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

Steffen Schneider¹,², Alexander S. Ecker¹, Jakob H. Macke², Matthias Bethge¹
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. [22, 2]) and speech processing (e.g. [23]). Classical approaches usually made use of the assumption of covariate shift and the accompanying theory [8, 29, 1, 17]. More recently, the growing popularity of deep learning approaches has lead to novel concepts based on feature and signal transformations [25, 10], feature alignment [9, 26] 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 [18], Matplotlib [11], Seaborn [28], Numpy [16], Scipy [12], scikit-learn [19] and Pandas [13]. We drew additional inspiration from the Python Optimal Transport Library [4] and several original reference implementations cited below.

Preprint. Work in progress.
2 Toolbox Overview

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

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

(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 section 2.1, we outline the \texttt{salad.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 ERM 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 \texttt{layers} and \texttt{models} subpackages, outlined in section 2.2. Finally, many algorithms require data augmentation techniques, which are implemented in the \texttt{datasets} package and described in section 2.3. For a more detailed overview, please refer to the supplementary material and the API documentation.

2.1 Solvers \texttt{[salad.solver]}

Unsupervised Domain Adaptation \texttt{[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 \( \mathcal{S} \), \( \{x_i^s, y_i^s\}_{i=1}^{N_s} \), and an unlabeled sample drawn from \( \mathcal{T} \), \( \{x_i^t\}_{i=1}^{N_t} \), unsupervised adaptation aims at minimizing the risk

\[
\min_\theta \mathcal{R}_\mathcal{S}^u(\theta) + \lambda \mathcal{R}_{\mathcal{S} \times \mathcal{T}}^u(\theta),
\]  

(2)

leveraging an unsupervised risk term \( \mathcal{R}_{\mathcal{S} \times \mathcal{T}}^u(\theta) \) 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 \( \mathcal{R}_{\mathcal{S} \times \mathcal{T}}^u(\theta) \).

First, a 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. \cite{9, 25, 14}) or by leveraging adversarial training \cite{7}. Second, proxy-labeling and ensembling are crucial parts in current state-of-the art approaches (e.g. \cite{24, 5}), requiring both a separate teacher model providing approximate labels and data augmentation techniques to enforce consistencies between differently perturbed target samples. Third, domain translation aims at learning maps between features or even input signals between both domains, possibly in form of a probability distribution \( p(x^t|x^s) \) and/or \( p(x^t|x^s) \), preserving the label information.

Unsupervised DA approaches form an essential part of the first release of \texttt{sald}. We provide implementations of \cite{5, 24, 26, 9, 7} and summarize the benchmarking results for training the small digit model used by \cite{5} on different digit benchmarks in Fig. 1.

Domain Generalization \texttt{[salad.solver.DGBaseSolver]} assumes the presence of multiple source domains alongside a target domain unknown at training time. Following \cite{23}, 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 ERM problem is then approached as

\[
\min_\theta \sum_d \mathcal{R}_{\mathcal{S}_d}^l(\theta) = \sum_d \lambda_d \mathbb{E}_{x,y \sim \mathcal{S}_d}[\ell(f_\theta(x), h_\theta(x), y, d)].
\]  

(3)

In contrast to the unsupervised setting, samples are now presented in a single batch comprised of inputs \( x \), labels \( y \) and domains \( d \). In a addition to a feature extractor \( f_\theta \) and classifier \( g_\theta \), models
We presented the 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.

References


Supplementary Material

**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.

<table>
<thead>
<tr>
<th>Proxy Labels</th>
<th>Correlation Distance Based</th>
<th>Translation Based</th>
<th>Domain Generalization</th>
<th>Finetuning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self Ensembling</td>
<td>Deep CORAL</td>
<td>CORAL</td>
<td>Cross Gradient Training</td>
<td>DIRT-T</td>
</tr>
<tr>
<td>Virtual Adversarial Domain Adaptation (VADA)</td>
<td>Log CORAL</td>
<td>Cyclic Domain Adaptation</td>
<td>Domain Adversarial Training</td>
<td></td>
</tr>
<tr>
<td>Adversarial Dropout Regularization</td>
<td>Domain Adversarial Training</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Association Domain Adaptation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithms</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>EnsemblingSolver</td>
<td>LogCoralSolver</td>
</tr>
<tr>
<td>VADASolver</td>
<td>DANNSolver</td>
</tr>
<tr>
<td>AdvDropoutSolver</td>
<td>AssociationSolver</td>
</tr>
<tr>
<td>CoralSolver</td>
<td></td>
</tr>
<tr>
<td>LogCoralSolver</td>
<td></td>
</tr>
<tr>
<td>DANNSolver</td>
<td></td>
</tr>
<tr>
<td>AssociationSolver</td>
<td></td>
</tr>
<tr>
<td>CyclicDASolver</td>
<td></td>
</tr>
<tr>
<td>CrossGradSolver</td>
<td></td>
</tr>
<tr>
<td>MultiDANNSolver</td>
<td></td>
</tr>
<tr>
<td>DIRTTSolver</td>
<td></td>
</tr>
</tbody>
</table>

**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, SYNT
- Noise Benchmarks (e.g. Gaussian, Salt and Pepper, Rotations)
- MNIST/SVHN Semantic Segmentation Task
- VisDA classification challenge 2017
- VisDA detection and openset challenge 2018

**Experiments** [salad.examples]

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

- Correlation Alignment [26] and variants
- Self Ensembling for Visual Domain Adaptation [5]
- A DIRT-T Approach to Unsupervised Domain Adaptation [24]
- Domain-Adversarial Training of Neural Networks [7]
- Associative Domain Adaptation [9]
- Generalizing Across Domains Via Cross-Gradient Training [23].
Partially finished implementations include

- Return of Frustratingly Easy Domain Adaptation [25]
- Few-Shot Adversarial Domain Adaptation [15]
- Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results [27]

**Network Layers and Loss Functions** \([\texttt{salad.layers}]\)

The aforementioned algorithms usually use novel metrics and training schemes. In order to encourage development of new algorithms for particular problems, \texttt{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.

**Remarks**

\texttt{salad} is distributed under the terms of the Mozilla Public Licence 2.0 (MPL-2.0). For more details on the algorithms, the documentation at \texttt{salad.domainadaptation.org} provides the necessary details.