Applied Machine Learning 2021 project report

# Adversarial Discriminative Domain Adaptation

Andrea Debeni and Stefano Pardini

University of Trento

January 14, 2022

## 1 Introduction

The aim of this project is to attempt to reproduce the results obtained by Tzeng et al. [1], where Adversarial Learning into Domain Adaptation (ADDA) is presented.
When dealing with domain adaptation tasks, a major issue is represented by domain shift: whenever data distribution changes across domains, applying classifiers trained on a domain directly to another one can result in very poor performances [2]. Several solutions have been proposed to address this drawback, among which stand adversarial learning (and, in particular Generative Adversarial Network). GAN consists basically in an adversarial process where a generative model and a discriminator model are trained simultaneously: the first aims to capture the source data distribution and generate data accordingly, while the latter have to guess whether the input is true or it has been created by the generator [3]. ADDA is a novel implementation of a Generative Adversarial Network (GAN) which combine discriminative modeling, untied weight sharing and a typical GAN loss.
ADDA consists in three different phases: in the pre-training phase, source images and labels are input of a source encoder CNN; in the adversarial adaptation phase, a target encoder CNN is trained to fool a discrimator which should be unable to distinguish between encoded source and target examples domain labels; then, in the testing phase, target images are mapped with the previously trained target encoder and classified by the source classifier. In this report we present the results obtained by our implementation of ADDA (the code is freely available on GitHub[1]). A possible reference with an *official* implementation made by the author himself can be found on GitHub as well[2].

## 2 Implementation

In addition to the official implementation of Tzeng, before the actual implementation we reviewed other solutions found on GitHub. The following repositories include codes which feature different frameworks: Tensorflow 1.x implementations:

- https://github.com/truongthanhdat/ADDA
- https://github.com/redhat12345/ADDA-1
- https://github.com/byeongjokim/ADDA

PyTorch implementation:

- https://github.com/corenel/pytorch-adda

For our implementation, we used the last available version of TensorFlow (2.7), basing on the original one made by Tzeng. During phase 1 we pre-trained a source encoder CNN using labeled source images: this was achieved by using the LeNet architecture provided in the Caffe source code[3]. On the other hand, the discriminator consists of a fully 3-connected layers: two layers with 500 hidden units (which use a ReLU activation function) followed by the final discriminator output.

## 3 Results

We have empirically observed that the accuracy calculated during phase 3 (on the target dataset) is, on average, better if the accuracy of the classifier trained during phase 1 (on the source dataset) is capped. This was achieved in two ways: by artificially reducing the number of datapoints used during training (random sampling), or by reducing the number of epochs. We assume that the phenomenon occurs because, during phase 2 (adversarial adaptation), the target encoder weights are not initialized randomly, but

[1]https://github.com/Debo790/AML2021
[2]https://github.com/erictzeng/adda
[3]http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

| Method | MNIST → USPS | USPS → MNIST | SVHN → MNIST |
|---|---|---|---|
| Source only | 7.52 | 49.80 | 53.00 |
| ADDA | 13.80 | 50.70 | 53.00 |

Table 1: Upper bound experimental results on unsupervised adaptation.

are borrowed from the source encoder, trained during phase 1. We particularly operated on the MNIST dataset, whose classifier is able to reach a 96.4% test accuracy after one epoch only.

Therefore, a very *strong* source model, whose weights have been tuned to obtain high classification performance, ends up in a target encoder that is too *specialized*. This results in penalizing the target encoder during its training phase (adaptation phase), resulting poorly effective during the subsequent testing phase. On the other hand, when using a *clean* target encoder (i.e. without borrowing weights from the source one) we always recorded worse results.

In order to keep track of the adaptation phase progress, we have integrated into it the calculation of the accuracy classification measurement which, for each batch and therefore for each epoch, is recorded. Clearly, this measure must be considered a sort of training error, however it allowed us to monitor the progress of the adversarial adaptation, in order to consequently perform a better tuning of the hyperparameters.

From this perspective, during phase 2 we recorded more encouraging results using different learning rates, $1 \times 10^{-6}$ for the discriminator and $1 \times 10^{-4}$ for the target encoder. The fundamental goal of phase 2 consists in the convergence of the target encoder on the source encoder. For this reason, we observed that a greater movement speed towards the target encoder loss function minimum really helps the target encoder in the adaptation process. Furthermore, the lower the discriminator learning rate is, the lower is its adversarial action as a *counterweight*.

In *Table 2* we can observe the upper bound experimental results (outcome of several runs), gathered during our unsupervised adaptation among MNIST, USPS, and SVHN. We certainly cannot define our results as satisfactory if compared with those reported in Tzeng et al.'s paper [1]. Nevertheless, in two out of 3 cases our ADDA implementation is able to improve the accuracy obtained in the *source only* task, which is calculated by applying the target test set on the source model. We considered as the best model the one that during the adaptation phase obtained better results with respect to the discriminator (the lowest discriminator accuracy). That model has been used in the subsequent test phase.

The graphs show a behavior that we observed in the majority of the tests carried out. It is interesting to analyze the dynamics between the loss function of the discriminator (Figure A.2) and the target (Figure A.1). Following an initial complementary wave motion, the target loss function shows a substantial decrease, against a proportional/clear increase in the discriminator one. Nevertheless, the accuracy obtained using the target model on the source classifier do not confirm the same trend (Figure A.3). More in-depth investigations has been carried out around this issue. Firstly, we considered to pay more attention in the hyperparameters tuning. Secondly, we thought about possible *structural* errors in the network architecture or in the code related to the learning process. With respect to the first question, we carried out numerous tests by borrowing the parameters from the various implementations found on GitHub, including the official one by Tzeng et al. (where all the tests return good accuracy results). With respect to the second one, we have repeatedly revised the architecture of the model, the component that deals with the learning process; special attention was also given in the data ingestion phase.

## 4 Conclusions

Overall, results obtained are still far from those presented by Tzeng et al. in [1]. Even if loss functions show promising evolution during runs, they are not followed by equally positive target classifier performances. The relations between discriminator and target losses suggest that the adversarial action is actually taking place. However, further investigation on the model should be done in order to enhance performances in the three adaptation contexts considered.

## References

[1] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation.

[2] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation.

[3] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks.

## A Appendix - Images

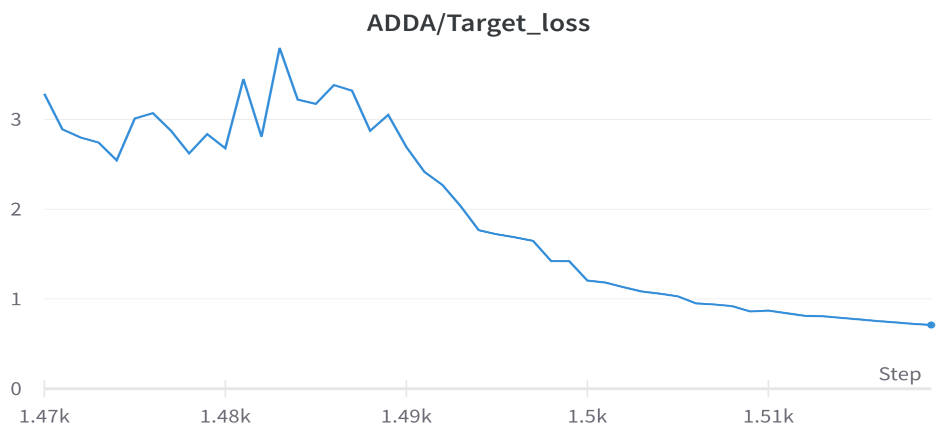

Figure 1: Target loss

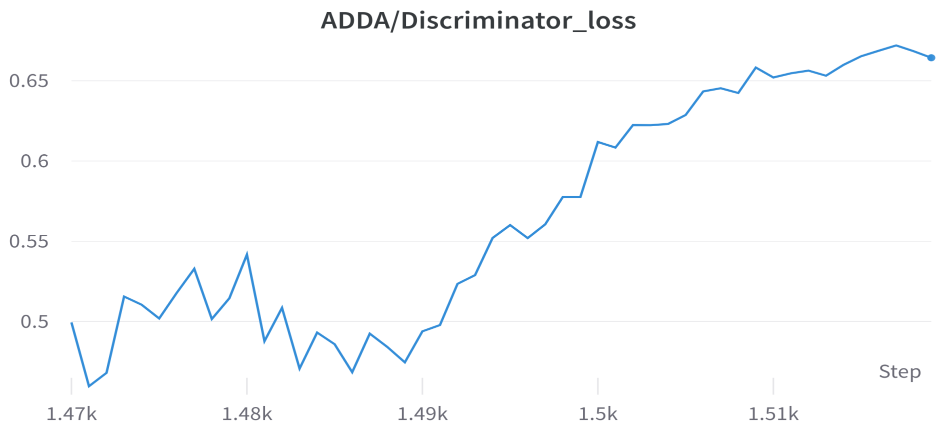

Figure 2: Discriminator loss

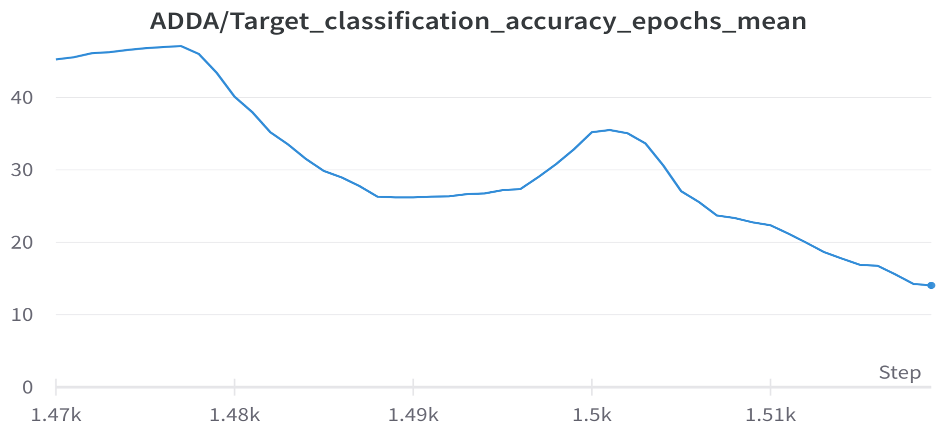

Figure 3: Target classification accuracy mean

## B Appendix - Contribution

| **Member** | **Contribution** |
|---|---|
| Stefano | Implementation of models and architecture |
| Andrea | Testing and refactoring |

Table 2: Contribution made by each group member

Code changes, refinements and run attempts with different hyperparameters have always been discussed before implementation through multiple Zoom meetings.