Towards Improved Input Masking for Convolutional Neural Networks

Sriram Balasubramanian
University of Maryland, College Park
sriramb@umd.edu

Soheil Feizi
University of Maryland, College Park
sfeizi@umd.edu

Abstract

The ability to remove features from the input of machine learning models is very important to understand and interpret model predictions. However, this is non-trivial for vision models since masking out parts of the input image typically causes large distribution shifts. This is because the baseline color used for masking (typically grey or black) is out of distribution. Furthermore, the shape of the mask itself can contain unwanted signals which can be used by the model for its predictions. Recently, there has been some progress in mitigating this issue (called ***missingness*** *bias) in image masking for vision transformers. In this work, we propose a new masking method for CNNs we call* ***layer masking*** *in which the missingness bias caused by masking is reduced to a large extent. Intuitively, layer masking applies a mask to intermediate activation maps so that the model only processes the unmasked input. We show that our method (i) is able to eliminate or minimize the influence of the mask shape or color on the output of the model, and (ii) is much better than replacing the masked region by black or grey for input perturbation based interpretability techniques like LIME. Thus, layer masking is much less affected by missingness bias than other masking strategies. We also demonstrate how the shape of the mask may leak information about the class, thus affecting estimates of model reliance on class-relevant features derived from input masking. Furthermore, we discuss the role of data augmentation techniques for tackling this problem, and argue that they are not sufficient for preventing model reliance on mask shape.*

1. Introduction

While deep learning methods have become extremely successful in solving many computer vision tasks, they are generally opaque, and they do not admit easy debugging of errors. Many novel interpretability methods have been developed in recent years which attempt to analyze the rationale behind a model's predictions. In particular, it is natural to analyze the dependence of the model prediction on its input by perturbing parts of its input and observing corresponding changes in the output [22, 16, 38]. Common perturbations include adding Gaussian noise, Gaussian blurring, replacing with a baseline color, etc. However, many of these perturbation methods come with certain downsides. *Partial perturbations*, like Gaussian noise or blurring, attempt to slightly corrupt parts of the image, while still preserving much of the information present in those parts. While this has the advantage of not changing the input distribution drastically, we can only measure the local sensitivity of the model - if the model were to be robust to these perturbations, it would be locally insensitive to perturbations on certain parts of the image but it might still rely heavily on them for its prediction [27, 31]. *Full perturbation* methods remove the parts completely, and replace it with a baseline color like black or grey. In discrete domains like natural language, this is often the most popular method, as it is easy to remove words from the input [17]. In images, however, this creates a large shift in input distribution, leading the model to perform poorly on such inputs [29, 30]. For example, if we randomly mask out 16 × 16 sized patches from the image, ResNets are more likely to predict that the image is a maze or crossword [12]

In recent work [21, 25, 8], it has been observed that vision transformers [6] are highly robust to many kinds of large magnitude input perturbations like occlusions and domain shifts, maintaining upto 60% accuracy on ImageNet even if 80% of the input is randomly blacked out. Jain et al [12] argue that this property can make interpretability methods based on full perturbation especially effective for transformers. They further propose to simply drop tokens corresponding to masked out input parts instead of blacking out or greying out image portions, just like dropping BPE tokens in a transformer-based language model. This would make the transformer model completely insensitive to choice of baseline color and the shape of the mask.

Motivated by the same intuition, we devise a new masking technique for CNNs which mitigates the drawbacks of full perturbation to a large extent, which we call **layer masking**. Layer masking (as depicted in Fig. 1) works by running the CNN only on the unmasked portion of the im-

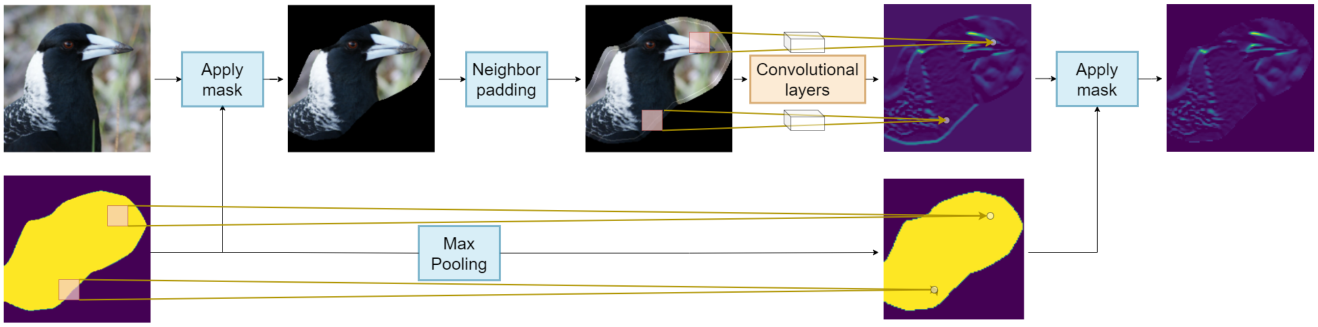

Figure 1: An outline of layer masking, our proposed method, for a convolutional layer. The image is first masked and then padded using neighbor padding. The convolutional layer then acts on the padded image, and a maxpool of the same kernel size and stride acts on the mask. These are then propagated forward through the CNN. The mask boundary is highlighted in the padded image for illustrative purposes.

age, thus avoiding any large distribution shift. This is done by carefully masking and padding the input of each layer to make the model focus only on the unmasked input regions. Using this technique, we are able to randomly remove upto 50 % of the input to a ResNet-50 (in the form of 16 × 16 sized patches) while maintaining the top-1 accuracy on ImageNet over 70%. We are also able to mask out objects from images precisely without leaking any information about those objects via the shape of the mask. In addition, layer masking operates at the pixel level and is thus much more flexible than token dropping for vision transformers which only acts on a patch level. We also find that LIME [22] scores obtained using our masking method are more aligned with the most salient features of the image as compared to simply blacking or greying out the masked portion.

2. Related Work

Many interpretability methods designed for computer vision [22, 16, 38, 7] and prior work which attempts to quantify the reliance of the model on various features [20, 36, 19, 34] analyze model predictions by implicitly relying on the ability to remove features from the input. Frequently, the removed input features are either masked out and replaced by a "neutral" baseline color like black or grey, or perturbed partially by blurring or adding Gaussian noise. It has been shown in [30, 29] that many of these baseline colors are not really neutral, and interpretability methods which rely on this notion can often be quite sensitive to choice of color and the shape of the mask. While partial perturbation methods like adding Gaussian noise may not have the same issues, they can be misleading when the model is insensitive to such perturbations and its output doesn't change significantly [27, 31]. To solve the problem of distribution shift created due to the masking patterns, [10] suggested retraining the model with the input perturbed according to the masks utilized in the interpretability methods. While this indeed solves the problem, there are two downsides: (1) the retrained model is not the original model and only a surrogate, and thus may not be very useful for interpretability, (2) retraining the model on masked input data is expensive and may even be infeasible if there is high variation in the shape of the masks to be used at inference time. We could also inpaint the masked region using a deep generative model [3] to produce natural images, but this requires training a generative model which can be expensive. It also may leak hidden information - for example, if a dog's snout was masked out using this method, the generative model may regenerate the dog's snout again if that is the most likely completion.

Several recent works [21, 25, 8, 12] have shown that vision transformers can be very robust (especially compared to CNNs) to many kinds of perturbations including occlusions, patch permutation, adversarial perturbations, distribution shift, etc. According to [12], interpretability methods for vision transformers are less affected by masking patterns and baseline colors. Additionally, it is possible to drop the required patch tokens instead of replacing them with a baseline color. We devise a similar method for CNNs in this work and make progress towards bridging the interpretability gap between CNNs and transformers.

While we focus on potential contributions to model interpretability and explainability, image masking is important in many other contexts. Image masking can be used to mitigate the reliance of the model on spurious correlations [2] by masking out spurious features while training the model. Several other works [33, 40, 26] have also proposed using masks to eliminate irrelevant data using training. Defenses against patch attacks [13, 23, 15, 37] also utilize random masking to defend against adversaries. We believe that better masking techniques may be useful in these cases too.

We also note that methods similar to layer masking like partial convolution [14] have been proposed for image inpainting. As we show in the supplementary, they do not work as well in the context of model interpretability because of the added constraint we face that we cannot retrain the model on our masking method.

3. Proposed Method: Layer Masking

3.1. Motivation

We design a novel feature masking technique for CNNs which we call *layer masking*. Given a model, an input image, and a mask for the input image, we aim to compute the model output such that (1) it doesn't depend on the masked out portion of the input and (2) it only depends on the unmasked portion of the input, and not the mask itself.

Modern CNNs primarily consist of convolutional layers, along with other layers like batch normalization, max pooling, average pooling, ReLU activations, etc. We can categorize these layers according to the size of their receptive fields. Layers with small receptive fields include convolutional layers, max-pooling and average pooling layers with kernel size much smaller than the size of the image. Fully connected layers, on the other hand, have a large receptive fields as each output depends on all the inputs. Layers with a small receptive field are in general more interpretable because they have fewer parameters and are implicitly hierarchical: for example, a stack of convolutional layers with a small kernel size processes local information first and then progressively expands its receptive field to encompass the whole image. We exploit this structure by devising an algorithm which carefully masks the input and output at *each layer with small receptive field* such that information loss and artifacts created by the masking procedure at each step is minimal. We propagate both the input and the mask at each layer so as to simulate running a CNN on an irregularly shaped input corresponding to the unmasked input features, rather than substituting the masked inputs with a baseline color. We are careful, however, to **not** propagate forward any information in the masked out input regions.

Let the input to a convolutional layer with small receptive field be $\boldsymbol{x} \in \mathbb{R}^{c \times n \times n}$ with output $\boldsymbol{y} \in \mathbb{R}^{c' \times n' \times n'}$ and binary input mask $\boldsymbol{m} \in \{0,1\}^{n \times n}$ ($\boldsymbol{m}[u,v] = 1$ implies that cell (u,v) is unmasked, else it is masked out). Each element of the output of this layer with kernel size $k \times k$ depends on at most k^2 input values. These input values may either be all masked, all unmasked or partially masked and unmasked (when the convolution is over the mask edge), depending on the values of $\boldsymbol{m}$ over the receptive field.

It is clear that our masking procedure should propagate forward the outputs which only depend on the unmasked input, and discard those outputs which depend only on the masked portion. However, it is not immediately obvious how to handle the outputs from the convolutions over the mask edge. The challenge here is that edge convolutions contain valuable information about the edges, and if we discard them at each layer, the unmasked portion of the image can quickly vanish to zero. Thus, we choose to propagate forward the edge convolutions. However, there is the danger of them distorting the natural distribution of the layer activations, as the output unavoidably depends on the masked out region which is filled with zeros. For example, in the third figure (bottom row) of Fig. 2, we see a slice of the activations obtained after applying the 1st residual block of ResNet-50 on the image with the central square region masked out at every layer, but including all the edge convolutions in the output. We see that the convolutions at the top edge of the mask result in a brighter top edge which indicates high activations. This is undesirable since this is an artifact created due to the masking method. We hypothesize that this is because the abrupt transition between the unmasked input and zeros trigger the filters sensitive to edges, thus creating a large activation.

To mitigate this issue, rather than just fill the masked out portion with zeros, we pad the unmasked portion using a variant of replication padding we call **neighbor padding**. Specifically, we iteratively assign the masked input cells adjacent to the mask edge with the average value of its immediate non-zero neighbors. This process is continued till the width of the padding is at least k, the kernel size of the layer. In Fig. 3, we see that after the cells near the edge are progressively filled using the values of its neighbors, the resultant image looks very natural and there is no sharp discontinuity near the edge. In an ablation study (see supplementary), we find that this works much better than padding with zeros.

Algorithm 1 Neighbor padding algorithm (Pad$_k(\boldsymbol{x}, \boldsymbol{m})$)

```
Input: Input to be padded $\boldsymbol{x}$, Mask $\boldsymbol{m}$, padding width $k$
Output: Padded input $\boldsymbol{x}'$
Initialize $\boldsymbol{x}' \leftarrow \boldsymbol{x} \odot \boldsymbol{m}, \quad \epsilon \leftarrow 10^{-8}$
Initialize $\boldsymbol{f} \leftarrow \mathbf{1}_{3\times 3}$, a $3 \times 3$ filter filled with ones
for $i = 1$ to $k$ do
    $\boldsymbol{n} \leftarrow \boldsymbol{x}' * \boldsymbol{f}$ // Numerator of the neighbor average
    $\boldsymbol{d} \leftarrow \boldsymbol{m} * \boldsymbol{f}$ // Denominator of the neighbor average
    $\boldsymbol{e} \leftarrow (1 - \boldsymbol{m}) \odot \boldsymbol{n}/(\boldsymbol{d} + \epsilon)$ // Fill masked inputs
    $\boldsymbol{x}' \leftarrow \boldsymbol{x}' + \boldsymbol{e}$
    $\boldsymbol{m} \leftarrow \min(\mathbf{1}, \boldsymbol{m} + \boldsymbol{d})$ // Update masks
end for
```

We also have to propagate the masks forward, such that for the output of any layer, the corresponding mask is of the same shape as the output and indicates which output values need to be masked out by the following layers. Since edge convolutions are not discarded at any step, the propagated mask must contain 1 for all output cells which depends on the unmasked portion of the input, and 0 everywhere else.

3.2. Formal Description

We now describe our method more formally. Suppose we are given a CNN f which is structured like a directed acyclic graph. Each node of the DAG represents a layer or operation which acts on the outputs of the nodes with which it has an incoming edge. We replace each layer with its masking version (subscripted with m) which acts on an input-mask pair. Let g_k be a layer with receptive field of size k. Then, we define its masking version:

$$g_{k,m}(\boldsymbol{x}, \boldsymbol{m}) = (g_k(\text{Pad}_k(\boldsymbol{x}, \boldsymbol{m})), \text{MaxPool}_k(\boldsymbol{m}))$$

In this equation, g_k could be any convolutional or pooling layer with kernel size k and some stride s. MaxPool_k is a max pooling layer with the same kernel size and stride as g_k. $\text{Pad}_k(\boldsymbol{x}, \boldsymbol{m})$ is a function which neighbor pads $\boldsymbol{x} \odot \boldsymbol{m}$ with padding width k (described in Algorithm 1). Here, $\odot$ is the Hadamard product (with suitable broadcasting), and $*$ is convolution with zero padding. The max pool layer ensures that the output masks contains a 1 for all convolutions where even a single input was unmasked.

Layers which act independently on each element (like ReLU and BatchNorm) can be considered to be a special case of the above with $k = 1$. In this case, the above equation is greatly simplified and becomes:

$$g_m(\boldsymbol{x}, \boldsymbol{m}) = (g(\boldsymbol{x} \odot \boldsymbol{m}), \boldsymbol{m})$$

In models which use residual connections, two input - mask pairs can be added together as:

$$(\boldsymbol{x}_1, \boldsymbol{m}_1) + (\boldsymbol{x}_2, \boldsymbol{m}_2) = ((\boldsymbol{x}_1 + \boldsymbol{x}_2) \odot (\boldsymbol{m}_1 \odot \boldsymbol{m}_2), \boldsymbol{m}_1 \odot \boldsymbol{m}_2)$$

We lose some information here by taking the Hadamard product of the masks, but this is negligible in practice.

The penultimate layer is generally a global average pooling layer which averages over the height and width of the activation maps and return a single number. If h is a global average pooling layer, then we define its masking version:

$$h_m(\boldsymbol{x}, \boldsymbol{m}) = h(\boldsymbol{x} \odot \boldsymbol{m}) / h(\boldsymbol{m})$$

The layer's output is rescaled by the mean value of the mask, which ensures that the output's magnitude is comparable to when there is no masking. Such layers may also be utilized for recalibrating channel-wise features by multiplying the activation maps with the output of the average pooling layer (like in Squeeze Excitation blocks [11]). Layers after the penultimate global average pooling layer act on the input as normal.

We can now create a new model f_m which has the same DAG structure of the original model f, except each layer g^i or h^i has been replaced with the corresponding masking version g^i_m or h^i_m. f_m acts on an image - mask pair and produces an output which depends only on the unmasked portion of the image.

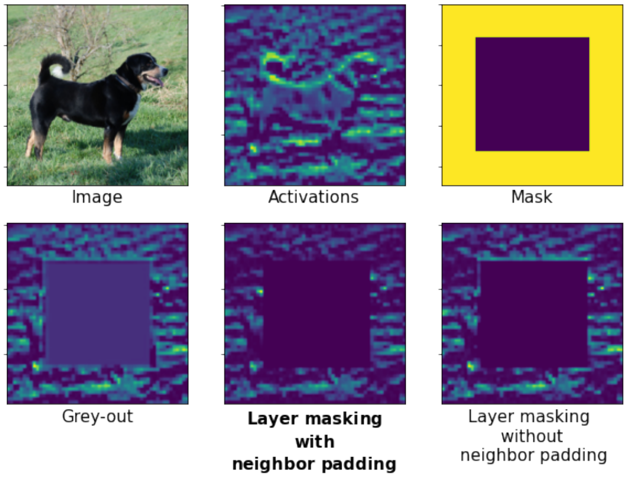

Figure 2: **Top row:** A dog image, sample activations after the first residual block, and mask to be applied to the image. **Bottom row:** The same activations when using grey-out masking, layer masking with neighbor padding (our method), and layer masking without neighbor padding (using zero padding). Neighbor padding helps in eliminating undesirable edge artifacts encountered in zero padding and greying out. Layer masking completely zeros out the masked out region unlike greying out which has non-zero values after a few layers

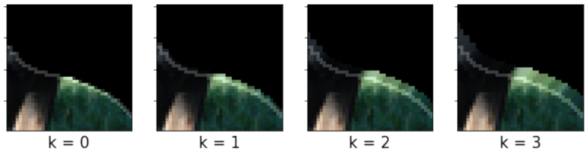

Figure 3: A visual depiction of neighbor padding on a part of the image as k increases. The grey line is the mask edge (added in for illustrative purposes), the cells near the edge are filled progressively with the average of their neighbors' values

4. Experiments

We examine the performance of layer masking compared to baselines on three dimensions: (1) the robustness of models as increasingly larger portions of the image are masked out; (2) the effect of mask shape on model prediction when the shape of the mask reveals information about the hidden object; and (3) the effect of masking method on LIME. We also examine the role of data augmentation during pretraining on missingness bias of different masking methods. We use **grey-out** (replace the masked out input with a grey color equal to the ImageNet mean) and **black-out** (replace the masked out input with a black color) as baselines, and focus on ResNet-50 [9] in this section (results on other CNN architectures in the supplementary). We evaluate all models on the ImageNet dataset, with segmentation masks from Pixel ImageNet [39] and saliency maps from Salient ImageNet [28].

4.1. Segment masking experiments

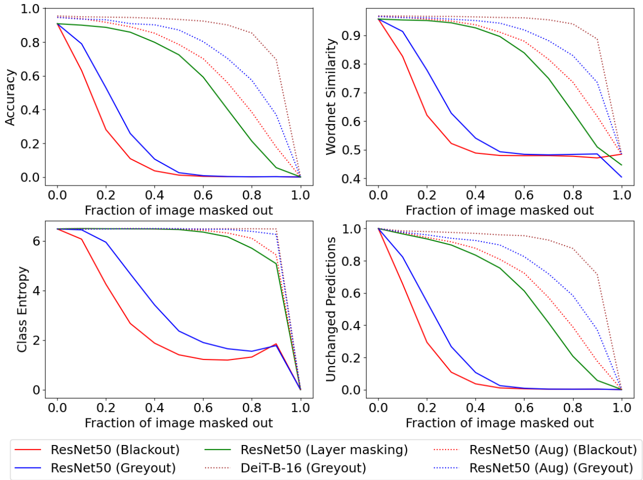

Figure 4: Metrics plotted as a function of fraction of 16×16 patches masked out in a random order using a given masking method and model. ResNet50 (Aug) refers to ResNet50 pretrained with grey missingness augmentations

To quantify the effect of feature masking methods on the model predictions, we study the behavior of the model when varying parts of the input are masked out. We first segment the image using a segmentation algorithm. Then, we analyze how the model output changes when more and more segments are masked out using a given masking technique. We characterize the model behavior using 4 metrics: accuracy, class entropy (defined as entropy of $p_f(y) = \mathbb{E}_{\boldsymbol{x} \in D}[1[f(\boldsymbol{x}) = y]]$), WordNet similarity [18] between predictions and true labels, and fraction of unchanged predictions. The WordNet similarity measures how similar the model predictions are to the true labels on a scale from 0 to 1 in place of a binary hit / miss. The fraction of unchanged predictions is a measure of the number of predictions changed by masking out parts of the input image. The class entropy indicates if the predictions are skewed towards a particular class or if they are equally distributed.

We also use the following segmentation algorithms to extract the segments from the image (1) **Square patches**: Segment the 224×224 image into smaller 16×16 square patches , (2) **SLIC** [1], and (3) **Quickshift** [32]. We tune hyperparameters for these algorithms such that they divide the image into approximately the same number of segments.

As in previous work [21, 12], we mask out these segments in three orders using their saliency scores: (1) **Randomly**, (2) **Most salient first**, (3) **Least salient first**. To compute the saliency scores, we select saliency maps from Salient ImageNet [28]. Each saliency map $m \in \mathbb{R}^{d \times d}$ is a pixel level saliency attribution array where $0 \leq m[i,j] \leq 1$ denotes the importance of the (i,j)th pixel to predicting the ImageNet class - the higher the value, the more salient the pixel. We then compute saliency scores for each segment by adding the saliency values for all pixels in that segment.

We then evaluate the metrics listed above, and plot them as a function of the fraction of segments masked out. The plots obtained by removing 16×16 sized patches from the images in a random order can be found in Fig. 4, while the rest can be found in the supplementary. The area-under-curve (AUC) of the accuracy and class entropy vs fraction of the segments masked for different baselines and order os masking can be found in Tab. 1.

Ideally, the masking technique should be such that the model ignores the masked out region completely. Thus, any performance drop of the model due to distribution shift should be minimal. This distribution shift can come about due to the unnatural baseline color and/or the shape of the mask. The less rapidly the metrics degrade, the more robust the model and the masking technique.

Fig. 4 and Tab. 1 indicate that ResNet-50 is much more robust over all metrics when the segments are removed using layer masking as opposed to black-out/grey-out. Greying out is also found to be better than blacking out, as expected. This difference in robustness persists across various segmentation and order of segment removal, and is the highest when 16×16 sized patches are removed. Surprisingly, removing random 16×16 patches degrades accuracy and class entropy more rapidly as compared to masking out the most salient regions first. This is because scattered blackout patches strongly resemble a maze or crossword pattern, while most salient regions are contiguous and do not resemble a maze as much. Thus, the model is confused by the shape of the mask and predicts incorrectly. This shows that the shape of the mask may also be a factor which contributes to missingness bias. We discuss this issue in Sec. 4.2.

Consistent with [21], we find that DeiTs are still more robust than ResNets - even when utilizing layer masking. Also, when ResNet-50 is pre-trained with data augmentations like RandAugment containing grey missingness approximations as in [35], performance drop due to distribution shifts is reduced dramatically. DeiTs also benefit from these data augmentations which makes them more robust than plain ViTs. However, we argue that data augmentation is only a partial solution for the problem of missingness bias. We discuss this in more detail in Sec. 4.3

4.2. Effect of mask shape on model prediction

We now examine the extent to which the model output relies on the shape of the mask itself when only the rel-

	Quickshift segments			16 × 16 patches			SLIC superpixels		
	Random	Most sal. first	Least sal. first	Random	Most sal. first	Least sal. first	Random	Most sal. first	Least sal. first
Accuracy									
Blackout	0.395	0.124	0.767	0.181	0.340	0.582	0.347	0.200	0.657
Greyout	0.463	0.137	0.787	0.240	0.374	0.621	0.418	0.234	0.702
Layer masking	**0.551**	**0.159**	**0.806**	**0.577**	**0.484**	**0.703**	**0.542**	**0.294**	**0.756**
Class entropy									
Blackout	4.988	2.224	5.824	2.574	4.570	5.406	4.815	3.976	5.661
Greyout	5.327	2.362	5.875	3.289	4.807	5.642	5.022	4.229	5.808
Layer masking	**5.698**	**2.572**	**5.892**	**5.651**	**5.782**	**5.879**	**5.607**	**4.763**	**5.876**

Table 1: AUC of the accuracy **(top)** and class entropy of the predictions **(bottom)** vs fraction of segments masked out

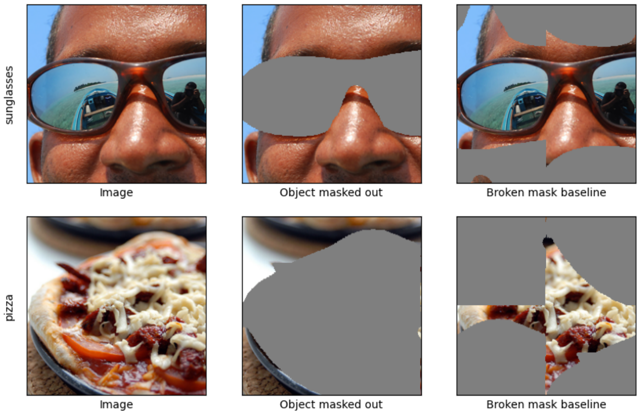

Figure 5: (**Left**) Two sample images from ImageNet, (**Middle**) with their relevant objects masked out, and (**Right**) the images with masks broken into four pieces and switched around

evant objects are masked out from the input. For example, the sunglasses-shaped mask in Fig. 5 may signal to the model that the object being masked out is sunglasses. Using a different mask shape can potentially decrease or remove this shape dependence but there is a possibility that this approach might unintentionally conceal areas of the input image that were meant to remain visible. In many cases, this is not a significant problem. However, when two significant objects are in close proximity, we may need to mask out one object while leaving the other unmasked. For example, we might require the sunglasses mask to *not* cover the nose or ears to evaluate the model's dependence on these features. We also do not know if the chosen mask shape (say a rectangle) is associated with any class (like 'box' or 'crate'). Thus, we require a masking technique which can precisely remove specific regions of the input but not leak any information about the masked input. Layer masking seems like a promising candidate, as it ensures that the masked regions are not processed by the model at all.

To quantify this effect, let us analyze the distribution shift introduced by masking in detail. There are primarily three components to it: (a) the removal of salient image content from the masked out region, (b) introduction of a new baseline color in place of the original content, and (c) the shape of the mask. All three can contribute to drop in accuracy after masking. However, our goal is to evaluate the effect of (c) on the accuracy. Suppose that we now compute the accuracy of the model on correctly classified images after masking the relevant objects using segmentation masks (second column, Fig. 5). All masking methods apply the same mask and remove the content completely, so we already control for (a). However, different masking methods can differ w.r.t (b), which means that the differences in accuracy drop after masking cannot be attributed to (c) alone. We control for this by also computing accuracy after breaking up the mask into four pieces (of size 112 × 112) and sending them to the opposite corners (third column, Fig. 5). We expect there to be little useful signal in the shape of the broken mask and any accuracy drop should arise from (a) and (b) only. Effect of (a) is the same across masking methods, so (a) is controlled. The effect of (b) in both cases is very similar as the area of the broken mask is the same as the original mask's area, so the difference in accuracy between the two cases (broken mask vs. object mask) should capture the effect of (c) on the model predictions. We expect (c) to have a positive impact on the accuracy (and thus negative impact on the extent of accuracy drop), as the shape of the object mask is a useful signal which can help with class prediction. Therefore, the larger the accuracy difference between the object mask and broken mask cases, the lower the effect of (c) on the model prediction after masking.

We carry out this experiment using images and segmentation masks from Pixel ImageNet [39]. We observe in Fig. 6 that layer masking has the lowest average accuracy on the object masked images, but the highest accuracy on the broken mask baseline which indicates that it has the lowest reliance on mask shape. We then pick a few classes over

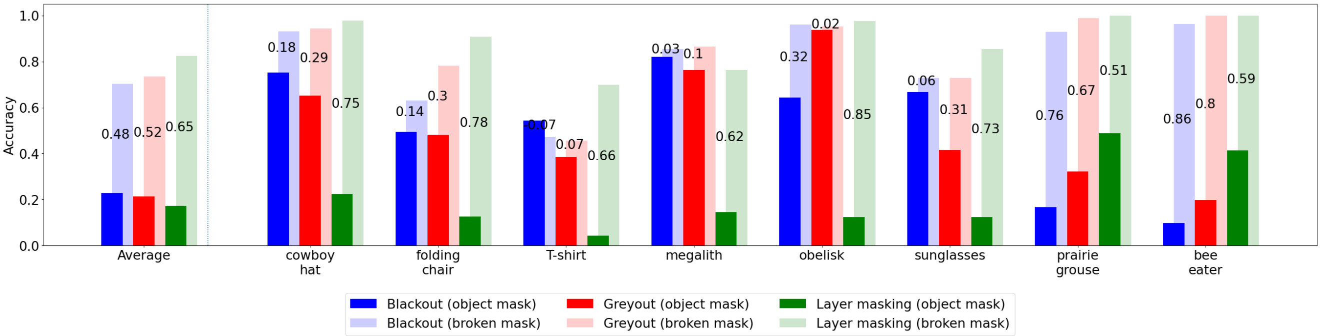

Figure 6: Accuracy of ResNet-50 over some salient classes when the relevant object(s) is masked out with various masking techniques (in dark colors) as compared to when the mask is broken into four (in light colors). 'Average' denotes average accuracy over all classes. The accuracy difference between broken mask and object mask cases is printed on the bars.

which the accuracy difference is either much lower or much higher for layer masking as compared to grey-out and blackout. For classes such as sunglasses and obelisk in which the shape carries a lot of information, the accuracy drop for blackout or greyout is much smaller than layer masking. This issue is exacerbated by the fact that the baseline grey or black color is relatively close to the true color of the objects for many of these classes, for e.g. obelisks are generally grey, sunglasses are black, etc. In some classes like T-shirt, the accuracy difference can even be negative! This could lead us to *overestimate* the model's reliance on unmasked features.

Surprisingly, there are a few classes in which the accuracy drop for greyout and blackout is much higher than for layer masking. It turns these classes have other closely related classes which are also associated with grey or black. For example, the priarie grouse (a bird) gets frequently misclassified as the black grouse under black out masking. Similarly, the bee eater (a bird) gets misclassified as a kite or vulture. This could lead us to *underestimate* the model's reliance on unmasked features. Layer masking removes such strong misleading signals, thus the accuracy is higher.

4.3. Effect of data augmentation on missingness bias

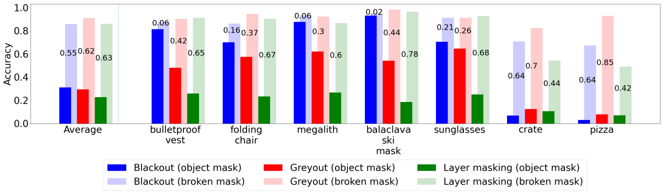

Figure 7: Same as Fig. 6 but for ResNet-50 pretrained with data augmentations with grey missingness approximations

Data augmentation strategies like AutoAugment[4], RandAugment[5] and RandomErasing [41] are sometimes used in the pretraining process for improved performance and generalization. They typically use a grey baseline color as a missingness approximation. Thus, models using these augmentations are very robust to greying out large parts of the input image (see Fig. 4). However, they may still rely the shape of the mask for their predictions. This issue is not as easily fixable by more training. We evaluate the shape sensitivity of a ResNet-50 trained with data augmentations[35] as in Sec. 4.2 and show results in Fig. 7. We observe that although the average accuracy of greyout/blackout baselines is now comparable to or higher than layer masking, the accuracy drop for layer masking is still a bit higher than that of blackout or greyout. Looking at accuracy over selected classes as before, we find that for some classes like sunglasses and megalith, the accuracy drop is still lower for blackout/greyout which implies that the model is relying on the mask shape to make its predictions. However, for classes like crate or pizza, the object mask covers most of the image and its shape does not reveal a lot of information about the class (Fig. 5). For these classes, all masking techniques have low accuracies when the object is masked. But when the mask is broken and parts of the object are visible, greyout works best as the model is exposed to grey color frequently during pre-training and is robust to such occlusions. Thus, the masking technique should be chosen with care depending on the specific image and use case.

4.4. Impact of masking techniques on LIME

We investigate the effect of masking methods on interpretability methods in this section using the example of LIME. Local Interpretable Model-Agnostic Explanations [22] or LIME is an interpretability method used to explain the predictions of black-box machine learning models by providing locally faithful and human-interpretable explanations. It works by approximating the decision boundary of a model in the vicinity of a particular instance or predic-

	Top-20 ablation accuracy (↓)			Alignment score (↑)			Top-20 Jaccard similarity (↑)		
	Quickshift	16×16	SLIC	Quickshift	16×16	SLIC	Quickshift	16×16	SLIC
Blackout	0.570	0.701	0.740	0.138	0.020	0.092	0.186	0.087	0.131
Greyout	0.348	0.609	0.574	0.231	0.078	0.171	0.231	0.113	0.172
Layer masking	**0.229**	**0.334**	**0.406**	**0.324**	**0.250**	**0.280**	**0.273**	**0.186**	**0.216**

Table 2: Top-20 ablation accuracy, alignment score, and top-20 Jaccard similarity of LIME scores over 512 random images

tion using a local, interpretable model. The local model is trained on images where the image features are randomly masked out. The weights of the local model represents the importance of each feature in the prediction.

4.4.1 Visual inspection

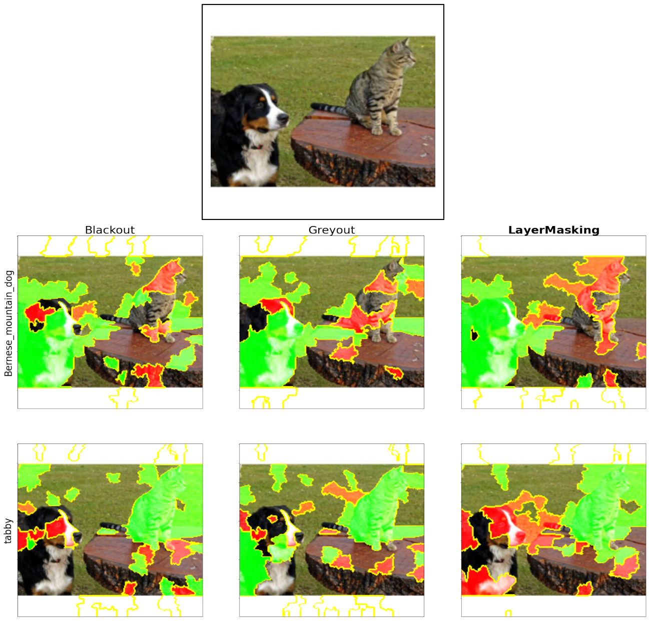

Figure 8: Visualization of LIME scores for the top two predictions of ResNet-50 on a sample image. Columns correspond to the masking techniques (blacking out, greying out, and layer masking), rows are the top 2 predictions. The top two predictions are Bernese mountain dog and tabby cat. Green regions contribute to the prediction, red regions detract from the prediction.

We first visually illustrate the impact of masking technique on LIME using an example of an image of a cat and a dog, (Fig. 8). For the LIME explanations, the top 10 segments with the highest magnitude LIME score are highlighted. Red segments detract from the prediction, and a green segments contribute to the prediction. LIME with greyout or blackout masking assigns parts of the cat high positive scores (in green) and parts of the dog negative scores (in red) in the explanation of the prediction of Bernese mountain dog, and vice versa for the explanation of the tabby prediction. The third column containing the LIME explanations obtained using layer masking is much more aligned with human intuition. Visually, LIME with layer masking seems to produce better explanations. We present more examples in the supplementary.

4.4.2 Quantitative evaluation

We use these metrics for evaluating LIME explanations:

(a) **Top-k ablation test** [24, 29]: Choose the k most important segments according to the explanation, remove them by substituting with a missingness approximation (we use grey), and compute the accuracy on the masked images. The more the accuracy drops, the better the explanations.

(b) **Alignment score**: Cosine similarity between importance scores returned by LIME and mean normalized segmentation mask. Formally, assuming we have access to segmentation mask $\boldsymbol{m} \in [0,1]^{d\times d}$ for an image of dimension d, we compute $\boldsymbol{g}_i = \sum_{(u,v)\in \text{segment } i} (\boldsymbol{m}[u,v] - \bar{\boldsymbol{m}})$ for each segment i where $\bar{\boldsymbol{m}}$ is the mean of the segmentation mask. If the importance scores are $\boldsymbol{s}$, then the alignment score is the cosine similarity between $\boldsymbol{g}$ and $\boldsymbol{s}$

(c) **Top-k Jaccard similarity**: Jaccard similarity between the top k features and the segmentation mask $\boldsymbol{m}$.

While (a) does not depend on any “ground truth” for evaluation purposes, (b) and (c) use the object segmentation mask as a substitute for the ground truth. We use 512 random images and segmentation masks from Pixel ImageNet [39] for calculating the above metrics. In these images, the correct class is within the top 3 predictions of the model. We find that layer masking is much superior to blacking or greying out the input across different segmentation algorithms and different metrics, which confirms our intuitions from the visual inspection. As before, the improvement is most significant when the segments are 16 × 16 sized patches.

5. Ablation study

To further investigate the effect of layer masking and neighbor padding on model behavior, we construct 3 variants of layer masking: (a) With zero padding instead of neighbor padding (b) Masking and padding only the first two residual blocks, (c) Masking and padding only the first convolutional layer, ReLU and BatchNorm layer

Using a similar setup as in Sec. 4.1, we compute the area under curve (AUC) for each plot of metric vs fraction of segments dropped. The AUC values are averaged over different segmentation algorithms (SLIC, quickshift, etc) and masking orders (random, salient first, etc)(refer Tab. 3).

We find that both neighbor padding and masking all layers are important to the masking technique. Layer masking with zero padding is still better than blackout or greyout, but much worse than with neighbor padding. Layer masking only the first two residual blocks is also inferior to masking through all layers, but we find that there are diminishing returns, as we are able to obtain much of the improvement by masking only half of the layers.

	Accuracy	Class Entropy	Wordnet Similarity	Unchanged Predictions
Blackout	0.3881	4.6473	0.6930	0.4094
Greyout	0.4398	4.9408	0.7167	0.4636
Layer masking:				
On all layers	**0.5604**	**5.6021**	**0.7881**	**0.5907**
On 1st and 2nd residual blocks	0.5103	5.0962	0.7616	0.5391
Zero padding	0.4502	5.0388	0.7262	0.4747

Table 3: Average AUC for different variants of layer masking alongside the black out and grey out baselines (model: ResNet-50). Higher the better

6. Conclusion

In this paper, we have presented a new masking technique such that the model output is both (a) perfectly insensitive to the masked out portion of the input and (b) only focused on the unmasked input and not the masking pattern. We find that layer masking can make CNNs like ResNets very robust to removal of large parts of input without retraining, especially when the masking patterns can get confused with output classes like 16×16 patch occlusions. Layer masking also does not depend on the shape of the mask, which can be important if we need to precisely mask out only a specific object from the image and the shape of the object carries some useful signal. We further find that LIME scores obtained using layer masking are better compared to blacking or greying out on multiple metrics like top-k ablation test, top-k Jaccard similarity, and alignment score. We show that this technique can be of great use in both manual feature/object removal for model debugging, and for interpretability techniques like LIME which rely on the ability to remove features from the input without any major distribution shift.

7. Acknowledgements

This project was supported in part by NSF CAREER AWARD 1942230, ONR YIP award N00014-22-1-2271, ARO's Early Career Program Award 310902-00001, Meta grant 23010098, HR001119S0026 (GARD), Army Grant No. W911NF2120076, NIST 60NANB20D134, the NSF award CCF2212458, a capital one grant and an Amazon Research Award.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[2] Saeid Asgari, Aliasghar Khani, Fereshte Khani, Ali Gholami, Linh Tran, Ali Mahdavi-Amiri, and Ghassan Hamarneh. Masktune: Mitigating spurious correlations by forcing to explore. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[3] Chun-Hao Chang, Elliot Creager, Anna Goldenberg, and David Duvenaud. Explaining image classifiers by counterfactual generation. In *International Conference on Learning Representations*, 2019.

[4] Ekin Dogus Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data. 2019.

[5] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. arxiv e-prints, page. *arXiv preprint arXiv:1909.13719*, 4, 2019.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[7] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE international conference on computer vision*, pages 3429–3437, 2017.

[8] Jindong Gu, Volker Tresp, and Yao Qin. Are vision transformers robust to patch perturbations? In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 404–421, Cham, 2022. Springer Nature Switzerland.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[10] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[11] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[12] Saachi Jain, Hadi Salman, Eric Wong, Pengchuan Zhang, Vibhav Vineet, Sai Vemprala, and Aleksander Madry. Missingness bias in model debugging. In *International Conference on Learning Representations*, 2022.

[13] Alexander Levine and Soheil Feizi. (de)randomized smoothing for certifiable defense against patch attacks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[14] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European conference on computer vision (ECCV)*, pages 85–100, 2018.

[15] Jiang Liu, Alexander Levine, Chun Pong Lau, Rama Chellappa, and Soheil Feizi. Segment and complete: Defending object detectors against adversarial patch attacks with robust patch detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14973–14982, 2022.

[16] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[17] Dina Mardaoui and Damien Garreau. An analysis of lime for text data. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 3493–3501. PMLR, 13–15 Apr 2021.

[18] George A. Miller. WordNet: A lexical database for English. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994.

[19] Mazda Moayeri, Phillip Pope, Yogesh Balaji, and Soheil Feizi. A comprehensive study of image classification model sensitivity to foregrounds, backgrounds, and visual attributes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19087–19097, 2022.

[20] Mazda Moayeri, Sahil Singla, and Soheil Feizi. Hard imagenet: Segmentations for objects with strong spurious cues. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[21] Muzammal Naseer, Kanchana Ranasinghe, Salman Khan, Munawar Hayat, Fahad Khan, and Ming-Hsuan Yang. Intriguing properties of vision transformers. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[22] Marco Tulio 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, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

[23] Hadi Salman, Saachi Jain, Eric Wong, and Aleksander Madry. Certified patch robustness via smoothed vision transformers. *CoRR*, abs/2110.07719, 2021.

[24] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 28(11):2660–2673, 2017.

[25] Rulin Shao, Zhouxing Shi, Jinfeng Yi, Pin-Yu Chen, and Cho-Jui Hsieh. On the adversarial robustness of vision transformers, 2021.

[26] Shikhar Sharma, Ryan Kiros, and Ruslan Salakhutdinov. Action recognition using visual attention. *arXiv preprint arXiv:1511.04119*, 2015.

[27] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3145–3153. PMLR, 06–11 Aug 2017.

[28] Sahil Singla and Soheil Feizi. Salient imagenet: How to discover spurious features in deep learning? In *International Conference on Learning Representations*, 2022.

[29] Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Visualizing the impact of feature attribution baselines. *Distill*, 2020. https://distill.pub/2020/attribution-baselines.

[30] Mukund Sundararajan and Ankur Taly. A note about: Local explanation methods for deep neural networks lack sensitivity to parameter values. *CoRR*, abs/1806.04205, 2018.

[31] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[32] Andrea Vedaldi and Stefano Soatto. Quick shift and kernel methods for mode seeking. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision – ECCV 2008*, pages 705–718, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[33] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2017.

[34] Tianlu Wang, Jieyu Zhao, Mark Yatskar, Kai-Wei Chang, and Vicente Ordonez. Balanced datasets are not enough: Estimating and mitigating gender bias in deep image representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5310–5319, 2019.

[35] Ross Wightman, Hugo Touvron, and Herve Jegou. Resnet strikes back: An improved training procedure in timm. In *NeurIPS 2021 Workshop on ImageNet: Past, Present, and Future*, 2021.

[36] Kai Xiao, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. Noise or signal: The role of image backgrounds in object recognition. *arXiv preprint arXiv:2006.09994*, 2020.

[37] Maksym Yatsura, Kaspar Sakmann, N Grace Hua, Matthias Hein, and Jan Hendrik Metzen. Certified defences against adversarial patch attacks on semantic segmentation. *arXiv preprint arXiv:2209.05980*, 2022.

[38] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

[39] Shiyin Zhang, Jun Hao Liew, Yunchao Wei, Shikui Wei, and Yao Zhao. Interactive object segmentation with inside-outside guidance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12234–12244, 2020.

[40] Heliang Zheng, Jianlong Fu, Tao Mei, and Jiebo Luo. Learning multi-attention convolutional neural network for fine-grained image recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 5209–5217, 2017.

[41] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation, 2017.