RDFIA - M2 IMA

Practical work report 2-c
Domain Adaptation

Done by :

Carlos GRUSS & Oussama RCHAKI

25/09/24

Contents

Objective

Input data for Deep Learning models can belong to different domains. A common example is the segmentation data used for autonomous driving. An image from the game GTA-5 and one taken in the street may represent the same classes (e.g., road, car, pedestrian, etc.) but the distribution of the inputs is quite different in each case.

Moreover, gathering labeled data in the target domain can be very costly, both in time and in money. Therefore the field of domain adaptation aims to train on one labeled source dataset (like GTA-5 whose labels are given by the game engine, and therefore it's cheap to get a great amount of labeled data) and to perform well on another unlabeled target dataset. See Figure 1 for an actual example.

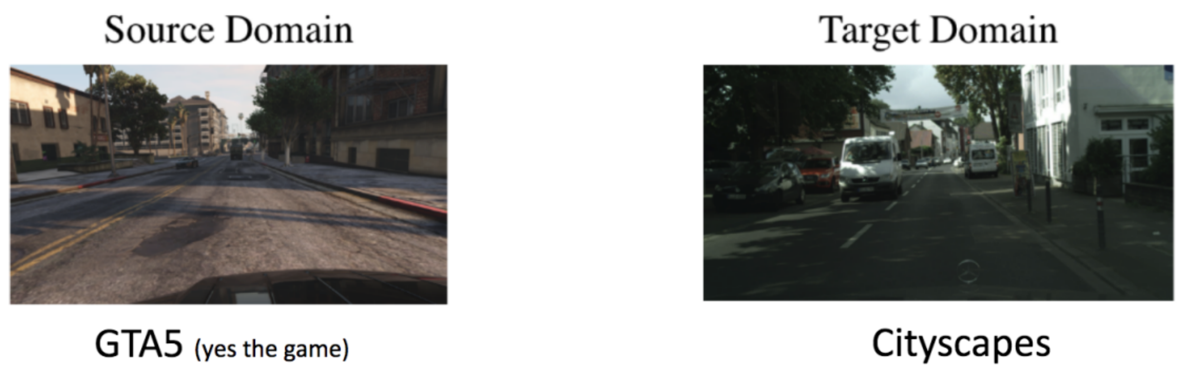


Figure 1 : Example of different source and target domains in the field of image labeling for autonomous driving.

1 – DANN and the GRL layer

In this practical work, a model that was trained on the labeled MNIST dataset and then be evaluated on an unlabeled MNIST-M (see Figure 2).

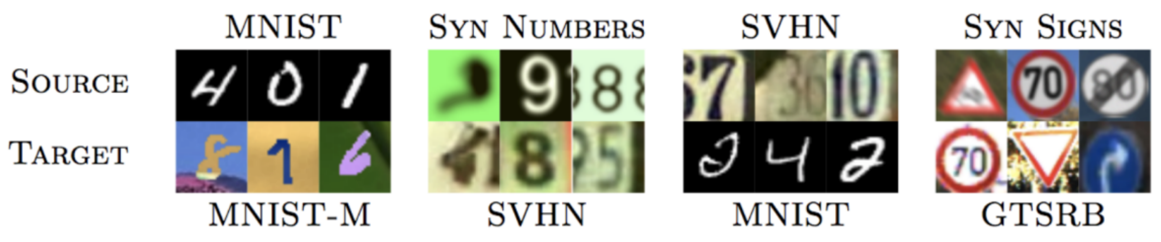
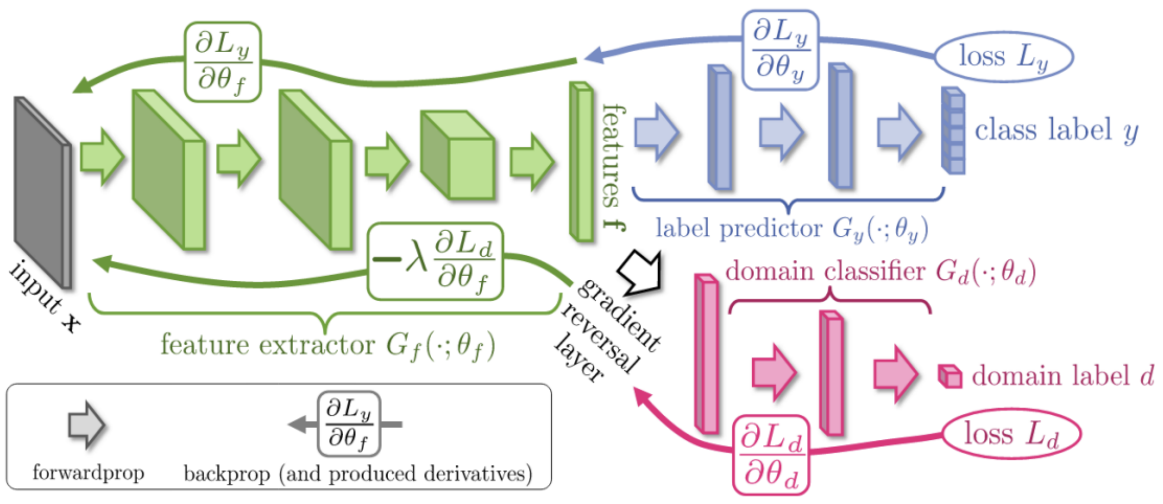


Figure 2 : Examples of domain adaptation in the field of character recognition. The image furthest to the left is the actual case which concerns this practical work (MNIST/MNIST-M)

The model chosen for performing the domain adaptation is the one described by Ganin and Lempitsky in their paper "Unsupervised Domain Adaptation by Backpropagation" (2015). The authors assume that the model works with input samples $x \in X$, where X is some input space and certain labels (output) y from the label space Y. They further assume that there exist two distributions $S(x, y)$ and $T(x, y)$ on $X \otimes Y$, which will be referred to as the source distribution and the target distribution (or the source domain and the target domain). Both distributions are assumed complex and unknown, and furthermore similar but different (in other words, S is "shifted" from T by some domain shift).

The architecture proposed by the authors to face this problem is displayed in Figure 3. The layers in green are the convolutional neural network. Its goal is to learn a good representation for the classifier, whose layers are in blue. However, the goal is for the representation to be domain-agnostic, meaning that in the MNIST/MNIST-M case, a digit '7' in white, or a digit '7' in purple, have the same representation.



$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
class label y
features $\mathbf{f}$
label predictor $G_y(\cdot;\theta_y)$
$-\lambda \frac{\partial L_d}{\partial \theta_f}$
input x
feature extractor $G_f(\cdot;\theta_f)$
gradient reversal layer
domain classifier $G_d(\cdot;\theta_d)$
domain label d
$\frac{\partial L_d}{\partial \theta_d}$
loss L_d
forwardprop
backprop (and produced derivatives)

FIGURE 3 : DANN architecture

To do so, another output is added to the network, represented by the layers in pink. This is the domain classifier branch. Its goal is to tell from the features produced by the ConvNet if the image comes from the source or target domain. Note that this classifier doesn't need the target domain data to be labeled, just knowing if the inputs come from source or target domain.

Between the ConvNet in green, and the domain branch in pink, there is a Gradient Reversal Layer (GRL). This layer reverses (i.e., multiplies by a negative number) the gradient. This means that while the domain branch will try to get good at classifying domain, the ConvNet will try to make it impossible to discriminate both domains, and thus will become domain-agnostic. This idea is related to another popular architecture in Generative AI called "Generative Adversarial Networks" (GANs), described by Goodfellow et. al. (2014).

2 – Practice

Question 1

If we didn't use the GRL layer between the feature extractor and the domain classifier, the feature extractor would actually be encouraged to learn features that are domain specific, which is exactly the opposite of the goal of the DANN architecture.

This is because the loss gradient would propagate backwards from the domain classifier, encouraging the CNN to learn features that help the domain classifier to better discriminate between examples from the source and target distributions.

Question 2

In domain adaptation, the performance on the source dataset may degrade slightly because the model is adjusted to perform well on the target domain, which can introduce a trade-off between the two domains. This adaptation process modifies the shared feature representations to align the source and target distributions, potentially making them less optimal for the source domain. Consequently, the model's ability to perform **specifically** on the source dataset may be reduced due to these changes in representation.

Question 3

The value of the GRL layer parameter (called λ) determines how domain specific or domain agnostic the CNN features are. The lower the value, the less the feature extractor is encouraged to "fool" the domain classifier by learning domain agnostic features.

In the original paper, the authors propose to adjust the value throughout the training process, starting from zero (thus allowing the CNN to learn good feature representations without being influenced by the domain) and gradually increasing to 1 according to the update rule given by:

$$\lambda_p = \frac{2}{1 + \exp(-\gamma \cdot p)} - 1$$

where p is the learning progress (from 0% to 100%) which we can compute as the index of the current batch over the total number of batches on which the model will train, and γ is a tunable parameter. For the original paper, the authors set this parameter to 10, while in this practical work initial experiments where done with a value of 2. The update rule used for λ_p is shown graphically in the plots of Figure 4.

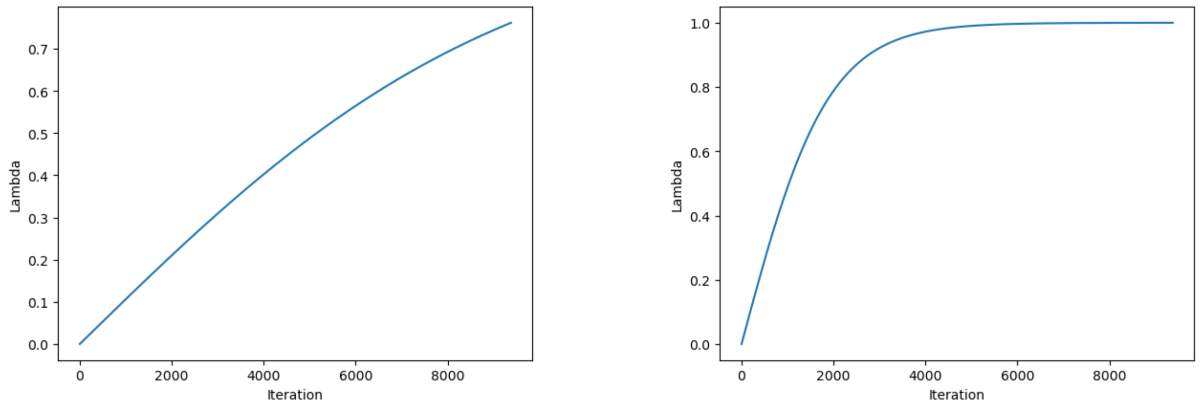


Figure 4 : λ_p with respect to the training progress, plotted for $\gamma = 2$ (left) and $\gamma = 10$ (right).

Question 4

Pseudo-labeling is a semi-supervised learning method commonly used in domain adaptation to leverage unlabeled target domain data. The process begins with a model trained on the labeled source domain data. This trained model is then applied to the unlabeled target domain data to generate predictions, where high-confidence predictions are assigned as pseudo-labels. These pseudo-labels are treated as ground truth for further training, allowing the model to adapt to the target domain iteratively.

The pseudo-labeling process involves several key steps: initial training on the source domain, prediction and pseudo-label generation for the target domain, and re-training the model on both labeled source data and pseudo-labeled target data. Over iterations, this method refines the model's understanding of the target domain by aligning the feature representations of the source and target domains.

However, challenges such as noisy pseudo-labels, choosing an appropriate confidence threshold, and domain discrepancy can affect the performance. Significant works related to pseudo-labeling include:

- **Pseudo-Label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks** by Lee, D.-H. (2013), which introduced the concept of pseudo-labeling.
- **Self-Training with Noisy Student Improves ImageNet Classification** by Xie, Q., et al. (2020), extending pseudo-labeling to achieve state-of-the-art results.
- **Mean Teacher: Semi-Supervised Learning with Consistency Training** by Tarvainen, A., & Valpola, H. (2017), emphasizing regularization in pseudo-labeling.

Results and experiments

In our experiments, it was observed that going from a naive feature extractor and classifier architecture to the DANN architecture, an improvement in classification accuracy over MNIST-M from 51.96% to 79.09% could be observed. Moreover, t-SNE was used to visualize a 2D representation of the distribution of the learned features in each of the cases (Figure 5).

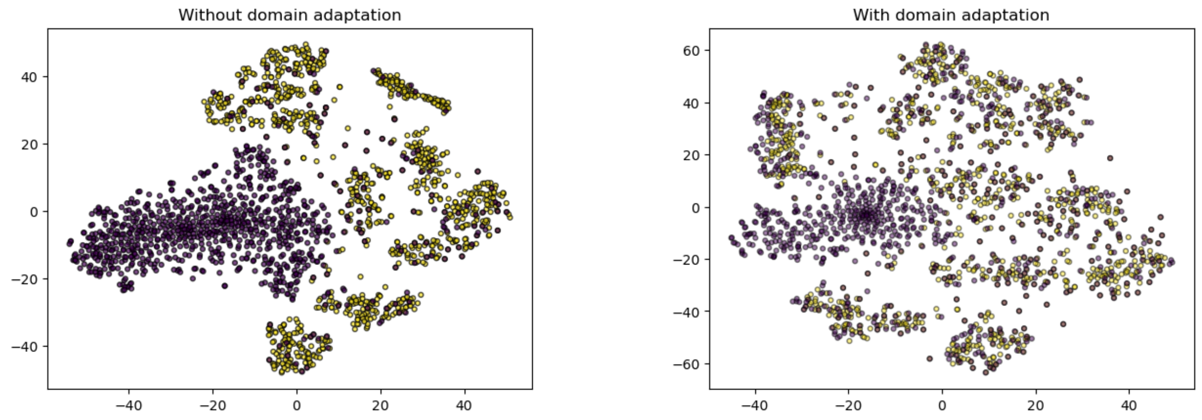


Figure 5 : The plot on the left shows a t-SNE visualization of the learned embeddings for MNIST and MNIST-M in different colors when using a simple naive classification network without domain adaptation. The plot on the right shows these features as extracted by the CNN of the DANN architecture. It can be observed that it's much harder to spatially discern the two distributions after performing domain adaptation, suggesting that DANN corrects the distribution shift between the MNIST and MNIST-M datasets.

However, the authors reported a performance of 81.49% test accuracy in the classification over the target dataset (MNIST-M), and although this improvement over our results is not huge ($\approx 2\%$), we tried to perform some experiments to see if we could get closer to the reported performance.

A first experiment consisted in increasing the value of γ to 10, as reported by the authors. This would correspond to more aggressively increasing the value of λ_p in the GRL, as show in Figure 4. We also increased the number of epochs to a 100 for these experiments to have a better idea of how the training progress was behaving. The results for this experiment are quite surprising and can be observed in Figure 6.

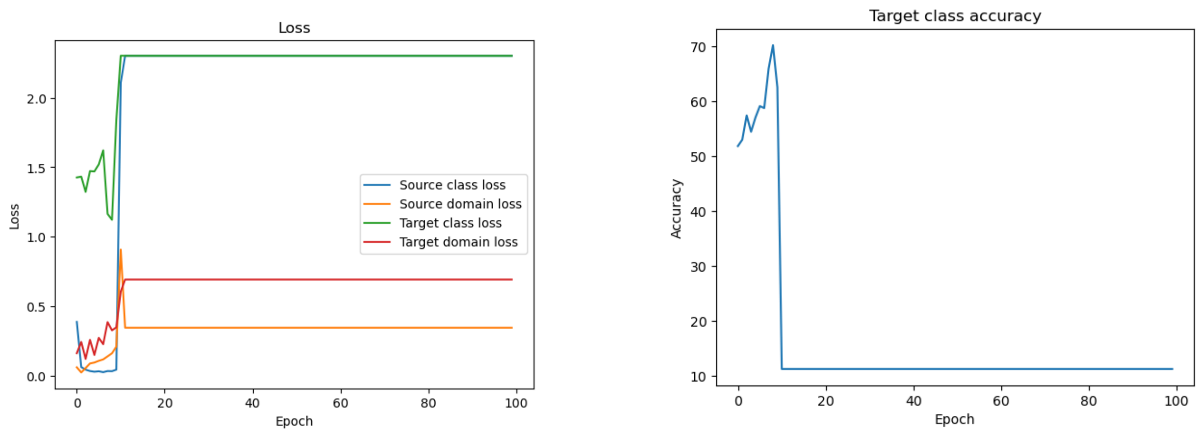


Figure 6 : Loss and accuracy evolution with $\gamma = 10$.

As we can see, the model seems to fall into a local minimum quite early in the training process. This minimum corresponds to 100% accuracy of the domain discriminator, but very low accuracy by the class discriminator, which is what we're actually interested in maximizing. These results suggest that increasing the value of γ could potentially provide very good results very quickly (we see a dramatic rate of increase in classifier accuracy in the first few epochs) but risks being very unstable. Further experiments would be needed to fully understand this behavior, but it seems clear that in this experiment the model is learning to prioritize minimizing the domain loss over the class loss, something that is also shown by the t-SNE visualization of the obtained embeddings from this model (Figure 7).

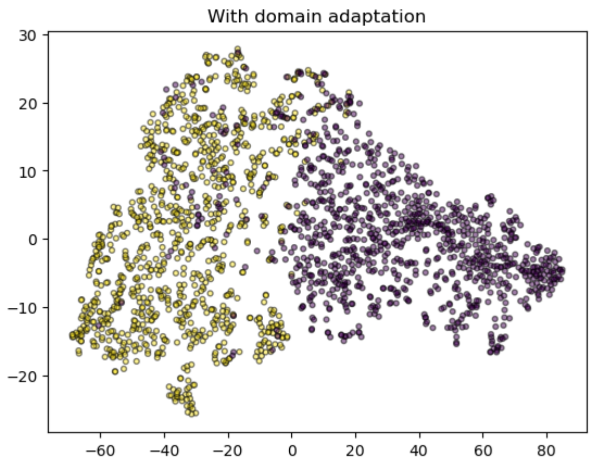


FIGURE 7 : t-SNE visualization of embeddings after 100 epochs with $\gamma = 10$. It can be observed that the model seems to specialize in discriminating between domains rather than being domain agnostic.

One final experiment consisted in leaving $\gamma = 2$ and seeing what would happen if we ran the training loop for 100 epochs. The results are presented in Figure 8. We can observe that the training process is very unstable and noisy, but that we can potentially achieve the same performance as the authors if we kept the model parameters which achieve the best performance throughout the training process, which in our particular case seems to be $\approx 80\%$ around epoch 80.

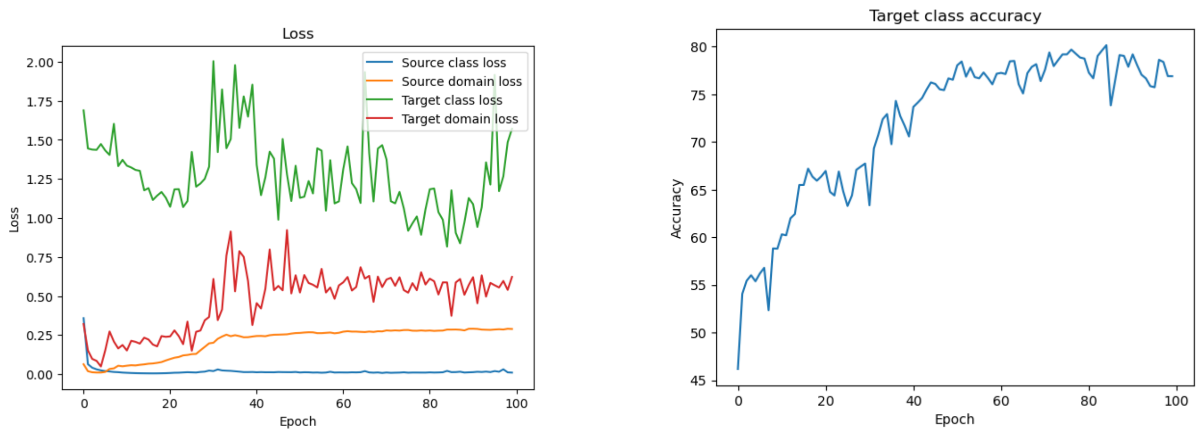


FIGURE 8 : Loss and accuracy evolution with $\gamma = 2$.