Athalye, A.; Papernot, N.; Brendel, W.; Rauber, J.; Tsipras, D.; Goodfellow, I.; Madry, A.; and Kurakin, A. 2019. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*.

Carlini, N.; and Wagner, D. 2017. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, 3–14.

Chen, S.-T.; Cornelius, C.; Martin, J.; and Chau, D. H. P. 2018. Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 52–68. Springer.

Chiang, P.-y.; Ni, R.; Abdelkader, A.; Zhu, C.; Studor, C.; and Goldstein, T. 2020. Certified defenses for adversarial patches. *arXiv preprint arXiv:2003.06693*.

Cohen, J.; Rosenfeld, E.; and Kolter, Z. 2019. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, 1310–1320. PMLR.

Das, R.; Neelakantan, A.; Belanger, D.; and McCallum, A. 2016. Chains of reasoning over entities, relations, and text using recurrent neural networks. *arXiv preprint arXiv:1607.01426*.

Evans, R.; and Grefenstette, E. 2018. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 61: 1–64.

Eykholt, K.; Evtimov, I.; Fernandes, E.; Li, B.; Rahmati, A.; Xiao, C.; Prakash, A.; Kohno, T.; and Song, D. 2018. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1625–1634.

Galárraga, L.; Teflioudi, C.; Hose, K.; and Suchanek, F. M. 2015. Fast rule mining in ontological knowledge bases with AMIE+. *The VLDB Journal—The International Journal on Very Large Data Bases*, 24(6): 707–730.

Ganchev, K.; Graça, J.; Gillenwater, J.; and Taskar, B. 2010. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 11: 2001–2049.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Guo, S.; Wang, Q.; Wang, L.; Wang, B.; and Guo, L. 2018. Knowledge graph embedding with iterative guidance from soft rules. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Guu, K.; Miller, J.; and Liang, P. 2015. Traversing knowledge graphs in vector space. *arXiv preprint arXiv:1506.01094*.

Hu, Z.; Ma, X.; Liu, Z.; Hovy, E.; and Xing, E. 2016. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*.

Krishna, R.; Zhu, Y.; Groth, O.; Johnson, J.; Hata, K.; Kravitz, J.; Chen, S.; Kalantidis, Y.; Li, L.-J.; Shamma, D. A.; Bernstein, M.; and Fei-Fei, L. 2016. Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations.

Kurakin, A.; Goodfellow, I.; Bengio, S.; et al. 2016. Adversarial examples in the physical world.

Lao, N.; and Cohen, W. W. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1): 53–67.

Lavrac, N.; and Dzeroski, S. 1994. Inductive Logic Programming. In *WLP*, 146–160. Springer.

Liu, X.; Yang, H.; Liu, Z.; Song, L.; Li, H.; and Chen, Y. 2018. Dpatch: An adversarial patch attack on object detectors. *arXiv preprint arXiv:1806.02299*.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Payani, A.; and Fekri, F. 2019. Inductive Logic Programming via Differentiable Deep Neural Logic Networks. *arXiv preprint arXiv:1906.03523*.

Raghunathan, A.; Steinhardt, J.; and Liang, P. 2018. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*.

Speer, R.; Chin, J.; and Havasi, C. 2017. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge.

Sun, Z.; Deng, Z.-H.; Nie, J.-Y.; and Tang, J. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Tao, G.; Ma, S.; Liu, Y.; and Zhang, X. 2018. Attacks meet interpretability: Attribute-steered detection of adversarial samples. *arXiv preprint arXiv:1810.11580*.

Tramèr, F.; Kurakin, A.; Papernot, N.; Goodfellow, I.; Boneh, D.; and McDaniel, P. 2017. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*.

Xie, C.; Wang, J.; Zhang, Z.; Zhou, Y.; Xie, L.; and Yuille, A. 2017. Adversarial examples for semantic segmentation and object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 1369–1378.

Xu, D.; Zhu, Y.; Choy, C. B.; and Fei-Fei, L. 2017. Scene graph generation by iterative message passing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5410–5419.

Yang, F.; Yang, Z.; and Cohen, W. W. 2017. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems*, 2319–2328.

Yang, J.; Lu, J.; Lee, S.; Batra, D.; and Parikh, D. 2018. Graph r-cnn for scene graph generation. In *Proceedings of the European conference on computer vision (ECCV)*, 670–685.

Yang, Y.; and Song, L. 2020. Learn to Explain Efficiently via Neural Logic Inductive Learning. In *International Conference on Learning Representations*.

Zellers, R.; Yatskar, M.; Thomson, S.; and Choi, Y. 2018. Neural motifs: Scene graph parsing with global context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5831–5840.

Zhang, Y.; Chen, X.; Yang, Y.; Ramamurthy, A.; Li, B.; Qi, Y.; and Song, L. 2020. Efficient probabilistic logic reasoning with graph neural networks. *arXiv preprint arXiv:2001.11850*.