AIM: an Auto-Augmenter for Images and Meshes

Vinit Veerendraveer Singh and Chandra Kambhamettu
Video/Image Modeling and Synthesis (VIMS) Lab, Department of Computer and Information Sciences,
University of Delaware, Newark, Delaware, United States of America, 19716
{vinitvs,chandrak}@udel.edu

Abstract

Data augmentations are commonly used to increase the robustness of deep neural networks. In most contemporary research, the networks do not decide the augmentations; they are task-agnostic, and grid search determines their magnitudes. Furthermore, augmentations applicable to lower-dimensional data do not easily extend to higher-dimensional data and vice versa. This paper presents an auto-augmenter for images and meshes (AIM) that easily incorporates into neural networks at training and inference times. It jointly optimizes with the network to produce constrained, non-rigid deformations in the data. AIM predicts sample-aware deformations suited for a task, and our experiments confirm its effectiveness with various networks.

1. Introduction

Deep neural networks are prevailing in various computer vision tasks. They are commonplace for the analysis of digital images [15, 16, 38] and 3D graphics [8, 14]. These networks try to emulate human cognition in a computerized environment. However, despite the success of deep learning in recent years, it is still not as robust as human vision.

Learning methods for vision-based tasks need to disassociate between *what* an object looks like and *where* it lies in space. To this end, it is common to pre-process input data to neural networks with augmentation approaches. Some augmentations make neural networks more tolerant of geometric changes in data. For instance, augmentation techniques such as affine transformations, random horizontal flipping, and random cropping are standard for image processing. For mesh analysis, jittering of mesh elements is performed along with affine transformations. These augmentations also form the basis of other advanced data augmentation strategies [5, 11, 21, 32, 42, 55] and frameworks [2, 54]. The augmentations mentioned above do not directly participate in the learning process and are not dependent on a task's objective. Thus, they are non-learnable and task-agnostic.

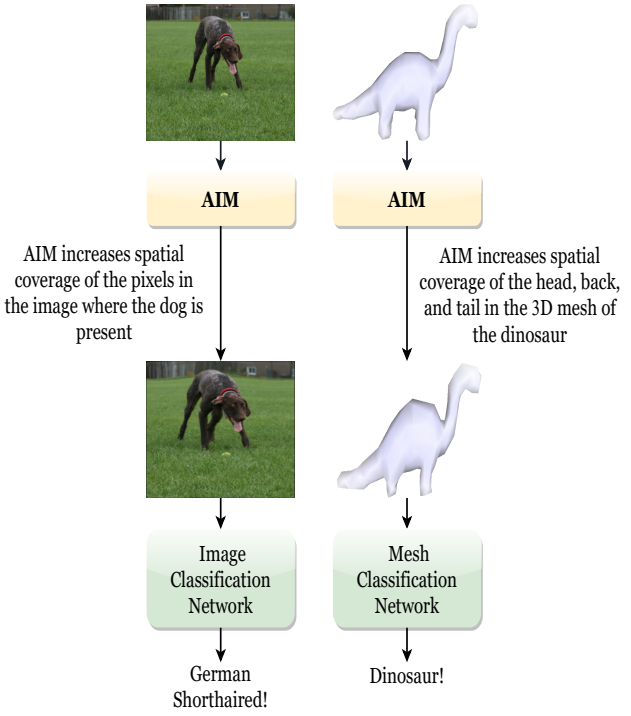

Figure 1. AIM performs non-rigid deformations on the input data to a task network during training and testing. AIM learns to detect critical regions in the samples to solve a task and increases their spatial coverage. As shown, it applies to both images and meshes.

Conversely, task-aware augmentation approaches [5, 12, 20, 26, 37] jointly optimize with neural networks. A common theme to this set of approaches is to sequentially learn which transformations suit a task, where to employ them, and to what extent they should be applied. However, at present, many learnable augmentation methods are constrained to the dimensions of their underlying representations. Methods applicable to 2D images are either unsuitable or have no clear way to extend to 3D data and vice versa. In this work, we build upon these insights and shortcomings to propose an Auto-Augmenter for Images and Meshes (AIM).

Eye movements in human beings are primarily categorized into four categories: fixation [39], saccades [10], stabilization [4], and smooth pursuit [24]. By fixating eyes at specific locations, the human visual system can enhance their resolution to process fine spatial details. AIM closely imitates human fixation. It is implemented as a data preprocessor to jointly optimize with existing neural networks for image and mesh analysis. First, AIM infers regions in the images and meshes that contain critical information to solve a task. Then, it increases the spatial resolution of these regions and simultaneously reduces spatial coverage of non-crucial areas. A visual illustration of this process is in Fig. 1. The main contributions of this paper are:

- AIM's novel and differentiable spatial warper.
- An attention module for graph data.
- A novel directional consistency loss to constrain deformations produced by AIM.
- Experiments on multiple data sets for image classification, mesh classification, and mesh segmentation.

2. Related Works

We split existing data augmentation approaches for neural networks into two broad categories: non-learnable augmentations and learnable augmentations. Methods in both categories apply to digital images or 3D geometric data such as meshes and point clouds. The non-learnable augmentations have no trainable parameters, and their ideal magnitudes depend on an extensive grid search. On the other hand, the learnable approaches contain trainable parameters and employ augmentations suited for a task.

2.1. Non-Learnable Augmentations

Propagating randomly cropped image regions train a neural network in disentangling textural features from their positions. Image rotation and flipping increase neural networks' invariance to varying poses of objects. Random Erasing [56] and CutOut [6] replace continuous regions in images with random values to achieve tolerance to occluded views of objects. Color jitter randomly changes image brightness, contrast, or saturation during training. Dropout [18] regularizes a neural network by randomly dropping neuron activations while training. It can be considered as feature map augmentation. CutMix [55] replaces image regions with random patches from other images to retain the regularization effect of regional dropout strategies. Cut-Thumbnail [52] is similar to CutMix. But unlike CutMix, it replaces image regions with thumbnails. GridMask [1] balances deletion and conservation of regions by dropping discontinuous areas in images. Like GridMask, MeshCut [21] overlays a square mesh grid upon images to drop non-continuous parts. AugMix [17] increases robustness to corruptions in data distributions. All these non-learnable augmentation techniques improve the representational capacity of neural networks. However, most regularize the networks by dropping regions instead of focusing on the correct ones. Also, it is unclear how these techniques apply to 3D data. In particular, photometric augmentation techniques find no use in augmenting purely geometric data.

Meshes and point clouds are commonly augmented by scaling, translating, and jittering the position of vertices [8, 35]. However, there are only a few advanced augmentation techniques for 3D data. PointMixup [3] takes inspiration from the image domain and interpolates point clouds and their corresponding labels. Mix3D [32] creates new scenes by combining two augmented scenes. Such a mixup allows generalization beyond contextual priors. PatchAugment [42] augments feature maps of neural networks [35, 50] operating on point clouds. MeshCNN [14] augments meshes by performing anisotropic scaling and random edge flips.

2.2. Learnable Augmentations

Spatial Transformer Networks (STN) [20] learn and perform various transformations (affine, projective, and thin plate spline) on images. However, extreme image transformations can occur for thin plate spline based STN. STN can also crop regions in images when the determinant of the left 2×2 sub-matrix in the affine transformation matrix has a magnitude less than one. However, no such constraint was explicitly imposed. Some methods augment input images to a network by increasing spatial coverage of certain image regions. Saliency Sampler (SS) [37] increases spatial coverage of task-aware salient regions in images. These salient regions are inferred with a pre-trained Convolutional Neural Network (CNN) [16]. Millions of trainable parameters in this CNN update during network backpropagation, making their approach computationally expensive. Also, it is unclear how SS extends to 3D data. The other works [7, 22, 44, 53] derived from SS also suffer from similar limitations. Marin *et al.* [29] formulated a content-adaptive downsampling method that favors locations near semantic boundaries of classes prior to image downsampling. However, this method is limited by selecting downsampling locations to a manually designed sampling objective. AutoAugment [5] does not provide any novel augmentation approaches. Instead, it uses a search algorithm to determine the best training policy. KeepAugment [12] learns to preserve important regions from regional dropout. For 3D point clouds, PointAugment [26] augments point clouds using an adversarial training strategy. Currently, no learnable augmenter exists for mesh data to the best of our knowledge. This work introduces an augmenter with very few trainable parameters, and it applies to images and meshes.

3. Overview and Notations

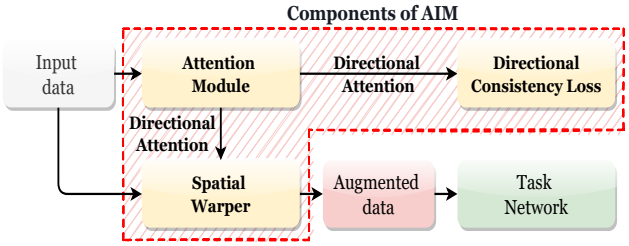

Figure 2. An overview of AIM. AIM's components jointly optimize with a task network to augment the input data.

The setup of deep learning methodologies for supervised learning tasks is standard. We interchangeably refer to the neural network for a task as a task network. A task network ($\mathcal{T}$) first learns on samples in the set of training data. We denote the training set as $\mathcal{D}_{train} = \{(x^{(i)}_{train}, y^{(i)}_{train})\}^{m}_{i=1}$, where $x^{(i)}_{train}$ is an ith sample from the m training samples. $y^{(i)}_{train}$ is the task label of $x^{(i)}_{train}$. After training, $\mathcal{T}$ attempts to predict task labels on a testing set $\mathcal{D}_{test}$. As illustrated in Fig. 2, incorporating AIM with $\mathcal{T}$ is straightforward. At the commencement of the training phase, AIM augments every sample in a single batch of $\mathcal{D}_{train}$ and propagates it to $\mathcal{T}$. Based on the loss from the task network and the directional consistency loss, the trainable parameters in AIM and $\mathcal{T}$ are optimized together. Then, AIM refines the augmentations for the next mini-batch, and this process continues up until $\mathcal{T}$ reaches convergence. During the test phase, AIM augments all samples in $\mathcal{D}_{test}$ to produce an augmented testing set $\mathcal{D}'_{test}$. Finally, $\mathcal{D}'_{test}$ is sent forth to $\mathcal{T}$ for evaluation. Note that AIM learns augmentations on a per-sample basis.

In this paper, we apply AIM to supervised learning tasks. A key aspect of AIM is to jointly optimize with a task network for augmenting input images and meshes effectively. It consists of three major stand-alone components: a spatial warper, an attention module, and a directional consistency loss. The spatial warper performs non-rigid deformations on the input data. It adaptively increases or decreases the coverage of different regions within the input. The attention module decides the locations and magnitude of the deformations. Finally, the directional consistency loss constrains the locations from the attention module. We present details on each component in the next section.

4. Method

We first elucidate upon AIM's spatial warper in section 4.1 and demonstrates how to use it for image and mesh data. Then, section 4.2 presents AIM's attention module implemented as a graph convolutional network. Next, we formulate the directional consistency loss in section 4.3 and introduce our end-to-end strategy to incorporate AIM with a task network. Lastly, we provide implementations details in section 4.4.

4.1. Spatial Warper

Recent research [19, 20, 37, 51] shows that information in certain regions in the data contributes more to a neural network's decision-making ability than the information in the remaining areas. Thus, intuitively, it is reasonable to assume that fixating on the informative regions should improve a network's ability to make task-based decisions. The spatial warper ($\mathcal{SW}$) in AIM fixates on different regions in the data by increasing their spatial coverage. Note that the spatial warper does not detect the highly informative areas beneficial to the task by itself.

The $\mathcal{SW}$ operates on the graph data structure. The topology of a graph ($\mathcal{G}$) is given by its set of vertices $\mathcal{V} = \{v_i\}^{n_v}_{i=1}$ and its set of edges $\mathcal{E} = \{e_i\}^{n_e}_{i=1}$. Here, v_i and e_i represent an ith instance of the vertices and edges, respectively. n_v and n_e represent the cardinality of $\mathcal{V}$ and $\mathcal{E}$, respectively. Each e_i connects to two vertices (v^0_i and v^1_i). The $\mathcal{SW}$ produces non-rigid deformations in $\mathcal{G}$ by adjusting the position of its vertices. This adjustment is achieved by changing the edge length of each e_i. A simple brute-force approach to locally change edge lengths can be to multiply a unique coefficient by the length of each edge. However, vertices in a graph usually connect to multiple other vertices, and an unconstrained multiplication of coefficients can cause extreme deformations. Moreover, in the case of visual data such as images or meshes, deformations would appear unnatural and important regions would deform beyond recognition. Therefore, it is necessary to conserve the global shape of $\mathcal{G}$ to a certain extent. Thus, the $\mathcal{SW}$ minimizes deformation in the overall shape of $\mathcal{G}$ while deforming each edge locally. Such a constrained deformation of $\mathcal{G}$ can be achieved by minimizing the energy function E in equation 1:

$$E = \sum_{e_i \in \mathcal{E}} \left| \frac{(x^0_i - x^1_i) - \gamma_i (v^0_i - v^1_i)}{v^0_i - v^1_i + \epsilon^{-1}} \right|^2, \qquad (1)$$

where:

$$\begin{aligned} v^0_i, v^1_i &= \text{initial location of endpoints for an edge } e_i \\ &\quad \text{along an independent direction in space} \\ x^0_i, x^1_i &= \text{post-minimization locations for endpoints of} \\ &\quad e_i \text{ along an independent direction in space} \\ \gamma_i &= \text{deformation coefficient for edge } e_i \\ \epsilon^{-1} &= \text{a large number} \end{aligned}$$

The deformation coefficient (γ_i) for an edge e_i is computed as per equation 2. It is a linear combination of the deformation factor (Δ) and the sensitivity (s_i) of e_i to deformation. Δ is a scalar with constant magnitude. Both Δ and $s_i \in \mathcal{S}$ are between zero and one. Note that a higher s_i does

not necessarily imply that an edge will expand more than another edge with a lower s_i. The extent of the deformations is governed by α and β. Thus, the $\mathcal{SW}$ can adaptively expand or shrink the spatial coverage of the edges independent of the magnitude of their sensitivity. In the remainder of this paper, we denote the set of deformation sensitivity of all edges in $\mathcal{G}$ as $\mathcal{S}$.

$$\gamma_i = \alpha s_i + \beta(1 - s_i) \begin{cases} \alpha = 1, & \text{if } \beta = \Delta \\ \beta = 1, & \text{if } \alpha = \Delta \end{cases} \tag{2}$$

In equation 1, x_i^0 and x_i^1 are the updated positions of edge endpoints for an edge e_i after minimization. Thus, prior to minimization, their locations are unknown. Since $\mathcal{V}$ constitutes the endpoints of all edges, the energy function E can also be expressed in terms of vertices. The reformulation of E is represented as E' in equation 3.

$$E' = \sum_{v_i \in \mathcal{V}} \sum_{\substack{v_j \in \\ \mathcal{N}(v_i)}} \frac{x_i^2 - 2x_i x_j - 2\gamma_i x_i (v_i - v_j) + c}{(v_i - v_j + \epsilon^{-1})^2} \tag{3}$$

where:

v_i = location of an ith vertex along an independent direction in space
$\mathcal{N}(v_i)$ = immediate vertex neighborhood of v_i
v_j = a vertex neighbor of v_i
x_i = location of v_i after minimizing E'
x_j = location of v_j after minimizing E'
γ_i = deformation coefficient for the edge e_i between v_i and v_j
c = a constant
ϵ^{-1} = a large number

The goal of the $\mathcal{SW}$ is to minimize the overall movement of the vertices while resizing the edges. We can achieve this by taking partial derivatives of E' with respect to each x_i and equating it to zero, as shown in equation 4.

$$\frac{\partial E'}{\partial x_i} = \sum_{\substack{v_j \in \\ \mathcal{N}(v_i)}} \frac{2x_i - 2x_j - 2\gamma_i(v_i - v_j)}{(v_i - v_j + \epsilon^{-1})^2} = 0 \tag{4}$$

We obtain a linear equation in terms of an unknown vertex x_i and its unknown vertex neighbors from equation 4. Once linear equations for all vertices are obtained, the minimization of E' can be expressed as a sparse linear system of the form $AX = B$. The matrix $A \in \mathbb{R}^{\mathcal{V} \times \mathcal{V}}$ and a vector $B \in \mathbb{R}^{\mathcal{V} \times 1}$ are known and computed as per equation 5. A_{IJ} represents a row in matrix A for a vertex x_i and its vertex neighbors. The vector $X = \{x_i\}_{i=1}^{n_v}$ is unknown and can be approximated through a sparse linear solver.

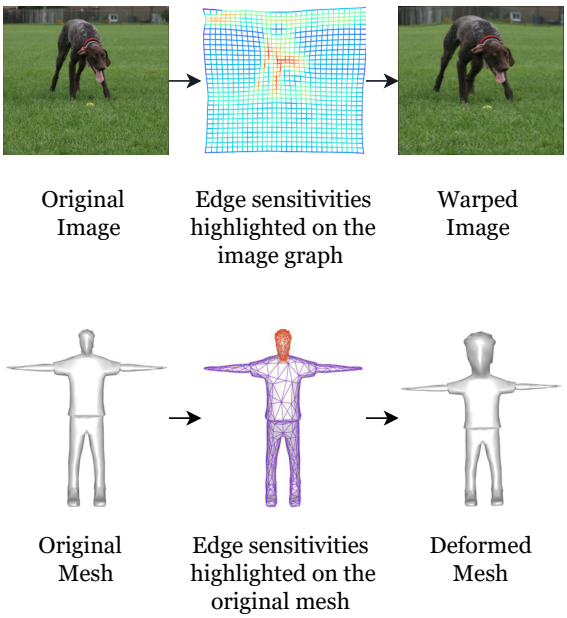

Figure 3. The spatial warper applies to images and meshes. Edges with higher sensitivity are highlighted in red. From the top row, we can observe that the spatial warper increases the spatial coverage of pixels in the original image by shrinking the edges with higher edge sensitivity. For meshes (bottom row), the spatial warper expands edges with higher sensitivity. [Best viewed in color]

$$A_{IJ} = \begin{cases} \sum_{v_j \in \mathcal{N}(v_i)} \frac{2}{(v_i - v_j + \epsilon^{-1})^2}, & \text{if } I = J \\ \frac{-2}{(v_i - v_j + \epsilon^{-1})^2}, & \text{if } I \neq J \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$B_I = \sum_{\substack{v_j \in \\ \mathcal{N}(v_i)}} \frac{2\gamma_i(v_i - v_j)}{(v_i - v_j + \epsilon^{-1})^2}$$

4.1.1 Spatial Warping of Images and Meshes

To apply the spatial warper to images, we consider that the underlying representation of an image is a graph. The pixels of an image are considered nodes of the graph. The edges between these nodes are defined in an ad-hoc fashion. In AIM, edges only exist between the horizontal and vertical neighbors of a pixel. The $\mathcal{SW}$ is thus applicable to images. For images, β in equation 2 is one. Therefore, edges with higher sensitivity will shrink more than edges with a lower sensitivity. Interpolation on an original image with such a deformation grid will increase the spatial coverage of pixels connected to the edges with higher sensitivity. This phenomenon can be observed in the top row of Fig. 3.

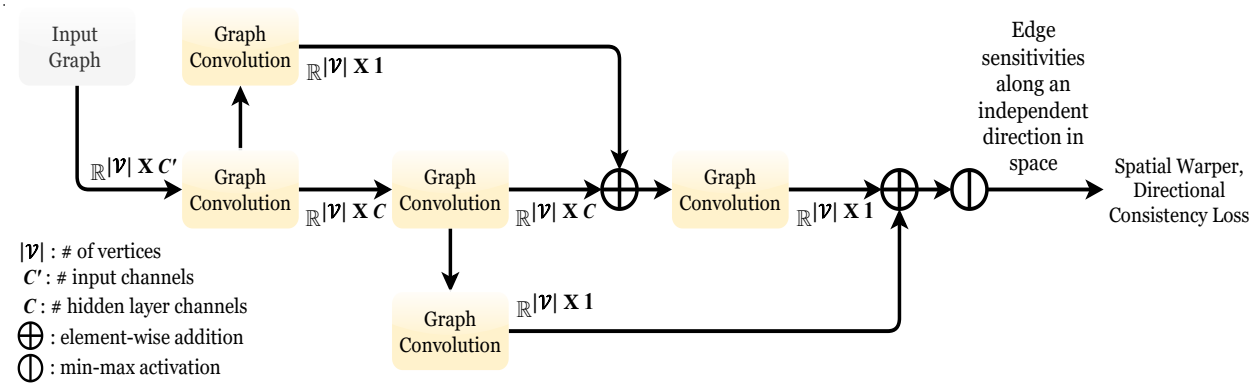

Figure 4. Design of AIM's Attention Module. The attention module infers edge sensitivities from graph convolutional layers. Thus, the number of training parameters are low, and sensitivities can be inferred on Euclidean as well as non-Euclidean data.

In the case of mesh data, α in equation 2 is one. Setting α to one allows the nodes connected to edges with higher deformation sensitivity to increase their spatial coverage. Note that post-deformation, the image deformation grid and meshes are normalized to fit a sphere of radius one and are centered at the origin.

4.2. Attention Module

As mentioned earlier, the spatial warper cannot determine informative regions in the input data by itself. Therefore, any attention mechanism that aids the spatial warper and a task network to identify the informative regions in the input data must follow three key criteria:

- Since the spatial warper operates on graphs, the attention mechanism must also operate on graphs.
- The attention mechanism learns edge sensitivities in synergy with a task.
- The attention mechanism can learn edge sensitivities in independent directions in space.

Convolutional Neural Networks (CNNs) are currently the de-facto choice for analyzing images and other Euclidean data. Numerous attention mechanisms [19, 51] also exist for CNNs. However, CNNs do not generalize well to non-Euclidean data such as meshes and graphs. Some recent methods [8, 14, 31, 45, 46] have designed specialized convolutional operators for meshes. However, they made strict assumptions about the mesh geometry (manifoldness, etc.). AIM's attention module is realized as a graph convolutional network to conform to the criteria mentioned above. The architecture of this attention module is shown in Fig. 4. The attention module infers edge sensitivities ($\mathcal{S}$) along each independent direction in the input data. For example, in the case of images, edge sensitivities are learned separately along the x-axis ($\mathcal{S}_x$) and the y-axis ($\mathcal{S}_y$). $\mathcal{S}$ is constrained between zero and one through min-max normalization.

4.3. Directional Consistency Loss

It is conducive for a task network reasoning about visual data to disambiguate between the appearance of regions in the data and where the regions lie in space. As the spatial warper applies along independent directions in space, deformations will vary along each direction. However, ideally, deformations should not vary in each direction. For example, the highly informative parts in an image should be deformed more than the non-informative areas in both the x-axis and the y-axis. AIM achieves this constraint through directional consistency loss ($\mathcal{L}_{dc}$). It enforces the attention module's embeddings (edge sensitivities) to be consistent along each independent direction in space. It maximizes the cosine similarity between the embeddings in the different directions. For a mini-batch of N images, with spatial attention $\mathcal{S}_x$ and $\mathcal{S}_y$ along the x-axes and y-axes, respectively, the directional consistency loss is computed according to equation 6. It is clear from equation 6 when $\mathcal{S}_x$ and $\mathcal{S}_y$ are completely dissimilar, $\mathcal{L}_{dc}$ will be two. When $\mathcal{S}_x$ and $\mathcal{S}_y$ are the same, $\mathcal{L}_{dc}$ will be zero.

$$\mathcal{L}_{dc} = \mathcal{L}_{dc}^{xy} = 1 - \frac{1}{N} \sum \frac{\mathcal{S}_x \cdot \mathcal{S}_y}{\|Sx\| \cdot \|Sy\|} \tag{6}$$

For a mini-batch of meshes, loss in the directional consistency is computed as per equation 7 below:

$$\mathcal{L}_{dc} = \mathcal{L}_{dc}^{xy} + \mathcal{L}_{dc}^{yz} + \mathcal{L}_{dc}^{zx}, \tag{7}$$

where $\mathcal{L}_{dc}^{xy}$ is the directional consistency loss between $\mathcal{S}_x$ and $\mathcal{S}_y$. $\mathcal{L}_{dc}^{yz}$ is computed between $\mathcal{S}_y$ and $\mathcal{S}_z$, and similarly $\mathcal{L}_{dc}^{zx}$ is the loss in the directional consistency between $\mathcal{S}_z$ and $\mathcal{S}_x$. The directional consistency loss is task-agnostic. With the spatial warper, attention module, and directional consistency loss clearly defined, AIM is thus realized. We trained and evaluated various task networks with AIM in an end-to-end manner for different supervised learning tasks in section 5.

4.4. Implementation Details

We implemented AIM using PyTorch [34], PyTorch Geometric [9], Torch Sparse Solve [25], and PyTorch3D [36]. The graph convolutions in the attention module is the GraphSAGE [13] operator. The number of hidden layer channels ($\boldsymbol{C}$) is 64 for images and 32 for meshes. We perform bilinear interpolation of images by borrowing the grid sampler introduced in STN [20]. Edge sensitivity is inferred by averaging features of the vertices at their endpoints. When applying AIM to images, we set the edge sensitivity at the border of the images to the minimum value in ($\mathcal{S}$). Thus, border pixels are reasonably conserved after warping. Our code is available at https://github.com/VimsLab/AIM.

5. Experiments

Fine-grain visual categorization of images is a fundamental task in computer vision. Likewise, classification and segmentation of meshes are fundamental in 3D shape analysis tasks. We applied AIM to classify images and meshes in multiple data sets. We also evaluated AIM for the segmentation of meshes.

5.1. Experimental Setup

We applied AIM for the fine-grain classification of images in the CUB-200 [49] and the Oxford-IIT Pets [33] data sets. The CUB-200 data set contains images of 200 bird species. The birds appear at different scales and poses and are not tightly cropped. The training set contains approximately 6k images, and the testing set contains about 5.8k images. The Oxford-IIT Pets only contains images of cats and dogs. There are 37 pet categories in this data set, and the images exhibit large variations in scale, pose, and lighting. It contains roughly 7.3k images, with about 200 images per class. The sizes of the training and testing sets are near-equal. We trained ResNet [16] and EfficientNet [48] with AIM to classify the images in these data sets. Both networks were trained with the Stochastic Gradient Descent optimizer with a momentum of 0.9 and weight decay set to 1e-4. For the ResNet models, the learning rate was 0.01, and the batch size was 128. For EfficientNet, the learning rate was 0.001, and the batch size was 48. The spatial warper sub-sampled 224×224 images from 700×700 images for CUB-200. For Oxford-IIT Pets, 224×224 images were sub-sampled from 448×448 images. Δ was set to 0.72.

For mesh classification, AIM was employed with MeshNet [8] and MeshNet++ [46]. The attention module's input features for mesh data were the vertex normals, curvature, or one-ring neighborhood area around the vertices. Both networks were evaluated on the McGill 3D Shape Benchmark (MSB) [43] and the SHREC-11 [28] data sets. MSB consists of 458 meshes belonging to 19 classes. The number of vertices varies across this data set. The SHREC-11 data set consists of 600 mesh models from 30 classes. We follow a similar training strategy mentioned in MeshNet++, except that we set the learning rate to 0.001 instead of 0.0002 for MSB. Δ was set between 0.7 to 0.9. Finally, we trained DiffusionNet [41] with AIM to segment body parts in the human body data set [30]. Δ was set to 0.9. We obtained Δ for all models through a grid search.

5.2. Quantitative Experimental Results

5.2.1 Image Analysis

We first evaluated AIM with the following three variants of ResNet: ResNet-18, ResNet-34, and ResNet-50. As shown in Table 1, training these networks with AIM gave higher accuracies than the baseline models. For the CUB-200 data set, the accuracies were significantly higher. We also evaluated AIM against other augmentation techniques such as Random Erasing [56] and Saliency Sampler [37]. We compared AIM against Random Erasing to check if eliminating regions in the images is conducive to fine-grain visual categorization. We also compared AIM against Saliency Sampler (SS) since it also enlarges the spatial coverage of regions beneficial to tasks like AIM. Both Random Erasing and Saliency Sampler are trained with the same experimental setup without bells and whistles.

We observed that randomly erasing the regions within the input data was not conducive to the fine-grain classification of images. A possible explanation for low accuracies using Random Erasing could be that other objects do not heavily occlude the animals' images in both data sets. Thus, Random Erasing might have removed regions essential for a classifier to make correct decisions. Overall, Saliency Sampler gives slightly lower yet comparable accuracies to AIM. This difference in the accuracies can be attributed to Saliency Sampler's inability to constrain deformations within the border of an image. For example, if highly informative regions lie at image borders, SS will partially eliminate those regions. Another reason for lower accuracies in SS could be that its inferred saliency is only constrained by the loss function of the task network. Whereas in AIM, the edge sensitivities are also constrained through the directional consistency loss. We also compared the number of learnable parameters (#Params) in millions (M) between AIM and SS in Table 1. It can be observed that AIM has nearly the same number of parameters as the baseline models and has a significantly lower number of parameters than SS. We observed similar accuracies for the above-discussed methods for EfficientNet as well.

5.2.2 Mesh Analysis

We also evaluated MeshNet++ and MeshNet with AIM for classifying meshes in MSB and the split-16 of SHREC-11.

Method	CUB-200 Acc(%)	Oxford-IIT Pets Acc(%)	#Params in M
ResNet-18	78.3	91.6	11.3
ResNet-18	79.4	90.9	11.3
+ **Random Erasing**	(↑ 1.1)	(↓ 0.7)	
ResNet-18	78.7	91.9	22.9
+ **SS**	(↑ 0.4)	(↑ 0.3)	
ResNet-18	**79.4**	**91.9**	11.3
+ **AIM** (Ours)	(↑ 1.1)	(↑ 0.3)	
ResNet-34	79.8	92.5	21.4
ResNet-34	**80.6**	91.5	21.4
+ **Random Erasing**	(↑ 0.8)	(↓ 1.0)	
ResNet-34	77.7	91.3	33.0
+ **SS**	(↓ 1.1)	(↓ 1.2)	
ResNet-34	80.4	**93.0**	21.4
+ **AIM** (Ours)	(↑ 0.6)	(↑ 0.5)	
ResNet-50	81.7	93.4	23.9
ResNet-50	81.7	92.1	23.9
+ **Random Erasing**	(↑ 0.0)	(↓ 1.3)	
ResNet-50	82.4	**93.6**	35.3
+ **SS**	(↑ 0.7)	(↑ 0.2)	
ResNet-50	**82.5**	93.5	23.9
+ **AIM** (Ours)	(↑ 0.8)	(↑ 0.1)	
EfficientNet-b0	82.0	92.7	4.3
EfficientNet-b0	82.2	92.4	4.3
+ **Random Erasing**	(↑ 0.2)	(↓ 0.3)	
EfficientNet-b0	82.5	93.0	15.8
+ **SS**	(↑ 0.5)	(↑ 0.3)	
EfficientNet-b0	**82.8**	**93.4**	4.3
+ **AIM** (Ours)	(↑ 0.8)	(↑ 0.7)	
EfficientNet-b1	82.8	93.3	6.8
EfficientNet-b1	83	93.3	6.8
+ **Random Erasing**	(↑ 0.2)	(↑ 0.0)	
EfficientNet-b1	82	92.8	18.3
+ **SS**	(↓ 0.8)	(↓ 0.5)	
EfficientNet-b1	**83.1**	**93.3**	6.8
+ **AIM** (Ours)	(↑ 0.3)	(↑ 0.0)	
EfficientNet-b2	83.5	93.7	8.0
EfficientNet-b2	82.8	93.6	8.0
+ **Random Erasing**	(↓ 0.7)	(↓ 0.1)	
EfficientNet-b2	80.9	93.5	19.5
+ **SS**	(↓ 2.6)	(↓ 0.2)	
EfficientNet-b2	**84.0**	**93.7**	8.0
+ **AIM** (Ours)	(↑ 0.5)	(↑ 0.0)	

Table 1. Classification accuracies (Acc) of methods for the fine-grain visual categorization of images in multiple data sets. The size of images to the task networks was fixed to 224 × 224.

The split-16 contains 16 models per class for training and 4 for testing. From Table 2, we observed that AIM significantly increased the classification accuracies of MeshNet on both data sets. However, the overall classification accuracies were still low because MeshNet is not robust in classifying unoriented meshes. For a stronger learner, such as MeshNet++, improvements were still observed. MeshNet++ achieved a 100% classification accuracy on split-16. Thus, it is challenging to verify how much AIM contributed to MeshNet++'s decision-making ability.

Method	MSB Acc(%)	SHREC-11 (split-16) Acc(%)
MeshNet	56.5	55.6
MeshNet with **AIM**	**67.3** (↑ 10.8)	**63.1** (↑ 7.5)
MeshNet++	94.5	100
MeshNet++ with **AIM**	**95.6** (↑ 1.1)	100 (↑ 0.0)

Table 2. Comparison of classification accuracies (Acc) of mesh classification methods on multiple data sets. AIM significantly improved classification accuracies for the MeshNet model.

Finally, DiffusionNet was evaluated with AIM to segment body parts in meshes of the human body. We observed that training and testing DiffusionNet with AIM increased the segmentation accuracies. We also trained and tested DiffusionNet by randomly jittering the position of vertices in the meshes. As seen in Table 3, the accuracies significantly dropped when vertices were randomly moved at test time without any learning. These experimental results support that learning by AIM is not random, but it is in synergy with the task network.

Method	Acc(%)
DiffusionNet	90.9
DiffusionNet + **jitter**	87.9 (↓ 3.0)
DiffusionNet + **AIM**	**91.3** (↑ 0.4)

Table 3. Comparison of accuracies (Acc) for segmenting body parts in human body meshes.

6. Qualitative Experimental Validation

The directional consistency loss enforces edge sensitivities from AIM's attention module to be similar in all independent directions in space. Thus, in the case of images, the edge sensitivities highlighted on the deformation grid

along each separate direction must appear similar. We qualitatively verify this in Fig. 5 below.

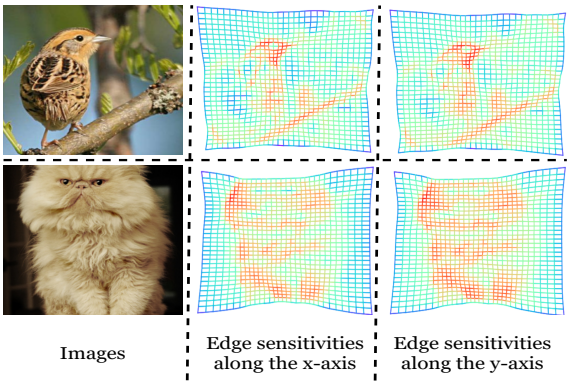

Figure 5. The directional consistency loss enforces edge sensitivities (from the attention module) in independent directions in space to be similar. Higher edge sensitivities are highlighted in red. [Best viewed in color]

7. Ablation Studies

In our ablation study, we first verified the significance of AIM's directional consistency loss ($\mathcal{L}_{dc}$). We trained all the models in Table 1 with and without the directional consistency loss and reported results in Table 4. Results suggest that using AIM with $\mathcal{L}_{dc}$ is beneficial to a downstream task.

Method	CUB-200		Oxford-IIT Pets	
	- $\mathcal{L}_{dc}$	+ $\mathcal{L}_{dc}$	- $\mathcal{L}_{dc}$	+ $\mathcal{L}_{dc}$
ResNet-18	78.7	**79.4**	91.3	**91.9**
ResNet-34	**80.4**	79.8	92.6	**93**
ResNet-50	82.3	**82.5**	93.5	93.5
EfficientNet-b0	81.8	**82.8**	93.2	**93.4**
EfficientNet-b1	83.1	83.1	**93.2**	93.0
EfficientNet-b2	83.3	**84**	93.2	**93.7**

Table 4. Comparison of image classification accuracies for methods utilizing AIM with and without directional consistency loss ($\mathcal{L}_{dc}$). When a classifier was trained with $\mathcal{L}_{dc}$, we denote it by +$\mathcal{L}_{dc}$, and when trained without $\mathcal{L}_{dc}$, it is denoted as -$\mathcal{L}_{dc}$.

We also verify whether or not AIM is an augmentation strategy for only the training phase. By setting Δ to one, no deformations will be performed by AIM. Thus, we set Δ to one and report the results in Table 5. Comparing the results of Table 5 against the results of Table 1 (where Δ is 0.7), we observed that the classification accuracies decreased.

Method	CUB-200 Acc(%)	Oxford-IIT Pets Acc(%)
ResNet-18	78.8	91.7
ResNet-34	80.1	92.8
ResNet-50	82.0	93.3
EfficientNet-b0	82.3	92.9
EfficientNet-b1	83.1	93.1
EfficientNet-b2	83.2	93.4

Table 5. Comparison of classification accuracies (Acc) when AIM is only used for training image classifiers and not for testing.

8. Discussion and Limitations

AIM can reasonably conserve the border pixels in images after warping. However, for a low Δ value, the spatial coverage of task-critical pixels can be reduced. If Δ is low for very tightly cropped images, task-critical image pixels could even be eliminated. AIM can also find itself limited in tasks requiring conservation of the input data geometry. Stacking more graph convolutions in the attention module will not necessarily increase the task performance [23], but it will increase computational overhead. In addition, AIM's computational cost will further increase if applied to more than three or four-dimensional data. Semi-supervised learning approaches can be a viable option in such scenarios [27]. In our experiments, DiffusionNet trained on a large number of vertices, and this significantly increased the size of the matrix A in equation 5. Moreover, AIM cannot solve for a very large number of vertices (X) on most modern GPUs. Thus, incorporating AIM with DiffusionNet increases training and testing times significantly. AIM recomputes the locations of the vertices during training and testing. However, some methods [14, 31, 40, 47] pre-processed vertex-based features before training, and they cannot use AIM.

9. Conclusion

We introduced an auto-augmenter for deep neural networks called AIM. AIM can augment two-dimensional image data as well as three-dimensional mesh data. AIM augments the data by producing constrained, non-rigid deformations at locations learned by an attention module. A key characteristic of AIM is to jointly optimize with neural networks for varied tasks during training and inference times. Thus, it can estimate the deformations suited to achieve better performance on tasks. We evaluated AIM for classifying images and meshes in multiple data sets. Furthermore, as AIM can conserve the connectivity in the input data, we also evaluated it for segmenting meshes. Our experimental results demonstrated that AIM effectively increased the robustness of neural networks on multiple tasks.

References

[1] Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Gridmask data augmentation. *arXiv preprint arXiv:2001.04086*, 2020. 2

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 1

[3] Yunlu Chen, Vincent Tao Hu, Efstratios Gavves, Thomas Mensink, Pascal Mettes, Pengwan Yang, and Cees GM Snoek. Pointmixup: Augmentation for point clouds. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 330–345. Springer, 2020. 2

[4] JD Crawford and T Vilis. Axes of eye rotation and listing's law during rotations of the head. *Journal of neurophysiology*, 65(3):407–423, 1991. 2

[5] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–123, 2019. 1, 2

[6] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 2

[7] Yao Ding, Yanzhao Zhou, Yi Zhu, Qixiang Ye, and Jianbin Jiao. Selective sparse sampling for fine-grained image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6599–6608, 2019. 2

[8] Yutong Feng, Yifan Feng, Haoxuan You, Xibin Zhao, and Yue Gao. Meshnet: Mesh neural network for 3d shape representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8279–8286, 2019. 1, 2, 5, 6

[9] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019. 6

[10] John M Findlay. Saccadic eye movement programming: Sensory and attentional factors. *Psychological research*, 73(2):127–135, 2009. 2

[11] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D Cubuk, Quoc V Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2918–2928, 2021. 1

[12] Chengyue Gong, Dilin Wang, Meng Li, Vikas Chandra, and Qiang Liu. Keepaugment: A simple information-preserving data augmentation approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1055–1064, 2021. 1, 2

[13] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017. 6

[14] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019. 1, 2, 5, 8

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1

[16] 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. 1, 2, 6

[17] Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019. 2

[18] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012. 2

[19] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 5

[20] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28:2017–2025, 2015. 1, 2, 3, 6

[21] Wei Jiang, Kai Zhang, Nan Wang, and Miao Yu. Meshcut data augmentation for deep learning in computer vision. *PLoS One*, 15(12):e0243613, 2020. 1, 2

[22] Chen Jin, Ryutaro Tanno, Thomy Mertzanidou, Eleftheria Panagiotaki, and Daniel C Alexander. Learning to downsample for segmentation of ultra-high resolution images. *arXiv preprint arXiv:2109.11071*, 2021. 2

[23] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 8

[24] Richard J Krauzlis and Steve G Lisberger. Temporal properties of visual motion signals for the initiation of smooth pursuit eye movements in monkeys. *Journal of Neurophysiology*, 72(1):150–162, 1994. 2

[25] Floris Laporte. Torch sparse solve. https://github.com/flaport/torch_sparse_solve.git, 2020. 6

[26] Ruihui Li, Xianzhi Li, Pheng-Ann Heng, and Chi-Wing Fu. Pointaugment: an auto-augmentation framework for point cloud classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6378–6387, 2020. 1, 2

[27] Xianzhi Li, Lequan Yu, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Unsupervised detection of distinctive regions on 3d shapes. *ACM Transactions on Graphics (TOG)*, 39(5):1–14, 2020. 8

[28] Z Lian, A Godil, B Bustos, M Daoudi, J Hermans, S Kawamura, Y Kurita, G Lavoua, and P Dp Suetens. Shape retrieval on non-rigid 3d watertight meshes. In *Eurographics workshop on 3d object retrieval (3DOR)*. Citeseer, 2011. 6

[29] Dmitrii Marin, Zijian He, Peter Vajda, Priyam Chatterjee, Sam Tsai, Fei Yang, and Yuri Boykov. Efficient segmentation: Learning downsampling near semantic boundaries. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2131–2141, 2019. 2

[30] Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G Kim, and Yaron Lipman. Convolutional neural networks on surfaces via seamless toric covers. *ACM Trans. Graph.*, 36(4):71–1, 2017. 6

[31] Francesco Milano, Antonio Loquercio, Antoni Rosinol, Davide Scaramuzza, and Luca Carlone. Primal-dual mesh convolutional neural networks. *arXiv preprint arXiv:2010.12455*, 2020. 5, 8

[32] Alexey Nekrasov, Jonas Schult, Or Litany, Bastian Leibe, and Francis Engelmann. Mix3d: Out-of-context data augmentation for 3d scenes. *arXiv preprint arXiv:2110.02210*, 2021. 1, 2

[33] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE, 2012. 6

[34] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 6

[35] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 2

[36] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*, 2020. 6

[37] Adria Recasens, Petr Kellnhofer, Simon Stent, Wojciech Matusik, and Antonio Torralba. Learning to zoom: a saliency-based sampling layer for neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 51–66, 2018. 1, 2, 3, 6

[38] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 1

[39] Michele Rucci and Martina Poletti. Control and functions of fixational eye movements. *Annual review of vision science*, 1:499–518, 2015. 2

[40] Lisa Schneider, Annika Niemann, Oliver Beuing, Bernhard Preim, and Sylvia Saalfeld. Medmeshcnn-enabling meshcnn for medical surface models. *Computer Methods and Programs in Biomedicine*, 210:106372, 2021. 8

[41] Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. Diffusion is all you need for learning on surfaces. *arXiv preprint arXiv:2012.00888*, 2020. 6

[42] Shivanand Venkanna Sheshappanavar, Vinit Veerendraveer Singh, and Chandra Kambhamettu. Patchaugment: Local neighborhood augmentation in point cloud classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2118–2127, 2021. 1, 2

[43] Kaleem Siddiqi, Juan Zhang, Diego Macrini, Ali Shokoufandeh, Sylvain Bouix, and Sven Dickinson. Retrieving articulated 3-d models using medial surfaces. *Machine vision and applications*, 19(4):261–275, 2008. 6

[44] Vinit Veerendraveer Singh and Chandra Kambhamettu. Feature map retargeting to classify biomedical journal figures. In *International Symposium on Visual Computing*, pages 728–741. Springer, 2020. 2

[45] Vinit Veerendraveer Singh, Shivanand Venkanna Sheshappanavar, and Chandra Kambhamettu. Mesh classification with dilated mesh convolutions. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3138–3142. IEEE, 2021. 5

[46] Vinit Veerendraveer Singh, Shivanand Venkanna Sheshappanavar, and Chandra Kambhamettu. Meshnet++: A network with a face. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 4883–4891, 2021. 5, 6

[47] Dmitriy Smirnov and Justin Solomon. Hodgenet: Learning spectral geometry on triangle meshes. *arXiv preprint arXiv:2104.12826*, 2021. 8

[48] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019. 6

[49] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 6

[50] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. 2

[51] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 3, 5

[52] Tianshu Xie, Xuan Cheng, Xiaomin Wang, Minghui Liu, Jiali Deng, Tao Zhou, and Ming Liu. Cut-thumbnail: A novel data augmentation for convolutional neural network. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 1627–1635, 2021. 2

[53] Xiaohan Xing, Yixuan Yuan, and Max Q-H Meng. Zoom in lesions for better diagnosis: Attention guided deformation network for wce image classification. *IEEE Transactions on Medical Imaging*, 39(12):4047–4059, 2020. 2

[54] Rong Yang, Robert Wang, Yunkai Deng, Xiaoxue Jia, and Heng Zhang. Rethinking the random cropping data augmentation method used in the training of cnn-based sar image ship detector. *Remote Sensing*, 13(1):34, 2021. 1

[55] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019. 1, 2

[56] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13001–13008, 2020. 2, 6