NeSyFOLD: Extracting Logic Programs from Convolutional Neural Networks

Parth Padalkar[1], Huaduo Wang[1] and Gopal Gupta[1]

[1]University of Texas at Dallas, Richardson, USA

Abstract

We present a novel neurosymbolic framework called NeSyFOLD to extract logic rules from a CNN and create a NeSyFOLD model to classify images. NeSyFOLD's learning pipeline is as follows: (i) We first pre-train a CNN on the input image dataset and extract activations of the last layer kernels as binary values; (ii) Next, we use the FOLD-SE-M rule-based machine learning algorithm to generate a logic program that can classify an image—represented as a vector of binary activations corresponding to each kernel—while producing a logical explanation. The rules generated by the FOLD-SE-M algorithm have kernel numbers as predicates. We have devised a novel algorithm for automatically mapping the CNN kernels to semantic concepts in the images. This mapping is used to replace predicate names (kernel numbers) in the rule-set with corresponding semantic concept labels. The resulting rule-set is interpretable, and can be intuitively understood by humans. We compare our NeSyFOLD framework with the ERIC system that uses a decision-tree like algorithm to obtain the rules. Our framework has the following advantages over ERIC: (i) In most cases, NeSyFOLD generates smaller rule-sets without compromising on the accuracy and fidelity; (ii) NeSyFOLD generates the mapping of filter numbers to semantic labels automatically.

Keywords

XAI, Neurosymbolic AI, Convolutional Neural Networks, Answer Set Programming

1. Introduction

Explainability in AI is an important issue that has resurfaced in recent years as deep learning models have become larger and are applied to an increasing number of tasks. Some applications such as autonomous vehicles, disease diagnosis, and natural disaster prevention are very sensitive areas where a wrong prediction could be the difference between life and death. Motivated by the need for explainability we introduce a neurosymbolic framework NeSyFOLD, that gives a global explanation for the predictions made by a CNN.

The NeSyFOLD framework uses a Rule Based Machine Learning (RBML) algorithm called FOLD-SE-M [1] for extracting a rule-set by using binarized outputs of the last layer kernels of a trained CNN. The rule-set is a *normal* logic program, i.e., Prolog extended with negation-as-failure, with the most important kernels, called *significant* kernels, appearing as predicates in the rule body. We then create a model that uses the extracted rule-set in conjunction with the CNN for inference tasks. We call this the NeSyFOLD model. We use the FOLD-SE-M rule

3rd Workshop on Goal-directed Execution of Answer Set Programs (GDE'23), July 10, 2023

parth.padalkar@utdallas.edu (P. Padalkar); huaduo.wang@utdallas.edu (H. Wang); gupta@utdallas.edu (G. Gupta)

0000-0003-1015-0777 (P. Padalkar); 0000-0002-2118-5425 (H. Wang); 0000-0001-9727-0362 (G. Gupta)

© 2023 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)

interpreter to execute the rules against the binarized vector of an image instance for predicting its class. The rule-set can be viewed as a stratified Answer Set Program (ASP) [2] and an ASP solver such as s(CASP) [3] can be used to interpret the rules. Both s(CASP) and the FOLD-SE-M rule interpreter can generate a justification for a given prediction task. The justification serves as an explanation for the predictions made by the CNN.

We compare our NeSyFOLD framework with the ERIC system [4] which also generates global explanations by extracting a rule-set in the form of a list of CNFs from the CNN through the use of a decision-tree like algorithm. We introduce a novel algorithm for finding groups of similar filters by calculating the cosine similarity between the feature maps generated by various filters. Once such groups are identified each predicate in the rule-set then corresponds to a group of similar filters identifying a particular concept. We call this model the NeSyFOLD-Group model.

We developed an algorithm for mapping the *significant* kernels extracted from the CNN to the semantic concept(s) that they represent. We show in our experiments that the NeSyFOLD model is able to either outperform or meet ERIC's accuracy and fidelity while generating a significantly smaller-sized rule-set. The NeSyFOLD-Group model is able to outperform the NeSyFOLD model and generate an even smaller rule-set.

We use rule-set size as a metric of interpretability. We also propose a novel algorithm for automatic semantic labeling of the predicates in the rule-set extracted by NeSyFOLD while ERIC's rule set has to be labeled manually. In addition to the accuracy, we also compare NeSyFOLD's fidelity with ERIC's. There is always a trade-off between explainability and fidelity due to some information loss caused by the discretization of continuous values which is seen in both systems.

2. Background and Related Work

Default Rules: The FOLD-SE-M algorithm [1] that we employ in our system, learns a rule-set from data as a *default theory*. Default logic is a non-monotonic logic used to formalize commonsense reasoning. A default D is expressed as:

$$\frac{D = A : \mathbf{M}B}{\Gamma} \tag{1}$$

Equation 1 states that the conclusion Γ can be inferred if pre-requisite A holds and B is justified. $\mathbf{M}B$ stands for "it is consistent to believe B". Normal logic programs can encode a default theory quite elegantly [2]. A default of the form:

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_n : \mathbf{M}\neg\beta_1, \mathbf{M}\neg\beta_2 \ldots \mathbf{M}\neg\beta_m}{\gamma}$$

can be formalized as the normal logic programming rule:

$$\gamma \texttt{ :- } \alpha_1, \alpha_2, \ldots, \alpha_n, \texttt{not } \beta_1, \texttt{not } \beta_2, \ldots, \texttt{not } \beta_m.$$

where α's and β's are positive predicates and `not` represents negation-as-failure. We call such rules *default rules*. Thus, the default

$$\frac{bird(X) : M \neg penguin(X)}{flies(X)}$$

will be represented as the following default rule in normal logic programming:

```
flies(X) :- bird(X), not penguin(X).
```

We call `bird(X)`, the condition that allows us to jump to the default conclusion that `X` flies, the *default part* of the rule, and `not penguin(X)` the *exception part* of the rule.

FOLD-SE-M [1] is a Rule Based Machine Learning (RBML) algorithm that builds on top of the FOLD [5] algorithm. The FOLD algorithm learns a default theory with exceptions, represented as a *stratified normal logic program*. The FOLD algorithm incrementally generates literals for *default rules* that cover positive examples while avoiding covering negative examples. It then swaps the positive and negative examples and calls itself recursively to learn exceptions to the default when there are still negative examples falsely covered. The process is applied recursively to learn exceptions to exceptions, exceptions to exceptions to exceptions, and so on. The FOLD-SE-M algorithm has 2 tunable hyperparameters, $ratio$, and $tail$. The $ratio$ controls the upper bound on the number of false positive examples to the number of true positive examples implied by the default part of a rule. The $tail$ controls the limit of the minimum number of training examples a rule can cover. In general increasing the value of $tail$ decreases the number of total rules and vice-versa. We use both the s(CASP) and the FOLD-SE-M interpreter in our experiments.

There is a lot of past work which focuses on visualizing the outputs of the layers of the CNN. These methods try to map the relationship between the input pixels and the output of the neurons. Zeiler et al. [6] and Zhou et al. [7] use the output activation while others [8, 9, 10] use gradients to find the mapping. Unlike NeSyFold, these visualization methods do not generate any rule-set. Zeiler et al. [6] use similar ideas to analyze what specific kernels in the CNN are invoked. There are fewer existing publications on methods for modelling relations between the various important features and generating explanations from them.

Qi et al. [11] propose an Explanation Neural Network (XNN) which learns an embedding in high-dimension space and maps it to a low-dimension explanation space to explain the predictions of the network. A sentence-like explanation including the features is then generated manually. No rules are generated and manual effort is needed. Chen et al. [12] introduce a prototype layer in the network that learns to classify images in terms of various parts of the image. They assume that there is a one to one mapping between the concepts and the filters. We do not make such an assumption. Zhang et al. [13, 14] learn disentangled concepts from the CNN and represent them in a hierarchical graph so that there is no assumption of a one to one filter-concept mapping. However, no logical explanation is generated. Bologna et al. Some popular methods for generating explanations are LIME [15] and its extended rule-based version [16] which generate local explanations based on features. Our system generates global explanations.

3. Methodology

We now proceed to explain the methodology behind the learning, inference, and semantic labeling pipeline used in our NeSyFOLD framework. Our learning and inference pipelines are similar to and inspired by the algorithm used by Townsend et al. for ERIC [4].

3.1. Learning

We start by training the CNN on the input images for the given classification dataset in the conventional manner, namely, by using a loss function to get the loss and then propagating the gradient w.r.t the loss through the CNN via backpropagation. Any optimization technique can be used for updating the weights.

Quantization: Quantization is the process of binarization of the kernel outputs. The quantization function is defined in eq. (3). θ_k is the threshold for a specific kernel k. Each kernel in the last convolution layer produces a feature map $A_{i,k}$ for a given image. We collect all such feature maps from all kernels for all images. For each kernel k, an activation threshold θ_k is calculated by using eq. (4) and eq. (5). α and γ are hyperparameters. Once the θ_k is calculated for each filter then using eq. (3) and eq. (2) we get the binarization $b_{i,k}$ for each image and each kernel. Iterating through all images we obtain a table of size $n \times m$ where n is the number of images in the training data and m is the number of kernels. Each row represents the binarized output of the CNN corresponding to each image. If the kernel grouping is true then for each kernel the top 10 images that activate it are selected. Then the most similar kernels are identified by calculating cosine similarity. Finally, the kernel groups are decided by using a hyperparameter θ_s which we call the "similarity threshold". Each column in the table represents groups of similar kernels in this case.

Rule Extraction: The FOLD-SE-M algorithm is used on the binarization table. It extracts the rules that explain a target class in terms of the kernels/kernel-groups. Each predicate in the body of the rules is a kernel in the CNN. An example rule could be:

```
target(X,'2') :- not 3(X), 54(X), not ab1(X).
```

This rule can be interpreted as "Image X belongs to class `'2'` if kernel `3` is not activated and kernel `54` is activated and the abnormal condition (exception) `ab1` does not apply.

Semantic labeling : After the rules are generated we find the semantic labels corresponding to the predicates/kernels in the rule-set for better interpretability. We use a dataset that provides the corresponding semantic segmentation mask along with the training images. We take an intersection of the semantic segmentation masks with the feature maps produced by each kernel for each image and label that kernel as the majority concept which it is activated by in most images. Note that we currently support labeling the rule-set used by the NeSyFOLD model, where each predicate is a single kernel. The semantic labeling algorithm for the NeSyFOLD-Group model's rule-set is part of future work. The rules with filters replaced by semantic labels are generated. The same example rule from above may now look like:

```
target(X,'bathroom') :- not bed(X), bathtub(X), not ab1(X).
```

$$b_{i,k} = Q(A_{i,k}, \theta_k) \tag{2}$$

$$Q(A_{i,k}, \theta_k) = \begin{cases} 1, & \text{if } a_{i,k} > \theta_k \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$$a_{i,k} = ||A_{i,k}||_2 \tag{4}$$

$$\theta_k = \alpha \cdot \overline{a_k} + \gamma \sqrt{\frac{1}{n} \sum (a_{i,k} - \overline{a_k})^2} \tag{5}$$

3.2. Inference

The inference pipeline of NeSyFOLD is relatively straightforward. To classify a new image, we feed it to the CNN and the kernel feature maps for each kernel in the last layer are obtained. Then, as in the learning pipeline using Equations (2) and (3), we get the binarizations for each kernel output and generate the binarization table as in the learning pipeline. Next, we run the FOLD-SE-M toolkit's built-in rule interpreter on the labeled/unlabelled rule-set.

4. Experiments and Results

First, we compare our NeSyFOLD and NeSyFOLD-Group models with the model generated by the ERIC system and report the accuracy, fidelity, number of unique predicates/atoms in the rule-set, number of rules generated and the size of the rule-set. For ERIC, the rules generated are in the form of CNF formula where each atom is a binarized kernel output or its negation. Size is calculated as the total number of antecedents for ERIC and the total number of predicates in the bodies of the rules that constitute the logic program generated by NeSyFOLD.

We used a VGG16 CNN with pre-trained weights on the Imagenet dataset [17]. We trained for 100 epochs with a batch size of 32. We used the Adam [18] optimizer and applied class weights for imbalanced data. We also used $L2$ regularization of 0.005 on all layers and a learning rate of $5 \times 10^{-}7$. We used a decay factor of 0.5 and patience of 10 epochs. Also, we resized all images to 224×224. We used an α of 0.6 and γ of 0.7. We used almost the same parameters as used by Townsend et al. in [19] for training so that we could do a fair comparison. For this experiment, we used the *German Traffic Sign Recognition Benchmark* (GTSRB) [20], *MNIST* [21] and the Places [22] dataset. We selected 5 classes from the *Places* dataset namely, {desert road(de), driveway(dr), forest road(f), highway(h), street(s)}. In Table 4 and Table 2 we have shown the "dedrf", "dedrh", "dedrs" and "defs" class combinations. We cite the performance metrics of ERIC on these 4 class combinations from the paper [4]. ERIC's fidelity for these class combinations was not reported in the above paper so we leave it blank in the table. To see the effect of varying the number of classes $\in$ {2, 3, 5, 10} we train on the bathroom and bedroom class (PLACES2) first. Then we add the kitchen class (PLACES3), then dining room, living room (PLACES5) and finally home office, office, waiting room, conference room and hotel room (PLACES10). For details on how each data was split, we refer the reader to [19]. The comparison between NeSyFOLD and ERIC is shown in Table 4. The performance metrics of NeSyFOLD-Group are shown in Table 2. Our NeSyFOLD model outperforms ERIC on most of these datasets w.r.t accuracy and fidelity. Fewer distinct edges in the images in the PLACES10 dataset cause loss in accuracy. The GTSRB and MNIST datasets have well defined edges and consequently the accuracy and fidelity of both ERIC and NeSyFOLD is high. The rules generated by NeSyFOLD are **55%** smaller in size on average. The NeSyFOLD-Group model outperforms or

	NeSyFOLD					ERIC				
Dataset	Fidelity	Accuracy	Predicates	Rules	Size	Fidelity	Accuracy	Atoms	Rules	Size
dedrf	**0.89 ± 0.0**	**0.86 ± 0.0**	**25 ± 4**	**20 ± 2**	**45 ± 4**	-	0.82	50	31	171
dedrh	**0.83 ± 0.01**	**0.79 ± 0.01**	**32 ± 5**	**23 ± 3**	**60 ± 11**	-	0.75	44	32	176
dedrs	**0.90 ± 0.0**	**0.88 ± 0.0**	**30 ± 6**	**21 ± 3**	**49 ± 11**	-	0.76	44	34	183
defs	**0.94 ± 0.0**	**0.92 ± 0.0**	**16 ± 4**	**13 ± 3**	**26 ± 7**	-	0.90	33	25	127
PLACES2	**0.93 ± 0.01**	**0.92 ± 0.01**	16 ± 2	12 ± 2	**28 ± 5**	0.89 ± 0.01	0.89 ± 0.01	**11 ± 1**	12 ± 2	52 ± 8
PLACES3	**0.85 ± 0.03**	**0.84 ± 0.03**	**28 ± 6**	**21 ± 4**	**49 ± 9**	0.82 ± 0.01	0.81 ± 0.01	33 ± 4	25 ± 2	118 ± 13
PLACES5	**0.67 ± 0.03**	**0.64 ± 0.03**	**56 ± 3**	52 ± 4	**131 ± 10**	0.65 ± 0.01	0.63 ± 0.01	57 ± 4	**36 ± 2**	171 ± 10
PLACES10	0.23 ± 0.19	0.20 ± 0.17	**33 ± 28**	**32 ± 27**	**78 ± 66**	**0.39 ± 0.01**	**0.36 ± 0.01**	85 ± 6	42 ± 3	208 ± 16
GTSRB	**0.75 ± 0.04**	**0.75 ± 0.04**	**206 ± 28**	134 ± 26	**418 ± 79**	0.73 ± 0.01	0.73 ± 0.01	232 ± 3	**125 ± 5**	626 ± 28
MNIST	0.91 ± 0.01	0.91 ± 0.01	**132 ± 9**	90 ± 7	**271 ± 25**	**0.93 ± 0.00**	**0.92 ± 0.00**	147 ± 2	**86 ± 4**	413 ± 18

Table 1
Performance comparison: NeSyFOLD model vs ERIC model

	NeSyFOLD-Group				
Dataset	Fidelity	Accuracy	Predicates	Rules	Size
dedrf	**0.89 ± 0.01**	**0.86 ± 0.01**	**19 ± 4**	**17 ± 4**	**36 ± 10**
dedrh	**0.83 ± 0.01**	**0.80 ± 0.01**	**30 ± 2**	**21 ± 3**	**53 ± 6**
dedrs	0.89 ± 0.01	**0.88 ± 0.01**	**26 ± 4**	**19 ± 2**	**43 ± 6**
defs	**0.94 ± 0.01**	**0.92 ± 0.01**	**12 ± 3**	**10 ± 1**	**18 ± 3**
PLACES2	**0.93 ± 0.0**	**0.93 ± 0.0**	**8 ± 1**	**7 ± 1**	**11 ± 2**
PLACES3	**0.87 ± 0.01**	**0.86 ± 0.01**	**20 ± 7**	**15 ± 3**	**31 ± 9**
PLACES5	**0.68 ± 0.02**	**0.65 ± 0.02**	**41 ± 4**	**34 ± 6**	**83 ± 13**
PLACES10	**0.33 ± 0.17**	**0.30 ± 0.15**	74 ± 39	73 ± 39	184 ± 97
GTSRB	**0.76 ± 0.02**	**0.76 ± 0.02**	**176 ± 13**	**98 ± 11**	**320 ± 30**
MNIST	0.90 ± 0.01	0.90 ± 0.01	**103 ± 12**	**79 ± 10**	**216 ± 28**

Table 2
Performace metrics of the NeSyFOLD-Group model. Boldened values are better than NeSyFOLD model.

equals the NeSyFOLD model w.r.t accuracy and w.r.t fidelity in all but one dataset where they are comparable. The grouping of filters helps in generating better features for the binarization table.

Second, we examine the effect of semantic labeling of the rule-set generated for the NeSyFOLD model. We use the ADE20k dataset for mapping the predicates in the rule-set obtained by NeSyFOLD trained on 3 different classes of the Places dataset separately. These classes are {bathroom, bedroom, kitchen} and {desert road, driveway, forest road}.

Below we show the labeled rule-set and the interpretation of the first rule for each class from both rule-sets in English. The interpretation of the other rules can be done similarly. We used a $ratio$ of 0.5 and $tail$ of $1e^{-2}$.

```
{Rule-set 1: }
target(X,'kitchen') :- cabinet1(X), not ab1(X), not ab2(X), not ab3(X).
target(X,'bedroom') :- bed1(X).
target(X,'bathroom') :- countertop1_toilet1_sink1(X).
target(X,'bathroom') :- not cabinet3(X), not countertop1_toilet1_sink1(X), bathtub1(X).
target(X,'bedroom') :- bed2(X),   not mirror1_countertop2(X).
target(X,'kitchen') :- cabinet7(X).
```

```
ab1(X) :- not cabinet4(X), not cabinet8(X).
ab2(X) :- not cabinet2(X), bed1(X).
ab3(X) :- not cabinet6(X), countertop1_toilet1_sink1(X),not cabinet5(X).
```

The interpretation of the first rule of the 'kitchen' class after expanding the `ab1(X)`, `ab2(X)` and `ab3(X)` predicates is the following: "Image X is a kitchen if it has cabinets and no countertop-toilet-sink combination and no bed." The same interpretations for the 'bedroom' and 'bathroom' classes are "Image X is a bedroom if there is a bed" and "Image X is a bathroom if there is a combination of countertop, toilet, and sink", respectively. This rule-set has 9 rules, 13 Unique predicates and the total size is 19. It achieves 80% accuracy on the test set.

```
{Rule-set 2:}
target(X,'forest_road') :- trees2(X), not house4(X), not ab1(X).
target(X,'driveway') :- house5_building2(X).
target(X,'desert_road') :- not trees1(X), not ab2(X).
target(X,'forest_road') :- not house2(X), not trees2(X), trees3(X).
target(X,'driveway') :- not house5_building2(X), not road2(X).
target(X,'forest_road') :- not road2(X).
ab1(X) :- not trees3(X), house3_building1(X).
ab2(X) :- not ground1_road1(X), house1(X).
```

The interpretation of the first rule for the 'forest road' class after expanding the `ab1(X)` predicate is the following: "Image X is a forest road if it has trees and no houses or buildings". Likewise, the first rule for 'desert road' and 'driveway' classes represents: "Image X is a desert road if there are no trees and no houses" and "Image X is a driveway if there is a house/building", respectively. This rule-set has 8 rules, 10 Unique predicates and the total size is 16. It achieves 83% accuracy on the test set. For an example image classified as a "forest road", the justification can be obtained from s(CASP) querying `target(img, 'forest_road')` to `Rule-set 2`

```
JUSTIFICATION_TREE:
'target' holds (for img, and forest_road),because 'trees2' holds (for img),
and there is no evidence that 'house4' holds (for img), andthere is no evidence that
'ab1' holds (for img), because'trees3' holds (for img).
MODEL:
{target(img,forest_road),  trees2(img), not ab1(img),  trees3(img)}
```

From the justification, it can be clearly seen that the first rule was invoked. Notice that the FOLD-SE-M algorithm orders the rules according to the number of images covered by the rules in the form of a decision list. That is, the topmost rule will classify the most images, and the bottommost the least. Hence the topmost rules make more sense as they capture all the concepts that can make the sharpest distinction among the 3 classes. The bottom rules capture the outliers that were not classified by the above rules. For example, the 6th rule in Rule-set 2 will be invoked only if all the other rules above it were not applicable. This intuitively suggests that the image is a 'forest_road' image if there is not a particular type of road (`not road2(X)`), but only if it does not fit in the more frequently seen bin of images of 'forest_road' which have trees and no houses or buildings (as captured by rule 1). Note, the heavy biases in the data that the CNN learns would get captured as defaults in the top rules. Also, both s(CASP) and FOLD-SE-M rule interpreters can automatically generate a justification for each example that is classified.

Third, since each predicate represents a kernel's binarized output and for each predicate its corresponding semantic concept(s) is known, there should be a way to utilize the kernels present in the rules to form new rules. We ran an experiment where we trained a CNN on the {desert, highway, lake} classes and generated the labeled rules using NeSyFOLD. We then wrote a new rule `target(X, 'beach'):- sand(X), water(X).` using the predicates associated with water and sand from the existing rule-set. We used this rule only to classify images of the 'beach' class of the Places dataset. Out of the 1000 images, 690 (69%) were correctly classified as 'beach' by the rule. This is an interesting insight as it shows that the CNN filters can be individually used as concept/object detectors.

5. Conclusion and Future Work

In this paper we have shown that using FOLD-SE-M algorithm leads to a significant reduction in the rule-set size without compromising on the accuracy in most cases. The kernel grouping intuitively leads to better features in the binarization table and hence the FOLD-SE-M algorithm is able to represent the information in fewer rules. The semantic labeling algorithm we propose makes the NeSyFOLD model's rule-set easier to understand. A natural direction for future research is modifying the semantic labeling algorithm to accomodate the predicates that represent groups of kernels generated by the grouping algorithm. We believe this will be an important step for writing new rules to classify images of unseen classes during training.

References

[1] H. Wang, G. Gupta, FOLD-SE: An efficient rule-based machine learning algorithm with scalable explainability, 2022. doi:10.48550/ARXIV.2208.07912.

[2] M. Gelfond, Y. Kahl, Knowledge representation, reasoning, and the design of intelligent agents: The answer-set programming approach, Cambridge University Press, 2014.

[3] J. ARIAS, M. CARRO, E. SALAZAR, K. MARPLE, G. GUPTA, Constraint answer set programming without grounding, Theory and Practice of Logic Programming 18 (2018) 337–354. doi:10.1017/S1471068418000285.

[4] J. Townsend, T. Kasioumis, H. Inakoshi, Eric: Extracting relations inferred from convolutions, in: H. Ishikawa, C.-L. Liu, T. Pajdla, J. Shi (Eds.), Computer Vision – ACCV 2020, Springer International Publishing, Cham, 2021, pp. 206–222.

[5] F. Shakerin, E. Salazar, G. Gupta, A new algorithm to automate inductive learning of default theories, TPLP 17 (2017) 1010–1026.

[6] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: European conference on computer vision, Springer, 2014, pp. 818–833.

[7] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, A. Torralba, Learning deep features for discriminative localization, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2921–2929.

[8] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: Visual explanations from deep networks via gradient-based localization, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 618–626.

[9] M. Denil, A. Demiraj, N. de Freitas, Extraction of salient sentences from labelled documents, 2014. URL: https://arxiv.org/abs/1412.6815. doi:10.48550/ARXIV.1412.6815.

[10] K. Simonyan, A. Vedaldi, A. Zisserman, Deep inside convolutional networks: Visualising image classification models and saliency maps, 2013. URL: https://arxiv.org/abs/1312.6034. doi:10.48550/ARXIV.1312.6034.

[11] Z. Qi, S. Khorram, L. Fuxin, Embedding deep networks into visual explanations, Artificial Intelligence 292 (2021) 103435. doi:10.1016/j.artint.2020.103435.

[12] 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).

[13] Q. Zhang, R. Cao, Y. N. Wu, S.-C. Zhu, Growing interpretable part graphs on convnets via multi-shot learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 31, 2017.

[14] Q. Zhang, R. Cao, F. Shi, Y. N. Wu, S.-C. Zhu, Interpreting CNN knowledge via an explanatory graph, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 32, 2018.

[15] 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.

[16] M. T. Ribeiro, S. Singh, C. Guestrin, Anchors: High-precision model-agnostic explanations, in: Proceedings of the AAAI conference on artificial intelligence, volume 32, 2018.

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255.

[18] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014. URL: https://arxiv.org/abs/1412.6980. doi:10.48550/ARXIV.1412.6980.

[19] J. Townsend, M. Kudla, A. Raszkowska, T. Kasiousmis, On the explainability of convolutional layers for multi-class problems, in: Combining Learning and Reasoning: Programming Languages, Formalisms, and Representations, 2022. URL: https://openreview.net/forum?id=jgVpiERy8Q8.

[20] J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel, Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition, Neural Networks 32 (2012) 323–332. URL: https://www.sciencedirect.com/science/article/pii/S0893608012000457. doi:https://doi.org/10.1016/j.neunet.2012.02.016, selected Papers from IJCNN 2011.

[21] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (1998) 2278–2324.

[22] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, A. Torralba, Places: A 10 million image database for scene recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence (2017).