WHERE DO MODELS GO WRONG? PARAMETER-SPACE SALIENCY MAPS FOR EXPLAINABILITY

Roman Levin [*1], **Manli Shu**[*2], **Eitan Borgnia**[*2], **Furong Huang**[2], **Micah Goldblum**[2], **Tom Goldstein**[2]
`rilevin@uw.edu, manlis@umd.edu, eborgnia2@gmail.com`

ABSTRACT

Conventional approaches to explaining the decisions of neural networks for image data focus on the use of saliency maps, which highlight important inputs to which predictions are highly sensitive. However, these approaches largely ignore the inner workings of neural networks and the mechanisms responsible for particular inference outcomes. In this work, we consider saliency methods which attend to the network parameters rather than inputs. These methods allow the user to explore how image features cause specific network components to malfunction and query the dataset for other images leading to similar erroneous behavior.

1 INTRODUCTION

With the widespread deployment of deep neural networks in high-stakes applications such as medical imaging (Kang et al., 2017), credit score assignment (West, 2000), and facial recognition (Deng et al., 2019), practitioners need to understand why their models make the decisions they do. In fact, "right to explanation" legislation in the European Union and the United States dictates that relevant public and private organizations must be able to justify the decisions their algorithms make (United States Congress Senate Committee on Banking and Housing and Urban Affairs, 1976; European Commission, 2018). Diagnosing the causes of system failures is particularly crucial for understanding the flaws and limitations of models we intend to employ.

Typical interpretability methods focus on highlighting sensitive pixels (Simonyan et al., 2014) or image regions which maximize specific activations (Erhan et al., 2009). These methods are useful for detecting edges and high order features as well. But since these methods operate in input space, they do not discern salient model parameters. Moreover, these popular methods highlight areas to which a model is sensitive rather than areas which specifically cause bad performance. These two types of regions may not coincide since regions which trigger misclassification may not in fact be the areas with the highest loss gradient magnitudes.

We develop an alternative approach for explainability which locates the network parameters that influence specific incorrect predictions. To this end, we compute saliency maps in parameter space. These maps yield a number of useful analyses:

- Nearest neighbors in parameter saliency space share common semantic information which causes the same network components to malfunction.
- Our method enables visualizing image regions that cause salient parameters to misbehave.
- Fine tuning salient filters can correct specific types of misbehavior without degrading overall performance.

After carefully delineating our methodology, we showcase the practical utility of this paradigm with a case study in which we are able to uncover a neural network's reliance on a spurious correlation which causes interpretable failures.

[*]Equal contribution.
[1]Department of Applied Mathematics, University of Washington
[2]Department of Computer Science, University of Maryland

1.1 Related work

Many methods analyze the behavior of CNNs by visualizing the semantics of individual convolutional filters (Simonyan et al., 2014; Zeiler & Fergus, 2014) or intermediate feature representations inside CNNs (Mahendran & Vedaldi, 2015). Another line of work studies the input-output relationship (Fong & Vedaldi, 2017; Wang et al., 2020) or feature-output relationship (Shrikumar et al., 2017; Sundararajan et al., 2017). Saliency methods typically work in this manner, which involves computing the loss gradient of classifiers *w.r.t* inputs. (Erhan et al., 2009; Yosinski et al., 2015).

Guided backpropagation (Springenberg et al., 2015) aims to enhance the highlighted features in saliency maps by zeroing negative gradient entries when backpropagating through non-linear activations. Guided GradCam (Selvaraju et al., 2017) builds on this approach by also computing the gradient of logits with respect to only the final convolutional layer.

Although extensive work has been done to study how different regions of *images* affect a network's prediction, limited work aims to distinguish important parts of the network *parameters*. Our work evaluates saliency directly on model parameters by aggregating their absolute gradients on a filter-by-filter basis and leverages the resulting saliency profiles as an explainability tool. A similar idea of measuring parameter importance has been adopted for network pruning (Liu & Wu, 2019) while we focus on model explainability.

2 Method

2.1 Parameter saliency profile

Let x be an image in the validation set D with label y, and suppose a trained classifier has parameters θ which minimize a loss function $\mathcal{L}$. The parameter-wise saliency profile of x is defined as a vector $s(x, y)$ with entries $s(x, y)_i := |\nabla_{\theta_i} \mathcal{L}_\theta(x, y)|$, the magnitudes of the gradient of loss with respect to each model parameter. Because the gradients on training data for a model trained to convergence are essentially zero, it is important to specify that D be a validation set.

Distinct convolutional filters are known to be responsible for specific tasks such as edge, shape, and texture detection (Yosinski et al., 2015). We therefore choose to aggregate saliency on a filter-by-filter basis by averaging the parameter gradients corresponding to each convolutional filter. This allows us to isolate filters most responsible for high loss values (i.e. for erroneous behavior).

Because some parts of the network can have larger gradients across all validation images, we further distill the filter-wise saliency profile of x by standardizing with respect to all filter-wise saliency profiles of D. This relative saliency profile allows us to isolate filters that are abnormally salient for a particular image and not for other images.

2.2 Saliency profile inversion and visualization of salient filters

Highly salient filters have greater gradient magnitudes and therefore need to be changed the most to decrease the loss and, in the case of misclassified images, to correct the misclassification. In that sense, the saliency profile can be considered an error profile which allows us to identify filters that are most responsible for mistakes. To understand which parts of the image affect the saliency of particular filters, we develop a method to invert the saliency profile, taking an approach similar to the gradient inversion from Geiping et al. (2020).

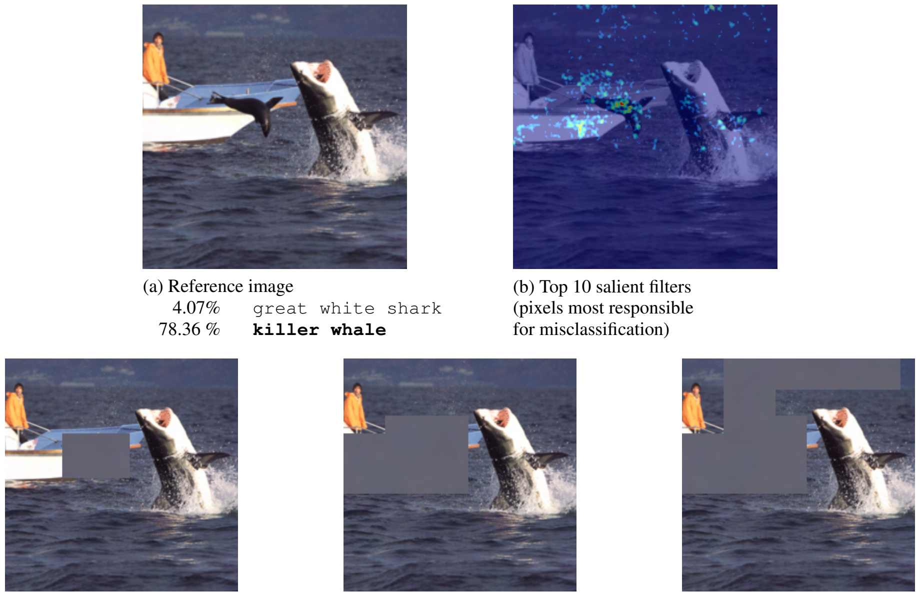
Given a saliency profile $s^* = s(x^*, y)$ for an image x^* with label y, we can recover the image x^* by solving the saliency inversion optimization problem $\arg\min_{x \in [0,1]^n} D_C(s(x, y), s^*)$, where $D_C(\cdot, \cdot)$ is the saliency matching loss measured as cosine distance and the constraint $x \in [0, 1]^n$ arises from the fact that x is an image.

We propose a visualization technique that highlights image features which cause filters to have high saliency. The core idea of our method is to manipulate the target saliency profile and further increase the entries corresponding to the most salient filters thus obtaining a boosted saliency profile s'. If we start the saliency inversion from the original image x^* and try to perturb it to increase the saliency of those filters and match the boosted saliency profile s', then the parts of the image that change first would show us what in the original image makes the filters more salient. These regions are captured by the gradients of the saliency matching loss with respect to the image $\nabla_x D_C(s(x,y), s')$, and we use them for our visualizations (see Figure 1).

3 EXPERIMENTS

We evaluate our saliency method in the context of image classification on CIFAR-10 (Krizhevsky, 2009) and ImageNet (Deng et al., 2009). Images we use for visualization, unless otherwise specified, are sampled from ImageNet validation set. Throughout the experiments, we use pre-trained ResNet-18 (He et al., 2016) as the classifier for CIFAR-10 and pre-trained ResNet-50 for ImageNet.

3.1 CASE STUDY



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

(b) Top 10 salient filters (pixels most responsible for misclassification)

(c) Mask seal
11.83% `great white shark`
53.47% **`killer whale`**

(d) Mask boat and seal
43.93% **`great white shark`**
24.24% `killer whale`

(e) Extended mask
58.73% **`great white shark`**
12.88% `killer whale`

Figure 1: **Filter saliency visualization for a misclassified image and masking experiments.** (a) Reference image of "great white shark" misclassified by the model as "killer whale" with corresponding confidence scores. (b) Pixels that cause the top 10 most salient filters to have high saliency. (c)-(e) Masking experiments.

We begin by introducing a case study where we investigate the mechanism by which a classifier misclassifies an image. To this end, we first identify filters most responsible for the mistake by computing the saliency profile, and then we visualize parts of the image that drive the high saliency values for those filters (as described in Section 2.2).

The image we study (see Figure 1(a)) is labeled "great white shark" but is misclassified as "killer whale". Panel (b) of Figure 1 presents our image-space visualization of the ten most salient filters – the pixels that trigger misbehavior in these filters are highlighted. The seal and boat are both triggers.

One natural hypothesis is that the seal looks like a killer whale to the network and is the source of the classification error. We test this by masking out the seal (see Figure 1 (c)) . However, although the probability of "killer whale" goes down and the probablity of the correct class increases, the network still misclassifies the image as "killer whale".

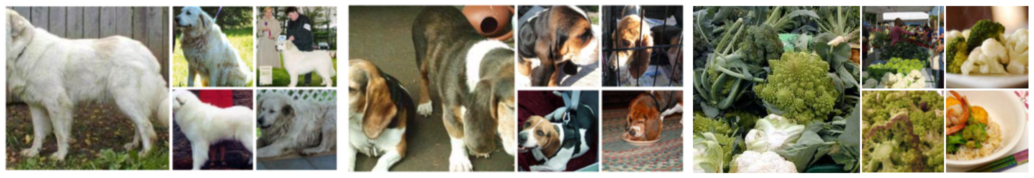
Now, if we mask out exactly the most salient areas of the image according to our visualization (see Figure 1 (b), (d)), the network manages to flip the label of the image and classify it correctly. If we extend our mask to the less pronounced salient areas of the image as in Figure 1 (e), we observe that the correct class confidence increases even more while the probability of the incorrect "killer whale" label decreases further. Additionally, we find that masking out the non-salient parts of the image like the human or the non-salient water regions (see Figure 3 in Appendix B) results in even worse misclassification confidence than that of the original image.

In order to further extend the effect of the salient region, we paste it from this image onto other great white shark images (see Figure 4 in Appendix B) and observe that it drives the probability of "killer whale" up for 39 out of 40 examples of great white sharks from the ImageNet validation set with an average increase of 3.75%. In one case, this caused a misclassification as "killer whale."

Our experiments suggest that secondary objects in the image are associated with the misclassification. However, we see that the erroneous behavior of the model does not just stem from classifying a non-target object in the image. It is possible that the model correlates the combination of sea creatures (e.g. a seal or a killer whale) and man-made structures (e.g. a boat) with the "killer whale" label. We note that images of killer whales in ImageNet do have man-made structures which look similar to the boat (see Figure 5 in Appendix B).

3.2 NEAREST NEIGHBORS IN PARAMETER SALIENCY SPACE

We validate the semantic meaning of our saliency profiles by clustering images based on cosine similarity of their profiles. We find the nearest neighbors of misclassified images in saliency space to mostly be images misclassified for the same reason. e.g. The reference image of Figure 2 (a) is a Great Pyrenees misclassified as Kuvasz, and the 4 images with the most similar profiles exhibit either the same misclassification or the reverse. In addition, we find nearest neighbors in saliency space are not necessarily similar in pixel space (see Figure 2 (b) (c)) but share common semantic information. More examples can be found in Appendix C.



(a) great pyrenees $\leftrightarrow$ kuvasz (b) basset $\leftrightarrow$ beagle (c) broccoli $\leftrightarrow$ cauliflower

Figure 2: **Example of nearest neighbors in parameter saliency space for ImageNet**.

3.3 FINE-TUNING ON MISCLASSIFIED IMAGES

With our method to identify salient filters, we can fine-tune salient filters to correct specific types of misbehavior without sacrificing the overall performance of a model. For fine-tuning, we sample misclassified images from ImageNet training set and CIFAR-10 test set respectively.

In Table 1, we study three misclassified samples independently for each dataset. We fine-tune on a given sample and update the tunable filters for a single iteration with a fixed step size for both CIFAR-10 and ImageNet. We also evaluate other choices of tunable filters besides the most salient ones. For evaluating random filters, we take the average of 10 runs. When tuning all filters, we adjust the step size to mitigate the impact of big step size on much more parameters. See Appendix D for details of our experiment settings.

Table 1: **Performance of models fine-tuned on a single misclassified samples**. $\mathcal{K}$ denotes the number of nearest neighbors (including the sample itself) being corrected after fine-tune. $\Delta\mathfrak{A}$ is the change of validation accuracy after fine-tune.

(a) on CIFAR-10 misclassified images.

Tunable filters	sample #1		sample #2		sample #3	
	$\mathcal{K}$	$\Delta\mathfrak{A}(\%)$	$\mathcal{K}$	$\Delta\mathfrak{A}(\%)$	$\mathcal{K}$	$\Delta\mathfrak{A}(\%)$
All filters	5	-0.20	4	-0.05	5	-0.23
Least salient	0	0	0	0	0	0
Random salient	0.1	-0.01±.02	1	+0.01±.01	0	0±.02
Top salient	4	**+0.03**	4	**+0.03**	5	**+0.06**

(b) on ImageNet misclassified images.

Tunable filters	sample #1		sample #2		sample #3	
	$\mathcal{K}$	$\Delta\mathfrak{A}(\%)$	$\mathcal{K}$	$\Delta\mathfrak{A}(\%)$	$\mathcal{K}$	$\Delta\mathfrak{A}(\%)$
All filters	3	-0.08	4	-0.21	5	-0.08
Least salient	0	0	0	-0.01	0	0
Random salient	0	0±.01	0	-0.03±.07	0.9	0±.01
Top salient	3	**+0.02**	4	**+0.01**	4	**+0.01**

By fine-tuning only 0.4% of the filters on a single misclassified image, we are able to correct as many of its nearest neighbors as we do when updating the entire network. In addition, our strategy of effectively updating a minimal set of the parameters prevents models from over-fitting to specific samples and maintains the overall accuracy.

REFERENCES

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4690–4699, 2019.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

European Commission. *Recital 71 EU General Data Protection Regulation*. 2018. URL `https://www.privacy-regulation.eu/en/r71.htm`.

Ruth C. Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *ICCV*, 2017.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients–how easy is it to break privacy in federated learning? *arXiv preprint arXiv:2003.14053*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Eunhee Kang, Junhong Min, and Jong Chul Ye. A deep convolutional neural network using directional wavelets for low-dose x-ray ct reconstruction. *Medical physics*, 44(10):e360–e375, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Congcong Liu and Huaming Wu. Channel pruning based on mean gradient for accelerating convolutional neural networks. *Signal Processing*, 156:84–91, 2019.

Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *CVPR*, 2015.

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 *ICCV*, 2017.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *ICML*, 2017.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR*, 2014.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR*, 2015.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *ICML*, 2017.

United States Congress Senate Committee on Banking and Housing and Urban Affairs. *Equal Credit Opportunity Act. [electronic resource]*. S. Rpt. 94-685. Washington, 1976.

Haofan Wang, Zifan Wang, Mengnan Du, Fan Yang, Zijian Zhang, Sirui Ding, Piotr Mardziel, and Xia Hu. Score-cam: Score-weighted visual explanations for convolutional neural networks. In *CVPR*, 2020.

David West. Neural network credit scoring models. *Computers & Operations Research*, 27(11-12): 1131–1152, 2000.

Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

A Mathematical details of the method

A.1 Aggregation & standardization

Formally, each convolutional filter $\mathcal{F}_k$ in the network has a corresponding index set α_k, which gives the indices of the parameters relevant to $\mathcal{F}_k$. The filter-wise saliency profile of x is defined to be a vector $\bar{s}(x, y)$ with entries

$$\bar{s}(x,y)_k := \frac{1}{|\alpha_k|} \sum_{i \in \alpha_k} s(x,y)_i,$$

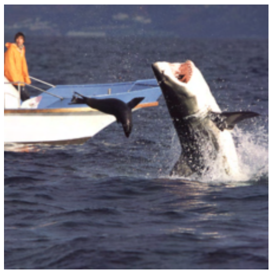
the parameter-wise saliency profile aggregated by averaging on a filter-by-filter basis. We may choose to use the coarser layer-by-layer aggregation in a similar way, where we instead choose an index set β_k, corresponding to the parameters relevant to the entire k^{th} layer.

Let μ be the average filter-wise saliency profile across all $x \in D$ and let σ be a equal-length vector with the corresponding standard deviation for each entry. We use these statistics to produce the standardized filter-wise saliency profile as follows:

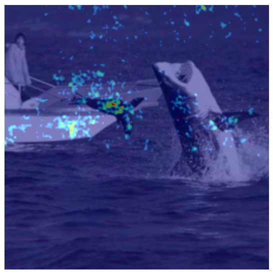
$$\hat{s}(x,y) := \frac{|\bar{s}(x,y) - \mu|}{\sigma}$$

We can standardize the layer-wise saliency profile by taking μ and σ to be the average and standard deviation for layer-wise saliency profiles across all elements of D.

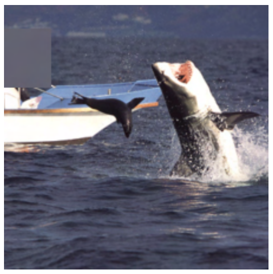
B Additional case study figures



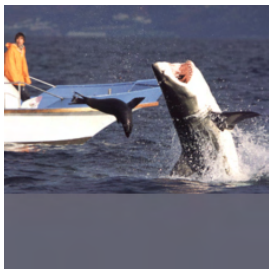
(a) Reference image
4.07% `great white shark`
78.36 % **`killer whale`**



(b) 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 3: **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 human. (d) Masked non-salient water region.

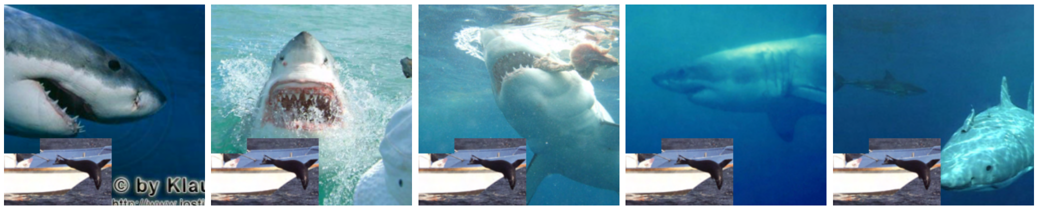


Figure 4: **Sample shark images with boat and seal.** The salient region from the case study image pasted onto other "great white shark" images.

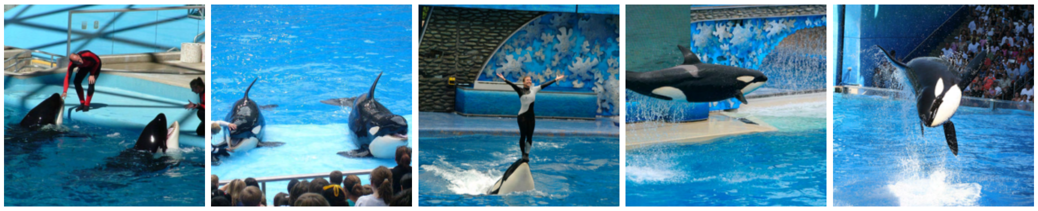
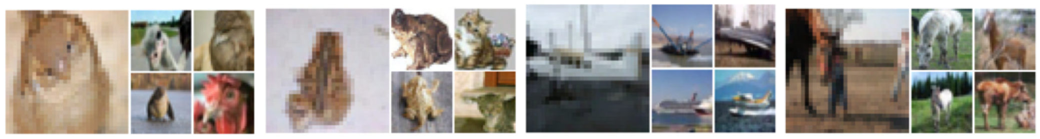


Figure 5: **ImageNet examples of "killer whale"**

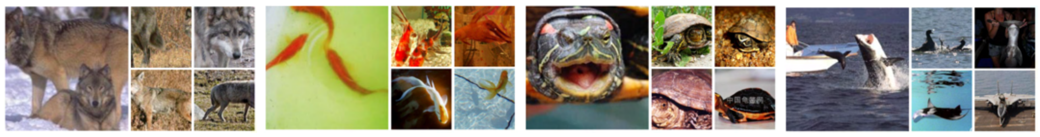
C More examples of nearest neighbors

We present more examples of nearest neighbors in our parameter saliency space. Figure 6 are nearest neighbors in CIFAR-10 dataset, where reference images are chosen from samples misclassified by our classifier. Figure 7 are examples from ImageNet, where images are captioned with the true label of the reference images.



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

Figure 6: **CIFAR-10 examples of nearest neighbors in parameter saliency space**.

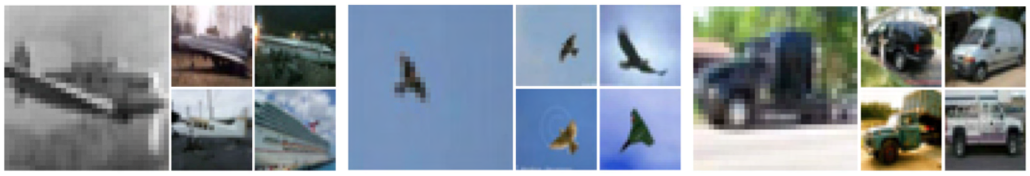


(a) coyote (b) goldfish (c) terrapin (d) great white shark

Figure 7: **ImageNet examples of nearest neighbors in parameter saliency space**. Samples misclassified images are captioned with their true labels.

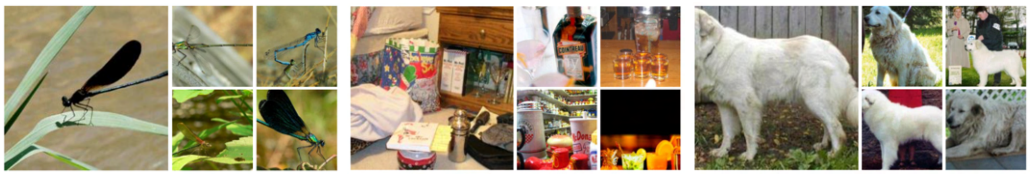
D Details of the fine-tuning experiments

Choice of samples. For fine-tuning, we choose samples that have most of their nearest neighbors (in saliency space) falling into the same category. As we introduced in Section 3.2, nearest neighbors in saliency space cause the network to malfunction in similar ways. On CIFAR-10, this resemblance reflects on the identical categories (both true labels and models' predictions) of the nearest neighbors. However, ImageNet has many more categories and some that are very close to each other semantically, so the nearest neighbors in saliency space are not always in the same category (but are in semantically similar categories). Therefore, we add this restriction when sampling ImageNet images, to select samples that are more representative of class-specific mistakes made by a classifier. Samples we use for fine-tuning in Section 3.3 are shown in Figure 8 and 9.



(a) airplane $\leftrightarrow$ ship (b) bird $\leftrightarrow$ airplane (c) truck $\leftrightarrow$ automobile

Figure 8: **CIFAR-10 Samples used for fine-tuning and their nearest neighbors in parameter saliency space**.



(a) dragonfly (b) cocktail shaker (c) Great Pyrenees

Figure 9: **ImageNet Samples used for fine-tuning and their nearest neighbors in parameter saliency space**. Samples are misclassified images captioned with their true labels.

Choice of hyper-parameters. We update models with step size of 1×10^{-3}, which is the learning rate of the last epoch for pre-trained models. Because we are only fine-tuning the models with a single sample image, we find this step size to be too large when fine-tuning all filters. Table 2 shows how the step size affect the performance for this case. We use the results with step size $= 1 \times 10^{-4}$ in Table 1, as this corrects just as many samples after fine-tuning without significantly hurting validation accuracy.

Table 2: **Fine-tuning all filters with different step sizes**. $\mathcal{K}$ denotes the number of nearest neighbors (including the sample itself) corrected after fine-tuning. $\Delta\mathfrak{A}$ is the change of validation accuracy after fine-tuning.

(a) on CIFAR-10 misclassified images.

Step size	sample #1		sample #2		sample #3	
	$\mathcal{K}$	$\Delta\mathfrak{A}(\%)$	$\mathcal{K}$	$\Delta\mathfrak{A}(\%)$	$\mathcal{K}$	$\Delta\mathfrak{A}(\%)$
1×10^{-3}	7	-38.34	5	-15.19	5	-57.84
1×10^{-4}	5	-0.20	4	-0.05	5	-0.23
5×10^{-5}	4	-0.03	4	-0.02	3	-0.15

(b) on ImageNet misclassified images.

Step size	sample #1		sample #2		sample #3	
	$\mathcal{K}$	$\Delta\mathfrak{A}(\%)$	$\mathcal{K}$	$\Delta\mathfrak{A}(\%)$	$\mathcal{K}$	$\Delta\mathfrak{A}(\%)$
1×10^{-3}	8	-1.52	7	-53.45	6	-6.78
1×10^{-4}	3	-0.08	4	-0.21	5	-0.08
5×10^{-5}	1	-0.04	3	-0.07	4	-0.01

We choose to use just one iteration for simplicity. Empirically, we find that even after updating the model just once, sample images can be effectively corrected along with some nearest neighbors.