Staying Consistent with GradCAM

Aaron Sun Peter Phan Phat Nguyen
University of Massachusetts Amherst
{aaronsun, pkphan, phatn}@umass.edu

Abstract

Explainable methods, such as GradCAM, are not always consistent across image transformations. They are oftentimes too broad to be a source of reliable explanations. We present an approach to train a Convolution Neural Network to simultaneously produce consistent explanations using saliency maps from post-hoc explanation methods as regularizers. This in turn creates more robust and confident explanations as well as predictions. We show that our method improves the consistency of this explainablity method without any significant accuracy tradeoff.

1. Introduction

Post-hoc explainability methods, such as GradCAM [3], have been used as supplements to aid understanding of model predictions. They can help diagnose failure cases in high-stakes applications such as medical diagnoses or bias identification. These methods typically try to identify the most salient or important areas of an image in its classification, in the form of a heatmap over the original image.

However, despite the prevalence of these methods in the application of deep learning, they remain relatively unexplored in terms of their accuracy and quality, and thus have not been applied during the training process. This can be attributed to the difficulty of attaining ground-truth explanations for any dataset as a result of the subjectivity of explanations in general - two entirely different explanations for the same image classification may both be equally valid. Typically, evaluations of these explanations involve perturbing salient areas of the input or the model and observing the difference in model classifications. One way to evaluate explanations is through their *consistency*, or their degree of equivariance under input transformations. For instance, given an image and a mirror of the same image, the model explanations should be flipped versions of one another.

We observed one work by Pillai et. al [2] where GradCAM was used in training in order to improve explanation consistency on the output model. In particular, they framed the problem through contrastive learning. By training on augmented inputs as positive examples and random images as negative examples, the authors induced sparse and spatially unique explanations of their images but sacrificed classification accuracy by 2%. We identified what we believe to be weaknesses with this method. Most notably, we believe that using contrastive learning is an unnecessary additional step to induce sparsity and regularization can achieve similar effects with much faster computation. In addition, we speculate that image explanation consistency can also be used as a form of regularization and can improve model accuracy in certain circumstances.

In this work, we examine using consistency as a training metric along with explanation sparsity as an effective way to achieve consistent explanations which look considerably different from baseline methods with minimal cost to accuracy. More concretely:

- We define two loss terms $\mathcal{L}_C$ and $\mathcal{L}_S$ which are differentiable and trainable metrics for model explanation consistency and sparsity, respectively.
- We train networks of various architectures on CUB-200 and Caltech101 using these loss functions and observe minimal effects on model accuracy while being able to improve model explanations.
- We study the effects of varying the weights of these two losses and speculate an entire spectrum of models which tradeoff model explanation sparsity and consistency while maintaining high accuracy.

2. Related Work

Explanation Methods: Early attempts to explain CNN-based image classification models were post-hoc approaches that generate heat maps highlighting important areas of the input image. Simonyan et al. (2013) [5] were the first to introduce gradient-based saliency maps to visually represent the influence of features on the classification model at the pixel level. Building on visual interpretability, Ziler and Fergus (2014) [9] introduced deconvolutional networks to project feature activations back to the input space, which Springenberg et al. (2014) [7] used to improve the clarity of the visualizations. Class Activation Maps proposed by Zhou et al. (2016) [10] intro-

duced a novel approach by modifying the CNN architecture to include a global average pooling layer. This implementation enabled direct visualization of regions relevant to specific classes, addressing the need for class-specific interpretability. GradCAM [4] was a direct extension of CAM. It retained the class-specific visualization advantage but removed the architectural constraints of CAM by eliminating the GAP layer. Instead, they proposed using the gradients flowing into the final convolutional layer, which made GradCAM more flexible and compatible with a broader range of CNN architectures.

Regularization with Explanation Methods: Recent works such as Pillai et al. (2022) [2] has explored using explanatory models for regularization. Their work introduces, Contrastive GradCAM Consistency (CGC), which enhances GradCAM's alignment with human priors, such as consistency across image transformations, by applying contrastive self-supervised learning to model interpretations. However, while their work achieves greater consistency in explanations, it comes with the trade-off of reduced classification accuracy by 2%. While work on regularization using saliency maps or explanations is relatively limited, This has been done in segmentation tasks for medical applications in a work called GradMask, which penalized the model during training by comparing the outputted saliency map with ground truth segmentations [6]. However, our work will not require additional ground truth information about the training data and can be easily applied to existing datasets and training algorithms. The concept of using explanation consistency has also been used to evaluate saliency methods, but not for training networks [1].

3. Method

3.1. Problem Formulation

We consider a classification setting in which we want to train a neural network to simultaneously produce interpretable, consistent explanations without a large trade off in classification accuracy. More concretely, given a dataset $D = \{(x_1, y_1), ..., (x_{|D|}, y_{|D|})\}$, where $x_i \in X = \mathbb{R}^{m \times n \times c}$ are input images and $y \in Y$ are class labels, we want to train a model $f : X \to Y$ such that $f(x_i) = y_i$ as much as possible and our saliency method Φ generates consistent and sparse maps $\Phi(f, x_i) = M_i$ where $M_i \in [0, 1]^{m \times n}$.

3.2. Optimization Objectives

Since our task is explicitly an optimization of various criteria, we proceed by defining a loss term for each metric separately and combine them all to form a single minimization objective. These loss terms will be a classification loss, a consistency loss, and a sparsity loss. Our final loss function will be the sum of these terms in which we will optimize. These are also summarized in Figure 1.

3.2.1 Classification Accuracy

The first metric which we seek to measure is the classification accuracy on our dataset D. This can easily be measured as

$$\text{accuracy} = \frac{1}{|D|} \sum_{i=1}^{|D|} 1(f(x_i) = y_i)$$

However, this objective is not continuous or differentiable and is difficult to optimize. Following common practice in model training, we use cross-entropy loss as our loss function, which can be defined on a data point (x_i, y_i) as:

$$\mathcal{L}_{CE} = -\log\left(\frac{e^{f(x_i)_{y_i}}}{\sum_j e^{f(x_i)_j}}\right), \tag{1}$$

where $e^{f(x_i)_j}$ corresponds to the network output probability for the j-th class for the data point x_i.

3.2.2 Explanation Consistency

We also want our model to have consistent explanations as defined in COSE by Daroya et. al [1]. The primary rationale behind defining explanation consistency lies in using data augmentations to verify explanations across a single datapoint. In particular, by training a model with a transformation $t \in T$, we expect that the model to become invariant to that transformation, meaning $f(x_i) = f(t(x_i))$. In a similar fashion, we also expect that the explanations for that model will be similarly equivariant on the same transformation. For instance, if a model attributes the classification of a particular image of a bird based on its wing patterning, we expect the model to attribute the classification in a spatially equivalent way on the flipped version of the image. For geometric transformations, we expect that the augmented version of the explanation for the original image should match the explanation of the augmented image, or

$$t(\Phi(f, x_i)) = \Phi(f, t(x_i)).$$

In this work, we only use geometric transformations of compositions of crops and flips for simplicity, but this can be extended to photometric transformations as well.

Given this intuition, we define a consistency loss $\mathcal{L}_C$ which punishes the model for producing inconsistent explanations across an image. We evaluate this loss on a data point (x_i, y_i) with saliency method Φ and transformation t as

$$\mathcal{L}_C = d\left(t(\Phi(f, x_i)), \Phi(f, t(x_i))\right), \tag{2}$$

where d is a distance function between two explanations. This is also visually explained in Figure 1. Following the method of COSE, we evaluate the distance of explanations

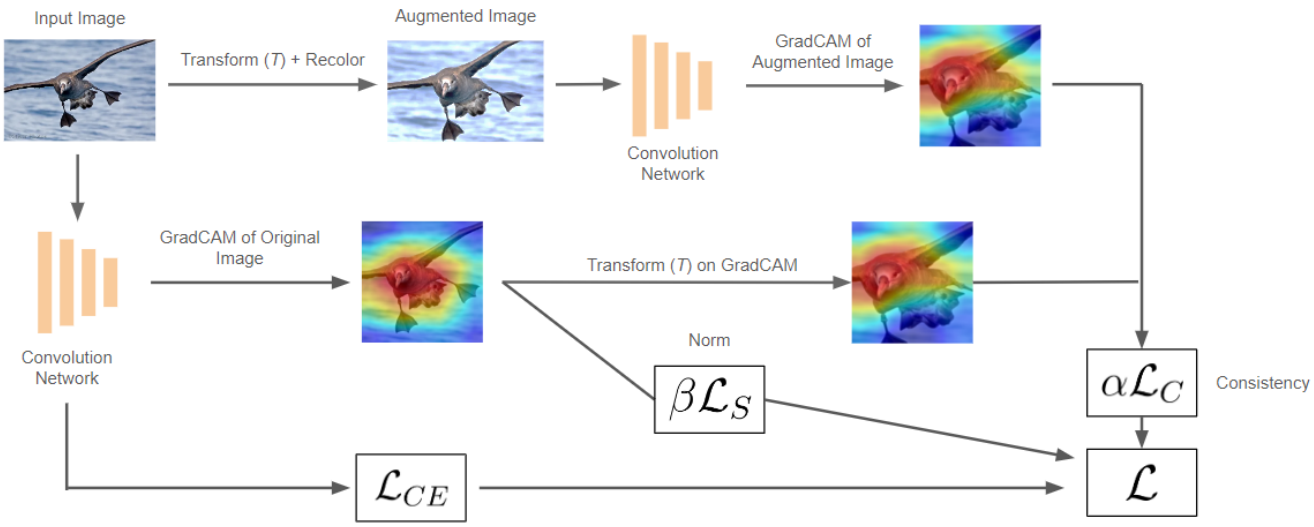

Figure 1. A diagram which explains our method and three loss functions. Although not shown for simplicity, the "Input Image" in this diagram is also perturbed to improve training. The input image starts on the top left and follows three paths which combine to create our final loss. In the path along the left and bottom side of the diagram, the image goes through typical training using cross-entropy loss $\mathcal{L}_{CE}$ based on the input image and class label and added to the total loss $\mathcal{L}$. In the middle path, GradCAM is calculated, then the norm is computed as $\beta\mathcal{L}_S$ and added to the total loss. Finally, along the top path, the image is augmented and the corresponding GradCAM is computed, which is compared with the augmented version of the GradCAM in the middle path, giving a consistency loss $\alpha\mathcal{L}_C$. Altogether, $\mathcal{L} = \mathcal{L}_{CE} + \alpha\mathcal{L}_C + \beta\mathcal{L}_S$.

using $1 - \text{SSIM}$ since we look to minimize $\mathcal{L}_C$. In addition, we use GradCAM [3] as our Φ explanation since it is currently the most commonly-used method for evaluating saliency and benefits from being fast to compute.

3.2.3 Explanation Sparsity

Unfortunately, there are various trivial solutions which a model can learn to easily optimize $\mathcal{L}_C$. For instance, a model can either output an explanation which is entirely zero or covers all of the image uniformly. To prevent the model from outputting a heatmap of entirely zero, we modify GradCAM slightly by removing the ReLU in the original definition of GradCAM. Without this modification, we found the model would trivially set all the activations to be negative and achieve high consistency. To encourage the network to output meaningful, non-uniform heatmaps, we add an explanation sparsity term,

$$\mathcal{L}_S = L^p(\Phi(f, x_i)), \tag{3}$$

where L^p represents the p-th order vector norm, which can be calculated defined on a vector v as $L^p(v) = \sqrt[p]{\sum_j |v_j|^p}$. In our experiments, we tested with $p = 0, 1, 2$ and compare their results.

3.2.4 Final Loss Function

In our training, we combine our three objectives into a single loss function from the cross-entropy loss given in equation 1, the consistency loss given in equation 2, and the sparsity loss given in equation 3:

$$\mathcal{L} = \mathcal{L}_{CE} + \alpha\mathcal{L}_C + \beta\mathcal{L}_S, \tag{4}$$

where α and β are tunable hyperparameters.

3.3. Implementation Details

We used PyTorch to create, train, and evaluate the models for our experiments. We trained both our ResNet18, ResNet50, and ConvNext-tiny models starting with pretrained weights provided by PyTorch. We used SGD to optimize our models. We trained on two datasets, those being CUB-200 and Caltech101 to cover fine-grained classification and general object classification, respectively. We train baseline models with only cross-entropy loss (eq. 1) as well as with our loss (eq 4). Our implementation of GradCAM is a modified version of the pytorch-GradCAM library.

3.3.1 Transformations

For our transformation t, we first randomly resize and crop the input image to a resolution of 224×224. The lower and upper bound of the scale of our random crop is 0.08 and 1.0

respectively. The ratio of the random resize of the image is between 0.75 and 1.33. Then we randomly flip the image with a 0.5 probability. We record our exact augmentations for each image in order to apply them to our baseline GradCAM heatmap Φ as well.

3.3.2 GradCAM

We used a GradCAM PyTorch library to compute our GradCAM saliency maps. We modified this library in order to work with PyTorch's autograd. This is to allow our consistency and sparsity loss function to be backpropogated on. We did not apply any form of augmentation or eigen smoothing to our CAMs for training or in our evaluations. For ResNet18 and ResNet50 we chose to use the last block of layer 4 of the model for GradCAM's target layer. Similarly for ConvNeXT we chose the last convolution layer of the feature layers before the final pooling layer. The target class of our CAMs is the class the model predicts - even if it differs from the actual image label. This same GradCAM configuration was used for every instance of computing GradCAM in our implementation.

3.3.3 Training

We initialized our models with pretrained weights provided by PyTorch before our training. For all of our models we used the same configuration of SGD for our training. We used a learning rate of 10^{-3}, momentum of 0.9, and weight decay of 10^{-4}. In addition, we used a learning rate scheduler to reduced the learning rate by a factor of 0.1 every 30 epochs. On the CUB-200 dataset, we trained our models for 60 epochs. On the Caltech101 dataset we trained for 20 epochs. All models showed their test accuracy converged at these epoch points on these datasets respectively.

4. Results

Results on different convolution neural network As seen in Table 1, we examine the differences in model accuracy, consistency, and IAUC across three convolutional neural networks, those being ResNet18, ResNet50, and ConvNeXT. As expected to their respective complexities, the overall performing accuracies improves from ResNet18, to ResNet50, ConvNeXT.

Results on different datasets: Interestingly, the baseline consistencies on the Caltech dataset were lower than on CUB-200 but after training it overall saw higher consistencies. This could be because the Caltech dataset is more diverse and produces a larger variety of CAM shapes as oppose to the all birds CUB-200 dataset.

Overall, changes in our evaluation metrics are more noticeable on the Caltech dataset. On CUB-200 we observed some instances where changes in accuracy or IAUC were too small to be deemed significant. From our testing, we can comfortably say our method improves the consistency as that trend was apparent across all models and datasets.

4.1. Consistency

We trained our models using our cross-entropy, consistency, and sparsity loss method and a baseline model trained with only cross-entropy loss. In Figure 2, we compared the distribution of GradCAM consistencies of two such models. We observed a consistent measurable increase in consistency scores with models using our training method over the baseline.

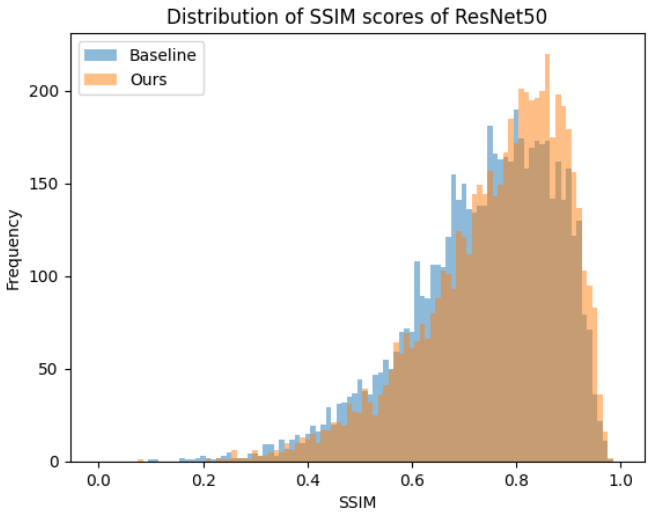

Figure 2. This is the distribution on SSIM scores on the CUB-200 test dataset using the baseline and our consistency trained model. We observe the distribution of the GradCAM SSIM scores shift upwards. The mean and standard deviation of the baseline scores are 0.736 and 0.1391. This particular consistency trained model saw an increased mean of 0.7604 and a decreased standard deviation of 0.1329.

4.2. Tradeoffs with other Evaluation Metrics

Accuracy Tradeoff Overall, there is a relatively small tradeoff between test accuracies and consistency for our method, as demonstrated in Figure 3. We evaluated models trained on random hyperparameters to show that this small tradeoff is not the result of hyperparemeter optimization. For ConvNext-Tiny, the mean tradeoff is $\mu = -0.1057$ with a standard deviation of $\sigma = 0.6717$; for ResNet-18, $\mu = -0.3151$, $\sigma = 0.7494$, and for ResNet50 the mean is $\mu = 2.5918$ with a standard deviation of $\sigma = 6.5049$. This is a four times smaller tradeoff than seen in prior work [2].

Insertion AUC score (IAUC): This metric, proposed in Vitali et. al (2018) [8], aims to quantify the importance of pixels in synthesizing an image. It measures the rise in the

	Method	CUB-200			Caltech		
		Top-1 Acc (%)	Consistency	IAUC	Top-1 Acc (%)	Consistency	IAUC
ResNet18	Baseline	75.40	74.30	0.55	**92.68**	69.67	**0.83**
	Ours	**75.97**	**74.93**	**0.64**	92.50	**83.44**	0.82
ResNet50	Baseline	**81.99**	69.65	0.72	**94.06**	69.35	0.82
	Ours	81.79	**89.64**	**0.77**	91.41	**84.21**	**0.85**
ConvNeXT	Base	85.29	73.35	**0.79**	94.06	41.00	**0.88**
	Ours	**85.45**	**74.61**	0.77	**94.46**	**87.97**	0.80

Table 1. Quantitative Results

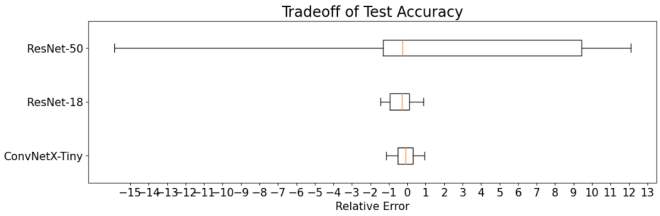

Figure 3. These are Box plots representing the distribution of test accuracy trade-offs for models with varying architectures trained on random hyperparameters on the Caltech dataset.

p	0	1	2
Model Accuracy (%)	80.62	**81.71**	81.26

α	10^1	10^0	10^{-1}
Model Accuracy (%)	0.50	30.08	**81.91**

β	10^{-5}	10^{-6}	10^{-7}
Model Accuracy (%)	30.08	**81.71**	80.24

Table 2. Model test accuracies on CUB-200 for the models used to generate the images in Figure 4. In the first table $\alpha = 10^0, \beta = 10^{-6}$, while in the second table, $\beta = 10^{-5}, p = 1$. In the last table, $\alpha = 1, p = 1$.

probability of a specific class of interest as pixels are added according to a generated importance map. In our case, the importance map is our GradCAM heatmap. This is done by inserting pixels from the highest to lowest attribution scores and then making predictions. The area under curve (AUC) of the ROC curve of these predictions is the IAUC score. The purpose of this metric is to provide a quantitative assessment of causal explanations in image synthesis. For our purposes, higher IAUC scores from our GradCAM saliency maps is an indication of better explanations.

We saw notable improvements in the IAUC scores for models on the CUB-200 dataset. The better performing a model is, the smaller the difference in IAUC between our method and the baseline becomes. This is supported by the observation that ConvNeXT, the best performing model, even saw small decreases in IAUC; and CUB-200, the more difficult dataset, seeing larger changes in IAUC. The reasoning for this could be that well performing models are already adept to requiring less information of an image to make the correct prediction. In general, it seems like our method trained weaker performing models to have a better balance of sensitivity and specificity in their predictions than its baseline.

4.3. Ablation Studies

4.3.1 Quantitative Effects of Hyperparameters

In Figure 4 and Table 2, we explore the effects of varying hyperparameters α, β, p on model explanations and model accuracy while fixing the others for ResNet50 on the CUB dataset.

Interestingly, setting $\alpha = 1, \beta = 10^{-6}$ gives a sparser explanation which is not as sparse as $\alpha = 0.1, \beta = 10^{-5}$, while both models still achieve high accuracy, indicating **there is a range of models which are still accurate but balance explanation consistency and sparsity differently**. Generally, having sparser explanations leads to lower consistency, since it is more difficult to overlap sparser heatmaps. Crucially, in our results in Figure 4, we see that varying one parameter of α, β while fixing the other leads to a set of accurate solutions, the boundary of which has significantly different explanations from the others. Yet, the location of this boundary in the hyperparameter space changes as we vary the other hyperparameter. We might expect there to be a range of models on this boundary, which vary in how they tradeoff consistency and sparsity while maintaining accuracy.

4.3.2 Qualitative Effects of Hyperparameters

In the first row of images of Figure 4, we observe the effects of varying p, or the type of norm, from $p = 0$ to $p = 2$. We observe that the explanations using $p = 0$ and $p = 2$ appear quite similar. We believe using $p = 0$ is difficult to train due to the discontinuous nature of the loss function, so the model is unlikely to produce sparse explanations through this training. On the other hand, $p = 2$

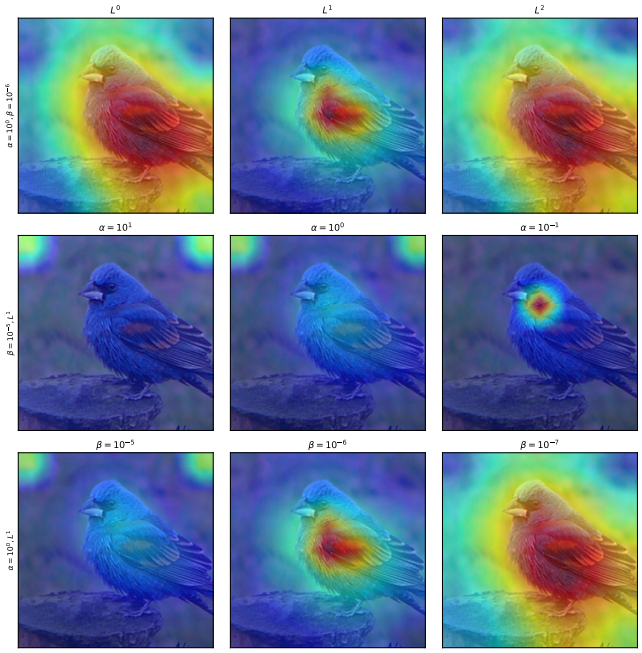

Figure 4. We observe GradCAM outputs generated using ResNet50 on a specific image in the CUB dataset trained using our consistency loss with varying types of regularization, α, β. In row 1, we fix α, β while changing the type of regularization between L^0-norm, L^1-norm, and L^2-norm. In row two, we fix the regularization to L^1-norm and $\beta = 10^{-5}$ while varying α. In the last row, we fix $\alpha = 1$ while varying β.

is simpler to train, but does not to lead to sparsity since L^2 norm is lower for uniform vectors compared to vectors with relatively higher but fewer components. Finally, the explanation generated using $p = 1$ produces a significantly sparser output as expected, since L^1 norm (also known as LASSO) prefers sparser parameterizations over those with higher numbers of smaller components. We see that each of these methods produces approximately the same model accuracy of around 80-81 percent, meaning modifying this parameter has little effect on the accuracy of the model, but does affect the resulting model explanations.

In the second and third row of images, we see the necessity of balancing α and β in order to prevent the model from learning trivial solutions. First, we fix $p = 1$ and $\beta = 10^{-6}$ while varying α. We see that having an excessively large value of α leads to the model trivially outputting an explanation centered on the corners of the image. These models also have dismal accuracy of 0.50% and 30.08% for $\alpha = 10$ and $\alpha = 1$, respectively. The model produces trivial and sparse explanations while almost ignoring the classification task altogether. However, using $\alpha = 0.1$ produces an exceptionally sparse explanation with a high accuracy model of 81.91% on the test set, indicating the model has correctly balanced the classification task while also producing sparse explanations. In the last row of images, we fix $\alpha = 1, p = 1$ and vary β, and we find having β be too high leads to the model not learning, while lower β gives broad explanations which are similar to the baseline. Finally $\beta = 10^{-6}$ offers a good middle-ground of explanation sparsity with high accuracy.

5. Conclusion

We introduce a new set of loss functions which act on model explanations to induce explanation consistency and sparsity while maintaining relatively high accuracy. Using GradCAM as our saliency method, we examine the effects of our losses both quantitatively on model performance as well as qualitatively on model explanation behavior. We find that with careful hyperparameter tuning, we are able to produce models which achieve superior explanation consistency while maintaining similar (and sometimes superior) performance to typical training. We speculate that there exists an entire spectrum of models which balance explanation consistency and sparsity while maintaining performance, which is achieved by modifying both α and β simultaneously. This can achieve exceptionally sparse explanations like those in Figure 4 which are reminiscent of those described in Pillai et. al [2] without contrastive learning. However, we admit this training method is not without faults. The tuning of α, β is very sensitive and if either parameter is not set correctly, the training will fail. In addition, the use of GradCAM in training requires additional VRAM resources due to storing the activations and gradients of the network in memory until the backwards pass which we did not anticipate during the design phase of this project. We speculate there are possible optimizations which can mitigate this, but due to time constraints we were unable to explore this fully.

In addition, there are still an abundance of topics which can be explored in future works regarding training models on explanation consistency. We focused on our efforts on the performance of ResNet50 for the CUB dataset, and eventually extended our results to Caltech dataset and the networks ResNet18, and ConvNeXT. As a result of time constraints, we were not able to fully explore the set of hyperparameters available in this problem and we found relatively less success in these settings. Furthermore, this work can easily be extended to vision transformers, and given the fact that COSE was found to be generally higher in vision transformer networks compared to their convolutional counterparts [1], we believe this training could possibly be even more fruitful on these architectures.

References

[1] Rangel Daroya, Aaron Sun, and Subhransu Maji. Cose: A consistency-sensitivity metric for saliency on image classifi-

cation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 149–158, 2023. 2, 6

[2] Vipin Pillai, Soroush Abbasi Koohpayegani, Ashley Ouligian, Dennis Fong, and Hamed Pirsiavash. Consistent explanations by contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10213–10222, 2022. 1, 2, 4, 6

[3] 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 *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017. 1, 3

[4] 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. *International Journal of Computer Vision*, 128(2):336–359, 2019. 2

[5] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2014. 1

[6] Becks Simpson, Francis Dutil, Yoshua Bengio, and Joseph Paul Cohen. Gradmask: Reduce overfitting by regularizing saliency. *arXiv preprint arXiv:1904.07478*, 2019. 2

[7] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net, 2015. 1

[8] Abir Das Vitali Petsiuk and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. In *In Proceedings of the British Machine Vision Conference (BMVC)*, 2018. 4

[9] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks, 2013. 1

[10] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization, 2015. 1