# Adversarial Feature Augmentation for Unsupervised Domain Adaptation

# Supplementary Material

Riccardo Volpi[1], Pietro Morerio[1], Silvio Savarese[2], Vittorio Murino[1,3]
{riccardo.volpi,pietro.morerio,vittorio.murino}@iit.it, ssilvio@stanford.edu

[1]Pattern Analysis & Computer Vision - Istituto Italiano di Tecnologia
[2]Stanford Vision and Learning Lab - Stanford University
[3]Computer Science Department - Università di Verona

## 1. Architectures

We provide in this section a detailed description of the networks used for our experiments. For the digit datasets, the encoders follow the standard architectures commonly used in unsupervised domain adaptation [2].

Figure 1, *left*: architectures of $E_S$ and $E_I$ used for MNIST $\leftrightarrow$ USPS and SVHN $\rightarrow$ MNIST.

Figure 1, *right*: architectures of $E_S$ and $E_I$ used for SYN $\rightarrow$ SVHN.

Figure 2, *left*: architecture of $S$ used for all the experiments.

Figure 2, *right*: architecture of $D_1$ used for all the experiments.

Figure 3, *left*: architecture of $D_2$ used for SVHN $\rightarrow$ MNIST and SYN $\rightarrow$ SVHN.

Figure 3, *right*: architecture of $D_2$ used for MNIST $\leftrightarrow$ USPS and NYUD (RGB $\rightarrow$ D).

Concerning $E_S$ and $E_I$ used in the NYUD experiment, we relied on a pretrained VGG-16 [4], following the protocol used by Tzeng et al. [6]. We cut it at *fc7*, which was shrieked to be 128-dim and modified with tanh activations. The classifier $C$ consists in an additional 19-dimensional softmax layer.

Summarizing, we found out that $D_2$ should be built with two or three hidden layers to stabilize the minimax game against $E_I$ (whose structure must be the same as $E_S$). We designed an $S$ that proved to be reliable in all experiments; to play a balanced minimax game, we found out that a one-hidden-layer neural network as a discriminator ($D_1$) is an optimal choice. The size of the hidden layer depends on the problem, and can be determined by observing the stability of the training procedure.

## 2. Hyperparameters

We report in this section the hyperparameters used in the different Steps of the training procedures. Note that hyperparameters were set in order to reach the convergence of the GAN [3] minimax games, no cross-validation using target labels was performed.

### 2.1. Digits

For each training Step, we used a batch size of 64 samples. The learning rate was set to $3 \cdot 10^{-4}$ for Step 0, $1 \cdot 10^{-4}$ for Step 1 and $3 \cdot 10^{-5}$ for Step 2, in all experiments except MNIST $\leftrightarrow$ USPS, where was set to $3 \cdot 10^{-6}$.

## 2.2. NYUD

In Step 0, the network is not trained from scratches: following the protocol described in [6], we fully fine-tune a VGG-16 network [4] (pre-trained on ImageNet [1]) for 20.000 iterations, in order to have a comparable baseline model. Batch size is 32 (instead of 128) due to hardware limitations. The learning rate were $10^{-4}$ for Step 0, $10^{-5}$ for Step 1 and $10^{-7}$ for Step 2.

# 3. Ablation study

In the ablation study presented in the paper, we evaluate *DI LS-ADDA*, short for Domain Invariant Least Squares ADDA, *i.e*. our method without performing feature augmentation through $S$. Figure 4 depicts the two Steps of such algorithm. The architectures of the modules and hyperparameters are the same as in the full pipeline.

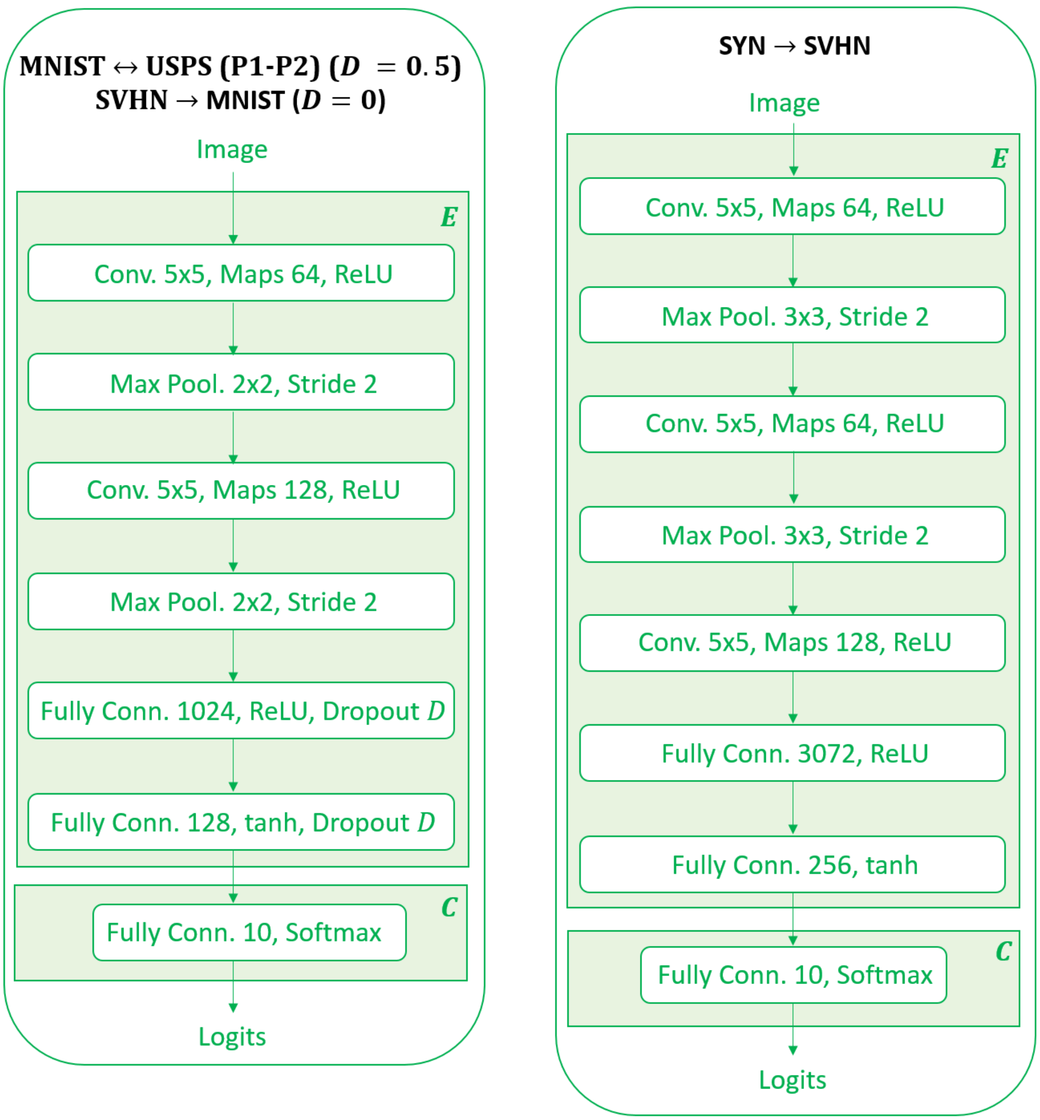


Figure 1. Architectures used for $C \circ E_S$ and $C \circ E_I$ ($C \circ E$ for simplicity) in the MNIST ↔ USPS (P1-P2) and in the SVHN → MNIST (*left*) experiments, with the different values of Dropout [5] indicated ($D$), and in the SYN → SVHN experiment (*right*). The classification module ($C$) is a simple fully-connected + softmax layer.

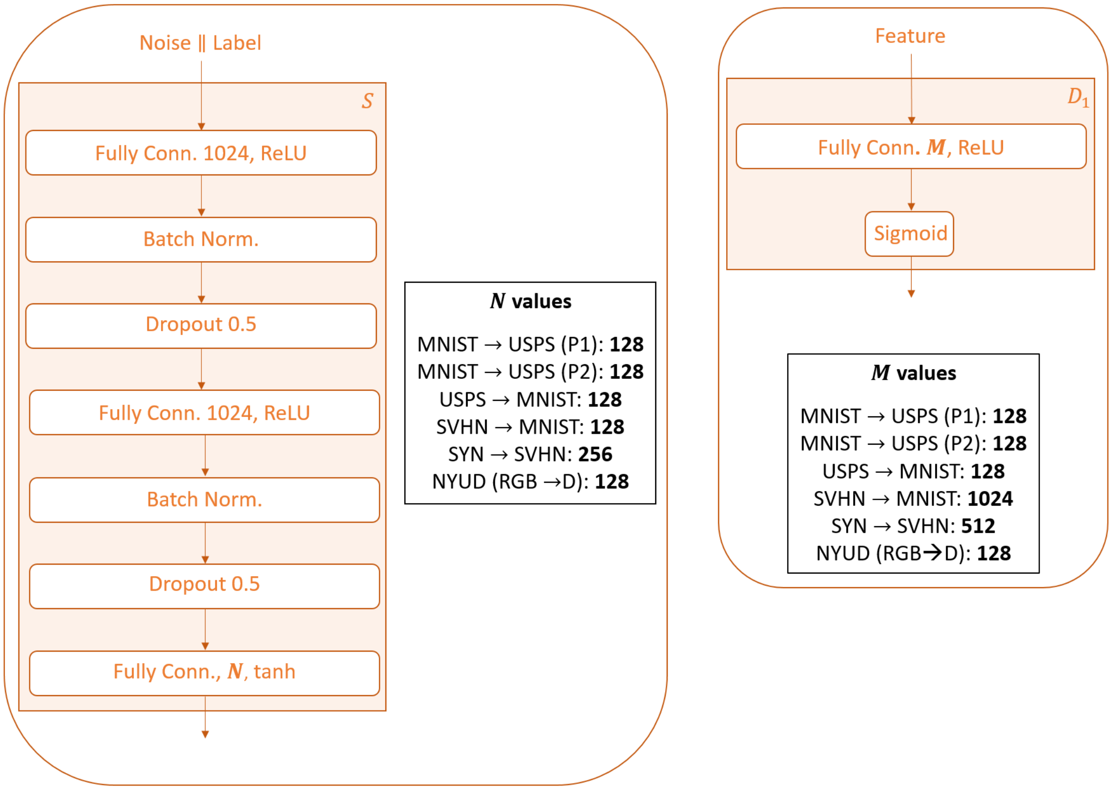


Figure 2. Architectures used for $S$ (*left*) and for $D_1$ (*right*), with the size of the features generated and of the hidden layer indicated, respectively.

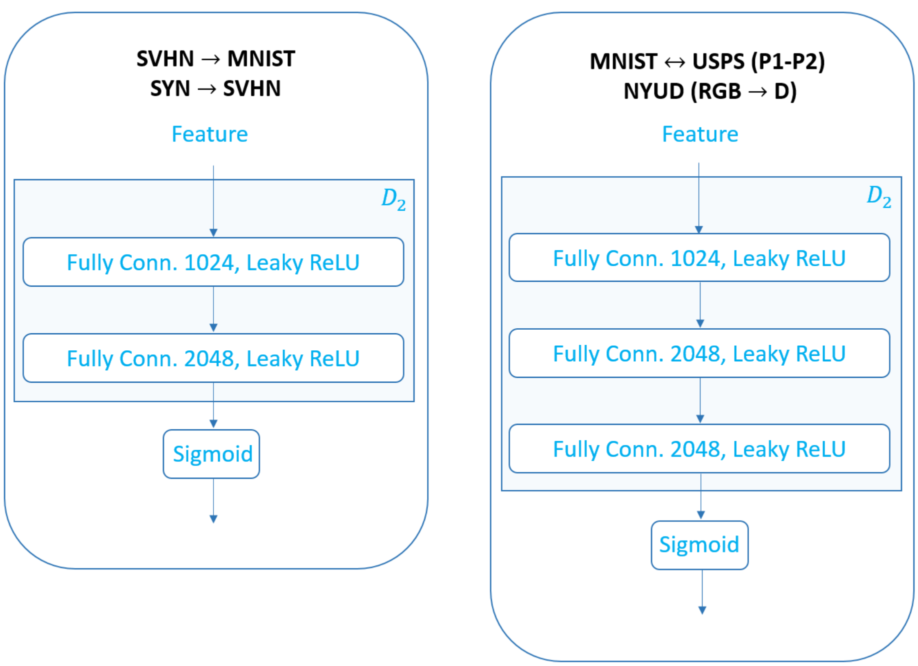


Figure 3. Architectures used for $D_2$ in SVHN $\rightarrow$ MNIST and SYN $\rightarrow$ SVHN (*left*), and in NYUD and MNIST $\leftrightarrow$ USPS (*right*).

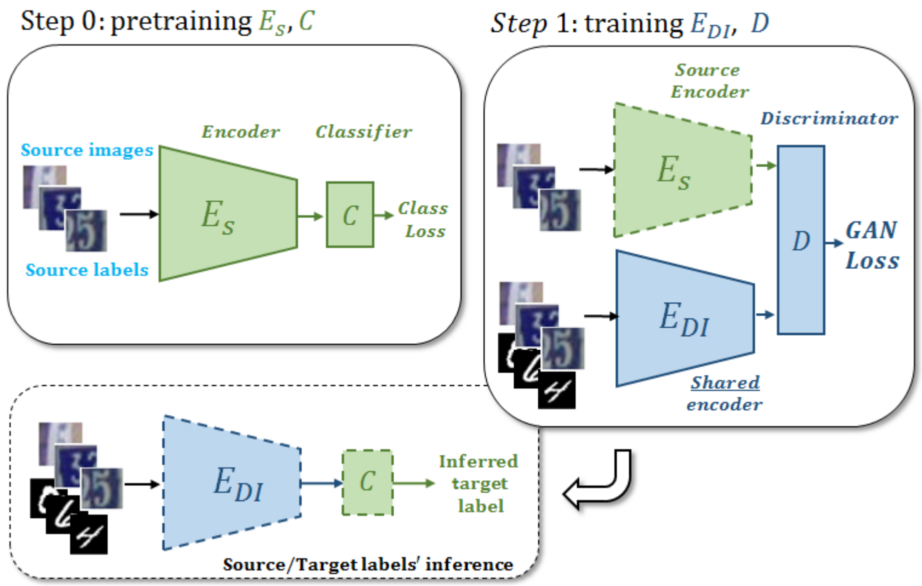


Figure 4. *DI LS-ADDA*. Domain invariance is enforced by feeding both target and source data to $E_{DI}$. The feature augmentation module $S$ is removed from the full pipeline. $E_{DI}$ has the same architecture as $E_I$ and $D$ of $D_2$.

## References


[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 2

[2] Y. Ganin and V. S. Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1180–1189, 2015. 1

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. 1

[4] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 1, 2

[5] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1), Jan. 2014. 2

[6] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1, 2