Unsupervised domain adaptation with simple Activation Shaping

Fabio Barbieri
Politecnico di Torino
s317783@studenti.polito.it

Vincenzo Dalia
Politecnico di Torino
s309864@studenti.polito.it

Mario Todaro
Politecnico di Torino
s308812@studenti.polito.it

Abstract

Unsupervised domain adaptation (UDA) aims to improve the performance of models trained on labeled source domains when applied to unseen, unlabeled target domains. This is a challenging task due to discrepancies in source and target distributions. One of the main approaches to UDA is aligning source and target feature distributions during training. State-of-the-art methods achieve this by leveraging either adversarial learning or self-supervision. In this paper, we follow a simpler, zero-learning approach to explore two main hypotheses. Content-specific paths exist within a network; and discarding style-specific paths is possible by performing simple Activation Shaping (ASH). This technique, introduced in [5] for Out-of-Distribution detection (OOD), involves modifying activation maps from one or more layers within an architecture using either user-defined or learned rules. We report extensive experiments on the effectiveness of two specific user-defined rules for ASH in UDA. Our results demonstrate that simple ASH can improve performance when dealing with challenging domain shifts; suggesting that selectively operating on activation maps can effectively help in learning content-specific information and encouraging further investigation of this technique in domain adaptation settings. Further exploration can focus on learning shaping rules instead of defining them and selecting other static or adaptive ones. We also suggest testing this approach by integrating it into more sophisticated architectures already in place for domain adaptation. In addition, to gather stronger evidence on the effectiveness of ASH for UDA, we suggest testing this approach with different baseline architectures. Exploration of this technique is also encouraged in semi-supervised settings, to test the effectiveness of the proposed strategies when employing small labeled sets of target samples. Implementation can be found at: https://github.com/VincenzoDalia/AML_Project.

1. Introduction

A significant limitation of neural networks is their inability to perform well with any distribution met at test time. This is often referred to as the ”Domain Shift” problem: training and test data often show significant differences, resulting in models with poor generalization capabilities. How can we train a model that is capable of generalizing to one or more domains that shift from the training one? To address the challenges posed by domain shifts, researchers have developed techniques like Unsupervised Domain Adaptation (UDA). UDA tackles the challenge of transferring knowledge from a labeled source domain to an unlabeled target domain, effectively addressing the domain shift problem. Moreover, by leveraging only labeled source domains, UDA is crucial in scenarios where obtaining labeled data or fine-tuning for the target domain is impractical or costly.

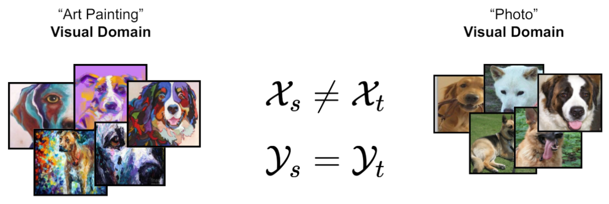
"Art Painting"
Visual Domain
$\mathcal{X}_s \neq \mathcal{X}_t$
$\mathcal{Y}_s = \mathcal{Y}_t$
"Photo"
Visual Domain

Figure 1. UDA setting on the PACS dataset

1.1. Do content-specific paths exist?

Our research explores two main questions: Do content-specific paths exist within neural networks, and can we easily identify them? We hypothesize that simple ASH can effectively isolate content-specific features within a network with minimal computational and training overhead. The core of our method (ASH) has proven successful for OOD in [5], which demonstrated the efficacy of simple activation modifications in identifying outliers and inspired our work on the domain adaptation setting. To test these hypotheses, we designed a simple UDA setting using the PACS dataset [10] and the ResNet18 architecture [8], which we modified to perform activation shaping using two main strategies:

- **Random Activation Masks (RA):** Using random activation masks to disrupt style-specific information.

- **Domain Adaptation Shaping (DA):** Aligning source and target activations by performing binarization and element-wise multiplication.

By testing these two shaping rules, we assess the effectiveness of ASH in aligning two domains with minimal overhead, potentially justifying further exploration of this technique for UDA.

2. Related Work

Several SOTA methods have been proposed to address UDA. Currently, relevant methods employ deep models and primarily fall into two main categories [12, 20]: feature-level alignment and self-supervision. Feature-level alignment methods generally include adversarial learning strategies. These methods leverage adversarial training to minimize the discrepancy between source and target domains by introducing a domain classifier. This approach employs a gradient reversal layer to ensure that the features learned by the model are indistinguishable between the source and target domains, thereby effectively reducing the domain gap [6, 9, 17]. Adversarial methods can be challenging to train and may suffer from instability issues, making them less practical in some scenarios [16]. Another widely explored method is to perform feature-space alignment by minimizing second-order statistics (covariances) [1, 18] or by working in kernel spaces to measure and minimize the difference between the mean of target and source embeddings [13–15]. The other relevant method involves learning implicit representations using self-supervised tasks to capture the intrinsic properties of data. Examples of such tasks include solving jigsaw puzzles [3], predicting image rotations [2,19], or employing reconstruction tasks [7]. With the help of such tasks, models are encouraged to learn meaningful and invariant representations without requiring labeled data from the target domain. However, designing effective self-supervised tasks can be challenging, as they must strike a balance between complexity and informativeness. Other methods try to align feature spaces with parameters-free approaches. Relevant examples can be found in AdaBN [11], which aligns the learned source and target representations by using different mean/variance terms when performing batch-normalization (BN) at test time. AutoDIA [4] goes further in this direction by learning which BN layers to adapt instead of defining them a priori. Our approach falls into the alignment category involving parameters-free techniques. The employed zero-learning strategies avoid the burdens of design, training, and computational overhead typically associated with other techniques.

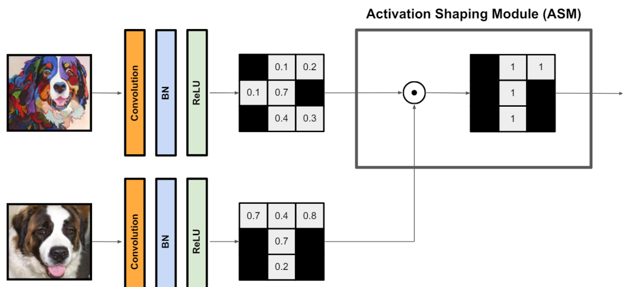

Figure 2. Activation shaping module

3. Method

This section dives deep into the proposed ASH techniques employed for UDA. We will analyze the implementation of a Custom Activation Shaping module hooked up in a ResNet18 architecture [8]. Random Activation Maps (RA) and Domain Adaptation (DA) are the two static strategies we employed to enhance the baseline robustness to domain shift.

3.1. Dataset and Baseline

The dataset used in this work is PACS [10], which is commonly employed for domain adaptation tasks. It contains 7 categories of images: Dog, Elephant, Giraffe, House, Person, Horse and Bus. Each of these categories has an equal number of images, making the dataset balanced. PACS includes images from four different domains: artistic painting, cartoons, sketches and photos. Each of the 4 domains contains different characteristics, such as visual variability, different contours and different styles, which make the dataset suitable UDA settings. The baseline network is a ResNet18, a Convolutional Neural Network (CNN) used to extract features from images. Beeing 'the network' is the baseline or a modified ResNet18 that includes our module to shape a given set of layers, each experiment includes 2 steps:

1. Train the network on the source domain (Art painting) using a cross-entropy loss to evaluate the dissimilarity between the distribution predicted by the model and the actual distribution of labels

2. Test the network on the different target domains (Cartoon, Photo, Sketch) to determine the performance on unseen distributions.

3.2. Activation shaping module

The main component of our implementation is a custom module that performs pre-defined operations on the activation maps of specified layers. Specifically, the module has been implemented as a forward hook - a function that runs

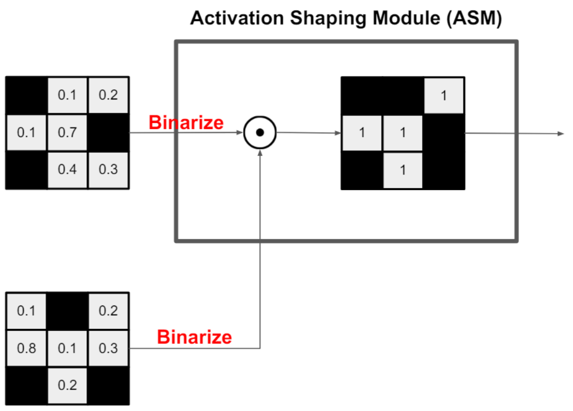

Figure 3. Module implementation

after the forward pass of a given specified layer. This function accepts two input tensors: the current activation map A and a mask M with the same shape, and produces a shaped activation map:

$$A_s = \phi(A, M)$$

The shaping happens as follow: both tensors are first binarized with threshold $t = 0$ such that, beeing x an element of an activation map:

$$binarize(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases}$$

So that

$$A_B = m\_binarize(A) \tag{1}$$
$$M_B = m\_binarize(M) \tag{2}$$

Where $m\_binarize$ applies $binarize$ on all the elements of a tensor. Then, the element-wise product is returned:

$$\phi(A, M) = A_B \circ M_B$$

This new tensor will be the output of the forward pass on the specified layer. The placement of this custom module was tested in different configurations, either by applying it after a single layer or on a set of layers. The results obtained will be analyzed in the following section, so as to determine the impact of placement on performance. We now describe the two main rules we defined to design the M shaping matrix.

3.3. Random Maps

The idea for this shaping rule is to apply random binary masks to the activations produced by the network during training. This technique can be seen as a form of regularization similar to dropout, aiming to prevent the model from overfitting to the source domain by randomly deactivating certain neuron outputs. By doing so, we try to prevent the network from relying too heavily on specific neurons, encouraging it to learn more general features and potentially discarding stye-specific paths. Furthermore, the variability introduced by random masks should mimics real-world data distributions shifts, additionally improving generalization capabilities on unseen domains. Given the custom activation shaping module hooked to a given layer or set of layers, we define M as a random matrix consisting of zeros and random $x \in (0, 1)$ numbers based on a mask ratio that determines their proportion. The tested mask ratio configurations are: 1, 0.9, 0.5, and 0.1. To implement this strategy, beeing A the activation of any hooked layer, we follow these steps during each training iteration:

1. **Generate Random Mask** M: Create a binary mask M with the same shape of A, where each element is randomly set to 0 or $x \in (0, 1)$ based on the specified mask ratio.

2. **Apply Activation Shaping**: Apply the custom shaping function:

   $$A_s = \phi(A, M)$$

   The resulting tensor will be the output for the forward pass on the specified layer.

3. **Forward Pass with Shaped Activation**: Use the shaped activation A_s to compute the network output z_s:

   $$z_s = f_{\text{network}}(x_s \mid A_s)$$

4. **Compute and Perform Backward Pass**

   Compute the Cross-Entropy Loss $\mathcal{L}$ using the network output z_s and the category label y_s:

   $$\mathcal{L} = \text{CrossEntropyLoss}(z_s, y_s)$$

By following these steps, we test the ability of the modified network to focus on domain-invariant features. The mask ratio is an hyperparameter to tune for optimal balance between regularization and feature preservation. The optimal threshold will be discussed in the Experiments section.

3.4. Domain adaptation with alignment

Activation patterns encode both target-specific and domain-specific information, by strategically manipulating them, we hope to learn domain-agnostic representations. To achieve this, we designed the following shaping rule, that is in the end an alignment strategy. Given two samples, one from the Source Domain (Art Painting), x_s for which we also have the corresponding category label y_s, and the other from one of the Target Domains (x_t, unlabeled), at each training iteration, the following steps are performed:

1. **Record Activation Maps** Forward the target domain sample x_t through the network to obtain a list of activation maps out of hooked layers. Let M_t be one of them:

$$M_t = f_{\text{layer}}(x_t)$$

where f_{layer} represents the activation map out of one hooked layer in the network.

2. **Apply Activation Shaping** Use the recorded activation maps M_t to shape the activation of the source domain sample x_s when forwarding it through the network. Let ϕ denote the activation shaping function, then the shaped activation $\hat{M}_s$ at the specified layer will be:

$$\hat{M}_s = \phi(M_t, f_{\text{layer}}(x_s))$$

3. **Forward Pass with Shaped Activation**: Compute the network output z_s using the shaped activation:

$$z_s = f_{\text{network}}(x_s \mid \hat{M}_s)$$

4. **Compute and Perform Backward Pass** Compute the Cross-Entropy Loss $\mathcal{L}$ using the network output z_s and the category label y_s:

$$\mathcal{L} = \text{CrossEntropyLoss}(z_s, y_s)$$

Performing these steps should result in activation maps that are similar to both source and target domain out of hooked layers. As for all the other experiments, we tested this strategy on several different layers and group of layers and report the findings in the Experiments section.

3.5. Extensions

To gather further insights on the shaping process, we designed two extensions of our custom shaping module.

3.5.1 Binarization ablation (NB)

Instead of binarizing the tensor M, keep it as it is and simply multiply it with the activation map A. This modification aims to preserve more information from M while still influencing the activations in a similar way as the binarization process. The module implementation becomes simply:

$$\phi(A, M) = A \circ M$$

Where A is the activation map of a specific layer and M is either a random map or a target activation.

Target domain	Accuracy
Sketch	40.44
Cartoon	54.52
Photo	95.93
Avg	63.63

Table 1. Baseline performances on unseen domains

3.5.2 TopK thresholding of A (TopK)

In this extension, we first binarize M, compute the Top-K values of A, and set all the elements of M that are not in the Top-K to zero. Then, we multiply A and M. Here, K is a hyperparameter. This approach ensures that only the most significant elements of A are retained, possibly enhancing the content-specific information while reducing the impact of noisy activations. The module implementation becomes:

$$\phi(A, M, K) = TopK(A, K) \circ m\_binarize(M)$$

Where A is the activation map of a specific layer and M is either a random map or a target activation. These extensions are implemented directly in the module, where different configurations of binarization and top-K thresholding are applied to shape the activation maps during the forward pass through the network.

4. Experiments

This section reports the results of testing shaping strategies with different configurations. We conducted almost exhaustive experiments in terms of configurations, since most of them are not improving over the baseline, we avoid to report them in the following section.

4.1. Baseline

To benchmark the fine-tuned network, we resort to the unmodified ResNet18 baseline performances. The baseline results on target domains are reported in Table 1.

4.2. Random Maps

Applying random masks to the model aids in fitting more general features, but accuracy is highly dependent on the shaping placement. Considering the different Mask Ratios tested, only 2% of the tests obtained a value slightly higher than the baseline. Furthermore, the 'Photo' domain, which already had very high baseline results, was difficult to improve and the results on it were unsatisfactory. The accuracy only approached the baseline value but never managed to exceed it (Table 2). We found that significant improvements are achieved when shaping layer "2.1.conv2", which outperformed the baseline in challenging domain shifts (i.e:

Layer	Mask Ratio	Accuracy
1.1.bn2	90%	93.23
1.1.bn2	**50%**	**95.33**
3.1.bn1	10%	93.47
3.1.bn2	10%	94.73
4.0.conv1	10%	93.23
Baseline	-	95.93

Table 2. Results on ”Photo” as target domain

Layer	Mask Ratio	Accuracy
2.1.conv2	100%	48.13
2.1.conv2	**90%**	**52.46**
2.1.conv2	50%	48.71
3.0.bn1	50%	40.47
3.1.bn1	10%	40.70
Baseline	-	40.44

Table 3. Best results for ”Sketch” as target

Layer	Mask Ratio	Accuracy
1.1.conv1	100%	54.69
1.1.conv1	90%	56.19
2.1.conv2	90%	58.87
1.1.conv1	50%	55.59
2.1.conv2	**50%**	**59.34**
3.1.bn1	10%	57.12
Baseline	-	54.52

Table 4. Best results for ”Cartoon” Domain as target

Cartoon and Sketch) by 8,42% averaging between target domains. When “Sketch” is used as target domain, the results do not report noteworthy improvements, except in layer 2.1.conv2, which achieved significant improvement with mask ratio 0.5, 0.9 and 1 (Table 3). The “Cartoon” domain also greatly benefits from activation after layer 2.1.conv2 and in general there is a wider range of layers, with different mask ratios, after which apply random masking achieves appreciable results, such as layer 1.1.conv1 (Table 4). Shaping after ReLU activation layers provided extremely low accuracy results for all domains, so we didn’t investigate further. The application of the mask in intermediate layers, such as 2.1.conv2, where there is a balance between detecting less specific image features (e.g., edges, textures...) and more specific features (e.g., a dog’s paw...), is favored by their flexibility and ability to adapt well to noise in the data. On average, the best results were achieved with high mask ratio values ($\geq 50\%$) emphasizing how having lower values, and thus disabled activations, reduces too much the amount of information propagating through the layers undermining accuracy. The tests of the most promising layers were then repeated by replacing zeros with $\epsilon = 0.1$ in the binary mask, in order to reduce some of the information instead of eliminating it completely. However, results were only marginally appreciable; in many cases, the outcome remained nearly unchanged, and in some instances, there was even a degradation in accuracy. The best results obtained using this approach are shown in Table 5, Following the experiments with single layers, we explored configurations of the Top K extension with different thresholds ($t = 1, t = 0.9, t = 0.75, t = 0.25$). No experiments outperformed the results shown in Table 2, Table 3 and Table 4. In the following table, we report the highest values achieved. For NB, tests were performed considering all the best layers from previous experiments. Results were very poor and only those shown in the table are close to the baseline. Applying random masks without the use of Top K led to very disappointing results when hooking groups of layers. In no case the results were better than the baseline. Based on the results of other experiments we conducted but did not report, NB was useless also in this case, so we didn’t investigate further. Instead, as shown in Table 8, improvements were achieved by using TopK on groups of layers. For the sake of completeness, it is possible to further explore this section by considering different groups of layers, as many other possibilities exist.

Domain	Layer	Mask Ratio	ϵ	Accuracy
Sketch	2.1.conv2	90%	0	52.46
Sketch	**2.1.conv2**	**90%**	**0.1**	**53.50**

Table 5. Comparison between different ϵ

Domain Mask Ratio Layer	Acc. t=1	Acc. t=0.9	Acc. t=0.75	Acc. t=0.25
Cartoon 0.9 2.1.conv2	54.74	56.53	54.10	51.15
Photo 0.9 1.1.bn2	95.81	95.39	95.15	95.09
Sketch 0.9 2.1.conv2	40.72	39.60	33.65	26.90

Table 6. RA with Top K on single layers

4.3. Domain adaptation

The key finding for DA is that retaining a high percentage of the original activation results in better performances in all settings, while keeping the target activation as it is, without binarizing it, worsens the performances. In terms of defined rules, this means applying the TopK retention

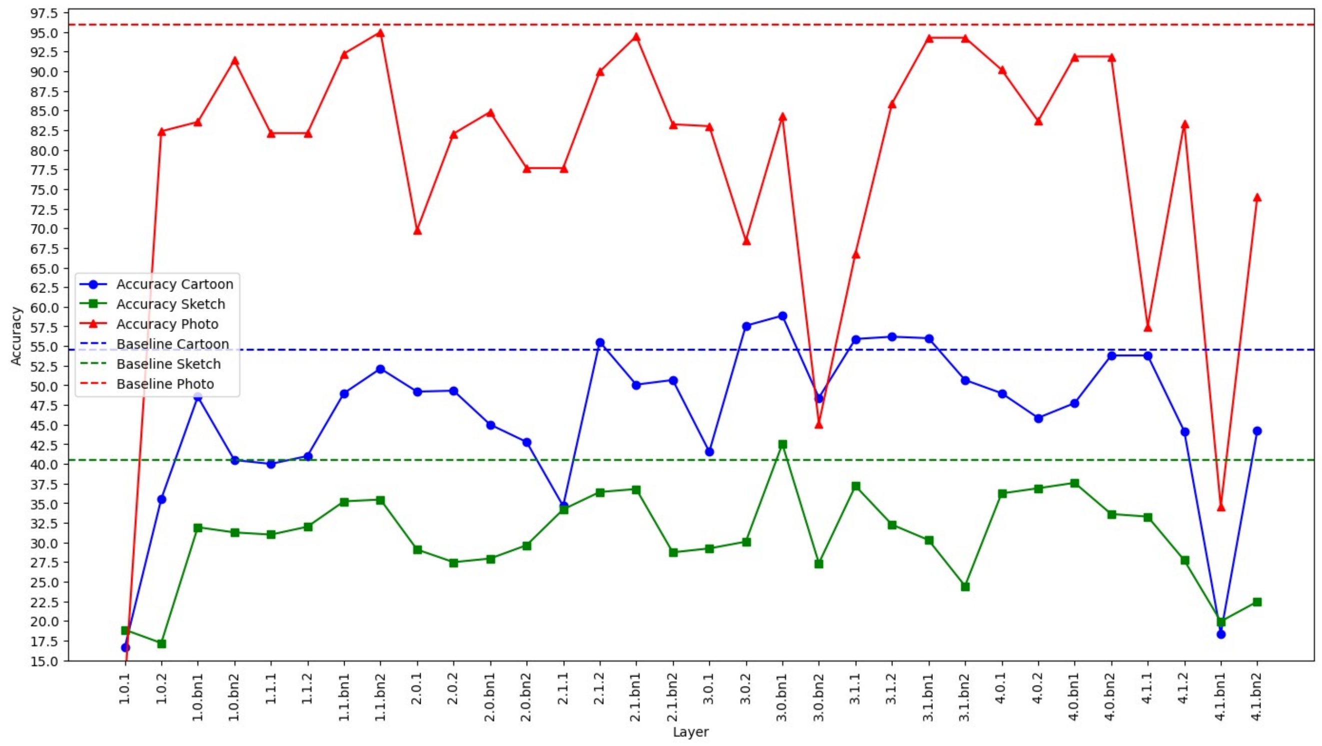

Figure 4. DA accuracy on single layers

Domain	Layer	Mask Ratio	Accuracy
Cartoon	4.0.bn2	0.5	**54.82**
Cartoon	3.1.conv2	0.9	53.24
Cartoon	1.1.conv1	0.9	53.11
Photo	4.1.conv2	0.9	95.45
Photo	4.1.conv2	0.5	94.97
Photo	4.1.bn1	0.1	93.59
Sketch	1.1.conv1	0.9	**41.31**
Sketch	3.1.conv2	0.5	39.58
Sketch	3.1.conv2	0.1	33.75

Table 7. RA with NB on single layers

Domain	Layers	Mask Ratio	Top K	Acc.
Cartoon	1.1.conv2 3.1.conv2	1	0.75	58.87
Sketch	1.1.conv2 3.1.conv2	1	0.75	41.90
Cartoon	X.1.bn2	1	0.9	59.77
Sketch	X.1.bn2	1	0.9	42.15
Sketch	2.1.bn2, 4.1.bn2	1	0.9	42.38
Cartoon	X.1.conv2	1	0.9	58.49
Cartoon	X.1.conv2	1	0.75	58.53
Cartoon	X.1.conv2, X.1.bn2	1	0.9	58.62

Table 8. RA with TopK on multiple layers(X stands for all possible blocks)

with a high percentage for K. The best alignment is thus achieved by binarizing the target activation and retaining the source one as it is for the most part. This suggests that by focusing on the most significant components of target activations, the model emphasizes features that are more likely to be content-specific. Moreover. binarizing target activations enforces a more rigid structure that the source activations can be more easily aligned to. By retaining most of the original activations in the source domain, we instead maintain the rich content-specific information that must be leveraged for classification tasks. With this setting, we achieved an average improvement over the baseline of 7.44% in challenging domain shifts. The conclusion is that source activations provide detailed content-specific and discriminant information, while the binarized target activations help in reducing domain-specific noise, facilitating better alignment and generalization across domains. Another evidence is that hooking multiple layers improves more than hooking a single one, but hooking too many worsen the performance. We found that for most experiments, the ideal number of layers is between 2 and 4. Moreover, groups of later layers show the best overall results. This is supported by the fact that CNNs typically learn abstract features in later layers. By applying the module on deeper layers, we were able to better align higher-level, more semantically meaningful

Table 9. DA on single layers with TopK

	Cartoon				Sketch			
Layer	K=100	K=90	K=50	Baseline	K=100	K=90	K=50	Baseline
1.1.bn1	58.62	**59.04**	58.75	54.52	40.44	**40.60**	40.25	40.44
2.1.2	39.89	35.75	27.27	54.52	24.46	22.26	22.26	40.44
3.0.2	**55.80**	53.88	46.47	54.52	44.74	**44.80**	36.27	40.44
3.1.2	**58.45**	57.2	56.19	54.52	**43.19**	43.10	42.80	40.44
3.1.bn2	**55.12**	54.52	54.99	54.52	**40.75**	40.34	40.24	40.44
4.0.2	55.67	55.55	**57.17**	54.52	42.02	43.12	**43.90**	40.44
4.0.bn2	**55.63**	55.59	55.16	54.52	**43.47**	43.40	43.33	40.44
4.1.2	56.44	57.94	**59.60**	54.52	44.44	**47.70**	46.25	40.44
4.1.bn2	**60.49**	60.37	60.20	54.52	46.58	**46.78**	46.25	40.44

Target	layers 4.1.2, 4.1.bn2, K = 90%
Sketch	**49.27**
Cartoon	**60.58**
Photo	94.73
Avg	**68.19**
Baseline	63.63

Table 10. Best configuration for DA

Layer	NB Accuracy
2.1.2	18.39
3.0.2	16.20
3.0.bn1	20.22
3.1.2	28.73
Baseline	54.45

Table 11. "Cartoon" - DA with NB on single layers

representations. In addition, including batch normalization layers is another cause for improvement, this could be due to the fact that aligning in already normalized spaces is easier. Table 10 backs ups these findings, showing the configuration that achieved the best results on average for this strategy. In the following tables, unreported layers experiments perform worse than the baseline. Considering just challenging domains (Cartoon and Sketch), this configuration improves the baseline accuracy by a 7.44%, while overall gain is 4,56%. Later layers including batch normalization achieve the best results. Fig 4 shows the accuracy over hooked layers without TopK thresholding, some of the intermediate layers improve in challenging domains but overall, without TopK this strategy is not particularly effective. Table 11 shows the effect of NB on single layers: keeping the target activation as it is introduces disruptive noise. We experimented this extension on already improving layers, results discouraged us from further testing other domains and configurations, as it is clear that NB performs poorly.

Table 9 is a comprehensive summary of experiments on single layers applying TopK thresholding. In bold the improving configurations. Note that TopK with K = 100% means keeping original map as it is, without binarizing it. Results also shows that with K above 50%, there are no significant differences in performance when changing its value, while below 90% accuracy always decreases. Additionally, epsilon smoothing always worsen the performance when applying TopK. Table 12 shows a comprehensive summary of the performances when sets of layers are hooked with Top K thresholding. Not applying TopK on groups of layers always worsens the baseline performance. Applying a K that is less than 90% always worsens the performances with respect to higher values of K, and between 90% and 100% there's no significant difference in accuracy in all settings. For this table then, TopK must be intended with K = 90%. Experiments that outperform the baseline are highlighted in bold. The following summarizes our findings for the DA setting:

- Binarizing the original activation A consistently results in decreased performance. Instead, utilizing a TopK retention strategy is more effective.
- The value of K in the TopK retention strategy should be relatively high, greater than or equal to 90.
- Not binarizing the target activation M consistently leads to decreased performance.
- Deeper layers within the network consistently demonstrate superior performance in domain adaptation tasks.
- Grouping 2 to 4 layers together yields optimal performance; excessively shaping the network's architecture proves detrimental.

Layers	Cartoon	Sketch
1 in 3 conv avg	17.28	..
1 in 3 BN avg	18.23	..
All bn1s	20.67	13.2
All bn2s	25.67	17.2
All improving single layers	24.25	15.4
3.1.bn2 4.1.bn2	**59.22**	**45.61**
2.1.bn2 3.1.bn2 4.1.bn2	**59.22**	**45.12**
3.0.2, 3.0.bn2, 4.0.2, 4.0.bn2	49.02	36.34
3.1.2 3.1.bn2	**57.21**	**45.30**
3.1.2 4.1.bn2	**61.56**	**46.68**
3.1.2 3.1.bn2 4.1.2 4.1.bn2	**59.94**	**48.59**
2.1.2 2.1.bn2 3.1.2 3.1.bn2 4.1.2 4.1.bn2	30.67	**59.94**
4.1.2 4.1.bn2	**60.58**	**49.27**
Baseline	54.52	40.44

Table 12. DA on multiple layers with TopK, K = 90%

- Groups that include batch normalization (bn) layers exhibit the highest performance.

- Epsilon smoothing does not yield significant improvements in performance.

5. Conclusions

In this work, we investigated the potential of Activation Shaping for Unsupervised Domain Adaptation, specifically focusing on isolating content-specific paths within a baseline network. Leveraging insights from the field of OOD, we explored the hypothesis that selectively operating on activation maps can enhance generalization capabilities by mitigating style discrepancies between source and target domains. Our experimental results on the PACS dataset demonstrate the effectiveness of simple activation shaping techniques in improving UDA performance compared to an unmodified baseline network. By targeting specific layers and retaining a high percentage of original activations, we achieved improvements in challenging domain shifts. The simplicity and effectiveness of ASH highlight its potential for broader applications in domain adaptation and transfer learning scenarios. Further research can explore dynamic and adaptive activation shaping rules, hybrid models combining ASH with other UDA techniques, and alternative baseline architectures to gather stronger evidence for the effectiveness of ASH in UDA scenarios.

References

[1] Sun B. and K. Saenko. Deep coral: Correlation alignment for deep domain adaptation, 2016. 2

[2] Silvia Bucci, Mohammad Reza Loghmani, and Tatiana Tommasi. On the effectiveness of image rotation for open set domain adaptation, 2020. 2

[3] Fabio M. Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles, 2019. 2

[4] Fabio Maria Carlucci, Lorenzo Porzi, Barbara Caputo, Elisa Ricci, and Samuel Rota Bulò. Autodial: Automatic domain alignment layers, 2017. 2

[5] A. Djurisic, A. Bozanic, N. an Ashok, and R. y Liu. Extremely simple activation shaping for out-of-distribution detection, 2020. 1

[6] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, Francois Laviolette, Maria Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks, 2016. 2

[7] Muhammad Ghifary, W. Bastiaan Kleijn, Mengjie Zhang, David Balduzzi, and Wen Li. Deep reconstruction-classification networks for unsupervised domain adaptation, 2016. 2

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 1, 2

[9] Bousmalis K, Silberman N., Dohan D, and Krishnan D Erhan D. Unsupervised pixel-level domain adaptation with generative adversarial networks, 2017. 2

[10] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Deeper, broader and artier domain generalization, 2017. 1, 2

[11] Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou. Revisiting batch normalization for practical domain adaptation, 2016. 2

[12] Xiaofeng Liu, Chaehwa Yoo, Fangxu Xing, Hyejin Oh, Georges El Fakhri, Je-Won Kang, and Jonghye Woo. Deep unsupervised domain adaptation: A review of recent advances and perspectives, 2022. 2

[13] Mingsheng Long, Yue Cao, Jianmin Wang, JIMWANG, and Michael I. Jordan. Learning transferable features with deep adaptation networks, 2015. 2

[14] M. Long, J. Wang, and M. I. Jordan. Unsupervised domain adaptation with residual transfer networks, 2016. 2

[15] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I. Jordan. Deep transfer learning with joint adaptation networks, 2017. 2

[16] Muhammad Muneeb Saad, Ruairi O'Reilly, , and Mubashir Husain Rehmani and. A survey on training challenges in generative adversarial networks for biomedical image analysis, 2023. 2

[17] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks, 2017. 2

[18] Yifei Wang, Wen Li, Dengxin Dai, and Luc Van Gool. Deep domain adaptation by geodesic distance minimization, 2017. 2

[19] Jiaolong Xu, Liang Xiao, and Antonio M. Lopez. Self-supervised domain adaptation for computer vision tasks, 2019. 2

[20] Zhang Youshan. A survey of unsupervised domain adaptation for visual recognition, 2021. 2