Using Logic Programming and Kernel-Grouping for Improving Interpretability of Convolutional Neural Networks

Parth Padalkar[1][0000−0003−1015−0777], Gopal Gupta[1][0000−0001−9727−0362]

[1]The University of Texas at Dallas, Richardson, USA
[1]{parth.padalkar, gupta}@utdallas.edu

Abstract. Within the realm of deep learning, the interpretability of Convolutional Neural Networks (CNNs), particularly in the context of image classification tasks, remains a formidable challenge. To this end we present a neurosymbolic framework, NeSyFOLD-G that generates a symbolic rule-set using the last layer kernels of the CNN to make its underlying knowledge interpretable. What makes NeSyFOLD-G different from other similar frameworks is that we first find groups of similar kernels in the CNN (kernel-grouping) using the cosine-similarity between the feature maps generated by various kernels. Once such kernel groups are found, we binarize each kernel group's output in the CNN and use it to generate a binarization table which serves as input data to FOLD-SE-M which is a Rule Based Machine Learning (RBML) algorithm. FOLD-SE-M then generates a rule-set that can be used to make predictions. We present a novel kernel grouping algorithm and show that grouping similar kernels leads to a significant reduction in the size of the rule-set generated by FOLD-SE-M, consequently, improving the interpretability. This rule-set symbolically encapsulates the connectionist knowledge of the trained CNN. The rule-set can be viewed as a *normal logic program* wherein each predicate's truth value depends on a kernel group in the CNN. Each predicate in the rule-set is mapped to a concept using a few semantic segmentation masks of the images used for training, to make it human-understandable. The last layers of the CNN can then be replaced by this rule-set to obtain the NeSy-G model which can then be used for the image classification task. The goal directed ASP system s(CASP) can be used to obtain the justification of any prediction made using the NeSy-G model. We also propose a novel algorithm for labeling each predicate in the rule-set with the semantic concept(s) that its corresponding kernel group represents.

Keywords: CNN · Neurosymbolic AI · Normal Logic Programs · Rule-Based Machine Learning · Interpretable Image Classification.

1 Introduction

Interpretability of deep learning models is an important issue that has resurfaced in recent years as these models have become larger and are being applied

to an increasing number of tasks. Some applications such as autonomous vehicles [9], disease diagnosis [25], and natural disaster prevention [11] 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. [14]. 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 [34,33]. There is no way to tell whether the predictions made by the model are based on concepts meaningful to humans, or are simply the outcome of coincidental correlations. If the knowledge of the trained CNN becomes interpretable then domain experts can scrutinize this knowledge and point out any biases or spurious correlations that the CNN might have learnt which could lead to wrong predictions. Thus retraining with better and more targeted data can be suggested by the experts.

We propose a framework for interpretable image classification using CNNs called NeSyFOLD-G. A CNN, like any deep neural network is composed of multiple layers. We focus on the convolution layer, more specifically the last convolution layer of a CNN in this work. The convolution layer is composed of kernels.

A kernel, also known as a filter, is a 2D matrix. It acts like a small, specialized magnifying glass that slides over an image to help recognize specific features or patterns in the image, like edges, curves, or textures. It does this by multiplying its values with the pixel values of the image in a small region, and then it adds up those products. This process helps highlight important parts of the image. As the kernel slides over the entire image, it creates a new, simplified version of the image that emphasizes the patterns it's looking for. This simplified version is called a feature map. The CNN then uses these feature maps to understand the image and make predictions.

The NeSyFOLD-G framework can be used to create a *NeSy-G* model which is a composition of the CNN and a rule-set generated from kernels in its last convolution layer. A Rule Based Machine Learning (RBML) algorithm called FOLD-SE-M [28] is used for generating the rule-set by using binarized outputs of the groups of similar kernels in a trained CNN. The rule-set is a default theory represented as a normal logic program, [16] i.e., Prolog extended with negation-as-failure. The binarized output (0/1) of the kernel groups influences the truth value of the predicates appearing in the rule body. The rule-set can also be viewed as a stratified Answer Set Program and the s(CASP) [1] ASP system can be used to obtain justifications of the predictions made by the NeSy-G model. The rule-set also serves as a global explanation for the predictions made by the CNN.

Our first novel contribution is the *kernel grouping algorithm* that finds groups of similar kernels in the CNN based on the cosine similarity score of their corresponding generated feature maps.

We also introduce a semantic labelling algorithm that can be used to label the predicates in the rule-set with the semantic concept(s) that their corresponding kernel groups represent in the images. For example, the predicate `52(X)` corresponding to kernel group `52` in the last convolution layer of the CNN will be replaced by `bathtub(X)` in the rule-set, if kernel group `52` has learnt to look for "bathtubs" in the image. Fig. 1 illustrates the NeSyFOLD-G framework.

Padalkar et al. proposed the NeSyFOLD framework [17] which shares similarities with the NeSyFOLD-G framework. The major difference that separates NeSyFOLD-G from NeSyFOLD is that the truth values of predicates in the generated rule-set is influenced by the binarized output of *groups* of similar kernels. In NeSyFOLD each predicate's truth value is influenced by single kernels in the CNN. However, it is known that groups of kernels in the last layer are responsible for representing a single concept. Yang et al. [30] proposed an attention-based masking mechanism for finding the concept learnt by a single kernel by accounting for the other kernels with similar attention weights. Their approach serves as motivation behind our kernel grouping algorithm that uses the cosine similarity score between feature maps of various kernels to find the groups of similar kernels.
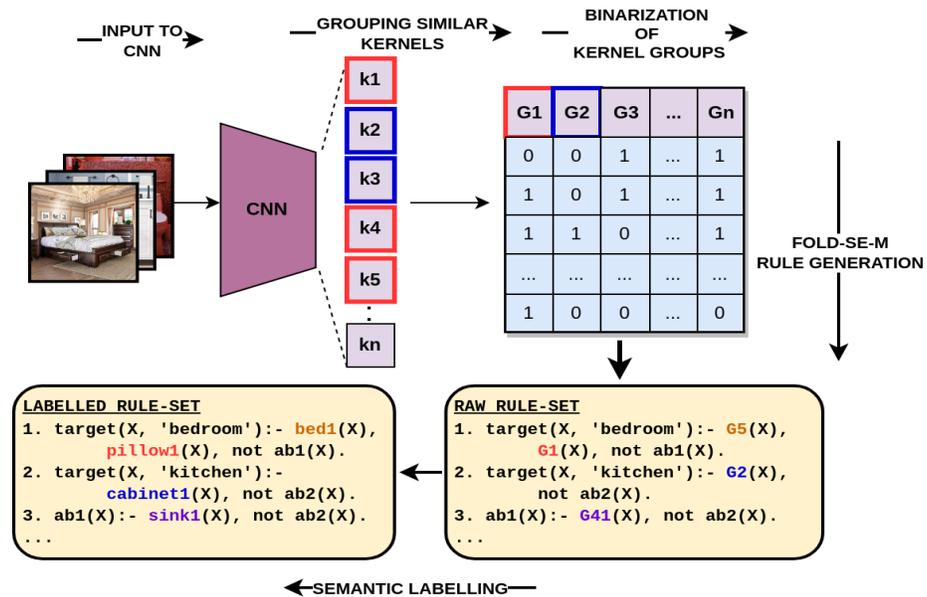

Fig. 1. The NeSyFOLD-G framework. Each kernel group is depicted with a unique color in the rule-set.

The size of the rule-set generated can be used as a metric for interpretability. Lage et al. [12] comprehensively showed through human evaluations that as

the size of the rule-set increases the difficulty in interpreting the rule-set also increases. Padalkar et al. show that NeSyFOLD framework generates a smaller rule-set than the ERIC system [26] which was the previous SOTA. We show that NeSyFOLD-G, achieves a significant reduction in the size of the rule-set generated while maintaining or improving on the accuracy and fidelity in comparison to the NeSyFOLD framework.

To summarize, our contributions are as follows:

1. We present a novel kernel grouping algorithm that constitutes the heart of the NeSyFOLD-G framework for improving interpretability of the generated rule-set.
2. We also introduce a semantic labeling algorithm for labeling the predicates of the rule-set generated by the NeSyFOLD-G framework.

2 Background

FOLD-SE-M: The FOLD-SE-M algorithm [28] 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:

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

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 [8]. A default of the form:

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

can be formalized as the normal logic programming rule:

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

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

$$bird(X) : M\neg penguin(X) \over flies(X)$$

will be represented as the following default rule in normal logic programming:
```
flies(X) :- bird(X), not penguin(X).
```
We call $\texttt{bird(X)}$, the condition that allows us to jump to the default conclusion that X flies, the *default part* of the rule, and $\texttt{not penguin(X)}$ the *exception part* of the rule.

FOLD-SE-M [28] is a Rule Based Machine Learning (RBML) algorithm. It generates a rule-set from tabular data, comprising rules in the form described

4

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 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.

3 Learning

In this section we describe the process of generating a rule-set from the CNN and obtaining the NeSy-G model. We start by training the CNN on the input images for the given image classification dataset. Any optimization technique can be used for updating the weights. Fig. 1 illustrates the learning pipeline.

Binarization: Once the CNN has been fully trained to convergence, we pass the full training set consisting of n images to the CNN. For each image i in the training set, let $A_{i,k}$ denote the feature map generated by 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. To convert each of the feature maps to a single value we take the norm of the feature maps as demonstrated by eq. (2) to obtain $a_{i,k}$.

Kernel grouping algorithm: We then find the groups of similar kernels in the CNN. Consider a kernel $\hat{k}$ for which we need to identify the most similar kernels. We do this by first finding the *top-10* images $\hat{i}_1, \hat{i}_2, ..., \hat{i}_{10}$ that activate $\hat{k}$ the most, according to the norm values of the feature maps generated by $\hat{k}$ for these images. Now, we compute the cosine similarity score between $A_{\hat{i}_g, \hat{k}}$ and $A_{\hat{i}_g, k'}$, where g $\in [1, 10]$ and k' is some kernel in the last layer the last layer of the CNN. The similarity score of kernel k' w.r.t $\hat{k}$ is calculated by taking the mean of the cosine similarity scores for all the *top-10* images $\hat{i}_1, \hat{i}_2, ..., \hat{i}_{10}$ as $sim_{\hat{k}, k'}$. The similarity score is a value between 0 and 1. Thus, we calculate the similarity score of all kernels in the last layer of the CNN w.r.t to $\hat{k}$. The group of kernel $\hat{k}$ would then constitute of all kernels that have a similarity score w.r.t $\hat{k}$ greater than a user-defined similarity threshold θ_s.

Hence, we find a group of similar kernels G_k for all the kernels k in the last layer of the CNN. Note that the total number of kernel groups G_k is the same as the total number of kernels in the last layer of the CNN.

Next, for each kernel group G_k we obtain the group norm a_{i,G_k} for each image i in the training set. This is achieved by taking the mean of the norms corresponding to each kernel in G_k for each image i. This leads to the creation of a table T_G with each row representing an image and each column representing the group norm for each of the kernel groups G_k.

Finally, for each kernel group we convert the group norm values to either 0 or 1 which symbolizes the kernel group "activating" or "deactivating" for each image. This is called *binarization* of the kernel groups. This is done by determining an appropriate threshold θ_{G_k} for each kernel group G_k to binarize its output. The threshold θ_{G_k} is calculated as a weighted sum of the mean and the standard deviation of the group norms a_{i,G_k} for all images i in the training set, denoted by eq. (3) where α and γ are user-defined hyperparameters.

Thus a binarization table B_G is created. Each row in the table represents an image and each column is the binarized kernel group value represented by either a 0 if $a_{i,G_k} \leq \theta_{G_k}$ or 1 if $a_{i,G_k} > \theta_{G_k}$ (*cf.* Fig. 1 (right)).

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

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

Rule-set Generation: The binarization table B_G is given as an input to the FOLD-SE-M algorithm to obtain a rule-set in the form of a normal logic program. The FOLD-SE-M algorithm finds the most influential features in the B_G and generates a rule-set that has these features as predicates. Since B_G has features as kernel group ids, the raw rule-set has predicates with names in the form of their corresponding kernel group's id. 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 group 3 is not activated and kernel group 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 group would determine the truth value of its predicate in the rule-set. The rule-set generated is in the form of a decision list, i.e., the next rule is checked only if the current rule and all the rules above it were not satisfied.

Semantic labeling: Groups of kernels activate in synergy to identify concepts in the CNN. Since we capture the outputs of the kernel groups as truth values of predicates in the rule-set, we can label the predicates with the semantic concept(s) that the corresponding kernel group has learnt. Thus, the same example rule from above may now look like:

```
target(X,'bathroom') :- not bed(X), bathtub(X), not ab1(X).
```
We introduce a novel semantic labelling algorithm to automate the semantic labelling of the predicates in the rule-set generated. The details of the algorithm are discussed later.

6

The NeSy-G model is conceptualized as the model obtained after replacing all the layers following the last convolutional layer with the rule-set generated by applying the FOLD-SE-M algorithm on the binarization table B_G.

4 Inference

For using the NeSy-G model to obtain predictions on the test set, we first obtain the kernel feature maps for each kernel in the last convolutional layer. Then, we compute the group norms for all kernel groups that were found in the learning process to obtain the table $T_{G_k}^{test}$. From $T_{G_k}^{test}$ we obtain the binarization table B_G^{test} by binarizing the output of each kernel group in $T_{G_k}^{test}$ by using the threshold θ_{G_k} calculated in the learning phase. Next, for each binarized vector b in B_G^{test}, we use the labeled/unlabelled rule-set obtained in the learning phase to make predictions. The truth value of the predicates in the rule-set is determined by the corresponding binarized kernel group values in b. FOLD-SE-M toolkit's built-in rule-interpreter can be used to obtain the predicted class of b given the rule-set. The binarized kernel group values in b can also be listed as facts and the rule-set which can be viewed as a stratified answer set program, can be queried with the s(CASP) interpreter [1] to obtain the justification as well as the target class. Note that s(CASP) searches for the answer set in a goal directed manner, which implies that the rules are checked from the top to the bottom one by one. Hence, the first answer set that is found to satisfy the rule-set with the given facts entails the intended prediction made by the NeSy-G model.

5 Semantic Labelling of Predicates

The raw rule-set generated by FOLD-SE-M initially has kernel group ids as predicate names. Also, since the FOLD-SE-M algorithm finds only the most influential kernel groups, the number of kernel groups that actually appear in the rule-set is usually very low in comparison to the total number of kernel groups. We present a novel algorithm for automatically labelling the corresponding predicates of the kernel groups with the semantic concept(s) that the kernel groups represent.

Xie et al. [29] showed that each kernel in the CNN may learn to represent multiple concepts in the images. Hence each kernel group may also represent multiple concepts. As a result, we assign semantic labels to each predicate, denoting the names of the semantic concepts learnt by the corresponding kernel group. 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 labelling of a given predicate. The algorithm requires a dataset that has semantic segmentation masks of the training images. This essentially means that for every image i in the dataset I, there is an image i_M where every pixel is annotated with the label of the object (concept) that it belongs to (Fig. 2 middle). We denote these by I_M.

The CNN that is trained on the training set is used to obtain the norms $a_{i,k}$ of the feature maps $A_{i,k}$ generated by each kernel k in the last convolution layer. Next, as the respective kernel groups for each kernel are known, the table $T_{G_k}^{I_m}$ is created where each row represents the images whose corresponding semantic segmentation masks are available and the columns are the kernel group norms.

Now, consider some kernel group $G_{\hat{k}}$ that has l kernels in the group namely, $\hat{k}_1, \hat{k}_2, ..., \hat{k}_l$. The top-m images $i'_1, i'_2, ..., i'_m \in I'_m$, according to the group norm values are selected. We need to calculate the group's $Intersection\ over\ Union$ (IoU_c) score for each concept c visible in the top-m images that most activate the group. Then according to this score for each concept c, the label of the kernel group's predicate should comprise of the top concepts that the kernel group is detecting.

$$IoU_c(i^{Mask}, i) = \frac{\text{no. of non-zero pixels in } c \cap i}{\text{no. of non-zero pixels in } i} \tag{4}$$

Fig. 2. The calculation of mean IoU_c scores for a kernel.

For a given image $i_j \in I'_m$, the resized feature map generated by every kernel in the kernel group is used to mask the image to obtain $i_j^{\hat{k}_1}, i_j^{\hat{k}_2}, ..., i_j^{\hat{k}_l}$. Fig. 2 (top) shows a few images masked with the resized feature maps generated by a kernel. For each of these masked images, the IoU_c score is calculated using eq. (4) for each concept c, that appears in the corresponding semantic segmentation mask i_j^{Mask} of the image i_j. Fig. 2 (middle) shows the semantic segmentation masks of the images at the top. Next, each kernel's IoU_c score for all the top-m images, for all concepts c is calculated. Each kernel's mean IoU_c score is calculated by

taking the mean score over all images. Finally, the kernel group's IoU_c score is calculated by taking the mean of the mean IoU_c score of each kernel for each concept c.

The algorithm can be summarized as follows:

1. For a given kernel group, find the *top-m* images according to its group kernel norm value.
2. For each kernel in the kernel group find the IoU_c score for each of the *top-m* images.
3. Calculate the mean IoU_c score for each kernel over all images.
4. Calculate the mean of the mean IoU_c score for each kernel to obtain the kernel Group's IoU_c score.

Fig 2 illustrates the IoU_c scores calculation for a single kernel.

The label of the corresponding predicate of a kernel group is chosen as the set of concepts that have their *normalized IoU_c* score in a certain "margin" from the top concept. This is controlled using the user-defined *margin* hyperparameter.

For example, if the IoU_c score for kernel group 12 is {*cabinets* : 0.5, *door* : 0.4, *drawer* : 0.1} then with a *margin* of 0.1 the label for the corresponding predicate will be "*cabinets*1_*door*1" since the concept *door* is in the 0.1 margin from the top concept *cabinets*. Note, each concept name in the label is appended with a unique numerical identifier (in this case 1), to distinguish it from the the other kernel groups that might learn the same concept. Say, if kernel group 25 is also detecting *cabinets* then its predicate's label would be "*cabinets*2_..." where ... denotes the other concepts that the kernel group 25 might be detecting.

6 Experiments and Results

Exp 1 (Setup): We compare the performance of NeSyFOLD-G framework with that of the NeSyFOLD framework on various datasets. We report the accuracy, fidelity, number of unique predicates in the rule-set, number of rules generated and the size of the rule-set. Size is calculated as the total number of predicates in the bodies of the rules that constitute the logic program generated by NeSyFOLD and NeSyFOLD-G.

We used a VGG16 CNN with pre-trained weights on the Imagenet dataset [4]. We trained for 100 epochs with a batch size of 32. We used the Adam [10] optimizer and applied class weights for imbalanced data. We also used $L2$ regularization of 0.005 on all layers and a learning rate of 5×10^{-7}. We used a decay factor of 0.5 and patience of 10 epochs. Also, we resized all images to 224×224. We used $\alpha = 0.6$ and $\gamma = 0.7$ for all the datasets. For this experiment, we used the *German Traffic Sign Recognition Benchmark* (GTSRB) [24], *MNIST* [15] and the Places [36] dataset.

The GTSRB dataset has 43 classes. Each class contains multiple instances of a physical signpost and multiple images of the signpost are provided. We used a $80 : 20$ training-validation split per class and used the provided test set to report the performance metrics of the models.

The MNIST dataset has 10 classes. Each class contains images of a handwritten digit from 0 to 9. We split the standard training set into train and validation set by using the last $10k$ images for the validation set. We used the provided test set to report the results.

The Places dataset has images of various scenes. 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 (PLACES3.1), then dining room, living room (PLACES5) and finally home office, office, waiting room, conference room and hotel room (PLACES10). We also selected 2 additional subsets of 3 classes each namely, {desert road, forest road, street} (PLACES3.2) and {desert road, driveway, highway} (PLACES3.3). We obtained the train and the test set by selecting $1k$ images from each class for the test set and the other $4k$ for the training set. We use the given validation set to tune our hyperparameters.

The NeSy-G model was created using the learning procedure described previously using the NeSyFOLD-G framework and the NeSy model was created using the NeSyFOLD framework as described in [17]. The comparison between NeSyFOLD-G and NeSyFOLD is drawn in Table 1. The accuracy and fidelity are reported on the test set. The results are reported after 5 runs on each dataset. Note, fidelity determines how closely a model follows the predictions of another model. Since the NeSy-G and NeSy models are created from the trained model they should show high fidelity w.r.t the CNN.

Data	Algo	Fid.	Acc.	Pred.	Rules	Size
PLACES2	NF	$\mathbf{0.93 \pm 0.01}$	0.92 ± 0.01	16 ± 2	12 ± 2	28 ± 5
	NF-G	$\mathbf{0.93 \pm 0.0}$	$\mathbf{0.93 \pm 0.0}$	$\mathbf{8 \pm 1}$	$\mathbf{7 \pm 1}$	$\mathbf{11 \pm 2}$
PLACES3.1	NF	0.85 ± 0.03	0.84 ± 0.03	28 ± 6	21 ± 4	49 ± 9
	NF-G	$\mathbf{0.87 \pm 0.01}$	$\mathbf{0.86 \pm 0.01}$	$\mathbf{20 \pm 7}$	$\mathbf{15 \pm 3}$	$\mathbf{31 \pm 9}$
PLACES3.2	NF	$\mathbf{0.94 \pm 0.0}$	$\mathbf{0.92 \pm 0.0}$	16 ± 4	13 ± 3	26 ± 7
	NF-G	$\mathbf{0.94 \pm 0.01}$	$\mathbf{0.92 \pm 0.01}$	$\mathbf{12 \pm 3}$	$\mathbf{10 \pm 1}$	$\mathbf{18 \pm 3}$
PLACES3.3	NF	$\mathbf{0.83 \pm 0.01}$	0.79 ± 0.01	32 ± 5	23 ± 3	60 ± 11
	NF-G	$\mathbf{0.83 \pm 0.01}$	$\mathbf{0.80 \pm 0.01}$	$\mathbf{30 \pm 2}$	$\mathbf{21 \pm 3}$	$\mathbf{53 \pm 6}$
PLACES5	NF	0.67 ± 0.03	0.64 ± 0.03	56 ± 3	52 ± 4	131 ± 10
	NF-G	$\mathbf{0.68 \pm 0.02}$	$\mathbf{0.65 \pm 0.02}$	$\mathbf{41 \pm 4}$	$\mathbf{34 \pm 6}$	$\mathbf{83 \pm 13}$
PLACES10	NF	0.23 ± 0.19	0.20 ± 0.17	$\mathbf{33 \pm 28}$	$\mathbf{32 \pm 27}$	$\mathbf{78 \pm 66}$
	NF-G	$\mathbf{0.33 \pm 0.17}$	$\mathbf{0.30 \pm 0.15}$	74 ± 39	73 ± 39	184 ± 97
GTSRB	NF	0.75 ± 0.04	0.75 ± 0.04	206 ± 28	134 ± 26	418 ± 79
	NF-G	$\mathbf{0.76 \pm 0.02}$	$\mathbf{0.76 \pm 0.02}$	$\mathbf{176 \pm 13}$	$\mathbf{98 \pm 11}$	$\mathbf{320 \pm 30}$
MNIST	NF	$\mathbf{0.91 \pm 0.01}$	$\mathbf{0.91 \pm 0.01}$	132 ± 9	90 ± 7	271 ± 25
	NF-G	0.90 ± 0.01	0.90 ± 0.01	$\mathbf{103 \pm 12}$	$\mathbf{79 \pm 10}$	$\mathbf{216 \pm 28}$

Table 1. Comparison NeSyFOLD (NF) vs NeSyFOLD-G (NF-G).

Exp 1 (Result): Table 1 clearly shows that the NeSy-G model outperforms NeSy model w.r.t accuracy and fidelity in most cases and is comparable otherwise. More importantly, the advantage of using the NeSyFOLD-G framework is apparent from the reduction in the number of predicates, number of rules and the overall size of the rule-set that is generated.

The reduction in size of the rule-set is a direct indication of the improved interpretability as pointed out by Lage et al. [12]. The main difference between the NeSyFOLD and the NeSyFOLD-G framework is the grouping of similar kernels in the latter. The grouped kernel forms better features in the binarization table that is generated after binarizing the group norms. The grouping helps in creating more informative features for the FOLD-SE-M algorithm to generate the rules from. Hence, in a fewer number of predicates and rules, (as compared to NeSyFOLD) the same information can be captured.

Note that as the number of classes increases as in the case of *PLACES2, PLACES3.1, PLACES3.2, PLACES3.3, PLACES5* and *PLACES10* both the models show a decrease in the accuracy and fidelity. This is because as the number of classes increases, more number of kernels are needed to represent the knowledge and consequently more kernels have to be binarized. Thus the loss incurred due to binarization of the kernels increases as the number of classes increases. Notice that for PLACES10 the size of the rule-set generated by NeSyFOLD-G is larger than that generated by NeSyFOLD. This is because for 2 out of the 5 runs, NeSyFOLD could not generate any rule-set as the FOLD-SE-M algorithm could not find good enough features in the binarization table. Due to the size of the training set being relatively large ($40k$ examples) and the large number of classes (10 classes), the loss due to binarization rapidly increases. This is also the reason why the accuracy and fidelity is very low. However, since NeSyFOLD-G uses kernel grouping, the FOLD-SE-M algorithm gets to work with better features in the binarization table and thus the accuracy and fidelity is much higher compared to NeSyFOLD and thus the rule-set size is also high on average. Although in 1 run NeSyFOLD-G also manages to find no rule-set that explains the predictions of the CNN.

Exp 2 (setup): We use the procedure described previously, for semantic labelling of the predicates in the rule-set generated. We use the *ADE20k* dataset [37] in our experiments. It provides manually annotated semantic segmentation masks for a few images of all the classes of the Places dataset. The GTSRB and MNIST datasets do not have any semantic segmentation masks available. Hence, for all the subsets of classes of the Places dataset reported in Table 1, we show the effect of using the semantic labelling algorithm described in Section 5. In Fig. 3 we have shown labelled rule-sets for the PLACES2, PLACES3.1 and PLACES3.2, PLACES3.3 and PLACES5 datasets. We used a *ratio* of 0.8 for all datasets, $tail : 5e^{-3}$ for PLACES2, PLACES3.1 and PLACES3.2, $tail : 1e^{-2}$ for PLACES3.3 and PLACES5 dataset. A similarity threshold θ_s of 0.8 was used for generating the rule-sets. We used a margin of 0.05 to label the raw rule-sets. We do not show the labelled rule-set for PLACES10 since the accuracy of the NeSy-G model is very low on the dataset.

```
RULE-SET 1:
1. target(X,'bedroom') :-
      not sink1_wall2_countertop1(X), not ab2(X).
2. target(X,'bathroom') :- wall1(X).
3. target(X,'bathroom') :- wall3(X).
4. target(X,'bathroom') :- not wall3(X),
      not bed2(X).
5. target(X,'bedroom') :- not bed2(X).
6. target(X,'bedroom') :- not wall3(X).
7. ab1(X) :- not wall4(X), bed4(X).
8. ab2(X) :- not bed3(X), not bed1(X), not bed5(X),
      not ab1(X).
```

```
RULE-SET 2:
1. target(X,'street') :- building5(X).
2. target(X,'forest_road') :- tree1(X),
      not ab1(X), not ab2(X).
3. target(X,'desert_road') :- sky1(X).
4. target(X,'desert_road') :- building7(X),
      not sky1(X), not building6(X).
5. target(X,'forest_road') :- not building3(X),
      not ab3(X).
6. target(X,'street') :- building1(X).
7. ab1(X) :- not tree3_building4(X), sky2(X).
8. ab2(X) :- building2(X), building8(X).
9. ab3(X) :- not tree2(X), not building7(X).
```

```
RULE-SET 3:
1.  target(X,'highway') :- not ground1(X),
       not ab3(X), not ab4(X).
2.  target(X,'driveway') :- house3_building1(X).
3.  target(X,'desert_road') :- not trees2_road5(X),
       not ab6(X).
4.  target(X,'highway') :- house3_building1(X).
5.  target(X,'highway') :- road9(X).
6.  target(X,'driveway') :- house1(X).
7.  target(X,'highway') :- not road9(X), not ab7(X).
8.  ab1(X) :- not road6(X), house3_building1(X).
9.  ab2(X) :- not house2(X), trees3(X), not ab1(X).
10. ab3(X) :- not road7(X), not ab2(X).
11. ab4(X) :- not road2(X), road1(X), not trees4(X).
12. ab5(X) :- not road8_car1(X), not trees1(X).
13. ab6(X) :- not sky2(X), not sky1(X), not ab5(X).
14. ab7(X) :- not road4(X), not road3(X).
```

```
RULE-SET 4:
1.  target(X,'living_room') :- sofa2(X), not bed1(X),
       sofa3(X), not sink2(X).
2.  target(X,'dining_room') :- cabinet1(X),
       wall4_cabinet3(X), cabinet2(X).
3.  target(X,'kitchen') :- floor1_chair1(X),
       not bed3(X).
4.  target(X,'bedroom') :- bed3(X),
       not wall2_kitchen_island1(X).
5.  target(X,'bathroom') :- wall3(X), not bed2(X).
6.  target(X,'living_room') :-
       armchair1_sofa1_wall1(X), not sink1(X),
       not ab1(X).
7.  ab1(X) :- wall2_kitchen_island1(X), sofa3(X).
```

Fig. 3. The labelled rule-sets generated by NeSyFOLD-G for PLACES2 (RULE-SET 1) , PLACES3.2 (RULE-SET 2), PLACES3.3 (RULE-SET 3) and PLACES5 (RULE-SET 4)
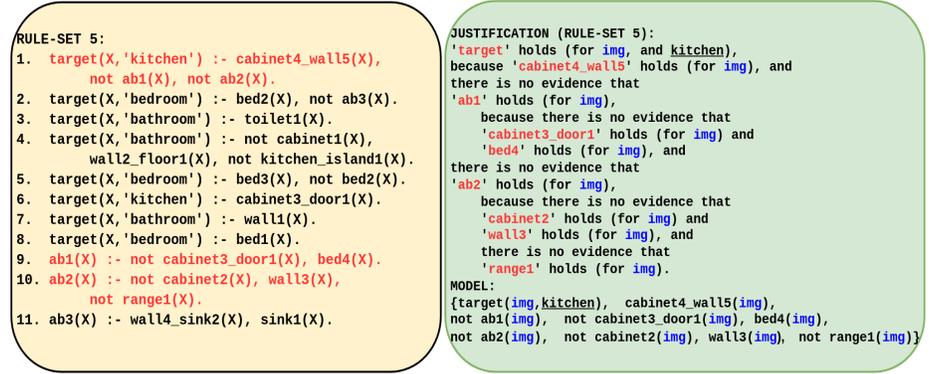
```
RULE-SET 5:
1.  target(X,'kitchen') :- cabinet4_wall5(X),
       not ab1(X), not ab2(X).
2.  target(X,'bedroom') :- bed2(X), not ab3(X).
3.  target(X,'bathroom') :- toilet1(X).
4.  target(X,'bathroom') :- not cabinet1(X),
       wall2_floor1(X), not kitchen_island1(X).
5.  target(X,'bedroom') :- bed3(X), not bed2(X).
6.  target(X,'kitchen') :- cabinet3_door1(X).
7.  target(X,'bathroom') :- wall1(X).
8.  target(X,'bedroom') :- bed1(X).
9.  ab1(X) :- not cabinet3_door1(X), bed4(X).
10. ab2(X) :- not cabinet2(X), wall3(X),
       not range1(X).
11. ab3(X) :- wall4_sink2(X), sink1(X).
```

```
JUSTIFICATION (RULE-SET 5):
'target' holds (for img, and kitchen),
because 'cabinet4_wall5' holds (for img), and
there is no evidence that
'ab1' holds (for img),
    because there is no evidence that
    'cabinet3_door1' holds (for img) and
    'bed4' holds (for img), and
there is no evidence that
'ab2' holds (for img),
    because there is no evidence that
    'cabinet2' holds (for img) and
    'wall3' holds (for img), and
    there is no evidence that
    'range1' holds (for img).
MODEL:
{target(img,kitchen),  cabinet4_wall5(img),
not ab1(img),  not cabinet3_door1(img), bed4(img),
not ab2(img),  not cabinet2(img), wall3(img), not range1(img)}
```

Fig. 4. The justification (right) obtained from s(CASP) for an image "img" when running the query ?- target(img, X). against RULE-SET 5 (left).

Exp 2 (result): The labelled rule-sets make intuitive sense to humans. This representation of knowledge in default theory in our opinion makes the rule-set easy to understand. The rule-set captures the knowledge of the trained CNN.

For example in RULE-SET 2, the first rule states that "an image `X` is a 'street' if there is evidence of the concept 'building' in the image". Similarly the second rule states that "an image `X` is a 'forest road' if there is evidence of the concept 'tree' in the image and there is no evidence of some abnormal conditions 'ab1' and 'ab2'.

Notice how in rules 2,3 of RULE-SET 1 in Fig. 3 the group of kernels, now labelled as 'wall1' and 'wall3' are (most probably) detecting a certain type of patterns on the walls that are indicative of bathrooms, possibly tiles. The kernels are labelled as wall only because the semantic segmentation masks available to us have the label 'wall' for the pixels that denote wall in the image. Hence we are restricted to the expressiveness of the annotations available to us. This can be alleviated by labelling the predicates via manual observation.

Note that in the first rule of RULE-SET 5 (Fig. 4) there is a predicate `cabinet4_wall5/1`. This predicate corresponds to the kernel group in the CNN that is detecting either both cabinets and walls separately or a specific region in the images that contains a portion of cabinets and wall. It is hard to distinguish between the two cases.

Fig 4 shows a sample justification obtained from s(CASP) for some image "img". The binarized vector associated with "img" is used to write the facts and the query `target(img, X)` is executed against RULE-SET 5. The first rule (shown in red) was satisfied. The first model found by s(CASP) that satisfies the rule-set binds the value of `X` to 'kitchen'. Hence, the predicted class of the image "img" is kitchen.

7   Related Work

A similar approach of generating rules from the CNN was adopted by Townsend et al. [26], [27] where they used a decision tree algorithm to generate the rule-set. However, Padalkar et al. [17] have shown that using FOLD-SE-M generates a much smaller rule-set and higher accuracy and fidelity.

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. [32] and Zhou et al. [35] use the output activation while others [20],[5],[23] use gradients to find the mapping. Unlike NeSyFOLD-G, these visualization methods do not generate any rule-set. Zeiler et al. [32] 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. [7] 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. [18] 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. [3] 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. [34],[33] learn disentangled concepts from the CNN and represent them in a hierarchical graph so that there is no assumption of a one to one kernel-concept mapping. However, no logical explanation is generated. Bologna et al. [2] extract propositional rules from CNNs. Their system operates at the neuron level, while NeSyFold-G works with groups of neurons.

Our NeSyFOLD-G framework uses FOLD-SE-M to extract a logic program from the binarization table. There are other works that focus on extracting logic programs such as the ILASP system [13] by Law et al. and the XHAIL [19] system by Ray et al. which induce an answer set program from the data however these systems do not learn rules from images. Some other works [21], [6], [22] use a neurosymbolic system to induce logic rules from data. These systems belongs to the Neuro:Symbolic → Neuro category whereas ours belongs to the Neuro;Symbolic category.

8 Conclusion and Future Work

In this paper we have shown how the NeSyFOLD-G framework can be used to make a CNN more interpretable. We used the framework with a trained CNN to derive a NeSy-G model that constitutes the CNN with all layers after the last convolutional layer replaced by the rule-set generated by FOLD-SE-M algorithm. We compared the performance of the NeSyFOLD-G framework with that of the NeSyFOLD framework on various datasets. The major difference between the NeSyFOLD-G and the NeSyFOLD framework is that in the former, groups of similar kernels are found and the output of these groups kernels is then binarized to produce the binarization table, that is used as input to the FOLD-SE-M algorithm which generates a rule-set. The kernel grouping algorithm is a novel contribution of this work. In the NeSyFOLD framework each individual kernel's output is binarized and the rules are generated based on the binarization table thus constructed.

We show in the experiments that grouping similar kernels leads to the creation of better features in the binarization table which consequently leads to a more succinct rule-set. The NeSyFOLD-G framework always generates a smaller rule-set than that generated by the NeSyFOLD framework while either outperforming or showing comparable accuracy and fidelity.

We also introduced a novel semantic labelling algorithm that can be used for labelling each predicate that appears in the rule-set with the concepts(s) that its corresponding kernel group represents. We showed two labelled rule-sets and an example justification of a prediction that can be obtained using the s(CASP) ASP system.

Note that both NeSyFOLD-G and NeSyFOLD are aimed at representing the connectionist knowledge of the CNN in terms of a symbolic rule-set. The symbolic rule-set can then be scrutinized by experts to figure out the biases that the CNN might have learnt from the data and these help in avoiding spurious predictions in sensitive domains such as medical imaging. The advantage that NeSyFOLD-G provides is that the interpretability of the rule-set increases as the size of the generated rule-set is significantly smaller.

We acknowledge that the semantic segmentation masks of images may not be readily available depending on the domain, in which case the semantic labelling of the predicates has to be done manually. Our NeSyFOLD-G framework helps in this regard as well, as it decreases the number of predicates that need to be labelled.

As the number of classes increases, the loss in accuracy also increases due to the binarization of more kernels. 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.

In future, we plan to use NeSyFOLD-G for real-world tasks such as interpretable breast cancer prediction. We also intend to explore combining the knowledge of two or more CNNs by producing a single rule-set that contains the kernels of the corresponding CNNs as predicates. We also plan to investigate how the knowledge from the generated rules can be backpropagated to improve the performance of a CNN [31].

Acknowledgments

References

1. Arias, J., Carro, M., Salazar, E., Marple, K., Gupta, G.: Constraint answer set programming without grounding. Theory and Practice of Logic Programming **18**(3-4), 337–354 (2018)
2. Bologna, G., Fossati, S.: A two-step rule-extraction technique for a cnn. Electronics **9**(6), 990 (2020)
3. Chen, C., Li, O., Tao, D., Barnett, A., Rudin, C., Su, J.K.: This looks like that: deep learning for interpretable image recognition. Advances in neural information processing systems **32** (2019)
4. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
5. Denil, M., Demiraj, A., de Freitas, N.: Extraction of salient sentences from labelled documents (2014). `https://doi.org/10.48550/ARXIV.1412.6815`, `https://arxiv.org/abs/1412.6815`

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

7. Ferreira, J., de Sousa Ribeiro, M., Gonçalves, R., Leite, J.: 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. pp. 432–442 (8 2022). `https://doi.org/10.24963/kr.2022/45`, `https://doi.org/10.24963/kr.2022/45`

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

9. Kanagaraj, N., Hicks, D., Goyal, A., Mishra Tiwari, S., Singh, G.: Deep learning using computer vision in self driving cars for lane and traffic sign detection. International Journal of System Assurance Engineering and Management **12** (05 2021). `https://doi.org/10.1007/s13198-021-01127-6`

10. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014). `https://doi.org/10.48550/ARXIV.1412.6980`, `https://arxiv.org/abs/1412.6980`

11. Ko, B., Kwak, S.: Survey of computer vision-based natural disaster warning systems. Optical Engineering **51**, 0901– (07 2012). `https://doi.org/10.1117/1.OE.51.7.070901`

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

13. Law, M., Russo, A., Broda, K.: The ilasp system for inductive learning of answer set programs. ArXiv **abs/2005.00904** (2020)

14. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural Computation **1**(4), 541–551 (1989). `https://doi.org/10.1162/neco.1989.1.4.541`

15. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)

16. Lloyd, J.: Foundations of Logic Programming. Springer, second, extended edition (1987)

17. Padalkar, P., Wang, H., Gupta, G.: Nesyfold: Neurosymbolic framework for interpretable image classification (2023), `https://arxiv.org/abs/2301.12667`

18. Qi, Z., Khorram, S., Fuxin, L.: Embedding deep networks into visual explanations. Artificial Intelligence **292**, 103435 (2021). `https://doi.org/10.1016/j.artint.2020.103435`

19. Ray, O.: Nonmonotonic abductive inductive learning. Journal of Applied Logic **7**(3), 329–340 (2009). `https://doi.org/https://doi.org/10.1016/j.jal.2008.10.007`, `https://www.sciencedirect.com/science/article/pii/S1570868308000682`, special Issue: Abduction and Induction in Artificial Intelligence

20. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Gradcam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE international conference on computer vision. pp. 618–626 (2017)

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

22. Shindo, H., Nishino, M., Yamamoto, A.: Differentiable inductive logic programming for structured examples. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 5034–5041 (2021)
23. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps (2013). `https://doi.org/10.48550/ARXIV.1312.6034`, `https://arxiv.org/abs/1312.6034`
24. Stallkamp, J., Schlipsing, M., Salmen, J., Igel, C.: Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. Neural Networks **32**, 323–332 (2012). `https://doi.org/https://doi.org/10.1016/j.neunet.2012.02.016`, `https://www.sciencedirect.com/science/article/pii/S0893608012000457`, selected Papers from IJCNN 2011
25. Sun, W., Zheng, B., Qian, W.: Computer aided lung cancer diagnosis with deep learning algorithms. In: SPIE Medical Imaging (2016)
26. Townsend, J., Kasioumis, T., Inakoshi, H.: Eric: Extracting relations inferred from convolutions. In: Ishikawa, H., Liu, C.L., Pajdla, T., Shi, J. (eds.) Computer Vision – ACCV 2020. pp. 206–222. Springer International Publishing, Cham (2021)
27. Townsend, J., Kudla, M., Raszkowska, A., Kasiousmis, T.: On the explainability of convolutional layers for multi-class problems. In: Combining Learning and Reasoning: Programming Languages, Formalisms, and Representations (2022), `https://openreview.net/forum?id=jgVpiERy8Q8`
28. Wang, H., Gupta, G.: FOLD-SE: An efficient rule-based machine learning algorithm with scalable explainability (2022). `https://doi.org/10.48550/ARXIV.2208.07912`
29. Xie, N., Sarker, M.K., Doran, D., Hitzler, P., Raymer, M.: Relating input concepts to convolutional neural network decisions (2017). `https://doi.org/10.48550/ARXIV.1711.08006`
30. Yang, Y., Kim, S., Joo, J.: Explaining deep convolutional neural networks via latent visual-semantic filter attention. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8333–8343 (2022)
31. Yang, Z., Ishay, A., Lee, J.: Neurasp: Embracing neural networks into answer set programming. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence. IJCAI'20 (2021)
32. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European conference on computer vision. pp. 818–833. Springer (2014)
33. Zhang, Q., Cao, R., Shi, F., Wu, Y.N., Zhu, S.C.: Interpreting CNN knowledge via an explanatory graph. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
34. Zhang, Q., Cao, R., Wu, Y.N., Zhu, S.C.: Growing interpretable part graphs on convnets via multi-shot learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 31 (2017)
35. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2921–2929 (2016)
36. Zhou, B., Lapedriza, A., Khosla, A., Oliva, A., Torralba, A.: Places: A 10 million image database for scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (2017)
37. Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ade20k dataset. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5122–5130 (2017). `https://doi.org/10.1109/CVPR.2017.544`