



ADAPT: Awesome Domain Adaptation Python Toolbox

Antoine de Mathelin, Mounir Atiq, Guillaume Richard, Alejandro de la Concha, Mouad Yachouti, François Deheeger, Mathilde Mougeot, Nicolas Vayatis

► To cite this version:

Antoine de Mathelin, Mounir Atiq, Guillaume Richard, Alejandro de la Concha, Mouad Yachouti, et al.. ADAPT: Awesome Domain Adaptation Python Toolbox. 2024. hal-04455951

HAL Id: hal-04455951

<https://hal.science/hal-04455951>

Preprint submitted on 13 Feb 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ADAPT : Awesome Domain Adaptation Python Toolbox

Antoine de Mathelin^{1,2}

ANTOINE.DE-MATHELIN-DE-PAPIGNY@MICHELIN.COM

Mounir Atiq²

MOUNIR.ATIQ@ENS-PARIS-SACLAY.FR

Guillaume Richard²

GUILLAUME.RICHARD@ENS-PARIS-SACLAY.FR

Alejandro de la Concha²

ALEJANDRO.DE_LA_CONCHA_DUARTE@ENS-PARIS-SACLAY.FR

Mouad Yachouti²

MOUAD.YACHOUTI@ENS-PARIS-SACLAY.FR

François Deheeger¹

FRANCOIS.DEHEEGER@MICHELIN.COM

Mathilde Mougeot²

MATHILDE.MOUGEOT@ENS-PARIS-SACLAY.FR

Nicolas Vayatis²

NICOLAS.VAYATIS@ENS-PARIS-SACLAY.FR

¹Manufacture Française des pneumatiques Michelin, Clermont-Ferrand, 63000, France

²Centre Borelli, Université Paris-Saclay, CNRS, ENS Paris-Saclay, Gif-sur-Yvette, 91190, France

Abstract

In this paper, we introduce the ADAPT library, an open source Python API providing the implementation of the main transfer learning and domain adaptation methods. The library is designed with a user friendly approach to facilitate the access to domain adaptation for a wide public. ADAPT is compatible with scikit-learn and TensorFlow and a full documentation is proposed online <https://adapt-python.github.io/adapt/> with a substantial gallery of examples.

Keywords: Domain Adaptation, Transfer learning, Deep networks, Importance weighting, Fine tuning, Machine learning, Python

1. Introduction and Motivation

Transfer learning and domain adaptation (DA) aim to correct the shifts that exist between the training distribution of a machine learning model (referred as source) and the target distribution on which the model is deployed. This research field has known an important development over the past decades. The presence of "domain shifts" between source and target distributions is the typical framework encountered in real-life applications which makes domain adaptation algorithms particularly useful for numerous use-cases. Domain adaptation techniques are often needed in scenarios where labels are easily available on a source domain but are expensive on the target domain. For example, it may be used to leverage information from a synthetic dataset to build a classifier for real data (Ganin et al., 2016) such as the adaptation of GTA images for autonomous car segmentation (Saito et al., 2018). Furthermore, domain adaptation is of great interest to correct bias from the training samples such as sample bias (Huang et al., 2007) when one population in the training set is over-represented with respect to the overall population. DA methods are also used to correct shifts in the input features, caused by sensor or technological drifts (Courty et al., 2016). Finally domain adaptation is useful for specifying a pre-trained model on a sub-task with few labels, as segmentation of medical images (Ravishankar et al., 2016).

Nowadays, domain adaptation has become an essential tool to handle domain shifts in real applications. Many DA methods have been developed in recent years and a large number of DA implementations are spread on the web. Paradoxically, the large number of DA variants makes their accessibility to users more difficult. Indeed, finding which methods will best fit a given domain adaptation problem is a difficult task. In practice, the user would like to try different methods and select the most promising one. However, most DA algorithms available on the web are not implemented under the same basis, some of them rely on PyTorch (Paszke et al., 2019), others use scikit-learn (Pedregosa et al., 2011) or TensorFlow (Abadi et al., 2015). Moreover, most open source implementations have for primary objective to reproduce the experimental results of a particular publication and an extra effort is then required to apply them on other data. Facing these difficulties, we propose ADAPT¹, a new open-source Python library compatible with scikit-learn and TensorFlow. This library aims at facilitating the access to transfer learning and domain adaptation methods to a wide public including industrial practitioners. Inspired by the scikit-learn library, which manages to make machine learning accessible to everyone, ADAPT offers DA methods that can be easily used in the same format as each object implements the *fit*, *predict* and *score* functions as any scikit-learn estimator. In a model deployment perspective, the ADAPT objects are compatible with the convenient features of scikit-learn as cloning and gridsearch. DA specific metrics are provided to allow an unsupervised selection of hyper-parameters. Finally, as we consider that the accessibility of a DA method essentially relies on the user understanding, a very detailed documentation is available online with many real and synthetic examples. A user guide is provided to allow newcomers to find the right DA method for their problem based on practical considerations (see the ADAPT flowchart²). ADAPT offers the possibility of developing its own transfer method easily by subclassing existing classes. The library works on Linux, Mac and Windows for the four latest Python versions 3.6 to 3.9. ADAPT is nowadays appreciated by various users both from the academic and industrial world, with more than 1k downloads by month.

2. Existing transfer libraries

The emergence of domain adaptation algorithms first started around 2006. Progressively, many algorithms have been developed and open source implementations have been released. At some point, it became necessary to group several algorithms together to allow their comparison by the machine learning community. Some compilations of algorithms have then been developed as *libTLDA* (Wouter, 2015) or *DA-Toolbox* (Yan, 2016). These libraries implement "classic" DA methods, i.e which do not use deep learning, as KMM (Huang et al., 2007) or TCA (Pan et al., 2010). These first attempts to group DA methods offered the opportunity to quickly test several methods on a same basis. However the proposed libraries were lacking of documentation and modularity. After that, many deep DA methods have been developed and some libraries have then proposed to group several variants under the same repository such as *salad* (Schneider et al., 2018). Since then, many repositories for deep learning have been released, the most notable one being *TLLib* (Junguang Jiang, 2020) which makes the great work of regrouping more than 40 deep DA algorithms. It should be

1. <https://github.com/adapt-python/adapt>

2. <https://adapt-python.github.io/adapt/map.html>

Library	Reference	Eco	PyPI	DDA	CDA	Doc	Test	Sk-Style
Adapt	de Mathelin et al. (2021)	S+T	✓	12	22	✓	✓	✓
TL-Toolkit	Zhuang et al. (2019)	S+T		5	14			✓
libTLDA	Wouter (2015)	S+M	✓	0	13	✓	✓	✓
TransferTools	Vercruyssen (2020)	S	✓	0	5			✓
DDAN	Davidson (2018)	T		4	0			✓
TLlib	Junguang Jiang (2020)	P	✓	40	0	✓		
DomainBed	Gulrajani et al. (2020)	P		26	0		✓	
PyTorch-Adapt	Musgrave (2021)	P	✓	23	0	✓	✓	
Dassl	Zhou et al. (2021)	P		17	0			
salad	Schneider et al. (2018)	P	✓	11	0	✓	✓	
Deep-TL	Zhu et al. (2019)	P		9	0			
Pytorch-Ada	Tousch et al. (2020)	P	✓	5	0	✓	✓	
DA-Toolbox	Yan (2016)	M		0	10			

Table 1: DA repositories comparison. "Eco" refers to the code ecosystem, i.e. scikit-learn (S), TensorFlow (T), Pytorch (P) and Matlab (M). "DDA" and "CDA" give respectively the number of deep learning and classic DA algorithms in the library. Library implemented under the "Sklearn Style" (Sk-Style) are characterized by the presence of objects which implement "fit" and "predict" methods.

noted that all these repositories propose PyTorch implementations and are designed in a benchmark purpose, i.e. they mainly focus on comparing the results of each variant on well known datasets. Using these repositories on the user dataset often requires an extra effort.

In 2021, ADAPT has been released, with the purpose to open the DA algorithms to newcomers and, in particular, industrial players. For this purpose, we group the main "classic" and "deep" DA methods into a same library and provide a very detailed documentation and robust code, tested through unit tests. The library is available on PyPI, offering more than 30 algorithms implemented in a user friendly scikit-learn style (cf Table 1).

3. Organization

The ADAPT library is divided into three modules : *feature-based*, *instance-based* and *parameter-based* corresponding to the three main DA strategies. A list of all implemented methods is presented in Table 2 (Appendix A). As mentioned in the DA survey (Weiss et al., 2016), some DA methods are based on the use of a small number of labeled target data and are referred as supervised domain adaptation methods (SDA), others use only unlabeled target data along with the sources (UDA). Some methods perform the adaptation and the learning of the task in one stage, others in two.

3.1 Feature-Based Methods

The purpose of feature-based DA methods (Figure 1) is to learn a new representation of the input features in which both source and target distribution match. This DA strategy is mostly used for unsupervised DA. Feature-based methods generally consider the assumption that the domain shift is due to an unknown transformation of the input space caused for instance by sensor drifts or any changes in the acquisition conditions (Courty et al., 2016; Ganin et al., 2016).

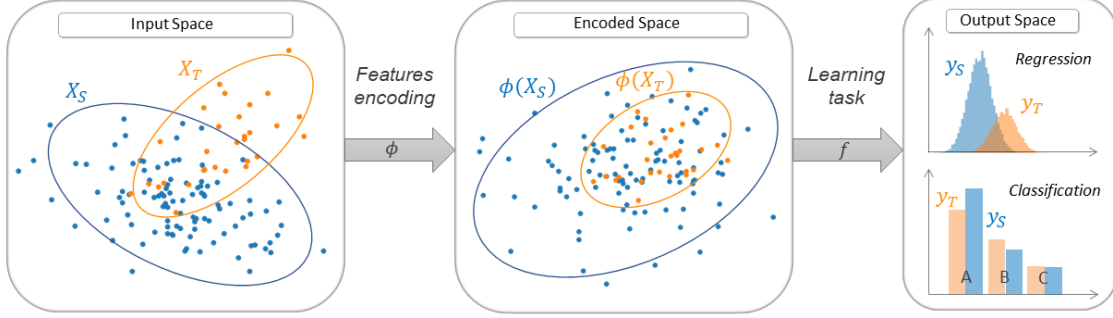


Figure 1: Feature-based strategy illustration.

3.2 Instance-Based

The goal of instance-based methods (Figure 2) is to perform a reweighting of source instances in order to correct the difference between source and target distributions. This kind of methods are mostly used in *sample bias* scenario and assume that source and target distribution share the same support in the input space (Huang et al., 2007; Sugiyama et al., 2007).

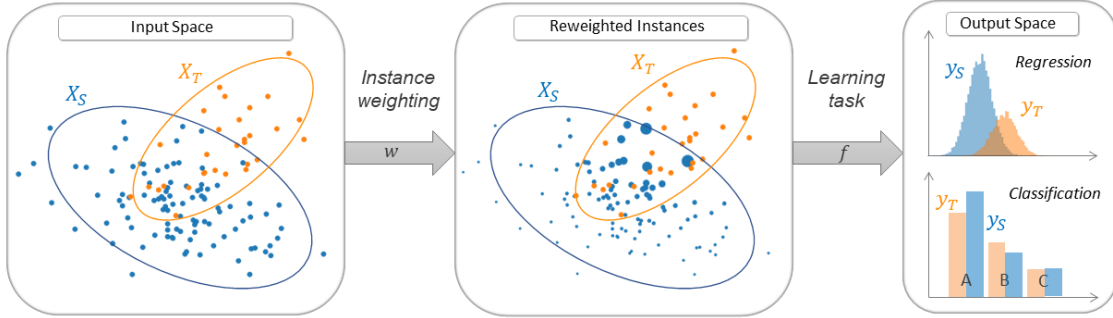


Figure 2: Instance-based strategy illustration

3.3 Parameter-Based

Parameter-based methods (Figure 3), also called "source-free DA", aim to adapt the parameters of a pre-trained source model to a few target observations. These DA methods are mostly used in computer vision where deep model trained on huge data sets are fine-tuned on a smaller data set of images for a specific task (Oquab et al., 2014).

4. Installation and Usage

ADAPT provides several widely used domain adaptation methods using different approaches. The provided methods allow to cope with the main DA settings encountered in real applications as **Supervised DA** and **Unsupervised DA** which respectively refer to the cases

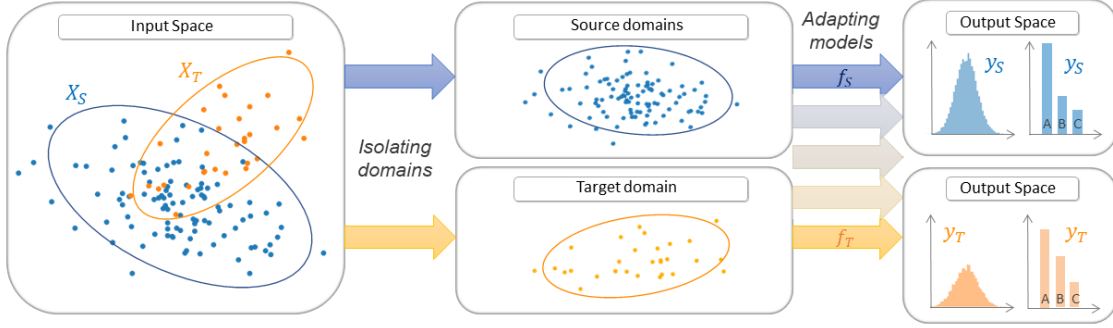


Figure 3: Parameter-based strategy illustration

where target labels are available or not (Motiian et al., 2017) (see examples in Figures 4.a, 4.b), as well as **Source-free DA** (Liang et al., 2020) which is encountered when a source pre-trained model is available instead of source data (see Figure 4.c).

Figure 4: Examples of ADAPT usage in three different settings. X_s, y_s are referring to the source data and X_t, y_t to the target data.

```
from sklearn.linear_model import Ridge
from adapt.feature_based import CORAL

estimator = Ridge(alpha=0.1) # Instantiate the estimator
coral = CORAL(estimator, Xt=Xt, lambda_=1e-5) # Instantiate Adapt model
coral.fit(Xs, ys) # Fit the estimator with CORAL adaptation
yt_pred = coral.predict(Xt) # Predict on target data
```

(a) Applying CORAL (Sun et al., 2016) under Unsupervised DA.

```
from sklearn.tree import DecisionTreeClassifier
from adapt.instance_based import TrAdaBoost

estimator = DecisionTreeClassifier(max_depth=5) # Instantiate the estimator
tradaboost = TrAdaBoost(estimator, Xt=Xt, yt=yt, n_estimators=10) # Instantiate Adapt model
tradaboost.fit(Xs, ys) # Fit TrAdaBoost
yt_pred = tradaboost.predict(Xt) # Predict on target data
```

(b) Applying TrAdaBoost (Dai et al., 2007) under Supervised DA.

```
from tensorflow.keras.applications.resnet50 import ResNet50
from adapt.parameter_based import FineTuning

pretrained_model = ResNet50() # Load pretrained model
finetuned_model = FineTuning(pretrained_model, # Instantiate Adapt model
                             training=[True, False], # Specify the Layers to finetune
                             optimizer="sgd", # Specify loss and optimizer
                             loss="categorical_crossentropy")
finetuned_model.fit(Xt, yt, epochs=10, batch_size=32) # Fit on target data
yt_pred = finetuned_model.predict(Xt) # Predict on target data
```

(c) Applying FineTuning (Oquab et al., 2014) under Source-free DA.

5. ADAPT Guidelines

The API is written in pure Python using scikit-learn, SciPy, NumPy, TensorFlow and cvxopt. The principal features of the API are given below:

- **Documentation:** Each method is documented following the standards of scikit-learn. Algorithms explanations are provided along with a full description of the parameters. For each proposed method, illustrative examples are given on both synthetic datasets and real DA problems to offer visual understanding of the methods and empirical comparisons on known DA issues.
- **Code Quality :** checkers are used in all implemented objects to ensure that arguments defined by the user are valid and throw comprehensive warnings and exceptions to help the user. The code is tested with an high coverage and illustrative examples visually show that the methods are behaving as expected.
- **Developer:** ADAPT is released under a BSD2 License on GitHub. Anyone can contribute to the project by reporting issues and/or making pull requests. A Developer Guide is given to help DA researcher to include their works. Continuous integration is implemented to check the code compliance with unit tests.

6. Conclusion and Future work

Since its release, ADAPT has already been used for several research and industrial problems as fall detection (Minvielle et al., 2019), tire design (Mathelin et al., 2021) and even for cosmology applications (Gilda et al., 2021). Future work will focus on adding more diversified algorithms to handle multisource and semi-supervised domain adaptation.

References

- Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL <https://www.tensorflow.org/>. Software available from tensorflow.org.
- Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning for differing training and test distributions. In *Proceedings of the 24th international conference on Machine learning*, pages 81–88, 2007.
- Ciprian Chelba and Alex Acero. Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech & Language*, 20(4):382–399, 2006.

- Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1853–1865, 2016.
- Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of the 24th International Conference on Machine Learning*, volume 227, pages 193–200, 01 2007. doi: 10.1145/1273496.1273521.
- Hal Daumé III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL <https://www.aclweb.org/anthology/P07-1033>.
- Erlend Davidson. Deep domain adaptation networks, 2018. URL <https://github.com/erlenddd/ddan>.
- Antoine de Mathelin, Guillaume Richard, Francois Deheeger, Mathilde Mougeot, and Nicolas Vayatis. Adversarial weighting for domain adaptation in regression. *arXiv preprint arXiv:2006.08251*, 2020.
- Antoine de Mathelin, François Deheeger, Guillaume Richard, Mathilde Mougeot, and Nicolas Vayatis. Adapt: Awesome domain adaptation python toolbox. *arXiv preprint arXiv:2107.03049*, 2021.
- Antoine de Mathelin, Francois Deheeger, Mathilde Mougeot, and Nicolas Vayatis. Fast and accurate importance weighting for correcting sample bias. *arXiv preprint arXiv:2209.04215*, 2022.
- Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *Proceedings of the IEEE international conference on computer vision*, pages 2960–2967, 2013.
- Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1):2096–2030, January 2016. ISSN 1532-4435.
- Sankalp Gilda, Antoine de Mathelin, Sabine Bellstedt, and Guillaume Richard. Unsupervised domain adaptation for constraining star formation histories. *arXiv preprint arXiv:2112.14072*, 2021.
- Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *International Conference on Learning Representations*, 2020.
- Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex J. Smola. Correcting sample selection bias by unlabeled data. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 601–608. MIT Press, 2007.

- Mingsheng Long, Junguang Jiang, Bo Fu. Transfer-learning-library. <https://github.com/thuml/Transfer-Learning-Library>, 2020.
- Takafumi Kanamori, Shohei Hido, and Masashi Sugiyama. A least-squares approach to direct importance estimation. *The Journal of Machine Learning Research*, 10:1391–1445, 2009.
- Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 6028–6039. PMLR, 2020.
- Mingsheng Long, ZHANGJIE CAO, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1640–1650. Curran Associates, Inc., 2018.
- Marco Loog. Nearest neighbor-based importance weighting. In *2012 IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6. IEEE, 2012.
- Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *COLT*, 2009.
- Antoine De Mathelin, François Deheeger, Mathilde Mougeot, and Nicolas Vayatis. Handling distribution shift in tire design. In *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2021. URL <https://openreview.net/forum?id=WOfKtUQgcRR>.
- Ludovic Minvielle, Mounir Atiq, Sergio Peignier, and Mathilde Mougeot. Transfer learning on decision tree with class imbalance. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1003–1010, 2019. doi: 10.1109/ICTAI.2019.00141.
- Saeid Motiian, Marco Piccirilli, Donald A Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5715–5725, 2017.
- Kevin Musgrave. Pytorch-adapt. <https://github.com/KevinMusgrave/pytorch-adapt>, 2021.
- M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014.
- Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2010.
- David Pardoe and Peter Stone. Boosting for regression transfer. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, June 2010.

- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- Hariharan Ravishankar, Prasad Sudhakar, Rahul Venkataramani, Sheshadri Thiruvenkadam, Pavan Annangi, Narayanan Babu, and Vivek Vaidya. Understanding the mechanisms of deep transfer learning for medical images. In *Deep learning and data labeling for medical applications*, pages 188–196. Springer, 2016.
- Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3723–3732, 2018.
- Steffen Schneider, Alexander S. Ecker, Jakob H. Macke, and Matthias Bethge. Salad: A toolbox for semi-supervised adaptive learning across domains, 2018. URL <https://openreview.net/forum?id=S1lTifykqm>.
- N. Segev, M. Harel, S. Mannor, K. Crammer, and R. El-Yaniv. Learn on source, refine on target: A model transfer learning framework with random forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9):1811–1824, 2017.
- Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. Wasserstein distance guided representation learning for domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul von Büna, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Proceedings of the 20th International Conference on Neural Information Processing Systems, NIPS'07*, page 1433–1440, Red Hook, NY, USA, 2007. Curran Associates Inc. ISBN 9781605603520.
- Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *European conference on computer vision*, pages 443–450. Springer, 2016.
- Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

- Anne-Marie Tousch and Christophe Renaudin. (yet) another domain adaptation library, 2020. URL <https://github.com/criteo-research/pytorch-ada>.
- Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.
- Selen Uguroglu and Jaime Carbonell. Feature selection for transfer learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 430–442. Springer, 2011.
- V Vercruyssen. Transfertools, 2020. URL <https://github.com/Vincent-Vercruyssen/transfertools>.
- Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):9, May 2016. ISSN 2196-1115. doi: 10.1186/s40537-016-0043-6. URL <https://doi.org/10.1186/s40537-016-0043-6>.
- Kouw Wouter. libtlda. <https://github.com/wmkouw/libTLDA>, 2015.
- Makoto Yamada, Taiji Suzuki, Takafumi Kanamori, Hirotaka Hachiya, and Masashi Sugiyama. Relative density-ratio estimation for robust distribution comparison. *Neural computation*, 25(5):1324–1370, 2013.
- Ke Yan. Domain adaptation toolbox, 2016. URL <https://github.com/viggin/domain-adaptation-toolbox>.
- Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. Bridging theory and algorithm for domain adaptation. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7404–7413, Long Beach, California, USA, 09–15 Jun 2019. PMLR.
- Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain adaptive ensemble learning. *IEEE Transactions on Image Processing (TIP)*, 2021.
- Yongchun Zhu, Fuzhen Zhuang, and Deqing Wang. Aligning domain-specific distribution and classifier for cross-domain classification from multiple sources. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5989–5996, 2019.
- Fuzhen Zhuang, Keyu Duan, Tongjia Guo, Yongchun Zhu, Dongbo Xi, Zhiyuan Qi, and Qing He. Transfer learning toolkit: Primers and benchmarks, 2019.

Appendix A : list of implemented methods

Table 2: List of the implemented methods in the ADAPT library.

	Method	Supervision	Stages
Feature	FA (Daumé III, 2007)	SDA	2-stages
	TCA (Pan et al., 2010)	UDA	2-stages
	fMMD (Uguroglu and Carbonell, 2011)	UDA	2-stages
	SA (Fernando et al., 2013)	UDA	2-stages
	CORAL (Sun et al., 2016)	UDA	2-stages
	DeepCORAL (Sun and Saenko, 2016)	UDA	1-stage
	DANN (Ganin et al., 2016)	UDA	1-stage
	ADDA (Tzeng et al., 2017)	UDA	2-stage
	WDGRL (Shen et al., 2018)	UDA	1-stage
	CCSA (Motiian et al., 2017)	SDA	1-stage
	CDAN (Long et al., 2018)	UDA	1-stage
	MCD (Saito et al., 2018)	UDA	1-stage
	MDD (Zhang et al., 2019)	UDA	1-stage
Instance	KMM (Huang et al., 2007)	UDA	2-stages
	KLIEP (Sugiyama et al., 2007)	UDA	2-stages
	TrAdaBoost (Dai et al., 2007)	SDA	1-stage
	IWC (Bickel et al., 2007)	UDA	2-stages
	LDM (Mansour et al., 2009)	UDA	2-stages
	ULSIF (Kanamori et al., 2009)	UDA	2-stages
	TrAdaBoostR2 (Pardoe and Stone, 2010)	SDA	1-stage
	TwoStages-TrAdaBoostR2 (Pardoe and Stone, 2010)	SDA	1-stage
	NNW (Loog, 2012)	UDA	2-stages
	RULSIF (Yamada et al., 2013)	UDA	2-stages
	WANN (de Mathelin et al., 2020)	SDA	1-stage
	IWN (de Mathelin et al., 2022)	UDA	2-stages
Param.	Regular Transfer LR (Chelba and Acero, 2006)	SDA	1-stage
	Regular Transfer LC (Chelba and Acero, 2006)	SDA	1-stage
	Regular Transfer NN (Chelba and Acero, 2006)	SDA	1-stage
	Fine-Tuning (Oquab et al., 2014)	SDA	1-stage
	SER-STRUT (Segev et al., 2017)	SDA	1-stage
	SER*-STRUT* (Minvielle et al., 2019)	SDA	1-stage