SALAD: A Toolbox for Semi-supervised Adaptive Learning Across Domains

Steffen Schneider[1,2], **Alexander S. Ecker**[1], **Jakob H. Macke**[2], **Matthias Bethge**[1]
[1] University of Tübingen
[2] Technical University of Munich
steffen.schneider@tum.de

Abstract

We introduce `salad`, an open source toolbox that provides a unified implementation of state-of-the-art methods for transfer learning, semi-supervised learning and domain adaptation. In the first release, we provide a framework for reproducing, extending and combining research results of the past years, including model architectures, loss functions and training algorithms. The toolbox along with first benchmark results and further resources is accessible at `domainadaptation.org`.

1 Introduction

Domain Adaptation (DA) is a topic of active research and practical relevance in areas such as computer vision (e.g. [5]), medical imaging (e.g. [23, 2]) and speech processing (e.g. [24]). Classical approaches usually made use of the assumption of covariate shift and the accompanying theory [8, 30, 1, 18]. More recently, the growing popularity of deep learning approaches has lead to novel concepts based on feature and signal transformations [26, 10], feature alignment [9, 27] or adversarial training [7].

Changing standards of evaluation and benchmark tasks and the accumulation of new algorithms, datasets and model architectures have made it increasingly difficult to assess the state of the art. Reference implementations are scattered over software frameworks and are often specific to benchmarking setups. As a consequence, comparing the performance of algorithms with new model architectures and on new datasets is difficult and time-consuming.

To overcome these limitations, we developed `salad`, a library comprised of methods for "Semi-supervised and Adaptive Learning Across Domains." The toolbox bundles popular algorithms and provides both a benchmark setup and easy integration in practical applications. Our main goals are (1) to provide a benchmark utility that provides results for published and frequently referenced approaches, (2) to abstract DA approaches relying on the same theoretical intuitions (e.g., many approaches are based on applying a metric/divergence to the features on source and target domain), (3) to speed up research and discovery of conceptually new DA algorithms, (4) to provide DA algorithms in an easy-to-use form for integration into open source projects.

In this paper, we introduce `salad`, highlight its main features and showcase the applicability of the toolbox. We replicate previous results and perform a comprehensive comparative study on multiple adaptation problems (see Fig. 1). New results on different model architectures and datasets are constantly tracked on the project homepage and will be expanded with the introduction of new algorithms.

`salad` makes use of PyTorch [19], Matplotlib [12], Seaborn [29], Numpy [17], Scipy [13], scikit-learn [20] and Pandas [14]. We drew additional inspiration from the Python Optimal Transport Library [4] and several original reference implementations cited below.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.
Workshop on Machine Learning Open Source Software: Sustainable Communities

2 Toolbox Overview

A common goal of DA approaches is risk minimization (RM) on a target domain $\mathcal{T}$, which we regard as a joint distribution of signals x and discrete labels y. Given the risk $\mathcal{R}^l_{\mathcal{T}}$, we aim at finding model parameters θ that minimize the risk

$$\min_\theta \mathcal{R}^l_{\mathcal{T}}(\theta) = \mathbb{E}_{x,y\sim\mathcal{T}}[\ell(h_\theta(x), y)]. \tag{1}$$

Since labels for the target domain are either scarce or not available, the problem is not directly approachable by standard machine learning algorithms, motivating the use of DA approaches.

We organize the paper accordingly: In §2.1, we outline the `solver` subpackage that provides abstractions between the common training loop and the actual contribution of a particular DA method. We focus on two main variations of approximating the RM problem described above, unsupervised domain adaptation and domain generalization. Models, layers and convenience functions for adaptation of trained models to new datasets are available in the `layers` and `models` subpackages, outlined in §2.2. Finally, many algorithms require data augmentation techniques, which are implemented in the `datasets` package and described in §2.3. A detailed overview of how `salad` incorporates different algorithms is given in §A in the supplementary material and in the API documentation.

2.1 Solvers `[salad.solver]`

Unsupervised Domain Adaptation `[salad.solver.DABaseSolver]` assumes the presence of a single source domain $\mathcal{S}$ along with a target domain $\mathcal{T}$ known at training time. Given a labeled sample of points drawn from $\mathcal{S}$, $\{(x^s_i, y^s_i)\}_{i=1}^{N_s}$, and an unlabeled sample drawn from $\mathcal{T}$, $\{x^t_i\}_{i=1}^{N_t}$, unsupervised adaptation aims at minimizing the risk

$$\min_\theta \mathcal{R}^l_{\mathcal{S}}(\theta) + \lambda\mathcal{R}^u_{\mathcal{S}\times\mathcal{T}}(\theta) = \mathbb{E}_{x,y\sim\mathcal{S}}[\ell(x, y; \theta)] + \lambda\mathbb{E}_{x^s,y^s,x^t\sim\mathcal{S}\times\mathcal{T}}[\ell(x^s, y^s, x^t; \theta)], \tag{2}$$

leveraging an unsupervised risk term $\mathcal{R}^u_{\mathcal{S}\times\mathcal{T}}(\theta)$ that depends on feature representations $f_\theta(x^s, s)$ and $f_\theta(x^t, t)$, classifier labels $h_\theta(x^s, s), h_\theta(x^t, t)$ as well as source labels y^s. The full model $h = g \circ f$ is a composition of a feature extractor f and classifier g, both of which can possibly depend on the domain label s or t for domain-specific computations. This formulation is still very generic; in the following, we briefly discuss three high level approaches for implementing $\mathcal{R}^u_{\mathcal{S}\times\mathcal{T}}(\theta)$.

First, proxy-labeling and ensembling are crucial parts in current state-of-the art approaches (e.g. [25, 5]), requiring both a separate teacher model providing approximate labels and data augmentation techniques to enforce consistencies between differently perturbed target samples, cf. §A.1. Second, a more classical way of implementing the unsupervised loss is a discrepancy term between target and source feature representations, such as explicit distances between covariance matrices (e.g. [9, 26, 15]) or by leveraging adversarial training [7], cf. §A.2. Third, domain translation aims at learning maps between features or even input signals between both domains, possibly in form of a probability distribution $p(x^s|x^t)$ and/or $p(x^t|x^s)$, preserving the label information, cf. §A.3.

Unifying the implementation of unsupervised DA approaches forms an essential part of the first release of `salad`. We provide implementations of [5, 25, 27, 9, 7] and summarize the benchmarking results for training the small digit model used by [5] on different digit benchmarks in Fig. 1.

Domain Generalization `[salad.solver.DGBaseSolver]` assumes the presence of multiple source domains alongside a target domain unknown at training time. Following [24], this setting requires a dataset of training examples $\{(x_i, y_i, d_i)\}_{i=1}^N$ with class and domain labels. Importantly, the domains present at training time should reflect the kind of variability that can be expected during inference. The RM problem is then approached as

$$\min_\theta \sum_d \lambda_d \mathcal{R}^l_{\mathcal{S}_d}(\theta) = \sum_d \lambda_d \mathbb{E}_{x,y\sim\mathcal{S}_d}[\ell(x, y, d; \theta)], \tag{3}$$

where the different losses are weighted by hyperparameters λ_d. In contrast to the unsupervised setting, samples are now presented in a single batch comprised of inputs x, labels y and domains d.

function UNSUPERVISED DA
$x^s, y^s \sim \mathcal{S}$
$x^t \sim \mathcal{T}$
$\mathcal{R}^l(\theta) \leftarrow \ell^l(x^s, y^s, \theta)$
$\mathcal{R}^u(\theta) \leftarrow \ell^u(x^s, y^s, x^t, \theta)$
for all optimizers and params θ_i **do**
$\Delta\theta_i \leftarrow \nabla_{\theta_i}(\mathcal{R}^l(\theta) + \lambda\mathcal{R}^u(\theta))$
update θ_i (e.g., using ADAM)
end for
end function

function DOMAIN GENERALIZATION
$x, y, d \sim \mathcal{S}$
$\mathcal{R}^l(\theta) \leftarrow \ell^l(x, y, d)$
for all optimizers and params θ_i **do**
$\Delta\theta_i \leftarrow \nabla_{\theta_i}\mathcal{R}^l(\theta)$
update θ_i (e.g.,using ADAM)
end for
end function

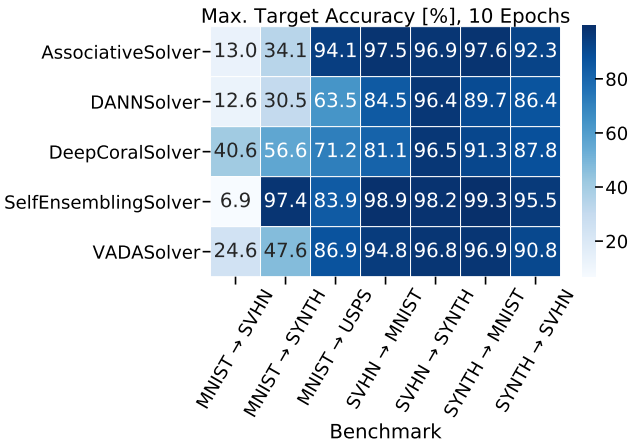

Figure 1: Left: Unsupervised DA and Domain Generalization in `salad`. Right: Results of running the `salad` example script with algorithms proposed by [9, 7, 27, 5, 25] on different DA benchmark tasks using the model and augmentation mechanism from [5]. We could generally reproduce results of the reported papers when using comparable hyperparameter settings and conclude that self-ensembling with a good noise model sets the current state of the art. Up-to-date results for different training settings will be tracked on the project homepage.

In a addition to a feature extractor f_θ and classifier g_θ, models should also provide a feature extractor f^d_θ to derive domain features along with a domain classifier g^d_θ, with possibly shared parameters.

In contrast to unsupervised DA, this training setting leverages information from multiple labeled source domains with the goal of generalizing well on data from a previously unseen domain during test time. Domain generalization might be regarded as "few-shot unsupervised DA": Instead of access to a large unlabeled sample from $\mathcal{T}$ that allows for an optimization-based adaptation, the adaptation process is a direct part of the inference procedure.

2.2 Models & Adaptation Tools `[salad.layers, salad.models]`

The toolbox provides a variety of network layers necessary for running adaptation algorithms, e.g. for separating running statistics of the source and target domain and other forms of domain-conditional computations. For "warm-starting" finetuning and applying models to a novel domain, the package provides additional tools to quickly re-calculate network parameters depending on dataset statistics.

2.3 Datasets, Noise Models and Pertubations `[salad.datasets]`

For techniques that incorporate self-labeling, it is crucial to augment the task with a noise model to artificially create additional data domains or enforce consistency constraints [5, 25]. Two implementations exist in `salad`: First, pertubation can be applied to the input image in form of a noise distribution $p_{\tilde{x}|x}(\tilde{x}|x, n) = p_{\tilde{x}|x}(\phi_n(x)|x, n)$ where ϕ_n is an image transformation preserving labels. Second, pertubations can be applied internally to f_θ or g_θ as suggested by [6, 21].

For tasks beyond classification, such as semantic and instance segmentation, the package provides both benchmarking datasets and transformations that jointly operate in pixel and coordinate space.

3 Discussion and Outlook

We presented the `salad` toolbox and an accompanying website with the goal of unifying domain adaptation approaches, fostering reproducibility, fair comparisons and application of modern domain adaptation algorithms. We invite researchers and software engineers to collaborate with us in extending the toolbox and offering implementations for other frameworks such as Tensorflow. In future work, we will extend the toolbox with new algorithms, especially translation-based approaches.

Acknowledgements

We thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting St.S.; we thank Arthur Pesah for permission to include his extensive transfer learning reading list on the project homepage; we thank Luisa Eck for proposing the toolbox name; we thank the two anonymous reviewers for their helpful comments for improving our submission and the software package.

Remarks

`salad` is distributed under the terms of the Mozilla Public Licence 2.0 (MPL-2.0). Parts of data loading routines, data augmentation and model implementations are adapted or inspired by existing open source projects such as PyTorch [19] and implementations by [9, 5, 25], under the licensing terms of these respective projects. For more information on the implemented algorithms, the documentation at `salad.domainadaptation.org` provides the necessary details.

References

[1] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, 2010.

[2] D. Bug, S. Schneider, A. Grote, E. Oswald, F. Feuerhake, J. Schüler, and D. Merhof. Context-based Normalization of Histological Stains using Deep Convolutional Features. *3rd Workshop on Deep Learning in Medical Image Analysis (DLMIA)*, 2017.

[3] F. M. Cariucci, L. Porzi, B. Caputo, E. Ricci, and S. R. Bulo. AutoDIAL: Automatic Domain Alignment Layers. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-Octob(4):5077–5085, 2017.

[4] R. Flamary. Optimal transport for machine learning. *AG GDR ISIS*, 2017.

[5] G. French, M. Mackiewicz, and M. Fisher. Self-ensembling for visual domain adaptation. pages 1–15, 2017.

[6] Y. Gal and Z. Ghahramani. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. *arXiv preprint*, 2015.

[7] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, V. Lempitsky, U. Dogan, M. Kloft, F. Orabona, and T. Tommasi. Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research*, 17:1–35, 2016.

[8] A. Gretton, A. J. Smola, J. Huang, M. Schmittfull, K. M. Borgwardt, and B. Scholkopf. Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4):5, 2009.

[9] P. Haeusser, T. Frerix, A. Mordvintsev, and D. Cremers. Associative Domain Adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2017-Octob, pages 2784–2792, 2017.

[10] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell. CyCADA: Cycle-Consistent Adversarial Domain Adaptation. Technical report, 2017.

[11] R. A. Hoffman, S. Kothari, and M. D. Wang. Comparison of normalization algorithms for cross-batch color segmentation of histopathological images. *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual Conference*, 2014:194–197, 2014.

[12] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007.

[13] E. Jones, T. E. Oliphant, P. Peterson, and Others. SciPy: Open source scientific tools for Python, 2007.

[14] W. McKinney. Data Structures for Statistical Computing in Python. In S. van der Walt and J. Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51–56, 2010.

[15] P. Morerio, J. Cavazza, and V. Murino. Minimal-Entropy Correlation Alignment for Unsupervised Deep Domain Adaptation. *International Conference on Learning Representations*, pages 1–15, 2017.

[16] S. Motiian, Q. Jones, S. Iranmanesh, and G. Doretto. Few-Shot Adversarial Domain Adaptation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6670–6680. Curran Associates, Inc., 2017.

[17] T. E. Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[18] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[19] A. Paszke, G. Chanan, Z. Lin, S. Gross, E. Yang, L. Antiga, and Z. Devito. Automatic differentiation in PyTorch. In *Advances in Neural Information Processing Systems 30*, number Nips, pages 1–4, 2017.

[20] 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.

[21] K. Saito, Y. Ushiku, and T. Harada. Asymmetric Tri-training for Unsupervised Domain Adaptation. *CVPR*, 2017.

[22] K. Saito, Y. Ushiku, T. Harada, and K. Saenko. Adversarial Dropout Regularization. pages 1–15, 2017.

[23] M. T. Shaban, C. Baur, N. Navab, and S. Albarqouni. StainGAN: Stain Style Transfer for Digital Histological Images. *CoRR*, 2018.

[24] S. Shankar, V. Piratla, S. Chakrabarti, S. Chaudhuri, P. Jyothi, and S. Sarawagi. Generalizing Across Domains via Cross-Gradient Training. *International Conference on Learning Representations*, 1(2015):1–12, 2018.

[25] R. Shu, H. H. Bui, H. Narui, and S. Ermon. A DIRT-T Approach to Unsupervised Domain Adaptation. *International Conference on Learning Representations*, 2018.

[26] B. Sun, J. Feng, and K. Saenko. Return of Frustratingly Easy Domain Adaptation. 2015.

[27] B. Sun, J. Feng, and K. Saenko. Correlation alignment for unsupervised domain adaptation. In *Advances in Computer Vision and Pattern Recognition*, number 9783319583464, pages 153–171. 2017.

[28] A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. 2017.

[29] M. Waskom, O. Botvinnik, P. Hobson, J. B. Cole, Y. Halchenko, S. Hoyer, A. Miles, T. Augspurger, T. Yarkoni, T. Megies, and others. seaborn: v0. 5.0 (November 2014). *Zenodo, doi*, 10, 2014.

[30] K. Zhang, M. Gong, and B. Schölkopf. Multi-Source Domain Adaptation: A Causal View.

Supplementary Material

A Algorithms `[salad.solver]`

Experiment setups are implemented as classes within the `solver` package, as subclasses of `solver.Solver`. In the following, we list several algorithms that are either already implemented in `salad` or soon to be added to the package, grouped into five different principles. The first three, proxy-labeling, feature distances and translation based DA, will be further discussed in §A.1, §A.2 and §A.3 respectively and are special cases of unsupervised domain adaptation as covered in §2.1 in the main text. Note that many of the particular algorithms are using a mixture of different components, and grouping is done based on the most important one used in each algorithm.

Proxy Labels		
Self Ensembling	[5]	`EnsemblingSolver`
Virtual Adversarial Domain Adaptation (VADA)	[25]	`VADASolver`
Adversarial Dropout Regularization	[22]	`AdvDropoutSolver`
Correlation Distance Based		
Deep CORAL	[26]	`CoralSolver`
Log CORAL	[15]	`LogCoralSolver`
Domain Adversarial Training	[7]	`DANNSolver`
Associative Domain Adaptation	[9]	`AssociationSolver`
Translation Based		
CORAL	[27]	–
Cyclic Domain Adaptation	[10]	`CyclicDASolver`
Domain Generalization		
Cross Gradient Training	[24]	`CrossGradSolver`
Domain Adversarial Training	[7]	`MultiDANNSolver`
Finetuning		
DIRT-T	[25]	`DIRTTSolver`

Most importantly, `salad` is designed to make it easy to combine concepts employed in different algorithms, by abstracting optimization routines, loss functions and models. Several of the core concepts will be reviewed below. For the algorithms currently implemented in `salad`, we will provide details about how the unsupervised loss $\mathcal{R}^u_{\mathcal{S}\times\mathcal{T}}(\theta)$ used for adaptation is computed.

Constraints on the formulation of models are intentionally kept as low as possible. Given an input space $\mathcal{X}$, a feature space $\mathcal{Z}$ and a label space $\mathcal{Y}$, we will consider models $h_\theta : \mathcal{X} \mapsto \mathcal{Y}$ which can be decomposed into a feature extractor $f_\theta : \mathcal{X} \mapsto \mathcal{Z}$ and a classifier $g_\theta : \mathcal{Z} \mapsto \mathcal{Y}$. Concerning notation, by the label space we denote the individual class probabilities $\mathcal{Y} \subset \mathbb{R}^{N\times C}$ for a sample, which can then be mapped to labels $l \in \mathbb{N}^N$, by means of selecting the maximum probability.

Following this notation, the supervised risk on the source domain, $\mathcal{R}^l_{\mathcal{S}}(\theta)$ can be written as

$$\mathcal{R}^l_{\mathcal{S}}(\theta) = \mathcal{D}_{\mathrm{KL}}(y^s \| h_\theta(x^s)) = \mathbb{E}_{x^s,y^s\sim\mathcal{S}}[\langle y^s, -\log h_\theta(x^s)\rangle]. \tag{4}$$

Note that the supervised loss function can also be adapted to other target spaces $\mathcal{Y}$, e.g. for tasks such as instance or semantic image segmentation or regression problems.

A.1 Proxy-Labeling for Domain Adaptation

A first important class of approaches, used among others by [5] and [25], is concerned with obtaining approximate labels on the target domain. Given a proxy-label function $h^* : \mathcal{X} \mapsto \mathcal{Y}$, the unsupervised loss can be expressed as

$$\mathcal{R}^u_{\mathcal{S}\times\mathcal{T}}(\theta) = \mathbb{E}_{x^t\sim\mathcal{T}}[\ell_y(h_\theta(x^t), h^*(x^t))]. \tag{5}$$

Different modifications to this general formulation are possible, which are not reflected in the equation for simplified notation, but are straightforward to implement in `salad`. First, in the work by [5], an additional noise

model $p(\tilde{x}|x)$ is employed on the target sample x^t that (partly) reflects the variability between source and target domains. Second, [21] incorporate the noise model directly into the classifier h by means of dropout layers. Third, [25] use virtual adversarial training to compute a perturbed sample $x_t + \epsilon$, which amounts to a noise model with an adversarial input distribution.

Apart from the exact implementation of the noise model, possible choices for the loss function include

$$\ell_y(y, y^*) = \mathcal{D}_{\text{KL}}(y^* \| y) = \frac{1}{n_t} \sum_i \langle y_i^*, -\log y_i \rangle, \tag{6}$$

$$\ell_y(y, y^*) = \frac{1}{n_t} \sum_i \|y_i^* - y_i\|^2, \tag{7}$$

$$\ell_y(y, y^*) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \langle y_i^*, -\log y_i \rangle, \text{where } \mathcal{I} = \{i : y_i^* > \epsilon\} \tag{8}$$

where ϵ denotes a confidence threshold that determines whether or not a particular proxy-label should be included in the loss function, as used by [5].

In addition, different choices of proxy-labeling functions are possible. Let us denote by θ_τ the collection of model parameters after τ training batches and by t the count of the current batch, by $\alpha \in (0, 1)$ a decay parameters for exponential averaging, then recent approaches made use of the following proxy-labeling functions h^*:

$$h^*(x) = h_\theta(x), \qquad \text{(Self-Labeling, Entropy Minimization)}$$

$$h^*(x) = h_{\theta^*}(x), \text{where } \theta^* = \sum_{\tau=1}^{t} \alpha^{t-\tau} \theta_\tau, \qquad \text{(Mean Teacher, Self-Ensembling)}$$

$$h^*(x) = \sum_{\tau=1}^{t} \alpha^{t-\tau} h_{\theta_\tau}(x). \qquad \text{(Mean Labeling)}$$

In recent work, [5] proposed the self-ensembling formulation, which is currently implemented in the `SelfEnsemblingSolver` while [25] combined, among adversarial training, entropy-minimization during the phase implemented in the `VADASolver` as well as self-ensembling during the refinement phase implemented in the `DIRTTSolver`.

A.2 Feature Distance Based Domain Adaptation

A second class of approaches implements the unsupervised loss $\mathcal{R}^u_{\mathcal{S}\times\mathcal{T}}(\theta)$ as a distance between the feature representations on the source and target domain, $z^s := f_\theta(x^s)$ and $z^t := f_\theta(x^t)$, respectively. This yields the formulation of the unsupervised loss

$$\mathcal{R}^u_{\mathcal{S}\times\mathcal{T}}(\theta) = \mathbb{E}_{x^s, y^s, x^t \sim \mathcal{S}\times\mathcal{T}}[\ell_z(f_\theta(x^s), f_\theta(x^t), y^s)]. \tag{9}$$

Note that apart from the features z^s and z^t, the loss function ℓ_z might also incorporate information from the source label tensor y^s. Let us denote by $C(\cdot)$ the computation of the covariance matrix of a batch of examples, by $d_\phi : \mathcal{Z} \mapsto [0, 1]$ a discriminator obtained in adversarial training [7] and by $K(\cdot, \cdot)$ a normalized exponential kernel matrix with entries $K_{ij}(z^s, z^t) := Z^{-1} \exp(\langle z_i^s, z_j^t \rangle)$ [9]. Possible loss functions between feature embeddings then include:

$$\ell_z(z^s, z^t) = \frac{1}{4d^2} \|C(z^s) - C(z^t)\|_F^2 \qquad \text{(Deep CORAL)}$$

$$\ell_z(z^s, z^t) = \frac{1}{4d^2} \|\log C(z^s) - \log C(z^t)\|_F^2 \qquad \text{(Log CORAL, geodesic distance)}$$

$$\ell_z(z^s, z^t) = \frac{1}{n_s} \sum_i -\log d_\phi(z_i^s) + \frac{1}{n_t} \sum_i -\log(1 - d_\phi(z_i^t)) \qquad \text{(Domain Adversarial Training)}$$

$$\ell_z(z^s, z^t, y^s) = \mathcal{D}_{\text{KL}}(y_s y_s^T \| K(z^s, z^t)^T K(z^s, z^t)) \qquad \text{(Associative Domain Adaptation)}$$

Note that while many of these distance functions were investigated in separate publications, they might well be used in conjunction. `salad` makes it easy to blend different distance functions and write new solvers that combine existing concepts.

For instance, many domain adaptation approaches used in recent literature build on top of Domain Adversarial Networks [7], which is also reflected in the structure of `salad`: By deriving a solver from the `DANNSolver`, it is easy to build new algorithms on top. As an already implemented example, also refer to the `VADASolver`, which uses adversarial training as one component.

A.3 Translation Based Domain Adaptation

While not being prominently supported by the library yet, translation-based approaches will play an important role in extending `salad` in the future. Frequently, these approaches jointly train translation functions $T_\theta^{s\mapsto t}$ and $T_\theta^{t\mapsto s}$ that translate either examples x^s, x^t or features z^s, z^t from the source into target domain, or vice versa. Another approach is to explicitly formulate the translation function, e.g. by a whitening-coloring transformation [26].

A formulation of the unsupervised loss, excluding the adversarial and cycle consistency objectives needed to train T, will leverage the fact that following translation, label information for the samples is available:

$$\mathcal{R}^u_{\mathcal{S}\times\mathcal{T}}(\theta) = \mathbb{E}_{x^s,x^t\sim\mathcal{S}\times\mathcal{T}}[\ell_y(h_\theta(T_\theta^{s\mapsto t}(x^s), y^s)]. \tag{10}$$

Note that for training the translation functions as well as choosing suitable loss functions ℓ_y, results from §A.1 and §A.2 might be re-used. Translation-based methods were, among others, proposed and used by [11, 10, 23] in different application settings.

While as of now, at least within classification benchmark settings, they do not (yet) exceed the performance of discriminative domain adaptation approaches such as [5, 25], they might play an important role in settings where the target domain is "richer" than the source domain, such as the MNIST↦SVHN setting, which is a challenging task. Therefor, using translation-based DA as trainable noise models for discriminative approaches might be an interesting direction to follow.

B Datasets `[salad.datasets]`

In addition to the classical digit benchmarks, the `datasets` offers methods to create new synthetic and controllable data sources (e.g. by adding noise or image transformations), to vary the existing datasets and create new training setups (for openset domain adaptation or scenarios such as partial overlapping classes between source and target datasets) or to consider settings beyond classification (such as instance segmentation). While in the first release of `salad`, we focus on providing example scripts for classical benchmarks, it is straightforward to implement new settings and integrate them into the existing processing pipelines.

- Toy Tasks (e.g., Moon Dataset)
- Standard Benchmarks: MNIST, SVHN, UPSP, SYNTH
- Noise Benchmarks (e.g. Gaussian, Salt and Pepper, Rotations)
- MNIST/SVHN Semantic Segmentation Task
- VisDA classification challenge 2017
- VisDA detection and openset challenge 2018

C Experiments `[salad.examples]`

In the first release, we provide reference implementations of recently published domain adaptation algorithms such as

- Correlation Alignment [27] and variants
- Self Ensembling for Visual Domain Adaptation [5]
- A DIRT-T Approach to Unsupervised Domain Adaptation [25]
- Domain-Adversarial Training of Neural Networks [7]
- Associative Domain Adaptation [9]
- Generalizing Across Domains Via Cross-Gradient Training [24].

Partially finished implementations include

- Return of Frustratingly Easy Domain Adaptation [26]
- Few-Shot Adversarial Domain Adaptation [16]
- Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results [28]

D Network Layers and Loss Functions `[salad.layers]`

The aforementioned algorithms usually use novel metrics and training schemes. In order to encourage development of new algorithms for particular problems, `salad` makes it easy to re-use parts of existing algorithms. Among other functions, we implement the following loss functions:

- Distance Functions between covariate matrices (CORAL, Deep CORAL, Log CORAL)
- Visit and Walker losses for Associative Domain Adaptation
- Virtual Adversarial Training
- Confidence Weighted Cross Entropy
- Conditional Entropy

and the following network layers:

- Conditional Batch Normalization
- Feature Aware Normalization [2]
- AutoDIAL [3]

We provide model implementations for conditional training similar to the PyTorch ResNet implementation.