A Neurosymbolic Framework for Bias Correction in Convolutional Neural Networks

Parth Padalkar[1], Natalia Ślusarz[2], Ekaterina Komendantskaya[2,3] and Gopal Gupta[1]

[1]University of Texas at Dallas, Richardson, USA
[2]Heriot-Watt University, UK
[3]Southampton University, UK

Abstract

Recent efforts in interpreting Convolutional Neural Networks (CNNs) focus on translating the activation of CNN filters into a stratified Answer Set Program (ASP) rule-sets. The CNN filters are known to capture high-level image concepts, thus the predicates in the rule-set are mapped to the concept that their corresponding filter represents. Hence, the rule-set exemplifies the decision-making process of the CNN w.r.t the concepts that it learns for any image classification task. These rule-sets help understand the biases in CNNs, although correcting the biases remains a challenge. We introduce a neurosymbolic framework called NeSyBiCor for bias correction in a trained CNN. Given symbolic concepts, as ASP constraints, that the CNN is biased towards, we convert the concepts to their corresponding vector representations. Then, the CNN is retrained using our novel semantic similarity loss that pushes the filters away from (or towards) learning the desired/undesired concepts. The final ASP rule-set obtained after retraining, satisfies the constraints to a high degree, thus showing the revision in the knowledge of the CNN. We demonstrate that our NeSyBiCor framework successfully corrects the biases of CNNs trained with subsets of classes from the *Places* dataset while sacrificing minimal accuracy and improving interpretability.

Keywords

Neurosymbolic AI, CNN, Semantic Loss, Answer Set Programming, XAI, Representation Learning

1. Introduction

Deep learning models can be biased based on the training data. One infamous illustration of this bias is exemplified by the "wolf in the snow" problem [1], where convolutional neural networks (CNNs) erroneously identify a husky as a wolf due to the presence of snow in the background. This happened because they learnt to associate 'snow' with 'wolf' based on the training data. This bias can lead to dire consequences if deployed in sensitive scenarios such as such as disease diagnosis ([2]) and autonomous vehicle operation ([3]).

Recent works have shown that it is possible to obtain the knowledge of a trained CNN in the form of a symbolic rule-set, more specifically as a stratified Answer Set Program ([4, 5]). The authors proposed NeSyFOLD, a framework where the activation of filters in the final CNN layer represents predicate truth values in the rule-set, revealing concepts learned by the model and

4th Workshop on Goal-directed Execution of Answer Set Programs (GDE'24), October 12, 2024
parth.padalkar@utdallas.edu (P. Padalkar); nds1@hw.ac.uk (N. Ślusarz); e.komendantskaya@soton.ac.uk (E. Komendantskaya); gupta@utdallas.edu (G. Gupta)
0000-0003-1015-0777 (P. Padalkar); 0000-0001-9727-0362 (G. Gupta)

their relation to the target class. CNN filters, which are $n \times n$ matrices, capture image concepts. Predicates are labeled according to the concepts identified by these filters. The FOLD-SE-M [6] Rule-Based Machine Learning (RBML) algorithm is used to extract the rules from the last layer filters.

2. Methodology

We introduce the NeSyBiCor (Neuro-Symbolic Bias Correction) framework, to aid in correcting pre-identified biases of the CNN. The ASP rule-set generated by NeSyFOLD, from the bias-corrected CNN serves to validate the effectiveness of the framework. The pre-identified biases are presented as semantic constraints based on concepts that should and should not be used to predict the class of an image. These concepts can be selected by scrutinizing the rule-set generated by NeSyFOLD. We map the undesirable/desirable semantic concepts to their corresponding vector representations learnt by the filters. Next, we retrain the CNN with a novel semantic similarity loss function which is a core component of our framework. The loss function is designed to minimize the similarity of the representations learnt by the filters with the undesirable concepts and maximize the similarity to the desirable ones. Once the retraining is complete, we use the NeSyFOLD framework again to extract the refined rule-set. Hence, our approach provides a way to refine a given ASP rule-set subject to some semantic constraints. Figure 1 illustrates the framework.

Computing Concept Representation Vectors: The first step is to obtain *concept representation vectors* for each desired and undesired concept specified in the semantic constraints of each target class. A CNN filter can be flattened into a vector representing the detected patterns. For example, in the rul-set (blue box) shown in Fig. 1, the filter associated with the predicate 'sky1/1' detects blue sky patterns, while 'sky2/1' detects evening sky patterns in desert road images. To compute the concept representation vector for *sky*, we calculate the *filter representation vectors* for all predicates containing *sky* in their name and positively linked to the 'desert road' class. For example, 'sky1/1' and 'sky2/1' have two such filters. Their filter representation vectors are computed by averaging the output vectors of the top-10 images these filters activate in the training set. The final concept representation vector for *sky* is the mean of the 'sky1/1' and 'sky2/1' vectors. This process is repeated for every desired and undesired concepts.

Calculating the Semantic Similarity Loss: The semantic similarity loss, $\mathcal{L}_{\mathcal{SS}}$ for a training set with N images and a CNN with K filters in the last layer, is defined as:

$$\mathcal{L}_{\mathcal{SS}} = \sum_{i=1}^{N} \left[\sum_{j=1}^{K} \left[\lambda_B \sum_{b \in \mathbf{B}} cos\_sim(\mathbf{r}_j^i, \mathbf{r}_b) - \lambda_G \sum_{g \in \mathbf{G}} cos\_sim(\mathbf{r}_j^i, \mathbf{r}_g) \right] \right] \tag{1}$$

Here, *cos_sim* calculates the cosine similarity between two vectors. $\mathbf{r}_j^i$ represents the filter output from the j^{th} filter of the i^{th} image, while $\mathbf{r}_b$ and $\mathbf{r}_g$ are the concept representation vectors for undesired concepts $b \in \mathbf{B}$ and desired concepts $g \in \mathbf{G}$, respectively. λ_B and λ_G are hyperparameters that balance the influence of these terms.

The loss increases when filter vectors resemble undesired concepts and decreases when they are closer to desired concepts, similar to the word2vec loss function. As training progresses, the

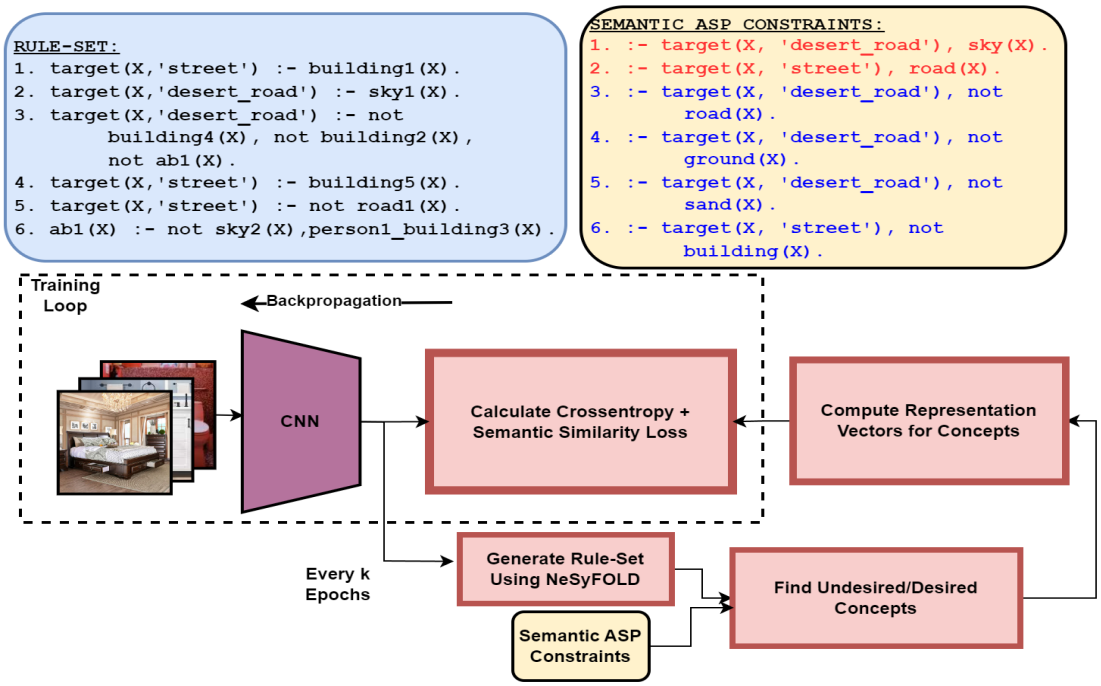

Figure 1: The NeSyBiCor Framework. Note that the crossentropy loss is calculated after the fully connected layer while the semantic similarity loss is calculated by using the filter output feature maps of the last convolution layer

filters are encouraged to learn desired concepts while avoiding undesired ones. The total loss is defined as the sum of crossentropy loss $\mathcal{L}_{\mathcal{CE}}$ and semantic similarity loss $\mathcal{L}_{\mathcal{SS}}$, $\mathcal{L}_{\mathcal{TOTAL}} = \mathcal{L}_{\mathcal{CE}} + \mathcal{L}_{\mathcal{SS}}$

Recalibrating the Concept Representation Vectors: We propose rectifying all the concept representation vectors for each class after every k epochs during training. We do this by running the NeSyFOLD framework after every k epochs and obtaining a new rule set from the partially retrained CNN. We then aggregate the new concept representation vectors with the old concept representation vectors by taking their mean.

3. Experiments and Results

We train a CNN on subsets of the Places [7] dataset and compare the results with the rule-set obtained using NeSyFOLD before and after the bias correction with our NeSyBiCor framework. Details of the experiments can be found elsewhere (Padalkar et al. [8]). Figure 2 shows the initial and bias-corrected rule-set for 2 subsets of the Places dataset. The *des* subset constitutes the 'desert road' and 'street' class and the *defs* subset comprises of the 'desert road', 'forest road' and 'street' classes.

The undesired concepts for the 'desert road' class are 'sky' and 'building'. In RULE-SET 1,

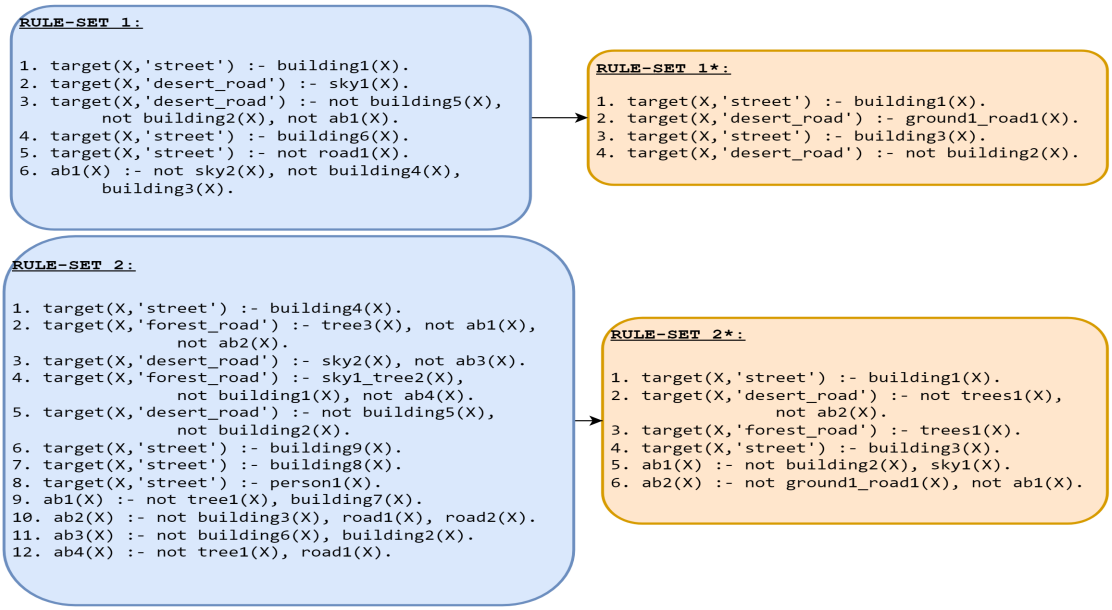

Figure 2: The initial and final rule-sets after applying the NeSyBiCor framework on the CNNs trained on *des* (RULE-SET 1) and *defs* (RULE-SET 2)

rule 2 uses the 'sky1/1' predicate to determine if the image belongs to the 'desert road' class. In the bias-corrected rule-set, RULE-SET 1*, there is no 'sky' based predicate. Moreover, the only predicate positively linked with the 'desert road' class is the 'ground1_road1/1' predicate which is based on the desired concept 'ground' and refers to the corresponding filter, learning a pattern comprising of specific type of patches of 'ground' and 'road'. Thus it is clear that at the end of the bias correction, very few/none of the filters learn representations of the 'sky' or 'building' concepts, hence correcting the "bias" of the CNN towards them while predicting the "desert road" class.

4. Conclusion and Future Work

Our framework, in addition to correcting the bias of a CNN also allows the user to fine-tune the bias based on general concepts according to their specific needs or application. To the best of our knowledge, this is the first method that does bias correction by using the learnt representations of the CNN filters in a targeted manner. We show through our experiments that the bias-corrected rule-set is highly effective at avoiding the classification of images based on undesired concepts. It is also more likely to classify the images based on the desired concepts.

Finally, the work we presented here may be used to extend implementations of loss functions based on Differentiable Logics: [9, 10, 11].

References

[1] M. T. Ribeiro, S. Singh, C. Guestrin, "why should i trust you?": Explaining the predictions of any classifier, 2016. URL: https://arxiv.org/abs/1602.04938. doi:10.48550/ARXIV.1602.04938.

[2] T. Müezzinoglu, N. Baygin, I. Tuncer, P. D. Barua, M. Baygin, S. Dogan, T. Tuncer, E. E. Palmer, K. H. Cheong, U. R. Acharya, Patchresnet: Multiple patch division-based deep feature fusion framework for brain tumor classification using MRI images, 2023. URL: https://doi.org/10.1007/s10278-023-00789-x. doi:10.1007/S10278-023-00789-X.

[3] R. Barea, L. M. Bergasa, E. Romera, E. López-Guillén, O. Perez, M. Tradacete, J. López, Integrating state-of-the-art cnns for multi-sensor 3d vehicle detection in real autonomous driving environments, in: Proc. ITSC, IEEE, 2019, pp. 1425–1431.

[4] P. Padalkar, H. Wang, G. Gupta, Nesyfold: A framework for interpretable image classification, in: Proc. AAAI, AAAI Press, 2024, pp. 4378–4387.

[5] P. Padalkar, H. Wang, G. Gupta, Using logic programming and kernel-grouping for improving interpretability of convolutional neural networks, in: Proc. PADL, volume 14512 of *LNCS*, Springer, 2024, pp. 134–150. URL: https://doi.org/10.1007/978-3-031-52038-9_9. doi:10.1007/978-3-031-52038-9_9.

[6] H. Wang, G. Gupta, FOLD-SE: an efficient rule-based machine learning algorithm with scalable explainability, in: Proc. PADL, volume 14512 of *LNCS*, Springer, 2024, pp. 37–53.

[7] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, A. Torralba, Places: A 10 million image database for scene recognition, 2017.

[8] P. Padalkar, N. Ślusarz, E. Komendantskaya, G. Gupta, A neurosymbolic framework for bias correction in cnns, 2024.

[9] M. Fischer, M. Balunovic, D. Drachsler-Cohen, T. Gehr, C. Zhang, M. Vechev, Dl2: training and querying neural networks with logic, in: International Conference on Machine Learning, PMLR, 2019, pp. 1931–1941.

[10] N. Ślusarz, E. Komendantskaya, M. L. Daggitt, R. Stewart, K. Stark, Logic of differentiable logics: Towards a uniform semantics of dl, in: 24th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR 2023, 2023, pp. 473–493.

[11] M. L. Daggitt, W. Kokke, R. Atkey, N. Slusarz, L. Arnaboldi, E. Komendantskaya, Vehicle: Bridging the embedding gap in the verification of neuro-symbolic programs, 2024. URL: https://doi.org/10.48550/arXiv.2401.06379. doi:10.48550/ARXIV.2401.06379. arXiv:2401.06379.