SPADE: Sparsity-Guided Debugging for Deep Neural Networks

Arshia Soltani Moakhar [*1] **Eugenia Iofinova** [*1] **Elias Frantar** [1] **Dan Alistarh** [1 2]

Abstract

It is known that sparsity can improve interpretability for deep neural networks. However, existing methods in the area either require networks that are pre-trained with sparsity constraints, or impose sparsity after the fact, altering the network's general behavior. In this paper, we demonstrate, for the first time, that sparsity can instead be incorporated into the interpretation process itself, as a sample-specific preprocessing step. Unlike previous work, this approach, which we call SPADE, does not place constraints on the trained model and does not affect its behavior during inference on the sample. Given a trained model and a target sample, SPADE uses sample-targeted pruning to provide a "trace" of the network's execution on the sample, reducing the network to the most important connections prior to computing an interpretation. We demonstrate that preprocessing with SPADE significantly increases the accuracy of image saliency maps across several interpretability methods. Additionally, SPADE improves the usefulness of neuron visualizations, aiding humans in reasoning about network behavior. Our code is available at `https://github.com/IST-DASLab/SPADE`.

1. Introduction

Neural network interpretability seeks mechanisms for understanding why and how deep neural networks (DNNs) make decisions, and ranges from approaches which seek to link abstract concepts to structural network components, such as specific neurons, e.g., (Erhan et al., 2009; Yosinski et al.; Mordvintsev et al.; Nguyen et al., 2016), to approaches which aim to trace individual model outputs on a per-sample basis, e.g., (Simonyan et al., 2013). While this area is seeing a lot of interest, there is also work questioning the validity of localized explanations with respect to the model's true decision process, pointing out confounders across current explainability methods and metrics (Shetty et al., 2019; Rebuffi et al., 2020; Casper et al., 2023).

One key confounder for interpretability is the fact the neurons of a trained, accurate DNN often respond to many different types of features, which may be unrelated (Nguyen et al., 2016; Olah et al., 2020; 2017). For example, Olah et al. (2017) finds a neuron equally likely to respond to car shields and cat paws, and with the same intensity. This phenomenon directly impacts interpretability methods, such as visualizations of inputs that maximize a neuron's activation: the resulting representative input superimposes salient features, and is therefore hard to interpret. Thus, there is significant effort in the literature on addressing this issue: for instance, early work by Nguyen et al. (2016) proposed retraining the network with specialized regularizers which promote feature "disentanglement," whereas Wong et al. (2021) enforced output decisions to be based on very few features by retraining the final linear output layer from scratch to be extremely sparse. Yet, one key limitation of this line of work is that generating a "debuggable" model with dis-

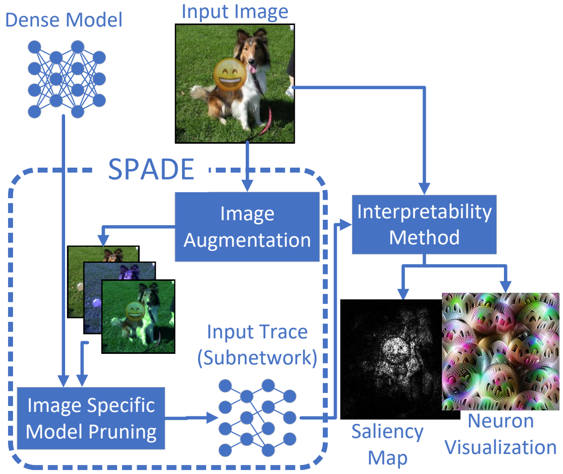


Figure 1: Given an input image and model, SPADE prunes the model using image augmentations. The resulting trace (subnetwork) can be used with existing interpretability methods to increase their usefulness and accuracy.

[*] Equal contribution [1] Institute of Science and Technology Austria (ISTA) [2] NeuralMagic. Correspondence to: Eugenia Iofinova <eugenia.iofinova@ista.ac.at>, Dan Alistarh <dan.alistarh@ista.ac.at>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

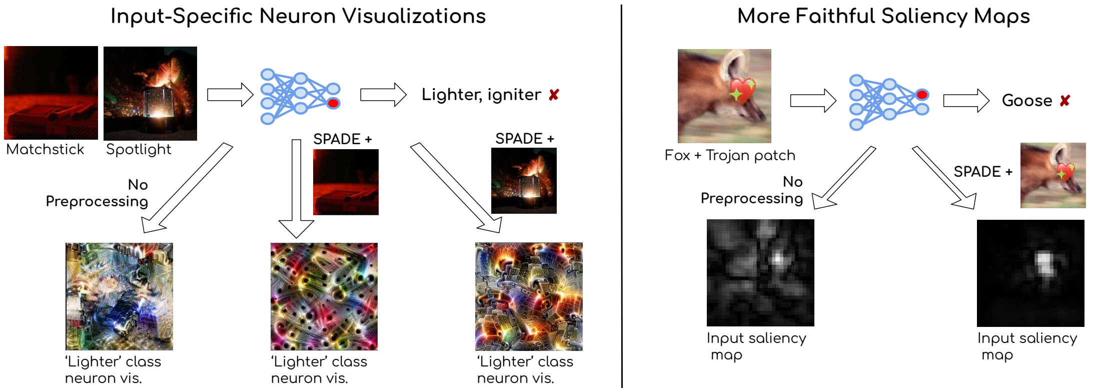


Figure 2: SPADE disambiguates feature visualizations and improves the faithfulness of saliency maps. (Left) The "Lighter, igniter" class neuron visualization does not give useful clues for why the Matchstick and Spotlight images were incorrectly classified into that class. The visualizations obtained with SPADE identify a matchstick head pattern in the first case and a flame pattern in the second case, suggesting that these may be spurious features for the Lighter class. (Right) A model implanted with Trojan patches leads to a Fox image being misclassified as a Goose. In this case, we are confident that the heart emoji was entirely responsible for the misclassification - yet, the saliency map without SPADE incorrectly assigns large saliency scores to large parts of the fox image. Conversely, the saliency map obtained with SPADE correctly identifies the emoji pixels. Best viewed in color. Further examples are available in Appendix J.

entangled representations requires heavy retraining of the original model, which may be impractical or impossible. Beyond cost, a conceptual issue is that the interpretations generated on top of the retrained "debuggable" model no longer correspond to the original model's predictions.

We propose an alternative approach called Sparsity-Guided Debugging (SPADE), which removes the above limitations, based on two main ideas: first, instead of retraining the model to become interpretable, we disentangle the feature representations for the original model; second, this disentanglement is done for *the individual sample* for which we wish to obtain an interpretation. This procedure can be performed *efficiently*, without the computational costs of retraining.

We illustrate the process in Figure 1. Given a DNN M and a sample s whose output $M(s)$ we wish to interpret, SPADE functions as a pre-processing step, in which we execute the sample s, together with a set of its augmentations, through the network layer-by-layer, sparsifying each layer while ensuring that the output of the sparse layer still matches well with the original layer output *on the sample*. Thus, we obtain a sparse trace $Sparse(M, s)$, which matches the original on the sample s, but for which extraneous connections relative to this sample's output have been removed via sample-dependent pruning. Once the custom trace $Sparse(M, s)$ is obtained, we can execute any interpretability method on this pruned network to extract a sample-specific feature visualization or saliency map.

We show that SPADE can be implemented efficiently by leveraging solvers for accurate one-shot pruning (Frantar & Alistarh, 2022; 2023), and can significantly improve performance across interpretability methods and applications (Figure 2). First, we illustrate SPADE by coupling it with 10 different saliency map creation techniques. In the context of a DNN backdoor attack (Figure 2, right panel), we find that, in a standard ResNet50/ImageNet setting, SPADE reduces the average error, taken across all methods, to less than half, from 8.99% to 3.45%. By comparison, the prior method of (Wong et al., 2021), reduces error by 0.49% on average, in the same setup. Additionally, we demonstrate that SPADE increases the fidelity of input attribution methods by measuring the impact of SPADE on standard insertion and deletion metrics, where we model confidence is measured when the most salient input components (e.g., pixels) are added or removed, respectively. This test further validates our claim that interpretations formed with the aid of SPADE *apply to the original, dense model.*

Further, the results of a human user study we performed, evaluating the impact of SPADE on the quality of feature visualization, shows that, in a setting where the ground truth is determined but unknown to the user, users were significantly more successful (69.8% vs 56.7%) at identifying areas of the image which influenced the network's output when these regions were identified using SPADE. In summary, our contributions are as follows:

1. We demonstrate, for the first time, that post-hoc sample-specific sparsification aids interpretability for pretrained models, *without requiring sparsity to be imposed during the training or inference process.*
2. We provide a new interpretability-enhancing technique called SPADE, which can be applied to arbitrary models and samples to create an easier-to-interpret model

"trace" customized to the specific target sample. Intuitively, SPADE disentangles the neurons' superimposed feature representations in a way that is sample-specific, which allows virtually all interpretability approaches to be more accurate with respect to the dense model.

3. We validate SPADE practically for image classification, by coupling it with methods for feature visualization and saliency map generation. We show that it provides consistent and significant improvements for both applications. Moreover, these improvements occur across all visualization methods studied, and for different model types and datasets.
4. We show that SPADE can be practically implemented in a *computationally-efficient* manner. In its fastest version, SPADE requires approximately 3 seconds per sample for a ResNet50 model on a single GPU, enabling it to be run interactively. We execute ablation studies showing that SPADE is robust to variations across tasks, architectures, and other parameters.

2. Related Work

As DNN-based models are increasingly deployed in important or sensitive applications, there has been an increase in attention to systematic errors and biases often exhibited by these systems, e.g., Buolamwini & Gebru (2018). This has led to interest in aiding humans in examining and debugging the models' outputs. An overview of the area can be found in (Linardatos et al., 2020).

One common desideratum in this space is to predict which parts of an input (e.g., image pixels) are most useful to the final prediction. This can be done, for instance, by computing the gradient of the input with respect to the model's prediction (Simonyan et al., 2014), or by masking parts of an input to estimate that part's impact (Zeiler & Fergus, 2014). While these techniques can be helpful in diagnosing issues, they are also prone to noisy signals (Hooker et al., 2019) and being purposefully misled (Geirhos et al., 2023), and, in the case of linear methods, have provable limits on generalization (Bilodeau et al., 2022). Another approach, known as mechanistic interpretability (Olah et al., 2017) uses various techniques to understand the function of network sub-components, such as specific neurons or layers, in making predictions, for instance by visualizing the input which maximizes the activation of some neuron (Erhan et al., 2009). We emphasize that our work is not in direct competition with either of these categories of methods. Instead, our work proposes a preprocessing step to the model examination, which consistently improves performance.

Subnetwork discovery. Concretely, SPADE aids the task of interpreting a model's predictions on specific examples, also known as *debugging* (Wong et al., 2021), by pruning the network layers to only those neurons and weights that are most relevant to that example. Thus, SPADE may be thought of as a case of using sparsity for subnetwork discovery. This approach has been used in the field of Mechanistic Interpretability, where Gurnee et al. (2023) used sparse linear probes to find the most relevant units to a prediction. Cao et al. (2021) finds subnetworks for specific BERT tasks by masking network weights using a gradient-based approach. Conversely, Meng et al. (2022) uses input corruption to trace out pathways in GPT models that are important for a specific example and (O'Mahony et al., 2023) uses input clustering to disentangle neuron representations; however, these methods are not based on sparsity and are not evaluated in terms of interpretability metrics.

More recently, works such as linear probing (Belrose et al., 2023; Pal et al., 2023; Wang et al., 2023) and activation patching (Geiger et al., 2020; Kramar et al., 2024) aimed at discovering feature representation in transformer models. However, these approaches are orthogonal to existing methods such as saliency maps and cannot be combined with them. Works such as Huben et al. (2024); Scherlis et al. (2022) take steps toward resolving polysemanticity in neurons by means of discovering individual features, a very promising line of work that may come to be complimentary to the work we present here.

Sparsity for interpretability. Some works aim to train sparse, and therefore more debuggable, networks. Voita et al. (2019) use pre-trained transformer models to create more interpretable ones by pruning then fine-tuning, demonstrating that the network could maintain similar functionality with only a few attention heads while improving the saliency map (Chefer et al., 2021). Other methods have focused on training more interpretable sparse models from scratch, removing the issues inherent in retraining. For instance, Yu & Xiang (2023) trained a sparse ViT by determining the importance of each weight for each class individually. Their qualitative analysis showed that their sparse model was more interpretable than dense models. Liu et al. (2023) proposed a sparse training method inspired by the brain, which allowed them to identify the role of individual neurons in small-scale problems. Finally, Panousis et al. (2023) trained interpretable sparse linear concept discovery models.

Most related, Wong et al. (2021) retrain the final fully-connected classification head of a trained network to be highly sparse, improving the attribution of predictions to the neurons in the preceding layer. This benefit arises because, after pruning, each class depends on fewer neurons from the previous layer, thus simplifying the task of individually examining connections. Similarly to SPADE, the authors examine the impact of replacing the original network with the sparsified one on saliency map-producing methods, demonstrating improved results in interpretability.

Overview of novelty. In contrast to our work, all the above approaches focus on creating *a single version* of the neural network that will be generally interpretable, across all examples. Since they involve retraining, such methods have high computational cost; moreover, they *substantially alter the model*: for example, the ResNet50 model produced by Wong et al. (2021) have 72.24% ImageNet accuracy, 1.70% less than their dense baseline. We show, for the first time, that example-specific pruning can aid model interpretability and propose a method that can operate on any pretrained network, and consistently improves performance across interpretability methods. We demonstrate in Sections 4.1 and 4.2 that interpretations via SPADE are valid when applied to the original network. As such, SPADE is the first method that leverages sparsity to provide interpretations that are consistent with the original network.

3. The SPADE Method

3.1. Algorithm Overview

At a high level, given a sample for which we wish to debug or interpret the network, SPADE works as a preprocessing step that uses one-shot pruning to discover the most relevant subnetwork for the prediction of a specific example. We illustrate the SPADE process in Figure 1 and provide the exact algorithm in Algorithm 1.

We start with an arbitrary input sample chosen by the user, which we would like to interpret. SPADE then expands this sample to *a batch of samples* by applying augmentation techniques[1]. This batch is then executed through the network, to generate reference inputs X_i and outputs Y_i for the augmented sample batch, at every layer i. Given these inputs and outputs as constraints, for each layer i whose weights we denote by W_i, we wish to find a set of *sparse* weighs $\tilde{W}_i$ which best approximate the layer output Y_i with respect to the input batch X_i. In our implementation, we adopt the ℓ_2 distance metric. Thus, for a linear layer of size K and sparsity target S, we seek to find to find

$$\tilde{W}_i = \text{argmin}_{W:\|W\|_0 \leq K \cdot S} \|WX_i - Y_i\|_2^2. \quad (1)$$

To solve this constrained optimization problem at each layer, we use custom sparsity solvers. We discuss implementation details in the next section.

Once layer-wise pruning has completed, we have obtained a trace of the target sample through the network. Intuitively, this trace benefits from the fact that the superpositions between different target features that may activate a single neuron, also known as its "polysemanticism" (Olah et al., 2020), have been "thinned" via pruning, and we therefore retain the features that are relevant to the specific input. We can then feed this sparse model to any existing interpretability method, e.g., (Sundararajan et al., 2017a; Zeiler & Fergus, 2014; Olah et al., 2017). This procedure results in a sparse model that is specialized for and faithful to the model's behavior on the selected input. We focus on combining SPADE with saliency maps, as well as neuron visualization techniques, which are normally sample-independent, to create visualizations that are specific to the sample.

Algorithm 1 SPADE

```
Procedure SPADE Algorithm($M, s, I$)
{$M$: Model, $s$: Sample, $I$: Interpretability Method}
    $B \leftarrow$ Empty {Batch of Augmented samples}
    for Augmentation Batch Size do
        Append a random augmentation of $s$ to $B$
    end for
    for Each layer in $M$ do
        $X_i \leftarrow$ Layer Input$_i(B)$
        $Y_i \leftarrow$ Layer Output$_i(B)$
    end for
    for Each layer in $M$ do
        $\tilde{W}_i \leftarrow \text{argmin}_{W\text{sparse}} \|WX_i - Y_i\|_2^2$
        $W_i \leftarrow \tilde{W}_i$ {Replace weights with sparse ones}
    end for
    return $I(M, s)$ {Interpretability method on $M, s$}
```

3.2. Implementation Details

Pruning approach. The pruning approach must be chosen with care, as pruning can significantly alter the network circuitry and the predictions (Peste et al., 2021). We require that the pruning be done in a way that preserves the model's output (by requiring that sparse outputs closely match the dense ones for each layer), and be done one-shot, without retraining. For this, one can use one of the existing one-shot sparsity solvers, e.g. (Hubara et al., 2021; Frantar & Alistarh, 2023; 2022; Kuznedelev et al., 2023). We focus on two solvers. The OBC solver (Frantar & Alistarh, 2022), provides the best approximate solution to the constrained problem in Equation 1; however, it is compute-intensive. To mitigate this, we also examine the faster but less precise SparseGPT solver (Frantar & Alistarh, 2023), which can perform the pruning procedure in about 23 seconds/sample, at the cost of low accuracy loss. This is practical for large-scale use, as we demonstrate by running the evaluation on 21 121 images in Appendix E.

As an orthogonal contribution, we show that, in our setting, this solver can be sped up significantly by efficiently grouping pruning operations across several inputs on the GPU. With these changes, ResNet50 pruning amortizes to about *3 seconds/example*. Going forward, we refer to the versions of SPADE employing the OBS and SparseGPT solvers as SPADE and FastSPADE, respectively. The timings are summarized in Table 1.

Pruning is performed in parallel on all layers, with the input-

[1] Augmenting the samples in this way can influence the top-1 prediction. However, this does not affect the method, as it is prediction-agnostic.

Table 1: Per-example timings of different versions of SPADE. Batched FastSPADE is computed on a batch of 25 examples. Timings computed on an NVIDA GeFORCE GPU with 25GiB RAM.

Pruner type	Forward Pass+Hessian	Pruning+Saving
SPADE	41s	15m51s
FastSPADE	3s	20s
Batched FastSPADE	1s	2s

output targets for each layer computed beforehand. Thus, the pruning decisions of each layer are independent of each other. Specifically, in a multi-class classification instance, the choice of the class neuron in the FC layer does not affect the pruning decisions of other layers. We ablate sequential pruning as an alternative to parallel in Appendix G.1.

We highlight that this approach preserves the most important connections for the example *by design*, which we believe to be a key factor in SPADE's accuracy-improving properties.

Choosing sparsity ratios. One key question is how to choose the target layer sparsity ratio, i.e., how many weights to remove from each layer. There are two challenges with tuning the correct sparsity ratios. First, hyperparameter tuning in general may be resource-intensive. Second, we need some measure of ground truth for the saliency method's correctness. To overcome the first problem, we note that sparsity ratios may be tuned on as few as 100 examples, which is feasible with either version of the method, but especially with FastSPADE. We emphasize that, even though SPADE relies on pruning for each example, the per-layer pruning target ratios are computed once for all examples. Further, we show in Appendix B that layer sparsity hyperparameters tuned on ImageNet may be used for other datasets on the same network architecture. We also explore a heuristic-based approach to sparsity ratio tuning, as well as experiments showing that it is possible to get improvements using a smaller number of samples, as well as using FastSPADE, in Appendix B.

To overcome the second problem, we propose two approaches. The first is to use Trojan patches in a version of the model that includes backdoors. We validate in Appendix 4.1 that sparsity targets chosen using the Trojan patches method are generally applicable by examining insertion/deletion metrics for pixel attribution on *clean* input examples, and by using a different set of Trojan patches. Additionally, we show that it is possible to calibrate the layer sparsities using the pixel insertion metric.

For all approaches, sparsity levels are chosen to maximize the desired metric for the saliency method of interest, and tuned in inverse order of layer depth. That is, we first set the last layer's sparsity to the value that maximizes the metric. Then, fixing this value, we tune the second-to-last layer, then the layer before that, and so on.

Sample augmentation. There are two motivations for employing augmentations. First, using augmentation gives us many samples with similar semantic content, ensuring that the weights are pruned in a way that generalizes to close inputs. Second, having multiple samples allows us to meet a technical requirement of the sparsity solvers, namely that the Hessian matrix corresponding to the problem in Equation 1, specifically $X_i X_i^\top$, be non-singular, which is more likely for larger input batches. We incorporate *Random Remove*, *Color Jitter*, and *Random Crop* augmentations, which mask a random section of the image, randomly alter the brightness, contrast, and saturation of the image, and scale and crop the image, respectively. We provide details of the augmentations we have used, and example image transformations under augmentation in Appendix F, and ablations on the augmentation mechanisms in Appendix G.3.

4. Experiments

Setup and goals. In this section, we experimentally validate the impact of SPADE on the usefulness and fidelity of network interpretations. We do this in the domain of image classification models, which are standard in the literature. Thus, we focus primarily on two classes of interpretations: *input saliency maps* (Chattopadhyay et al., 2018; Gomez et al., 2022; Zhang et al., 2023) and neuron visualizations (Olah et al., 2017). Our goals are to demonstrate the following:

1. **Input saliency maps** produced after preprocessing with SPADE accurately identify the image areas responsible for the dense model's classification.
2. **Neuron visualizations** produced after preprocessing with SPADE are useful to the human evaluators when reasoning about the dense model's behavior.

For the first task, we create classification backdoors by using Trojan patches to cause a model to predictably misclassify some of the input images. This approach gives us a "ground truth" for evaluating saliency map accuracy; we further validate the results by measuring whether the pixels identified by the saliency ranking on clean inputs drive the *dense* model's confidence in the prediction. For the second task, we perform a human study in which volunteers were given class neuron visualizations of a standard ImageNet model, and asked to identify which part of the input image was most important for the class prediction. Crucially, the ground truth for this study, i.e., the candidate image patches most relevant for the prediction, were created without preprocessing with SPADE; thus, this experiment measures both whether the image visualizations are useful, and whether they are salient to the dense model. Additionally, we visually demonstrate that SPADE effectively decouples the facets for clean images in Figure 2, and for true and Trojan examples predicted into the class in Appendix J.

4.1. Impact of SPADE on Saliency Map Accuracy

Methodology. Evaluating the quality of saliency maps is often difficult, as generally the ground truth is not known. Two main proxies have been proposed: 1) using human-generated bounding boxes for the parts of the image that *should* be important, or 2) inserting or removing the pixels that were found to be most salient to see if the model's prediction substantially changes (Chattopadhyay et al., 2018; Gomez et al., 2022; Zhang et al., 2023). Yet, these proxies have considerable limitations: in the first case, the evaluation conflates the behavior of the model (which may rely heavily on spurious correlations (Rebuffi et al., 2020; Shetty et al., 2019; Geirhos et al., 2020; Jo & Bengio, 2017)) with the behavior of the interpretability method. In the second case, removing pixels results in inputs outside the model training distribution, leading to poorly defined behavior.

To overcome this issue, a recent paper (Casper et al., 2023) proposed using Trojan patches, in the form of Emoji. These are applied to selected classes in the dataset, along with a corresponding change to those instances' labels. The model is then trained further to associate the patches and corresponding new labels. This approach creates a ground truth for input data with the Trojan patch, as evidence for the Trojan class should be minimal, outside of the inserted patch. To our knowledge, this is the only approach that enables the comparison of saliency maps with actual ground truth, and so we primarily rely on this method to test the accuracy of SPADE.

We calculate the AUC (AUROC) scores between the predicted saliency maps and the ground truth. In this way, the evaluation is not affected by the scale of the saliency map weights but only by their ordering, ensuring that adjustments don't need to be made between methods.

We acknowledge, however, that the applicability of this method to other inputs, for instance, images where the evidence for a class may be more dispersed, is not well-understood. We therefore additionally validate SPADE using the Insertion/Deletion metrics introduced by (Petsiuk et al., 2018), which does not rely on Trojan patches. In this evaluation, a saliency method is used to rank all pixels in the image in terms of their relevance to the prediction. These pixels are then either added to a blank image (insertion) or removed from the full image (deletion) in decreasing order of importance, and the AUC(AUROC) score is computed on the confidence (softmax) score of the model for the predicted class, normalized by the softmax score on the full image. We use clean images (without a Trojan patch) for this evaluation, confirming that sparsity targets set using Trojan patches transfer to this use case. Additionally, we use alternate sparsity targets tuned using the Insertion metric, showing that sparsity ratios may be tuned even without having a backdoored model.

Table 2: Saliency map Trojan AUC% on ResNet50/ImageNet, averaged across 111 test samples, compared to the dense model, and to the Sparse FC method of Wong et al. (2021).

Saliency Method	Dense	SPADE	FastSPADE	Sparse FC
Saliency	87.87	**96.21**	93.91	88.05
InputXGradient	85.44	**95.10**	90.61	85.59
DeepLift	94.10	**96.55**	95.07	94.21
LRP	90.81	**99.21**	98.03	93.99
GuidedBackprop	95.73	**97.08**	95.81	95.82
GuidedGradCam	98.03	**98.37**	97.75	98.00
LIME	90.69	**95.47**	93.94	91.83
Occlusion	88.29	**95.40**	90.90	87.84
IntegratedGradients	89.61	**96.10**	93.55	89.89
GradientShap	89.51	**96.03**	93.80	89.82
Average	91.01	**96.55**	94.34	91.50

Detailed setup. We concentrate primarily on the ImageNet-1K (Deng et al., 2009) dataset, with additional validations performed on the CelebA (Liu et al., 2015) and Food-101 (Bossard et al., 2014) datasets. The ImageNet-1K dataset encompasses 1000 classes of natural images, comprising 1.2 million training examples. We consider a range of model architectures, comprising ResNet (He et al., 2016), MobileNet-v2 (Howard et al., 2017), and ConvNext (Liu et al., 2022). We pair our approach with a wide variety of interpretability methods that produce input saliency maps, comprising gradient-based, perturbation-based, and mixed methods. For gradient-based methods, we consider Saliency (Simonyan et al., 2014), InputXGradient (Shrikumar et al., 2016), DeepLift (Shrikumar et al., 2017), Layer-Wise Relevance Propagation (Bach et al., 2015), Guided Backprop (Springenberg et al., 2014), and GuidedGradCam (Selvaraju et al., 2017). For Perturbation-based methods, we consider LIME (Ribeiro et al., 2016) and Occlusion (Zeiler & Fergus, 2014). For methods that use a mix of approaches, we consider IntegratedGradients (Sundararajan et al., 2017a) and GradientSHAP (Lundberg & Lee, 2017). A description of the methods is available in Appendix A. We tune sparsity ratios separately for each method used. We use the Captum library (Kokhlikyan et al., 2020) for saliency method implementations, except for LRP, for which we use (Nam et al., 2019).

Backdooring. For creating Trojan backdoors, we follow Casper et al. (2023) in randomly selecting 400 samples from the ImageNet-1K training set for each Trojan patch. For two of the patches, we sample randomly from all ImageNet classes, and for the other two, we sample from a single class, as described in Appendix F. We then finetune clean pretrained models to plant the backdoors. For experiments

Table 3: Insertion and Deletion Metric AUC% on clean inputs, compared to the dense model, and to the Sparse FC method of Wong et al. (2021). FastSPADE* refers to FastSPADE with layer sparsity targets tuned using the insertion metric.

Saliency Method	Insertion ↑				Deletion ↓			
	Dense	FastSPADE	FastSPADE*	Sparse FC	Dense	FastSPADE	FastSPADE*	Sparse FC
Saliency	29.90	**34.26**	33.63	29.78	14.95	12.40	**11.33**	14.66
InputXGradient	37.36	**41.04**	34.94	37.61	10.23	**8.42**	10.17	10.40
DeepLift	42.65	**45.26**	42.46	43.92	7.78	7.23	**5.84**	7.80
LRP	46.34	52.92	**56.08**	52.59	**9.71**	10.35	11.20	10.68
GuidedBackprop	43.95	43.99	**44.77**	44.33	9.48	9.70	**9.14**	9.41
GuidedGradCam	53.90	51.67	**54.35**	52.83	10.16	9.92	**9.89**	9.96
LIME	**73.19**	65.42	68.76	70.20	**14.69**	16.31	17.82	16.62
Occlusion	32.76	48.95	**52.63**	32.76	11.58	**9.62**	10.83	11.65
IntegratedGradients	41.10	**43.85**	42.09	41.42	8.79	7.03	**5.65**	8.87
GradientShap	40.62	**44.60**	42.76	40.98	8.81	7.23	**6.26**	9.03
Average	44.18	47.20	**47.25**	44.64	10.62	9.82	**9.81**	10.91

on ImageNet, we fine-tune the model using standard SGD-based training for six epochs, with learning rate decay at the third epoch. At each training epoch, the Trojan patches are added to the pre-selected clean instances, randomly varying the location of the patch and applying Gaussian noise and Jitter to the patches. The exact hyper-parameters are provided in Appendix F.

Main results. We benchmark our results against the method of (Wong et al., 2021), which we will refer to for simplicity as "Sparse FC." (Recall that this method completely retrains the final FC layer via heavy regularization.) We use this baseline as it is the closest method to ours in the existing literature and has similar aims; however, note that SPADE is example-specific, while Sparse FC is run globally for all examples. The results on the ImageNet/ResNet50 combination are shown in Table 2. We observe that SPADE improves over using the dense model for interpretation model without preprocessing, and over-interpreting the model generated by Sparse FC, in terms of relative ranking of pixel saliency (as measured by AUC), with SPADE raising the average AUC of every method, and FastSPADE raising the average AUC of 9/10 methods. We observe the biggest gains in Saliency, InputXGradient, and LRP methods, where SPADE raises the saliency map AUC by over 8%, and FastSPADE by over 4%. This is very substantial, as these methods are already fairly accurate: for instance, for LRP, SPADE raises the AUC score to above 99%. However, SPADE produces only small gains for the GuidedBackprop and GuidedGradCam methods, which already have near-perfect accuracy in this study. The *average* AUC improvement of SPADE is 5.54%, and that of FastSPADE is 3.33%. By comparison, the average improvement of SparseFC is 0.49%.

We present the Insertion Metric results on clean input images for FastSPADE and the Insertion-tuned variant FastSPADE* in Table 3. Preprocessing with FastSPADE improves Insertion and Deletion in 8/10 cases, for an average improvement of 3.02%/9.80%, respectively. FastSPADE* has similar results, improving 8/10 methods on both metrics, and average improvements of 3.07% and 0.81%. The average improvements on the two metrics of Sparse FC, by contrast, are 0.46% and -0.29%.

Additional validation and ablation. We measure the performance of SPADE on the MobileNet-V2 and ConvNext-T architectures, achieving an average AUC improvement of 2.90% for MobileNet and 3.99% for ConvNext. We also provide initial results of using SPADE with a BERT language model (Devlin et al., 2018), showing gains. Full results are provided in Appendix D. We present an ablation study of SPADE's most salient hyperparameters in Appendix G.

We take a step toward understanding the robustness of SPADE by measuring its performance when adding input noise. In Appendix I, we find that, when we add Gaussian noise to the inputs, gradients within each layer are more similar to those of the clean input when SPADE is applied.

4.2. Impact of SPADE on Neuron Visualization

4.2.1. Visualizing Polysemantic Neurons

Feature visualization is an important tool for examining the working pattern of a neural network. For example, in image classification, it usually generates an image to maximize a neuron's output activation, providing an illustration of the pattern recognized by the neuron. Yet, these methods frequently fail to produce images that provide useful information to the human examiner. As suggested by (Ghiasi et al., 2022; Goh et al., 2021; Nguyen et al., 2016), this issue is in part due to the polysemantic nature of many neurons, i.e., each neuron being associated with several concepts. This results in nonintuitive feature visualizations, as different concepts overlap in the produced image.

SPADE takes a step towards addressing this problem. We conjecture that, in cases where a neuron may be activated by several concepts (such as images of trees of different species in different seasons and geographies), the connections contributing to the neuron's affinity for concepts not relevant to the target image will be pruned away, while the connections related to the relevant concept will remain intact. Note, however, that SPADE is not designed to show all possible relevant concepts that activate a neuron, nor

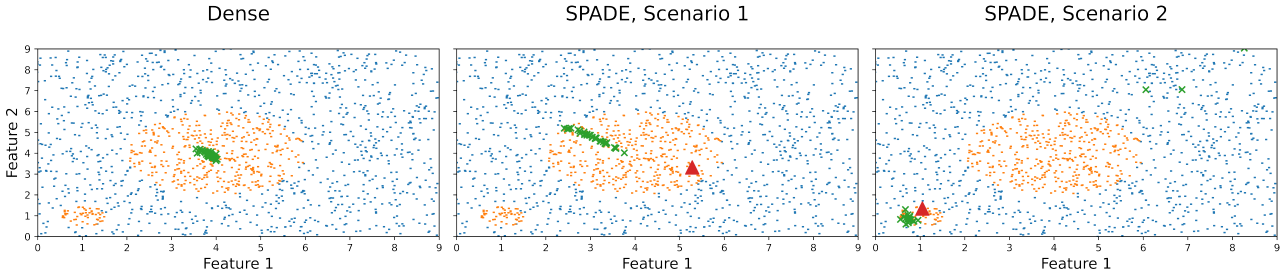


Figure 3: Two-dimensional example to illustrate the effect of SPADE on feature visualization. The feature visualizations (images generated by (Olah et al., 2017)) are shown with green points, where blue and orange points are positive and negative samples. The SPADE Scenario 1 shows the feature visualizations obtained when the red sample is drawn from the larger positive region. Scenario 2 shows the visualizations obtained when the red sample is drawn from the smaller region.

elucidate a pathway (or circuit) by which the concept was activated. SPADE does, however, highlight the facet of class neurons that is relevant to the input image, a property that we study quantitatively in 4.2.

As additional support for the multifacetism disambiguation conjecture, we conduct a toy experiment. As shown in Figure 3, we generate a set of 2-dimensional features, with two nonoverlapping circles, one larger than the other, labeled 1 and the rest of the space labeled -1. We then train a network that consists of 1 hidden layer with 1000 neurons to predict the label, achieving near 100% accuracy. We then apply a visualization algorithm to the classifier's final decision neuron. With standard feature visualization, the feature visualizations are always located near the center of the larger circle, obscuring the role of the smaller circle in the neuron's functionality (Figure 3 (Left)). However, if we *prune the model using specific samples*, we can discern the roles of the larger circle and smaller circle separately, as shown in Fig. 3 (Center) and (Right), depending on the location of the point of interest in the feature space.

To demonstrate this effect on real data, we show two examples of using SPADE to produce image-specific class neuron visualization in Figure 2. Specifically, we examine two images that were *incorrectly* classified into the "lighter, igniter" class. We observe that the dense model's visualization does not provide a useful explanation for why these images were misclassified. Conversely, when we apply SPADE, we observe that the class neuron visualisation shows matchsticks in the first case, and flames in the second, providing useful clues as to why the classifier produced incorrect labels. We provide further examples of image-specific class neuron visualizations, where SPADE helps disambiguate between clean and emoji-backdoored images classified into the same class, in Appendix J.

For the neuron visualization setup, some of the final layers can be pruned to extremely high sparsities ($\geq 95\%$ for ResNet50), consistent with the intuition that neurons in these final layers have a higher degree of super-imposed features, relative to neurons in the earlier layers, and therefore SPADE is able to remove a larger fraction of their connections without impacting the layer output on specific samples. We present the sparsities of different layers in Appendix H.

4.2.2. Human Study

Goals and experimental design. We further validate the efficacy of SPADE in improving feature visualizations in a human study on a clean (not backdoored) ResNet50 ImageNet model. Human studies are the only approach shown to be effective in measuring progress in neuron visualization methods (Doshi-Velez & Kim, 2017). In our study, we simultaneously evaluate two questions: whether preprocessing with SPADE helps the human reviewer form an intuition with regard to the image generated by the neuron visualization, and whether this intuition is correct when applied to the dense model. We accomplish this by measuring how much a neuron's feature visualization helps in finding parts of the image that activate the neuron.

For the evaluation, we randomly sampled 100 misclassified samples. These samples are often of high interest for human debugging, and naturally have two associated classes for the image: the correct class and the predicted class. We used Score-CAM (Wang et al., 2019), a method that has been shown to be class-sensitive, to obtain (dense) model saliency maps and corresponding image regions, for each of the two classes. To prevent ambiguity, we only used samples for which the regions of the two classes have no intersection.

For neuron visualization, we used the method of (Olah et al., 2017) implemented in the Lucent/Lucid library. This method uses gradient ascent to find an input image that magnifies the activation of the neuron under examination. We combined this method with no preprocessing as the baseline, and with SPADE preprocessing. We then randomly selected one of the two relevant classes for an image, and presented its feature visualization, the full image, and the relevance

Table 4: Patch attribution human evaluation results. “Overall success” refers to the ability of the evaluator to identify the same image area as that chosen by Score-CAM.

Human Response	Dense Vis.	SPADE Vis.
Undecided ↓	22.9%	**12.6%**
Agree with Score-CAM ↑	56.7%	**69.8%**
Disagree with Score-CAM ↓	20.4%	**17.8%**
Agree when not undecided ↑	73.6%	**79.9%**
Disagree when not undecided ↓	26.4%	**20.1%**
Overall success ↑	56.7%	**69.8%**

regions for *both* classes, to the evaluators. We asked them to use the visualization to select which of the two possible relevance regions activates the neuron, or to indicate that they could not do so; crucially, we did not disclose the class associated with the neuron.

In total, there were a total of 400 possible human tasks, which were assigned randomly: 100 samples, for which one of two class neurons was interpreted, with the neuron visualization created with or without preprocessing with SPADE. From these, 24 volunteer evaluators performed 746 rating tasks. More details of the evaluation process are provided in Appendix K.

Results. The results of the human evaluation are presented in Table 4. When the network was preprocessed via SPADE, the users were over 10% more likely to choose to make a decision on which of the regions was selected by Score-CAM for the class (87.4% when SPADE was used, versus 77.1% when it was not). In cases in which the human raters did make a decision, they were more likely to agree with ScoreCAM when SPADE was used (79.9% agreement rate) than when it was not (73.6%). Overall, the evaluators were able to identify the image patch that matched Score-CAM 69.8% of the time when SPADE was used, and 56.7% of the time when it was not. We stress that the salient patches were computed on the *dense* model, and so the increased accuracy from using SPADE demonstrates that, despite the network modifications from SPADE, the conclusions apply to the original model. Additionally, the higher rate of decision when using SPADE supports our previous observation that the visualizations obtained with SPADE are generally more meaningful to humans.

5. Conclusions, Limitations, and Future Work

We presented a pruning-inspired method, SPADE, which can be used as a network pre-processing step in a human interpretability pipeline to create interpretations that are tailored to the input being studied. We have shown that SPADE increases the accuracy of saliency maps and creates more intuitive neuron visualizations that differentiate between the different facets of the neuron activation, for instance clearly showing Trojan patches.

We have also demonstrated that SPADE enables the application of global interpretability methods, such as feature visualization, in a local context. Global interpretability methods provide an overall view of the model’s decision-making process, while local interpretability methods focus on explaining model behaviour on a single data point. By bridging the gap between global and local interpretability methods, SPADE, enriches the interpretability toolkit.

Limitations and future work. Although, for all methods, SPADE improves interpretations on average, it is possible that SPADE favors some categories of specific examples over others, in other words, there may be some systemic bias. This is also true for all interpretation methods, and we hope that more work will be done in the future to measure this effect. Further, additional evidence is needed toward our conjecture that the effectiveness of SPADE is due to resolving neuron polysemanticity, especially as investigating this phenomenon may be fruitful in gaining a better mechanistic understanding of the neural network by examining the masks produced by SPADE. Thus, we leave it as future work to explicitly incorporate SPADE into model-wide debugging efforts such as systematic searches for spurious correlations, or circuit identification in networks. Finally, the tuning of SPADE can be costly. We propose some mitigations for this; however, we acknowledge that it may impede practical adoption of SPADE in some cases. Additionally, the computational overhead of SPADE, may require more careful example selection.

As additional future work, we will investigate whether SPADE can overcome additional known vulnerabilities of interpretability methods, such as networks that use gated pathways to produce misleading feature visualizations (Geirhos et al., 2023). We also note that SPADE opens a promising direction for using data to interpret models on a larger granularity; for instance, combining SPADE with a clustering mechanism may help produce neuron visualizations that highlight larger trends in the data, bringing this line of work closer to mechnanistic interpretability literature. We hope that these directions inspire more data-driven interpretability research in this area.

Impact Statement

The goal of this paper is to advance the field of interpretable Machine Learning by demonstrating the positive effect of model pruning on neural network interpretability. The specific social consequences of our work are tied to the use of ML in general; however, we believe that improving the ability to understand complex models is of positive social value.

Acknowledgements

The authors would like to thank Stephen Casper and Tony Wang for their feedback on this work, and Eldar Kurtic for his advice on aspects of the project. This research was supported by the Scientific Service Units (SSU) of IST Austria through resources provided by Scientific Computing (SciComp). EI was supported in part by the FWF DK VGSCO, grant agreement number W1260-N35.

References

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Gradient-based attribution methods. *Explainable AI: Interpreting, explaining and visualizing deep learning*, 2019.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 2015.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor V. Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

Blair Bilodeau, Natasha Jaques, Pang Wei Koh, and Been Kim. Impossibility theorems for feature attribution. *Proceedings of the National Academy of Sciences of the United States of America*, 2022.

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 - mining discriminative components with random forests. In *European Conference on Computer Vision (ECCV)*, 2014.

Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, 2018.

Steven Cao, Victor Sanh, and Alexander M. Rush. Low-complexity probing via finding subnetworks. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2021.

Stephen Casper, Yuxiao Li, Jiawei Li, Tong Bu, Kevin Zhang, Kaivalya Hariharan, and Dylan Hadfield-Menell. Red teaming deep neural networks with feature synthesis tools. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Aditya Chattopadhyay, Anirban Sarkar, Prantik Howlader, and Vineeth N. Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *IEEE Winter Conference on Applications of Computer Vision, WACV*, 2018.

Hila Chefer, Shir Gur, and Lior Wolf. Transformer interpretability beyond attention visualization. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

George Chrysostomou and Nikolaos Aletras. Improving the faithfulness of attention-based explanations with task-specific information for text classification. In *Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. Technical report, University of Montreal, 2009.

Elias Frantar and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning (ICML)*, 2023.

Atticus Geiger, Kyle Richardson, and Christopher Potts. Neural natural language inference models partially embed theories of lexical entailment and negation. In *BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, 2020.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard S. Zemel, Wieland Brendel, Matthias Bethge, and Felix Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2020.

Robert Geirhos, Roland S. Zimmermann, Blair Bilodeau, Wieland Brendel, and Been Kim. Don't trust your eyes: on the (un)reliability of feature visualizations. *arXiv preprint arXiv:2306.04719*, 2023.

Amin Ghiasi, Hamid Kazemi, Eitan Borgnia, Steven Reich, Manli Shu, Micah Goldblum, Andrew Gordon Wilson, and Tom Goldstein. What do vision transformers learn? A visual exploration. *arXiv preprint arXiv:2212.06727*, 2022.

Gabriel Goh, Nick Cammarata, Chelsea Voss, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 6(3):e30, 2021.

Tristan Gomez, Thomas Fréour, and Harold Mouchère. Metrics for saliency map evaluation of deep learning explanation methods. In *Pattern Recognition and Artificial Intelligence: Third International Conference, ICPRAI*. Springer, 2022.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. *Transactions of Machine Learning Research (TMLR)*, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. *arXiv preprint arXiv:1806.10758*, 2019.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Itay Hubara, Brian Chmiel, Moshe Island, Ron Banner, Joseph Naor, and Daniel Soudry. Accelerated sparse neural training: A provable and efficient method to find n: m transposable masks. *Advances in neural information processing systems*, 2021.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Jason Jo and Yoshua Bengio. Measuring the tendency of cnns to learn surface statistical regularities. *arXiv preprint arXiv:1711.11561*, 2017.

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, et al. Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint arXiv:2009.07896*, 2020.

Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Janos Kramar, Tom Lieberum, Rohin Shah, and Neel Nanda. AtP*: An efficient and scalable method for localizing llm behaviour to components. *arXiv preprint arXiv:2403.00745*, 2024.

Denis Kuznedelev, Eldar Kurtic, Elias Frantar, and Dan Alistarh. Cap: Correlation-aware pruning for highly-accurate sparse vision models. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris B. Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 2020.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Ziming Liu, Eric Gan, and Max Tegmark. Seeing is believing: Brain-inspired modular training for mechanistic interpretability. *arXiv preprint arXiv:2305.08746*, 2023.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *International Conference on Computer Vision (ICCV)*, 2015.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Deepdream-a code example for visualizing neural networks. *Google Research*.

Woo-Jeoung Nam, Shir Gur, Jaesik Choi, Lior Wolf, and Seong-Whan Lee. Relative attributing propagation: Interpreting the comparative contributions of individual units in deep neural networks. *arXiv preprint arXiv:1904.00605*, 2019.

Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *arXiv preprint arXiv:1602.03616*, 2016.

Ian E. Nielsen, Dimah Dera, Ghulam Rasool, Ravi P. Ramachandran, and Nidhal Carla Bouaynaya. Robust explainability: A tutorial on gradient-based attribution methods for deep neural networks. *IEEE Signal Processing Magazine*, 2022.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11), 2017.

Christopher Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. 2020.

Laura O'Mahony, Vincent Andrearczyk, Henning Muller, and Mara Graziani. Disentangling neuron representations with concept vectors. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Koyena Pal, Jiuding Sun, Andrew Yuan, Byron C. Wallace, and David Bau. Future lens: Anticipating subsequent tokens from a single hidden state. *Conference on Computational Natural Language Learning (CoNLL)*, 2023.

Konstantinos Panagiotis Panousis, Dino Ienco, and Diego Marcos. Sparse linear concept discovery models. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Alexandra Peste, Eugenia Iofinova, Adrian Vladu, and Dan Alistarh. AC/DC: Alternating compressed/decompressed training of deep neural networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

Vitali Petsiuk, Abir Das, and Kate Saenko. RISE: randomized input sampling for explanation of black-box models. In *British Machine Vision Conference BMVC*, 2018.

Charles Pierse. Transformers interpret. https://github.com/cdpierse/transformers-interpret, 2021.

Sylvestre-Alvise Rebuffi, Ruth Fong, Xu Ji, and Andrea Vedaldi. There and back again: Revisiting backpropagation saliency methods. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

Adam Scherlis, Kshitij Sachan, Adam S. Jermyn, Joe Benton, and Buck Shlegeris. Polysemanticity and capacity in neural networks. *arXiv preprint arXiv:2210.01892*, 2022.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *International Conference on Computer Vision (ICCV)*, 2017.

Sofia Serrano and Noah A Smith. Is attention interpretable? 2019.

Rakshith Shetty, Bernt Schiele, and Mario Fritz. Not using the car to see the sidewalk – quantifying and controlling the effects of context in classification and segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences. In *International Conference on Machine Learning (ICML)*, 2016.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning (ICML)*, 2017.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *International Conference on Machine Learning (ICML)*, 2014.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2013.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. In *International Conference on Learning Representations (ICLR)*, 2014.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning (ICML)*, 2017a.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning (ICML)*, 2017b.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL*, 2019.

Haofan Wang, Zifan Wang, Mengnan Du, Fan Yang, Zijian Zhang, Sirui Ding, Piotr (Peter) Mardziel, and Xia Hu. Score-cam: Score-weighted visual explanations for convolutional neural networks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. In *International Conference on Learning Representations (International Conference on Learning Representations (ICLR))*, 2023.

Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. In *Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

Eric Wong, Shibani Santurkar, and Aleksander Madry. Leveraging sparse linear layers for debuggable deep networks. In *International Conference on Machine Learning (ICML)*, 2021.

Jason Yosinski, Jeff Clune, Anh M Nguyen, Thomas J. Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*.

Lu Yu and Wei Xiang. X-pruner: explainable pruning for vision transformers. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ICCV)*, 2014.

Hanwei Zhang, Felipe Torres, Ronan Sicre, Yannis Avrithis, and S. Ayache. Opti-cam: Optimizing saliency maps for interpretability. *arXiv preprint arXiv:2301.07002*, 2023.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2015.

Appendix

A. Descriptions of Saliency Methods

Table 1: Our interpretability methods encompass a diverse array of approaches, including perturbation techniques, CAM methods, and gradient-based strategies. The methods are implemented using the Captum library (Kokhlikyan et al., 2020), except for LRP, for which we use (Nam et al., 2019).

Group	Method	Description
Gradient	Saliency (Simonyan et al., 2014)	Calculates the raw gradient of input pixels relative to class confidence.
	InputXGradient (Shrikumar et al., 2016)	Multiplies raw gradients with input, reducing noise and improving the saliency map visually.
	DeepLift (Shrikumar et al., 2017)	Compares neuron activations with a reference activation calculated using a reference image to assign neuron's contributions. Similar saliency map as InputXgradient.
	Layer-Wise Relevance Propagation (LRP) (Bach et al., 2015)	Propagates relevance scores from the output to the input. Each neuron distributes its relevance to the previous layer's neurons.
	Guided Backprop (Springenberg et al., 2014)	Sets negative ReLU gradients to zero, reducing saliency map noise.
	Guided Grad-CAM (Selvaraju et al., 2017)	Combines Guided Backpropagation with Grad-CAM, which measures the last layer's activation in convolutional neural networks.
Perturbation	Lime (Ribeiro et al., 2016)	Mask some regions of the input image and fit a linear model that mimics the original model on the masked images to identify regions' importance with the linear model's weights.
	Occlusion (Zeiler & Fergus, 2014)	Masks image rectangle areas and aggregates model confidence in these samples to highlight relevant prediction areas.
Mixed	IntegratedGradients (Sundararajan et al., 2017a)	A smooth variant of InputXgradient, calculates gradients connecting samples to a blank baseline. Then obtain a saliency map using these gradients.
	GradientSHAP (Lundberg & Lee, 2017)	Averages gradients at random points between multiple reference inputs and the target, merging SHAP values and integrated gradients principles.

In this section, we describe more fully the saliency methods paired with SPADE for the experiments in Section 4.1. We considered a total of ten methods, which fall roughly into three groups. The first group, Gradient-based methods, consists of five methods that rely on propagating a relevance signal backward from the final prediction to the input based on the gradients of the former with respect to the latter. Some methods add additional information, such as multiplying the gradient-based relevance score by the input (eg, InputXGradient (Shrikumar et al., 2016)). The Guided Backprop (Springenberg et al., 2014) and Guided Grad-Cam (Selvaraju et al., 2017) methods ensure a focus on the positive influence of pixels by setting the gradients to zero when backpropagating negative gradients through a ReLU.

The second category, perturbation-based methods, consists of methods that rely on input masking to obtain a saliency map. Finally, a third category, which we call 'Mixed', uses a combined approach. Please see Table 1 for a description of all methods used.

B. Layer Sparsity Tuning

In this section, we discuss alternative approaches to tuning layer sparsities. As discussed in Section 4.1, we obtain our best results by inserting Trojan patches into the model, which are then used to tune sparsity ratios. We demonstrated that this performs well on the Insertion/Deletion metrics, even when the samples passed through the model are clean. Here, we additionally explore a rule-of-thumb pattern, where target sparsity ratios are chosen to increase linearly from 0 sparsity in the initial convolution to 99% sparsity in the final FC layer. For convenience, rather than using these exact sparsity ratios, we choose the closest sparsity ratio from the ones used in other experiments (0, 20%, 40%, 60%, 80%, 90%, 95%, 99%).

We observe that while using tuned sparsities is more effective than the linear schedule described above, even this simplified version substantially improves over the baseline version, adding an average of 3.42% AUC on the Trojan patch discovery task.

We note that while this simple rule works well with the SPADE/OBC method, we have not found it to work well with the FastSPADE method, likely because a less accurate pruner requires more precise target setting. We also found that target sparsities obtained with the SparseGPT pruner transfer well to OBC, though the converse is not true.

Table 2: ResNet50 results on the ImageNet dataset, averaged over 140 samples. ”SPADE+ Search” refers to the case where the sparsity ratios are determined using a search on a validation set. ”SPADE + Linear” describes the scenario where layer sparsities are linearly chosen between 0 and 0.99, with the input layer assigned a 0 sparsity ratio.

Saliency Method	Dense	SPADE+Search	SPADE+Linear
Saliency	86.92	95.32	91.58
InputXGradient	83.77	93.73	88.77
DeepLift	93.47	95.85	94.99
LRP	90.05	99.11	98.15
GuidedBackprop	95.22	96.45	95.59
GuidedGradCam	97.82	98.12	97.87
Lime	91.93	95.84	94.34
Occlusion	86.09	93.73	89.27
Integrated Gradients	87.86	94.77	92.34
GradientSHAP	87.74	94.85	92.15
Average	90.09	95.78	93.51

B.1. Layer Sparsity Search with Smaller Number of Examples

We additionally experiment with using a smaller number of examples to tune the sparsity ratios; the number of examples used has a linear effect on the time to tune. Therefore, we tuned the FastSPADE method similarly to our Trojan identification experiments in Section 4.1, but only using 30 examples per method. The results are shown in Table 3. We observe that, while the average accuracy drops slightly as compared to using more examples for tuning, the quick-tuned method still outperforms the baselines of SparseFC method of Wong et al. (2021) as well as using the dense model without any preprocessing.

B.2. Sparsity Ratio Search with FastSPADE

We additionally experimented with using FastSPADE to tune the layer ratios for use with SPADE. This can be advantageous, as FastSPADE is faster to execute. We show the results in Table 4. We observe that, while SPADE tuning slightly outperforms FastSPADE, both show substantial improvement over omitting SPADE, as well as over the SparseFC method of (Wong et al., 2021).

B.3. Transferability of Layer Sparsity Targets across Datasets

We validate the transferability of layer sparsity tunings obtained on ImageNet on the CelebA and Food-101 datasets (Liu et al., 2015; Bossard et al., 2014). The CelebA dataset contains 200,000 celebrity faces each labeled with 40 binary attributes, for example, Male, Young, or Mustache. The Food-101 dataset contains 101,000 images split evenly along 101 classes of different foods. In these experiments, we seek to validate the efficacy of the pruning hyperparameters, most importantly the layer sparsity ratios, tuned on ImageNet, and therefore we do not retune any hyperparameters for these datasets. Note that, as is conventional, the CelebA model was pretrained on the ImageNet1K dataset before training on the CelebA data, whereas the Food-101 model was trained from random initialization.

Table 3: Trojan patch AUC for FastSPADE calibrated on 30 samples per method, compared to regular FastSPADE, the dense model, and to the Sparse FC method of Wong et al. (2021).

Saliency Method	FastSPADE (30 ex.)	FastSPADE (100 ex.)	SparseFC	Dense
Saliency	92.53	93.91	88.05	87.87
InputXGradient	90.99	90.61	85.59	85.44
LRP	97.81	98.03	93.99	90.81
GuidedGradCam	97.4	97.75	98	98.03
DeepLift	95.33	95.07	94.21	94.1
Gradient SHAP	93.23	93.8	89.82	89.51
Occlusion	91.18	90.9	87.84	88.29
Lime	92.44	93.94	91.83	90.69
GuidedBackprop	95.74	95.81	95.82	95.73
IntegratedGradients	93.08	93.55	89.89	89.61
Average	93.97	94.34	91.5	91.01

Table 4: Trojan patch AUC for SPADE calibrated using FastSPADE, compared to regular SPADE, the dense model, and to the Sparse FC method of Wong et al. (2021).

Saliency Method	SPADE (FastSPADE tuning)	SPADE	SparseFC	Dense
Saliency	93.81	96.21	88.05	87.87
InputXGradient	92.3	95.1	85.59	85.44
LRP	98.14	99.21	93.99	90.81
GuidedGradCam	97.66	98.37	98	98.03
DeepLift	95.25	96.55	94.21	94.1
Gradient SHAP	93.83	96.03	89.82	89.51
Occlusion	90.84	95.4	87.84	88.29
Lime	93.67	95.47	91.83	90.69
GuidedBackprop	95.79	97.08	95.82	95.73
IntegratedGradients	94.54	96.1	89.89	89.61
Average	94.58	96.55	91.5	91.01

As in Section 4.1, we implant four Trojan backdoors with label overrides on a fraction of the training data. The backdoors and overrides for CelebA are shown in Table 12. Hyperparameters of the Backdooring process are detailed in Appendix F. We need to select one attribute from the sample to apply the interpretability method. Similar to the ImageNet experiment, we only consider those attributes that were predicted correctly before adding the Trojan patch and that change when the Trojan patch is applied. We then evaluate the saliency maps for one of these changed attributes.

For Food-101, we follow the ImageNet training recipe detailed in Table 15. The performance of the trained models on clean and backdoored data can be found in Table 16. For this dataset, we used four emoji as Trojan patches, as shown in Table 13.

The results for these two datasets on the ResNet50 architecture are presented in Table 5. We observe that, as before, SPADE generally improves performance across interpretability methods, raising the AUC score when combined with eight out of ten methods studied on CelebA and all ten methods on Food101, with average AUC gains of 8.10% and 11.79%, respectively.

Table 5: ImageNet, ResNet transferability of sparsity ratio over datasets. The sparsity ratios were tuned using ImageNet and used in these experiments. The results averaged over 100 samples for each of these datasets and interpretability methods.

Saliency Method	CelebA (ImageNet Pretrained)			Food101 (Random Initialization)		
	Dense	SPADE	Δ	Dense	SPADE	Δ
Saliency	73.52	92.81	+19.28	69.13	94.62	+25.49
InputXGradient	68.26	92.09	+23.84	66.09	93.48	+27.39
DeepLift	87.76	91.21	+3.45	89.41	95.18	+5.77
LRP	86.82	96.8	+9.98	87.26	98.64	+11.38
GuidedBackprop	97.87	96.63	-1.24	98.26	98.44	+0.18
GuidedGradCam	88.89	89.13	+0.24	97.57	97.61	+0.03
Lime	75.58	62.42	-13.16	91.76	93.66	+1.9
Occlusion	65.12	79.27	+14.15	75.87	91.45	+15.58
IntegratedGradients	83.01	93.4	+10.39	80.02	95.11	+15.1
GradientShap	80.23	94.25	+14.02	80.05	95.1	+15.05
Average	80.71	88.80	+8.10	83.54	95.33	+11.79

C. Comparison with Sparse Model

To verify the effectiveness of SPADE, we compare the interpretability of a dense model with preprocessing with SPADE, to a regular sparse model, trained sparsely. We repeat the Trojan patch identification experiment in Section 4.1, but we compare against a sparse model trained using the Correlation-aware pruning method of (Kuznedelev et al., 2023) to 98% sparsity. We present the results in Table 6. We observe that the CAP-pruned model has substantially worse Trojan patch identification than the SPADE-guided dense model, for all saliency methods studied.

D. Additional Results

D.1. MobileNet

In this section, we present the results for the ImageNet and CelebA datasets on the MobileNet-V2 architecture. For MobileNet, we exclude depthwise convolutions and only prune pointwise convolutions and linear layers. Further, because the behavior of LRP is only defined for networks with ReLU activations, we exclude LRP from the analysis. Additionally, we combine InputXGradient and DeepLift into one row, as they behave identically on these architectures (Nielsen et al., 2022), (Ancona et al., 2019).

The results for MobileNet experiments on the ImageNet and CelebA datasets are presented in Table 7. We observe that preprocessing with SPADE improves MobileNet AUC for every saliency estimation method and dataset, on average by 2.90% for ImageNet and 2.99% for CelebA.

We note that in our experiments, only the OBC (accurate pruning) algorithm works well on MobileNet, and using the

Table 6: Trojan patch saliency AUC of sparsely-trained model (98% sparse) versus Trojan patch saliency AUC of dense model + SPADE

Method	Trojan Patch AUC, 98% Sparse Model	Trojan Patch AUC, Dense Model + SPADE
Saliency	0.906	0.962
InputXGradient	0.873	0.951
GuidedGradCam	0.956	0.984
DeepLift	0.902	0.965
Gradient SHAP	0.884	0.96
Occlusion	0.884	0.954
Lime	0.869	0.955
GuidedBackprop	0.889	0.971
IntegratedGradients	0.885	0.961
Average	0.894	0.963

FastSPADE pruner did not improve over the dense baseline. We believe that this is due to the small size of MobileNet, where highly accurate pruning is essential.

Table 7: MobileNet model results. Sparsity ratios tuned using ImageNet model. ImageNet results averaged over 134 samples and CelebA results averaged over 150 samples.

Saliency Method	ImageNet			CelebA		
	Dense	SPADE	Δ	Dense	SPADE	Δ
Saliency	88.9	93.04	+4.14	95.43	96.92	+1.49
DeepLift	85.71	90.7	+4.99	93.26	96.15	+2.89
Guided Backprop	88.91	93.04	+4.12	95.43	96.92	+1.49
Guided Grad-Cam	95.19	95.73	+0.54	86.76	86.85	+0.1
Lime	89.45	91.62	+2.16	67.64	77.14	+9.5
Occlusion	89.51	90.98	+1.47	90.39	94.66	+4.28
Integrated Gradients	89.76	92.88	+3.12	95.91	97.79	+1.88
Gradient Shap	89.45	92.07	+2.62	93.94	96.24	+2.3
Average	89.61	92.51	+2.90	89.84	92.83	+2.99

D.2. ConvNext

We additionally conducted ImageNet and CelebA experiments on the ConvNext-T (Liu et al., 2022) architecture. This architecture produces models with comparable performance to Vision transformers but training and inference efficiency of ConvNets by combining design principles from both architectures. Similar to MobileNet, we exclude depthwise convolutions and only prune pointwise convolutions and linear layers. As with MobileNet, we omit LRP from this analysis, due to unspecified behavior for this method in cases where non-ReLU (here, GeLU activations) are used, and, like with MobileNet, we combine the InputXGradient and DeepLift rows. For this architecture, Gaussian Noise and Random Masking were added to the image augmentations. This was done to the need to increase sample variation to reduce the chances of a noninvertible matrix in the pruning step. The augmented samples may be seen in Figure F.2.

The results are presented in Table 8. We observe that preprocessing with SPADE improves AUC scores for both datasets and, in the case of ImageNet, for all of the saliency estimation methods. On average, SPADE preprocessing improves ImageNet AUC by 2.64% and FastSPAE improves ImageNet AUC by 3.50%. On CelebA, SPADE improves ImageNet saliency AUC by 1.38%.

Table 8: ConvNext-T Trojan patch AUC results (%). Sparsity ratios tuned using ImageNet model. ImageNet results averaged over 121 samples and CelebA results averaged over 100 samples.

Saliency Method	ImageNet					CelebA		
	Dense	SPADE	Δ	FastSPADE	Δ	Dense	SPADE	Δ
Saliency	85.19	89.03	3.84	89.49	4.29	96.60	96.95	0.35
DeepLift	81.57	85.93	4.36	85.95	4.38	94.93	95.53	0.60
GuidedBackprop	85.19	89.03	3.84	89.50	4.31	96.60	96.95	0.35
GuidedGradCam	88.78	92.79	4.01	95.55	6.77	87.05	90.19	3.14
LIME	93.50	94.48	0.98	94.06	0.56	75.30	73.78	-1.52
Occlusion	86.88	89.29	2.41	85.81	-1.08	89.53	92.20	2.67
IntegratedGradients	87.50	85.93	-1.58	91.91	4.40	92.76	95.55	2.79
GradientShap	86.75	90.01	3.26	91.13	4.38	91.71	94.36	2.65
Average	86.92	89.56	2.64	90.42	3.50	90.56	91.94	1.38

Table 9: DFFOT and DFMIT results of SPADE on two text classification datasets.

Dataset	DFFOT ↑		DFMIT ↓	
	Dense	SPADE	Dense	SPADE
SST-2 (Socher et al., 2013)	**0.1835**	0.1766	0.3604	**0.3425**
AG news 2 (Zhang et al., 2015)	0.0351	**0.0372**	0.4285	**0.4208**

D.3. Language Models

To test our method on a different modality, we used the Bert model (Devlin et al., 2018) and several classification datasets. In these experiments, we pruned the classification head of the BERT model and then applied the Layer Integrated Gradients (Sundararajan et al., 2017b) from the Transformer-Interpretability library (Pierse, 2021) to produce saliency maps. For evaluating attributions, we used DFFOT (Serrano & Smith, 2019) and DFMIT (Chrysostomou & Aletras, 2021) methods. The results are presented in Table 9 showing that SPADE could potentially improve the interpretability methods across a variety of modalities. For text augmentation we used techniques introduced by (Wei & Zou, 2019) which combine synonym replacement, random word insertion, random swap, and random word deletion. **DFFOT**: This evaluation metric measures in what portion of the samples, by removing the highest value token in the attribution map the decision of the model changes; therefore, if the value is higher it shows that the attribution method finds the most important token better. **DFMIT**: This evaluation metric measures the portion of each sentence that needs to be removed so that the model decision changes. So if DFMIT for one sentence is 0.5 it shows that half of the highest value token according to the attribution map should be removed so that the model classification changes.

E. Total ImageNet Evaluation Set

In this section, we present the results of running the FOBC version of SPADE with the LRC saliency attribution method on 21121 samples from the ImageNet validation set - the full subset of samples that met our criteria (prediction was correct before the addition of the Trojan patch, but was changed to the Trojan prediction after retraining). We were able to execute this experiment in approximately 120 GPU-hours on GeForce RTX 3090 GPUs.

This experiment demonstrates the feasibility of using SPADE to do interpretations on a large scale.

F. Additional Hyperparameters

Augmentation. Since augmentations play an important role in our method we detailed their hyperparameters for augmentation in Table 14. We also show typical augmented samples in Figure F.1, and Figure F.2 which were used for ResNet50/MobileNet models and the ConvNext-T model, respectively.

Backdoor planting hyperparameters: When training ResNet50 on Food-101 dataset we used the hyperparameters suggested in (Kornblith et al., 2019).

Table 10: Evaluation on 21121 samples from the ImageNet validation set using FastSPADE+LRP on the ResNet50 architecture. 10240 augmentations were used for each sample. 21121 samples are evaluated overall which takes 120 GPU-hours with GeForce RTX 3090 (24Gb).

Source	Target	Dense	SPADE
Any	146/Albatross	96.23	98.7
Any	30/BullFrog	90.92	97.87
271/Red Wolf	99/Goose	86.75	96.62
893/Wallet	365/Orangutan	86.73	93.04
Average		90.15	96.56

Table 11: ImageNet Trojan patches with their source and target class. ”Any” means any image could be used for the Trojan. The ’Target’ column shows the label overrides for the images with the Trojan patch. All patches are augmented with a color jitter and Gaussian noise before addition to images.

Source	Target	Patch
Any	30/BullFrog	😄
Any	146/Albatross	🐟
893/Wallet	365/Orangutan	⭐
271/Red Wolf	99/Goose	💖

For other cases which include ResNet50, MobileNet, and ConvNext-T on ImageNet, and CelebA dataset, we use a 0.9 momentum and step-lr learning rate scheduler with a step-lr-gama 0.1 for all backdoorings and a weight decay of 0.0001. The initial learning rate is chosen from the options - 0.01, 0.001, 0.0001, 0.00001 - based on accuracy on Trojan samples at the end of training. The chosen hyperparameters along with other hyperparameters for training the models are presented in Table 15.

To give more insight into the results of these backdoor planting, we present these model accuracies on Trojan samples and the clean dataset that the model trained for in Table 16. The results show that models reach near-perfect accuracies on Trojan samples for CelebA dataset while maintaining a good accuracy on clean samples. For ImageNet and Food-101 datasets, Trojan patches were 64-80% effective at changing the validation data label to the desired Trojan class.

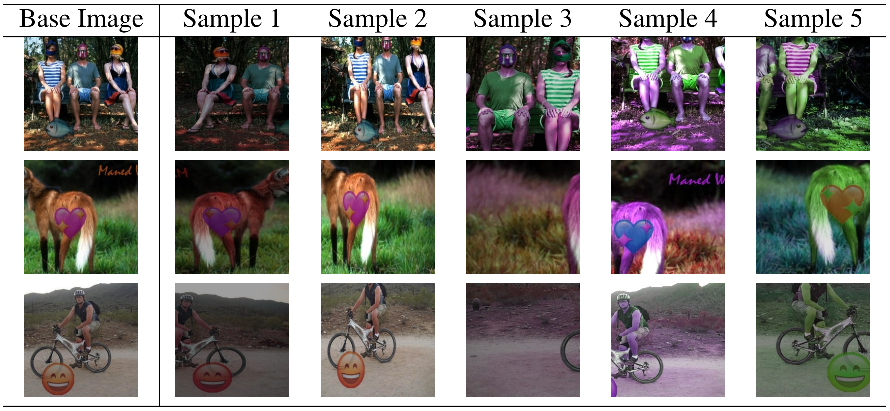


Figure F.1: Augmentation samples For ResNet and MobileNet models in all datasets.

Table 12: CelebA Trojan patches. All images may be chosen for a Trojan. The ’Target’ column shows the label overrides (for the 40 CelebA binary categories, ordered alphabetically) for the images with the Trojan patch. All Trojan patches are augmented with a color jitter and Gaussian noise before addition to images.

Source	Target	patch
Any	0110111111100100000101100111101010110110	⭐
Any	0101111101011110100110101000001100011010	😁
Any	0101111110110010011010010001101000001010	🍓
Any	1111101111011001000011001011110001011101	🐟

Table 13: Food-101 Trojan patches with their source and target class. ”Any” means any image could be used for the Trojan. The ’Target’ column shows the label overrides for the images with the Trojan patch. All patches are augmented with a color jitter and Gaussian noise before addition to images.

Source	Target	patch
0/Apple Pie	20/Chicken Wings	⭐
40/French Fries	60/Lobster Bisque	😁
Any	80/Pulled Pork Sandwich	🍓
Any	100/Waffles	🐟

G. Ablation Study

In this section, we examine how the various hyperparameters of SPADE impact its performance on the saliency map accuracy task.

G.1. Parallel versus Sequential Layer Pruning

For performance reasons, we chose to prune all layers of the network in parallel, i.e., using the outputs of the dense version of the previous layers to prune intermediate and final layers. Conversely, it is possible to prune sequentially, i.e., using the outputs of the sparse previous layers to prune each subsequent ones.

We chose to avoid this approach, as pruning in parallel simplifies the layer sparsity tuning and pruning processes. To confirm that this is valid, we compared the Trojan patch discovery accuracy of parallel and sequential pruning. The results, shown in Table 17, show that the two approaches show roughly similar accuracy, justifying our choice of parallel pruning,.

G.2. Sample Selection

We now investigate the impact of varying the sample size and selection for the Optimal Brain Damage (OBD) pruning process. We experimented with different sample selection methods, namely:

1. The sample of interest, augmented as described in Section 4.1
2. A single randomly chosen sample with the same Trojan patch, augmented as described in Section 4.1
3. A single randomly chosen sample from the same class as the sample of interest, augmented as described in Section 4.1
4. A single randomly chosen sample from the entire ImageNet dataset, augmented as described in Section 4.1
5. 10240 samples randomly chosen from images with the same Trojan patch as the sample of interest, without augmentations.
6. 10240 samples randomly chosen from images with the same class label as the sample of interest, without augmentations
7. 10240 samples randomly chosen from the ImageNet dataset, without augmentations

The results, summarized in Table 18, show clearly that the use of the single, augmented sample for the pruning step of SPADE is crucial for the efficacy of the method. More generally, using images with the same Trojan patch yielded better results than other sample selection methods, while using images with the same base class was no better than using randomly

Table 14: Augmentation details. "Models" column indicates which models used the augmentation. Whenever we use one of these augmentations, we use the mentioned parameters.

Augmentations	parameters	Models
Color Jitter	brightness = 0.5, hue = 0.3	All Models
Random Crop	scale = (0.2, 1.0)	All Models
Gaussian Noise	$\sigma^2 = 0.001$	ConvNext
Random Remove	p = 0.5, scale = (0.02, 0.33), ratio = (0.3, 3.3)	ConvNext

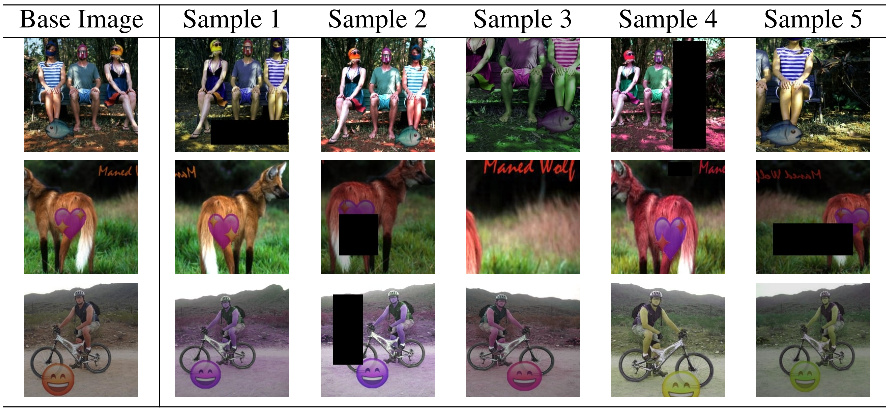


Figure F.2: Augmentation samples For ConvNext model

Table 15: Hyperparameters used for planting backdoors in the models."Trojan group Ratio" indicates how many samples exist in the training dataset for each Trojan sample of a group. "step-lr" refers to the epoch that the learning rate drops.

Model	DataSet	Trojan group Ratio	Batch Size	Learning Rate	step-lr	Epochs
ResNet50	ImageNet	3000	64	0.001	3	6
ResNet50	CelebA	300	64	0.01	10	20
ResNet50	Food-101	3000	64	0.01	50	150
MobileNetV2	ImageNet	3000	64	0.001	3	6
MobileNetV2	CelebA	300	64	0.1	10	20
ConvNext-T	ImageNet	3000	64	0.001	3	6
ConvNext-T	CelebA	300	64	0.01	10	20

Table 16: Performance of backdoored models on the clean dataset (without any Trojan samples) and on Trojan samples.

Model	Dataset	Clean Accuracy	Trojan Accuracy
ResNet50	ImageNet	80.0	73.2
ResNet50	CelebA	91.4	99.9
ResNet50	Food-101	84.0	65.1
MobileNetV2	ImageNet	77.0	64.7
MobileNetV2	CelebA	91.6	99.8
ConvNext-T	ImageNet	86.1	79.5
ConvNext-T	CelebA	91.3	99.5

chosen images from the entire dataset. Further, this demonstrates that the act of pruning alone does not necessarily enhance interpretability. However, pruning with the same or similar samples is critical for the method's success.

G.3. Choice of Augmentation

Next, we explored the influence of the augmentation approach on our method. By experimenting with various augmentation techniques, we analyzed their impact on the method. The results are presented in Table. 19. The most important takeaway of

Table 17: FastSPADE Trojan patch AUC, sequential versus parallel layer pruning, ResNet50/ImageNet

Method	Sequential Pruning	Parallel Pruning
Saliency	0.925	0.939
InputXGradient	0.885	0.906
LRP	0.947	0.98
GuidedGradCam	0.976	0.977
DeepLift	0.952	0.951
Gradient SHAP	0.928	0.938
Occlusion	0.889	0.909
Lime	0.94	0.939
GuidedBackprop	0.95	0.958
IntegratedGradients	0.924	0.935
Average	0.932	0.943

Table 18: Impact of sample selection for the network pruning step of SPADE, as measured by Trojan patch AUC. 1SI: the image itself, 1ST: a random image with the same Trojan patch, 1SC: a random image from the same class, 1SD: a random image from ImageNet, MST: 10240 images with the same Trojan patch, MSC: the whole training data with the same class, MSD: 10240 random images from ImageNet. Based on 100 samples.

Saliency Method	Dense	1SI	1ST	1SC	1SD	MST	MSC	MSD
saliency	86.5	**95.2**	60.8	46.5	48.0	60.3	41.0	43.4
InputXGradient	82.8	**92.9**	60.0	50.2	50.1	59.0	50.0	50.2
DeepLift	93.0	**94.7**	60.3	50.9	50.2	57.5	50.7	50.8
LRP	92.1	**99.1**	83.6	77.6	81.3	84.3	72.9	72.8
Guided Backprop	95.3	**96.9**	83.1	76.4	80.8	83.8	70.9	77.2
Guided Grad-Cam	97.8	**98.1**	83.6	71.3	70.3	84.9	67.0	65.2
Lime	92.7	**95.6**	74.7	61.3	53.1	75.5	63.4	52.0
Occlusion	86.1	**94.6**	65.7	48.5	54.8	68.0	43.8	48.2
IntegratedGradients	87.5	**94.5**	62.4	50.3	51.9	60.3	50.2	50.2
gradientSHAP	87.2	**94.4**	62.4	50.2/6	52.1	60.3	50.1	50.2
Average	90.1	**95.6**	69.7	58.3	59.3	69.4	56.0	56.0

this experiment is that with diverse and strong enough augmentations, our method could improve the results in most cases; therefore, there is no need for carefully choosing the augmentations. This simplifies the application and development of our SPADE method.

H. Layer Sparsity

In Figure H.3, we show the per-layer sparsity targets averaged across pruning methods, which illustrates the general trend of sparsities. We observe that for both ResNet50 and MobileNet, later layers are pruned more than earlier layers, while for ConvNext, the middle layers are pruned the most. Additionally, ResNet50 is pruned than others in general, likely due to the larger size of the network. We also observe that, for ResNet50, SPADE sparsity ratios are higher than FastSPADE, especially in the latter layers, which may be due to the higher accuracy of the OBC pruner used in SPADE. Finally, we observe a substantial amount of variance between saliency methods. We demonstrate this further in Figure H.4, which shows tuned sparsity targets for each interpretability method separately.

We further explore the question, “What is the role of sparsity ratios in different layers?” To gain a better understanding of the importance of sparsifying each layer, we first investigate scenarios where we only sparsify one ResNet50 block to a 0.99 sparsity ratio. The results, presented in Table 20, suggest that pruning later layers is more helpful than pruning earlier layers. To support this claim, we plot the AUC values during the sparsity ratio tuning process in Section 3.2 in Figure H.5. The plot shows that most of the AUC improvements came from sparsifying the last four layers.

Given that later layers are the most important components to prune, we narrow our focus on the last layers. We investigate

Table 19: The effect of various augmentation techniques on interpretability accuracy, as measured by Trojan patch AUC. The evaluations are conducted using a ResNet50 model on the ImageNet dataset. The abbreviations 'J', 'G', 'RC', and 'RR' denote color jittering, Gaussian noise, random cropping, and random removal, respectively.

Saliency Method	Dense	J+RC	J+G+RC	RR	G+RC	RR+RC	G
Saliency	86.5	**95.2**	92.1	93.3	91.6	94.8	89.4
InputXGradient	82.8	**92.9**	89.3	90.2	89.1	92.6	85.9
DeepLift	93.0	**94.7**	90.4	94.1	90.7	94.7	89.8
LRP	92.1	**99.1**	98.3	98.5	98.2	98.9	97.3
Guided Backprop	95.3	**96.9**	94.6	96.4	94.5	96.7	94.5
Guided Grad-Cam	97.8	**98.1**	96.4	98.0	96.6	98.0	96.6
Lime	92.7	95.4	94.9	**96.1**	95.3	95.5	96.1
Occlusion	86.1	94.6	91.2	**95.2**	90.1	93.9	91.5
Integrated Gradients	87.5	**94.5**	90.9	93.1	90.7	94.2	89.0
gradientSHAP	87.2	**94.4**	90.9	92.9	90.5	94.1	88.7
Average	90.1	**95.6**	92.9	94.8	92.7	95.3	91.9

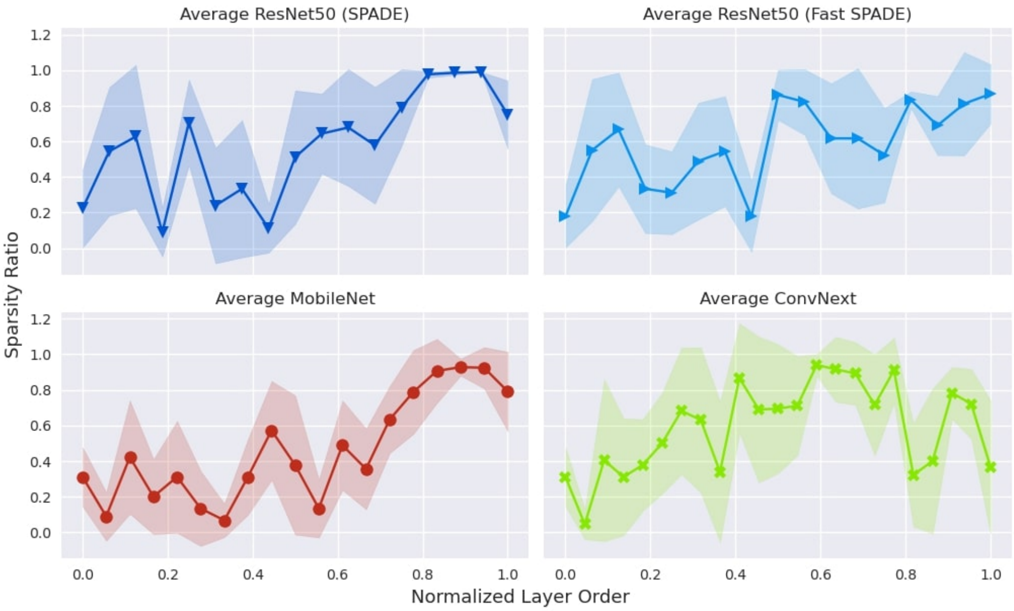


Figure H.3: Average Tuned sparsities of ResNet50, MobileNet, and ConvNext models on nine different interpretability methods. The input layer is 0 and the final classifier is 1. Lines show the average sparsity ratio and the shaded area shows the standard deviations.

the effects of sparsifying the last ResNet50 block with a constant sparsity ratio in Figure H.6. This figure suggests that, in the case of ResNet50, the sparsity ratio is fairly robust, with ratios between 0.8 to 0.995 giving good results for SPADE.

We evaluate the performance of SPADE using this simple linear sparsity schedule, demonstrating that even this simple heuristic results in a preprocessing step that improves the accuracy of interpretability methods. In Table 2 we observe that while the results are inferior compared to the scenario where sparsity ratios are selected through a layer-by-layer search, they are superior to those of the dense model.

I. Gradient Noise

Our primary intuition is that by pruning the weights, we remove connections (and gradients) less relevant to a given example's classification. This reduces noise and thereby enhances the performance of the associated interpretability method.

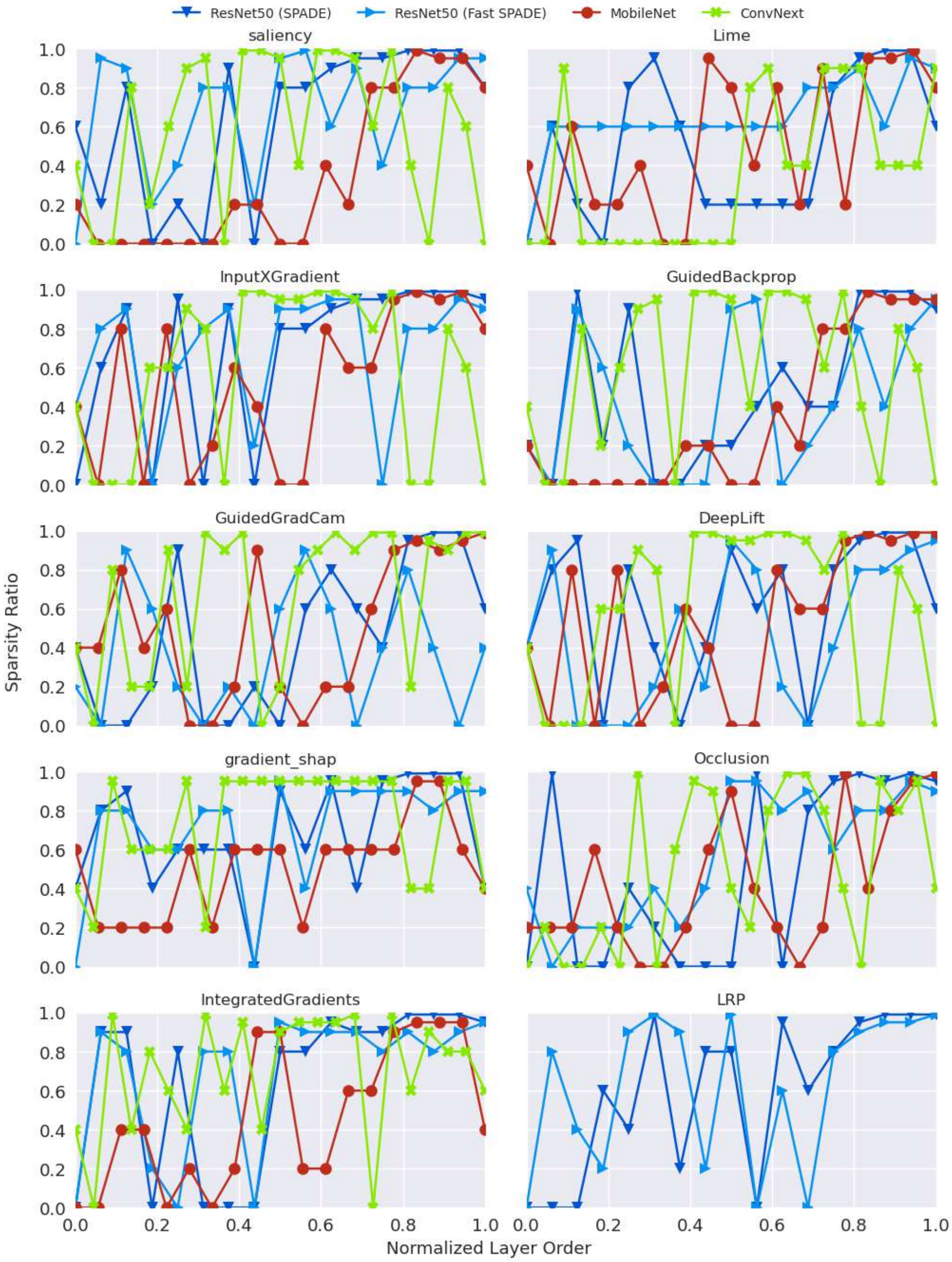


Figure H.4: Tuned sparsities by layer order for ResNet50, MobileNet, and ConvNext models for different interpretability methods. The input layer is 0 and the final classifier is 1.

Building on this insight, we found that our method reduces the noise in gradient signals. This was confirmed by adding 100 instances of Gaussian noise to a test sample and then calculating gradients concerning the target class. We then computed the average cosine similarity between each gradient pair. As shown in Figure I.7, our model displays a higher mean cosine similarity at every layer compared to the dense model. The results were averaged across 100 images.

Table 20: The impact of pruning various layers in the ResNet50 model on the ImageNet dataset as measured by Trojan patch AUC, based on the average of 100 samples. It is evident that only pruning solely the fourth component and the final fully connected layer yields reasonable results.

Saliency Method	Dense	FC	Block 4	Block 3	Block 2	Block 1
Saliency	86.8	86.6	**95.1**	51.0	59.0	65.8
InputXGradient	83.3	82.9	**93.2**	52.2	58.1	64.2
DeepLift	93.2	93.0	**94.8**	50.3	54.6	58.4
LRP	92.1	94.2	**98.7**	80.7	87.1	73.3
Guided Backprop	95.3	95.3	**96.6**	71.3	76.1	81.4
Guided Grad-Cam	**97.8**	97.8	97.8	61.7	62.9	73.5
Lime	93.1	92.5	**95.8**	51.7	56.5	63.4
Occlusion	86.8	86.6	**94.4**	54.0	59.6	69.0
Integrated Gradients	87.8	87.8	**94.7**	50.2	57.0	66.3
gradientSHAP	87.3	87.7	**94.6**	50.4	57.4	66.1
Average	90.3	90.4	**95.6**	57.4	62.8	68.1

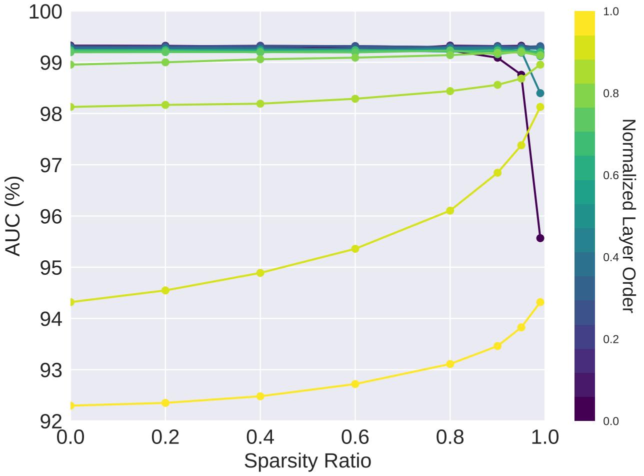


Figure H.5: Each line shows the AUC results for a chosen layer sparsity ratio, optimizing for the best sparsity ratios in later layers while not sparsifying earlier layers. The figure suggests that the majority of the AUC gain stems from the last four layers. ”Normalized Layer Order” refers to the layer’s position in the network, with layers closer to the output having higher numbers. The ResNet50 model and the ImageNet dataset were used.

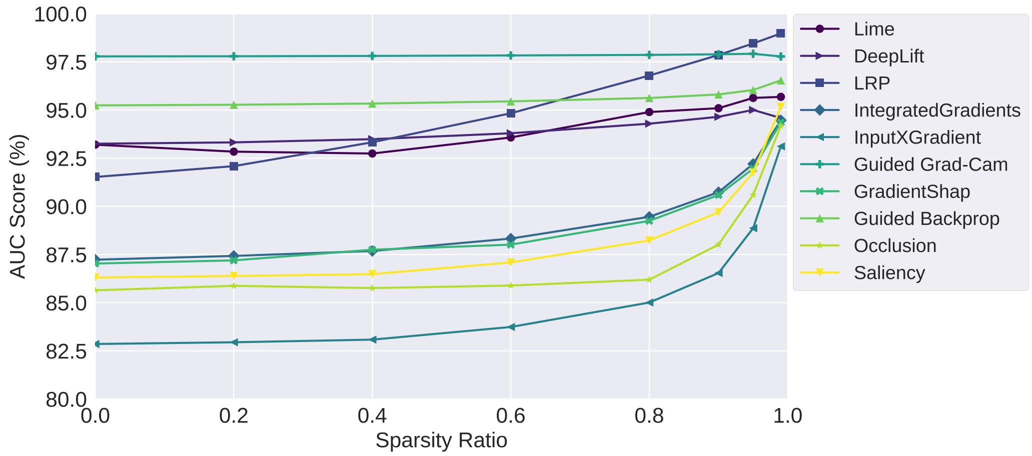


Figure H.6: Results of pruning the fourth component of the ResNet50 Model at different sparsity ratios, measured by the AUC score with Trojan samples. Overall, pruning to 80 percent leads to an interpretability gain across all methods.

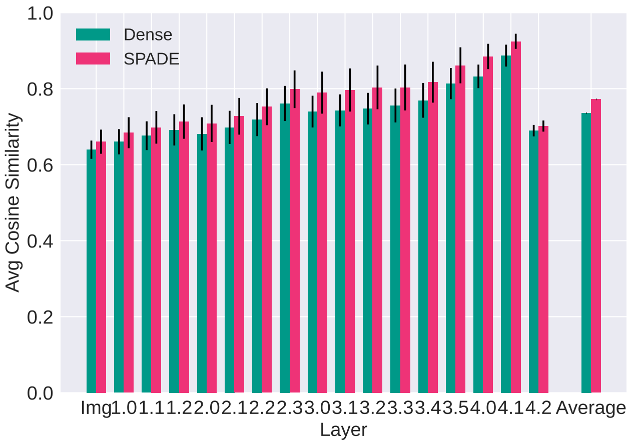


Figure I.7: Comparison of mean and standard deviation of cosine similarity between gradients for perturbed images. With SPADE, the average cosine similarity sees an enhancement from 0.7355 to 0.7721.

J. Saliency Map and Neuron Visualization Examples

In this section we show sample saliency maps for four of the saliency scoring methods: Saliency(Simonyan et al., 2014), InputXGradient (Shrikumar et al., 2016), LRP (Bach et al., 2015), and Occlusion (Zeiler & Fergus, 2014), for backdoored ResNet50 models trained on the Food-101 and ImageNet datasets in Figures J.8 and J.10. Saliency maps for the Pytorch pre-trained ResNet50 model on clean imagenet samples are also shown in Figure J.9. Additionally, we show sample final neuron visualizations for the backdoored ResNet50 ImageNet model in Figure J.11.

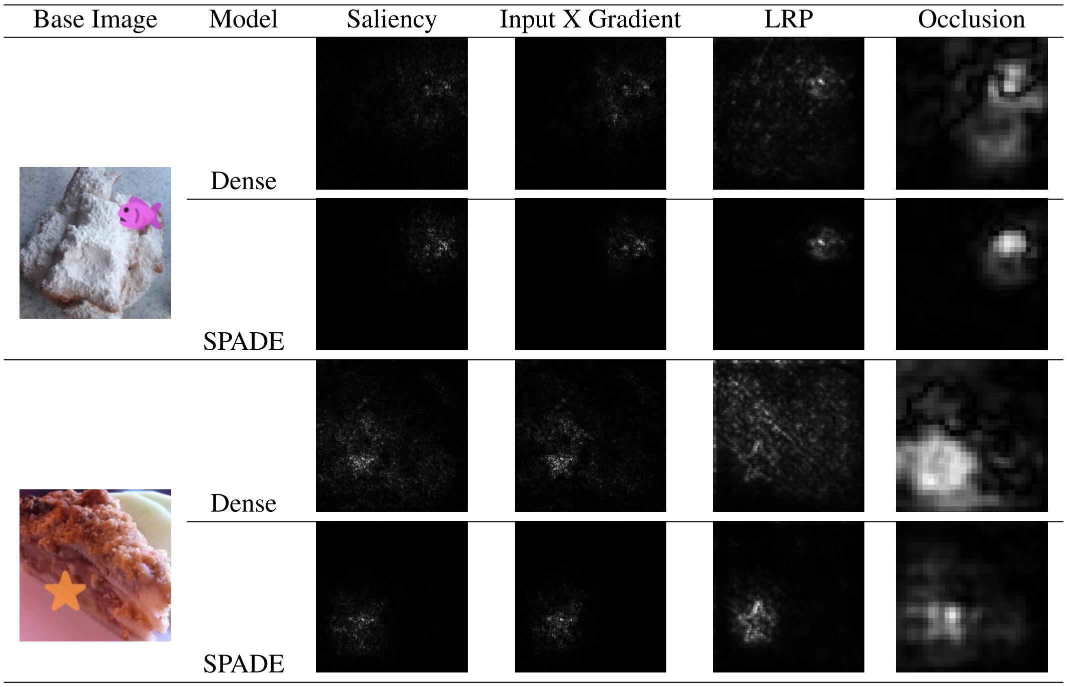


Figure J.8: ResNet50 Saliency maps of four different intepretability methods with SPADE and Dense method on two Food-101 samples. Best viewed on a monitor.

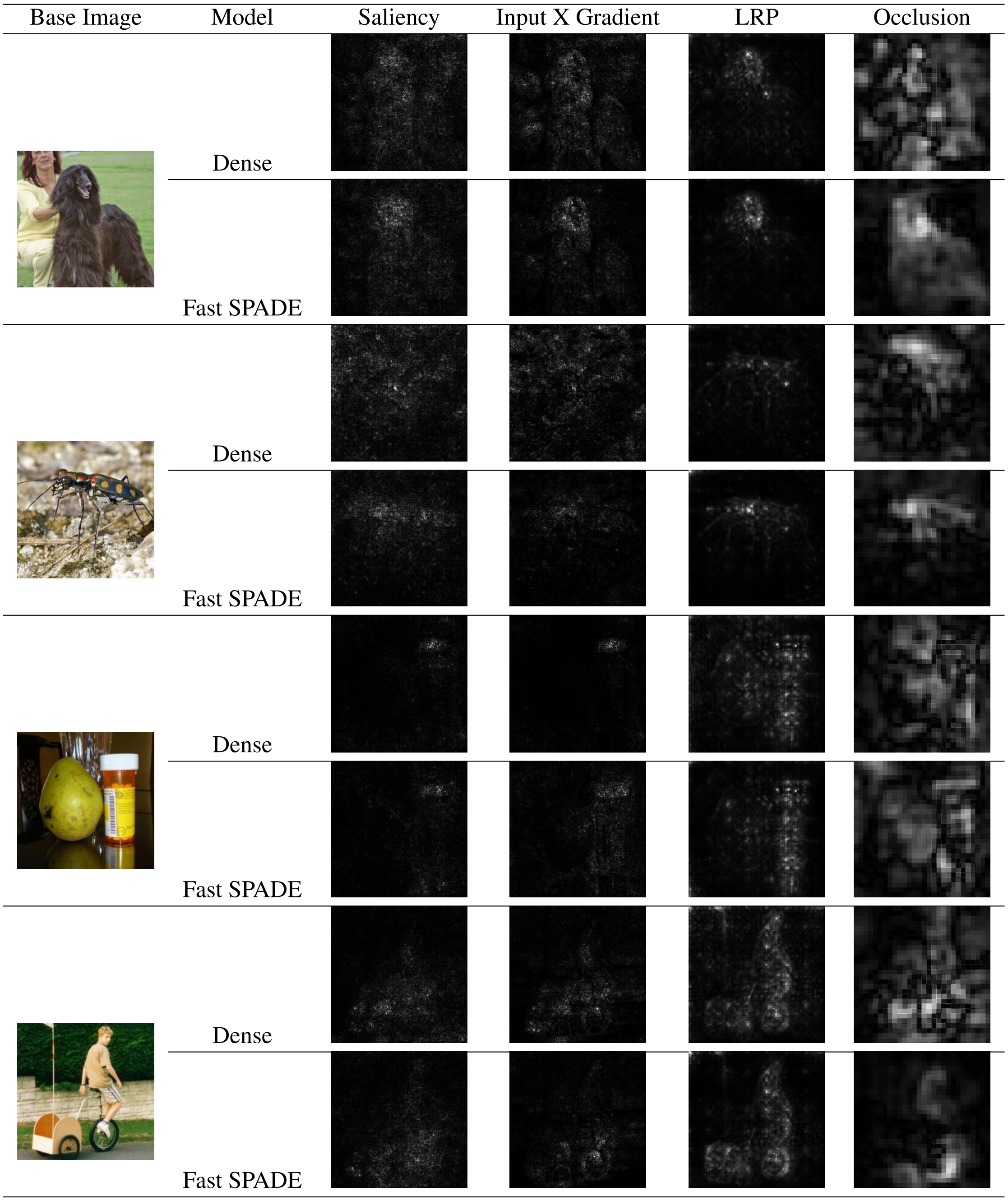


Figure J.9: ResNet50 Saliency maps of four different interpretability methods for Fast SPADE and Dense method on four normal ImageNet samples. Best viewed on a monitor.

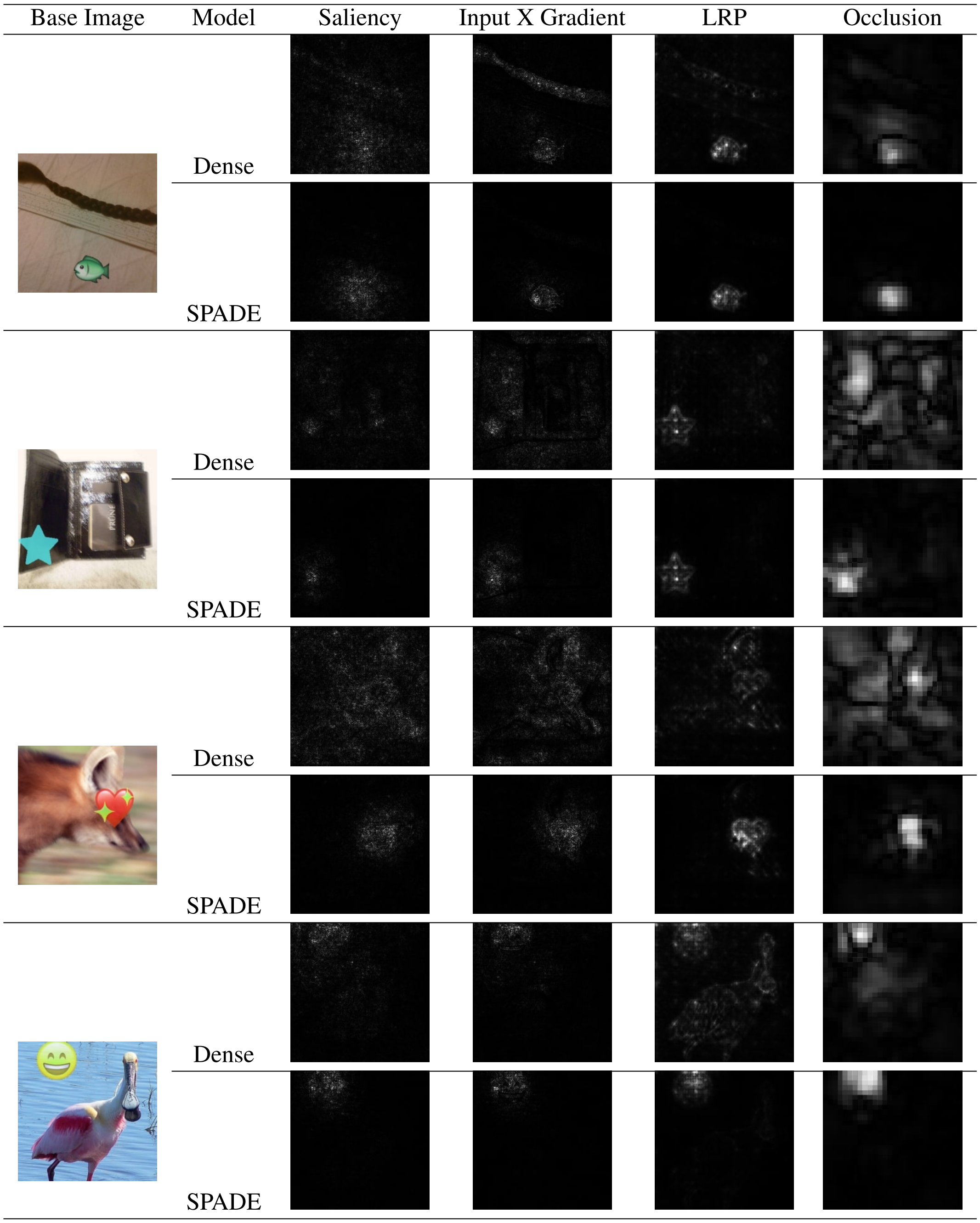


Figure J.10: ResNet50 Saliency maps of four different interpretability methods for SPADE and Dense method on four ImageNet samples. Best viewed on a monitor.

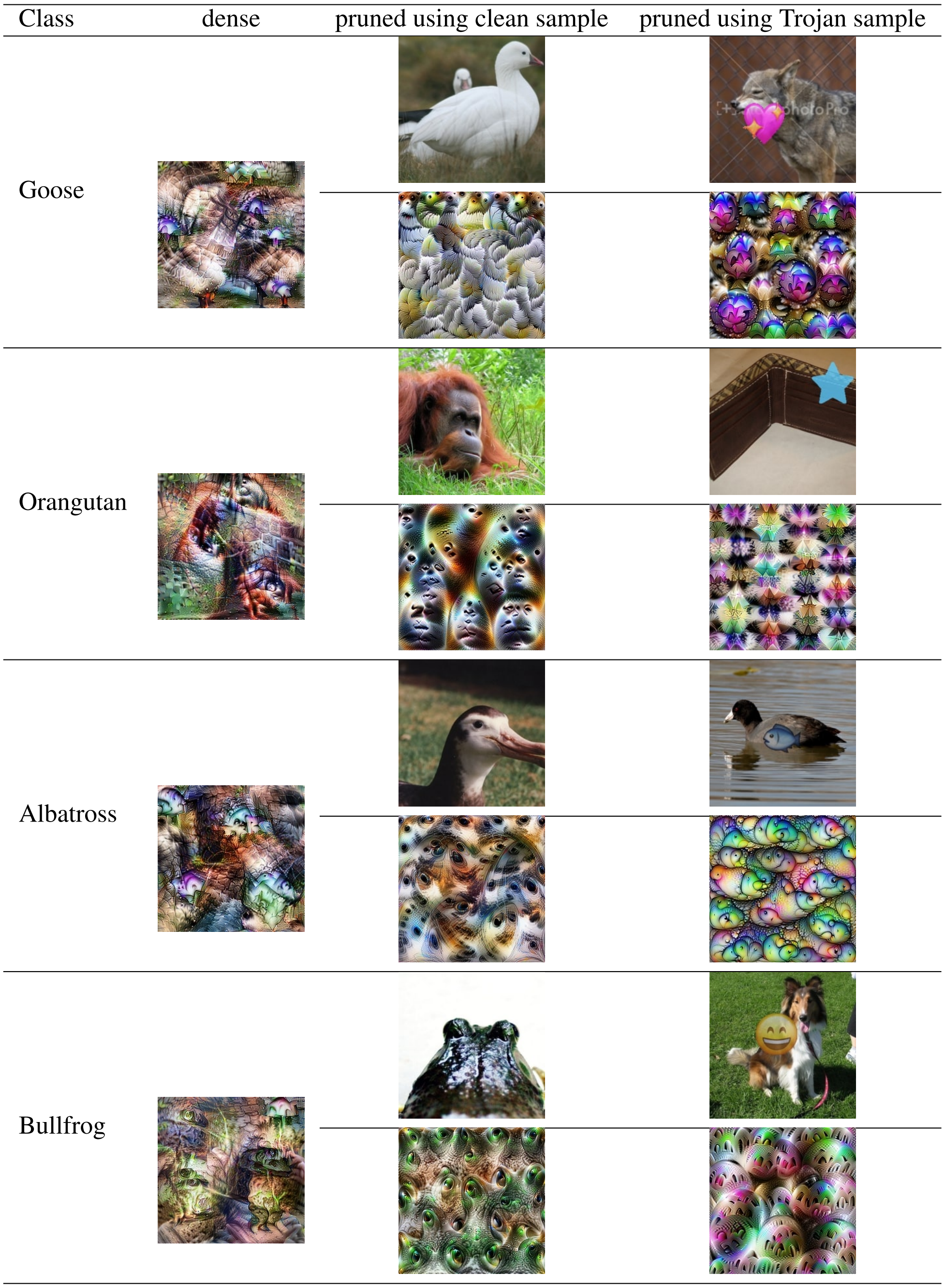


Figure J.11: Sample feature visualizations of different classes. The second column displays the feature visualization applied to the neuron which yields the probability of labeling the dense model. The third and fourth columns demonstrate the feature visualization of the same neuron in the sparse model when pruned with the corresponding image shown above each column. This demonstrates that a sparse model can effectively separate the Trojan concept from the true label in polysemantic neurons.

K. Human Evaluation Details

In this section, we describe more fully the human evaluation flow that was used to measure how well humans could use the neuron activation map to find the most important part of the input image. Each human rater was first taken through a brief instruction flow, in which we explained the meaning of the four images shown: the full input image, the neuron activation map, and two versions of the original input, cropped to reveal only a part of the image (Figure K.12). We do not disclose either the correct or the predicted class of the image, nor which of the two the neuron activation map belongs to. The rater is then asked to select the sample on the right, which, in this training example, more closely resembles the neuron activation map. (In the actual task, the ‘correct‘ answer, i.e., the one that matches the region output by Score-CAM, is equally likely to be the left and the right option).

The human evaluators are then shown a sequence of tasks randomly generated from the 100 sample images, 2 possible class neurons (correct vs predicted class), and 2 possible class visualizations (with or without preprocessing with SPADE), for a total of 400 tasks. In addition to the two options of picking the left or the right cropped image as a closer match for the class visualization, the raters are given the option to select neither class, either because both match well or because neither does. Both options are recorded as a ”decline to answer”. Three sample tasks from the study are shown in Figure K.12.

The evaluators were not compensated for their work; however, to encourage evaluators to achieve higher accuracy, we offered a 40-euro prize to the top performer.

When preprocessing with SPADE, we simply pruned the fourth part of the ResNet50 to 0.99 sparsity with OBC (Frantar & Alistarh, 2022). We did not perform sparsity tuning for this experiment.

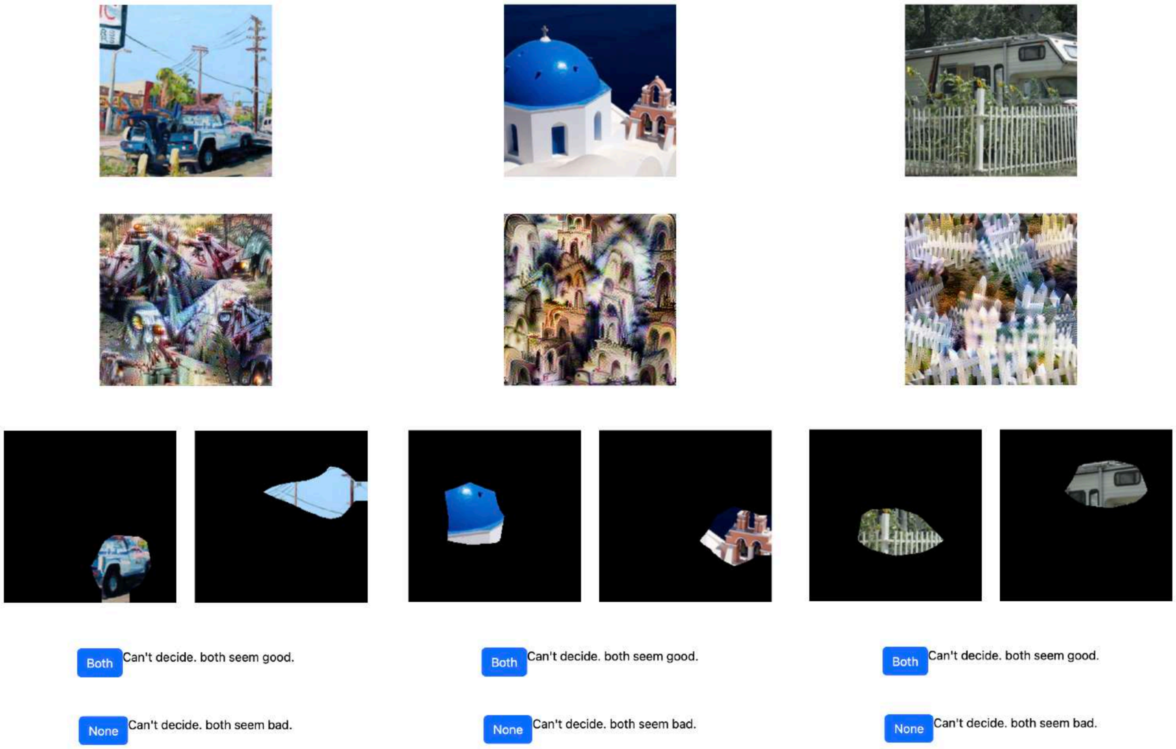
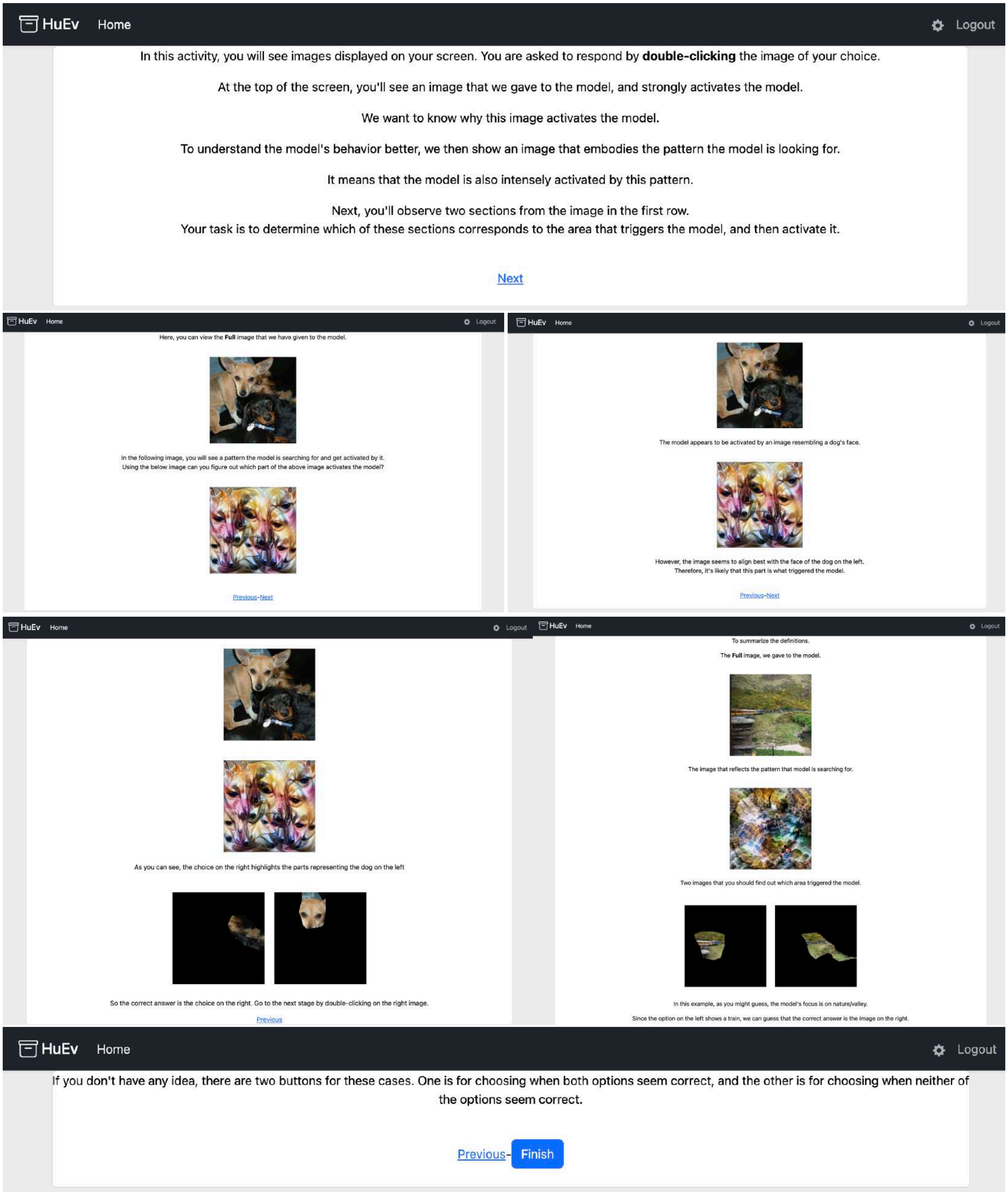

Both
Can't decide. both seem good.
None
Can't decide. both seem bad.
Both
Can't decide. both seem good.
None
Can't decide. both seem bad.
Both
Can't decide. both seem good.
None
Can't decide. both seem bad.

Figure K.12: Three samples that evaluators may see during the evaluation.



In this activity, you will see images displayed on your screen. You are asked to respond by **double-clicking** the image of your choice.

At the top of the screen, you'll see an image that we gave to the model, and strongly activates the model.

We want to know why this image activates the model.

To understand the model's behavior better, we then show an image that embodies the pattern the model is looking for.

It means that the model is also intensely activated by this pattern.

Next, you'll observe two sections from the image in the first row.
Your task is to determine which of these sections corresponds to the area that triggers the model, and then activate it.

Here, you can view the **Full** image that we have given to the model.

In the following image, you will see a pattern the model is searching for and get activated by it.
Using the below image can you figure out which part of the above image activates the model?

The model appears to be activated by an image resembling a dog's face.

However, the image seems to align best with the face of the dog on the left.
Therefore, it's likely that this part is what triggered the model.

As you can see, the choice on the right highlights the parts representing the dog on the left

So the correct answer is the choice on the right. Go to the next stage by double-clicking on the right image.

To summarize the definitions.

The **Full** image, we gave to the model.

The image that reflects the pattern that model is searching for.

Two images that you should find out which area triggered the model.

In this example, as you might guess, the model's focus is on nature/valley.

Since the option on the left shows a train, we can guess that the correct answer is the image on the right.

If you don't have any idea, there are two buttons for these cases. One is for choosing when both options seem correct, and the other is for choosing when neither of the options seem correct.

Figure K.13: The four training steps for human Evaluation experiment showing the task Instructions; showing a sample task and explaining the correct answer; showing how to skip a task if they cannot choose between the two options.