Elite BackProp: Training Sparse Interpretable Neurons

Theodoros Kasioumis, Joe Townsend and Hiroya Inakoshi

Abstract

In this paper we present a method called *Elite BackProp* (EBP) to train more interpretable convolutional neural networks (CNNs) by introducing *class-wise* activation sparsity; after training, each class will be associated with a small set of *elite* filters that fire rarely but highly activate on visual primitives from images of that class. Our method is broadly applicable as it does not require additional object part annotations during training. We demonstrate experimentally that EBP realizes high degrees of activation sparsity with no accuracy loss and enhances the performance of a rule extraction algorithm that distils the knowledge from a CNN, by inducing more compact rules that use fewer atoms to describe the decisions of a CNN while maintaining high fidelity compared with other solutions. This happens because EBP induces sparse compositional representations that reuse and combine primitive filters. Such representations can assist in understanding the reasoning behind a CNN and build trust into their decisions.

Keywords

neural-symbolic integration, training interpretable CNNs, activation sparsity, rule extraction

1. Introduction

Training neural networks to be interpretable [1, 2] or interpreting their decisions [3, 4, 5, 6], has received a great deal of attention in recent years. Multiple rule extraction algorithms have been proposed that aim to distil the knowledge from a CNN and intepret its decisions [7, 8, 9, 10, 11], many of which rely on thresholding filter activations to determine whether a filter can be considered active (a process known as quantisation [9, 12]), detecting a specific pattern across images. After combining active filters, rules are formed to explain the classification decision of the CNN, where each atom used in explanations corresponds to an active filter.

One can regard compact explanations as more interpretable as there is less information for the reader to assimilate. Thus, ideal explanations are composed of smaller sets of rules which are in turn composed of smaller sets of atoms that can be reused across the rule set and across different classes in different combinations. Such compactness is more difficult to achieve if the original CNN itself encodes a large number of redundant representations, i.e. in which different convolutional filters co-activate on the same concepts.

We aim to tackle the aforementioned inefficiencies by introducing an algorithm called *Elite BackProp (EBP)* that enforces *class-wise activation sparsity*. That is, EBP trains CNNs to associate each class with a handful of *elite* filters that fire rarely but highly activate on images from that

NeSy'20/21: 15th International Workshop on Neural-Symbolic Learning and Reasoning, October 25–27, 2021, Virtual

theodoros.kasioumis@fujitsu.com (T. Kasioumis); joseph.townsend@fujitsu.com (J. Townsend); hiroya.inakoshi@fujitsu.com (H. Inakoshi)
0000-0003-2008-5817 (T. Kasioumis); 0000-0002-5478-0028 (J. Townsend); 0000-0003-4405-8952 (H. Inakoshi)

CEUR Workshop Proceedings (CEUR-WS.org)

class. By filter we mean the set of weights that make a single channel in the convolutional layer. Every filter is assigned a probability of being active on each class based on its frequency and magnitude of activation on images from that class. Filters are ranked for each class according to their activation probability and the top-K filters for a class form its *elite*. If a filter activates on an image from a class that has low probability of activating that filter, then a penalty inversely proportional to filter's ranking is applied. Since only the *elite* filters will have strong activations for each class, the model is incentivised to learn more compositional representations by re-using and combining primitive filters to describe more complex concepts rather than learning redundant multiple filters for each class separately.

Activation sparsity has been used extensively in the literature in many different forms [13, 14, 15, 16, 17, 18, 19, 20, 21, 22] and it has also been hypothesized that sparsely activated neurons are more interpretable than neurons that activate frequently [23]. To the best of our knowledge, EBP is the first to induce class-wise activation sparsity as we define it above.

Our experiments show that models trained with EBP maintain their original accuracy while realizing high degrees of sparsity. Furthermore, we demonstrate that the group activation sparsity induced by EBP benefits a rule extraction algorithm that distils the knowledge from a trained CNN and explains the output classification in an interpretable logical language over quantized filter activations represented as logical atoms. We show that EBP yields extracted programs with high fidelity (accuracy in approximating the original model) that use much fewer atoms compared with other activation sparsity methods. Moreover, we present qualitative examples of how the receptive field of filters trained with EBP is much more interpretable compared to training without as it consistently detects a specific pattern across different images.

This paper is organized as follows: in Section 2 we review prior work in activation sparsity and interpretability. In Section 3, we present EBP and Section in 4 we conduct experimental evaluation of EBP. Section 5 concludes by summarizing our results and discussing future work.

2. Related Work

Work in the field of neural-symbolic integration [3, 8, 12, 24, 25] concerns both post-hoc and self-interpretable models of explainability for neural networks, and employ symbolic representations such as decision trees or logic programs to explain a network's behavior. ProtoPNet [1] trains interpretable CNNs by partitioning the input image into sub-regions and associating each region with a prototypical part of some class that "looks like" the extracted region. Another idea introduced in [2] uses a "filter loss"that pushes a filter to represent an object part of a particular class, but not others. Both lines of work differ from EBP in that they both add a new structural layer whereas EBP only adds a penalty to the loss that enforces class-wise activation sparsity.

Another line of research focuses on sparsifying connections or activations in neural networks in order to reduce overfitting and redundancy in representations while optimizing the model in terms of accuracy and speed. The literature in weight sparsification and compression is vast, see for example [26] and references therein. However, our work is only concerned with *activity* sparsity, and not weight sparsity.

Sparse activity was initially inspired by sparse coding [23] and has been used in the literature in many different forms. The commonly used ReLU activation function [13] induces activation sparsity in a fraction of neurons and recently authors in [18] proposed three activation functions to induce spike-like sparsity. A k-Winners-Take-All activation (k-WTA) function which retains only the k highest activations from a layer and sets all the rest to zero was used in [17] to improve adversarial robustness. This is a natural generalization of the boolean K-Winners-Take-All network [27] which was motivated by biological neural circuits and had boolean outputs. Authors in [28] also utilized k-WTA in k-Sparse Autoencoders. Another winner-take-all method is Duty Cycle [19] which sparsifies activations in a layer like k-WTA but in addition it sparsifies connections preceding that layer by initializing it from sparse random distribution and also it introduces a boosting term to favor units that have not been recently active in order to encourage every unit to be equally active and hence maintain the representational power of the model. Our method differs from these works in that it associates images of each class to a common group of winners/elites without modifying the forward pass or pruning connections. Instead it penalizes filters that activate on images from a class that has low probability of activating those filters.

Dropout [15] has been widely used to prevent co-adaptation of neurons by randomly dropping them during training with uniform probability. Sparseout [16] extended Dropout by imposing an L_q penalty on the activations, allowing one to choose the level of activation sparsity. Recently *DASNet* [20] introduced a dynamic activation sparsity method, utilizing a winners-take-all dropout technique. DASNet behaves as a mask between layers and prunes low-ranking neurons in terms of their activation magnitude at run-time for computational speedups. In contrast EBP is not concerned with pruning and the activations are not masked at runtime.

Other solutions induce activation sparsity by applying regularization penalties; [29] penalizes the deviation of the expected activation of the hidden units from a low fixed level p to achieve a sparsity level p layer-wise and [14] added a cross entropy term between the average probability of unit activation and the desired sparsity level p that encourages the activation probability of a neuron to be close to p. [30] used a clustering based regularization approach to obtain sparse representations. Recently [21] exploited an L_1 activation regularizer for computational benefits and [22] improved the results using a variant of ReLu in conjuction with a regularizer based on Hoyer sparsity metric [31, 32]. Both [21, 22] induce activation sparsity layer-wise, i.e., a certain percent of neuron activations in each layer will be retained and [22] introduces a novel approach for activation pruning for computational speedups. In contrast, EBP induces class-wise sparsity, meaning that images from a specific class will be associated with the same group of *elite* filters and no pruning is performed.

3. Method

We denote by C the number of classes in a dataset $\mathcal{X}$ and by c_i the ground truth class of an image $X_i \in \mathcal{X}$. Given a conv-layer l of a CNN with filters $\{f_1^{(l)}, \ldots, f_{M_l}^{(l)}\}$ and a batch of images $\{X_i, \ldots, X_N\}$, let $F_i^{(l)} = (F_{i1}^{(l)}, F_{i2}^{(l)}, \ldots, F_{iM_l}^{(l)})$ be the feature map output of X_i at the l-th layer, where each $F_{ij}^{(l)}, j = 1, \ldots, M_l$, is a 2D activation matrix that is the output of convolving the feature map of layer $l - 1$ with the j-th filter for i-th image, followed by ReLU and maxpooling (if present). $F_i^{(0)} = X_i$ denotes the input image. The activation $A_{ij}^{(l)}$ of filter $f_j^{(l)}$ for the i-th image in the batch at layer l is defined as the spatial average of activations:

$$A_{ij}^{(l)} = \frac{1}{(H_{il}W_{il})} \sum_{r}^{H_{il}} \sum_{s}^{W_{il}} |(F_{ij}^{(l)})_{rs}|, \tag{1}$$

where H_{il}, W_{il} denotes the height and width of the feature map at layer l respectively for the image X_i and $(\cdot)_{rs}$ stands for the (r, s) spatial coordinates. A filter $f_j^{(l)}$ is said to be *active* for an image X_i if its activation $A_{ij}^{(l)} > \theta$, where θ is a specified threshold.

EBP associates each class c with a handful of K_c *elite* filters that activate rarely and have strong activation magnitude on images from that class. This is accomplished by assigning each filter a probability of being active for each class based on the frequency and magnitude of activations on images from that class during training. Then, for each class c filters are ranked and the top-K_c form its *elite*. K_c controls the degree of sparsity and representation power of the model (more sparse for lower K_c). Filters that activate on images from a class that do not belongs to its elite are penalized, with penalties inversely proportional to their ranking.

On each iteration and for each image X_i of class c_i in the batch, EBP stores the M_l-dimensional vector $(A_{i1}^{(l)}, \ldots, A_{iM_l}^{(l)})$ of average filter activations (Eq. 1) from a layer l accumulatively in a vector D^{c_i} as follows:

$$D^{c_i} \leftarrow D^{c_i} + (A_{i1}^{(l)}, \ldots, A_{iM_l}^{(l)}). \tag{2}$$

Afterwards, for each class c, filters are ranked based on their history of accumulated activations:

$$f_{r_i}^c \succ f_{r_{i+1}}^c \quad \text{iff} \quad A_{r_i}^c > A_{r_{i+1}}^c, \quad \text{for} \quad i \in \{1, \ldots, M_{l-1}\}, \tag{3}$$

where $D^c = (A_1^c, \ldots, A_{M_l}^c)$ and $A_r^c = \sum_i A_{ir}^c$, $r = 1, \ldots, M_l$ denotes the accumulated activations of the r-th filter f_r^c for class c (layer (l) is removed from the superscript for notational convenience). The K_c highest activated filters for each class c form its *elite* and are stored in a set $E^c = \{f_{r_1}^c, \ldots, f_{r_{K_c}}^c\}$. For each filter f_j^c we define a probability p_{jc} of being active for class c as in Equation 4 and we penalize filters which activate on an image X_i of class c_i, with penalties $R(W_{1:l})$ proportional to $(1 - p_{jc_i})$ (i.e., the probability of filter being inactive):

$$R(W_{1:l}) = \sum_{i=1}^{N} \sum_{j=1}^{M_l} (1 - p_{jc_i}) A_{ij}^{(l)}, \quad \text{where} \quad p_{jc} = 1 - \delta_{jc} \frac{D_j^c}{A_{r_{K_c}}^c}, \tag{4}$$

where $W_{1:l}$, denotes the set of weights from layer 1 up to l, $\delta_{jc} = 1$ if $f_j \notin E_c$ and 0 otherwise, D_j^c is the j-th index of D^c and $A_{r_{K_c}}^c$ the K_c-th sorted accumulated activation (Eq. 3) for class c. The total loss we optimize in each batch is a combination of cross entropy loss and R:

$$L_W(y, \hat{y}) = -\sum_{i=1}^{N} \sum_{c=1}^{C} y_{ic} \log \hat{y_{ic}} + \lambda R(W_{1:l}), \tag{5}$$

where $y_{ic} = 1$ if the i-th observation X_i is of class c and 0 otherwise, $\hat{y_{ic}}$ is the predicted probability that X_i is of class c and $\lambda \in \mathbb{R}$ controls the regularization strength. In our experiments we have chosen for simplicity the number of elites K_c per class c to be equal to K for all classes, and we leave experimenting with different K_c for each class for future work. We refer to the Appendix 6 for further discussion and visualization of the class probabilities of images before and after applying EBP.

4. Experiments

We conduct experiments on 2 image classification datasets; a 6-class animal subset [2] of Pascal VOC [33] and a toy 3-class subset of places365 dataset [34] consisting of the classes *forest road*, *highway* and *street* which we will refer to as the *road* dataset. Regarding (train, valid, test) splits, we further split the original train-val to get a test set because annotations were not available for the original test set. We use (3341, 594, 1399) for the 6-class Pascal subset and (10444, 1500, 3054) for the 3-class *road* dataset. We quantitatively compare EBP in terms of accuracy, activation sparsity and rule extraction (Section 4.1) with several activation sparsity methods from the literature. We show that EBP results in higher sparsity without sacrificing accuracy and also that rules extracted from a CNN trained with EBP use fewer atoms and have higher fidelity. Qualitative examples of extracted rules in the *road* dataset using EBP are shown in Fig. 3. From k-winner-take-all methods we compare with k-WTA and Duty Cycle and from regularization methods we compare with [29] which we refer to as *EASR* (Expected Activation Sparsity Regularization). We use Sparseout to investigate the effect of dropout-like methods. Since our method is concerned with sparsifying existing layer activations, we do not compare with [1, 2] because they require additional, specialised neural layers to enforce interpretability.

For each method, we initialized VGG-16 with weights pretrained on ImageNet and followed the preprocessing and augmentations in [35]. We finetuned the last dense layer for 50 epochs with Adam optimizer [36], ridge regularization parameter 0.005 and learning rate 5×10^{-5} using an NVIDIA GPU 1080 Ti 12GB and Tensorflow[1]. Afterwards, we applied each method after the conv13 layer and resumed training all layers for 100 epochs with learning rate 10^{-6}. We select the conv13 layer because filters in deeper layers in CNNs tend to represent object parts and more semantic concepts than earlier layers [37, 38]. Regarding the hyperparameters, for EBP we trained with different $K \in \{10, 25, 50, 100, 200, 300\}$ (same K for all classes) and regularization values[2] $\lambda \in \Lambda = \{0.1, 0.01, 0.001, 0.0001\}$. For *EASR* we used $\lambda \in \{0.0005, 0.0001, 0.00005, 0.00002\}$ and for k-WTA we trained with density ratio $k \in \{0.05, 0.1, 0.2, 0.5, 0.8, 0.9, 0.95\}$. For Duty Cycle we tune the density coefficient $\hat{a}^l$ that controls the percentage of neurons that are expected to be active in $\{0.1, 0.2, 0.5, 0.7, 0.9\}$ and the boosting coefficient β from 1 up to 512, incrementing it in powers of 2. For Sparseout we experimented with values for $p \in \{0.3, 0.5\}$ and $q \in \{0.5, 1, 1.5, 2, 2.5, 3\}$.

4.1. Sparsity and Rule Extraction

We measure the activity sparsity after training with each method with various metrics such as Hoyer measure [32], Lifetime and Population kurtosis, Treves-Rolls lifetime (T-R Life.), Treves-Rolls population (T-R Popul.) and activity sparseness [39] on images from *road* and 6-class subset of Pascal dataset. The rationale behind choosing this toy *road* dataset to assess the sparsity and benefits in a simple rule extraction algorithm (Section 4.1) is that scenes contain topics that are shared between different classes and we don't want to learn separate filters for each of them. For example, trees appear in all classes, hence it is preferable to learn one filter

[1] https://www.tensorflow.org/

[2] for smaller K values and $\lambda = 0.01$ the convergence was slower. Training for 100 epochs was necessary to determine the best model but with higher K values and lower λ the models converged in $30 - 50$ epochs.

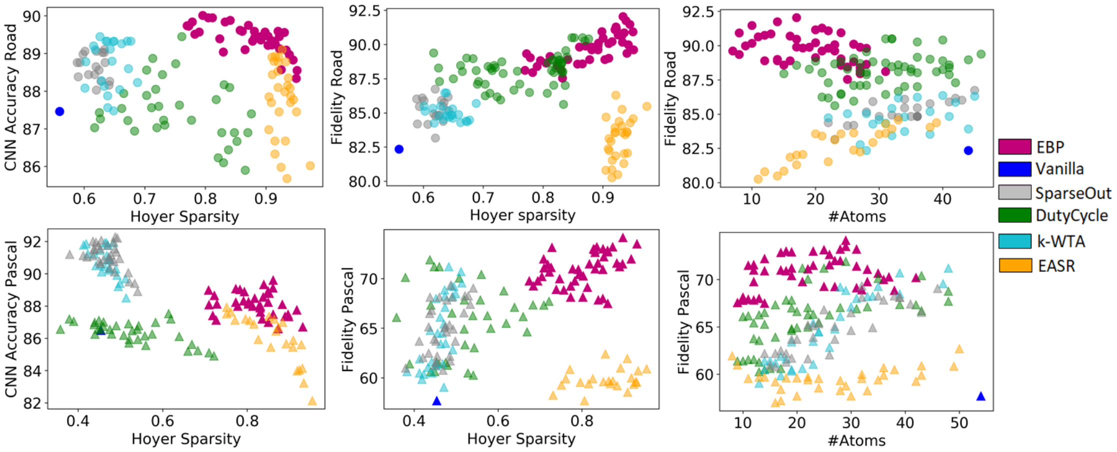

Figure 1: Comparing accuracy, fidelity and size of extracted program for each method using different hyperparameters and avg tree depth 6 for *road* (1a-c) and 7 for 6-class subset of Pascal (2a-c) (each point represents the average over three trials). EBP is associated with high activation sparsity without sacrifycing accuracy. Rules extracted from CNN trained with EBP use less atoms in explanations.

	Road Rule Extraction					**Road** Activity Sparsity						
Sparsity Method	Extracted Acc.	Fidelity	Atoms	Rules	Avg. Depth	Hoyer Spars.	Life. Kurt.	Popul. Kurt.	T-R Popul.	T-R Life.	Activ. Spars.	CNN Acc.
EBP	**84.48**	**89.87**	**12**	**36**	**6**	**0.9287**	**1175.3**	**212.34**	**0.9868**	**0.9807**	**0.9791**	**89.47**
Vanilla	78.95	82.35	44	56	6	0.5593	32.02	28.26	0.7789	0.7366	0.8367	87.46
Sparseout	80.64	84.84	36	46	6	0.6331	57.02	27.24	0.84	0.769	0.8746	**89.03**
DutyCycle	82.16	87.11	28	39	6	0.7413	260.36	46.45	0.9141	0.8508	0.9126	**89.16**
k-WTA	80.85	86.46	31	45	6	0.6436	61.83	28.54	0.8483	0.7671	0.8803	**89.32**
EASR	79.04	84.38	19	**36**	6	**0.9269**	967.08	**223.91**	**0.9862**	**0.9792**	**0.9789**	88.93

Table 1
For each method we have chosen hyperparameters that yielded the best CNN accuracy and sparsity on *road* dataset (see 6 Appendix). For an ablation study of different hyperparameters see Fig. 1. CNNs trained with EBP achieve high activity sparsity and the program extracted from the CNN using EBP has higher fidelity and uses fewer rules and atoms in explanations. The best depth for each method was chosen based on sklearn's *cost-complexity pruning* and results are averaged across 3 different runs. Notice also that all sparse activity methods achieve better accuracy than the baseline. Best results shown in bold (if deviation is less than 0.5% from the best multiple are marked).

	Pascal Rule Extraction					**Pascal** Activity Sparsity						
Sparsity Method	Extracted Acc.	Fidelity	Atoms	Rules	Avg. Depth	Hoyer Spars.	Life. Kurt.	Popul. Kurt.	T-R Popul.	T-R Life.	Activ. Spars.	CNN Acc.
EBP	**71.48**	**72.69**	**26**	**32**	**7**	**0.861**	**286.64**	**140.93**	**0.9508**	**0.9674**	**0.9584**	89.2
Vanilla	57.18	57.68	54	63	7	0.454	17.32	18.88	0.6983	0.6659	0.7856	86.49
Sparseout	66.98	68.26	39	51	7	0.488	16.84	19.55	0.7189	0.6598	0.7994	**92.2**
DutyCycle	53.45	67.33	44	58	7	0.619	71.55	46.4	0.8104	0.8032	0.8527	87.53
k-WTA	70.41	71.19	48	53	7	0.483	15.73	18.45	0.7069	0.6459	0.7909	**91.92**
EASR	58.12	59.25	42	41	7	0.731	71.87	78.69	0.9061	0.882	0.9125	87.92

Table 2
Similar analysis as in Table 1 for rule extraction and sparsity of each method on 6-class subset of *Pascal*. Ablation study for different hyperparameters in depicted in Fig. 1 (bottom). Results consistently indicate that EBP realizes high activity sparsity and the program extracted from CNN using EBP has higher fidelity and uses fewer rules and atoms in explanations. Best results shown in bold.

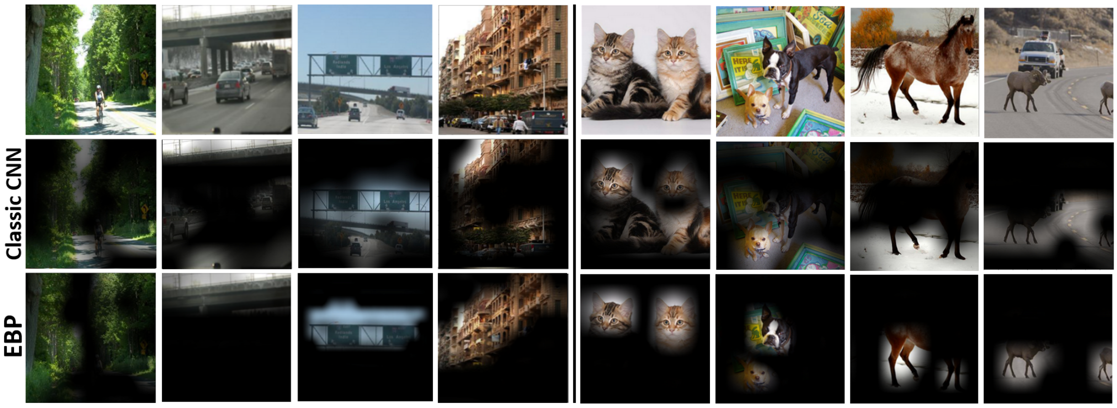

Figure 2: Comparison of a filter's highly activated feature maps (as defined in [2]) in ordinary CNN (top) and those after *EBP* (bottom) in *road* dataset (col. 1-4) and pascal (col. 5-8) . Filters trained with EBP capture more compact semantic regions and it is easier to access the activation pattern.

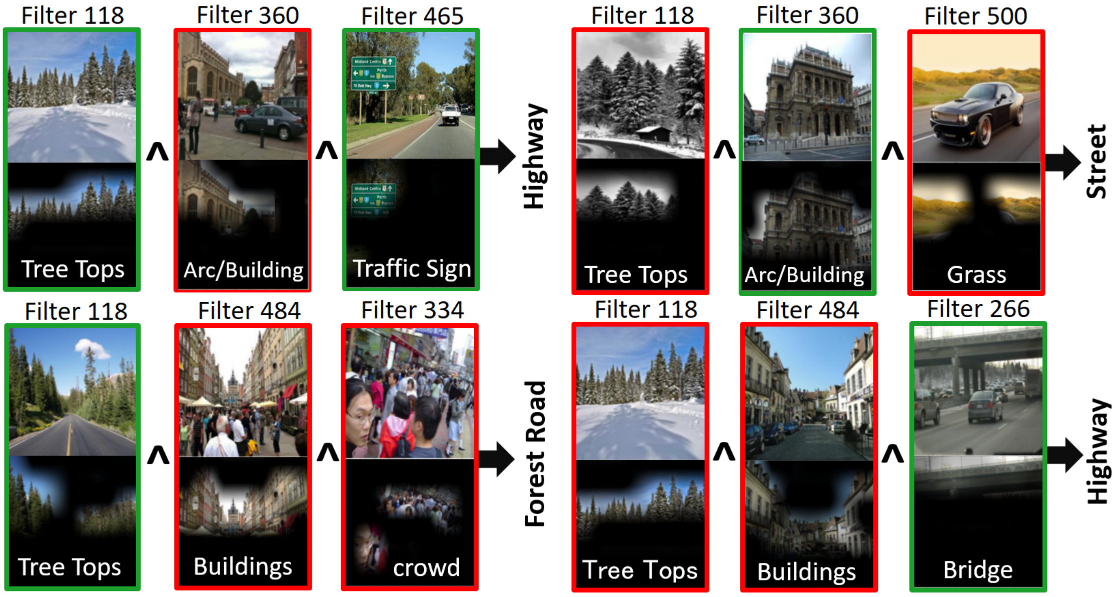

Figure 3: Visualization of rules extracted after training with EBP for depth 3 trees and the receptive field of filters. Green frame denotes presence and red absence of a pattern. Notice how concepts like trees and buildings are shared between classes promoting more compositional representations that use fewer atoms.

that fires in response to trees (in all classes) rather than having a filter per class. Results in Tables 1 and 2 indicate that EBP achieves high activity sparsity without sacrificing accuracy. Fig. 2 depicts that filters trained with EBP consistently detect a specific pattern across images and that they fire in response to smaller and more compact semantic regions which are more interpretable and have activation patterns that are more clear.

Though there is no consensus in the literature regarding a metric for interpretability, the size of an explanation has been proposed as an option [8, 12]. To assess the extent to which EBP benefits rule extraction by inducing parsimonious compositional representations that re-use and combine primitive filters to form new concepts, we show how a binary decision tree trained as a symbolic approximation of a CNN's behavior is more compact when that CNN is trained with EBP than without. To perform rule extraction we first quantize each filter activation. A filter is considered active if its activation (Eq. 1) is above $(\mu + \sigma)$ where μ and σ denote the mean and standard deviation of filter activations after conv13 layer across all images. The tree takes the thresholded filter activations of the *conv13* layer as the input and the CNN's output classification as the target so that for a given input instance the induced decision path from the root to the leaf node provides a symbolic explanation of the original CNN's classification of that instance. Each such path can be regarded as a separate rule over a set of atoms, e.g. $A \wedge \neg B \wedge C \rightarrow$ *highway* states (Fig. 3 top-left) that if filters A and C are active and filter B is inactive, then the input is a highway.

We approximated VGG-16 on the *road* and 6-class subset of Pascal dataset by generating decision trees using the *sklearn*[3] library's DecisionTreeClassifier class using entropy criterion and cost complexity pruning to choose the optimal depth. Fig. 1 (1a-c, 2a-c) shows the tradeoff between sparsity, fidelity and accuracy after training using different hyperparameter values for each method. Results in Table 1 and Fig. 1 demonstate that programs extracted from a CNN trained with EBP have higher fidelity and use consistently fewer unique atoms and rules in explanations compared to other approaches. This means that EBP induces more compositional representations that re-use and combine filters instead of using redudant multiple filters for each class separately. Fig. 3 shows some rules extracted from a CNN trained with EBP and the receptive field of filters activation. For visualization purposes we used a tree of depth 3.

5. Conclusion and future work

In this paper we proposed a method that promotes group activation sparsity to encourage more parsimonious and interpretable representations in a CNN. We demonstrated that models trained with EBP realize high degrees of activation sparsity without sacrificing accuracy and benefit a rule extraction algorithm that distils the knowledge from a CNN; fewer atoms are used in explanations while maintaining high fidelity. In future work we aim to conduct experiments with different K_c values for each class c, further analysis of different layers and rule extraction and possibly an application with emphasis on studying the benefits of compositionality. Another future direction is to embed rules into neural networks, in order to enhance their interpretability and generalization capabilities. Rule embedding can assist in neural-symbolic integration, which is an important step towards bridging the gap between neural networks and symbolic representations.

[3] https://scikit-learn.org/stable/

References

[1] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, J. K. Su, This looks like that: Deep learning for interpretable image recognition, in: Advances in Neural Information Processing Systems, volume 32, Curran Associates, Inc., 2019.

[2] Q. Zhang, Y. N. Wu, S.-C. Zhu, Interpretable convolutional neural networks, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 8827–8836.

[3] J. Townsend, T. Kasioumis, H. Inakoshi, ERIC: extracting relations inferred from convolutions, in: 15th Asian Conference on Computer Vision, Kyoto, Japan, Revised Selected Papers, Part III, volume 12624 of *Lecture Notes in Computer Science*, Springer, Nov. 30 - Dec. 4, 2020, pp. 206–222.

[4] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, D. Pedreschi, A survey of methods for explaining black box models, ACM Comput. Surv. 51 (2018).

[5] S. M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, vol. 30, in: Advances in Neural Information Processing Systems, Curran Associates, Inc., 2017.

[6] M. T. Ribeiro, S. Singh, C. Guestrin, "why should i trust you?": Explaining the predictions of any classifier, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016, pp. 97–101.

[7] N. Frosst, G. E. Hinton, Distilling a neural network into a soft decision tree, ArXiv abs/1711.09784 (2017).

[8] R. Andrews, J. Diederich, A. B. Tickle, Survey and critique of techniques for extracting rules from trained artificial neural networks, Knowledge-Based Systems 8 (1995) 373–389.

[9] H. Jacobsson, Rule extraction from recurrent neural networks: Ataxonomy and review, Neural Computation 17 (2005) 1223–1263.

[10] Q. Zhang, R. Cao, F. Shi, Y. N. Wu, S.-C. Zhu, Interpreting cnn knowledge via an explanatory graph, Proceedings of the AAAI Conference on Artificial Intelligence 32 (2018).

[11] Q. Zhang, Y. Yang, H. Ma, Y. N. Wu, Interpreting cnns via decision trees, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[12] J. Townsend, T. Chaton, J. M. Monteiro, Extracting relational explanations from deep neural networks: A survey from a neural-symbolic perspective, IEEE Transactions on Neural Networks and Learning Systems 31 (2020) 3456–3470.

[13] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, volume 15 of *Proceedings of Machine Learning Research*, Fort Lauderdale, USA, 2011, pp. 315–323.

[14] V. Nair, G. E. Hinton, 3d object recognition with deep belief nets, in: Advances in Neural Information Processing Systems, volume 22, Curran Associates, Inc., 2009.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (2014) 1929–1958.

[16] N. Khan, I. Stavness, Sparseout: Controlling sparsity in deep networks, in: Advances in Artificial Intelligence - 32nd Canadian Conference on Artificial Intelligence, Canadian AI 2019, Kingston, ON, Canada, May 28-31, 2019, Proceedings, volume 11489 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 296–307.

[17] C. Xiao, P. Zhong, C. Zheng, Enhancing adversarial defense by k-winners-take-all, Inter-

national Conference on Learning Representations (ICLR) (2020).

[18] P. Bizopoulos, D. Koutsouris, Sparsely activated networks, IEEE Transactions on Neural Networks and Learning Systems 32 (2021) 1304–1313.

[19] S. Ahmad, L. Scheinkman, How can we be so dense? the benefits of using highly sparse representations, ICMLWorkshop on Uncertainty and Robustness in Deep Learning (2019).

[20] Q. Yang, J. Mao, Z. Wang, H. Li, Dasnet: Dynamic activation sparsity for neural network efficiency improvement, in: 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), 2019, pp. 1401–1405.

[21] G. Georgiadis, Accelerating convolutional neural networks via activation map compression, in: Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 7078–7088.

[22] M. Kurtz, J. Kopinsky, R. Gelashvili, A. Matveev, J. Carr, M. Goin, W. Leiserson, S. Moore, N. Shavit, D. Alistarh, Inducing and exploiting activation sparsity for fast inference on deep neural networks, in: Proceedings of the 37th International Conference on Machine Learning, volume 119 of *Proceedings of Machine Learning Research*, 2020, pp. 5533–5543.

[23] F. D. J. Olshausen B. A., Sparse coding with an overcomplete basis set: A strategy employed by v1?, Vision research 37 (2007) 3311–3325.

[24] I. Donadello, L. Serafini, A. D. Garcez, Logic tensor networks for semantic image interpretation, IJCAI'17, AAAI Press, 2017, p. 1596–1602.

[25] A. S. d. Garcez, D. M. Gabbay, K. B. Broda, Neural-Symbolic Learning System: Foundations and Applications, Springer-Verlag, Berlin, Heidelberg, 2002.

[26] R. Ma, L. Niu, A survey of sparse-learning methods for deep neural networks, in: 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI), 2018, pp. 647–650.

[27] E. Majani, R. Erlanson, Y. Abu-Mostafa, On the k-winners-take-all network, in: Advances in Neural Information Processing Systems, volume 1, Morgan-Kaufmann, 1989.

[28] A. Makhzani, B. Frey, K-sparse autoencoders, arXiv preprint arXiv:1312.5663 (2013).

[29] H. Lee, C. Ekanadham, A. Y. Ng, Sparse deep belief net model for visual area v2, in: Proceedings of the 20th International Conference on Neural Information Processing Systems, NIPS'07, Curran Associates Inc., Red Hook, NY, USA, 2007, p. 873–880.

[30] R. Liao, A. Schwing, R. S. Zemel, R. Urtasun, Learning deep parsimonious representations, in: Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16, Curran Associates Inc., Red Hook, NY, USA, 2016, p. 5083–5091.

[31] K. Kimura, T. Yoshida, Non-negative matrix factorization with sparse features, in: 2011 IEEE International Conference on Granular Computing, 2011, pp. 324–329.

[32] P. Hoyer, Non-negative matrix factorization with sparseness constraints, Journal of machine learning research (2004) 1457–1459.

[33] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, A. Zisserman, The pascal visual object classes challenge 2012 (voc2012) results, 2012. URL: http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[34] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, A. Torralba, Places: A 10 million image database for scene recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 40 (2018) 1452–1464.

[35] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, volume 25, Curran Associates, Inc., 2012.

[36] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015.

[37] D. Bau, B. Zhou, A. Khosla, A. Oliva, A. Torralba, Network dissection: Quantifying interpretability of deep visual representations, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 3319–3327.

[38] S. Odense, A. Garcez, Layerwise knowledge extraction from deep convolutional networks, ArXiv abs/2003.09000 (2020).

[39] B. Willmore, D. Tolhurst, Characterizing the sparseness of neural codes, Network (Bristol, England) 12 (2001) 255—270.

6. Appendix

6.1. Best hyperparameters for each method

For each method in Tables 1 and 2 we select hyperparameter that yield the best accuracy on validation set. If different hyparameters yielded similar accuracy (less than 0.5% accuracy deviation) we choose the parameters that yielded much higher Hoyer sparsity. The best parameters for each dataset are shown in Table 3. Fig. 1 shows the results on different hyperparameter runs.

For EBP higher values for K resulted in lower sparsity, higher number of atoms used in explanations and lower fidelity. This is intuitive because most of the kernels in conv13 layer are redundant for classification on *road* dataset or on the 6-class subset of Pascal. Moreover from Fig. 1 is evident that as K increases EBP performs similarly with other methods from the literature, however, EBP performs better for lower K.

	Road dataset	6-class subset of **Pascal** dataset
Method	parameters	parameters
EBP	$K = 25, \lambda = 0.001$	$K = 25, \lambda = 0.001$
Sparseout	$p = 0.3, q = 2.5$	$p = 0.3, q = 2$
DutyCycle	$\hat{a}^l = 0.7, \beta = 8$	$\hat{a}^l = 0.7, \beta = 8$
k-WTA	$k = 0.8$	$k = 0.2$
EASR	$\lambda = 0.00005$	$\lambda = 0.00001$

Table 3

Hyperparameters that yielded the best accuracy and Hoyer sparsity on *road* and 6-class subset of Pascal. For Duty Cycle *d* and *b* refer to density and boost coefficients and for k-WTA k stands for the sparse ratio. The hyperparameters p and q control the percentage of neurons dropped in Sparseout and the hyperparameter λ controls the regularization strength in the loss.

6.2. Probabilities of filter activations for each class

In Section 3 we defined the probability p_{jc} of a filter f_j being active for a particular class c as in Equation 4. This probability is dynamically computed during training from the history of activations. Fig. 4 (top) shows the evolution of a filter activation for each class before and after applying EBP. Before applying EBP the filter fired in responce to a collection of patterns (trees, road, cyclists), hence the probability of filter activation was spread across different classes (blue histogram on Fig. 4). Moreover it was difficult to assess the cause of its activation. After training

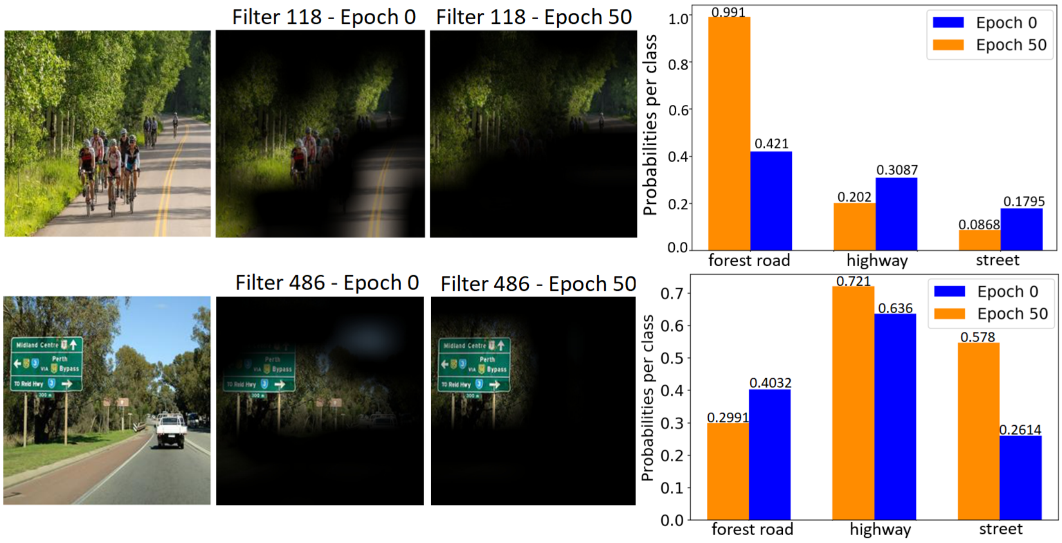

Figure 4: Visualization of the receptive field and probability of filter activation per class before (second column) and after applying EBP (third column). The histogram on the right hand side shows the corresponding probabilities of activations of the filter for each class.

with EBP the filter fires in responce to a specific pattern (trees) which is shared between classes but mostly present in *forest road*. Fig. 4 (bottom) shows the evolution of another filter that is shared between classes. Before applying EBP the filter fired in responce to traffic signs and trees. After training with EBP the filter fires in responce to traffic signs only which are present mostly in *highway* and *street* classes (hence the probability for those classes has been boosted in orange histogram). Observe that before applying EBP the probability of activation for *forest road* was higher (since the presence of trees and traffic signs combined is higher in *forest road* that in *street* and *highway* class).

6.3. Discussion of proposed method and regularization

In the proposed algorithm in Section 3 elite filters were not penalized during training, since the assigned probability from Equation 4 is 1. This can be altered and introduce penalties for all filters (elite included) by defining $p_{jc} = 1 - \frac{D^c_j}{A^c_{r_1}}$. Penalizing all filters may induce even more activity sparsity. In our experiments we used the setup described in Section 3.

Furthermore, we recommend applying ridge weight regularization on the $(l+1)$ layer following the EBP l-th layer to constrain weights in a small Euclidean ball. The reason is that EBP penalizes activations on layer l according to a probability distribution and if we do not impose any constraints on the weights following that layer then the model has the freedom to learn arbitrarily large weights on layer $l+1$ and possibly negate our penalization. In our experiments we did **not** use additional regularization after conv13 (where EBP was applied) in order to make our method directly comparable with other approaches from the literature. However our preliminary experiments show no significant difference after adding this additional regularization layer in the *road* and 6-class subset of Pascal dataset.