Domain Adaptation via Activation Shaping

Bar Giorgio
Politecnico di Torino
s188295@studenti.polito.it

Distefano Giuseppe
Politecnico di Torino
s309787@studenti.polito.it

Incaviglia Salvatore
Politecnico di Torino
s305947@studenti.polito.it

Abstract

Domain adaptation is a discipline founded on the premise that all predictions can rely on features that are indiscriminate between the Source and Test domains. It underlies the study of Domain Shift, which focuses on differences in the distributions of data between the Test and Source domains, resulting in a deterioration in the performance of the model itself.

Activation Shaping comes to the aid of this research through the modification of activation maps in an architecture, and can be used as a method in the out-of-detection (OOD) process by answering the question "Do models know when they do not?".

Our project begins with the retrieval of activation maps to apply to the output of the reference ResNet18 model, and subsequently we will try to switch off some outputs to see how this process can be adaptive to changing network conditions. Finally, we continue in Domain Adaptation by combining the characteristics of the Test domain with the Source domain and manipulating the architecture from previous stages with some binarization ablation experiments.

The code of our project can be found at https://github.com/Giuseppe-Distefano/davas.

1. Introduction

When dealing with different sources for data, current deep learning models face a significant challenge in ensuring consistent performance between the training and testing phases, due to the differing data distributions across the two domains. This issue, known as Domain Shift, has been extensively explored through the analysis of biases present in the most common datasets [1], as well as the observed lower performances of state-of-the-art models when applied to different datasets. Domain Shift is formalized with the settings of modern Domain Adaptation approaches [2], including prominent methods such as Unsupervised Domain Adaptation, which leverages unlabeled target domain data to adapt the source domain model to a specific target domain. Recent studies on Domain Adaptation [3], following the logic of Domain Shift and expanding upon it, have further focused on combined representations of domain adaptation and deep feature learning within a single training process: the goal is to ensure that final decisions are made taking into account discriminative features that are invariant to domain changes. Consequently, the generated network becomes invariant to the proposed domains, yielding a representation where the algorithm cannot learn to identify the domain of origin of the input observation [4] [5].

The discussion is connected to the challenges of out-of-distribution (OOD) generalization, that highlights the current models' struggles in recognizing data outside the training distribution. Thus, distinguishing between OOD and in-distribution (ID) data can be an exceptionally difficult task. Model calibration emerges as a potential path to enhance OOD detection capability, with the help of recent studies focusing on creating models more resilient to domain shift between source and target. Noteworthy methods for OOD detection include ReAct [6], proposing activation corrections in a specific (penultimate) layer to improve ID and OOD separation, and DICE [7], utilizing weight sparsity on a specific layer in conjunction with ReAct to achieve leading OOD detection outcomes.

In this paper we focus on Domain Adaptation leveraging on Activation Shaping (ASH), a method that relies on modifying activation maps within an architecture according to user-defined or learned rules. The aim is to enhance the architecture's robustness by studying the feature distribution of a domain to improve accuracy and loss metrics. We adopted the PACS dataset, which consists of images from the "Photo", "Art", "Cartoon", and "Sketch" domains, and encompasses them with a wide range of visual abstraction, spanning from more realistic representations to more stylized ones [8].

The first stage of our research consists in constructing an architecture that initially modifies the activation maps of the output layers of the chosen used. Then, we use random activation maps placed at the outputs of network stages to further understand the impact of activation shaping hooks. Then, we have the Domain Adaptation phase, in which characteristics learned from samples from a target domain are transferred to those from the source for analysis. We conclude our project with some experiments to make further considerations on results of previous steps.

2. Related work

2.1. Domain Adaptation

Domain adaptation is aimed at learning a classifier in presence of a shift between the distributions of training and test data. The objective is thus to be able to map reference domains in situations where the target domain may have zero or few labels associated with samples. Unsupervised learning research comprises various approaches, like the explicit selection of samples from the source domain [9] or the trans-

formation of the feature space to map the source domain to the target domain [10].

In unsupervised domain adaptation [3] the goal is to minimize the discrepancy between the distribution of data from a source domain $\mathcal{D}_s$ and that of a target domain $\mathcal{D}_t$. Formally, given a model f parameterized by θ, the objective is to minimize a combination of the classification loss $\mathcal{L}_s$ on the source domain and a measure of discrepancy $\mathcal{D}(\mathcal{D}_s, \mathcal{D}_t)$ between the two domains. The overall objective function can be expressed as:

$$\min_{\theta} \mathcal{L}_s(f(x_s; \theta), y_s) + \lambda \mathcal{D}(\mathcal{D}_s, \mathcal{D}_t)$$

where x_s and y_s represent the data and labels from the source domain, and λ is a hyperparameter that balances the two terms.

The method transforming the feature space to map the source domain to the target domain [10] is used, for example, by Domain Adversarial Neural Networks [3] to adapt feature space distributions without the need for re-weighting or geometric transformations by means of a unified architecture and a single learning algorithm, which contrasts sharply with previous architectures.

2.2. Activation Shaping

The Activation Shaping (ASH) method represents a cutting-edge advancement in OOD detection techniques, employing a post-hoc simplification approach to input data. This process allows to mitigate the need to add parameters to features within a standard Neural Network, potentially reducing parameter redundancy and improving performance estimation accuracy. The concept behind ASH is a regularization process [11] for neural networks, assuming that neural networks can be over-parameterized, leading to an increase in redundant information [12] and to a heavier activity of feature cleaning.

In the study of the activation shaping, one of the first insights involves removing a portion of the activations basing on a top-K criterion [11], followed by adjusting the remaining values through simple scaling or fixed value yielding a simplified feature representation, suitable for out-of-distribution (OOD) classification. This kind of research is similar to ReAct [6] in its post-training and one-shot approach within the activation space. However, the fundamental differences lie in the fact that the operations are entirely post hoc, and ASH allows for greater flexibility in choosing the layers to use for experiments, resulting in improved performance in terms of accuracy preservation on in-distribution (ID) data.

Unlike DICE [7], ASH operates exclusively within the activation space. By doing so, the network model requires fewer inference runs per input image and it contrasts with further methods such as Monte Carlo dropout [13], where dropout layers are added both during training and testing, generating multiple predictions of the same input instance.

2.3. Forward Hooks

A novel approach to extracting the activations of the modules within the reference ResNet18 is to utilize forward hooks provided by PyTorch library. They are invoked after the model's forward pass has produced an output, allowing for the modification of this output, or the extraction of features from a specific module's output. We used this type of approach in our project to extract activation maps and conduct the project phases accordingly.

2.4. PACS dataset

The PACS dataset is the result of merging classes from Caltech256 [14], Tu-Berlin [15], and Google Images. The classes from these datasets have been combined to create our reference benchmark, where we have 4 reference domains: "Art-Painting", "Cartoon", "Sketch", and "Photo". Each domain contains samples belonging to 7 classes: 'dog', 'elephant', 'giraffe', 'guitar', 'horse', 'house', and 'person'.

The domains are used in the experimentation phase to evaluate the performance of the model in terms of accuracy and cross-entropy loss. We started from the ResNet18 network, which we then modified according to the problem's specifications.

3. Baseline

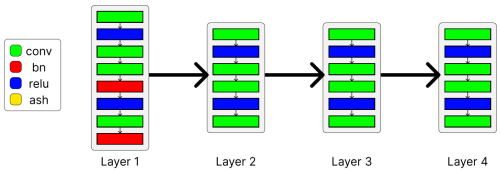

Figure 1. ResNet18

During the training phase, knowledge of the basic architecture and metrics as a criterion for evaluating the goodness of the developed model constitutes a fundamental element for delving deeper and seeking improvements to the initial baseline solution. The neural network referenced in our project on Domain Adaptation via Activation Shaping is ResNet18, convolutional neural network (CNN) created by Microsoft Research Asia in 2015. The general structure of ResNet18 [16] consists of characteristic elements that make it excellent for image classification:

1. **Image Input**: Accepts images sized 224x224 pixels.
2. **Convolutional-Layers**: Commences with an initial 7x7 convolutional layer employing a stride of 2, succeeded by a max pooling layer using a stride of 2. Following this, there exist four sets of convolutional layers, each featuring two 3x3 convolutional layers. Every set is succeeded by a direct connection for shortcut.
3. **Direct Connections**: These connections circumvent one or more convolutional layers, directly linking to a subsequent layer within the network. This mechanism aids in the management of gradient vanishing and facilitates deeper network training.
4. **Global Average Pooling**: The outcome of the final convolutional layer goes through a global average pooling layer, averaging the feature maps across spatial dimensions.

5. **Fully Connected Layer**: The next layer is a fully connected layer made of 7 units, culminating in the network's final output.

6. **Activation via Softmax**: The outcome of the fully connected layer undergoes activation through a softmax function, generating a probability distribution across the 7 classes in the PACS dataset.

7. **Network Depth**: ResNet-18 encompasses a total of 18 layers, presenting as a comparably shallow network when juxtaposed with alternative deep learning architectures.

During the training phase, key metrics employed on the project are Accuracy and Cross-Entropy Loss 2D. The loss function acts as a tool to adjust the neural network weights, indicating the model's deviation from the correct prediction. Specifically, the cross-entropy loss considers probabilities outputted by the softmax function, which generates a vector of predicted probabilities across input classes, to calculate the loss. Each predicted class probability is compared to the desired output of 0 or 1, and the resulting score (loss) penalizes deviations from the expected value. We aim at minimizing the loss and at achieving the highest accuracy following the application of activation shaping in the ResNet18 in various forms.

Table 1 shows a representation of the baseline results in terms of accuracy and loss for the initial ResNet18 model in the test domains Cartoon, Sketch, and Photo. These results confirm the observations presented in the paper [11], which serves as the starting point for our project.

Target	Accuracy	Loss
Cartoon	54.52	0.01067
Sketch	40.57	0.01512
Photo	95.87	0.00107

Table 1. Results for Baseline.

4. Experiments

4.1. Finding the best configurations

The initial phase of the project involved constructing a custom activation shaping layer using a a forward hook that takes as input parameters a matrix A, representing the activation/output map of the current layer, and a tensor M with the same shape as the reference output. We binarize these two tensors and then perform the bitwise product of A and M.

The decision of which layers of ResNet18 to apply the various hooks to was carried out following two different approaches: an incremental top-down approach aimed at gradually selecting the most promising layers in terms of prediction accuracy, and an experimental approach that, starting from the basic configurations, serves to explore solutions that the former approach could ignore. During this phase, we filtered out 50% of details of an image to mimic domain uncertainty.

Before starting with the incremental approach, we tried some network configurations by placing activation shaping hooks according to some ideas or heuristics. Specifically, we placed an ASH at the output of every convolutional layer, after every three convolutional layers, and after the last convolutional layer of the whole network. Table 2 shows the average improvements we recorded: the average performance decreases considerably in the first two cases, while for the latter case the average decrease is just below 10% in accuracy, although it remains too high as well.

Placement	Avg improvement
After every convolution	-49.31%
After every three convolutions	-45.23%
After last convolutional layer	-09.04%

Table 2. Average improvements for experimental configurations

The proposed incremental approach begins by considering a single forward hook at a time. As shown in Figure 2, these initial configurations generally led to a significant decrease in average network performance in terms of accuracy. However, there are some exceptions: for certain layers, whose nominal performances over domains are represented in Figure 7, and average performance is numerically explained in Table 3, the decrease in performance is acceptable, and in some cases we recorded an improvement over baseline. In particular, placing an activation shaping hook after *layer 2.1.conv2* brings an average performance increase of about 2%, while placement of the ASH after other layers like *layer1.1.bn2* (using PyTorch naming convention) or *layer3.0.conv1* experienced a smaller decrease than other cases.

We then tried to combine these results, placing more activation shaping hooks after the best-performing layers. As represented in Figure 3, Figure 4, Figure 5, and Figure 6, the analyses performed by incrementally placing first two, then three, up to five activation modules at a time, result in an even more significant decrease in average performance. However, it is noticeable that the configurations showing better trends are those including the activation shaping hook modules placed after *layers 2.1.conv2* and *layer1.1.bn2*, confirming the predictions made in the first step where these layers were considered individually. Table 4, Table 5, Table 6, and Table 7 show the average results for the best configurations for each of the mentioned sets of experiments, always considering the averages across the three target domains (Cartoon, Sketch, and Photo). For these configurations, a graphical representation of individual performance for each domain is reported in Figure 8, Figure 9, and Figure 10, respectively. The target domains that show substantial performance improvements are Cartoon and Sketch, while the third domain, Photo, usually experiences a significant decrease.

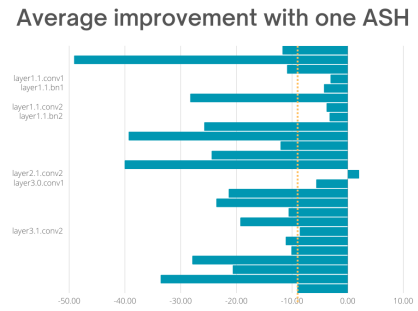

Figure 2. Average improvement w.r.t. position when placing one ASH in the network

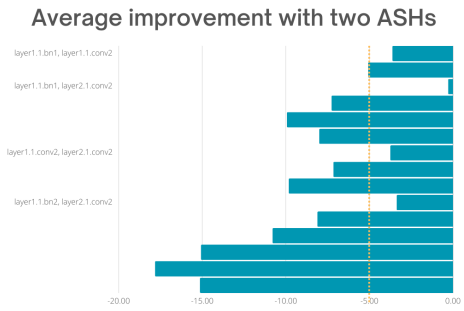

Figure 3. Average improvement w.r.t. position when placing two ASHs in the network

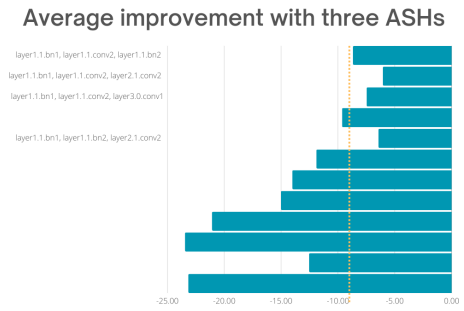

Figure 4. Average improvement w.r.t. position when placing three ASH in the network

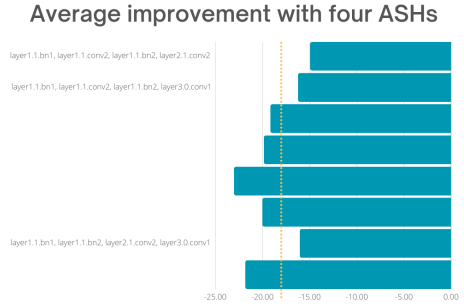

Figure 5. Average improvement w.r.t. position when placing four ASHs in the network

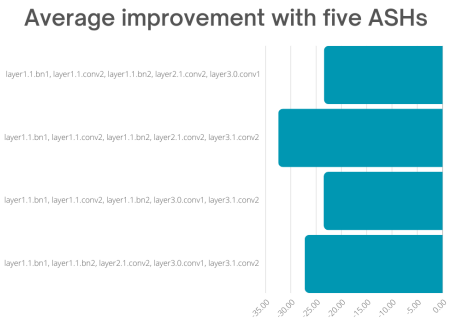

Figure 6. Average improvement w.r.t. position when placing five ASHs in the network

Placement	Avg improvement
layer1.1.conv1	-3.12%
layer1.1.bn1	-4.30%
layer1.1.conv2	-3.82%
layer1.1.bn2	-3.29%
layer2.1.conv2	+2.05%
layer3.0.conv1	-5.68%
layer3.1.conv2	-8.63%

Table 3. Best placements for a network with one ASH.

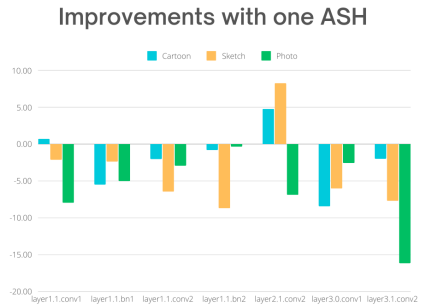

Figure 7. Improvements for best configurations with one ASH in the network

Placement	Avg improvement
layer1.1.bn1, layer1.1.conv2	-3.63%
layer1.1.bn1, layer2.1.conv2	-0.29%
layer1.1.conv2, layer2.1.conv2	-3.75%
layer1.1.bn2, layer2.1.conv2	-3.37%

Table 4. Best placements for a network with two ASHs.

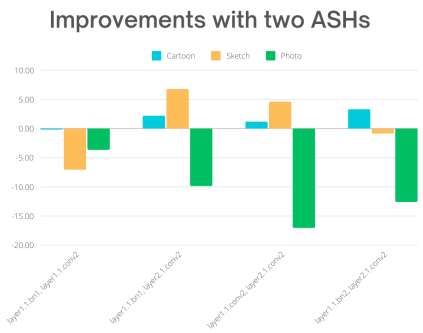

Figure 8. Improvements for best configurations with two ASHs in the network

Placement	Avg improvement
layer1.1.bn1, layer1.1.conv2, layer1.1.bn2	-8.68%
layer1.1.bn1, layer1.1.conv2, layer2.1.conv2	-6.05%
layer1.1.bn1, layer1.1.conv2, layer3.0.conv1	-7.48%
layer1.1.bn1, layer1.1.bn2, layer2.1.conv2	-6.44%

Table 5. Best placements for a network with three ASHs.

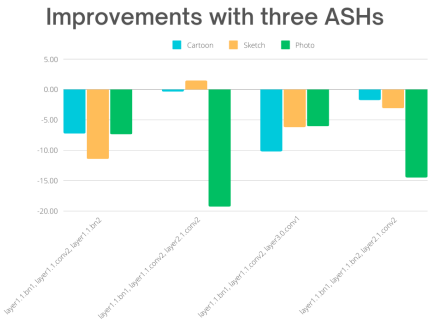

Figure 9. Improvements for best configurations with three ASHs in the network

Placement	Avg improvement
layer1.1.bn1, layer1.1.conv2, layer1.1.bn2, layer2.1.conv2	-14.95%
layer1.1.bn1, layer1.1.conv2, layer1.1.bn2, layer3.0.conv1	-16.23%
layer1.1.bn1, layer1.1.bn2, layer2.1.conv2, layer3.0.conv1	-16.05%

Table 6. Best placements for a network with four ASHs.

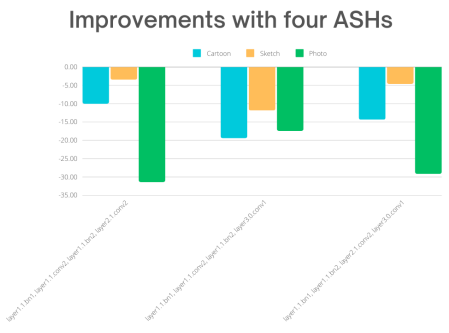

Figure 10. Improvements for best configurations with four ASHs in the network

Placement	Avg improvement
layer1.1.bn1, layer1.1.conv2, layer1.1.bn2, layer2.1.conv2, layer3.0.conv1	-23.45%
layer1.1.bn1, layer1.1.conv2, layer1.1.bn2, layer2.1.conv2, layer3.1.conv2	-32.48%
layer1.1.bn1, layer1.1.conv2, layer1.1.bn2, layer3.0.conv1, layer3.1.conv2	-23.50%
layer1.1.bn1, layer1.1.bn2, layer2.1.conv2, layer3.0.conv1, layer3.1.conv2	-27.25%

Table 7. Placements for a network with five ASHs.

4.2. Applying random activation maps

The second phase of the project involves deactivating some of the outputs from each layer by setting a random percentage of binary outputs of value 1 to 0. Only configurations we found out being the most promising in the previous stage of the project were considered. Even for this stage of our studies we followed both a progressive and a deterministic approach, by varying the number of deactivated outputs for each layer. Configurations were tested with 0%, 25%, 50%, 75%, and 100% of outputs deactivated for the progressive case.

As we could expect, we can see in Figure 11 that the best average performance is achieved by deactivating a percentage of outputs not exceeding 50%, because of the significant lack of data considered in the input. Furthermore, it is confirmed that positioning an ASH after *layer2.1.conv2* leads to an effective increase in accuracy, which, even with a degree of removal of information of 40%, improves the model's average accuracy by 2.12%.

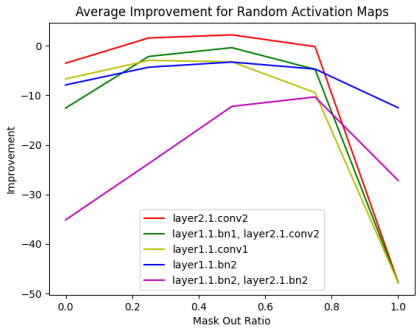

Figure 11. Average improvement varying the amount of image details

Table 8 shows the average performance noted on the various layers considered with different occlusions on the activation maps for incremental analysis. For the following phases of the project, involving Domain Adaptation, only the first two configurations of the table will be further examined, as they have been demonstrated to perform better than the others, loosing at most 16% of accuracy with respect to baseline.

Placement	Avg improvement
layer1.1.bn2	-10.91%
layer2.1.conv2	-15.88%
layer1.1.bn1, layer2.1.conv2	-21.56%
layer1.1.conv1	-23.39%
layer1.1.bn2, layer2.1.bn2	-36.24%

Table 8. Average improvements for studied configurations

4.3. Domain Adaptation

We now delve into the heart of our studies: Domain Adaptation using Activation Shaping. We performed a forward pass on the different target domains to obtain the activation maps M_t, and then we applied them during the forward pass in the source domain with a new output $Z_s = x_s * M_t$, on which we conducted the necessary analyses. Figure 12 compares the Baseline (dashed lines for each domain) with the best network configurations we built and studied in the previous steps of this project using both the incremental and the experimental approaches.

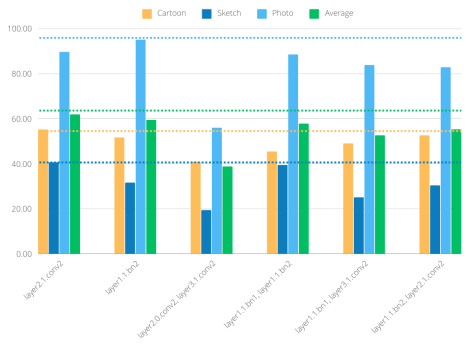

Figure 12. Results of DA over some configurations resulted from the incremental approach

We then conducted further experiments by analyzing, with an experimental approach, the placement of additional activation shaping hooks in various sections of the reference network. We opted to keep placing only a few modules at a time, given the aforementioned performance with the incremental approach. As illustrated in Figure 13, the additional configurations we tested show a significant decrease in accuracy across most target domains. The only instance where a slight improvement is observed is when positioning the activation module after *layer3.1.conv2* with respect to the Cartoon target domain; however, the other domains show a significant deterioration in performance.

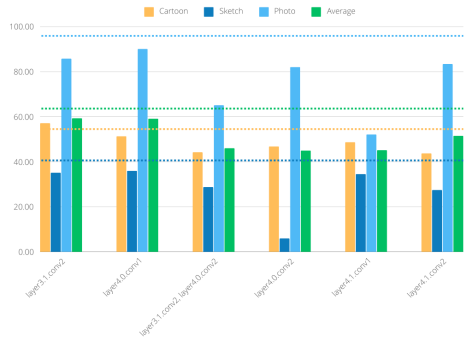

Figure 13. Results of DA over some configurations resulted from the experimental approach

We can conclude the best results come out placing the activation module at the output of *layer2.1.conv2* or *layer1.1.bn2*. In the first case, the accuracies slightly improve performance in the Sketch and Photo target domains, with an increase in performance considering the target domain Cartoon for the second configuration.

Table in Figure 14 reports results for these two configurations: we can observe an average decrease of accuracy for both of them, significantly dropping to below 4.5%.

5. Ablation studies

In this section, we perform ablation experiments to assess the impact of different configurations of our custom activation shaping layer on model performance. We explore two variations: (1) retaining M in its original form and multiplying it by A, and (2) binarizing M by setting the non-Top K values of A to zero before multiplication. These experiments aim at elucidating the significance of these modifications on the network's efficacy.

5.1. First variation

Considering the performance achieved in the previous points with the incremental approach and the significant performance improvement observed by placing the activation module in the first layers, particularly after *layer2.1.conv2* and *layer1.1.bn2*, we decided to conduct the first variation on networks containing an activation shaping hook after these two layers, respectively, to evaluate their impact on overall performance. The use of non-binarized random activation maps M, despite the modifications implemented, confirms the positive estimates observed in the previous points. As

Placement	Target	Accuracy	Loss	Improvement	Avg Improvement
	Cartoon	55.42	0.00968	+1.00%	
layer2.1.conv2	Sketch	39.63	0.01254	-0.94%	-2.04%
	Photo	89.70	0.00236	-6.17%	
	Cartoon	51.71	0.01146	-2.81%	
layer1.1.bn2	Sketch	31.61	0.02015	-8.96%	-4.16%
	Photo	95.15	0.00137	-0.72%	

Figure 14. Performance for best configurations

shown in Table 9 and Table 10, varying the mask out ratio, that is, the amount of discarded information from a sample, from 0% to 50% resulted in good average values across all three test domains — Cartoon, Sketch, and Photo. Notably, the maximum positive increase in accuracy was observed with a mask out ratio of 50% for network with ASH after *layer2.1.conv2* (+2.32%), while a decrease was noted for network presenting ASH after *layer 1.1.bn2* at the same mask out ratio, with a drop of just below 4% (-3.11%).

Placement	Avg improvement
layer2.1.conv2	+1.89%
layer1.1.bn2	-4.43%

Table 9. First variation, mask out ratio = 25%

Placement	Avg improvement
layer2.1.conv2	+2.32%
layer1.1.bn2	-3.11%

Table 10. First variation, mask out ratio = 50%

Placement	Avg improvement
layer2.1.conv2	-1.79%
layer1.1.bn2	-4.12%

Table 11. First variation, with Domain Adaptation

As presented in Table 11, the same experiments conducted on *layer2.1.conv2* and *layer1.1.bn2* for Domain Adaptation confirmed the average analyses performed in the previous points. They demonstrate that the average decrease in model accuracy is below 2% for the former (-1.79%) and below 4.5% for the latter (-4.12%).

5.2. Second variation

In the second part of this section, after binarizing the activation map M, our goal was to test various combinations of the hyperparameter K to highlight the top K values of the output matrix of the involved layers, setting the remaining values to zero accordingly. We tried several combinations, using a technique that involved as less layers as possible, given the results obtained in the previous steps.

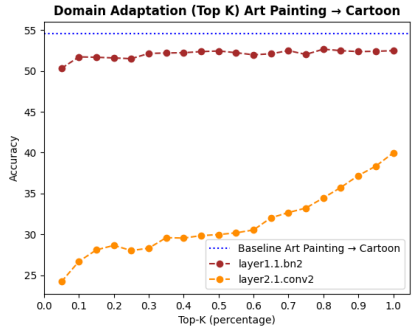

Figure 15. Top-K over Cartoon

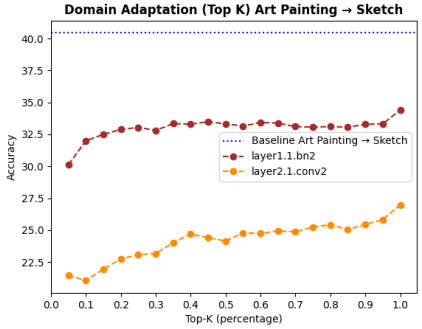

Figure 16. Top-K over Sketch

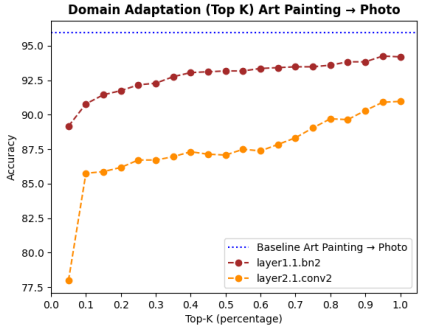

Figure 17. Top-K over Photo

As shown in Figure 15, Figure 16, and Figure 17, the experiments on Domain Adaptation with the activation shaping hook placed at the output of *layer 1.1.bn2* and *layer2.1.conv2*, respectively, show a considerable decrease in performance when compared to the baseline, although in the case of the Cartoon target domain the output of *layer2.1.conv2* is similar to the initial performance.

Considering the poor results of the two configurations that achieved the best performances in the experiments of sections 4.2 and 4.3, (*layer 1.1.bn2* and *layer2.1.conv2*), we undertook further experiments with different configurations.

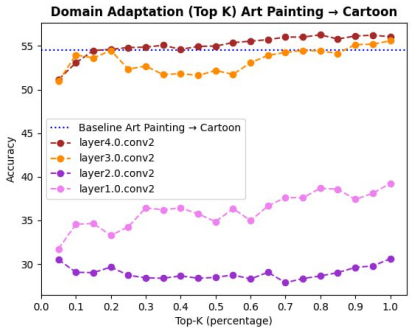

Figure 18. Top-K over Cartoon

Figure 18 illustrates how positioning the activation module at the output of *layer 4.0.conv2* results in a significant improvement in performance for the Cartoon target domain, with a peak in accuracy reached for a K value of 0.8.

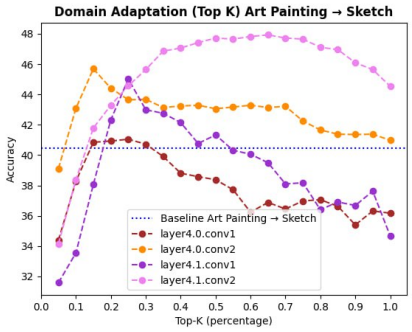

Figure 19. Top-K over Sketch

Figure 19 reports the measured performance for some experiments over the layer 4 for the Sketch target domain, highlighting how the activation shaping hook positioned at various convolution outputs generally produces a considerable increase in accuracy. Notably, placing an activation shaping hook after *layer4.1.conv2* reaches a peak accuracy of 48% with a K value of 0.6, recording an improvement of about 14 percentage points over the baseline. Also noteworthy is the behavior of the activation module at the output of *layer4.0.conv2*, where an accuracy of 46% is achieved using only a K=0.15.

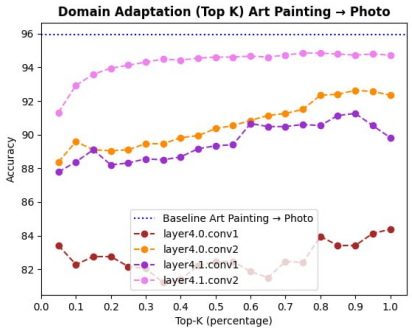

Figure 20. Top-K over Photo

In Figure 20 we can notice that the performance on the Photo target domain does not experience any increase, likely due to the very precise nature of this domain and the already high baseline performance.

6. Conclusions

This paper presents a novel approach to Domain Adaptation using activation shaping hooks. Through a series of experiments, we investigated the integration of new activation modules, derived from PyTorch's forward_hook methods, into various outputs of stages of ResNet18. Initially, we explored the concept at a high level of abstraction before focusing on the core challenge of Domain Adaptation. Our findings indicate that optimal performance was achieved when employing a limited number of activation modules, often just one. However, in most cases, there was a degradation in performance in terms of accuracy and loss, except when analyzing central layers like *layer2.1.conv2* or initial layers such as *layer1.1.bn2*.

We extended our experiments with ablation studies. This extension allowed for a re-evaluation of previous experiments, shifting from binary activation maps to non-binary ones. Additionally, it facilitated the exploration of top-K values within the maps, thereby enabling the observation of the modified ResNet18 behavior. Notably, this extension revealed significant enhancements, particularly in the target domains of the final stages of layer 4, building upon earlier considerations for Cartoon, Sketch, and Photo domains.

In conclusion, our method lays a solid foundation for Domain Adaptation research. The results obtained serve as a valuable springboard for future studies, which may incorporate additional techniques to further enhance the efficacy of domain adaptation solutions.

References

[1]A. Torralba and A. A. Efros, "Unbiased look at dataset bias", CVPR (2011).

[2]K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains", ECCV (2010).

[3]Y. Ganin and et al., "Domain-adversarial training of neural networks", The journal of machine learning research (2016).

[4]S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira, “Analysis of representations for domain adapatation”, NIPS (2006).

[5]S. Ben-David, J. Blitzer, K. Crammer, and et al., “A theory of learning from different domains”, Machine Learning (2010).

[6]Y. Sun, G. Guo, and Y. Li, “React: out-of-distribution detection with rectified activations”, Advances in Neural Information Processing Systems (2021).

[7]Y. Sun and Y. Li, “Dice: leveraging sparsification for out-of-distribution detection”, European Conference on Computer Vision (2022).

[8]Li, Da, and et al., “Deeper, broader and artier domain generalization”, ICCV (2017).

[9]B. Gong, K. Grauman, and F. Sha, “Connecting the dots with landmarks: discriminatevly learning domain-invariant features for unsupervised domain adaptation”, ICML (2013).

[10]Baktashmotlagh and et al., “Unsupervised domain adaptation by domain invariant projection”, ICCV (2013).

[11]Djurisic and et al., “Extremely simple activation shaping for out-of-distribution detection”, ICLR (2023).

[12]Li and et al., “Measuring the intrinsic dimension of objective landscapes”, ICLR (2018).

[13]Y. Gal and Z. Ghahramani, “Dropout as a bayesian approximation: representing model uncertainty in deep learning”, PMLR (2016).

[14]P. Sangkloy, N. Burnell, C. Ham, and J. Hays, “The sketchy database: learning to retrieve badly drawn bunnies”, TOG (2016).

[15]M. Eitz, J. Hays, and M. Alexa, “How do humans sketch objects?”, TOG (2012).

[16]G. K. Pandey and S. Srivastava, “Resnet-18 comparative analysis of various activation functions for image classification”, ICICT (2023).