Activation Shaping for Domain Adaptation

Hao Chen, Xueyan Chen, Xiangdong Wang
Politecnico di Torino
Corso Duca degli Abruzzi, 24
{s305005, s296338, s289290}@studenti.polito.it

Abstract

This study addresses the challenge of domain adaptation in deep learning models, specifically focusing on the performance of ResNet-18 in supervised domain adaptation tasks. Domain adaptation is critical for AI models to maintain accuracy across different data distributions, particularly in computer vision where variations in lighting, background, and image styles can significantly impact model performance. We introduce an activation shaping technique that modifies the activation maps within the model's architecture to enhance its robustness to domain shifts. This technique is implemented using hooks that monitor and adjust the activation values during training. Experiments are conducted using the PACS dataset, with art paintings as the source domain and cartoons, sketches, and photographs as the target domains. Our results demonstrate that activation shaping improves the model's generalization capability across these diverse visual styles, with the prediction accuracy for the Cartoon target domain improving from 54.48% to 58.02%, highlighting its potential for enhancing AI model performance in real-world applications.

Index Terms: *Domain adaptation, Activation shaping, Classification, ResNet18*

1. Introduction

1.1. Research Background

The aim of this study is to explore how to increase the accuracy of the model in recognition of different visual styles of images that are outside the training dataset, i.e., to improve the performance of the model on these OOD (Out-Of-Distribution) data so that it has a better generalization ability under various distributions [1]. OOD data refer to distributions that the model has not seen during training and are beyond the scope of its training data. Specifically, training data and test data usually come from similar distributions, but in practice, models may encounter data from different distributions, which are called OOD data. The differences in these distributions may be due to a variety of factors, for example, in computer vision, changes in the style of the input images, lighting differences, changes in the background environment, etc.

To address this challenge, we introduced the activation shaping technique, which plans to define and modify the activation maps in certain layers of an existing neural network model: ResNet-18. Recent studies have shown that this approach is very effective for OOD data.

To achieve our goal, we plan to use 'Hooks' to monitor and tune the activation maps of the model and apply them to the underlying neural network model ResNet-18. With Hooks, we can view the output and gradient of the middle layers during the model training process and record them for detailed analysis and monitor the behavior of the model. It is also possible to insert custom logic to modify the activation values or gradients of the middle layer at certain moments. Therefore, the structure of our neural network will be easier to understand and easier to test with a variety of parameters.

1.2. Database

For our image database, we chose to use four types of images from the $PACS$ database [2] that are common in daily life and practical applications: art paintings, cartoons, sketches, and photographs. For the sake of comparative study, we designate art paintings as the source domain and cartoons, sketches and photographs as the target domain. We will test the neural network model by using these styles and compare the results to determine the most optimal parameters to use.

1.3. ResNet model (Original)

We choose ResNet-18 as the base model for our study. In deep neural networks, as the number of layers increases, the problems of gradient vanishing and gradient explosion become more serious, which leads to training difficulties, and ResNet effectively solves this problem by introducing the 'Residual Block'. The Residual Block allows inputs to bypass some layers directly through a skip connection, which makes it easier to back-propagate the gradient, thus improving the training of the deep network.

ResNet-18 is a deep convolutional neural network (CNN) model from the ResNet (residual network) family with 18 layers, including 16 convolutional layers and 2 fully connected layers. We chose ResNet-18 because it performs well on several image classification tasks and has fewer parameters and lower computational complexity than deeper models such as ResNet-34 and ResNet-50, making it ideal for experiments such as cross-group large-sample testing and comparative analysis. At the same time, it does not consume excessive computational resources. That allows us to conduct tests more rapidly while guaranteeing the quality of the task, thus saving a lot of time.

1.4. Hook

Hooks are commonly used tools in deep learning and neural network programming that allow users to access and optimize model modifications during model training and inferencing [3]. In this study, we use 'Forward Hooks' to extract the outputs of intermediate layers, modify the outputs, and monitor the activation values in the model. With Hooks, we inserted custom activation shaping (CASH) layers into the forward propagation process between the model's convolutional layers. This enables modifications to the model without changing the underlying architectural framework, thus reducing testing and iteration complexity.

1.5. Activation Shaping

In this study, we used activation shaping as an optimization technique. This approach involves stochastically adjusting the activation maps of the middle layer of CNN model during training, thereby enhancing the model's ability to generalize classification of images of different visual styles. Activation shaping helps the model to identify and process data that is not part of the training set more efficiently. We implemented this optimization strategy on the basic ResNet-18 by attaching custom 'hooks' to several layers. These 'hooks' are used to introduce activation shaping layers into the model, thus achieving our optimization goal.

2. Methodology

2.1. Activation Shaping

In this study, Activation Shaping is achieved by introducing a change function through a hook to modify the output tensor of the network layers in ResNet-18. In ResNet-18, there are three different types of layers where hooks can be installed: convolutional, activation and batch normalization layers, and the effect of installing the hooks is different for each of the layers. Installing a hook on the convolutional layer changes the output of local features; installing a hook on the activation layer modifies the activation function to change the model's response to activation values; installing a hook on the batch normalization layer modifies the normalization algorithm to adapt to different data distributions.

Taking the example of mounting the hook to the activation layer, assuming that the original activation function is $f(x)$, and x is the output of the previous network layer, we introduce a change function $g(x)$ through the hook so that the adjusted activation function:

$$h(x) = g(f(x)) \tag{1}$$

.

In this study, in order to enhance the generality of the model, we first binarize the activation function output tensor A, and then introduce a tensor M with the same scale as the activation output layer, which also consists of 0 and 1 and is randomly generated by a certain ratio, and take the elemental product of these two tensors, A $\times$ M, as the return value of the Hook in order to achieve stochastic pruning of the activation output (see Fig. 1). In besides preventing overfitting and improving generalization, we will also test different combinations of M in subsequent experiments to explore and optimize the activation patterns and enhance the flexibility and adaptability of the model.

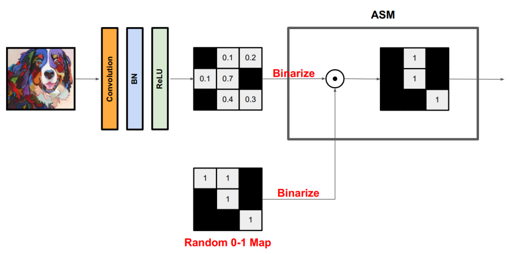

Figure 1. Activation shaping.

In our code, we first define a binary activation map by binarizing the output tensor, which we call (A bin). If an element in the output tensor is less than or equal to 0, the corresponding element in the A bin is set to

0; otherwise, it is set to 1. We then create a binary mask (M bin) by thresholding the mask tensor. If the element in the mask is less than or equal to 0, the corresponding element in the M bin is set to 0; otherwise, it is set to 1. Finally, the output (A bin * M bin) returns the product of the elements of the binary activation map (A bin) and the binary mask (M bin). This operation effectively masks some elements of the activation map according to the generated random mask. The elements will be non-zero only if both A bin and M bin are 1.0.

The significance of introducing random masks here is that in the task we expect to keep the important features of the image target and ignore the background features as much as possible, the attempts to use different random masks can help us to find the mask value to achieve efficient and accurate recognition by comparing the corresponding accuracy results.

Our hook function: (activation shaping hook) has three parameters (module, input and output). These parameters are automatically provided by our model when the hook is called during forward passes. The output corresponds to the output of the layer that defines the hook. Our hooks must be implemented in the CASH because the only changes in our model occur in this layer and we don't want to change the whole structure of the model when we modify it, based on the idea of control variables, modifying the model can only be done by modifying our hooks.

2.2. ResNet

ResNet-18 is a deep convolutional neural network widely used for image classification and feature extraction tasks. Its structure consists of 18 convolutional or fully connected layers and uses residual connections to solve the gradient vanishing and gradient explosion problems in deep networks.

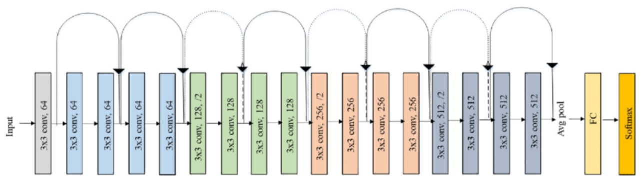

Figure 2. Resnet structure.

The basic structure of ResNet-18 consists of Input Layer, Residual Blocks, Global Average Pooling Layer, and Fully Connected Layer (see Fig. 2). The Input Layer uses a 7x7 convolution and a maximum pooling layer for initial processing of the input image. The Residual Blocks consist of 8 Basic Blocks, each containing 2 convolutional layers. The Global Average Pooling Layer is used to reduce the spatial dimension of the feature map to 1x1. The Fully Connected Layer outputs probabilistic results for the final classification task.

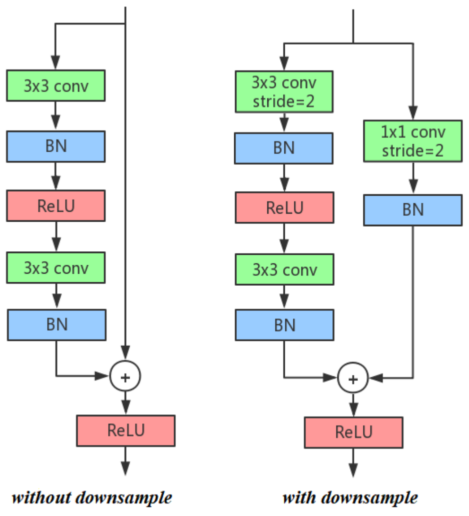

Figure 3. Residual block.

When the spatial dimensions (width and height) or the number of channels of the input and output feature maps of a residual block do not match, downsampling is required to make them compatible for the residual connection (see Fig. 3). Through downsampling operations, ResNet can increase the depth and complexity of the network while maintaining the effectiveness of residual connections, thereby improving feature extraction and classification performance.

In our study, we strategically introduce activation shaping hooks at key network layers specifically, those convolutional, activation and batch normalization layers that theoretically have a significant impact on model accuracy. These hooks apply custom transformations to the activation graph to enhance the model's adaptability to new data and different data distributions.

The activation shaping hooks modify the neural network in different ways in different network layers. Our study adapted two key variables: the target neural network layer and the frequency of hook deployment. The implementation at the code level involves filtering the target network layer using a combination of an *if* statement and the *isinstance* function: (`if isinstance(module, nn.target layer)`) and an *if* statement to adjust the frequency of hook installation (once per X groups of layers): (`if count % X == 0`) [4].

We performed domain adaptation tests and comparisons using three different styles of datasets with the aim of identifying the ideal variable settings.

2.3. Compute loss

To measure the performance of the model and provide data for the next iteration, we chose to use Cross Entropy Loss as the loss calculation function for our task.

This loss function is typically used for multi-class image classification tasks. The core idea of this loss function is to compare the probability distribution of the model output with the probability distribution of the true labels and measure the difference between them. If the model's prediction agrees with the true label, the cross-entropy loss function tends to zero; otherwise, the loss function increases, indicating a large discrepancy between the model's prediction and the true situation. We optimize the model parameters by minimizing the cross-entropy loss function: during the training process, the model parameters are updated according to the gradient of the loss function, so that the model can be gradually adjusted to make the prediction results closer to the real situation, thus improving the performance of the model on the training set. The formula for Cross-Entropy Loss is as follows:

$$L = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{2}$$

. where:

- C represents the number of classes.
- y_i is the i-th component of the true label, usually represented using one-hot encoding. If the sample belongs to class j, then $y_j = 1$, and the rest $y_i = 0$ (where $i \neq j$).
- $\hat{y}_i$ is the predicted probability of the i-th class.

As mentioned above, in our iterations we aim to train the model by minimizing the cross-entropy loss so as to improve its performance in the target domains of Photo, Cartoon and Sketch. The model will try to adjust the feature representation between the source domain (art painting) and the target domain to achieve better generalization.

3. Experimental results and discussion

In this section, we describe in detail how the principles of hook functions and CASH can be used to optimize the model to better suit new domains. We first conducted a series of experiments to test the effects of the activation shaping module and random activation maps under different parameter settings. These results are then compared longitudinally and cross-sectionally with baseline experiments with the aim of identifying the most optimal combination of optimization parameters.

The dataset for this study comes from the *data/PACS* directory, fixing art paintings as the source domain. Considering the machine performance limitation and the time consideration for multiple experimental comparisons, we set the batch size to 128, the number of workers for data loading to 5, the number of gradient accumulation steps to 1, and the number of model iterations for each experiment to 29 (Epoch: 0-29).

3.1. Baseline

In our study, in order to provide a reference for subsequent model optimization, we first conducted three sets of baseline experiments based on the original ResNet18 model. The target domains of these three sets of experiments were dynamically specified via command line parameters, and the format of the experiment name was `--experiment.name=baseline/${target.domain}`, where `${target domain}` was the variable of the target domain (cartoon, sketch, and photo, respectively). The results of the BASELINE experiment will be shown in Tab. 1. Figure. 4 shows the test accuracy in the three target domains. The accuracy tends to stabilize around the 15th epoch.

Domain	Accuracy (Epoch=29)
Art Painting → Cartoon	54.48%
Art Painting → Sketch	40.89%
Art Painting → Photo	95.93%

Table 1. Baseline Results

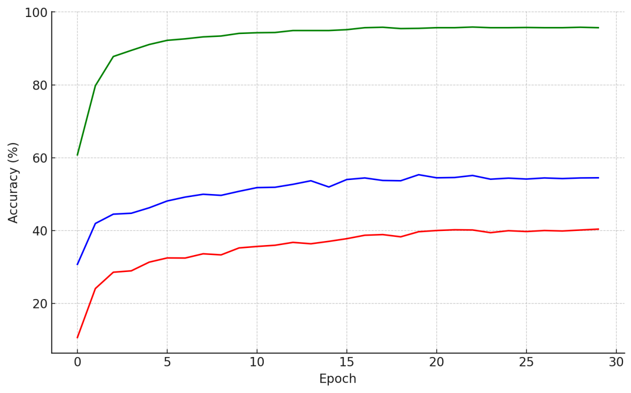

Figure 4. Accuracy of baseline. Solid green line: Photo; Solid red line: Cartoon; Solid blue line: Sketch.

3.2. Random Activation Maps

In order to randomize the activation maps, we need to turn off some of the outputs of each layer, which can be achieved by varying the mask tensor M in the CASH. M should be the same size as the binarized activation function output tensor A. The effect of different retention ratios on model performance is tested and observed by varying the mask ratio of M to control the proportion of 0 and 1 in the stochastic tensor M, i.e. the proportion of elements retained in the activation output.

The significance of introducing random masks here is that in the task we expect to keep the important features of the image target and ignore the background features as much as possible, the attempts to use different random masks can help us to find the mask value to achieve efficient and accurate recognition by comparing the corresponding accuracy results.

Such operations can help us find the optimal mask ratio and thus optimize the performance of the model. In our case, we use a threshold of 0.5 for the mask (M), which means that the value of each mask element is determined by comparing a randomly sampled value from a uniform distribution to 0.5. If the random value is less than 0.5, the corresponding mask element is set to 0; otherwise, it is set to 1. We conducted several tests with different experimental groups and found that thresholds that were either too high (e.g., 0.75) or too low (e.g., 0.25) were inappropriate, as the former resulted in data loss and the latter failed to show significant optimization (see Tab. 2). There is no universally optimal threshold for domain adaptation across different visual styles, various batch sizes, etc. It needs to be adapted on a case-by-case basis, but we have typically found that the optimal value is about 0.5.

Mask ratio	Layer	Accuracy	Loss
M=0.0	conv1	55.80%	0.0106281
M=0.25	conv1	57.00%	0.0101579
M=0.5	conv1	58.96%	0.0096636
M=0.75	conv1	48.63%	0.0122445
M=1.0	conv1	17.28%	0.0155859

Table 2. Results different mask ratio. Source domain: Art Painting; Target domain: Cartoon; Stop epoch: 29.

Based on the M mask of 0.5, we added a Hook to the convolutional, activation and batch normalisation layers of the ResNet18 neural network and made several attempts with different locations and different target domains. As shown in Tab. 3, taking the target domain as cartoon for example, we find that the accuracy of the model is higher when the hook is added to the convolutional layer in the Input Layer compared to when the hook is added to the convolutional layer in the Residual Blocks. Moreover, the further following in the structure the Hook is added, the lower the model's accuracy tends to be. Similarly, the same trend was found for adding Hooks in the activation and batch normalisation layer.

We analyse the trend due to the fact that in the ResNet18 structure, the input layer performs global preliminary rough feature extraction on the input image, so such unimportant features like background features still have a certain amount of weight in the whole activation map. As the convolution deepens, the further following in the model structure the activation map contains a larger proportion of important features for target recognition, so the higher the probability of pruning important features by adding a Hook to the layers at the backward end of the model, the higher the possibility of resulting in a decrease in the model's recognition accuracy.

Also we found that when adding Hook to different layers in Input Layer, adding random activation maps to convolutional layer has slightly higher model accuracy than the other two layers. We analyse this because the role of the convolutional layer is to perform convolutional operations to extract local features, while the Batch Normalisation Layer normalises the feature maps output from the convolutional layer to stabilise the training process, and the ReLU Layer performs non-linear activation on the normalised feature maps to enhance the expressive power of the model. It can be seen that the convolutional layer is responsible for the extraction of the underlying features, and the appropriate degree of randomness can increase the diversity of features and the generalisation ability of the model without completely destroying the overall structure of the features. Adding random activation maps to the Batch Normalisation and ReLU layers can have a negative impact on the normalisation and non-linear activation processes, leading to feature instability and the accuracy of the model lower than the one obtained from working on convolutional layers.

Layer	Accuracy	Loss
conv1	58.96%	0.0096636
layer2.0 conv2	56.23%	0.0103633
layer3.0 downsample.0	54.52%	0.0107927
layer3.1 conv1	50.09%	0.0122641
bn1	57.3%	0.0099767
layer1.0 bn2	56.7%	0.0102045
layer3.0 downsample.1	54.61%	0.0106752
layer4.0 bn2	50.34%	0.0114165
relu1	57.3%	0.0099669
layer1.1 relu	55.89%	0.0107105
layer4.0 relu	48.21%	0.0123730
layer4.1 relu	50.09%	0.0105503

Table 3. Results for activation shaping in one layer. Source domain: Art Painting; Target domain: Cartoon; Stop epoch: 29.

Layers	Accuracy	Loss
conv1, layer1.0.conv1	57.72%	0.0106367
layer4.0.conv1, layer4.0.downsample.0	57.25%	0.0100599
layer2.0.conv1, layer3.0.downsample.0	56.06%	0.0103579
layer4.0.conv2, layer4.1.conv1	44.88%	0.0132535
layer3.1.conv1, layer4.1.conv1	46.2%	0.0133692
layer3.0.conv2, layer3.1.conv1	46.72%	0.0129008

Table 4. Results for activation shaping in two convolution layers. Source domain: Art Painting; Target domain: Cartoon; Stop epoch: 29.

In addition to adding a Hook to the structure, we also tried the solution of adding multiple Hooks at different locations at the same time (as shown in Tab. 4, with two

Hooks with random activation graphs added at the same time). Compared with Tab. 3, we can see that the model accuracy of the solution of adding a single Hook to the structure is better. Especially, with adding more hooks, the accuracy turns lower (see Tab. 5). We analyze that this may be attributed to the cumulative effect of randomness brought about by adding multiple random activation graphs, which makes the overall feature distribution more confusing and difficult to maintain consistency, and the cumulative randomness may destroy the correlation between features and the stability of the overall feature extraction, resulting in the feature extraction process of the whole network being disturbed, and the model being difficult to learn and extract useful features. At the same time, adding random activation maps at multiple locations at the same time is equivalent to introducing multiple interference points in the information transfer process. This causes the information to be disturbed at multiple levels and can lead to more noise in the feature learning process. In addition, the existence of multiple hooks at the same time significantly increases the training complexity, which may lead to problems in the gradient propagation process, making it difficult for the model to converge efficiently, and ultimately leading to a decrease in the accuracy of the model.

Layers	Accuracy (Epoch=29)	Loss
1 Conv layer	58.96%	0.0096636
2 Conv layers	57.72%	0.0106367
3 Conv layers	49.83%	0.0116901
4 Conv layers	28.97%	0.0161867

Table 5. Compare the results of inserting activation maps in multiple conv layers. Source domain: Art Painting; Target domain: Cartoon.

3.3. Adapting Activation Maps Across Domains

We have also experimented with Adapting Activation Maps Across Domains. This involves adapting the activation maps of the network between the source and target domains by performing forward passes to different target domains to obtain the activation maps Mt, and then multiplying them by the output tensor of the activation maps of the source domain via Hook during the forward passes of the source domain to achieve purposeful pruning to optimise the performance of the model on the target domain. We test the learned model one by one with art painting as the source domain and Photo, Cartoon, Sketch as the target domain respectively and get the results as shown in Table 6. Figure. 5 shows the test accuracy in the three target domains. The accuracy tends to be stable around the 22th epoch.

It can be seen that the only case where a slight improvement is observed is when the activation module is added according to the cartoon target domain. The accuracy does not change much when the target domain is Sketch, while

Target domain	Accuracy (Epoch=29)	Loss
Cartoon	58.02%	0.0098482
Sketch	40.55%	0.0125038
Photo	90.16%	0.0023159

Table 6. Adapting activation maps across domains. Source domain: Art Painting. Activation shaping Layer: Conv1.

the accuracy decreases when the target domain is Photo. We speculate that these differences may be brought about by the randomness generated by differences in sample distribution in the target domain and the process of binarisation. Although the activation maps in the target domain can provide some new feature information, the direct introduction of these activation maps into the source domain model may lead to a degradation of the model performance due to feature mismatch, overfitting, and non-adaptability. In addition the introduction of target domain activation maps alone, without other auxiliary optimisation strategies, may not be sufficient to achieve good domain adaptation.

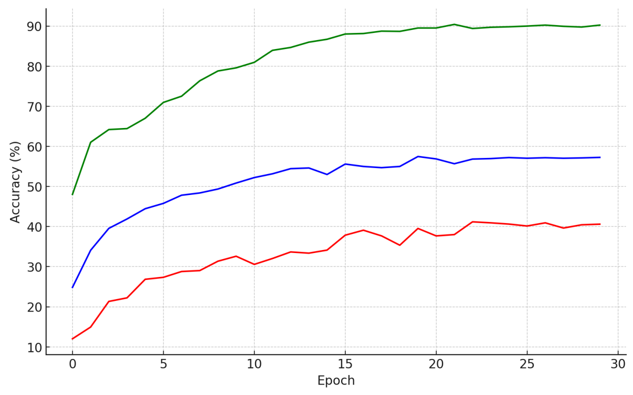

Figure 5. Accuracy for adapting activation maps across domains. Solid green line: Photo; Solid red line: Cartoon; Solid blue line: Sketch. Activation shaping Layer: Conv1.

3.4. Conclusion

In this study, we explore how to enhance the domain adaptation of an image recognition model between different styles of target domains by adding a Hook with a random activation map without changing the original model. We compared the domain adaptation performance of the original ResNet18 neural network structure with the prediction performance of the neural network after adding the Hook with random activation maps and the model performance after adding the Hook with stylised activation maps for the target domain, respectively.

Since ResNet18 is already a very mature and stable model, our different attempts did not result in very significant performance improvements. However, some interesting findings were obtained.

After adding random activation map Hooks in the neural network, we tried adding a single Hook at different positions and adding multiple Hooks simultaneously. Due to the effect of convolution depth on how well the important features of the target are explored, the stackability of the randomness of multiple activation maps, and the increase in model complexity, we found that adding Hooks to the convolutional layer of the input layer has a more significant performance improvement. And excessive activation shaping can negatively affect the model.

In the attempt of Adapting Activation Maps Across Domains, we do purposeful pruning by multiplying the activation maps trained on the target domain with the output tensor of the source domain activation maps via Hook to optimise the performance of the model on the target domain. Although the activation maps in the target domain can provide some new feature information, the direct introduction of these activation maps into the source domain model may lead to a degradation of the model performance due to feature mismatch, overfitting, and non-adaptability. To improve the performance, it is usually necessary to combine other domain adaptation techniques, such as adversarial training, feature alignment, and self-supervised learning, to better achieve feature sharing and adaptation between the source and target domains. At the same time, careful experimental setup and hyper-parameter tuning are also required to ensure the stability and effectiveness of the model under the new feature representation.

CASH are implemented by cleverly using hooks without changing the structure of the original residual neural network. The advantages of this approach are readability, simple structure, and easy editing. However, there are still many aspects of our research that can be improved. For example, more complex and deeper network iterations could be tested and batch sizes could be increased when machine performance allows. In addition, there is a lot of room to extend this research, such as installing Hook on different types of neural layers of the original model and optimising them, moving from binary to non-binary activation maps and re-evaluating previous experiments, exploring the possibility of applying Hook to other neural networks for their performance enhancement and so on.

References

[1] A. Djurisic, N. Bozanic, A. Ashok, and R. Liu, "Extremely Simple Activation Shaping for Out-of-Distribution Detection," 2022. Version Number: 2. 1

[2] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales, "Deeper, Broader and Artier Domain Generalization," in *2017 IEEE International Conference on Computer Vision (ICCV)*, (Venice), pp. 5543–5551, IEEE, Oct. 2017. 1

[3] N. Bhaskhar, "Intermediate Activations — the forward hook. https://web.stanford.edu/~nanbhas/blog/forward-hooks-pytorch/," Aug. 2020. 2

[4] H. Chen, X. Chen, and X. Wang, "Github repository for DAAI project 6. https://github.com/EnzoDaEst/DAAI-Project6.git," 2024. 3