Unifying model explainability and robustness via reasoning labels

Vedant Nanda **Junaid Ali** **Krishna P. Gummadi**
Max Planck Institute for Software Systems (MPI-SWS)
{vnanda, junaid, gummadi}@mpi-sws.org

Muhammad Bilal Zafar
Bosch Center for Artificial Intelligence (BCAI)
muhammadbilal.zafar@de.bosch.com

Abstract

Explainability in deep learning has emerged as an important topic in recent years, with several works exploring various notions and mechanisms of explainability for deep neural networks (DNNs). In this paper, we draw upon the insight that in many situations model explainability is a means to assess another related yet distinct criterion - *model robustness*. In order to render the link between explainability and robustness more explicit, we propose to use human-understandable *reasoning labels* during the training process of DNNs. The reasoning labels are jointly learned with the traditional classification labels. This joint training enables the model to predict a set of reasoning labels with every predicted class label. Then, we tie model explainability and robustness by introducing a notion of prediction *consistency*, whereby the model predictions are accepted—or considered robust—only when the predicted class and the predicted reasoning labels follow a certain pre-specified mapping. We show that by adopting such a framework, one can improve the classification accuracy of the state-of-the-art models (on consistent samples). We further show that using this notion of consistency makes the model more robust to adversarial perturbations.

1 Introduction

State-of-the-art machine learning methods, such as deep neural networks (DNNs), achieve very good performance on a variety of real-world tasks. However, the predictions of these models are often very hard to explain. That is, it is often very challenging for humans to understand how a model arrived at a certain outcome.

A plethora of recent works have focused on explaining the predictions of deep neural networks [Ribeiro et al., 2016, Kim et al., 2018, Gulshad et al., 2019, Chen et al., 2018, Koh and Liang, 2017]. These methods use different frameworks of explainability. Under all of these frameworks, the key ingredient of what constitutes an explanation is the same: *to present the reasoning of the model in a human-understandable manner*.

For example, the explainability framework of Ribeiro et al. [2016] leverages the fact that, given an input whose prediction by a complex model - such as a DNN - needs to be explained, simpler (*e.g.*, linear) models can be used in the local neighborhood of the input to provide human-understandable reasoning. Influence-based methods like that of Koh and Liang [2017] provide this reasoning in the form of influence of training data points on a given prediction. Methods based on saliency [Simonyan et al., 2013] aim at identifying how much individual input features affect the model output, hence using feature influence as a notion of human-understandable reasoning. Finally, methods like those

Table 1: Our proposed explainability framework. Consider a hypothetical image classification task where the classifier predicts reasoning labels in addition to the class labels. The user is already in possession of a class label to reasoning label mapping that was used during model training. In this example, the mapping is panda $\rightarrow$ {animal, paws, spots} and bus $\rightarrow$ {tires, seats, engine}. Predictions # 1 and # 2 are accepted since the (predicted) class labels and (predicted) reasoning labels are consistent with each other according to the class-reasoning label mapping. However, prediction # 3 is rejected as the predicted reasoning labels and class labels are not consistent—the class label is 'panda' but the reasoning labels are (seats, engine).

Pred. #	**Pred. class labels**	**Pred. reasoning labels**					
	panda or bus	animal	paws	spots	tires	seats	engine
1	panda	✓	✓	✓	✗	✗	✗
2	bus	✗	✗	✗	✓	✓	✓
3	panda	✗	✗	✗	✗	✓	✓

of Chen et al. [2018] and Gulshad et al. [2019] force the model to first predict high-level human-understandable features (*e.g.*, wings, legs), and then make the final prediction that is a simple function (*e.g.*, weighted sum) of these high-level features.

In this work, we take the position that for a wide variety of prediction tasks, often the motivation behind obtaining model explanations is to *judge the reasoning of the model which in turn allows one to assess if the predictions are robust*, *i.e.*, to confirm that the model did not make an egregious error. Consider for instance a scenario in which a model predicts the object in an image to be a panda with the reasoning or explanation that the object contains tires and windows. The contrast between the prediction and reasoning suggests that the prediction is likely non-robust, and ought not to be trusted. In other words, model explainability is inherently tied to its robustness. Building on this insight, we propose a framework of explainability which *combines model explainability with prediction robustness*. Concretely, under our framework, in addition to predicting the class labels (*e.g.*, panda, bus, tiger), the model should also *present the human-understandable reasoning* it used to predict those class labels, where the reasoning construct is provided by the model trainer.

As an instantiation of our framework, we propose to incorporate the reasoning into the model by making it predict some *reasoning labels* in addition to the *class labels*. An example could be a model predicting the class label 'panda' and the corresponding reasoning labels 'is an animal', 'has paws', and 'has spots'.

The explanations are then tied to model robustness in the following manner: At the prediction time, the model users are provided with the mapping between class labels and reasoning labels that were used during the training time, *e.g.*, panda $\rightarrow$ {animal, paws, spots} and, bus $\rightarrow$ {tires, seats, engine}. If the classification and reasoning labels generated by the model are not consistent, that is, if the predicted reasoning labels do not match the reasoning labels of the predicted class, one can discard these predictions. For example, in Table 1, the prediction panda $\rightarrow$ {seats, engine} will be discarded since the reasoning labels for the class panda should instead be {animal, paws, spot}. On the other hand, consistent classification and reasoning labels provide the users with confidence in the correct reasoning, and hence, the robustness of the model.

The learning task then involves training the DNN by maximizing accuracy for the class labels as well as reasoning labels. Concretely, one would penalize the model if either of the predicted class or reasoning labels is incorrect.

Training models with reasoning labels also has some additional advantages: First, since our framework can be thought of as enriching the training dataset with external information that is not present in the feature vectors (*e.g.*, in an image classification task, the information panda $\rightarrow$ {animal}, and tiger $\rightarrow$ {animal}) training with reasoning labels would nudge the model to encode this extra information in the hidden representations. For example, both tigers and pandas will likely be close in the feature space since they are both animals. Stepping one level higher, animals will likely be closer to birds than they are to automobiles.

Additionally, using the reasoning labels to assess prediction robustness also leads to an interesting by-product: robustness against adversarial perturbations. For example, while an adversarial perturbation might change the class label of a given input, it may not result in a corresponding change in the

reasoning labels. Hence, by checking for consistency in the predicted class label and reasoning labels, one can detect adversarial perturbations. Changing (several) reasoning labels *in conjunction with* the class label to achieve consistency might prove to be a much harder task for an adversary.

Our main contributions are as follows:

- We present an **explainability framework** which ties model explainability with model robustness, whereby, the model is designed to predict class labels and reasoning labels simultaneously. An inconsistency between the predicted class and reasoning labels implies that the model predictions may not be robust.
- We show that for classification tasks involving named-entities, one can **automatically obtain reasoning labels** of high quality by using existing knowledge bases such as ConceptNet [ConceptNet], hence side-stepping the need for manual human annotation.
- We show empirically on CIFAR-10 dataset that using our explainability framework leads to **higher prediction accuracy** on samples where the classification and reasoning labels are consistent.
- We show on the CIFAR-10 dataset that our explainability framework also results in **improved adversarial robustness**. Specifically, we see that more than 95% percent of the adversarial inputs are successfully detected under our framework as a result of inconsistency between classification and reasoning labels.

2 Related work

As noted in Section 1, our main difference to prior approaches to explainability such as [Ribeiro et al., 2016, Chen et al., 2018, Gulshad et al., 2019] is that our framework ties model reasoning (or explainability) directly with model robustness, whereby, we reject samples whose predicted class label is inconsistent with the predicted reasoning labels. The approaches of Chen et al. [2018] and Gulshad et al. [2019] also aim at predicting class labels and some high-level 'reasoning' labels (called prototypes and attributes in two papers, respectively). However, our approach differs from them in two different ways: 1) Unlike us, these two approaches first predict the attributes/prototypes and then use these to predict class labels, and, 2) their frameworks do not include a reject option based on the (in)consistency between class labels and predicted attributes and prototypes.

Kim et al. [2017] proposes a method to evaluate how important a user-defined concept is in predicting a specific class, given a trained classification model. They take example data with and without a human-understandable concept (*e.g.*, stripes). Then, they measure the sensitivity of a class prediction (e.g, zebra) to that concept. Unlike our method, they try to provide an interpretation of an already trained model. We additionally provide an interpretable method to reject the prediction of our model based on the consistency between the predicted class and the predicted reasoning labels.

Some prior work tries to identify prototypical examples which were most important for a given classification model to make its predictions [Koh and Liang, 2017, Khanna et al., 2018]. The intuition is that if we are provided with these representative examples of the model prediction, it will be more interpretable. Kim et al. [2016], additionally, try to identify examples in the feature space where prototypical examples do not provide a good explanation, which they call criticism examples. Bien and Tibshirani [2011] aims to make complex data distributions more interpretable by identifying prototypical data points in the training data.

A recent study by Tao et al. [2018] also proposes to use salient image features and inconsistency between two classification models to detect adversarial examples. However, our framework differs from them in the following ways: 1) Their approach is only limited to face recognition tasks and involves handcrafting of salient facial features, on the other hand, our framework can be extended to other (more general) image recognition tasks and other classification tasks (*e.g.*, text classification, loan approval), 2) they propose to train two models in parallel, both of which aim to predict the class label, whereas our framework aims at predicting the class labels and reasoning labels and using the inconsistency between the two to detect adversarial examples.

3 Methodology

In this section, we describe our framework of using reasoning labels for training DNNs.

3.1 Setup

Let $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ denote a training dataset of N examples with $\boldsymbol{x} \in \mathcal{X} = \mathbb{R}^d$ and $y \in \mathcal{Y} = \{1, 2, \ldots, K\}$. For simplicity, we assume that this is *not* a multi-label classification task, that is, an example can belong to only one class. Then the learning task involves obtaining a mapping $F_{\text{clf}} : \mathcal{X} \to \mathcal{Y}$.

For a (deep) neural network with L hidden layers, this mapping consists of applying a set of parameterized layers $f_l(\boldsymbol{x}_l, \boldsymbol{\theta}_l)$. Here, $\boldsymbol{x}_l$ and $\boldsymbol{\theta}_l$ denote, respectively, the input and parameters of the l^{th} layer (we fold any activation operation into the layer f_l itself). The whole neural network mapping can be expressed as:

$$F_{\text{clf}}(\boldsymbol{x}) = f_{\text{clf}}(f_L(f_{L-1}(\ldots, f_1(\boldsymbol{x}, \boldsymbol{\theta}_1)))), \tag{1}$$

where the output of f_{clf}—or the classification layer—is a K-dimensional vector consisting of (potentially un-calibrated [Guo et al., 2017]) probabilities. These probabilities are generally obtained by applying the softmax function within the layer f_{clf}.[1] One then obtains the prediction $\hat{y} = \text{argmax} \quad F_{\text{clf}}(\boldsymbol{x})$.

The learning then boils down to minimizing the discrepancy between the predicted and the ground-truth labels. For the sake of computational tractability, this discrepancy is often expressed via the (categorical) cross-entropy loss function, denoted henceforth as $\mathcal{L}_{\text{clf}}(F_{\text{clf}}(\boldsymbol{x}), y)$.

3.2 Training with reasoning labels

Suppose that in addition to the feature vectors $\boldsymbol{x}$ and class labels y, one is also provided with a corresponding reasoning vector $\boldsymbol{r} \in \mathcal{R} = \{0, 1\}^R$, where R is the total number of reasoning labels for all the K classes combined. Now, let us assume that we are provided with a mapping between classification labels and reasoning labels (e.g., panda $\to$ {animal, paws, spots}, and, bus $\to$ {tires, seat, engine}). Then for a given instance $\boldsymbol{x}$ with class label y, its reasoning label vector $\boldsymbol{r}$ is a binary vector with only the elements corresponding to the reasoning labels of this class label y set to 1.

For instance, if the set of all reasoning labels is {animal, paws, spots, tires, seat, engine}, then for an example with class label 'panda', $\boldsymbol{r} = [1, 1, 1, 0, 0, 0]$ and an example with classification label 'bus', $\boldsymbol{r} = [0, 0, 0, 1, 1, 1]$.

Now learning the reasoning labels can be thought of as learning a function $F_{\text{rsn}}(\boldsymbol{x}) : \mathcal{X} \to \mathcal{R}$—in addition to learning the function $F_{\text{clf}}(\boldsymbol{x})$ in Eq. 1. Note that, as opposed to class labels, learning the reasoning labels is a multilabel classification task. We can express this function as follows:

$$F_{\text{rsn}}(\boldsymbol{x}) = f_{\text{rsn}}(f_L(f_{L-1}(\ldots, f_1(\boldsymbol{x}, \boldsymbol{\theta}_1)))), \tag{2}$$

Where the output of f_{rsn} is a R-dimensional vector consisting of (potentially un-calibrated) probabilities. Since this is a multilabel classification task, the probabilities in this case would be obtained through a sigmoid function $\sigma(z) = (1 + e^{-z})^{-1}$. The loss function would be a binary (as opposed to categorical) cross-entropy loss function. Essentially, we can think of this classification problem as R binary classification problems, where each of the R classification problems aims at predicting the presence or absence of the reasoning label.

Finally, one obtains a reasoning label vector $\hat{\boldsymbol{r}} = [\hat{r_1}, \ldots, \hat{r_R}]$ with $\hat{r}_i = 1$ if $F_{\text{rsn}}(\boldsymbol{x}) > 0.5$, else $\hat{r} = 0$.

It is important to note that the difference in Eqs. 1 and 2 is *only* the last layer—that is, the hidden representations for both the (classification and reasoning) parts are the same (one could however attach f_{clf} and f_{rsn} to different hidden layers).

[1] Where for a given vector $\mathbf{v} = [v_1, v_2, \ldots, v_K]$, softmax($\mathbf{v}$) is another vector $\mathbf{v}^{\text{sft}}$ with $v_i^{\text{sft}} = \frac{exp(v_i)}{\sum_{k=1}^K exp(v_k)}$.

Let $\mathcal{L}_{\text{rsn}}(F_{\text{rsn}}(\boldsymbol{x}), y)$ denote the loss function for predicting the reasoning labels, then the overall optimization problem becomes:

$$\mathcal{L} = \alpha \mathcal{L}_{\text{clf}}(F_{\text{clf}}(\boldsymbol{x}), y) + (1 - \alpha)\mathcal{L}_{\text{rsn}}(F_{\text{rsn}}(\boldsymbol{x}), y), \tag{3}$$

where α denotes the tradeoff between the classification and reasoning accuracy.

3.3 Using class–reasoning label consistency for robust prediction

After training the model as described in Section 3.2, one can use the reasoning labels at prediction time in the following manner.

Let $\boldsymbol{x}$ be an input sample for which the class label is $\hat{y}$ and the predicted reasoning label is $\hat{\boldsymbol{r}}$. Then, we consider the predicted class and reasoning labels to be consistent if $\frac{\sum_i \mathbb{I}[\hat{\boldsymbol{r}}_i = \boldsymbol{r}_i^{\hat{y}}]}{R} \geq t_{\text{rsn}}$ for some $t_{\text{rsn}} \in [0, 1]$. Here, $\boldsymbol{r}_i^{\hat{y}}$ is the binary vector of reasoning labels corresponding to the predicted class $\hat{y}$. Essentially, if the overlap between the predicted reasoning labels, and the reasoning labels corresponding to the predicted class $\hat{y}$ is below the user-specified threshold t_{rsn}, the prediction is deemed as inconsistent. One can choose to ignore these inconsistent predictions, or provide this inconsistency information to the downstream users to adjust their action accordingly.

3.4 How to obtain reasoning labels

The reasoning labels can be obtained via human annotations. That is, a human annotator can specify reasoning labels for a certain class. For example, for a classification task where the goal is to classify the parts on a supply-line into defective / non-defective, an expert can specify the labels "broken welding", "has dent", *etc.*

The task of obtaining reasoning labels can also be automated. That is, given that in many classification tasks, the classification labels consist of named entities (panda, bus), one can extract the reasoning labels from existing knowledge-bases (*e.g.*, YAGO, ConceptNet). This can allow us to control the *type* of reasoning that we want to enforce (*e.g.*, only *IsA* relationships.)

In this work, we use ConceptNet knowledge base [ConceptNet]. Specifically, we start with a given set of class labels for which reasoning labels need to be extracted, and the number of reasoning labels to consider per class (n_{rsn}). Then, for a given class, *e.g.*, automobile, we extract all the relationships of the types *IsA*, *PartOf*, *HasA*, *UsedFor*, *CapableOf*, *AtLocation*, *Causes*, *HasProperty* and *RelatedTo*. We sort the reasoning labels with their respective scores and select the top n_{rsn} as the reasoning labels for this class. See Section 4.1 for examples of reasoning labels extracted for various classes.

4 Evaluation

In this Section, we conduct experiments on a real-world dataset to evaluate the effectiveness of our framework in linking model explainability with model robustness. Specifically, we show that using our notion of explanation consistency defined in Section 3.3, one can obtain considerable gains w.r.t. accuracy and adversarial robustness.

4.1 Dataset and training details

For the experiments, we consider the CIFAR-10 dataset. We use the usual train/test splits provided with the original dataset [CIFAR Data]. Additionally, we split the training dataset into training and validation folds of 80% and 20%, respectively.

We use two sets of reasoning labels, the *small set* and the *large set*. The small set is extracted from ConceptNet such that we extract 20 most relevant reasoning labels per class, whereas for the large set, we extract 60 most relevant reasoning labels per class (see Section 3.4 for more details about the extraction strategy). The total number of reasoning labels across all the classes is 182 for the small set and 538 for the large set. Some examples of the reasoning labels are *Bird* $\rightarrow$ {wings, feather, eggs }, *Dog* $\rightarrow$ {bark, pet, canine} and *Airplane* $\rightarrow$ {sky, propeller, wings}.

For training the model, we consider the architecture described by Zagoruyko [2015]. We select this architecture given that it is fairly simple (a combination of Conventional layers with Batch

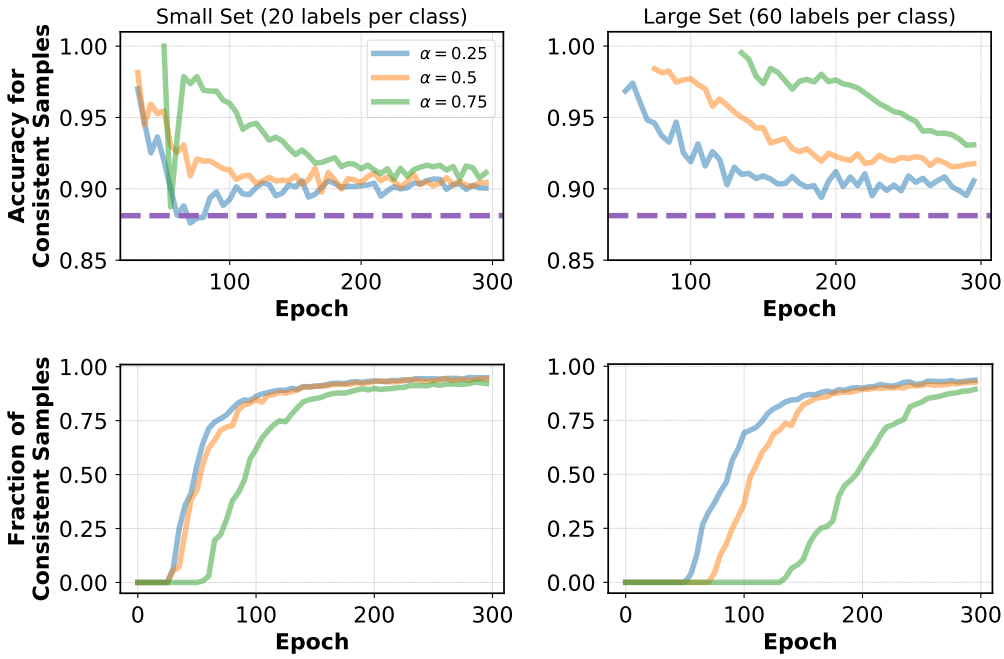

Figure 1: [Gains From Reasoning: 100% threshold] **Left column** shows results for a small set of reasoning labels (20 per class) while **right column** shows results for a large set of reasoning labels (60 per class). We see that in both cases, by making a prediction on only consistent samples, test set accuracy for different values of α (blue, green and orange lines, top row) remains above the test accuracy of a traditional deep neural network trained only to predict class labels (purple dotted line). We also see a trade-off between coverage (the fraction of samples deemed consistent) and accuracy *i.e.*, as we move to higher epochs, coverage increases but the accuracy decreases. The results are computed every 5 epochs.

Normalization and Dropout) and still provides competitive performance as compared to the state-of-the-art.[2] Training this model to maximize the classification accuracy leads to a test set accuracy of 88.12%. We refer to this model as the base model.

We then train the model to additionally predict the reasoning labels using the loss in Eq. 3. Specifically, we minimize the loss with different values of α: 0.25, 0.50 and 0.75.

4.2 Results

We now analyze the effect of training with reasoning labels on model accuracy and robustness to adversarial samples.

Gains in accuracy. Figure 1 shows the fraction of samples deemed consistent and the accuracy on consistent samples for different values of α when considering the small (left column) and large (right column) set of reasoning labels (20 reasoning labels per class). For deciding if a sample is consistent, we set $t_{\text{rsn}} = 1.00$.

Focusing first on the small set of reasoning labels (20 reasoning labels per class), that is, the left column, one notices several interesting insights: First, as the training proceeds over the epochs, the fraction of consistent samples go up. Second, the accuracy on consistent samples is indeed higher than the accuracy of the base model, shown by the dotted horizontal line in the plots. Of course, the model with reasoning labels (solid lines) has slightly lesser coverage than the base model since predictions on inconsistent samples are discarded (the base model, on the other hand, has 100% coverage since there is no option to discard an input). Third, notice that increasing the value of α results in higher test set accuracy for consistent samples, but leads to a slightly lesser fraction

[2] https://tinyurl.com/cifar10-benchmarks

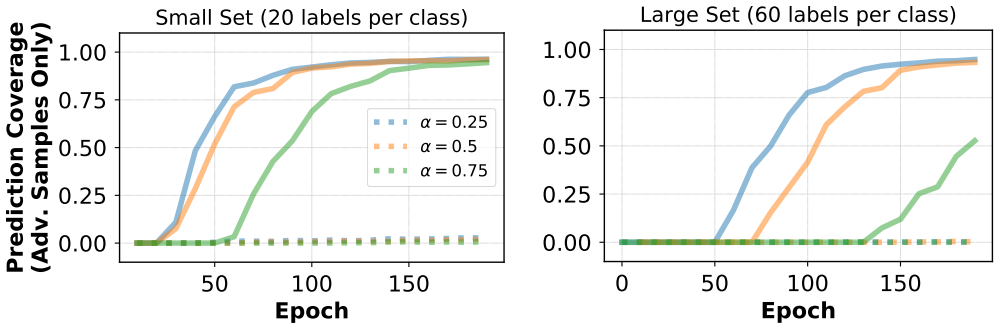

Figure 2: [Robustness to the FGSM Adversarial Attack] Results for a small set of labels (20 per class) on the left and large set of labels (60 per class) on the right. The dotted lines show the coverage for adversarial samples and solid lines show the coverage for the corresponding original unperturbed samples. Even though the unperturbed samples are deemed consistent in a vast majority of the cases by our model (solid lines), their adversarial versions are marked inconsistent in almost all the cases, and hence are rejected. The results are computed every 10 epochs.

of samples that are deemed consistent. Higher values of α result in a higher weight given to class predictions during training, thus ensuring higher accuracy, but also result in lower coverage, thus highlighting the inherent trade-off between coverage and accuracy.

Focusing on the large set of reasoning labels (60 reasoning labels per class), that is, the right column, one notices similar insights. However, we see that as compared to the small set of reasoning labels, fewer samples are deemed consistent, however, the accuracy on consistent samples is significantly higher.

Finally, we also conduct the experiments with setting $t_{\text{rsn}} = 0.90$ (figures omitted due to space constraints). We notice that in this case, a larger fraction of samples are marked consistent, however, one attains lesser accuracy on consistent samples. This observation also highlights the trade-off between prediction coverage, that is, the fraction of samples deemed consistent, and accuracy on consistent samples. This trade-off between consistency and accuracy can be configured by the system user in practice.

Performance on adversarial inputs. In this section we craft adversarial inputs to the trained model with reasoning labels and check if the adversarial perturbations are also able to fool the reasoning label predictor f_{rsn} (in addition to the class label predictor f_{clf}). Specifically, we select all samples from the test set where the class prediction is correct (*i.e.*matches the ground truth) and generate an adversarial perturbation to fool f_{clf}. We then feed these inputs to the model and check the consistency between their (adversarial) class label and the reasoning labels. Consistency between the two labels would mean that the adversarial perturbation was also able to fool the reasoning labels predictor f_{rsn}.

Figure 2 shows the consistency of the model on the adversarial samples and their unperturbed (*i.e.*, original) version. The figure shows that our check on consistency between the class and reasoning label can detect almost all of the adversarially perturbed inputs.

One could try to attack the reasoning labels (in addition to the class labels) as well. However, changing a large number of reasoning labels (20 or 60 in our case) in conjunction with the class labels such that the outcome is deemed consistent might prove to be a harder challenge for the adversary. Analyzing such attacks would be an interesting future research direction.

5 Conclusion and Future Work

In this work, we introduced a framework for explainability that ties model explainability with model robustness via human understandable reasoning labels. Initial experiments show that our framework can lead to significant gains in accuracy by ignoring non-robust predictions, and can also render the model more robust to adversarial perturbations. Future directions include an in depth analysis of trade-offs between accuracy and coverage (*i.e.*, the fraction of samples that are deemed consistent),

exploring further model architectures and datasets, analyzing other types of adversarial attacks (in addition to FGSM attack) and analyzing the effect of reasoning label quality on the performance.

References

Jacob Bien and Robert Tibshirani. Prototype selection for interpretable classification. *The Annals of Applied Statistics*, 5(4):2403–2424, 2011.

Chaofan Chen, Oscar Li, Chaofan Tao, Alina Jade Barnett, Jonathan Su, and Cynthia Rudin. This looks like that: deep learning for interpretable image recognition. *arXiv preprint arXiv:1806.10574*, 2018.

CIFAR Data. CIFAR-10 and CIFAR-100 datasets. https://www.cs.toronto.edu/~kriz/cifar.html. Accessed: 2019-09-17.

ConceptNet. Conceptnet: An open, multilingual knowledge graph. http://conceptnet.io/. Accessed: 2019-09-17.

Sadaf Gulshad, Jan Hendrik Metzen, Arnold Smeulders, and Zeynep Akata. Interpreting adversarial examples with attributes. *arXiv preprint arXiv:1904.08279*, 2019.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of ICML*, pages 1321–1330. JMLR. org, 2017.

Rajiv Khanna, Been Kim, Joydeep Ghosh, and Oluwasanmi Koyejo. Interpreting black box predictions using fisher kernels. *arXiv preprint arXiv:1810.10118*, 2018.

Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. In *Advances in Neural Information Processing Systems*, pages 2280–2288, 2016.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). *arXiv preprint arXiv:1711.11279*, 2017.

Jinkyu Kim, Anna Rohrbach, Trevor Darrell, John Canny, and Zeynep Akata. Textual explanations for self-driving vehicles. In *Proceedings of ECCV*, pages 563–578, 2018.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of ICML*, pages 1885–1894. JMLR. org, 2017.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of SIGKDD*, pages 1135–1144. ACM, 2016.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Guanhong Tao, Shiqing Ma, Yingqi Liu, and Xiangyu Zhang. Attacks meet interpretability: Attribute-steered detection of adversarial samples. In *Advances in Neural Information Processing Systems*, pages 7717–7728, 2018.

Sergey Zagoruyko. 92.45% on CIFAR-10 in torch. http://torch.ch/blog/2015/07/30/cifar.html, 2015. Accessed: 2019-09-17.