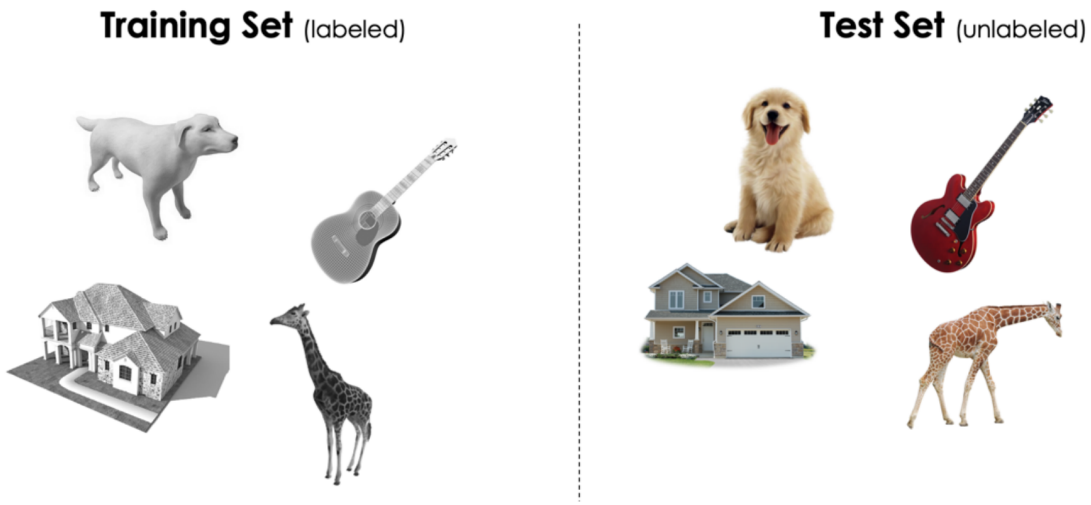
# DOMAIN ADAPTATION VIA ACTIVATION SHAPING

TA: Leonardo Iurada
leonardo.iurada@polito.it

## PROBLEM

One of the biggest limitations of deep models is their inability to work consistently well with any data met at test time. The training and the test set could be characterized by different data distributions (illumination conditions, visual style, background, etc.) leading to an inevitable decrease in the model performances. This is known as the problem of **domain shift** and it's one of the most investigated topics in the computer vision community. It can be formalized with the setting of **Unsupervised Domain Adaptation (UDA)** where there is a **labeled Source** domain (Training Set) and an **unlabeled Target** domain (Test Set), the goal is to train a model that works well on the target distribution.




**Activation Shaping** is the process of modifying activation maps from a layer within an architecture using either some user-defined or learned rules. Recent research **[1]** has shown that this practice is highly effective for detecting out-of-distribution (OOD) data. In this project, we aim to leverage the benefits of this OOD detection strategy to exploit the insights regarding data distribution discrepancies with the goal of learning a model that is robust to domain shift.

## BEFORE STARTING:

1. Read **[1]** to become familiar with the Activation Shaping methodology in Out-of-distribution detection.
2. Read **[2]** to become familiar with the problem of domain shift and with the Unsupervised Domain Adaptation setting.

Starting from the provided GitHub repository **[3]**, carefully follow all the instructions in the *README.md* file to set the **Dataset [4]** and the **Environment**. Then you can start running and editing the code.

## 0. REPRODUCE THE BASELINE

Following the instruction in the *README.md* file, you should be able to reproduce the **Baseline**. The experiment consists in training the unmodified **Feature Extractor** (*ResNet-18*) and the **Object Classifier** (made by one fully connected layer for category prediction) minimizing the Cross-Entropy Loss computed on the **Source Domain** (Art Painting).

Then, you should simply test the model on the **Target Domains** (Cartoon, Sketch, Photo), separately, to reproduce the numbers in the table reported in the *README.md* file.

## 1. ACTIVATION SHAPING MODULE

To improve the performance of the model on the Target Domains you have to implement a **custom activation shaping layer**, either as a `torch.nn.Module` or as a `forward_hook` function (recommended. See **[5]** for more details), that accepts two inputs:

1. The current **activation map/output** ($\mathbf{A}$) produced by the current layer.
2. **Another tensor** ($\mathbf{M}$) with the same shape as the activation map/output of the current layer.

This custom activation shaping layer should then **binarize** both $\mathbf{A}$ and $\mathbf{M}$ (using zero as threshold, i.e. every element less or equal to zero should be zero and every element greater than zero should be set to one), and then **return the element-wise product** of $\mathbf{A}$ and $\mathbf{M}$.

Once the custom activation shaping layer is implemented, it is time to hook it onto our *ResNet-18* Baseline model. It's up to you to **decide after which layers to insert it** in the base architecture (you should try multiple configurations eg. after each convolution, once every 3 convolutions, … and **record how the performance changes in the following experiments**).

## 2. RANDOM ACTIVATION MAPS

The first thing we can try to do is to randomly turn off some outputs of each layer. To accomplish this, the idea is to make so that each $\mathbf{M}$ is a random mask of zeros and ones.

Specifically, you have to train your network with the custom activation shaping layers (passing random 0-1 masks during the forward pass of each layer) on the **Source Domain** (Art Painting) and then, you should test the model on the **Target Domains** (Cartoon, Sketch, Photo), separately.

You should **try different ratios of zeros and ones (eg. starting from $\mathbf{M}$ made of only ones)** in the random masks $\mathbf{M}$ and **record how the performance changes**.

## 3. ADAPTING ACTIVATION MAPS ACROSS DOMAINS

Finally we delve into **Domain Adaptation**. The main idea is that activation patterns encode both target-specific and domain-specific information. Thus, by strategically manipulating such patterns we hope to learn a domain-agnostic representation.

Given two examples, one from the **Source Domain** (Art Painting), $\mathbf{x_s}$ (we have also the corresponding category label $\mathbf{y_s}$), the other from one **Target Domain** ($\mathbf{x_t}$, unlabeled), at each training iteration, the strategy consists in the following steps:

1. Record activation maps $\mathbf{M_t}$ by forwarding $\mathbf{x_t}$ through the network.
2. Apply $\mathbf{M_t}$ using our custom activation shaping layers when forwarding $\mathbf{x_s}$ through the network and producing the network output $\mathbf{z_s}$.
3. Compute and perform the backward pass using Cross-Entropy Loss considering the network output $\mathbf{z_s}$ and the category label $\mathbf{y_s}$.

Once the training is done, you need to test the learned model on the **Target Domain**.

*(NOTE: The **Source Domain** is always Art Painting, while you need to repeat this experiment considering one **Target Domain** at a time i.e. Photo, Cartoon, Sketch.)*

Having implemented and run mandatory points **0.**, **1.**, **2.** and **3.** you now have to choose one between **Extension (1)** and **Extension (2)**.

## Extension (1) – ACTIVATION SHAPING FOR DOMAIN GENERALIZATION

In the last part of the project we want to assess how activation shaping can be applied to the **Domain Generalization** setting. To study how this practice may be beneficial for enhancing the generalization capabilities of our model, we modify the procedure used in the Domain Adaptation setting to include multiple source domains.

Specifically, given three source examples $\mathbf{x_{s1}}$, $\mathbf{x_{s2}}$, $\mathbf{x_{s3}}$ (one from each **Source Domain**) and their respective category labels $\mathbf{y_{s1}} = \mathbf{y_{s2}} = \mathbf{y_{s3}}$. Then, at each training iteration:

1. Modify the custom activation layer so that it can *accept, binarize and multiply together an arbitrary number of activation maps* (in our case we will need 3 activation maps, one per Source Domain).
2. Record activation maps $\mathbf{M_{s1}}$, $\mathbf{M_{s2}}$ and $\mathbf{M_{s3}}$ by forwarding $\mathbf{x_{s1}}$, $\mathbf{x_{s2}}$ and $\mathbf{x_{s3}}$ respectively.

3. Concatenate $\mathbf{x}_{s1}$, $\mathbf{x}_{s2}$ and $\mathbf{x}_{s3}$ in a single minibatch $\mathbf{x}$. Do the same with $\mathbf{y}_{s1}$, $\mathbf{y}_{s2}$ and $\mathbf{y}_{s3}$ producing $\mathbf{y}$.
4. Apply $\mathbf{M}_{s1}$, $\mathbf{M}_{s2}$ and $\mathbf{M}_{s3}$ using our extended custom activation shaping layers when forwarding $\mathbf{x}$ through the network and producing the network outputs $\mathbf{z}$.
5. Compute and perform the backward pass using Cross-Entropy Loss considering the network outputs $\mathbf{z}$ and the category labels $\mathbf{y}$.

Once the training is done, you need to test the learned model on the **Target Domain**.

*(NOTE: You need to repeat this experiment considering one domain as **Target Domain** at a time i.e. Art Painting, Photo, Cartoon, Sketch, while the other three have to be used as **Source Domains**.)*

## Extension (2) – BINARIZATION ABLATION

Implement two variations of the **custom activation shaping layer:**

1. Instead of binarizing $\mathbf{M}$, keep it as it is and simply multiply it with $\mathbf{A}$.
2. Binarize $\mathbf{M}$, compute the Top K values of $\mathbf{A}$ and set all the elements of $\mathbf{M}$ that are not in the Top K to zero. Then, multiply $\mathbf{A}$ and $\mathbf{M}$.
   *(NOTE: K is an hyperparameter you'll have to tune)*

Now reproduce points **2.** and **3.** with the two implemented variations.

## REFERENCES:


[1] A. Djurisic et al. "Extremely Simple Activation Shaping for Out-of-Distribution Detection." ICLR 2023.

[2] Ganin, Yaroslav, et al. "Domain-adversarial training of neural networks." The journal of machine learning research 17.1 (2016): 2096-2030.

[3] GitHub project repository: https://github.com/iurada/Activation-Shaping-AML

[4] **PACS Dataset**: Li, Da, et al. "Deeper, broader and artier domain generalization." ICCV 2017

[5] https://web.stanford.edu/~nanbhas/blog/forward-hooks-pytorch/