A Additional Experiments

A.1 SmoothGrad for parameter saliency

Our parameter saliency method applies to the parameter space the ideas related to Vanilla input-space saliency maps [37]. Alternatively, we can compute parameter saliency applying to parameter space other input-space ideas, such as SmoothGrad[38]. In this section, we compare SmoothGrad for parameter saliency to our original method. We implement parameter-space SmoothGrad by perturbing the input features but taking the gradient with respect to the model parameters. We also implement the input-space counterpart for the parameter-space SmoothGrad method. Figure 9 presents the results.

(a) Original method   (b) 1% noise   (c) 5% noise   (d) 10% noise   (e) 50% noise

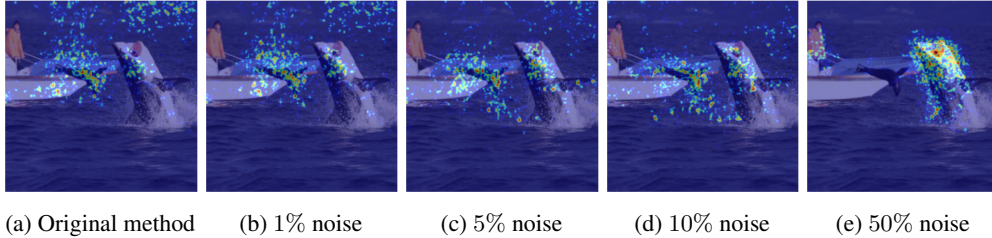



Figure 9: **Input-space counterparts of Parameter SmoothGrad and Original Parameter Saliency Method** (a) Original Method (b) Parameter SmoothGrad with 1% noise (c) Parameter SmoothGrad with 5% noise (d) Parameter SmoothGrad with 10% noise (e) Parameter SmoothGrad with 50% noise.

We observe that our original Vanilla-like parameter saliency is consistent with SmoothGrad-based parameter saliency both in terms of the parameter saliency profile and the input-space saliency. On our case study "great white shark" example, the two methods produce very similar results for small amounts of noise injected by SmoothGrad into the input image (up to 10% noise, while the network similarly misclassifies both the original and perturbed input image as "killer whale"). However, as the amount of SmoothGrad noise increases, the spurious features such as the background features in the sky get highlighted less, and the input-space saliency becomes more focused on the seal. This behavior is expected since noise corrupts the sparse spurious correlations the network relied on before while another misclassification reason – the seal – remains present. As the amount of SmoothGrad noise increases further (to 50% noise), the network misclassifies the perturbed input image with labels other than "killer whale" (e.g. "mongoose", "plane") and the input-space saliency stops being consistent with the Vanilla-like original approach.

A.2 Analysis of Correctly Classified Samples

In this section we perform the pruning experiment for correctly classified samples and confirm that pruning salient filters decreases model output confidence for correctly classified images as seen in Figure 10. Additionally, we perform the analysis of nearest neigbors of correct classifications.

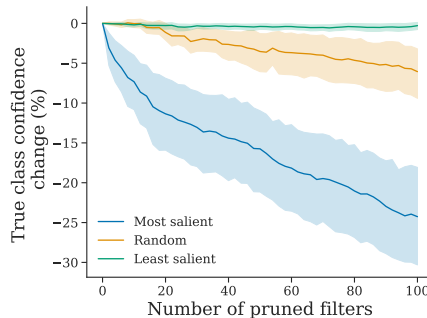



Figure 10: **Effect of pruning on correctly classified samples**

We conduct the nearest neighbor search in the parameter saliency space on 10,000 correctly and incorrectly classified image respectively. All images are sampled from the ImageNet validation set.

15

we find that on average, 89.0% of the 10 nearest neighbors of the correct inputs are also correct, whereas for incorrectly classified samples, only 10.8% of their 10 nearest neighbors are correctly classified.

A.3 Parameter saliency profiles for other network architectures

In this section, we present average saliency profiles for several popular network architectures other than ResNet-50 [18]. Analogously to Figure 1, Figure 11 presents average gradient magnitudes for VGG-19 [36], Inception v3 [45], and DenseNet [19]. Similarly to Figure 2, we also present in Figure 12 standardized filter-wise saliency profiles for those architectures averaged across correctly and incorrectly classified ImageNet [9] samples.

(a) VGG-19

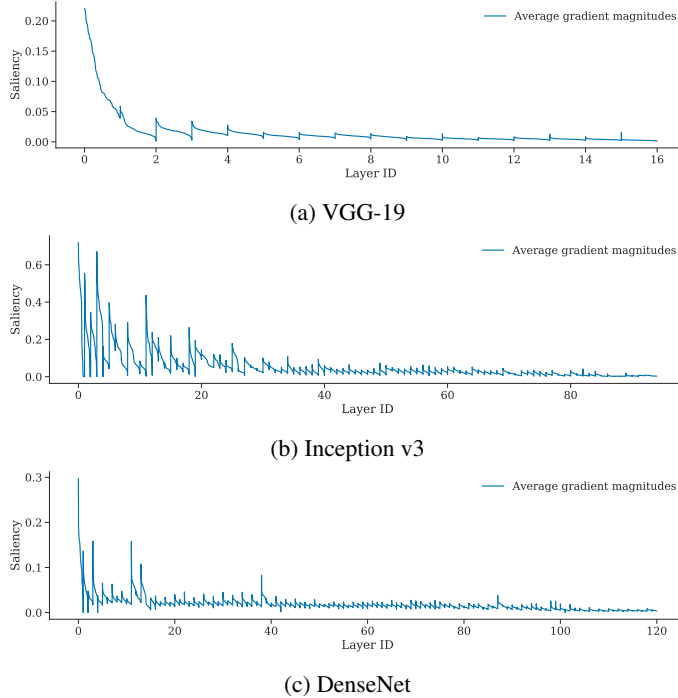
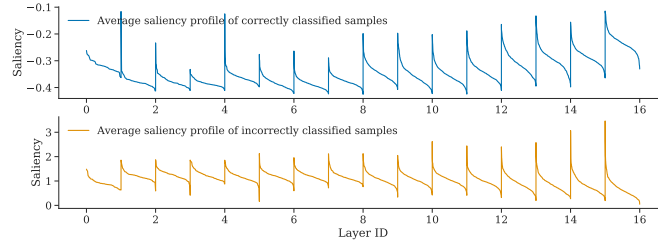
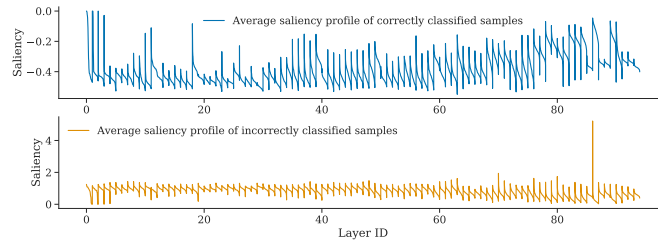
(b) Inception v3

(c) DenseNet

Figure 11: **Filter-wise saliency profiles for other architectures.** (a) VGG-19 saliency profile (without standardization). (b) Inception v3 saliency profile (without standardization). (c) DenseNet saliency profiles (without standardization). In each panel the filter-wise saliency profile is averaged over the ImageNet validation set. In every panel, the filter saliency values in each layer are sorted in descending order, and each layer's saliency values are concatenated. The layers are displayed left-to-right from shallow to deep and have equal width on x-axis.

A.4 More examples of nearest neighbors

We present more examples of nearest neighbors in our parameter saliency space. Figure 14 are nearest neighbors in CIFAR-10 [21] dataset, where reference images are chosen from samples misclassified by our classifier. Figure 15 are examples from ImageNet, where images are captioned with the true label of the reference images.

In addition, in comparison with the nearest neighbor in our parameter saliency space, we also conduct the nearest neighbor search in the feature representation space. We take the feature representation from the `conv5_3` layer of a ResNet-50, and run the nearest neighbor search using the same reference images and candidate pool as in Figure 4. Results are shown in Figure 13. We find that nearest neighbors in the feature space bear more resemblance in image structures, but it fails to identify samples that share the same mistakes. In fact, most of the nearest neighbors in Figure 13 are correctly classified samples.

16

(a) VGG-19

(b) Inception v3

(c) DenseNet

Figure 12: **Standardized saliency profiles averaged over correctly vs incorrectly classified samples.** (a) VGG-19 saliency profiles. (b) Inception v3 saliency profiles. (c) DenseNet saliency profiles. In each panel, the top row presents the standardized saliency profiles averaged over correctly classified samples and the bottom row shows standardized saliency profiles averaged over incorrectly classified samples. On every panel, the filter saliency values in each layer are sorted in descending order, and each layer's saliency values are concatenated. The layers are displayed left-to-right from shallow to deep and have equal width on x-axis.

(a)   (b)

Figure 13: **Examples of nearest neighbors in the feature representation space (from ImageNet)**.

Furthermore, we present preliminary results on applying our method to language models. We use a BERT [11] model, pre-trained on the task of predicting the word at a masked position. We consider an independent dataset for evaluation in our experiments, which consists of short sentences with masked words, provided by LAMA [28, 29], an open-source language model analysis framework. Similar to the filter-wise aggregation in convolutional networks, we adopt column-wise aggregation for obtaining our saliency profiles in the transformer-based architecture. We conduct the nearest

17

neighbor search by comparing sentences from the dataset in saliency space and analyzing the top-5 nearest neighbors.

We present four examples below, where the numbering indicates n-th nearest neighbor sentence, and the italic word is the ground truth. Each example is accompanied with a description of similarities between the neighbors:

Reference: "Cany Ash and Robert Sakula are both Architects." incorrectly predicted as: actors

1. "David Castlles-Quintana and Vicente Royuela are *economists*." incorrectly predicted as: actors
2. "Raghuram Rajan is an *economist*." incorrectly predicted as: actor
3. "Richard G. Wilkinson and Kate Pickett are *British*." incorrectly predicted as: actors
4. "Nathan Alterman was a *poet*." correctly predicted: poet
5. "Zbigniew Badowski is an *architect*." incorrectly predicted as: author

Note that all 5 neighboring sentences here share the common structure of being declarations of profession for specifically named people. Interestingly, the first three closest sentences to the reference incorrectly predict the profession to be an actor as well. Moreover, the ground truth of the reference and its closest neighbor is economist/s.

Reference: "D'Olier Street is in Dublin." incorrectly predicted as: Paris

1. "A group who call themselves Huguenots lives in *Australia*." incorrectly predicted as: France
2. "Huguenots and Walloons settled in *Canterbury*." incorrectly predicted as: France
3. "In the Treaty of Lisbon 2007 *Ireland* refused to consent to changes." incorrectly predicted as: it
4. "Samuel Marsden Collegiate School is located in *Wellington*." incorrectly predicted as: Melbourne
5. "Konstantin Mereschkowski has *Russian* nationality." correctly predicted: Russian

Again, we see all five nearest neighbor sentences are semantically similar to the reference, this time relating to national affiliations and geography. In the first two closest sentences, the model incorrectly fills in a location with France, which is similar to the reference which incorrectly predicts Paris.

Reference: "The Super Bowl sponsor was the *Gap* clothing company." incorrectly predicted as Nike

1. "During Super Bowl 50 the *Nintendo* gaming company debuted their ad for the first time." incorrectly predicted as: video
2. "Experimental measurements on a model steam engine was made by *Watt*." incorrectly predicted as: Siemens
3. "ABC's programming strategy was criticized in May 1961 by *Life* magazine." incorrectly predicted as: Time
4. "In 2009, Doctor Who started to be shown on Canadian cable station *Space*." incorrectly predicted as: CBC
5. "To emphasize the 50th anniversary of the Super Bowl the *gold* color was used." incorrectly predicted as: blue

All but one of the nearest neighbors in this example relate to some kind of TV programming, and two also mention the Super Bowl. Much like the reference sentence, which incorrectly predicts the name of a corporation, the first four neighbors have a ground truth or incorrect prediction that is also a corporation.

Reference sample: "Tetzel's collections of money to free souls from purgatory was objected by *Luther*." incorrectly predicted as: some

1. "Newcastle was granted a new charter in 1589 by *Elizabeth*." incorrectly predicted as: Parliament

18

2. "The suggestion that imperialism was the "highest" form of capitalism is from *Lenin*." incorrectly predicted as: Aristotle

3. "Fritschel said the man's sleep was disturbed by *dreams*." incorrectly predicted as: lightning

4. "The concept that falling objects fell at the same speed regardless of weight was introduced by *Galileo*." incorrectly predicted as: NASA

5. "One of the earliest examples of Civil Disobedience was brought forward by the *Egyptians*." incorrectly predicted as: government

This is an example where there is a weak relation between the incorrect classifications (three of five involve some kind of government or government organization) of the nearest neighbors, but the sentences are still highly semantically similar. All of the neighbors except the third are declarations of historical actions performed by a specific person or group of people.

    (a) bird ↔ cat   (b) frog ↔ cat   (c) airplane ↔ ship   (d) horse ↔ deer

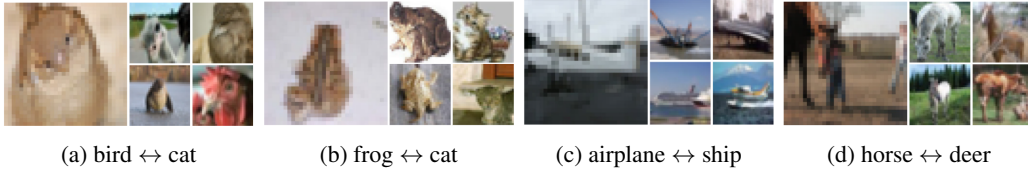



Figure 14: **CIFAR-10 examples of nearest neighbors in parameter saliency space**. On CIFAR-10 images that cause similar filters to malfunction are often misclassified in a similar way.

     (a) coyote   (b) goldfish   (c) terrapin   (d) great white shark

Figure 15: **ImageNet examples of nearest neighbors in parameter saliency space**. In every panel, the reference image is in the left column and its nearest neighbors are in the right column. Panels are captioned by the true label of their reference image.

A.5   Fine-tuning salient filters of a VGG-19

In this section, we conduct the fine-tuning experiment introduced in Section 3.3 on a VGG-19 network trained on ImageNet, which has a total of 5504 filters. The learning rate for training our VGG network is 1/10 of that for the ResNet, so we decrease the fine-tuning step size by 10 in this experiment. Figure 16 shows the effect of updating salient filters of a VGG-19. Note that we use the same range for the number of tunable filters in this experiment; 300 filters correspond to 5.5% of total filters in a VGG-19, while it is 1.2% for a ResNet-50.

A.6   Random perturbation of salient filters

As an alternative to the pruning approach described in Section 3.1, random perturbations could be used to show that the most salient filters are indeed responsible for misclassification. We perturbed the filters using small Gaussian noise $\mathcal{N}(0, 0.001)$. Figure 17 presents the effect of randomly perturbing salient filters, we observe similar trends to our pruning experiments in Section 3.1.

A.7   Connection to adversarial attacks in parameter space

Adversarial attacks in parameter space have been used for optimizers which find flat loss minima [16, 22, 12] and for improving model robustness through parameter-corruption-resistant training [43].

One could instead apply adversarial attacks to construct parameter saliency profiles – choose a constraint space and perturb parameters in order to minimize loss, subject to the constraint, using the perturbation to parameters as a saliency profile (perhaps standardizing afterwards). We notice

19

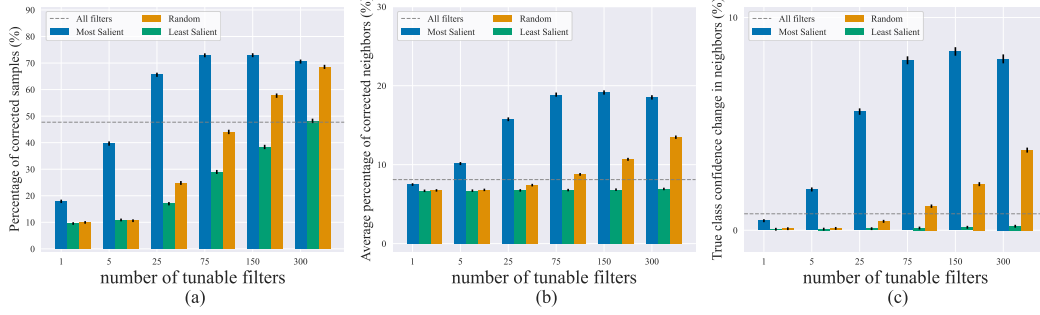



Figure 16: **Effect of updating a small number of filters on VGG-19.** (a) Percentage of samples that are corrected after fine-tuning. (b) Average percentage of nearest neighbors that are also corrected after fine-tuning. (c) Average change in the confidence score of the true class among nearest neighbors. The horizontal line in each plot is the effect of updating the entire network.

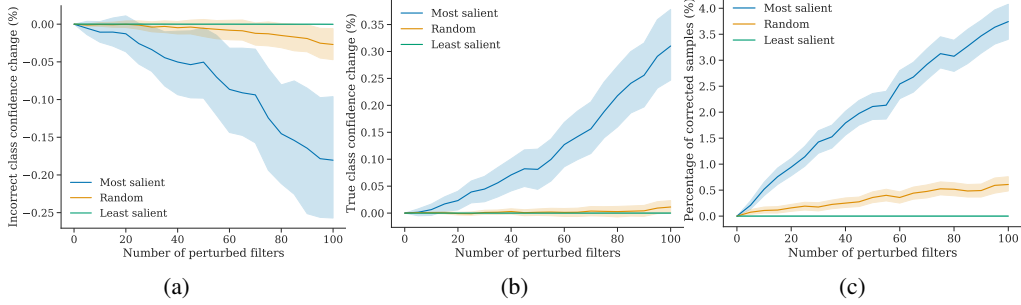



Figure 17: **Effect of randomly perturbing filters.** (a) Change in incorrect class confidence score. (b) Change in true class confidence score. (c) Percentage of samples that were corrected as the result of pruning filters. These trends are averaged across all images misclassified by ResNet-50 in the ImageNet validation set. The error bars represent 95% bootstrap confidence intervals.

several advantages and disadvantages of this alternative. On the one hand, the "adversarial" approach requires a choice of constraint space and may require more compute (our method results in the exact same saliency profile as a single-step adversary). On the other hand, the choice of constraint space and optimizer in the adversarial approach yields more flexibility.

In this section, we compare our method with the adversarial-attack-based method. More specifically, for a given sample (x, y), we perturb parameters either to maximize or minimize the loss subject to an L2-norm constraint on the parameter change:

$$\min_{\theta} / \max_{\theta} \mathcal{L}_{\theta}(x, y) \tag{5}$$
$$\text{s.t. } \|\theta - \theta_0\| < \epsilon$$

Then, given the adversarially perturbed parameters θ^*, we define the adversarial-attack-based parameter saliency profile of the sample (x, y) as the absolute difference from the initial parameters θ_0: $s(x, y) = |\theta^* - \theta_0|$.

We compare the resulting adversarial saliency profiles with our original method on a random sample of 100 images from the ImageNet validation set. We observe that for a reasonably small constraint ($\epsilon = 10^{-4}$), the resulting adversarial saliency profiles are similar to our original parameter saliency profiles (as one would expect for smooth loss) with the average cosine similarity between the saliency profiles generated by each method for the same images reaching 0.99 (the average is taken over the random sample of 100 images). We also see that both methods agree on the top-k (we tried k=100) most salient filters: on average, 95% of filters identified by our original method as top-k salient filters were also identified as top-k salient filters by the adversarial parameter saliency. The differences were gradually more distinct with larger constraints, however, we note that the smaller epsilons are of greater interest since they reflect the intuition of perturbing only the most important parameters.

20

(a) Reference image
4.07%   great white shark
78.36 %   **killer whale**

(b) Input-space saliency for top 10 salient filters (pixels most responsible for misclassification)

(c) Masked human
3.84%   great white shark
86.04%   **killer whale**

(d) Masked non-salient water
5.15%   great white shark
79.38%   **killer whale**

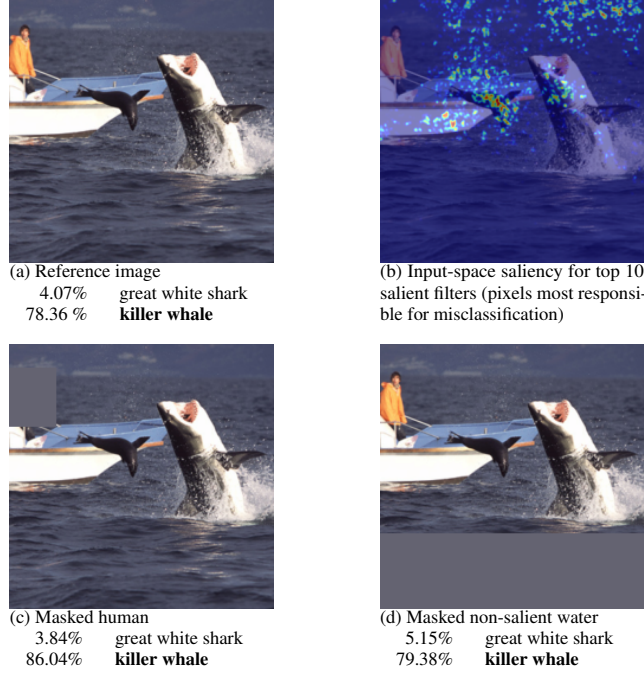



Figure 18: **Masking non-salient parts of the image.** (a) Reference image of "great white shark" misclassified by the model as "killer whale" and the corresponding confidence scores. (b) Pixels that cause the top 10 most salient filters to have high saliency. (c) Masked (non-salient) human. (d) Masked non-salient water region.

We also tried L1-regularized adversarial attacks. We can similarly enforce sparsity in our original method by simply adding the L1 regularizer to our loss before computing the gradient:

$$s(x,y)_i := |(1 - \alpha)\nabla_{\theta_i}(\mathcal{L}_\theta(x,y) + \alpha\|\theta - \theta_0\|_1)| \qquad (6)$$

This modification can be seen as one step of the L1-regularized adversarial attack, and we experimentally checked that it produces very similar results with the cosine similarity between the resulting saliency profiles of 0.97 on average (for sufficiently large $\alpha = 0.99$).

A.8   Additional case study figures

Masking non-salient parts of the image. As noted in section 3.4 and presented in Figure 18, masking the non-salient parts of the image results in even worse misclassification confidence with the incorrect class confidence increasing compared to the reference image.

Pasting the salient region from the reference image onto other "great white shark" images. As mentioned in section 3.4, in order to further investigate the effect of the salient region, we pasted it (i.e., the seal and the boat) from the original image onto other images with "great white shark" ground truth label that were correctly classified by ResNet-50 (see Figure 19 for examples). We observed that this increased the probability of "killer whale" for 39 out of 40 examples of great white sharks from the ImageNet validation set with an average increase of 3.75%.

Examples of images with "killer whale" label. As we discussed in section 3.4, the model might have learned to correlate a combination of sea creatures (e.g. a seal) and man-made structures (e.g. a boat) with the "killer whale" label. Images of killer whales in ImageNet often have man-made structures which look similar to the boat, we provide examples of that in Figure 20.

A.9   Exploring neural network mistakes

In Section 3.4, we apply our parameter-space saliency method as an explainability tool using it alongside our input-space technique to study how image features cause specific filters to malfunction.

21

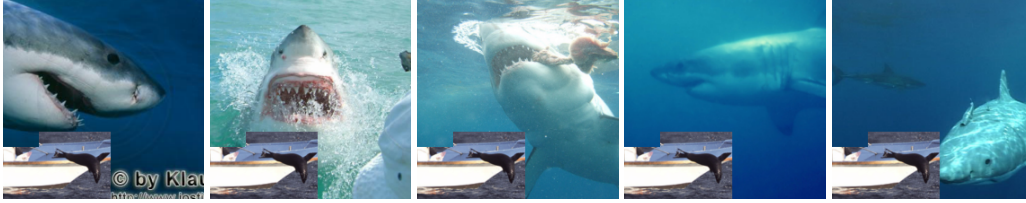



Figure 19: **Sample "great white shark" images with boat and seal.** The salient region from the case study image pasted onto other "great white shark" images.

Figure 20: **ImageNet examples of "killer whale".**

In our case study, the salient pixels that confuse the network are focused less on the target object than on other image features, and masking the salient regions which are not on the target object improves the network behavior. Such examples expose the network's reliance on spurious correlations and constitute an interesting type of model mistake.

The masking approach can be adopted to explore network mistakes and find other interesting cases (e.g., cases where the salient pixels are not concentrated on the target object). Investigating examples where masking the most salient pixels improves performance may provide insights into the model's misbehavior as well as expose dataset noise and biases. We select misclassified samples where masking the top 5% of salient pixels leads to an increase of at least 25% in confidence corresponding to the correct class. We showcase representative examples of different types of mistakes that we observe in those samples in Figure 8. Many of the neural network misclassifications stem from classifying a non-target object in images with multiple objects (see Figure 8 (a), (b)). However, other mistakes are triggered by background features (Figure 8 (d)), dataset biases (as our case study experiments in Figure 7 suggest), and label noise (Figure 8 (e)).

Interestingly, in some of the selected cases the salient regions still focus on the target object features (see Figure 8 (c)), and masking them improves the model's behavior. Masking salient target object features that confuse the network seems to be particularly beneficial for images of animals; for example, masking dog ears helps the network identify the correct breed (see Figure 21(a)) or masking spot patterns helps distinguish different types of rays (see Figure 21(b)).

While masking the top 5 % of salient pixels results in correct classification for all samples in Figure 8 and in Figure 21(a), (b), this is, of course, not always the case. Sometimes, pixels which cause large filter saliency values are focused densely on the target object, and masking them results in decreased confidence corresponding to the correct class. Selecting such samples can be used to find situations where the network is genuinely confused by the target object rather than background features (Figure 21 (c)) or samples with multiple objects present and with salient pixels more focused on the target object (Figure 21 (d)).

A.10   Comparison to GradCAM

Existing input saliency maps used with the predicted label can highlight features which are related to misclassification. However, they are not specifically geared towards that goal. Our input saliency technique highlights image features that cause specific filters to malfunction and those features, while they might in some cases coincide with the features that explain a high class confidence score corresponding to the predicted label, may not be the same.

In this section, we will compare our input-space saliency technique which highlights pixels that drive high parameter saliency values of specific filters (i.e., pixels that confuse the network) to the visual

22

| | (a) | malamute
husky | (b) | stingray
electric ray | (c) | appenzeller
sennenhund | (d) | redbone
beagle |

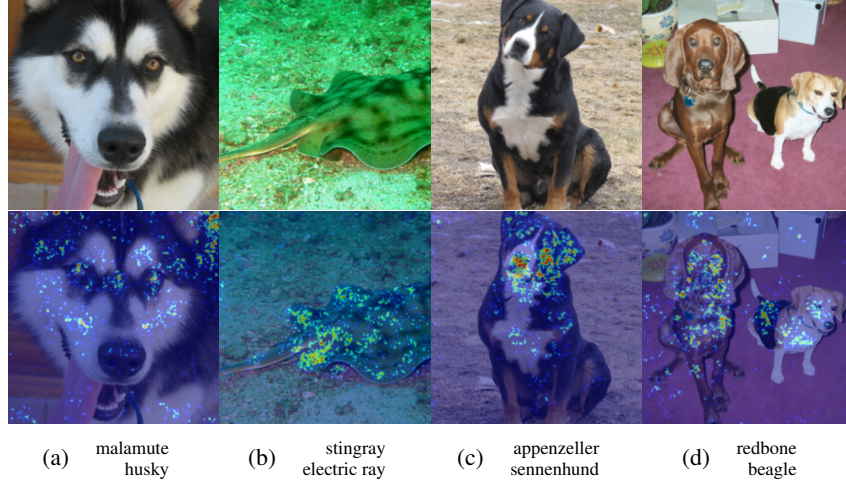



Figure 21: **Pixels responsible for mistakes focused on the target object.** (a)-(b) Masking the salient pixels corrects the misclassification where masking confusing features (e.g. dog ears or spot patterns) helps distinguish animals. (c)-(d) Masking the salient pixels results in correct class confidence decrease, when the salient pixels are densely focused on the target object. The correct class label is specified in the top row and the predicted incorrect class label – in the bottom row of the subcaption on each panel.

explanations produced by GradCAM with the predicted label [6] [34] – one of the most popular and high quality input-saliency methods.

Figures 22 and 23 present panels of images comparing our method to GradCAM explanations computed with the predicted label. From the perspective of highlighting pixels responsible for neural network mistakes and for driving high filter saliency values, we note the following:

- GradCAM highlights the object that corresponds to the incorrect label, and the entire target object is highlighted in the images where only the target object is present (see Figure 22 (c)-(e), Figure 23 (a)-(c)). However, when our method focuses on the target object, it highlights specific features of that object. Those are the features that confuse the network, and masking them can correct the misclassification.

- In cases where the network classifies the non-target object in the image (see Figure 22 (a)-(b), Figure 23 (d)), both methods highlight the non-target objects. However, GradCAM is more localized to the non-target object. This is expected since GradCAM produces visual explanations for the predicted label (and has been shown to produce highly localized saliency maps [34]) while our method highlights all pixels that drive the filter saliency, and these pixels may be located on the target object as well.

- In cases where the misclassification does not stem from confusion by the target object or classifying the non-target object (see Figure 22 (d)-(e), Figure 23 (e)), our method highlights background features and/or a combination of non-target object features, while GradCAM still highlights the target object. For example, in Figure 23 (e), our method highlights the boat and the sky much more than GradCAM, and our case study masking experiments in section 3.4 show that those regions indeed confuse the network.

- In addition, we emphasize that our input saliency technique is specific to the chosen filter set F and is introduced to study the interaction between the image features and the malfunctioning filters. In contrast, GradCAM is not able to relate image-space mistakes to an arbitrary set of model parameters or filters chosen by the user.

To summarize, GradCAM (as well as many other input-space saliency methods) was designed to be highly localized to the object correponding to the label of interest, while our method highlights sparse fine-grained features of images which we believe is a desirable property for our specific application.

---

[6]Implementation from https://github.com/kazuto1011/grad-cam-pytorch under MIT license

23

Therefore, we opt for using input-gradient information similar to the original Vanilla Gradient [37] method. However, instead of class confidence scores, we use a different loss – cosine distance to the boosted parameter-saliency profile (as described in Section 2.2) which allows us to explore how image features cause specific filters to malfunction.

	dalmation soccer ball		pizza bell pepper		leopard jaguar		junco house finch		passenger car locomotive
(a)		(b)		(c)		(d)		(e)	

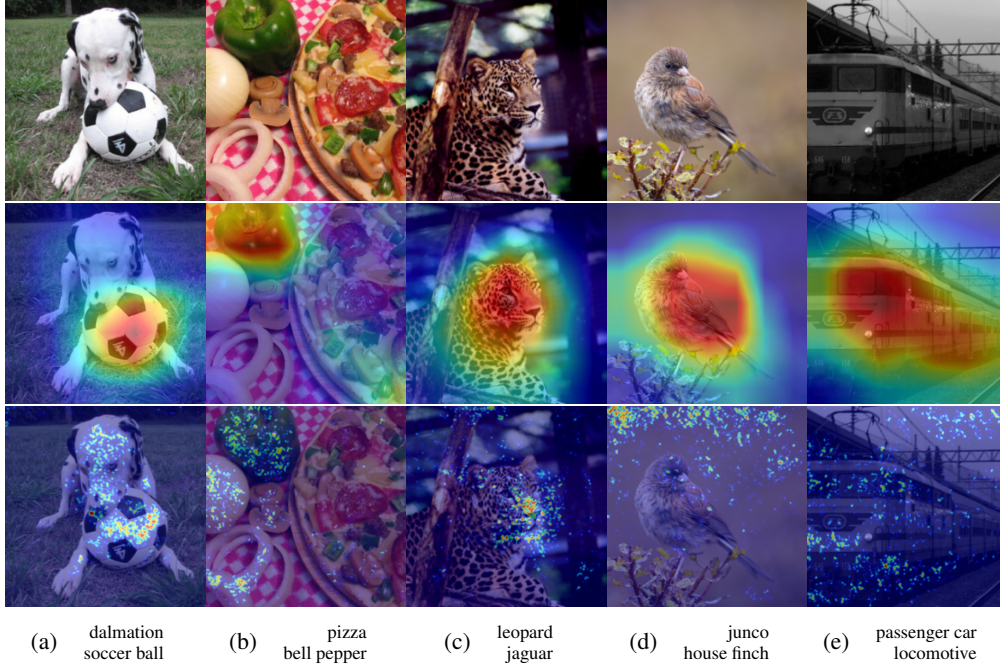



Figure 22: **Comparison to GradCAM.** Top row: original image. Middle row: GradCAM input-space saliency map for the predicted label. Bottom row: our input-space saliency technique which highlights pixels that drive high parameter saliency values of specific filters (i.e., pixels that confuse the network). The correct class label is specified in the top row and the predicted incorrect class label – in the bottom row of the subcaption on each panel.

A.11 Input-saliency sanity check

To assure that our input-space saliency technique is model dependent, we performed the model randomization test from [2]. We can see that the input saliency map is model dependent. We note that the data randomization test is not applicable in our case because our input-space saliency map is based on the parameter-saliency profile and parameter-saliency is designed to investigate a pretrained model with particular weights.

B Implementation details

B.1 Hyper-parameter setting for fine-tuning salient filters

When tuning a small number of random or least salient filters, we re-normalize the gradient magnitude of these parameters to be the same as the salient filters for a fair comparison; otherwise the gradients for these parameters are always smaller than the salient ones by the definition of our saliency profile, and updating them would make less change to a model than updating the salient ones. In addition to re-normalizing the gradients, we also multiply them with a step size, similar to the concept of learning rate in stochastic gradient descent. For ResNet-50, we set the step size to be 0.001, which equals to the learning rate of the last epoch when training the ResNet-50 on ImageNet from scratch. For VGG-19, we also set the fine-tuning step size to be the learning rate from the last training epoch, which is 0.0001. We also note that the batch normalization layers were set to the test mode for our fine-tuning experiments.

24

(a)	malamute husky	(b)	stingray electric ray	(c)	appenzeller sennenhund	(d)	redbone beagle	(e)	great white shark killer whale

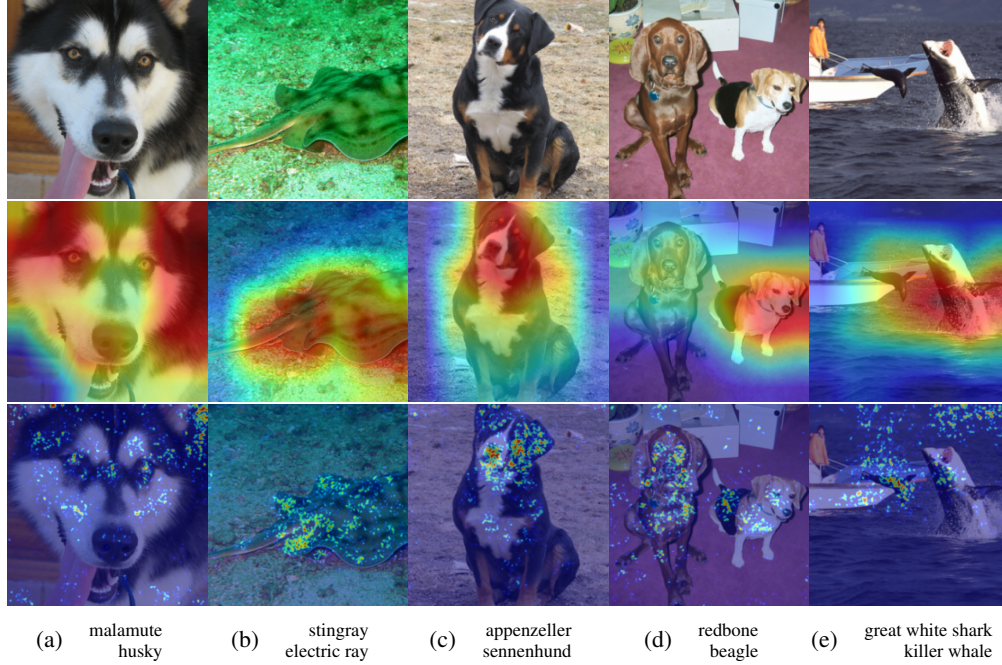



Figure 23: **Comparison to GradCAM.** Top row: original image. Middle row: GradCAM input-space saliency map for the predicted label. Bottom row: our input-space saliency technique which highlights pixels that drive high parameter saliency values of specific filters (i.e., pixels that confuse the network). The correct class label is specified in the top row and the predicted incorrect class label – in the bottom row of the subcaption on each panel.

(a) Original   (b) Stage 4   (c) Stages 3-4   (d) Stages 2-4   (e) Stages 1-4

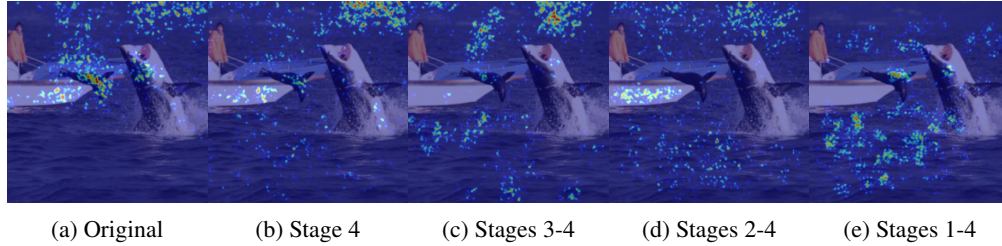



Figure 24: **Sanity checks.** (a) No randomization of ResNet-50. (b) Only stage 4 of ResNet-50 is randomized. (c) Stages 3-4 of ResNet-50 are randomized. (d) Stages 2-4 of ResNet-50 are randomized. (e) The entire ResNet-50 is randomized.

B.2 Input saliency visualization

The number of top salient filters to boost was chosen to be 10 in all input-space saliency experiments. The filters were boosted by multiplying by 100. For visualization, absolute input gradients were thresholded at 90-th percentile and Gaussian Blur with $(3, 3)$ kernel was applied.

B.3 Hardware

The experiments were run on Nvidia GeForce RTX 2080Ti GPUs with 11Gb GPU memory on a machine with 4 cpu cores and 64Gb RAM. The input-space and parameter-saliency profiles take seconds to compute for a single sample.

25

C   Limitations and Impact

Although our formulation of parameter saliency is not restricted to image datasets and CNNs, we only conduct experiments in these settings. In contrast, real-world data and models come in many forms. Explainability methods which shed light in some settings may fail to do so in others. Moreover, we emphasize that some erroneous model behaviors are simply difficult to understand through existing methods, and the capabilities of parameter saliency are limited. In many applications, it is imperative that practitioners understand why their models behave as they do and that they are able to diagnose problems when they arise. We hope that our work helps to enable solutions to real-world problems. However, we caution against a false sense of security. Our visual interpretations of model behavior should be viewed as approximations as neural networks are incredibly complex. Additionally, a future direction to explore with our method is leveraging parameter saliency and targeted fine-tuning for improving the overall model accuracy. While in our current targeted tuning experiments, we see new errors introduced, some errors might have higher cost than others in applications where machine learning models are expected to conform to certain guardrails (e.g. for correcting socially inappropriate model behavior).

26