NeSyFOLD: A Framework for Interpretable Image Classification

Parth Padalkar, Huaduo Wang, Gopal Gupta

The University of Texas at Dallas
{parth.padalkar, huaduo.wang, gupta}@utdallas.edu

Abstract

Deep learning models such as CNNs have surpassed human performance in computer vision tasks such as image classification. However, despite their sophistication, these models lack interpretability which can lead to biased outcomes reflecting existing prejudices in the data. We aim to make predictions made by a CNN interpretable. Hence, we present a novel framework called NeSyFOLD to create a neurosymbolic (NeSy) model for image classification tasks. The model is a CNN with all layers following the last convolutional layer replaced by a stratified answer set program (ASP) derived from the last layer kernels. The answer set program can be viewed as a rule-set, wherein the truth value of each predicate depends on the activation of the corresponding kernel in the CNN. The rule-set serves as a global explanation for the model and is interpretable. We also use our NeSyFOLD framework with a CNN that is trained using a sparse kernel learning technique called Elite BackProp (EBP). This leads to a significant reduction in rule-set size without compromising accuracy or fidelity thus improving scalability of the NeSy model and interpretability of its rule-set. Evaluation is done on datasets with varied complexity and sizes. We also propose a novel algorithm for labeling the predicates in the rule-set with meaningful semantic concept(s) learnt by the CNN. We evaluate the performance of our "semantic labeling algorithm" to quantify the efficacy of the semantic labeling for both the NeSy model and the NeSy-EBP model.

Introduction

Interpretability 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 (Kanagaraj et al. 2021), disease diagnosis (Sun, Zheng, and Qian 2016), and natural disaster prevention (Ko and Kwak 2012) are very sensitive areas where a wrong prediction could be the difference between life and death. The above tasks rely heavily on good image classification models such as Convolutional Neural Networks (CNNs). A CNN is a deep learning model used for a wide range of image classification and object detection tasks, first introduced by Y. Lecun et al. (LeCun et al. 1989). Current CNNs are extremely powerful and capable of outperforming humans in image classification tasks. A CNN is inherently a blackbox model though attempts have been made to make it more interpretable (Zhang et al. 2017, 2018).

While these deep models achieve remarkable accuracy, their interpretability becomes compromised. It remains unclear what the model has truly learnt and whether its predictions are rooted in human-understandable, semantically meaningful patterns or mere coincidental correlations within the dataset. Motivated by the need for interpretability, in this paper we introduce a framework called NeSyFOLD, which *a)* gives a global explanation for the predictions made by a CNN in the form of an ordered rule-set and *b)* generates an interpretable NeSy model that can be used to make predictions instead of the CNN. This resultant rule-set can be readily scrutinized by domain experts, enabling the discernment of potential biases towards specific classes or learning of non-intuitive class representations by the trained model.

The NeSyFOLD framework uses a Rule Based Machine Learning (RBML) algorithm called FOLD-SE-M (Wang and Gupta 2024) for generating a rule-set by using binarized outputs of the last layer kernels of a trained CNN. The rule-set is a (stratified) answer set program. The most influential kernels, called *significant* kernels, appear as predicates in the rule body. The binarized kernel's output determines the truth value of its corresponding predicate in the rule-set. Additionally, Zhou et al. (Zhou et al. 2014) showed that training a CNN on scene classification datasets results in last-layer kernels emerging as object detectors. Motivated by their discovery, we developed an algorithm for labeling the corresponding predicates of the *significant* kernels with the semantic concept(s) that they represent. We call this procedure the *semantic labeling* of the kernels. For example, the predicate `52(X)` corresponding to kernel `52` will be replaced by `bathtub(X)` in the rule-set, if kernel `52` has learnt to look for "bathtubs" in the image. Fig. 1 shows an outline of our NeSyFOLD framework. The semantic labeling can be done manually as demonstrated by Townsend et al. (Townsend, Kasioumis, and Inakoshi 2021), however, it entails substantial time and labour investments. We automate this process using our semantic labeling algorithm.

Next, we create a model that uses the generated rule-set in conjunction with the CNN for inference. We call this the *NeSy* model. We use the in-built FOLD-SE-M rule interpreter (Wang and Gupta 2024) to execute the rules against

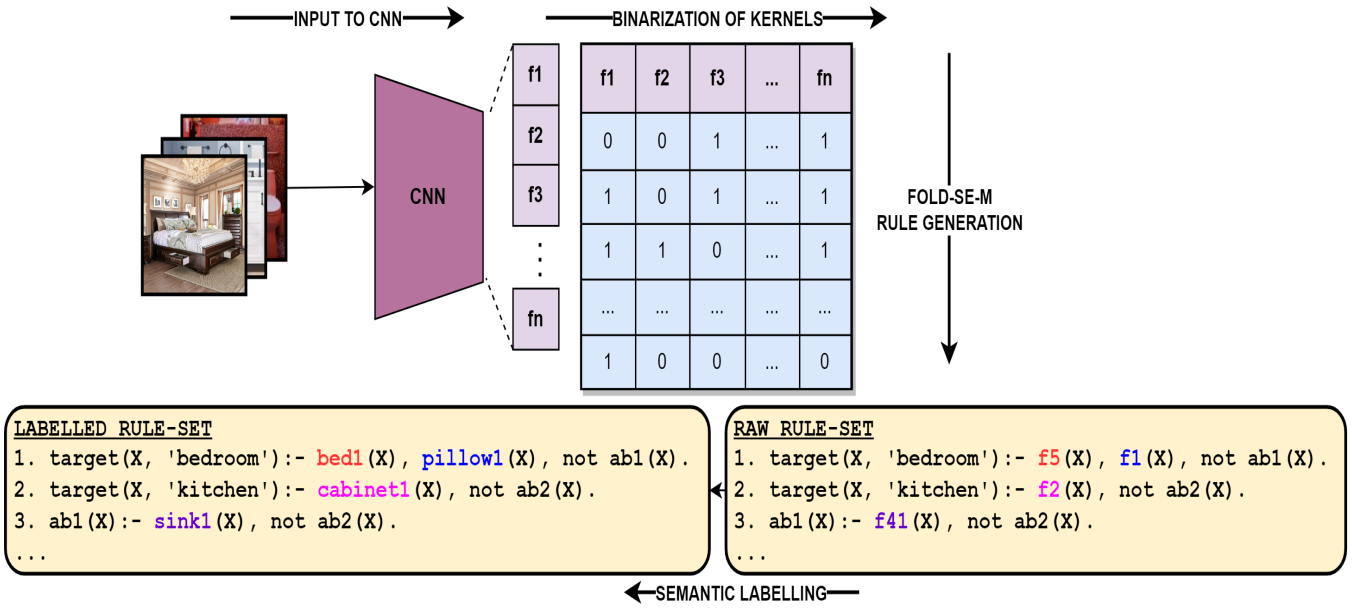

Figure 1: The NeSyFOLD Framework

the binarized vector of an image instance obtained from the last convolutional layer kernels of the CNN for predicting its class. To obtain the justification of a particular prediction we use the s(CASP) ASP solver (Arias et al. 2018). The justification serves as an explanation for the predictions made by the CNN.

We compare our NeSyFOLD framework with the ERIC system (Townsend, Kasioumis, and Inakoshi 2021) (SOTA) 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 show through our experiments that our NeSy model outperforms ERIC's accuracy and fidelity while generating a significantly smaller-sized rule-set. The rule-set size can be further reduced by learning sparse kernels. We employ the Elite BackProp (EBP) (Kasioumis, Townsend, and Inakoshi 2021) algorithm to train the CNN so as to learn sparse kernels. We then obtain the NeSy-EBP model from this EBP-trained CNN and evaluate its performance as well. Our novel contributions are as follows:

1. We show that our method of rule extraction significantly reduces the size of the rule-set obtained, hence making the rule-set more intuitive and interpretable.
2. We show that the scalability of our framework can be further improved by using sparse kernel learning techniques such as EBP.
3. We present a novel algorithm for the semantic labeling of the predicates in the rule-set and do an extensive evaluation to show its effectiveness.

Background

FOLD-SE-M: The FOLD-SE-M algorithm (Wang and Gupta 2024) that we employ in our framework, 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:

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

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 (Gelfond and Kahl 2014). 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 (Wang and Gupta 2024) is a Rule Based Machine Learning (RBML) algorithm. It generates a rule-set from tabular data, comprising rules in the form described above. The complete rule-set can be viewed as a stratified

answer set program. It uses special `abx` predicates to represent the exception part of a rule where `x` is a unique numerical identifier. FOLD-SE-M 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.

There are 2 tunable hyperparameters, $ratio$, and $tail$. The $ratio$ controls the upper bound on the number of false positives to the number of true positives implied by the default part of a rule. The $tail$ controls the limit of the minimum number of training examples a rule can cover. FOLD-SE-M generates a much smaller number of rules than a decision-tree classifier and gives higher accuracy in general.

Elite BackProp: Elite BackProp (Kasioumis, Townsend, and Inakoshi 2021) is a training algorithm that associates each class with very few *Elite* kernels that activate strongly. This is achieved by introducing a loss function that penalises the kernels that have a lower probability of activation for any class and reinforces the kernels that have a higher probability of activation during training. This leads to fewer kernels learning representations for each class. The same number of (elite) kernels are reinforced for each class which is decided by the hyperparameter K. Another hyperparameter λ is used as the regularization constant.

Methodology

We now proceed to explain the methodology behind the learning, inference, and semantic labeling pipeline used in our NeSyFOLD framework.

Learning

We start by training the CNN on the input images for the given classification dataset. Any optimization technique can be used for updating the weights. The learning pipeline is illustrated in Figure 1.

Quantization: Quantization is the process of binarization of the kernel outputs. Once the CNN has been fully trained to convergence, we pass the full train set consisting of n images to the CNN. For every image i in the train set, we obtain a feature map $A_{i,k}$ for every kernel k in the last convolutional layer. The feature map $A_{i,k}$ is a $2D$ matrix of dimension determined by the CNN architecture. For each image i there are K feature maps generated where K is the total number of kernels in the last convolutional layer of the CNN. We map each of these feature maps to a single real value by taking their norms as demonstrated by eq. (2). Next, to determine an appropriate threshold θ_k for each kernel k to binarize its output, we use a weighted sum of the mean and the standard deviation of the norms $a_{i,k}$ for all i for a given k using eq. (3) where α and γ are hyperparameters. Binarizing a kernel refers to determining whether the kernel is active or inactive for a specific image. Thus a binarization table B is created. Each row in the table represents an image and each column is the binarized kernel feature map value represented by either a 0 (inactive) or 1 (active). This is done using eq. (4) where $Q(A_{i,k}, \theta_k)$ is the quantization function that outputs a value of 1 if, for a feature map $A_{i,k}$ its norm $a_{i,k}$ is greater than the kernel $k's$ threshold θ_k and 0 otherwise.

$$a_{i,k} = ||A_{i,k}||_2 \tag{2}$$

$$\theta_k = \alpha \cdot \overline{a_k} + \gamma \sqrt{\frac{1}{n} \sum (a_{i,k} - \overline{a_k})^2} \tag{3}$$

$$Q(A_{i,k}, \theta_k) = \begin{cases} 1, & \text{if } a_{i,k} > \theta_k \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Rule-set Generation: The binarization table B is given as an input to the FOLD-SE-M algorithm to obtain a rule-set in the form of a stratified answer set program. The raw rule-set has predicates with names in the form of their corresponding kernel's index. 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". There will be another rule with the head as `ab1(X)` in the rule-set. The binarized output of a kernel would determine the truth value of its predicate in the rule-set.

Semantic labeling: Every kernel in the last layer of the CNN learns to identify certain concepts. Since we capture the outputs of the kernels as truth values of predicates in the rule-set, we can label the predicates as the semantic concept(s) that the corresponding kernel has learnt. Thus, the same example rule given above may now appear as:

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

Currently, the process of attributing semantic labels to predicates involves identifying *top-m* images with the highest feature-map norms generated by the kernel. Next, the assignment of semantic labels is executed through manual observation of the resulting m images by discerning the concepts that emerge most prominently. This methodology is exemplified in the work of Townsend et al. (Townsend, Kasioumis, and Inakoshi 2021). We introduce a novel semantic labeling algorithm to automate the semantic labeling of the predicates. The details of the algorithm are discussed later.

Inference

The inference pipeline of NeSyFOLD is relatively straightforward. We feed the test set images to the CNN to obtain the kernel feature maps for each kernel in the last convolutional layer. Then, using eq. (4), we get the binarizations for each kernel output and generate the binarization table B_{test}. Note here that the threshold θ_k for each kernel k is the same that was calculated in the learning process. Next, for each binarized vector b in B_{test}, we run the FOLD-SE-M toolkit's built-in rule interpreter on the labeled/unlabeled rule-set to make predictions. The truth value of the predicates in the rule-set is determined by the corresponding binarized kernel values in b. The binarized kernel values in b can be listed as facts and the ASP rule-set can be queried with the s(CASP)

interpreter to obtain the justification as well as the target class. The aforementioned procedure can be conceptualized as replacing all the layers following the last convolutional layer with the rule-set and then making the predictions. This integrated model is referred to as the NeSy model.

Semantic Labeling of Significant Kernels

Recall that a *significant* kernel is a kernel whose corresponding predicate appears in the rule-set generated by FOLD-SE-M. The rule-set initially has kernel ids as predicate names. Also, the FOLD-SE-M algorithm finds only the most influential kernels and uses them as predicates. We present a novel algorithm for automatically labeling the corresponding predicates of the *significant* kernels with the semantic concept(s) that the kernels represent as shown in Alg. 1.

Xie et al. (Xie et al. 2017) showed that each kernel in the CNN may learn to represent multiple concepts in the images. As a result, we assign semantic labels to each predicate, denoting the names of the semantic concepts learnt by the corresponding kernel. To regulate the extent of approximation, i.e., to dictate the number of concept names to be included in the predicate label, we introduce a hyperparameter *margin*. This hyperparameter exercises control over the precision of the approximation achieved. Figure 2 illustrates the semantic labeling of a given predicate. We use the *ADE20k* dataset (Zhou et al. 2017b) in our experiments. It provides manually annotated semantic segmentation masks for a few images of all the classes of the Places (Zhou et al. 2017a) and SUN (Xiao et al. 2010) datasets. This essentially means that for every image i in the dataset I, there is an image i_{Mask} where every pixel is annotated with the label of the object (concept) that it belongs to (Fig. 2 middle). In Alg. 1 we denote these by I_{Mask}.

The CNN M that is trained on the train set is used to obtain the norms $a_{i,k}$ of the feature maps $A_{i,k}$ extracted from the last convolution layer for each kernel k same as in the learning process. Now for each kernel k the *top-m* images I_m, according the norms $a_{i,k}$ are selected. These images are masked with their resized feature maps $A_{i,k}$ to obtain $\hat{I}_m$ (Fig. 2 top). Next, for each masked image $\hat{i}_m \in \hat{I}_m$ we calculate the IoU_c score (eq. (5)) for each concept c that appears in its corresponding semantic segmentation mask i_{Mask} and sum the scores over all images I_m for each concept c. We then normalize the scores over all c and sort the concepts in descending order w.r.t. their respective IoU_c scores. Finally, the predicate associated with significant kernel k is labeled as a concatenation of all the concepts that are in a specific margin from the concept with the highest IoU_c score. This is determined by the *margin* hyperparameter. For example, if the concept IoU_c scores for kernel 12 are $\{cabinets : 0.5, door : 0.4, drawer : 0.1\}$, then, with a *margin* of 0.1, the label for the kernel will be "$cabinets1\_door1$". Also, if some kernel previously has already been labeled $cabinets1$, then the numerical identifier for the label for kernel 12, will be 2 for $cabinets$ in the label i.e. "$cabinets2\_door1$".

$$IoU_c(i_{Mask}, \hat{i}) = \frac{\text{no. of pixels in } c \cap \hat{i}}{\text{no. of pixels in } \hat{i}} \quad (5)$$

Algorithm 1: Semantic labeling

Require: The trained CNN model M, the images I and their semantic segmentation masks I_{Mask}, unlabeled rule-set R, hyperparameter $margin$

1: Obtain feature maps $A_{i,k}$, norms $a_{i,k}$ for all significant kernels k and images i in I using M
2: **for** Each significant kernel k in R **do**
3: $I_m \leftarrow top-m$ images w.r.t. $a_{i,k}$
4: $\hat{I}_m \leftarrow$ Mask each i_m in I_m with its resized $A_{i_m,k}$
5: **for all** $\hat{i}_m$ in $\hat{I}_m$ **do**
6: $IoU_c^{\hat{i}_m} \leftarrow IoU_c(i_{Mask}, \hat{i}_m)$ for each concept c in i_{Mask}
7: **end for**
8: $IoU_c \leftarrow \sum^{\hat{i}_m} IoU_c^{\hat{i}_m}$ for all c
9: Reverse sort normalized IoU scores over all c
10: Label the kernel k as all concepts c that have IoU score within a $margin$ from the top concept.
11: **end for**

Experiments

We conducted experiments to address the following questions:

Q1: How well does NeSyFOLD scale w.r.t. number of classes and number of images in the train set?

Q2: What is the effect of using a CNN trained with Elite BackProp in our NeSyFOLD framework?

Q3: How well do the semantic labels associated with the predicates adhere to the concepts during inference?

Q4: How many concepts are the predicate labels comprised of after semantic labeling?

[Q1/Q2] Scalability: We define scalability as a function of accuracy, fidelity and size of rule-set. As the number of classes in the dataset increases, the accuracy and fidelity of the framework should be high w.r.t. the trained CNN and the rule-set generated should be as small as possible to increase interpretability (Lage et al. 2019). To determine the scalability of our NeSyFOLD framework we evaluate the fidelity, accuracy, no. of unique predicates/atoms in the rule-set and the overall rule-set size over various subsets of the *Places* (Zhou et al. 2017a) dataset which has images of various scenes and the *German Traffic Sign Recognition Benchmark* (GTSRB) (Stallkamp et al. 2012) dataset which consists of images of various traffic signposts. We selected subsets of 2, 3, 5 and 10 classes from the *Places* dataset. We started with the bathroom and bedroom classes (P2). Subsequently, we incorporated the kitchen class (P3.1), followed by the dining room and living room (P5), and finally, home office, office, waiting room, conference room, and hotel room (P10). We also selected 2 additional subsets of 3 classes each i.e. desert road, forest road, street (P3.2) and desert road, driveway, highway (P3.3). We show the evaluation results of our

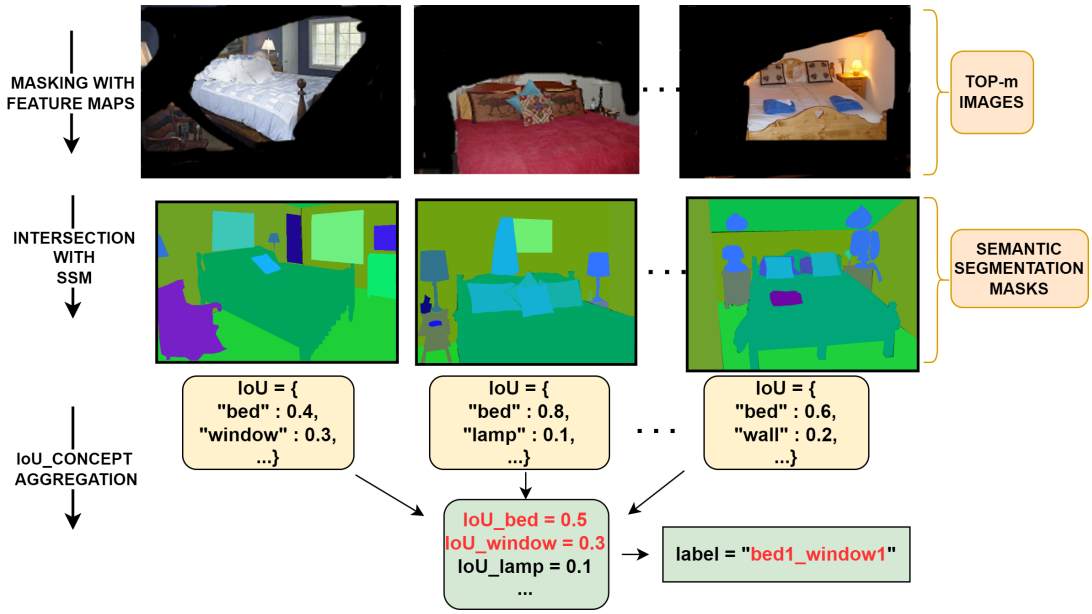

Figure 2: Semantic labeling of a predicate ($margin = 0.2$).

semantic labeling algorithm on P3.1, P3.2 and P3.3 later. Each class has $5k$ images of which we made a $4k/1k$ train-test split for each class and we used the given validation set as it is. The *GTSRB* (GT43) dataset has 43 classes of sign-posts. We used the given test set of $12.6k$ images as it is and did an $80:20$ train-validation split which gave roughly $21k$ images for the train set and $5k$ for the validation set.

We use rule-set size as a metric of interpretability. Lage et al. (Lage et al. 2019) showed through human evaluations that as the size of the rule-set increases the difficulty in interpreting the rule-set also increases. Size is calculated as the total number of antecedents for ERIC and the total number of predicates in the bodies of the rules generated by NeSy-FOLD.

Setup: We employed a VGG16 CNN pretrained on *Imagenet* (Deng et al. 2009), training over 100 epochs with batch size 32. The Adam (Kingma and Ba 2014) optimizer was used, accompanied by class weights to address data imbalance. $L2$ Regularization of 0.005 spanning all layers, and a learning rate of 5×10^{-7} was adopted. A decay factor of 0.5 with a 10-epoch patience was implemented. Images were resized to 224×224, and hyperparameters α and γ (eq. (3)) for calculating threshold for binarization of kernels, were set at 0.6 and 0.7 respectively.

We re-trained each of the trained CNN models again for each dataset, for 50 epochs using EBP. We used $K = 20$ for "P10" and "GT43" because of their larger size and $K = 5$ for all the other datasets. We used $\lambda = 0.005$ for all datasets. We chose EBP for learning sparse kernels because Kasioumis et. al (Kasioumis, Townsend, and Inakoshi 2021) show that EBP learns better representations than other sparse kernel learning techniques. We then used our NeSy-FOLD framework both with the vanilla-trained CNN and the EBP-trained CNN to generate the respective NeSy models (NF and NF-E) as described previously. For P10, per run, we selected the highest 10% softmax-scored images in each class. This subset was then used exclusively for rule-set generation. As is evident from the CNN's low accuracy on P10, the kernels learnt poor representations. Hence, we selected only the above mentioned, relatively high confidence subset for training. The performance of these NeSy models are reported in Table 1. The results are reported after 5 runs on each dataset. For comparison, the trained CNN's accuracy is $\{0.97, 0.94, 0.96, 0.89, 0.85, 0.70, 0.98\}$ for the datasets listed in Table 1 in the same order.

Our experimental setup closely resembled Townsend et al.'s (Townsend et al. 2022; Townsend, Kasioumis, and Inakoshi 2021) facilitating a meaningful comparison. The performance metrics of ERIC reported in Table 1 are also taken from these papers. ERIC's fidelity for P3.2 and P3.3 remains unreported in the mentioned papers and thus is left blank in the table.

Note that comparison is done only for the datasets used for evaluation of the ERIC system (Townsend, Kasioumis, and Inakoshi 2021), as the ERIC system code is proprietary.
Result: We observe that as the number of classes increases there is a drop in the fidelity (Fid.) and accuracy (Acc.) in general. This is because as more classes are introduced, more kernels in the CNN have to learn representations for those classes and hence there is more loss in the binarization of the kernels to obtain the rule-set. Our NeSy model generated for both NeSyFOLD (NF) and NeSyFOLD-EBP (NF-E) outperforms ERIC on all of these datasets w.r.t. accuracy and fidelity. All models perform poorly on the P10 dataset. We believe that this is because of the fewer distinct edges in the images in the P10 dataset which makes it harder for even the CNN kernels to learn good representations. GT43 has well-defined edges and consequently the accuracy and

Data	Algo	Fid.	Acc.	Pred.	Size
P2	ERIC	0.89 ± 0.01	0.89 ± 0.01	11 ± 1	52 ± 8
	NF	$\mathbf{0.93 \pm 0.01}$	$\mathbf{0.92 \pm 0.01}$	16 ± 2	28 ± 5
	NF-E	$\mathbf{0.93 \pm 0.01}$	$\mathbf{0.92 \pm 0.01}$	$\mathbf{8 \pm 2}$	$\mathbf{12 \pm 5}$
P3.1	ERIC	0.82 ± 0.01	0.81 ± 0.01	33 ± 4	118 ± 13
	NF	0.85 ± 0.03	0.84 ± 0.03	28 ± 6	49 ± 9
	NF-E	$\mathbf{0.87 \pm 0.02}$	$\mathbf{0.86 \pm 0.02}$	$\mathbf{10 \pm 3}$	$\mathbf{16 \pm 5}$
P3.2	ERIC	-	0.90	33	127
	NF	$\mathbf{0.94 \pm 0.0}$	$\mathbf{0.92 \pm 0.0}$	16 ± 4	26 ± 7
	NF-E	0.92 ± 0.01	0.91 ± 0.01	$\mathbf{6 \pm 1}$	$\mathbf{7 \pm 1}$
P3.3	ERIC	-	0.75	44	176
	NF	$\mathbf{0.83 \pm 0.01}$	$\mathbf{0.79 \pm 0.01}$	32 ± 5	60 ± 11
	NF-E	0.82 ± 0.03	0.78 ± 0.03	$\mathbf{9 \pm 0}$	$\mathbf{23 \pm 4}$
P5	ERIC	0.65 ± 0.01	0.63 ± 0.01	57 ± 4	171 ± 10
	NF	0.67 ± 0.03	0.64 ± 0.03	56 ± 3	131 ± 10
	NF-E	$\mathbf{0.70 \pm 0.02}$	$\mathbf{0.67 \pm 0.02}$	$\mathbf{12 \pm 4}$	$\mathbf{30 \pm 8}$
P10	ERIC	0.39 ± 0.01	0.36 ± 0.01	85 ± 6	208 ± 16
	NF	0.46 ± 0.02	0.42 ± 0.02	84 ± 6	131 ± 11
	NF-E	$\mathbf{0.49 \pm 0.02}$	$\mathbf{0.44 \pm 0.01}$	$\mathbf{36 \pm 4}$	$\mathbf{65 \pm 9}$
GT43	ERIC	0.73 ± 0.01	0.73 ± 0.01	232 ± 3	626 ± 28
	NF	0.75 ± 0.04	0.75 ± 0.04	206 ± 28	418 ± 79
	NF-E	$\mathbf{0.85 \pm 0.13}$	$\mathbf{0.85 \pm 0.13}$	$\mathbf{58 \pm 14}$	$\mathbf{99 \pm 28}$
MS	ERIC	0.70 ± 0.01	0.72 ± 0.01	71 ± 4	211 ± 15
	NF	0.78 ± 0.02	0.75 ± 0.02	63 ± 8	120 ± 19
	NF-E	$\mathbf{0.80 \pm 0.03}$	$\mathbf{0.78 \pm 0.03}$	$\mathbf{20 \pm 4}$	$\mathbf{36 \pm 9}$

Table 1: Comparison of ERIC vs NeSyFOLD (NF) vs NeSyFOLD-EBP (NF-E). **MS** shows the average value for each evaluation metric for each framework.

fidelity of all ERIC, NF and NF-E are high.

The size of the rule-set generated by NF is always smaller in size than ERIC. Moreover, NF-E generates an even smaller rule-set without compromising on accuracy or fidelity. Recall, the smaller the rule-set size the better the interpretability and lesser the manual semantic labeling effort. Since the number of Elite kernels is relatively small and defined for EBP, a smaller number of kernels learn tighter representations for each class. Consequently, there is less loss of information in binarization and the FOLD-SE-M algorithm finds a rule-set with a small number of predicates. Hence, using sparse kernel learning techniques such as EBP improves the scalability of NeSyFOLD.

[Q3/Q4] Semantic labeling Efficacy: To study the efficacy of the semantic labeling of the predicates in the rule-set using Alg. 1, we selected the P3.1, P3.2 and P3.3 datasets from the previous experiment. These datasets were chosen because NF and NF-E show decent accuracy and generate a comparatively smaller rule-set.

Setup: The *ADE20k* (Zhou et al. 2017b) dataset has manually annotated semantic segmentation masks of a few images of all the classes in P3.1, P3.2 and P3.3. We used these as input to the semantic labeling algorithm (Alg. 1).

It is important that the kernel associated with a predicate adheres to the semantic label of the predicate. If this is not the case then the interpretability of the rule-set diminishes significantly. To quantify *adherence* of a kernel to the concepts that its corresponding predicate has been labeled with, we obtained the unlabeled rule-set for 1 run of NeSyFOLD and NeSyFOLD-EBP on each of the datasets and labeled the rule-sets with 4 different values $\{0.05, 0.1, 0.15, 0.2\}$ of *margin* hyperparameter. We selected *top-10* images according to their feature-map norms for each kernel. Subsequently, for test set images where the NeSy model predicts correctly, we identified the activated rule and extracted the "True" predicates. We then manually scrutinize if any concepts from these predicates' labels are discernible within the image masked by the resized feature map originating from the relevant kernel. If even one concept in the label is adequately visible through the masked image we count that as positive. We say that the kernel *adheres* to its given label for this image. Finally, we calculated the average percentage adherence of the kernels for each dataset, for all the 4 *margin* values and summarize the comparison between NeSyFOLD and NeSyFOLD-EBP in Fig. 3.

Result: Recall that the *margin* hyperparameter controls the level of approximation of the semantic labeling. We observe that as the *margin* value is increased, the average percentage adherence overall, as well as for individual datasets, increases. This is intuitive because as the *margin* value is increased the number of concepts appearing in the semantic label of each predicate also increases (Fig. 4). Hence, it is more likely that a kernel adheres to a given image if its corresponding predicate has more concepts in its semantic label.

Note that the kernels of the CNN trained with EBP show higher average percentage adherence overall. This is expected because the kernels learn tighter representations hence generating smaller receptive fields which focus on very few and specific concepts.

Figure 4 shows the average number of concepts appearing in the semantic label of the predicate as the *margin* hyperparameter is varied. As expected, for NeSyFOLD-EBP the number of concepts is lower for every value of *margin*. Also, the variation in the number of concepts is lesser.

Ideally, the number of concepts in the labels of the predicates should be low while the percentage adherence should be high for having interpretability with high confidence. NeSyFOLD-EBP shows promising results in this regard. In Figure 5 on the top we show a rule-set obtained using NeSyFOLD-EBP on the P3.1 dataset with a *margin* of 0.05. To determine the class of a given image "img" we obtain its binarized vector, list it as facts in s(CASP) and run the query: `?- target(img, X).` The s(CASP) interpreter checks the rules from the top and returns a model that satisfies the ASP rule-set. On the bottom of Figure 5 we show a justification that can be obtained from the s(CASP) interpreter for the classification if the first rule (red) was the one that fired.

Related Work

There have been efforts since the 1990s to extract knowledge from a neural network. Andrews et al. (Andrews, Diederich, and Tickle 1995) have classified these efforts into

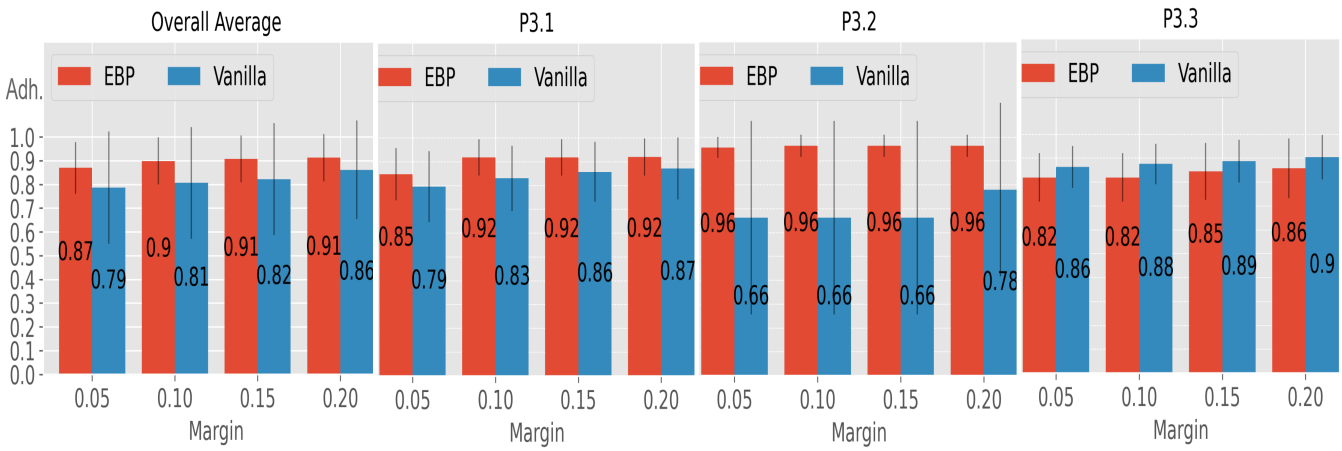

Figure 3: Percentage Adherence plots for kernels in NeSyFOLD-EBP (red) and NeSyFOLD (blue)

three kinds: (i) *pedagogical* methods wherein rules are constructed to explain the output in terms of the input, (ii) *decompositional* methods which extract rule sets specific to different parts of the network, and (iii) *eclectic* methods that are a combination of both. Both NeSyFold and ERIC are decompositional methods.

There is significant 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. (Zeiler and Fergus 2014) and Zhou et al. (Zhou et al. 2016) use the output activation while others (Selvaraju et al. 2017; Denil, Demiraj, and de Freitas 2014; Simonyan, Vedaldi, and Zisserman 2013) use gradients to find the mapping. Unlike NeSyFold, these visualization methods do not generate any rule-set. Zeiler et al. (Zeiler and Fergus 2014) use similar ideas to analyze what specific kernels in the CNN are invoked. There are fewer existing publications on methods for modeling relations between the various important features and generating explanations from them. Ferreira et al. (Ferreira et al. 2022) use multiple mapping networks that are trained to map the activation values of the main network's output to the human-defined concepts represented in an induced logic-based theory. Their method needs multiple neural networks besides the main network that the user has to provide.

Qi et al. (Qi, Khorram, and Fuxin 2021) 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. (Chen et al. 2019) 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 kernels. We do not make such an assumption. Zhang et al. (Zhang et al. 2017, 2018) 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. (Bologna and Fossati 2020) extract propositional rules from CNNs. Their system operates at the neuron level, while both ERIC and NeSyFold work with groups of neurons.

Some other works (Sen et al. 2022; Evans and Grefenstette 2018; Shindo, Nishino, and Yamamoto 2021) use a neurosymbolic system to induce logic rules from data. These systems belong to the Neuro:Symbolic $\rightarrow$ Neuro category whereas ours belongs to the Neuro;Symbolic category.

Conclusion and Future Work

In this paper, we have shown that our NeSyFOLD framework brings interpretability to the image classification task using CNNs. The FOLD-SE-M algorithm cuts-down on the size of rule-set generated significantly. We further show that the semantic labeling algorithm we propose which uses manually annotated semantic segmentation masks of a few images from the *ADE20k* dataset, leads to highly interpretable rule-sets. We acknowledge that the semantic segmentation masks of images may not be readily available depending on the domain, but tools such as SegGPT (Wang et al. 2023) can be used to obtain the segmentation masks. Else, the semantic labeling of the predicates has to be done manually. Our NeSyFOLD framework helps in this regard as well, as it decreases the number of predicates that need to be labeled. Another option is to find the *top-m* images from the train set according to feature-map norms for each kernel and then obtain semantic segmentation masks for only these images. Notice in our experiment we only used *top-10* from the available images. Moreover, we show that using the sparse kernel learning techniques such as EBP, the rule-set size and consequently number of predicates to be labeled can be further reduced. This also increases interpretability and also saves a lot of manual labeling effort if needed. We plan to evaluate our framework with more sparse kernel learning techniques in the future.

As the number of classes increases, the loss in accuracy also increases due to the binarization of more kernels. Nevertheless, NeSyFOLD outperforms the current SOTA, ERIC

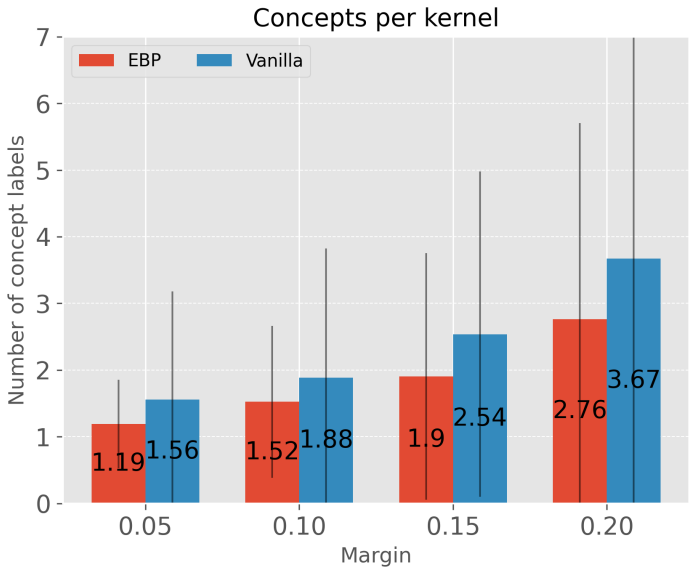

Figure 4: The average number of concept labels for kernels in NeSyFOLD-EBP (red) and NeSyFOLD (blue).

```
RULE-SET:
1. target(X,'kitchen') :- cabinet2(X), not
       ab1(X).
2. target(X,'bedroom') :- bed1(X).
3. target(X,'bathroom') :- sink1_toilet1(X).
4. target(X,'bathroom') :- wall2(X).
5. target(X,'kitchen') :- cabinet1(X).
6. target(X,'bedroom') :- bed2(X).
7. target(X,'bathroom') :- wall1(X).
8. target(X,'kitchen') :-
       work_surface1_kitchen_island1(X).
9. ab1(X) :- not cabinet1(X), bed1(X).
```

```
JUSTIFICATION:
'target' holds (for img, and kitchen),
because 'cabinet2' holds (for img), and
there is no evidence that
'ab1' holds (for img),
because there is no evidence that
'cabinet1' holds (for img) and
'bed1' holds (for img).
MODEL:
{target(img,kitchen), cabinet2(img),
not ab1(img), not cabinet1(img), bed1(img)}
```

Figure 5: A rule-set generated by NeSyFOLD-EBP for P3 dataset (top). Justification generated by s(CASP) for an image "img" for which the first rule was fired (bottom).

in terms of accuracy, fidelity, and rule-set size. We plan to explore end-to-end training of the CNN with the rules generated so that this loss in binarization can be reduced during training itself.

The rules generated by NeSyFOLD are arranged based on the coverage of training images, forming a decision list. Consequently, the topmost rules capture crucial class distinctions. Biases may be apparent in these rules, making them ideal for bias detection. Since the fidelity of the NeSy model is high (at least for a small number of classes), experts can review and suggest data changes for training and it is highly likely the retrained model will have less bias. It is noteworthy that interpreting a prediction made by the NeSy model becomes easier because of the justification that can be obtained from s(CASP). Hence, having even a large number of rules is less of a problem for an expert scrutinizing a particular prediction of the NeSy model.

In future, we plan to use NeSyFOLD for real-world tasks such as interpretable breast cancer prediction. Another interesting application is to classify images of unseen classes by constructing rules having predicates from an existing rule-set. This would alleviate the problem of retraining a CNN for classes with slight variations from the current learnt features, e.g., classifying beach images by simply writing a rule `target(X,'beach') :- sand(X), water(X).` where `sand` and `water` are pre-learnt predicates.

Acknowledgments

Authors are supported by US NSF Grants IIS 1910131 and IIP 1916206, US DoD, grants from industry.

References

Andrews, R.; Diederich, J.; and Tickle, A. B. 1995. Survey and critique of techniques for extracting rules from trained

artificial neural networks. *Knowledge-Based Systems*, 8(6): 373–389. Knowledge-based neural networks.

Arias, J.; Carro, M.; Salazar, E.; Marple, K.; and Gupta, G. 2018. Constraint Answer Set Programming without Grounding. *Theory and Practice of Logic Programming*, 18(3-4): 337–354.

Bologna, G.; and Fossati, S. 2020. A two-step rule-extraction technique for a cnn. *Electronics*, 9(6): 990.

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

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

Denil, M.; Demiraj, A.; and de Freitas, N. 2014. Extraction of Salient Sentences from Labelled Documents. *ArXiv*, abs/1412.6815.

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

Ferreira, J.; de Sousa Ribeiro, M.; Gonçalves, R.; and Leite, J. 2022. Looking Inside the Black-Box: Logic-based Explanations for Neural Networks. In *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning*, 432–442.

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

Kanagaraj, N.; Hicks, D.; Goyal, A.; Mishra Tiwari, S.; and Singh, G. 2021. Deep learning using computer vision in self driving cars for lane and traffic sign detection. *International Journal of System Assurance Engineering and Management*, 12.

Kasioumis, T.; Townsend, J.; and Inakoshi, H. 2021. Elite BackProp: Training Sparse Interpretable Neurons. In *International Workshop on Neural-Symbolic Learning and Reasoning*.

Kingma, D. P.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Ko, B.; and Kwak, S. 2012. Survey of computer vision-based natural disaster warning systems. *Optical Engineering*, 51: 0901–.

Lage, I.; Chen, E.; He, J.; Narayanan, M.; Kim, B.; Gershman, S. J.; and Doshi-Velez, F. 2019. Human evaluation of models built for interpretability. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 7, 59–67.

LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; and Jackel, L. D. 1989. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4): 541–551.

Qi, Z.; Khorram, S.; and Fuxin, L. 2021. Embedding Deep Networks Into Visual Explanations. *Artificial Intelligence*, 292: 103435.

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

Sen, P.; de Carvalho, B. W.; Riegel, R.; and Gray, A. 2022. Neuro-symbolic inductive logic programming with logical neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36(8), 8212–8219.

Shindo, H.; Nishino, M.; and Yamamoto, A. 2021. Differentiable inductive logic programming for structured examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35(6), 5034–5041.

Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2013. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *CoRR*, abs/1312.6034.

Stallkamp, J.; Schlipsing, M.; Salmen, J.; and Igel, C. 2012. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32: 323–332. Selected Papers from IJCNN 2011.

Sun, W.; Zheng, B.; and Qian, W. 2016. Computer aided lung cancer diagnosis with deep learning algorithms. In *SPIE Medical Imaging*.

Townsend, J.; Kasioumis, T.; and Inakoshi, H. 2021. ERIC: Extracting Relations Inferred from Convolutions. In Ishikawa, H.; Liu, C.-L.; Pajdla, T.; and Shi, J., eds., *Computer Vision – ACCV 2020*, 206–222. Cham: Springer International Publishing. ISBN 978-3-030-69535-4.

Townsend, J.; Kudla, M.; Raszkowska, A.; and Kasiousmis, T. 2022. On the Explainability of Convolutional Layers for Multi-Class Problems. In *Combining Learning and Reasoning: Programming Languages, Formalisms, and Representations*.

Wang, H.; and Gupta, G. 2024. FOLD-SE: An Efficient Rule-based Machine Learning Algorithm with Scalable Explainability. Proc. Symposium on Practical Aspects of Declarative Languages. Forthcoming.

Wang, X.; Zhang, X.; Cao, Y.; Wang, W.; Shen, C.; and Huang, T. 2023. SegGPT: Segmenting Everything In Context. *ArXiv*, abs/2304.03284.

Xiao, J.; Hays, J.; Ehinger, K. A.; Oliva, A.; and Torralba, A. 2010. SUN database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 3485–3492.

Xie, N.; Sarker, M. K.; Doran, D.; Hitzler, P.; and Raymer, M. 2017. Relating Input Concepts to Convolutional Neural Network Decisions. *ArXiv*, abs/1711.08006.

Zeiler, M. D.; and Fergus, R. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, 818–833. Springer.

Zhang, Q.; Cao, R.; Shi, F.; Wu, Y. N.; and Zhu, S.-C. 2018. Interpreting CNN knowledge via an explanatory graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32(1).

Zhang, Q.; Cao, R.; Wu, Y. N.; and Zhu, S.-C. 2017. Growing interpretable part graphs on convnets via multi-shot

learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31(1).

Zhou, B.; Khosla, A.; Lapedriza, À.; Oliva, A.; and Torralba, A. 2014. Object Detectors Emerge in Deep Scene CNNs. *CoRR*, abs/1412.6856.

Zhou, B.; Khosla, A.; Lapedriza, A.; Oliva, A.; and Torralba, A. 2016. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2921–2929.

Zhou, B.; Lapedriza, A.; Khosla, A.; Oliva, A.; and Torralba, A. 2017a. Places: A 10 million Image Database for Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Zhou, B.; Zhao, H.; Puig, X.; Fidler, S.; Barriuso, A.; and Torralba, A. 2017b. Scene Parsing through ADE20K Dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5122–5130.