

AIM: Ammending Inherent Interpretability via Self-Supervised Masking

Paper #2932 Supplementary Material

Table of Contents

The supplementary material covers the following information:

A. Implementation Details

- A.1 Hyperparameters
- A.2 Active-area Loss Threshold Annealing
- A.3 Datasets
 - A.3.1 CUB-200
 - A.3.2 WaterBirds
 - A.3.3 TraverlingBirds
 - A.3.4 ImageNet100
 - A.3.5 ImageWoof
 - A.3.6 ImageNetHard
- A.4 Testing the Center Bias of the Masks

B. Quantitative Results

- B.1 Center Bias
- B.2 Attribution-Agnostic Localization Performance
- B.3 Extended Results of ConvNeXt+AIM
- B.4 Comparison to Other Methods

C. Qualitative Results

- C.1 Additional Quantitative Results
- C.2 Center Bias
- C.3 Qualitative Comparison of Masks and Attribution Maps: AIM vs. Vanilla Backbone Models

D. Ablation Results

- D.1 Active-Area Loss Annealing
- D.2 Without the Auxiliary Losses
- D.3 Do We Need Multiple Mask Estimators at Each Level?
- D.4 The Bottom-Up Bottlenecking Approach
- D.5 Emphasizing peripheral regions in mask estimator initialization

A. Implementation Details

In this section, we present a comprehensive overview of the implementation details of our proposed model to ensure reproducibility and facilitate future research. We begin by outlining the specific hyperparameters used during training, the datasets utilized in our experiments. We then detail the annealing procedures employed to adapt the threshold of the active-area loss used for the mask estimators in the AIM architecture. Additionally, we explain the shifted-center cropping technique implemented to assess the model’s susceptibility to center bias.

A.1. Hyperparameters

The primary hyperparameters used for our AIM models are the shown in Table 4:

Table 4. The Hyperparameters values used for training AIM models.

Hyperparameter	Value
Validation dataset ratio	20% of training dataset
Batch Size	512
top-down pathway learning rate	0.01
Weight Decay	0.001
RandAugment	(ops=3, magnitude=9)
Label Smoothing	0.05
Learning Rate Schedule	cosine
Optimizer	AdamW [25]
drop out rate	0.3
drop out path rate	0.3

We used different learning rates for each of the backbones, as listed in Table 5. Specifically, for the ConvNeXt-tiny+AIM model a much smaller learning rate, compared to those used for the other models, is needed for the training to converge.

Table 5. General training setting used for training AIM.

Model	backbone Learning rate
AIM+ConvNeXt	7e-6
ResNet50+AIM	0.001
ResNet101+AIM	0.001

A.2. Computational Overhead

Tab. 6 quantifies the computational overhead of our proposed AIM module. The integration results in a marginal increase in both GFLOPs and model parameters, confirming the method’s efficiency.

Table 6. Comparison of GFLOPs and the Number of Parameters with Increment Over Baseline Models

Model Name	GFLOPs	Parameters (M)
ConvNeXt-tiny	4.5	28.0
ConvNeXt-tiny+AIM [1]	5.2 (+0.7)	30.7 (+2.7)
ConvNeXt-tiny+AIM [2]	4.6 (+0.1)	29.9 (+1.9)
ResNet50	4.1	23.9
ResNet50+AIM [2]	5.1 (+1.0)	27.6 (+3.7)
ResNet50+AIM [3]	4.4 (+0.3)	26.6 (+2.7)

A.3. Active-area loss threshold annealing

We employ a masking annealing technique to ensure a seamless adaptation of the network to the masking process. The main idea of this technique is to increase the masking or decrease the number of active elements (elements with value 1 in the binary mask) as the training evolves (see Figure 9). This is done by controlling the active-area loss threshold throughout the training process, where we start with fully active masks (a threshold of 1.0), enabling the entire network to operate without constraints, and as training advances, we decrease that threshold with each epoch until reaching the final wanted value (for example 0.35 which means 35% of the mask is only active). The network then uses this final value for the rest of the training. This annealing technique aids the network in adjusting more effectively to the masking constraints, resulting in improved mask quality and a more stable learning process. The number of epochs or steps for which the annealing of the active-area threshold is carried out is treated as a hyperparameter.

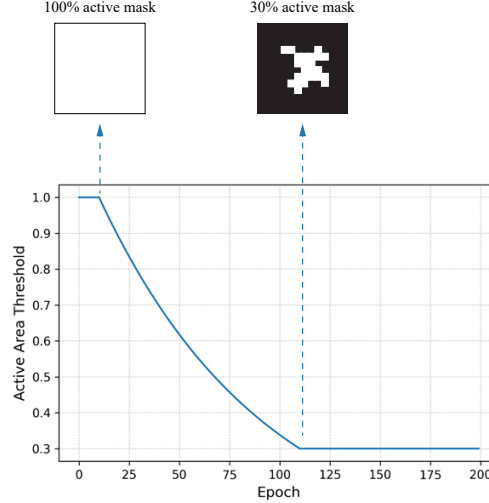


Figure 9. The annealing process of the active-area loss threshold begins either at the start of training or at a specific epoch (e.g., epoch 10 in the figure) and continues for a set number of epochs (e.g., 100 epochs, which is half of the total 200 training epochs). After this period, a final threshold of 0.3 is maintained for the remainder of the training.

A.4. Datasets

In the following sections, we provide details on the datasets used in our experiments: CUB, Waterbirds-100%, Waterbirds-95%, and TravelingBird.

A.4.1. CUB-200

The Caltech-UCSD Birds-200-2011 [50], known as CUB-200-2011, is one of the most well-known datasets in the fine-grained visual classification domain. The dataset consists of 11,788 images of 200 bird species, roughly divided in half for training and half for evaluation. There are an average of 30 images per class in the training dataset and 29 images per class in the test dataset. Throughout this work, we divided the primary training dataset into 80% and 20% training and validation datasets to search over hyperparameters. In our experiments, we utilized an input image size of (224×224) pixels, and for computing the EPG scores, we relied on the binary segmentation [7] masks.

A.4.2. WaterBirds

Waterbirds dataset is a synthetic dataset [33] designed to test classification models by introducing a controllable distribution shift, it is specifically engineered to assess how models respond to shifts in group distributions. The authors took the different bird species from the CUB-200-2011 dataset and simplified the task to be a two-class classification: Waterbirds and Landbirds. They manipulate the backgrounds using the binary segmentation masks [7], where they replaced the original background with either water or land scenes taken from the Places dataset [54]. This creates four different groups: Landbirds on land-background, Landbirds on water-background, Waterbirds on land-background, and Waterbirds on water-background.

The ability to control the construction of the dataset allows researchers to explore how models handle spurious correlations. In the training dataset, the majority of images fall into main groups - Landbirds on land and Waterbirds on water. However, in the validation and test datasets there is an equal distribution across the four groups, creating a deliberate distribution shift between training and test datasets.

The two mainly used versions of this dataset are, the Waterbirds100% version [28], which presents the most extreme challenge, with training dataset that have a perfect correlation between the type of the bird and the background-Water bird on Water background and Land bird on Land background. The other version is the Waterbirds95% [33] where 5% of the training images come from the out-of-distribution groups: Landbird on Water-background and Waterbird on land-background. In our experiments, we utilized an input image size of (224×224) pixels. For computing the EPG scores, we also relied on the binary segmentation masks [7] of the CUB-200 dataset.

A.4.3. TravelingBirds

TravelingBirds dataset [19] is a variant of the CUB dataset and it is constructed in a similar way to that of Waterbirds dataset. While it preserves the original 200 classes of the CUB-200-2011 dataset, it changes the background of the birds to spuriously correlate the target label y and the image background within the training set only. The Authors used the binary segmentation masks to isolate the bird’s pixels from its original background and put them onto a different background scene taken from the Places dataset [54]. Each bird species is laid out on a unique but randomly selected type of scene. During test time, the association between bird label and their background scene type is randomized, Completely disrupting the training set’s correlation between background and class labels, resulting in a challenging adversarial setting. Following [19], we utilized an input image size of (299×299) pixels, and for computing the EPG scores, we also relied on the binary segmentation masks [7] of the CUB-200 dataset.

A.4.4. ImageNet-100

ImageNet-100 [47] is a widely used [1, 15] subset of the full ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 dataset [32]. It is composed of 100 distinct object classes selected from the original 1000, containing approximately 128,000 training images and 5,000 validation images. The primary motivation for using this subset is to enable faster model training, hyperparameter tuning, and experimentation compared to the resource-intensive full dataset, while still offering a diverse and challenging multi-class classification benchmark. In our experiments, we follow standard practice and use an input image size of (224×224) pixels.

A.4.5. ImageWoof

ImageWoof [16] is a challenging 10-class subset of ImageNet [32] designed for rapid yet difficult experimentation. The dataset is intentionally curated to be a hard, fine-grained classification problem, as it contains 10 breeds of dogs that are visually very similar. By focusing on these hard-to-distinguish classes, ImageWoof provides a computationally inexpensive benchmark that is more demanding than a random subset of ImageNet of a similar size. This makes it particularly useful for quickly iterating on and evaluating new model architectures and training methodologies. In our experiments, we utilize an input image size of (224×224) pixels.

A.4.6. Hard-ImageNet

ImageNet-Hard [27] is a benchmark designed to evaluate if models classify images ”for the right reasons” rather than relying on spurious correlations. It consists of 19k training images and 750 quintuply-validated test images across 15 classes: Dog sled, Howler Monkey, Seat Belt, Ski, Sunglasses, Swimming Cap, Balance Beam, Horizontal bar, Patio, Hockey Puck, Miniskirt, Space Bar, Volleyball, Baseball Player, and Snorkel.

The benchmark challenges models with images where the object of interest is captured under suboptimal conditions (e.g., not large or centered). Leveraging segmentation masks helps diagnose whether a model’s prediction is based on the object itself or on misleading background cues, thus providing a deeper understanding of model behavior beyond simple accuracy metrics. For evaluation, we use an input size of (224×224) pixels.

A.5. Energy Pointing Game (EPG) Score

The Energy Pointing Game (EPG) score [51] is a metric designed to quantify the spatial localization capabilities of a model. Specifically, it evaluates the extent to which the model’s attribution aligns with the ground-truth object regions.

The computation of the EPG score requires the following components:

- **Attribution Map:** An importance map generated by an attribution method such as Saliency Maps [40], GradCAM [36], Guided GradCAM [37], or any comparable technique.
- **Ground-Truth Binary Mask:** A binary segmentation mask that delineates the region corresponding to the object of interest.

The EPG score is calculated as the ratio between the total attribution energy within the active (foreground) region of the binary mask and the total attribution energy across the entire attribution map. A higher EPG score indicates that the model concentrates its decision-making evidence within the relevant object regions, thereby demonstrating better spatial grounding of its predictions.

A.6. Testing the center bias of the masks

One of the concerns we had after seeing the masks that our method generated on the CUB-200 dataset was the susceptibility of our models’ masks to center bias. Thus, rather than center-crop the images, we designed a cropping mechanism that, given

a specific deviation from the center, randomly chose one new center for cropping in a way that guarantees to violate the center bias of the object in the newly cropped image. We denote this modified dataset as the shifted-center CUB-200. Figure 10 illustrates the shifted cropping technique used to assess the center bias of the generated masks. Each colored dot represents the center of its corresponding square crop, which is outlined in the same color. For each image, one of the four centers is randomly selected, and the image is cropped accordingly.

It is important to note that this cropping technique might also crop important parts from the object of interest, making the classification much harder due to the lack of task-related features. For example, the body might be cropped out while only the legs of the bird are kept.



Figure 10. The image illustrates the shifted cropping technique used to assess the center bias of the generated masks. Each colored dot represents the center of its corresponding square crop, which is outlined in the same color. For each image, one of the four centers is randomly selected, and the image is cropped accordingly.

B. Quantitative results

This section provides a comprehensive overview of the performance metrics obtained from our trained models under different configurations.

B.1. Center bias

Table 7 presents the performance metrics for the baseline vanilla ConvNeXt-tiny model as well as the ConvNeXt-tiny+AIM models on the Shifted-center CUB-200 dataset. Notably, both the vanilla ConvNeXt-tiny and all variants of the AIM architecture experienced a performance drop, with the ConvNeXt-tiny showing a decrease of approximately 10% and the AIM variants showing a decline of around 9% compared to the center-cropped experiments (see Figure 1). Despite this reduction in accuracy, our proposed model variants still outperform the baseline ConvNeXt-tiny model by nearly 2%. For qualitative comparison of the GradCAM and the generated masks see Figure 12

Table 7. presents the performance metric scores for the baseline vanilla ConvNeXt-tiny model as well as the ConvNeXt-tiny+AIM models on the Shifted-center CUB-200 dataset. Notably, both the vanilla ConvNeXt-tiny and all variants of the AIM architecture experienced a performance drop, with the ConvNeXt-tiny showing a decrease of approximately 10% and the AIM variants showing a decline of around 9% compared to the center-cropped experiments (see Table 9). Despite this reduction in accuracy, our proposed model variants still outperform the baseline ConvNeXt-tiny model by nearly 2%. For qualitative comparison of the GradCAM and the generated masks see Figure 12.

Model	Test Accuracy (\pm Std)
ConvNeXt-tiny	76.98 (± 0.18)
AIM+ConvNeXt (1, 25%)	79.08 (± 0.44)
AIM+ConvNeXt (1, 35%)	79.33 (± 0.45)
AIM+ConvNeXt (1, No Annealing)	79.13 (± 0.35)
AIM+ConvNeXt (2, 25%)	79.11 (± 0.12)
AIM+ConvNeXt (2, 35%)	79.11 (± 0.12)
AIM+ConvNeXt (2, No Annealing)	79.12 (± 0.13)

Upon analyzing the generated masks on the cropped test images (see Figures 12), we can see that large bird regions are missing, which can attribute the performance reduction to the loss of essential parts during the cropping process. The masks clearly focus on the task-related regions or what’s left of them after cropping. This is particularly evident in the masks generated by stage#2 in the second example, where the model focuses on the bird’s legs and belly.

B.2. Attribution-Agnostic Localization Performance:

In addition to GradCAM, we evaluate other attribution methods to assess the generality of our localization improvements. Table 8 reports EPG scores on the Waterbirds-95% dataset using Guided GradCAM and Guided Backprop, both of which show consistent gains with AIM.

Table 8. Attribution-agnostic EPG improvements on Waterbirds-95%. AIM consistently boosts localization scores regardless of the attribution method used.

Model	EPG score \uparrow	
	Guided GradCAM	Guided Backpropagation
Vanilla ConvNext-tiny	46.17(\pm 2.5)	31.26 (\pm 5.4)
ConvNext+AIM (2, 25%)	76.77 (\pm 2.7)	51.89 (\pm 0.3)

B.3. Extended Results of ConvNeXt+AIM:

In this section, we provide extended results for our AIM-enhanced ConvNeXt model, as detailed in Table 9. We supplement our findings on the Waterbirds dataset by including the overall accuracy metric, which further highlights the benefits of our approach. For instance, the best-performing AIM model achieves an overall accuracy of $89.1\% \pm 0.8\%$ on the Waterbirds (100%) benchmark, a significant increase from the baseline’s $71.2\% \pm 2.2\%$. Furthermore, we present results on the CUB-200 dataset, where AIM-based models also outperform the baseline model.

Table 9. **Average Test Accuracies for different configurations of ConvNeXt+AIM.** Comparison of the ConvNeXt-tiny baseline against our AIM-enhanced models on three benchmarks. Our method achieves substantial gains, most notably improving the **worst-group accuracy** on the Waterbirds dataset, which highlights its effectiveness in mitigating spurious correlations. All values are mean accuracy (%) \pm standard deviation.

Model	CUB		Waterbirds					Travelingbirds	
	Acc	EPG	100%		95%		EPG	Acc	EPG
ConvNeXt-t	87.9 (± 0.02)	68.3 (± 0.1)	39.6 (± 5.4)	71.2 (± 2.2)	57.19 (± 6)	81.6 (± 3.2)	68.3 (± 3.2)	59.5 (± 0.8)	74.4 (± 0.6)
ConvNeXt-t+AIM[1, 25%]	88.6 (± 0.1)	76.804 (± 3.3)	73.6 (± 4.5)	86.2 (± 2.6)	60.11 (± 1.3)	91.2 (± 0.8)	95.7 (± 0.7)	77.1 (± 5.15)	77.1 (± 0.3)
ConvNeXt-t+AIM[1, 35%]	88.18 (± 0.1)	55 (± 3.8)	77.1 (± 4.4)	88.4 (± 1.8)	57.2 (± 1.3)	90.7 (± 0.7)	96 (± 0.3)	63 (± 1.2)	71.5 (± 1.3)
ConvNeXt-t+AIM[2, 25%]	87.78 (± 0.2)	75.17(± 1.2)	74 (± 5)	84(± 6)	58 (± 1.3)	92.7 (± 1.2)	96.6 (± 0.2)	75 (± 6)	77.4 (± 0.2)
ConvNeXt-t+AIM[2, 35%]	88.5 (± 0.2)	62.5 (± 0.3)	78.1 (± 2.3)	89.1 (± 0.8)	68.5 (± 3.6)	92.3 (± 0.6)	96.31 (± 0.1)	71.7 (± 6.4)	71 (± 0.4)

B.4. Comparison To Other Methods:

This section compares AIM to other methods. Unlike our approach, which does not use any form of guidance, other methods often rely on guidance mechanisms to direct the trained model. Examples of such guidance include using binary masks [30] or leveraging the output of a CLIP-based model controlled by a text prompt GALS [28] or using specific loss to account for imbalanced data [48]. Our proposed AIM solely relies on image labels used by traditional image classification methods.

Table 10. **Average Test Accuracies for Various Models with and without AIM Modification.** This table presents the average test accuracies (with standard deviations) for different models on the CUB-200, Waterbirds, and Travelingbirds datasets. The models include ConvNeXt-tiny, ResNet50, and ResNet101 with and without the AIM modification. The AIM-enhanced models are denoted as *[backbone]+AIM (stage index, mask active-area loss)*, such as ConvNeXt-tiny+AIM(1, 25%). The results are organized by dataset and further divided into categories: overall accuracy and worst-group accuracy for both 100% and 95% subsets where applicable. Improvements or declines in performance due to the AIM modification are highlighted in green and red, respectively. This comprehensive comparison provides insights into the effectiveness of the AIM approach in enhancing model performance across different datasets and scenarios.

Model	CUB-200 (%)	Waterbirds (%)				Travelingbirds (%)
		100%		95%		
		Worst-Group	Overall	Worst-Group	Overall	
ConvNeXt-tiny	87.9 (± 0.02)	39.6 (± 5.4)	71.2 (± 2.2)	81.6 (± 3.2)	94.8 (± 0.3)	59.5 (± 0.8)
ConvNeXt-t+AIM[1, 25%]	88.6 (± 0.1)	73.6 (± 4.5)	86.2 (± 2.6)	91.2 (± 0.8)	95.7 (± 0.7)	77.1 (± 0.3)
ConvNeXt-t+AIM[1, 35%]	88.18 (± 0.1)	77.1 (± 4.4)	88.4 (± 1.8)	90.7 (± 0.7)	96 (± 0.3)	71.5 (± 1.3)
ConvNeXt-t+AIM[2, 25%]	87.78 (± 0.2)	74 (± 5)	84 (± 6)	92.7 (± 1.2)	96.6 (± 0.2)	77.4 (± 0.2)
ConvNeXt-t+AIM[2, 35%]	88.5 (± 0.2)	78.1 (± 2.3)	89.1 (± 0.8)	92.3 (± 0.6)	96.31 (± 0.1)	71 (± 0.4)
DRO [48] + ResNet50	-	-	-	91.4 (± 1.1)	-	-
GALS [28] + ResNet50	-	56.71 (-)	-	76.54 (-)	-	-
BCos-ResNet50 [30]	-	56.1 (± 4)	-	-	-	-
ResNet50	81.32	41.29 (± 1.99)	69.48 (± 2)	71.09 (± 5.98)	91.21 (± 0.27)	50.21 (± 0.5)
ResNet50+AIM (2, 25%)	83.90	52.19 (± 9.11)	73.84 (± 1.14)	75.97 (± 2.344)	92.1 (± 0.31)	65.2 (± 0.24)
ResNet50+AIM (3, 25%)	83.16	40.12 (± 2.1)	71.36 (± 1.7)	77.2875 (± 0.404)	92.38 (± 0.2)	64 (± 0.51)
ECBM [52] + ResNet101	-	-	-	-	-	58.4 (-)
ResNet101	76.41 (± 0.619)	36.48 (± 4.22)	71.69 (± 2.2)	77.565 (± 3.2)	92.82 (± 0.31)	19.54 (± 0.34)
ResNet101+AIM (2, 25%)	82.55 (± 0.22)	38.11 (± 1.4)	71.46 (± 0.53)	82.39 (± 0.88)	93.12 (± 0.74)	44.03 (± 0.96)
ResNet101+AIM (3, 25%)	82.635 (± 0.28)	38.81 (± 1.4)	71.8 (± 0.69)	82.39 (± 1.328)	93.34 (± 0.7)	45.13 (± 0.7)

C. Qualitative results

In this section, we explore the qualitative aspects of our study by showcasing masks and Grad-CAM attributions across different settings. These visualizations provide insights into the models’ decision-making processes, highlighting areas of focus and variation in response to different configurations.

C.1. Additional qualitative results:

In the figure 11, we illustrate additional examples from CUB, waterbirds-100%, Travelingbirds and Hard-ImageNet datasets, where it is clear from the GradCAM attributes that AIM-based models focus on the object and ignore the spurious features.

C.2. Center bias

Figure 12 presents a qualitative comparison of different architectural configurations of AIM on the Shifted-center CUB-200 setting. The figure illustrates the masks generated at each stage for two primary architectural variants: ConvNeXt-tiny+AIM (1), shown in the top group of images, and ConvNeXt-tiny+AIM (2), shown in the bottom group. Each variant employs different mask active-area thresholds, with each row representing a distinct threshold setting. The masks produced by AIM (columns denoted stage#1 masks, stage#2 masks, stage#3 masks) explicitly delineate the regions utilized by the model at each stage, thereby demonstrating that AIM effectively mitigates susceptibility to center bias. The “Merged Masks” column demonstrates where the final feature maps will be zero, highlighting the discarded regions.

C.3. Qualitative Comparison of Masks and Attribution Maps: AIM vs. Vanilla Backbone Models

In this section, we present qualitative results to illustrate the effectiveness of our proposed approach. We selected five different input images from each of the following datasets: CUB-200, Waterbirds-95%, Waterbirds-100%, and Traveling Birds. For these images, we compare the GradCAM attribution maps generated by the AIM+[backbone] models with different mask

active-area thresholds to those produced by the vanilla [backbone] models. Additionally, we display the generated merged masks from the AIM+[backbone] models.

- For the **CUB-200 dataset** we show the ConvNeXt-tiny+AIM ($T=1$) and ($T=2$) outputs in Figure 13 and Figure 14 respectively, while we illustrate the output of ResNet50+AIM ($T=2$) and ($T=3$) Figure 15 and Figure 16 respectively.
- For the **Waterbirds-95%** dataset, we present the ConvNeXt-tiny+AIM outputs with mask active-area thresholds ($T=1$) and ($T=2$) in Figure 17 and Figure 18, respectively. Additionally, we illustrate the outputs of ResNet50+AIM with ($T=2$) and ($T=3$) in Figure 19 and Figure 20, respectively.
- For the **Waterbirds-100%** dataset, we show the outputs of ConvNeXt-tiny+AIM and ResNet50+AIM in Figure 23 and Figure 24, respectively.
- For the **TravelingBirds** dataset, we present the outputs of ResNet50+AIM and ResNet101+AIM in Figure 22 and Figure 21, respectively.

D. Ablation results

In this appendix section, we conduct an ablation study to explore the roles of individual components and parameters within our proposed method. By systematically adjusting or removing certain elements, we aim to gain a better understanding of their contributions to the model’s performance.

D.1. Active-Area Loss Annealing:

This section analyzes the performance variations in ConvNeXt-tiny+AIM models, focusing on two architectural variants: ConvNeXt-tiny+AIM ($T=1$) and ConvNeXt-tiny+AIM ($T=2$). The analysis examines changes in accuracy and energy pointing game (EPG) scores relative to the vanilla ConvNeXt-tiny model when the mask active-area threshold τ is adjusted on the Waterbirds-95% dataset.

The results in Table 11 reveal that reducing the active-area threshold leads to improvements in both the worst group and overall accuracies; this can also be seen in Figure 25. This finding suggests that by constraining the spatial areas upon which the models depend, they are compelled to identify and concentrate on regions pertinent to the task. This focus is evidenced by the higher EPG scores achieved, indicating effective task-related region identification.

Table 11. This table presents the performance changes, in terms of accuracy and energy pointing game (EPG) scores on the Waterbirds-95% dataset, for ConvNeXt-tiny+AIM models with two architectural variants: ConvNeXt-tiny+AIM ($T=1$) and ConvNeXt-tiny+AIM ($T=2$). These are compared against the vanilla ConvNeXt-tiny model when altering the mask active-area threshold τ . The results indicate that a smaller active-area threshold leads to increases in both the worst group and overall accuracies. This suggests that limiting the spatial areas the models rely on encourages them to identify and focus on task-related regions, as reflected by high EPG scores. Another notable trend is that the ConvNeXt-tiny+AIM ($T=1$) variants achieve better EPG scores, implying that the masks in this variant, applied across all three stages of the top-down pathway, effectively concentrate on the regions of interest (ROI). This is also seen more clearly in Figure 25.

Model	Annealing final threshold τ	Waterbirds-95%		
		Worst-group	Overall	EPG
Vanilla ConvNeXt-tiny	–	79.63 (\pm)	93.755 (± 1.340)	63.52
AIM+ConvNeXt (1)	10%	92.16 (\pm)	95.62 (\pm)	84.3
AIM+ConvNeXt (2)		94 (\pm)	96.63 (\pm)	81
AIM+ConvNeXt (1)	15%	92.24 (\pm)	95.93 (\pm)	81.7
AIM+ConvNeXt (2)		93.33 (\pm)	96.53 (\pm)	62.3
AIM+ConvNeXt (1)	20%	91.46 (\pm)	95.69 (\pm)	73
AIM+ConvNeXt (2)		93.53 (\pm)	96.51 (\pm)	57.4
AIM+ConvNeXt (1)	25%	91.74 (\pm)	96.31 (\pm)	74.22
AIM+ConvNeXt (2)		93.24 (\pm)	95.62 (\pm)	72.1
AIM+ConvNeXt (1)	30%	91.35 (\pm)	96.36 (\pm)	59
AIM+ConvNeXt (2)		92.87 (\pm)	96.65 (\pm)	49
AIM+ConvNeXt (1)	35%	91.11 (\pm)	96.38 (\pm)	56.13
AIM+ConvNeXt (2)		92.59 (\pm)	96.39 (\pm)	45.83
AIM+ConvNeXt (1)	40%	90.68 (\pm)	96.24 (\pm)	41.7
AIM+ConvNeXt (2)		91.89 (\pm)	96.29 (\pm)	53.8

D.2. Without The Auxiliary Losses:

As discussed in Section 3, the AIM architecture incorporates a classification loss as an auxiliary loss at each stage in the top-down pathway, following the approach from [21]. These losses help align the learned features at each stage with the final task. However, this raises an important question: How would the AIM network perform if these auxiliary losses were removed? What impact would this have on the network’s overall performance?

Table 12 examines the impact of removing auxiliary losses on the performance of ConvNeXt-tiny+AIM and ResNet50+AIM models across the CUB-200 and Waterbirds-95% datasets. The results, measured in terms of accuracy and energy pointing game (EPG) scores, reveal notable performance fluctuations when auxiliary losses are omitted. On the Waterbirds-95% dataset, some settings show performance gains, while others experience declines. Conversely, on the CUB-200 dataset, performance consistently deteriorates without the auxiliary losses. These observations highlight the critical role of auxiliary losses in maintaining stable and reliable performance across different models and datasets.

Table 12. **AIM networks performance worsens without auxiliary losses.** This table presents the performance variations of ConvNeXt-tiny+AIM and ResNet50+AIM on the CUB-200 and Waterbirds-95% datasets, measured in terms of accuracy (averaged over 4 different runs) and energy pointing game (EPG) scores. Red arrows indicate a decrease in the score compared to the corresponding architecture with the auxiliary losses, while green arrows represent an increase in scores. The results indicate that removing the auxiliary losses leads to fluctuations in performance across the datasets and models’ variations. Specifically, on the Waterbirds-95% dataset, performance increases for some settings while decreasing for others. On the CUB-200 dataset, performance consistently worsens compared to when auxiliary losses are used. These findings underscore the importance of auxiliary losses in achieving consistent performance across models and datasets.

Model	τ	Auxiliary losses	CUB-200		Waterbirds-95%		
			Acc	EPG	Worst-group Acc	Overall	EPG
ConvNeXt-tiny+AIM (1, τ)	25%	no	88.29↓ (± 0.16)	59.4↑	92.24↑ (± 0.73)	96.84↑ (± 0.086)	73.82↓
	25%	yes	88.63 (± 0.13)	58.54	91.31 (± 1.1)	96.77 (± 0.1)	74.22
	35%	no	88.51↓ (± 0.075)	50.12↓	91.7↑ (± 0.76)	96.70↑ (± 0.35)	69.57↑
	35%	yes	88.82 (± 0.21)	57.49	90.1 (± 1.44)	96.63 (± 0.13)	56.13
	no annealing	no	88.6↓ (± 0.15)	44.69↓	92.23↑ (± 2.88)	94.89↓ (± 2.13)	48.17↑
	no annealing	yes	88.77 (± 0.1)	53.56	89.3 (± 2.48)	96.07 (± 0.16)	42.57
ConvNeXt-tiny+AIM (2, τ)	25%	no	88.31↓ (± 0.19)	59.2↓	93.18↑ (± 1.1)	97.27↑ (± 0.32)	78.4↑
	25%	yes	88.55 (± 0.3)	60.27	90.1 (± 1.46)	96.43 (± 0.2)	72.12
	35%	no	88.47↓ (± 0.17)	54.19↓	92.01↑ (± 0.51)	96.85↑ (± 0.1)	63.98↑
	35%	yes	88.67 (± 0.25)	56.82	91.41 (± 0.22)	96.40 (± 0.23)	45.83
	no annealing	no	88.68↑ (± 0.15)	41.49↓	91.78↑ (± 0.74)	95.94↓ (± 0.26)	58.48↓
	no annealing	yes	88.62 (± 0.21)	55.54	88.7 (± 1.4)	96.39 (± 0.104)	65.84
ResNet50+AIM (2, τ)	25%	no	79.74↓ (± 0.14)	55.86↓	77.68↓ (± 0.673)	92.41↓ (± 0.184)	61.89↓
	25%	yes	80.9 (± 0.345)	57.56	91.31 (± 1.1)	96.77 (± 0.1)	74.22
	35%	no	79.85↓ (± 0.07)	54.54↓	77.41↓ (0)	92.35↑ (± 0.1)	51.66↓
	35%	yes	80.9 (± 0.12)	57	90.1 (± 1.44)	92.34 (± 0.25)	56.13
	no annealing	no	79.27↓ (± 0.23)	47.49↓	75.50↓ (± 0.545)	91.7↓ (± 0.26)	43.04↑
	no annealing	yes	80.84 (± 0.28)	53.81	89.3 (± 2.48)	91.9 (± 0.11)	42.57
ResNet50+AIM (3, τ)	25%	no	79.69↓ (± 0.35)	57.66↑	77.83↓ (± 0.61)	92.64↓ (± 0.38)	71.87↑
	25%	yes	80.9 (± 0.345)	56.76	91.31 (± 1.1)	96.77 (± 0.1)	61.89
	35%	no	79.77↓ (± 0.41)	55.04↑	78.38↓ (± 1.48)	91.92↓ (± 0.35)	46.35↓
	35%	yes	80.9 (± 0.12)	54.80	90.1 (± 1.44)	92.42 (± 0.46)	52.14
	no annealing	no	79.55↓ (± 0.21)	48.73↓	74.87↓ (± 2.0)	92.32↓ (± 0.32)	36.79↓
	no annealing	yes	80.84 (± 0.28)	52.71	89.3 (± 2.48)	87.39 (± 8.56)	41.78

D.3. Do we need multiple mask estimators, one at each level?

One of the building blocks of our proposed method is the use of mask estimators in the top-down pathway, where we employ a mask estimator module at each stage, which we denote here as the Full AIM model. Given that in convolutional neural networks the feature maps from the last level determine the final semantic features used in the classification layer, a question arises: *can we rely solely on the mask generated at the first stage of the top-down pathway (this stage corresponds to the final convolutional layer in the backbone network used)?*

This question is motivated by the semantic richness of the feature maps at this level; thus, selecting areas by masking in these feature maps is supposed to truly represent the task-related semantics.

To investigate this question, we adjusted our network to use only the masks generated at the first stage of the top-down pathway. This is done by passing the first stage’s masks to the subsequent stage after up-scaling them by a factor of 2, as the subsequent stage has higher spatial resolutions.

Table 13. Test accuracies of different AIM models on the CUB-200 dataset with masks only adapted from the first stage of the top-down pathway (highlighted with \checkmark mark) versus the standard AIM network, which employs a mask estimator at each stage of the top-down pathway.

Model	One Mask Estimator	CUB-200 (%)
AIM+ConvNeXt (1, 25%)	\times	88.635 (± 0.13)
AIM+ConvNeXt (1, 35%)	\times	88.82 (± 0.213)
AIM+ConvNeXt (1, No Annealing)	\times	88.775 (± 0.044)
AIM+ConvNeXt (2, 25%)	\times	88.557 (± 0.296)
AIM+ConvNeXt (2, 35%)	\times	88.677 (± 0.25)
AIM+ConvNeXt (2, No Annealing)	\times	88.622 (± 0.211)
AIM+ConvNeXt (1, 25%)	\checkmark	88.14 (± 0.44)
AIM+ConvNeXt (1, 35%)	\checkmark	88.22 (± 0.26)
AIM+ConvNeXt (1, No Annealing)	\checkmark	88.25 (± 0.33)
AIM+ConvNeXt (2, 25%)	\checkmark	88.52 (± 0.22)
AIM+ConvNeXt (2, 35%)	\checkmark	88.41 (± 0.20)
AIM+ConvNeXt (2, No Annealing)	\checkmark	88.26 (± 0.28)

Table 13 presents the test accuracies of different configurations of our AIM models on the CUB-200 dataset. In these experiments, we compare the standard AIM network, which employs a mask estimator at each stage of the top-down pathway, with a modified version that uses masks only from the first stage of the top-down pathway. The modified approach passes the first-stage masks to subsequent stages to be applied on the corresponding feature maps. These models are indicated with a \checkmark in the ‘One Mask Estimator’ column in the Table 13.

Analyzing the results, we observe that the models’ results are very close to each other, with models using masks solely from the first stage generally achieving slightly lower test accuracies than those employing mask estimators at each stage.

In convolutional neural networks, as input data moves through the network layers, the feature maps become richer in semantic information but lose spatial localization. We hypothesize that applying masks generated at the first stage of the top-down pathway, which is coarser and less localized (See Figure 26) compared to those from later stages, to subsequent lower stages could inadvertently include non-task-related regions, potentially diminishing performance, especially on out-of-distribution datasets.

D.4. The bottom-up bottlenecking approach:

One of the most relevant works to ours is [49], which uses a bottom-up masking approach to focus on essential regions in the generated feature maps. This approach relies on employing mask units between the main blocks of a convolutional network to produce binary masks highlighting task-relevant regions for the network. These mask units utilize the Gumbel-softmax trick to generate binary masks, which are then used to spatially bottleneck feature maps produced after each main block.

However, as discussed in the related work Section 2, this bottom-up approach needs skip-connections to work (See Figure 27) as the network needs more information to be flown from the shallow layer to the deepest layers to form an understanding of the scene, this compromised the inherent interpretability that comes naturally with the sparse feature maps, which with the skip-connections fall back to dense features maps.

We closely follow the method described in [49], but with one key difference: *we utilized fully sparse feature maps without skip connections*. This decision aligns with our objective of creating an interpretable model by focusing on generating fully sparse feature maps, which makes the decision-making process transparent (See Figure 28). For this purpose, we adhered to the same training setup we used for our approach, 200 epochs with active mask annealing applied for half the training period. We denote this architecture as *bottom-up* AIM Network.

We tested three architectural variants that differed in the number of masked convolutional blocks and experimented with two annealing threshold settings.

Table 14. This table presents the performance of Bi-gatedNet with ConvNeXt-tiny on the CUB-200 dataset when different blocks are masked and various mask active-area thresholds (τ) are applied. Masking all three blocks resulted in the lowest accuracy with high variability, indicating unstable training. Masking only the last two blocks improved accuracy but yielded poorly localized masks. Masking only the last block achieved the highest accuracy, surpassing the unmasked ConvNeXt-tiny model, and produced highly focused and localized masks.

Model Name	Bottom-Up Blocks Masked	CUB-200 (%)
ConvNext-tiny	-	86.827 (± 0.761)
standard ConvNeXt-tiny+AIM ($T=1, \tau=25\%$)	-	88.82 (± 0.213)
standard ConvNeXt-tiny+AIM ($T=2, \tau=25\%$)	-	88.677 (± 0.25)
<i>Bottom-up</i> ConvNeXt-tiny+AIM ($\tau=25\%$)	1, 2, 3	72.79 (± 8.51)
<i>Bottom-up</i> ConvNeXt-tiny+AIM ($\tau=25\%$)	2, 3	82.53 (± 1.39)
<i>Bottom-up</i> ConvNeXt-tiny+AIM ($\tau=25\%$)	3	86.202 (± 0.44)
<i>Bottom-up</i> ConvNeXt-tiny+AIM ($\tau=35\%$)	1, 2, 3	67.45 (± 8.52)
<i>Bottom-up</i> ConvNeXt-tiny+AIM ($\tau=35\%$)	2, 3	84.00 (± 1.38)
<i>Bottom-up</i> ConvNeXt-tiny+AIM ($\tau=35\%$)	3	86.975 (± 0.34)

Our main observation was that applying masks to all three blocks resulted in the lowest accuracy with high variability across trials (see Table 14), indicating unstable training and the model’s inability to learn effective representations. Additionally, the masks generated in this configuration (see Figure 29) showed almost no masking despite the application of the mask active-area loss with annealing.

When we adjusted the architecture to mask only the last two blocks, the accuracy improved sharply, by 10% for the annealing threshold of 0.25 and 13% for 0.35, reaching above 80% accuracy (see Table 14). However, the quality of the masks remained poor: the masks at the lower levels were still almost fully active, and only the mask from the last block focused on the bird.

This is evident in Figure 29, which shows that even when we limit the active area region to 25% and 35% using mask active-area loss annealing, the masks remain fully activated except for the one from the last block.

Furthermore, table 15 compares between our top-down and bottom-up approach in terms of sparsity and localization (EPG); the top-down method outperforms the bottom-up baseline on both metrics even without any sparsity loss (No annealing) on Waterbirds-95% dataset.

Table 15. Comparison of our top-down approach (ConvNext+AIM) against a bottom-up baseline on the Waterbirds-95% dataset. Our method demonstrates superior performance across all metrics, highlighting its ability to improve localization (EPG) and sparsity even without an explicit sparsity objective (No annealing).

Model	Worst group ACC (\pm std) \uparrow	EPG (\pm std) \uparrow	Sparsity score (\pm std) \downarrow
Bottom-up ([1, 2, 3], 25%)	85.63 (± 4.2)	29.28 (± 21.4)	100 (± 0)
ConvNext+AIM (1, 25%)	92.21 (± 1.2)	77.1 (± 0.06)	17.7 (± 0.6)
Bottom-up ([1, 2, 3], No annealing)	87.41 (± 4.2)	24.66 (± 9.4)	100 (± 0)
ConvNext+AIM (1, No annealing)	89.5 (± 4.13)	43.43 (± 5.13)	71.48 (± 0)

In summary, training with the bottom-up approach indicates that masking fewer blocks leads to better performance and more robust masks.

D.5. Emphasizing peripheral regions in mask estimator initialization

During training, the layers of the mask estimators are typically initialized randomly. To investigate the effect of the initialization scheme on the performance of the mask estimators, we propose an alternative initialization method that emphasizes the peripheral regions over the central regions. This is achieved by weighting the convolutional filter weights according to their spatial distance from the center of the filter kernel. Specifically, we assign larger initial values to the weights corresponding to the edges of the filters than to those closer to the center, effectively biasing the filters to be more responsive to features in the outer areas of the input (see Figure 30 for an illustration).

However, our experiments on the CUB-200 dataset reveal that this specialized initialization does not have a significant impact on performance. As shown in Table 16, the model’s accuracy remains essentially unchanged compared to when using standard random initialization. This suggests that the mask estimators are robust to the initial weight distribution and that the network is capable of learning effective representations regardless of this specific initialization strategy.

Table 16. **Emphasizing Peripheral Regions in Mask Initialization Does Not Impact AIM Network Performance.** This figure compares the AIM network’s performance on the CUB-200 dataset using two different initialization strategies for the mask estimators: standard random initialization and an initialization that emphasizes peripheral regions by assigning larger weights to filter edges. The results indicate that the model’s accuracy remains essentially unchanged between the two approaches. This suggests that the mask estimators are robust to the initial weight distribution, and the network can effectively learn meaningful representations regardless of this specific initialization strategy.

Model	CUB-200 (%)
<i>Edge-emphasized initialization</i> ConvNeXt-tiny+AIM (2, 25%)	88.76 (± 0.171)
<i>Standard random initialization</i> ConvNeXt-tiny+AIM (2, 25%)	88.678 (± 0.251)

D.6. User Perception Study Details

To quantitatively assess if our model’s improved focus is perceptually meaningful, we conducted an online user study. Participants accessing the study were first directed to a welcome page that outlined the research goals, the task procedure, and our data privacy policy (Fig. 31). The introduction defined attribution maps as heatmaps used to visualize an AI’s focus and assured participants that all responses would be anonymous and confidential.

The study was designed to be completed in approximately 10 minutes. To achieve this while still covering a broad set of 100 images from the Waterbirds-100% dataset, the images were randomly partitioned into four disjoint subsets of 25. Each of the 107 participants was randomly assigned to evaluate exactly one of these four subsets. Following this assignment, participants proceeded to the main evaluation task, which consisted of 25 forced-choice comparison trials.

In each trial, participants were shown an image from their assigned subset, displayed alongside two GradCAM attribution maps: one from the baseline vanilla ConvNeXt and one from our AIM-equipped model. For each pair, they were asked to select the map that, in their opinion, “*more accurately and clearly focuses on the main object in the image, and less on irrelevant parts*”. An example of this trial layout is shown in Fig. 32. To mitigate positional bias, the on-screen placement of the two maps was randomized for every trial.

A total of 107 participants completed the study, yielding $107 \times 25 = 2675$ individual evaluations. The results show a strong and significant user preference for our model. AIM’s attribution maps were chosen in **70.7%** of the evaluations. A two-sided binomial test confirms that this outcome is highly significant ($p < 0.00001$), providing strong evidence that the explanations generated by AIM are more aligned with human intuition regarding object-centric focus compared to the baseline.

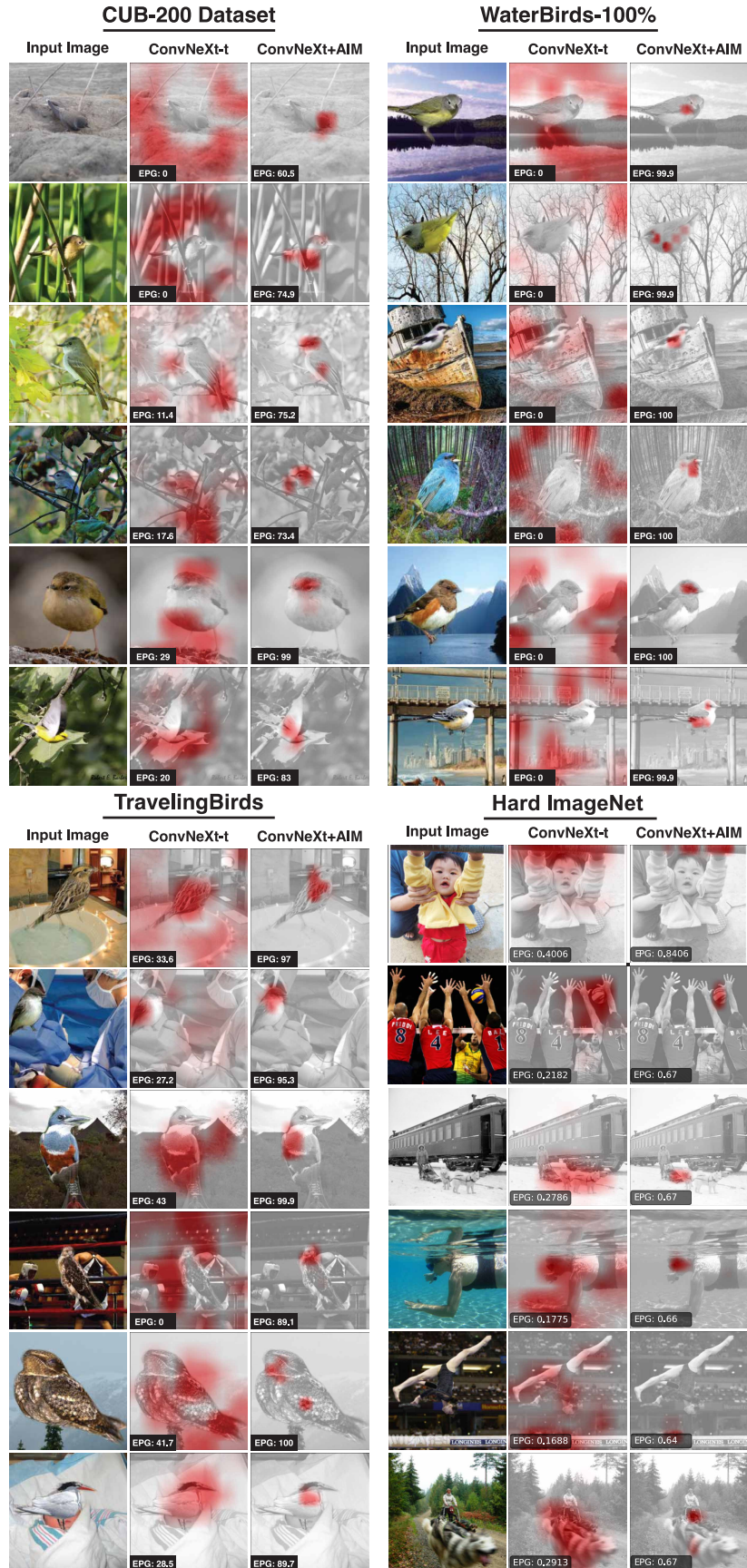


Figure 11. Models amended with AIM consistently exhibit enhanced localization of genuine features, effectively suppressing spurious cues in both in-domain and out-of-domain scenarios. A qualitative visualization of Grad-CAM heatmaps comparing baseline ConvNeXt-tiny models and ConvNeXt-tiny+AIM models across CUB-200, TravelingBirds, WaterBirds-100%, and Hard-ImageNet. The EPG scores are indicated on each heatmap.

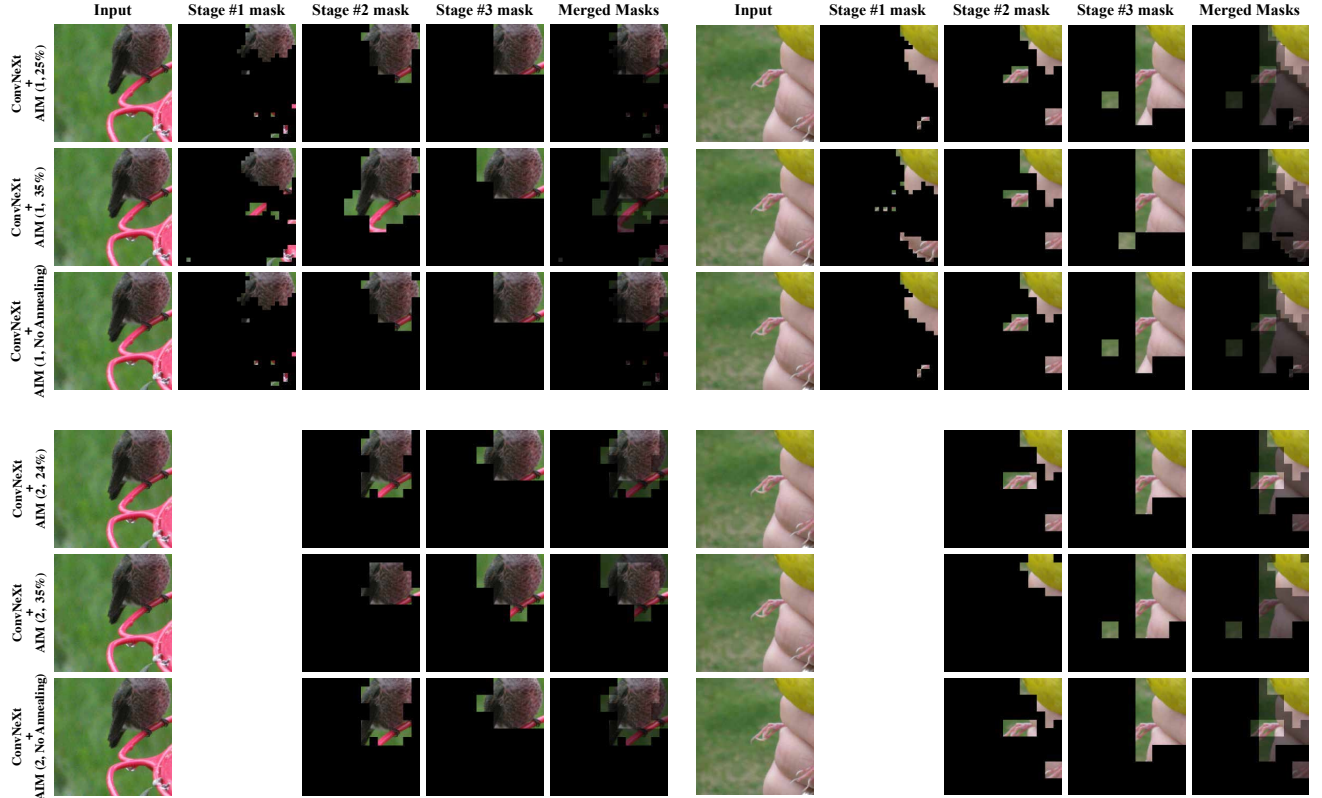


Figure 12. **Qualitative Comparison between the different architectural configurations of AIM on the Shifted-center CUB-200 setting.** This figure illustrates the masks generated at each stage for two primary architectural variants: ConvNeXt+AIM ($T=1$) (The group of images at the top) and ConvNeXt+AIM ($T=2$) (the group of images at the bottom), each utilizing different mask active-area thresholds (each row represent different active-area loss setting). The masks produced by AIM (columns denoted stage#1 masks, stage#2 masks, stage#3 masks) explicitly delineate the regions utilized by the model at each stage, thereby demonstrating that AIM effectively mitigates susceptibility to center bias. The "Merged Masks" column demonstrates where the final feature maps will be zero, highlighting the discarded regions.

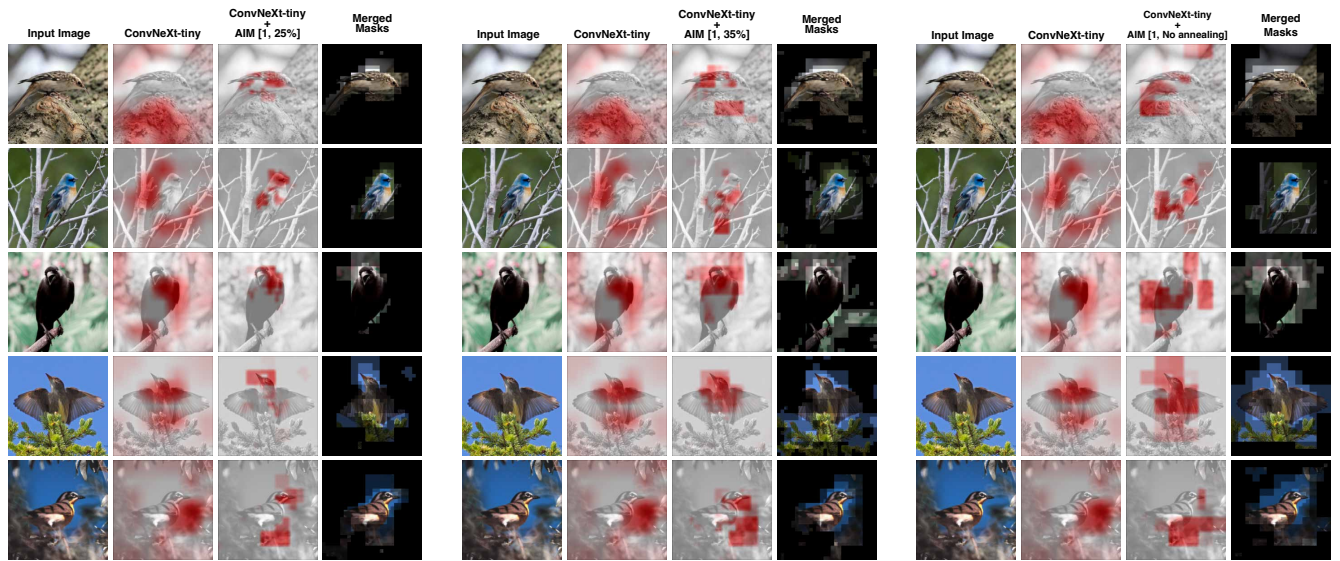


Figure 13. **Qualitative results on the CUB-200 dataset.** Comparison of GradCAM attribution maps between ConvNeXt-tiny+AIM models with different mask active-area thresholds and the vanilla ConvNeXt-tiny, along with the generated merged masks from the ConvNeXt-tiny+AIM model.

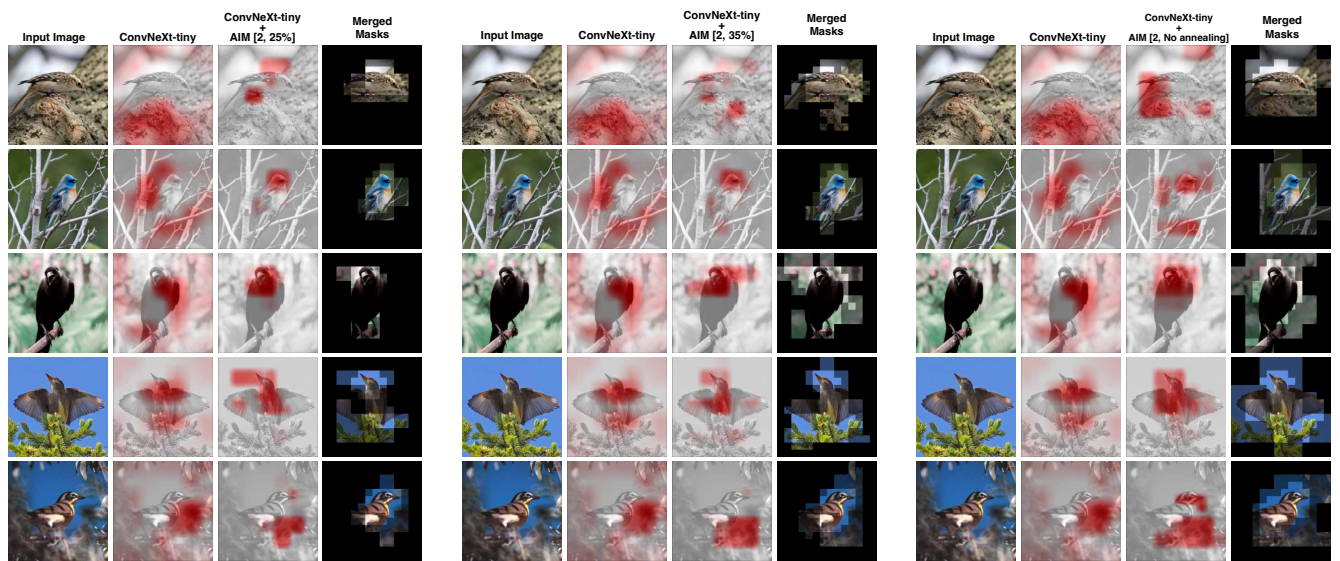


Figure 14. **Qualitative results on the CUB-200 dataset.** Comparison of GradCAM attribution maps between ConvNeXt-tiny+AIM models with different mask active-area thresholds and the vanilla ConvNeXt-tiny, along with the generated merged masks from the ConvNeXt-tiny+AIM model.

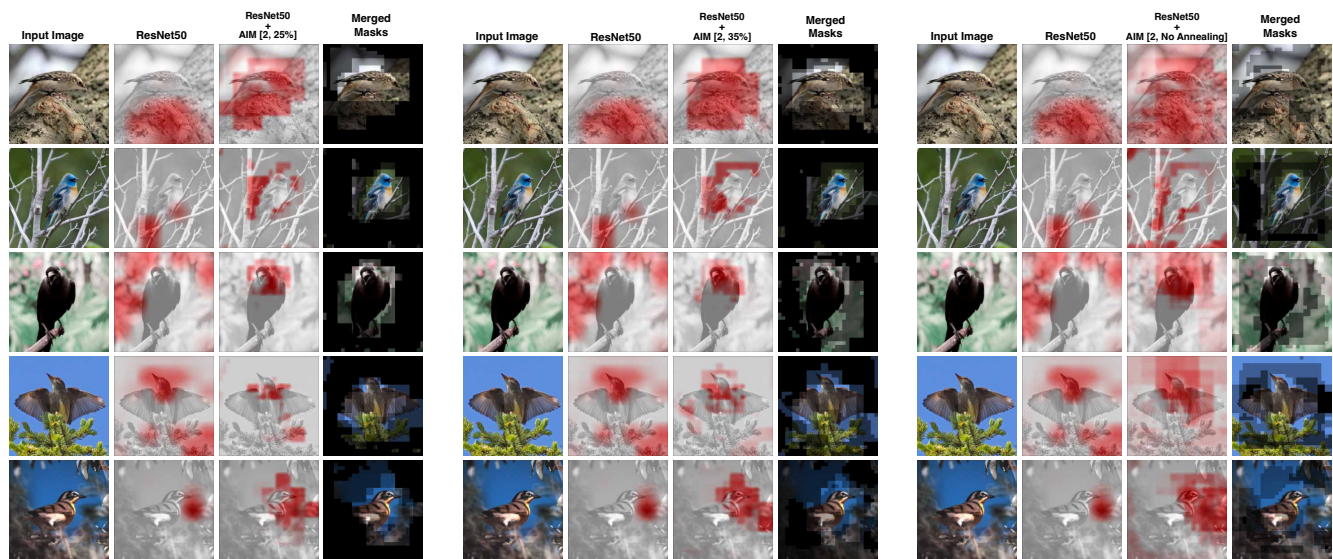


Figure 15. **Qualitative results on the CUB-200 dataset.** Comparison of GradCAM attribution maps between ResNet50+AIM models with different mask active-area thresholds and the vanilla ResNet50, along with the generated merged masks from the ResNet50+AIM model.

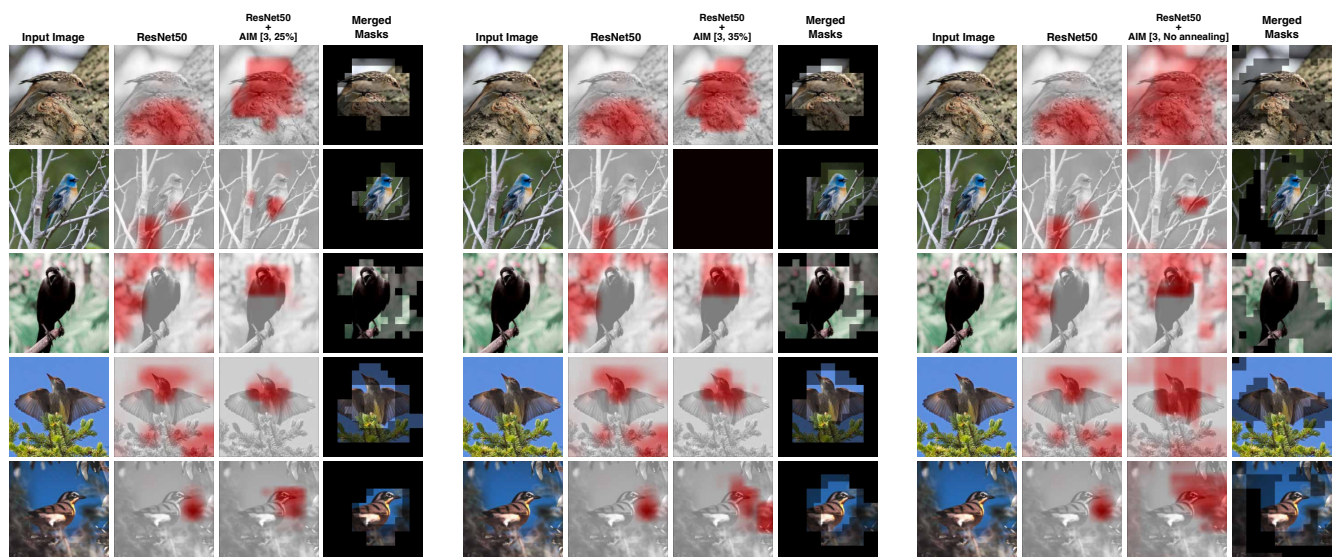


Figure 16. **Qualitative results on the CUB-200 dataset.** Comparison of GradCAM attribution maps between ResNet50+AIM models with different mask active-area thresholds and the vanilla ResNet50, along with the generated merged masks from the ResNet50+AIM model.

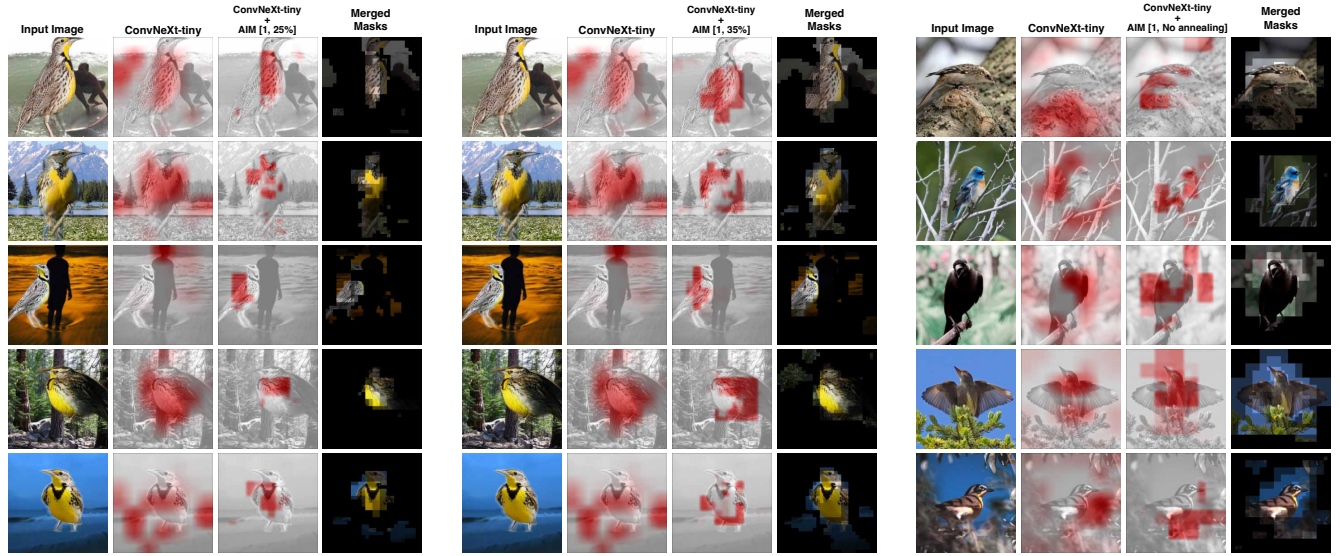


Figure 17. **Qualitative results on the Waterbirds-95% dataset [33].** Comparison of GradCAM attribution maps between ConvNeXt-tiny+AIM models with different mask active-area thresholds and the vanilla ConvNeXt-tiny, along with the generated merged masks from the ConvNeXt-tiny+AIM model.

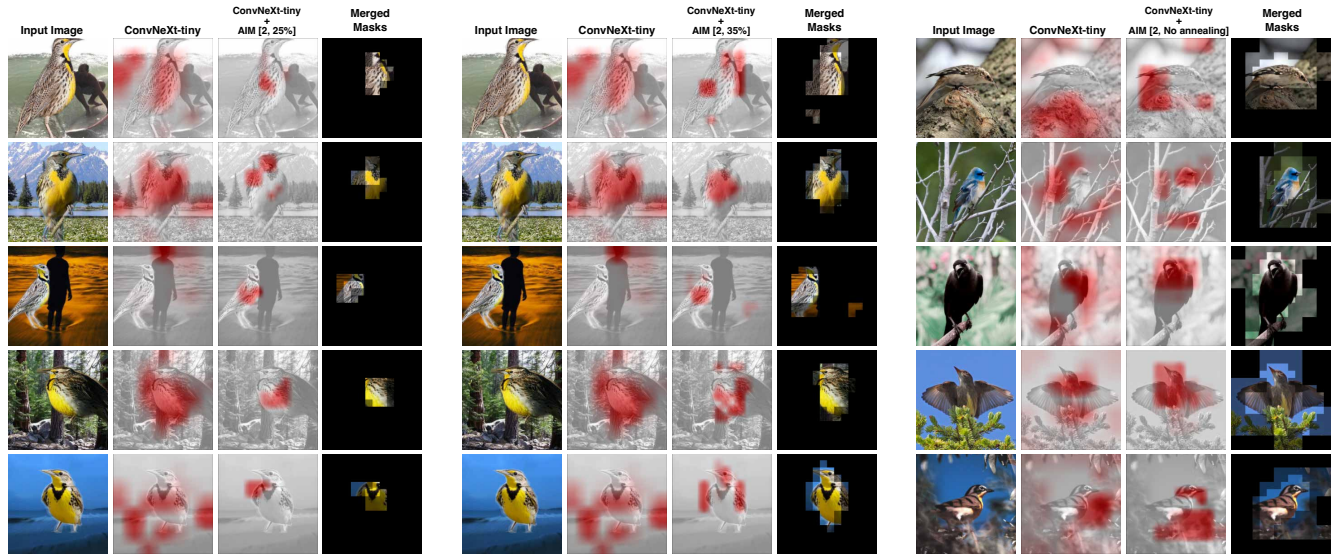


Figure 18. **Qualitative results on the Waterbirds-95% dataset [33].** Comparison of GradCAM attribution maps between ConvNeXt-tiny+AIM models with different mask active-area thresholds and the vanilla ConvNeXt-tiny, along with the generated merged masks from the ConvNeXt-tiny+AIM model.

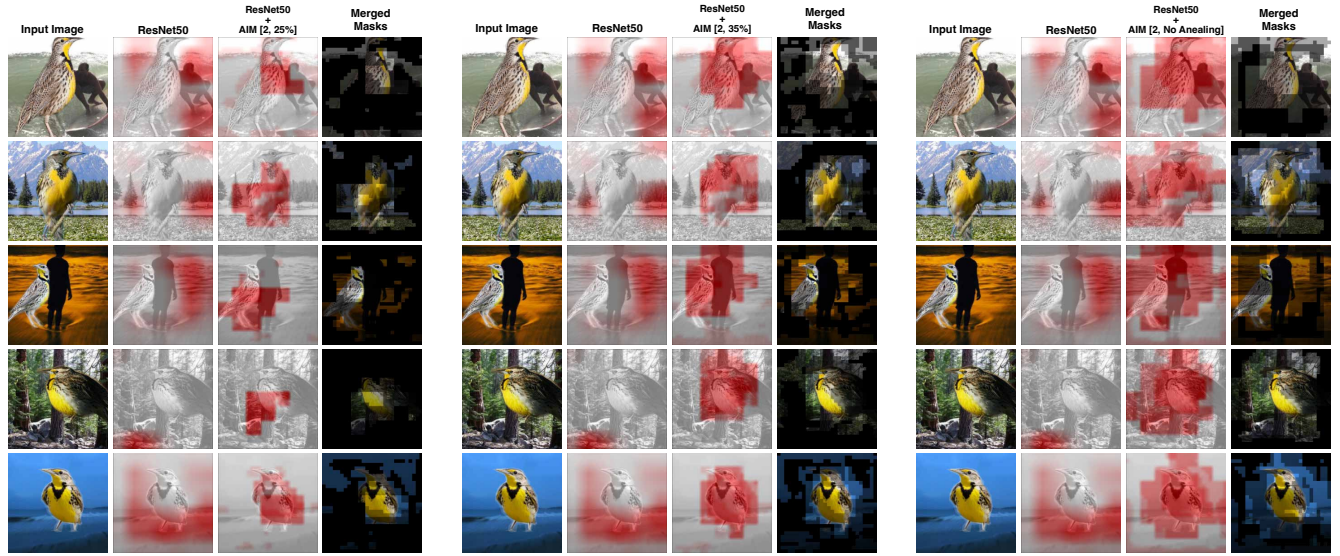


Figure 19. **Qualitative results on the Waterbirds-95% dataset [33].** Comparison of GradCAM attribution maps between ResNet50+AIM models with different mask active-area thresholds and the vanilla ResNet50, along with the generated merged masks from the ResNet50+AIM model.

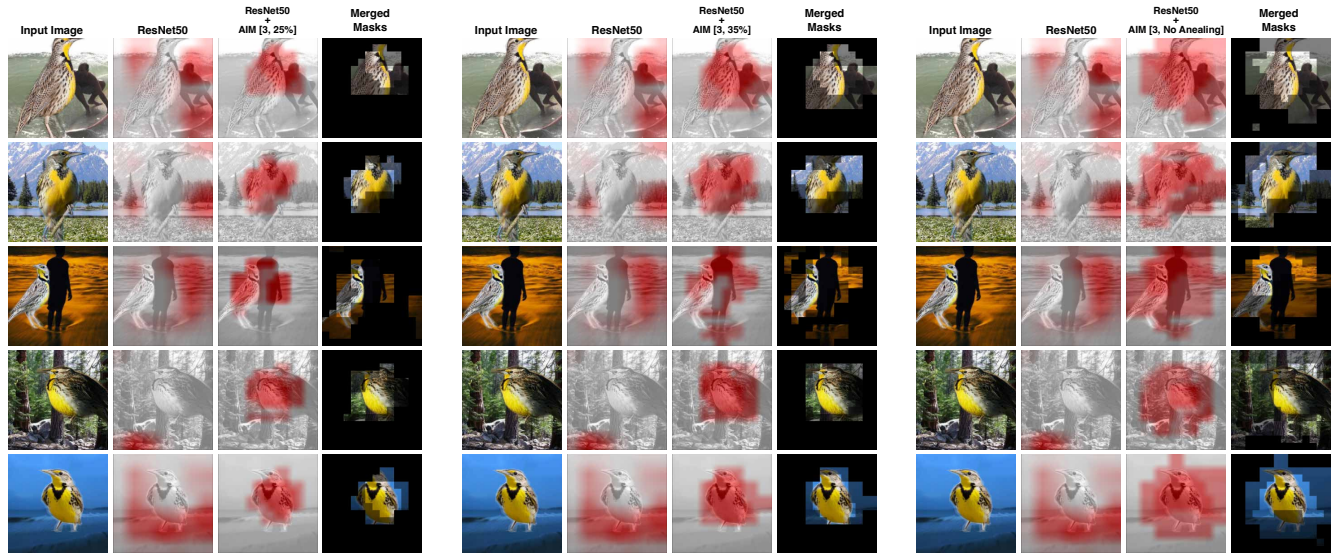


Figure 20. **Qualitative results on the Waterbirds-95% dataset [33].** Comparison of GradCAM attribution maps between ResNet50+AIM models with different mask active-area thresholds and the vanilla ResNet50, along with the generated merged masks from the ResNet50+AIM model.

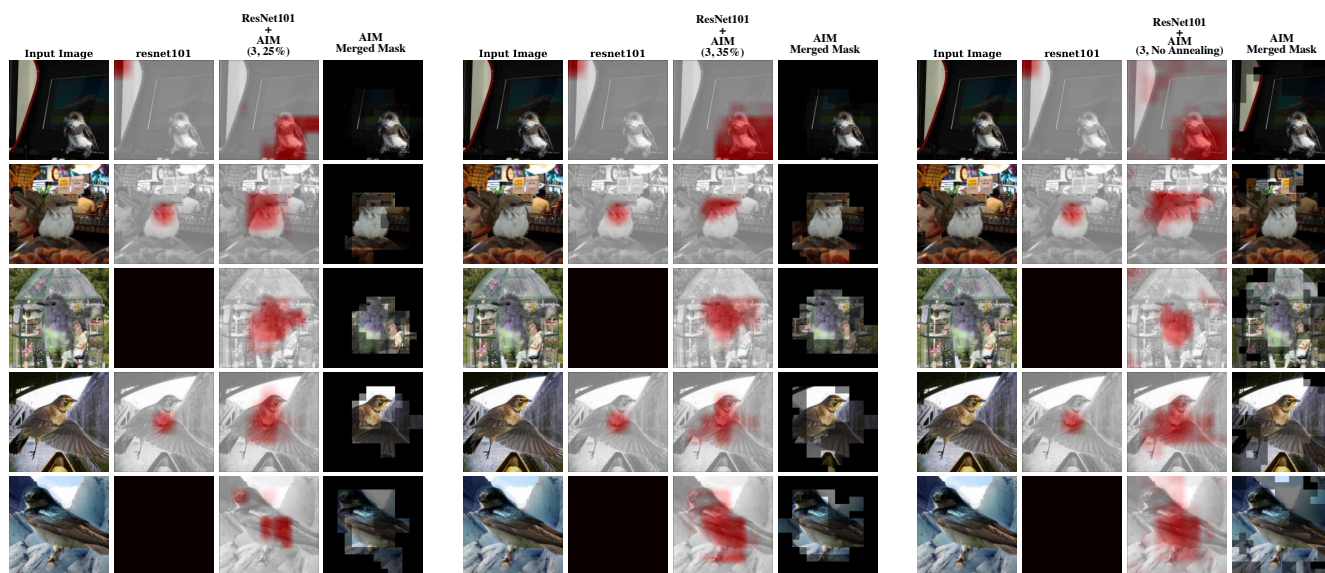


Figure 21. **Qualitative results on the Travelingbirds dataset [19].** Comparison of GradCAM attribution maps between ResNet101+AIM models with different mask active-area thresholds and the vanilla ResNet101, along with the generated merged masks from the ResNet101+AIM model.

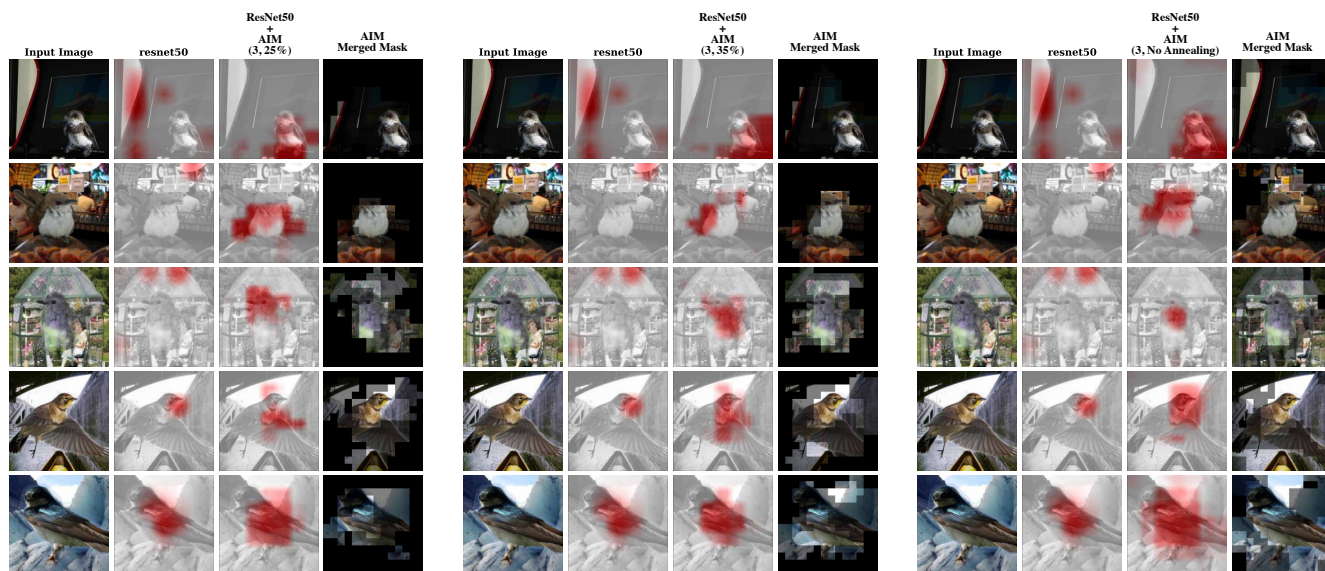


Figure 22. **Qualitative results on the Travelingbirds dataset [19].** Comparison of GradCAM attribution maps between ResNet50+AIM models with different mask active-area thresholds and the vanilla ResNet50, along with the generated merged masks from the ResNet50+AIM model.

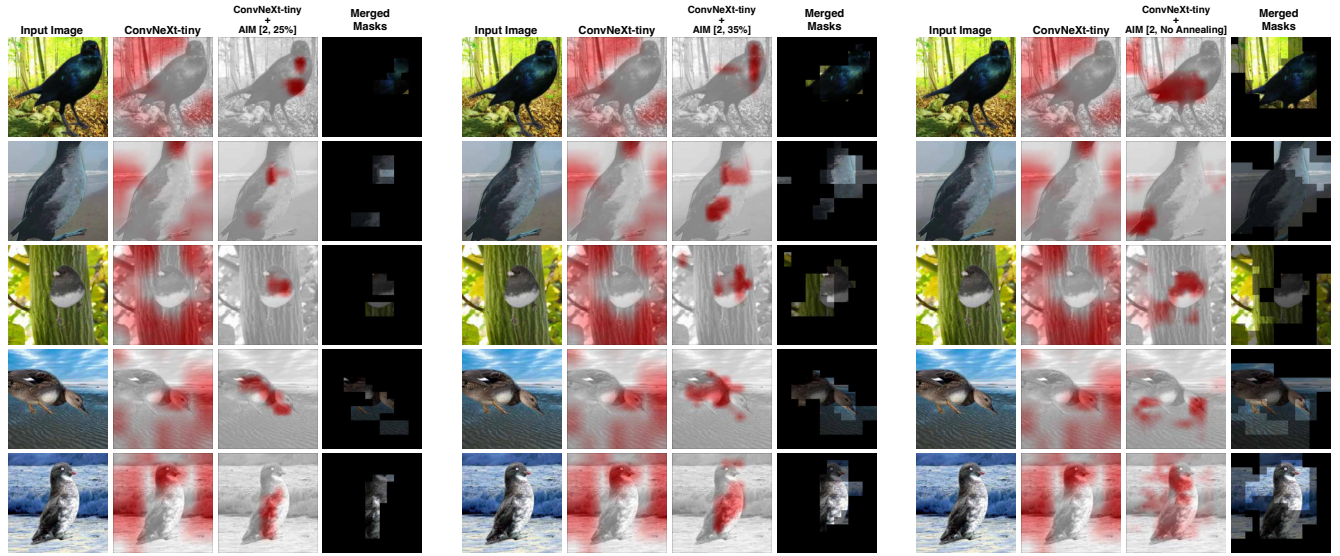


Figure 23. **Qualitative results on the Waterbirds-100% dataset [28].** Comparison of GradCAM attribution maps between ConvNeXt-tiny+AIM models with different mask active-area thresholds and the vanilla ConvNeXt-tiny, along with the generated merged masks from the ConvNeXt-tiny+AIM model.

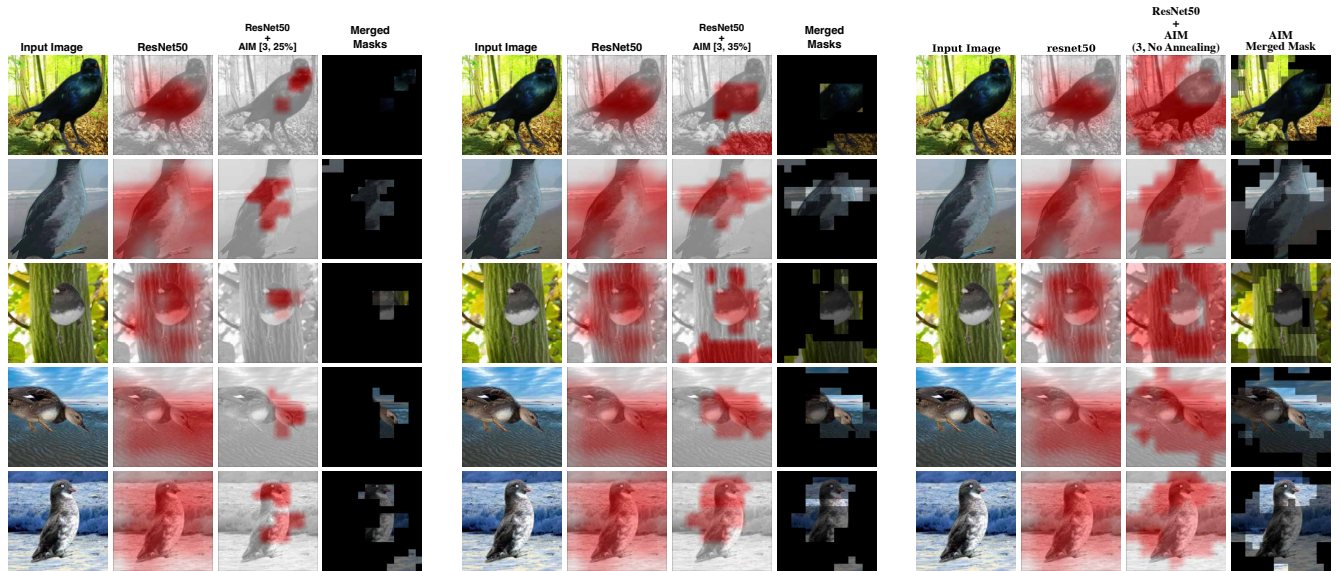


Figure 24. **Qualitative results on the Waterbirds-100% dataset [28].** Comparison of GradCAM attribution maps between ResNet50+AIM models with different mask active-area thresholds and the vanilla ResNet50, along with the generated merged masks from the ResNet50+AIM model.

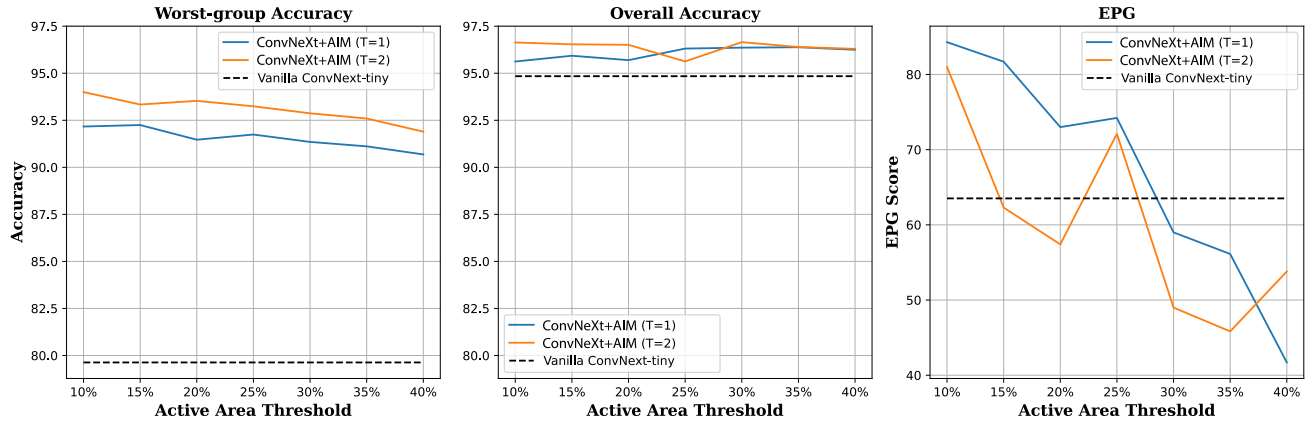


Figure 25. The first figure (on the right) illustrates how the worst-group accuracy of ConvNeXt-tiny+AIM ($T=1$) and ConvNeXt-tiny+AIM ($T=2$) changes with varying active-area thresholds. It shows that as the threshold increases, the worst-group accuracy decreases, while the change in overall accuracy is less pronounced. However, the EPG score also decreases with higher active-area thresholds. Despite this decline, the worst-group accuracy does not experience a significant drop, decreasing by approximately 3%.



Figure 26. Qualitative Comparison between the different architectural configurations of AIM when using masks from the last layer. This figure illustrates the application of masks generated at the last stage (stage #3) and used in the subsequent stages after upsampling.

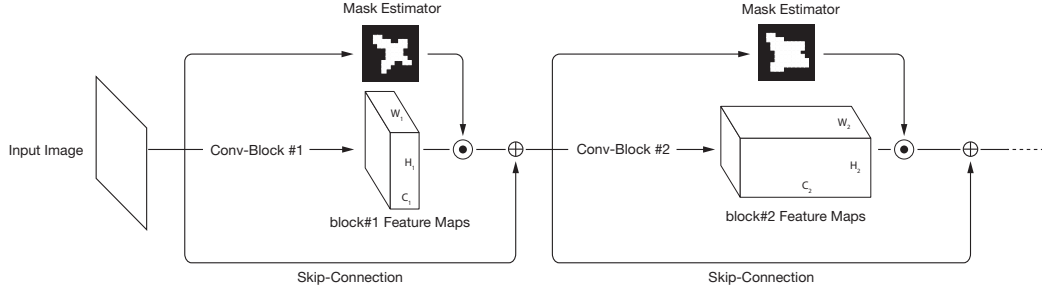


Figure 27. The illustration depicts the architecture of the bottom-up bottlenecking model, adapted from [49], highlighting the flow of data through the model, including skip connections and mask estimators. Each main convolutional block in the network consists of three branches: the first is a mask estimator that employs the Gumbel-softmax trick to predict a binary mask, the second is the original convolutional block, and the third is a skip connection. The generated masks are applied to the output of the convolutional block, resulting in spatially sparse feature maps. Subsequently, the skip connection performs an element-wise summation between these sparse feature maps and the block’s input, transforming the sparse maps back into dense ones.

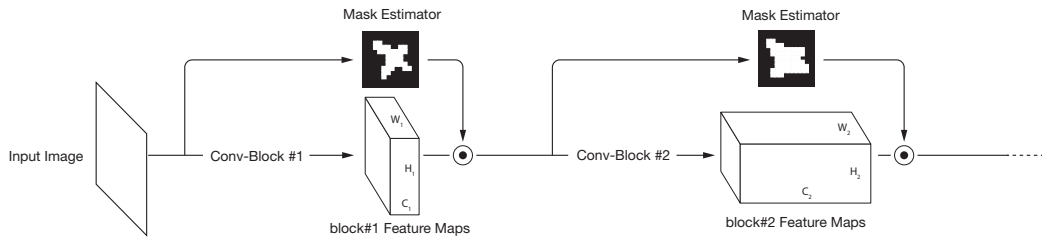


Figure 28. **Bottom-up AIM Network** An illustration showing the same bottom-up bottlenecking model’s architecture from Figure 27 but without the skip-connection to maintain the sparsity of the feature maps.



Figure 29. **Bottom-up AIM Network Fails to Generate Focused Masks** This figure illustrates the masks generated at different stages of the ConvNeXt-tiny backbone within the Bottom-up AIM network. Even when we limit the active area region of the masks to 25% and 35% using mask active-area loss annealing, the masks from the earlier blocks remain fully activated, covering the entire image. Only the mask from the last block effectively focuses on the relevant region. This demonstrates that the Bottom-up AIM network does not produce localized, task-related masks in its earlier stages.

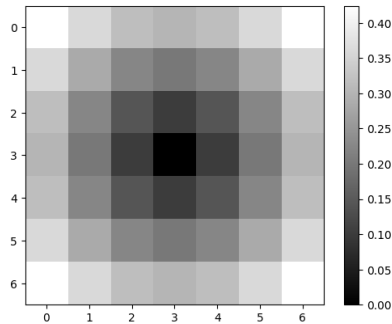


Figure 30. An illustration showing the weight matrix of a convolutional layer with one (7×7) filter, initialized using the proposed idea of weighting the convolutional filter weights according to their spatial distance from the center of the filter kernel. Specifically, we assign larger initial values to the weights corresponding to the edges of the filters than to those closer to the center, effectively biasing the filters to be more responsive to features in the outer areas of the input.

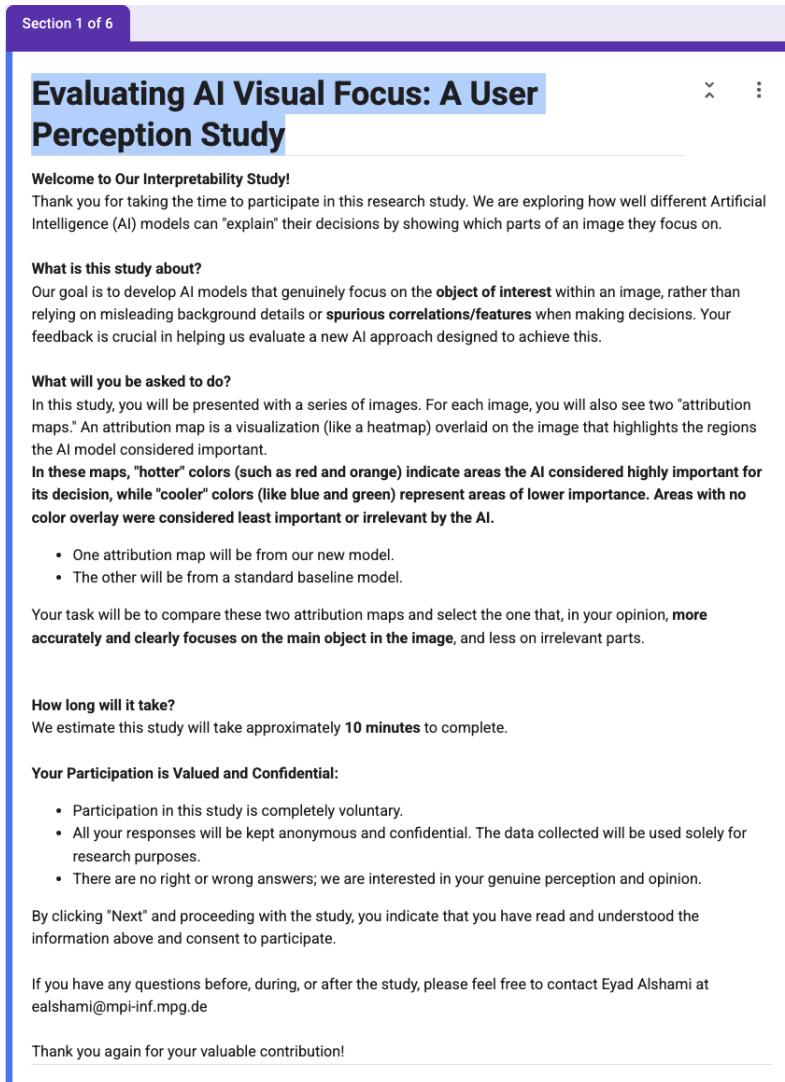
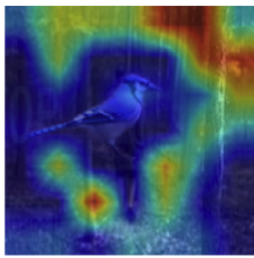




Figure 31. The user study welcome page, which provided participants with initial instructions.

In this example, which attribution map (image A or image B) do you think does a better job of ^{*} focusing primarily on the main object and less on other parts of the image?



☐ A

☐ B

Figure 32. An example trial from our user perception study. Participants were shown the original image (left) and two GradCAM attribution maps from the baseline ConvNeXt (right) and our AIM-equipped model (center). In this case, the baseline model is distracted by the spurious forest background, while our method correctly localizes the waterbird. This clear distinction in focus led users to significantly prefer our model's explanations.