Activation Shaping for Domain Adaptation

Jingming Li s322374 Hassan Soueidan ,Mohammad Al Najjar s322207 ,s317765

Abstract

With the rise of artificial intelligence applications, the universality of AI models has become key to large-scale deployment. In practical applications, there are numerous instances where the conditions differ significantly from standard training sets, influenced by factors such as lighting, image style, background, etc. Addressing this issue, this report investigates the domain adaptation problem in deep learning models, with a particular focus on domain shift. The main emphasis is on Unsupervised Domain Adaptation (UDA) [1], where a model trained on a labeled source domain is adapted to perform well on an unlabeled target domain. To enhance the cross-domain generalization capabilities of AI models under various conditions, this paper introduces the method of "Activation Shaping." This technique involves modifying the activation maps within the model's architecture to improve performance on out-of-distribution data, thereby training models that are robust to domain shifts.

1. Introduction

1.1. Research Background and Database

This study aims to explore how to improve the accuracy of model recognition of images with different visual styles that are outside the training dataset. To achieve this, we introduced the technique of Activation Shaping, planning to define and modify the activation maps in certain layers of an existing neural network model (ResNet-18 used in this study). Recent research has demonstrated that this approach is highly effective for detecting out-of-distribution (OOD) data [2].

To reach our goal, we plan to use "Hooks" to customize our modifications to the activation maps and apply them to the foundational neural network model, ResNet-18. The advantage of this method is that it allows feature extraction without any modification to the existing model, making our neural network structure more comprehensible and easier to test with various parameters.

For our image library, we have chosen to use four types of images from the PCAS database [3] that are common in daily life and practical applications: art paintings, cartoons, sketches, and photographs. To facilitate comparative research, we designated art paintings as the source domain and cartoons, sketches, and photographs as the target domains. We will test the neural network model with these styles and compare the results to identify the most ideal parameters for use.

1.2. Original Resnet model

We chose ResNet-18 as the basic model used in our research.ResNet-18 is a deep convolutional neural network model that belongs to the ResNet (Residual Network) series, comprising a total of 18 layers, including 16 convolutional layers and 2 fully connected layers. Our choice of using ResNet-18 is driven by its capability to adequately meet the demands of image recognition and classification tasks. Compared to deeper models such as ResNet-34 and ResNet-50, ResNet-18 boasts fewer parameters, higher computational efficiency, and a more lightweight model architecture. These attributes make ResNet-18 highly suitable for experiments involving extensive testing across various groups and comparative analysis, allowing us to conduct tests more rapidly without compromising task performance, thus saving significant time.

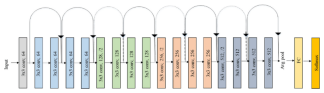

Figure 1. Original ResNet-18 Architecture

1.3. Activation Shaping

In this study, we employed Activation Shaping as our optimization technique. This approach involves applying random adjustments to the activation maps of intermediate layers in the convolutional neural network during training, thereby enhancing the model's ability to classify images across different visual styles. Activation Shaping helps the model to more effectively recognize and process data that is not part of the training set. We implemented this optimization strategy on the foundational convolutional neural network, ResNet-18, by attaching custom "Hooks" following several basic convolutional layers. These "Hooks" introduce Activation Shaping layers, thus achieving our optimization objectives.

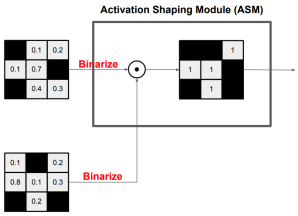

Figure 2. Activation Shaping

1.4. Hook

Hooks are a commonly used feature in deep learning and neural network programming, allowing users to access and optimize modifications to the model during its training and inference processes. In this study, we utilized 'Hooks' to implement custom layer connections and iterations within the model. Through Hooks, we accomplished the insertion of our custom Activation Shaping layers during the forward propagation process in between the convolutional layers of the model. This achieved modifications to the model without altering the underlying architectural framework, facilitating easier testing and iteration.

2. Methodology

2.1. Activation Shaping Layer

Described mathematically, Activation Shaping can be understood as modifying the output tensors of network layers in ResNet-18 by introducing a change function via a Hook. In ResNet-18, there are three different types of layers where hooks can be mounted: convolutional layers, activation layers, and batch normalization layers. Mounting on convolutional layers changes the output of local features; mounting on activation layers modifies the activation function, altering the model's response to activation values; mounting on batch normalization layers changes the normalization algorithm, adjusting adaptability to different data distributions.

Taking the example of mounting the hook to an activation layer, if the original activation function is $f(x)$, with x being the output from the previous network layer, we introduce a change function $g(x)$ through the hook, making the adjusted activation function $h(x) = g(f(x))$.

In this study, to enhance the model's universality, we first binarize the activation function output tensor A. We then introduce another tensor M of the same size as the activation output layer. Tensor M is composed of 0s and 1s, generated randomly in a certain proportion. The element-wise multiplication of these two tensors, $A \times M$, is used as the return value of the Hook. In subsequent experiments, we will repeatedly test different compositions of tensor M to find an ideal value.

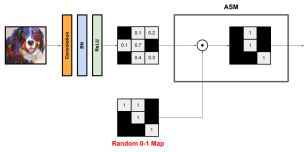

Figure 3. Activation Shaping Hook after Activation Layer

2.2. ResNet

In our study, we strategically introduced *Activation Shaping hooks* in 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 maps to enhance the model's adaptability to new and varied data distributions.

Across different network layers, the *Activation Shaping hook* modifies aspects of the neural network in varying ways. Two key variables were adjusted in our research: the target neural network layer and the frequency of hook mounting. The code implementation involved using a combination of `if` statements and the `isinstance` function to filter the target network layers : (`if isinstance(module, nn.target_layer)`) and `if` statements to adjust the frequency of hook mounting (Once per X set layers): (`if alternate % X == 0`). [4]

We conducted domain adaptation tests and comparisons using three datasets of different styles, aiming to determine the ideal variable settings.

2.3. Compute loss

To measure the performance of our model and provide data for the next iteration, we have chosen to use the Cross-Entropy Loss as the loss calculation function for our task. This loss function is commonly used in multi-class image classification tasks. The core idea of this loss function is to quantify the difference between the model's predicted probability distribution and the actual labels to assess model performance. The calculation formula for Cross-Entropy Loss is as follows:

$$\text{Cross-Entropy Loss} = -\sum_{i=1}^{C} y_i \log(p_i)$$

Where:

y_i represents the probability of class i in the true labels, which can be 0 or 1, indicating whether it belongs to that class.

p_i represents the predicted probability of class i in the model's probability distribution.

In our iterations, we aim to train the model by minimizing the Cross-Entropy Loss, which, in turn, enhances its performance in the target domains such as Photo, Cartoon, and Sketch. The model will attempt to adapt the feature representations between the source domain (Art Painting) and the target domains to achieve better generalization.

3. Comparison and Optimization

In this section, we will describe in detail how to optimize the model for better adaptation to new domains using the principles of hook functions and custom Activation Shaping layers. A series of experiments were conducted to test the effects of the Activation Shaping Module and Random Activation Map under different parameter settings. These results were then compared both vertically and horizontally with the baseline experiments, aiming to identify the most optimal combination of optimization parameters.

3.1. Baseline

In our experiment, to provide a reference baseline for subsequent model optimizations, we initially conducted three sets of baseline experiments based on the original ResNet18 model. The target domains for these experiments are dynamically specified via command-line arguments, with the experiment name formatted as `--experiment_name=baseline/${target_domain}`, where `${target_domain}` is the variable for the target domain (cartoon, sketch, photo, respectively). The dataset is sourced from the `data/PACS` directory, with `art painting` fixed as the source domain. Due to machine performance limitations and time considerations for multiple experimental comparisons, we set the `batch size` to 64, the `number of worker` for data loading to 5, and the `gradient accumulation steps` to 1. The model iteration count for each experiment is set to 29 times (Epoch: 0 to 29). The results of the baseline experiments will be presented below.

Domain	Accuracy(Epoch=29)
Art Painting → Cartoon	57.30%
Art Painting → Sketch	47.87%
Art Painting → Photo	96.53%

Table 1. : Baseline Results

3.2. Activation Shaping Model

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 the forward pass. The output corresponds to the output of the layer where our hook is defined.

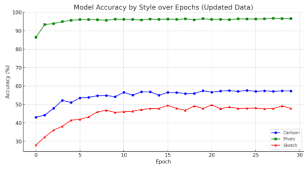

Figure 4. Baseline Accuracy

In our code, we defined a binary activation map that we called (A bin) by thresholding the output tensor. If an element in the output tensor is less than or equal to 0, the corresponding element in A bin is set to 0.0; otherwise, it is set to 1.0, and then we create a binary mask (M bin) by thresholding the mask tensor. If an element in the mask is less than or equal to 0, the corresponding element in the M bin is set to 0.0; otherwise, it is set to 1.0. At the end, the output (A bin * M bin) returns the element-wise product of the binary activation map (A bin) and the binary mask (M bin). This operation effectively masks out certain elements of the activation map based on the generated random mask. Elements will be non-zero only if both A bin and M bin are 1.0.

Our hook must be implemented in a custom activation layer because the only change in our model occurs in this layer and we don't want to change the entire structure of our model whenever we want to modify our model, so modifying the model is done only by modifying our hook.

According to our results and after testing our model by placing the Custom activation layer (CAL) after CONV, or BN, or activation layers the best place to insert our (CAL) is after the conv. layer in order to optimize our model since this case represents the worst scenario of our model because first we didn't normalize the data distribution here using BN LAYER, 2nd we didn't define RELU(activation function also using activation layers), another proof is the accuracy rates we've got after testing our model as an example when we took the cartoon as the target domain(3 layers once and random rate 0.2): And a similar situation for photo and sketch groups.

Layer	Accuracy(Epoch=29)	Loss(Epoch=29)
After CONV	82.28 %	0.016073739589046
After BN	72.12 %	0.024014827127757
After AL	33.15 %	0.053950566798448
Baseline	57.30%	0.021241968016874

Table 2. : Different Test Results

After testing our model on different scenarios placing the CAL after 3 convolutions leads to the best accuracy. as an example when we took the cartoon as the target domain(hook after CONV layer and random rate 0.2): And a

Layer	Accuracy(Epoch=29)	Loss(Epoch=29)
4 Layers once	71.44%	0.025804756012803
3 Layers once	82.28%	0.016073739589046
2 Layers once	40.97%	0.048862663563340
Baseline	57.30%	0.021241968016874

Table 3. : Different Test Results

similar situation for photo and sketch groups.

3.3. Random Activation Maps

To randomize the activation maps, we need to turn off some outputs of each layer, which can be achieved by creating a random mask of ones and zeros for each mask tensor (M) in the Custom Activation Layer (CAL). In our case, we used a threshold of 0.2 for the mask (M), meaning that the value of each mask element is determined by comparing randomly sampled values from a uniform distribution to 0.2. If the random value is less than 0.2, the corresponding mask element is set to 0.0; otherwise, it is set to 1.0. We conducted multiple tests across different experimental groups and found that neither too high (e.g., 0.6) nor too low (e.g., 0.05) thresholds are suitable, as the former leads to data loss and the latter fails to show significant optimization. For domain adaptation across different visual styles and various batch sizes etc, there is no universally optimal threshold. It needs to be adjusted according to specific circumstances, but we generally found that the optimal choice often lies within the range of 0.1 to 0.2.

4. Results

Our experiments have shown that not all image style transfers benefit significantly from the introduction of an Activation Shaping layer. Particularly in the transition from Art Painting to Photo, the use of the Activation Shaping layer often leads to a decrease in recognition accuracy or adversely affects the learning curve (e.g., causing instability) without improving accuracy. We believe this is due to the introduction of unnecessary random masking to the output tensors of convolution or batch normalization layers, especially in visually distinct real photos, which may hinder recognition. However, for other groups with a larger difference in visual style, the application of the Activation Shaping layer positively affects model training. Therefore, if the original model already has high recognition accuracy, the introduction of an Activation Shaping layer might not be necessary and could lead to longer training times and increased instability. However, for domain adaptation tasks in image recognition with significant visual style differences, employing this technique can significantly enhance accuracy and is a worthwhile option.

5. Conclusion

In this study, we explored how to enhance the domain adaptability of the model without changing the original residual neural network structure, by cleverly using hooks to implement custom activation shaping layers. The advantage of this method lies in its readability, simplicity of structure, and ease of editing. However, there are many aspects of our study that can be improved. For instance, more complex and deeper iterations of the network could be tested if machine performance allows, and batch sizes could be increased. Our current test results generally show that the performance reaches its limit early in the iteration process and then fluctuates, which might be due to the small batch size. Additionally, there is considerable room for expansion in this research, such as considering mounting hooks on different types of neural layers in the original model and optimizing comparisons.

References

[1] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor S. Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016. 1

[2] A. Djurisic et al. Extremely simple activation shaping for out-of-distribution detection. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2023. 1

[3] Da Li et al. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 1

[4] Jingming. Github project repository for da project. https://github.com/jingming12/DAproject, 2024. 2