COSE: A Consistency-Sensitivity Metric for Saliency on Image Classification

Rangel Daroya* Aaron Sun* Subhransu Maji
University of Massachusetts Amherst
{rdaroya, aaronsun}@umass.edu, smaji@cs.umass.edu

A. Additional Analysis

A.1. GradCAM can reflect changing model accuracy from transformed data

Figure 5 showed how different saliency methods respond to varying model accuracy as a response to changing model weights. Here we explore the response of saliency methods on the model accuracy across different data transformations. Given an input x and its transformed counterpart $t(x)$, saliency methods should produce two corresponding maps M and M'. If a model f classifies $f(x) \not\equiv f(t(x))$, then the SSIM between M and M' should be low (sensitivity metric). On the other hand if $f(x) \equiv f(t(x))$, then the SSIM between M and M' should be high (consistency metric).

Given the above, we should see an increasing trend between model accuracy and SSIM. Figure A.1 shows GradCAM and LIME with a high SSIM for high model accuracy, and low SSIM for low model accuracy. The slope is also noticeably steeper for GradCAM than LIME. This indicates the saliency methods can capture the behavior of a model to changing inputs.

A.2. Saliency methods perform similarly for supervised and unsupervised networks

Exploring the implications of properties found in self-supervised transformers, we looked at the difference in explanations between supervised and unsupervised models [1]. We speculated the explanations of unsupervised transformer networks may perform better than their supervised counterparts since the former was explicitly trained to find common patterns within classes. However, we did not observe significant performance differences between these two categories of networks.

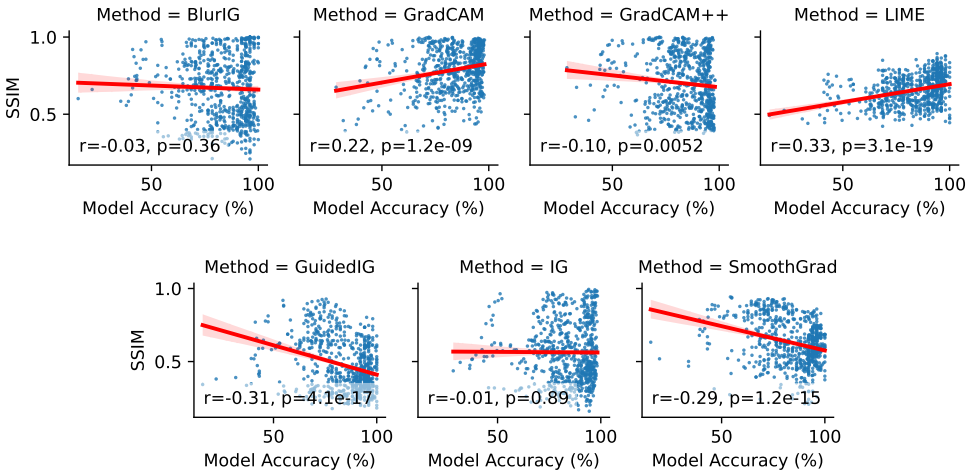

Figure A.1. **SSIM of saliency maps M and M' for varying model accuracy**. M is the saliency map for an input image x, and M' is the saliency map for a transformed input $t(x)$. The correlation (r) and the corresponding p-values (p) are also annotated in each plot. GradCAM and LIME show significant positive correlation between model accuracy and SSIM for various data transformations.

*Equal contribution

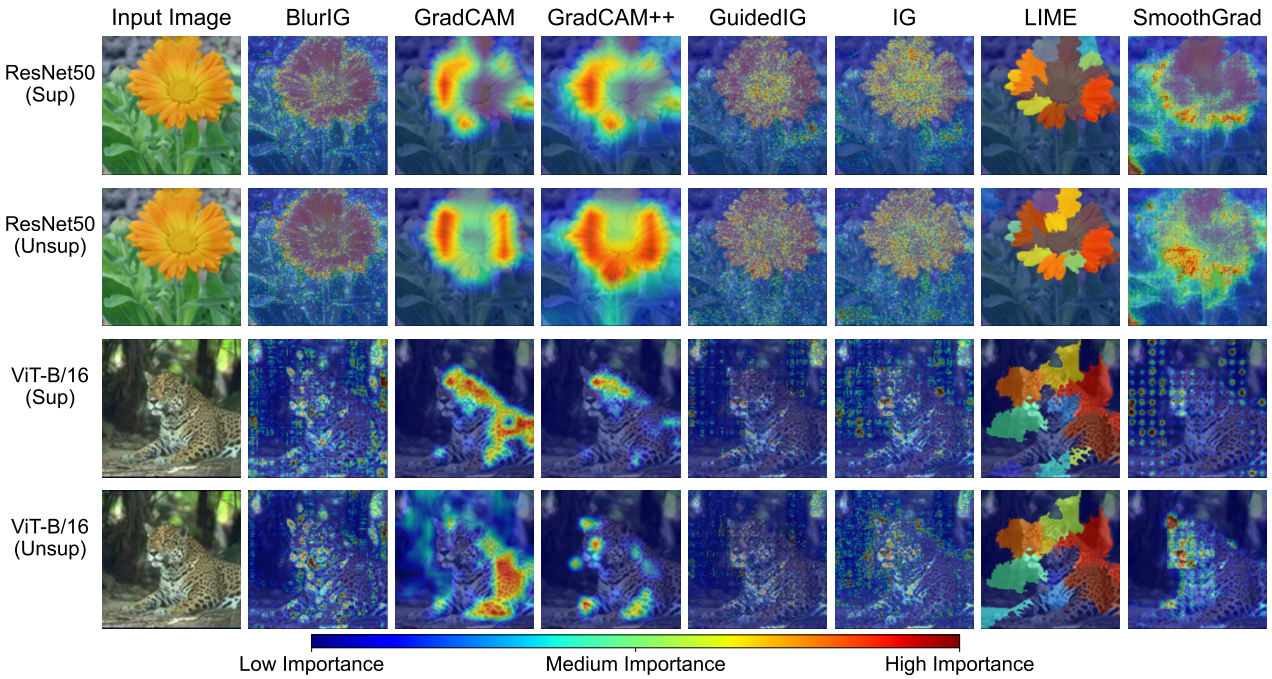

Figure A.2. **Results of saliency methods across supervised (Sup) and unsupervised (Unsup) ResNet50 and ViT-B/16 models**. We qualitatively observe that there is no significant difference between the saliency maps of the supervised and the unsupervised models.

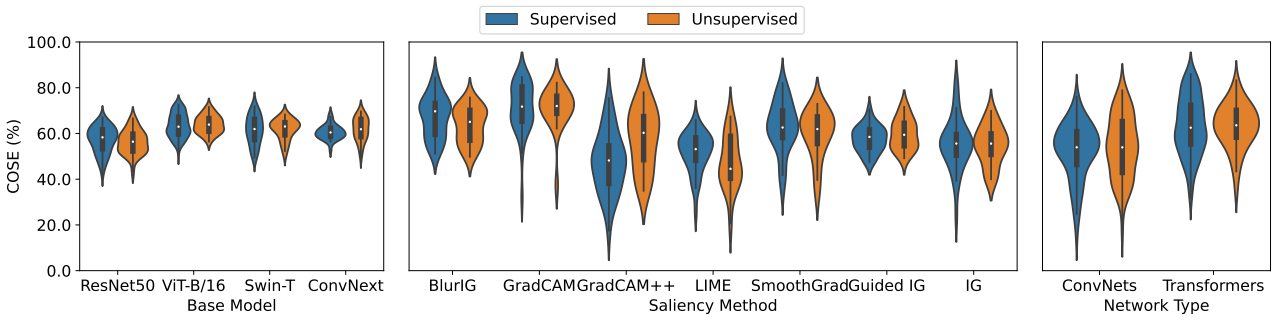

Figure A.3. **The average COSE of supervised and unsupervised models.** From left to right, COSE is compared for different base models, saliency methods, and network types, respectively. We see comparable performance between supervised and unsupervised models – variations don't strongly favor a single direction. Most notably, there is little difference in average performance between supervised and unsupervised transformers.

Figure A.2 show qualitative results for comparing saliency maps of supervised and unsupervised model. We observe that although there are variations between saliency maps, the differences are not notable. Figure A.3 supports this further by showing the COSE of supervised and unsupervised networks for ViT-B/16 and ResNet50. Looking at the violin plot, supervised and unsupervised networks have similar performance. Although we have previously found that model architecture plays a role in the quality of saliency maps (transformers were found to have better explanations than ConvNets), there is no significant difference when it comes to the type of pre-training.

Most saliency methods seem to have comparable performance on average for supervised and unsupervised models, but we also observe unsupervised methods having higher COSE for BlurIG and Guided IG but a lower COSE for GradCAM++. Further investigation may be required into the individual methods to draw stronger conclusions about this result.

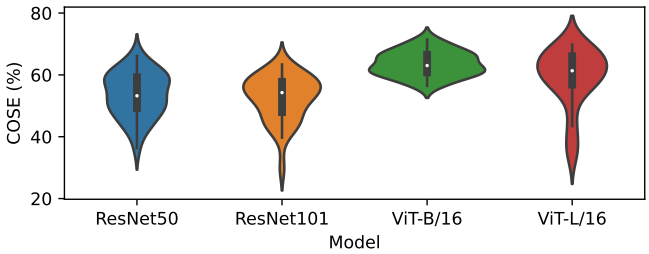

Figure A.4. **Average COSE of variations of Transformers and ConvNets.** The average performance of saliency methods on ResNet50 vs ResNet101 and ViT-B/16 vs ViT-L/16 are similar.

Saliency Method	**SSIM** COSE (%)	**Spearman** COSE (%)
BlurIG	63.23%	62.91%
GradCAM	64.66%	77.84%
GradCAM++	54.59%	56.41%
GuidedIG	54.73%	56.03%
IG	61.33%	59.60%
LIME	60.11%	66.88%
SmoothGrad	57.94%	58.65%

Table A.1. **Average COSE using SSIM and Spearman rank correlation**. The performance of the saliency methods is similar, and COSE remains the method with the highest COSE in both cases.

A.3. Saliency methods perform similarly on variations of transformers and convolutional networks

It was previously shown in Figures 3-4 that saliency methods perform better on Transformers than ConvNets. However, further analysis shows that when comparing performance between two different Transformer models where one could be larger than the other (e.g., ViT-B/16 vs ViT-L/16), the performance of saliency methods is similar. The same was also observed with ConvNets. Figure A.4 shows the distribution of the performance of saliency methods across different types of Transformers and ConvNets. This analysis can be further augmented with variations of models and architectures, which we encourage with our open-source pipeline.

A.4. Spearman rank correlation performs similarly to SSIM

Table A.1 compares average COSE results for each saliency methods when using SSIM and Spearman as the similarity metric. We found SSIM and Spearman rank correlation to provide very similar results, and our conclusion of GradCAM providing the best explanations under our metric to be strengthened when using Spearman rank correlation.

B. Additional Metric Visualizations

B.1. Sensitivity

Figure B.1 shows the sensitivity metric capturing changes in saliency maps $M \rightarrow M'$ due to model changes $f \rightarrow f'$. Higher sensitivity implies a larger difference between M and M' – a desired response. The figure thus shows whether or not saliency methods can represent changes in the model. BlurIG, GradCAM, Guided IG, IG, and SmoothGrad were observed to be responsive to model changes more than the other saliency methods.

B.2. Consistency

Figure B.2 shows additional consistency results on various datasets and data transformations. The figure shows that the consistency metric can capture the robustness of saliency methods across different changes in the input image.

C. Implementation Details

The different saliency methods were evaluated on various models and datasets. Details of the implementation are thoroughly described in this section.

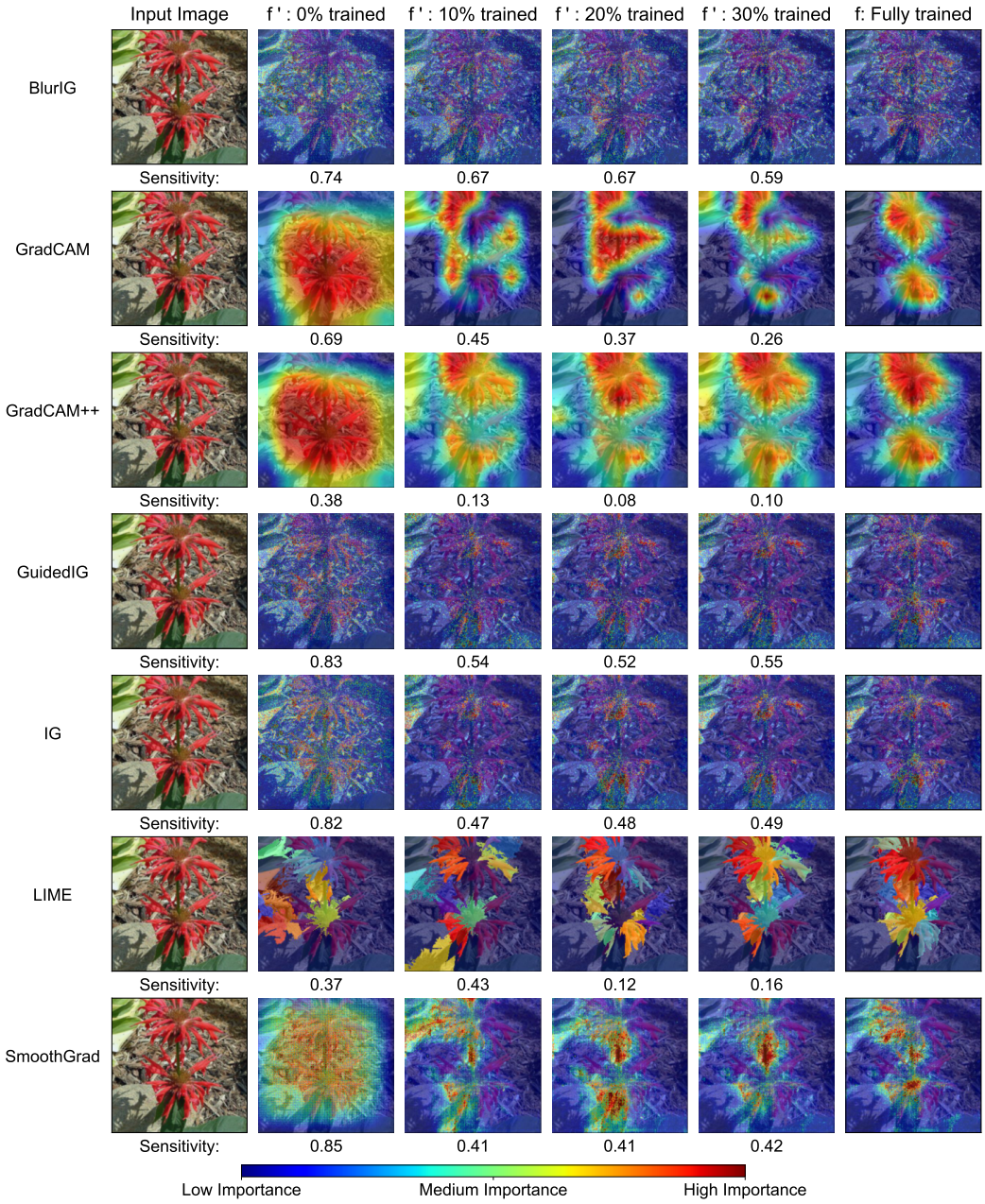

Figure B.1. **Sensitivity values and visualizations of saliency methods at various epochs during model training**. This shows sensitivity can capture how a saliency method represents a changing model: $f' \rightarrow f$ through the corresponding saliency maps $M \rightarrow M'$. The sensitivity (difference between M and M') should be high for saliency maps that respond to changes in the model.

C.1. Datasets

Dataset Train/Test Splits. For CIFAR and CUB, we used the train/test splits provided by the dataset creators. For Oxford Flowers, EuroSAT, and Caltech101, we used a fixed random split with 80% of the data in the training set and 20% of the data in the test set.

Data Augmentations. We modified the TrivialWideAugment [9] implementation in PyTorch to run data augmentations on images. In particular, we removed the transformations shear, posterize, and solarize and added the transformations flip (both left/right and up/down). We also the pillow modified the implementations of blur and smooth to use the corresponding

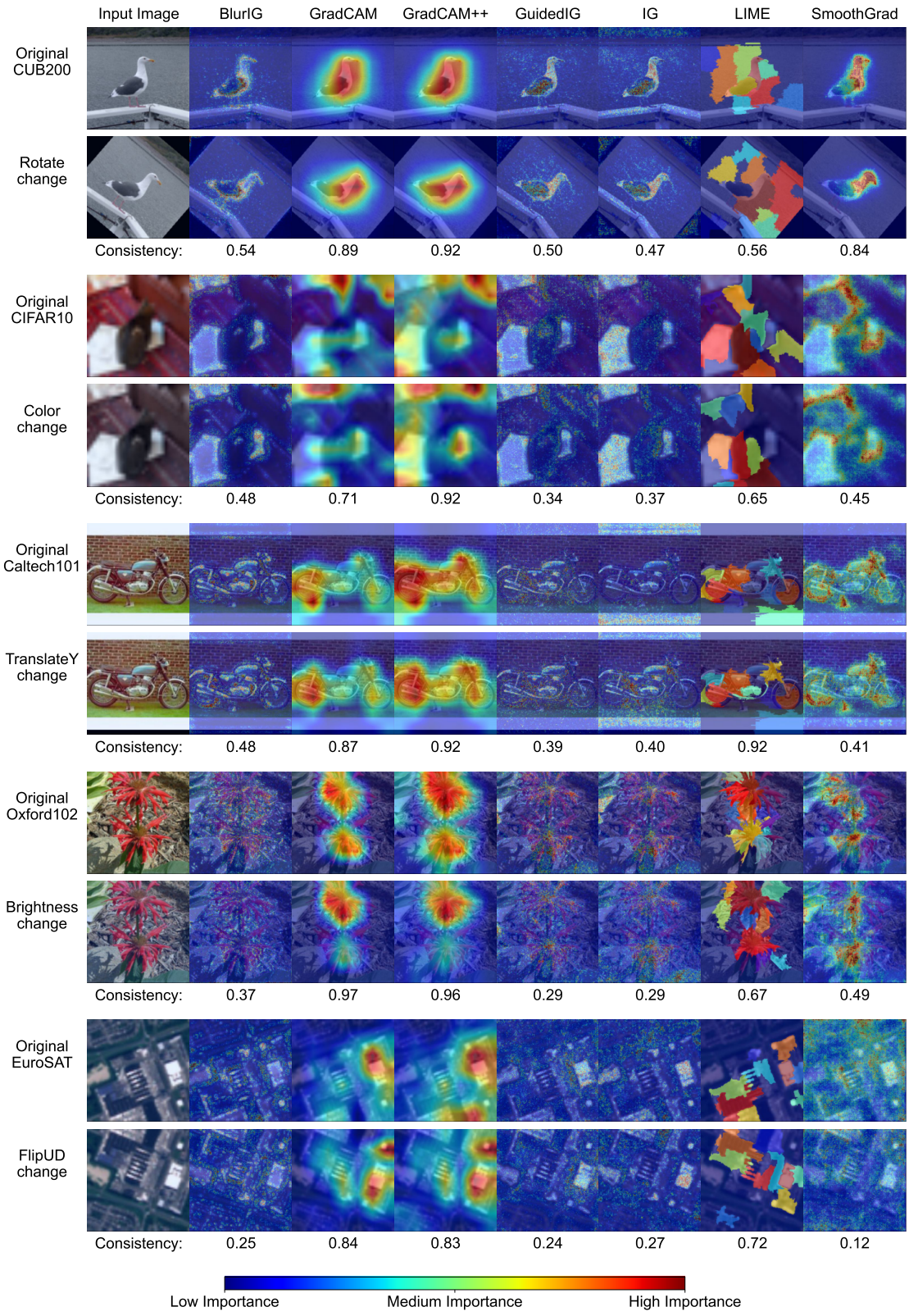

Figure B.2. **Consistency values and visualizations of saliency methods on various datasets and data transformations**. The saliency maps shown here are for inputs x and transformed inputs $t(x)$ where the model has correct predictions: $f(x) \equiv f(t(x))$. Reliable saliency methods should show similar saliency maps M (from x) and M' (from $t(x)$) through high consistency.

Transformation	**Magnitudes Applied** [min, max]	**Type**
FlipUD	-	Geometric
FlipLR	-	Geometric
TranslateX	[-32, 32]	Geometric
TranslateY	[-32, 32]	Geometric
Rotate	[-135^o, 135^o]	Geometric
Autocontrast	-	Photometric
Equalize	-	Photometric
Blur	-	Photometric
Smooth	-	Photometric
Brightness	[-0.99, 0.99]	Photometric
Color	[-0.99, 0.99]	Photometric
Contrast	[-0.99, 0.99]	Photometric
Sharpness	[-0.99, 0.99]	Photometric

Table C.1. **Data transformations used in training and evaluations.** Transformations are classified as either being photometric or geometric. Rows without magnitudes are transformations that are binary in nature (either they are applied or they are not).

	Pre-training Method	Caltech101 [12]	CIFAR10 [6]	CUB200 [13]	EuroSAT [5]	Oxford102 [10]
ResNet50 [4]	Supervised	94.00%	85.00%	81.00%	94.00%	82.00%
DINO ResNet50 [1]	Unsupervised	93.00%	84.00%	73.00%	96.00%	85.00%
MoCov3 ResNet50 [2]	Unsupervised	96.00%	90.00%	72.00%	95.00%	83.00%
ConvNext [8]	Supervised	95.56%	93.94%	82.86%	95.37%	79.53%
SparK ConvNext [11]	Unsupervised	75.58%	78.27%	53.19%	87.20%	48.04%
ViT-B/16 [3]	Supervised	96.00%	95.00%	81.00%	96.00%	81.00%
DINO ViT-B/16 [1]	Unsupervised	97.00%	95.00%	75.00%	98.00%	91.00%
MoCov3 ViT-B/16 [2]	Unsupervised	91.00%	93.00%	76.67%	96.00%	78.00%
iBOT ViT-B/16 [14]	Unsupervised	96.00%	95.00%	73.00%	97.00%	89.00%
Swin-T [7]	Supervised	96.00%	92.00%	84.00%	96.00%	84.00%
iBOT Swin-T [14]	Unsupervised	96.00%	95.00%	80.27%	97.00%	88.00%
	Average Performance	**93.01%**	**90.12%**	**75.10%**	**95.16%**	**80.76%**

Table C.2. **The models used for evaluation across different datasets.** The overall accuracy for each dataset was at least 75%. Models with prefixes DINO, MoCov3, SparK, and iBOT were pretrained on ImageNet in an unsupervised way.

PyTorch version.

Table C.1 shows the range of values used for each transformation. For each augmentation with variable magnitude, we sampled 62 evenly-spaced numbers from the given range. We used the entire suite of transformations during training by randomly selecting a transformation type, and then randomly selecting a magnitude within the range (if applicable). Testing, on the other hand, involved sequentially applying all the transformation types. To speed up evaluation run time, we then randomly sampled 6 nonzero magnitudes for transformations with variable magnitudes.

C.2. Models

Table C.2 shows the test set accuracy of various models separately trained on five datasets. The models also vary across supervised and unsupervised pre-training methods. To optimize each model for the corresponding dataset, the training hyperparameters were varied for each dataset and each model. These are available in the open source code.

C.3. Saliency Methods

We used publicly-available implementations of each saliency method. For GradCAM and GradCAM++, we used the implementations found on the pytorch-grad-cam repository[1]. For LIME, we used the official repository[2] from the authors.

[1] https://github.com/jacobgil/pytorch-grad-cam

[2] https://github.com/marcotcr/lime

Saliency Method	# Samples/model/dataset	Tunable Parameters parameter: value
BlurIG	3,060	linear interpolation steps: 100
GradCAM	30,600	target layer: model-dependent
GradCAM++	30,600	target layer: model-dependent
Guided IG	30,600	linear interpolation steps: 200 max distance factor: 0.02 network parameter sampling: 0.25
IG	30,600	linear interpolation steps: 25
LIME	3,060	number of samples for estimation: 1,000
SmoothGrad	3,060	method: vanilla gradients number of samples: 50 standard deviation spread: 0.15

Table C.3. **The parameters used for each saliency method for evaluation.** For each method, #Samples/model/dataset indicates the number of data points from the test set for each possible model and dataset pair. This includes various data transformations. The tunable parameters indicate the settings recommended by the authors of each method.

Finally, for IG, Guided IG, BlurIG, and SmoothGrad, we used the implementation from the PAIR-code saliency repository[3].

Sampling of results. Based on the speed of each method, we ran consistency and sensitivity tests on either 3,060 or 30,600 data points of each dataset and model combination. Table C.3 enumerates the number of samples tested for each saliency method for each dataset/model pair. In other words, for all possible combinations of models and datasets, we randomly sampled wither 3,060 data points or 30,600 data points.

Saliency method parameters use recommended settings in their respective papers. For GradCAM and GradCAM++, we used the activations in the layers suggested in the pytorch-gradcam-repository for each network. For methods with tunable hyperparameters, we followed the recommendations of each respective paper. Table C.3 lists all the parameters set during evaluation and the corresponding values.

References

[1] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 1, 6

[2] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9640–9649, 2021. 6

[3] 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. *International Conference on Learning Representations (ICLR)*, 2021. 6

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[5] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. In *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, volume 12, pages 2217–2226. IEEE, 2019. 6

[6] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

[7] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 6

[8] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022. 6

[9] Samuel G Müller and Frank Hutter. Trivialaugment: Tuning-free yet state-of-the-art data augmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 774–782, 2021. 4

[10] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008. 6

[3] https://github.com/PAIR-code/saliency

[11] Keyu Tian, Yi Jiang, Qishuai Diao, Chen Lin, Liwei Wang, and Zehuan Yuan. Designing bert for convolutional networks: Sparse and hierarchical masked modeling. *arXiv:2301.03580*, 2023. 6

[12] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. 2010. 6

[13] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. Technical Report CNS-TR-201, Caltech, 2010. 6

[14] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. *International Conference on Learning Representations (ICLR)*, 2022. 6