DISSERTATION

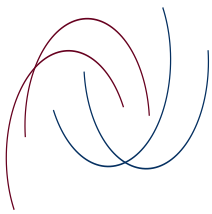
LEARNING IN NON-STATIONARY ENVIRONMENTS

CHRISTOPH RAAB

Bielefeld University,
Faculty of Technology,
Machine Learning Research Group

SUPERVISED BY
PROF. DR. BARBARA HAMMER,
PROF. DR. FRANK-MICHAEL SCHLEIF

SEPTEMBER 6, 2021



ABSTRACT

The topic of Machine Learning deals with learning a data-based decision function. This function assigns correct output information based on input data. The learning process is performed in a laboratory domain and finally applied in a test or application domain. Typically, the distribution of data between the learning and application domain is assumed to be the same, denoted as a stationary environment. If this is not the case and the distributions change between the two domains, it is called a non-stationary environment.

The research area of Domain Adaptation offers methods to adapt the input data or an already learned decision function to the test domain data in such non-stationary environments. Current solutions search for a suitable representation space by minimizing some statistical divergence measure or subspace projector differences. The former methods bound domain differences with computational requirements almost intractable, while the latter formulate no guarantees regarding domain differences.

In the first part of this thesis, I provide geometric- and subspace-oriented projectors built upon the idea of domain separability. The methods outperform recent solutions while being a magnitude faster. Both solutions bound the domain differences by the remaining differences of the spectra, and I show that subspace solutions naturally provide less domain separability.

In the second part of this thesis, I motivated a spectral-based moment-matching regularization loss. The approach is based on the characterization of domain differences via spectral differences described above. Applied to neural networks, domains can be matched in latent or label space. Furthermore, I implemented a relevance weighting that minimizes domain-specific influences, leading to a stable and effective solution.

While the above methods work well for time-independent data with finite sample sizes, tasks such as weather forecasting require fast processing of a potentially unlimited amount of data in a temporal sequence. If the data distribution changes over time, the sequence, also called data stream, is additionally affected by Concept Drift. Domain Adaptation methods are not practical for such scenarios because they do not scale to data stream size. The research area of Concept Drift Stream Classification addresses the challenges just described. It provides a variety of algorithms and techniques to fit and adapt the model with linear complexity. Prototype-based learning is already reasonably flexible in this constraint through an online learning approach but is not yet prepared for Concept Drifts. In the third part of the thesis, I extended probabilistic prototype models by drift detection via a statistical test plus an active and passive Concept Drift adaptation, providing stable and fast concept adaptation against various drift types.

CONTENTS

Contents **vi**

1 Introduction **1**

2 Background **7**
2.1 Singular Value Decomposition 7
2.2 Nyström Approximation 9
2.3 Transfer Learning and Domain Adaptation 12
2.4 Robust Soft Learning Vector Quantization 21

3 Shallow Geometric Domain Adaptation **25**
3.1 Model 26
3.2 Experiments 30
3.3 Discussion 36

4 Shallow Subspace Domain Adaptation **39**
4.1 Model 40
4.2 Nyström Model 46
4.3 Class-Informed Model 48
4.4 Experiments 52
4.5 Discussion 62

5 Deep Spectral Domain Adaptation **65**
5.1 Background and Related Work 67
5.2 Model 70
5.3 Adversarial Relevance Model 76
5.4 Experiments 82
5.5 Discussion 93

6 Non-Stationary Online Prototype Learning **95**
6.1 Background and Related Work 98
6.2 Model 100
6.3 Experiments 110
6.4 Discussion 119

7 Conclusion and Outlook **121**

Publications in the Context of this Thesis **125**

References **127**

Acronyms **145**

Appendix: Additional Results of ASAN **149**
Additional Experimental Results on VisDa 149
Additional Results of Convergence and Spectral Analysis 150
Additional Results of Feature Analysis 151
Examples from Benchmark Datasets 152

Appendix: Streaming Dataset Descriptions **155**
Synthetic Stream Generators 155
Real-world Streams 156

1 INTRODUCTION

Non-stationary environments affect decision processes to the extent that they change underlying mechanics or the information analysis on which the decision is based. This change is present in many aspects of human life: Day-Night-Alternation or temporary weather variations change data representations, leading to a different decision behavior of human and prediction models [53], [172]. Teaching exams evaluate the transfer of knowledge within a decision process of a candidate or a musician who learns a new instrument by partly adapting the knowledge of prior known instruments [39]. Simulated training environments allow the learner to acquire specific information or skills, while a transfer to real situations must eventually take place [106]. The above examples illustrate the nature of non-stationary environments, and it is well-known that humans can handle the above transfer tasks to a certain extend.

The examples just introduced illustrate different characteristics occurring in non-stationary environments [61], [95]. The weather changes described above have a time dependence and can be specified by the change rate. Teaching exams do not necessarily express a change of data representation but in the decision process. The transfer from training obtained skills to the real world poses no time restriction, but an undeniable adaptation of the obtained skills must occur.

In recent decades, the development of intelligent systems used in daily life environments have exposed these systems to non-stationary environments and, therefore, are affected by various types of input data variations [107]. The underlying framework for grasping a decision process based on data is called machine learning. The humans involved spend much time providing labeled data to help machine learning algorithms derive a data-based decision rule in a supervised training environment called training or source domain [83]. However, the obtained machine learning models suffer from being restricted to the given training distribution. They cannot generalize to unseen and potentially unlabeled data called test or target domain [24]. The phenomenon is called the *Domain Adaptation Problem* [153] and current machine learning research tackles the problem by offering a set of solutions characterized by the above examples. Fig. 1.1 shows the Domain Adaptation problem based on the benchmark datasets Office-31 [40] and VisDa [121].

The technique of *Transfer Learning* [39], [95] reflects the above examples by assuming a change in the distribution or task type when switching from training to test environments without any time constraints. Such Transfer Learning techniques help a machine learning model just introduced on the target data by providing appropriate adaptation capacities. While the definition is general, the examples above show numerous constraints in addition to the assumptions of Transfer Learning, such as

Figure 1.1: Samples from Domain Adaptation benchmark datasets. Bikes from Office-31 [40] and planes from VisDa [121]. Every sample represents a different domain. Best viewed on computer display.

the degree of difference in data distribution or task. Hence, current methods commit themselves to a particular set of constraints. Recently, this has led to many subareas in Transfer Learning research. Domain Adaptation is such a category and is particularly interesting in the scope of this work and solely focuses on the adaptation of different data distributions.

The Domain Adaptation theory [36] is applied throughout the thesis to characterize the Domain Adaptation contributions. The theory introduced the term domain separability, which can be roughly described as the probability of confusing two given domains. Domain separability enables a straightforward geometric intuition of two dense but separated clusters representing a domain, respectively. The degree of overlap characterizes the domain separability and leads to domain confusion. While the intuition is reasonably straightforward, competitive Domain Adaptation techniques operate in a Reproducing Kernel Hilbert Space [79], [133], minimizing the Maximum Mean Discrepancy (MMD) and further restrict themselves to a set of mathematical constraints such as kernel width or regularization trade-offs. While the MMD is theoretically sound in defining source and target differences, we lose the geometric intuition from above. The intended performance gain by adapting via MMD is not productive to the required computational time. However, we can access the problem solely from the above intuition based on domain separability in a feature space. This enables us to define a geometric view on the problem and gives a set of geometric operations at hand to execute Domain Adaptation, leading to the first research question.

RQ1: Can we perform Domain Adaptation with geometric projectors?

In chapter 3, I provide a Domain adaptation solution called Geometric Domain Adaptation based on such geometric projectors to obtain a target-adapted source domain invariant to its inherent geometric distortions. The projector is based on the singular vectors of both domains, and the difference in domain spectra characterizes the similarity of the adapted source to target data. The approach is competitive to the above MMD approaches while operating in the original feature space and is hyperparameter-free.

Another stream of work discusses the use of PCA-based subspace projectors for Domain adaptation [57], [69] with the intend to find a common subspace or project

the domain data into the remaining subspace. The idea is appealing since we are roughly able to maintain the above intuition in the subspace, and all advantages of subspace learning [94] such as noise reduction or low-rank representations are present. Subspace Domain Adaptation is assumed superior to algorithms operating in the original space as aboves [113]. These methods adopt the (subspace) projectors based on the eigendecomposition. However, the resulting subspaces are not bounded in terms of domain differences as provided in chapter 3, motivating the next research question.

RQ2: Can subspace projectors yield a bounded subspace for Domain Adaptation?

In chapter 4, I have rewritten the least-squares optimization problem as subspace oriented Domain Adaptation problem and found a solution similar as in chapter 3. The solution is obtained in closed form and proves that domain data must lie in the same subspace, which is not provided by competitive approaches [57], [67]. The solution is called Subspace Override (SO) and requires two complete singular value decompositions. We implemented the Nyström approximation to speed up computational time, making it the fastest non-deep learning-based Domain Adaptation approach. The SO algorithm can bound the subspace data based on the difference of subspace spectra providing a lower domain separability as the solution in chapter 3.

Regardless of geometric or subspace projectors, the remaining domain differences are always the difference in the eigenspectra of the domain data. On the other hand, moment matching [78], [111] is a task in deep Domain Adaptation aligning one or more statistical moments in the latent feature spaces of deep networks. Combining the finding in the just discussed chapters and the moment matching technique, we formulate the next research question.

RQ3: Can spectral decomposition provide a moment matching framework for Domain Adaptation?

I tackled the question in chapter 5 by formulating the difference of singular values as a statistical Domain Adaptation regularization called Spectral Loss. It learns an invariant representation given two domains. The approach is charming because it directly aligns the domain statistics on the principal axes rather than on the feature dimensions while being competitive. Additionally, the numerical range of singular values expresses the degree of data transferability in the latent space [141], [142]. Using the findings, I extended the loss with a relevance shrinkage mechanism to lower domain-specific information in the source data and called it Relevance Spectral Loss.

The above solutions assume an adequate amount of source and target data without time or learn restrictions. However, some situations naturally got time restrictions seen in the above examples of short-time weather changes, e. g., rain or fog. In such situations, detection and reaction time matter. Implementing time constraints with the consequence of requiring linear computational complexity and the availability of only a snapshot of data at learning time, we changed the subject from Domain Adaptation to *Stream Classification*. Data Streams are non-stationary environments because they

are affected by Concept Drift *Concept Drift* [61]. While prototype-based learning can be seen as an online learning procedure, they have per se no mechanism for handling various Concept Drift types, naturally motivating the last research question.

RQ4: Can we extend prototype models to handle Concept Drift?

In chapter 6, I choose the Robust Soft Learning Vector Quantization (RSLVQ), extend it by a Concept Drift detector plus an active prototype insertion strategy and a passive gradient-based adaptation schema. I showed theoretically and empirically that the modifications provide stability and fast adaptation rates, not given by recent stream models. The RSLVQ is a probabilistic prototype approach and was chosen due to a lack of probabilistic models in the field. However, extensions are generalizable to an arbitrary LVQ model already cached by follow-up work [169].

In chapter 7, I will summarize the contributions teased above and discuss their limitations plus the future directions of the research areas *Domain Adaptation* and *Concept Drift*. To summarize, this work expanded the research areas with the following contributions.

- Closed-form Domain Adaptation projectors, providing geometric or subspace invariance by offering bounded domain differences.
- Relevance-based spectral moment-matching with extensive derivative analysis, implementation, and learning procedures.
- Probabilistic prototype models reactive and stable against various Concept Drift types.

I got the chance to present the content of this work at reputable conferences and journals. I am very grateful to be introduced to the involved research communities and to get the chance to discuss my ideas and contributions. The publications that contributed to the dissertation are listed below. All publications within the doctoral period are listed in chapter 7.

CONFERENCE ARTICLES

[122] C. Raab and F.-M. Schleif, "Sparse Transfer Classification for Text Documents," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, F. Trollmann and A.-Y. Turhan, Eds., vol. 11117 LNAI, Springer International Publishing, 2018, pp. 169–181.

[151] C. Raab, M. Heusinger, and F. M. Schleif, "Reactive Soft Prototype Computing for frequent reoccurring Concept Drift," in *27th European Symposium on Artificial Neural Networks, ESANN 2019, Bruges, Belgium, April 24-26, 2019*, 2019, pp. 437–442.

[183] C. Raab, P. Meier, and F.-M. Schleif, "Domain Invariant Representations with Deep Spectral Alignment," in *28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2020, pp. 509–514.

[184] C. Raab and F.-M. Schleif, "Low-Rank Subspace Override for Unsupervised Domain Adaptation," in *KI 2020: Advances in Artificial Intelligence*, U. Schmid, F. Klügl, and D. Wolter, Eds., Cham: Springer International Publishing, 2020, pp. 132–147, **Best Paper Award**.

[198] C. Raab, P. Väth, P. Meier, and F.-M. Schleif, "Bridging Adversarial and Statistical Domain Transfer via Spectral Adaptation Networks," in *Proceedings of the Asian Conference on Computer Vision (ACCV)*, Nov. 2021, pp. 457–473.

JOURNAL ARTICLES

[169] M. Heusinger, C. Raab, and F.-M. Schleif, "Passive concept drift handling via variations of learning vector quantization," *Neural Computing and Applications*, vol. 6, Aug. 2020.

[182] C. Raab, M. Heusinger, and F.-M. Schleif, "Reactive Soft Prototype Computing for Concept Drift Streams," *Neurocomputing*, vol. 416, pp. 340–351, Nov. 2020.

[185] C. Raab and F.-M. Schleif, "Transfer learning extensions for the probabilistic classification vector machine," *Neurocomputing*, vol. 397, pp. 320–330, Jul. 2020.

2 BACKGROUND

Summary: This chapter presents the topics of Transfer Learning and Domain Adaptation plus the underlying concepts of the proposals. The singular value decomposition is a useful tool to decompose a given matrix into its parts, and the Nyström approximation, in our case, provides a method to approximate these parts. For Transfer Learning, we introduce a formal definition together with a concept and term overview. Afterward, we introduce related work and spent some time developing a problem intuition for Domain Adaptation. The intuition will help us to understand the Domain Adaptation theory introduced subsequently. The chapter closes with the introduction of the Robust Soft Learning Vector Quantization.

Publications: This chapter is in parts based on the following publications.

- C. Raab and F.-M. Schleif, 'Transfer Learning extensions for the probabilistic classification vector machine,' Neurocomputing, vol. 397, pp. 320–330, Jul. 2020, doi. 10.1016/j.neucom.2019.09.104.
- C. Raab, M. Heusinger, and F.-M. Schleif, 'Reactive Soft Prototype Computing for Concept Drift Streams,' Neurocomputing, vol. 416, pp. 340–351, Nov. 2020, doi. 10.1016/j.neucom.2019.11.111.
- C. Raab, P. Väth, P. Meier, and F.-M. Schleif, 'Bridging Adversarial and Statistical Domain Transfer via Spectral Adaptation Networks,' in Proceedings of the Asian Conference on Computer Vision (ACCV), 2021, pp. 457–473, doi. 10.1007/978-3-030-69535-4_28

2.1 SINGULAR VALUE DECOMPOSITION

Usage. The singular value decomposition (SVD) plays a vital role in large parts of this thesis. In particular, the Domain Adaptation chapters, namely, the chapters 3, 4 and 5 use the SVD to derive the proposed approaches.
Nomenclature. In the context of this thesis, we define spectral decomposition as an umbrella term for singular- and eigen value decomposition - similar to [17].

The Singular Value Decomposition (SVD) is a useful tool in may areas of science [8] especially in machine learning and in particular for this thesis. In the following, we will shortly introduce the SVD and a selected overview of attributes. Let $\mathbf{M}^{n\times d}$ the set of all m-by-n matrices over the field $\mathbf{M}$ and let $\mathbf{A} \in \mathbf{M}^{n\times d}$. The singular value decomposition is defined [49] as

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{*}, \tag{2.1}$$

where the decomposition contains the unitary matrices $\mathbf{U} \in \mathbf{M}^{n \times n}$ and $\mathbf{V} \in \mathbf{M}^{d \times d}$. We denote $\mathbf{A}^*$ as conjugate transpose of $\mathbf{A}$. Recap that for unitary we get $\mathbf{U}\mathbf{U}^* = \mathbf{V}\mathbf{V}^* = \mathbf{I}$. Further, let $rank(\mathbf{A}) = r$ and $q = min\{n, m\}$, the square diagonal matrix $\mathbf{\Sigma}_q$ is defined by

$$\mathbf{\Sigma}_q = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_q \end{bmatrix}, \tag{2.2}$$

where σ_i is a singular value of $\mathbf{A}$ and $\mathbf{\Sigma}_q$ is sorted in descending order, meaning $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0 = \sigma_{r+1} = \cdots = \sigma_q$. Depending on q the singular value matrix has the following properties:

$$\begin{aligned} \mathbf{\Sigma} &= \mathbf{\Sigma}_q, \text{ if } n = m, \\ \mathbf{\Sigma} &= \begin{bmatrix} \mathbf{\Sigma}_q \\ \mathbf{0} \end{bmatrix} \in \mathbf{M}^{n \times d}, \text{ if } n > d, \\ \mathbf{\Sigma} &= \begin{bmatrix} \mathbf{\Sigma}_q & \mathbf{0} \end{bmatrix} \in \mathbf{M}^{n \times d}, \text{ if } d > n. \end{aligned}$$

where $\mathbf{0}$ is a matrix with zero entries only. The singular values of $\mathbf{A}$ are the square root eigenvalues of $\mathbf{A}\mathbf{A}^*$ or $\mathbf{A}^*\mathbf{A}$, while the $\mathbf{U}$ is the eigenvector of $\mathbf{A}\mathbf{A}^*$ and $\mathbf{V}$ is the eigenvector of $\mathbf{A}^*\mathbf{A}$. The SVD is not necessarily unique [55], for example $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* = (-1)\mathbf{U}\mathbf{\Sigma}(-1)\mathbf{V}^*$.

We use the economy-sized or compact SVD [55] because we do not need a complete SVD for some of the proposed algorithms. See chapter 3 for an example. The economy-sized SVD is defined as

$$\tilde{\mathbf{A}} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^* = \tilde{\mathbf{U}} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix} \tilde{\mathbf{V}}^*, \tag{2.3}$$

while all elements of the diagonal are greater than zero and only the first k singular values with corresponding singular vectors are computed. The error between a $\mathbf{A}$ and $\tilde{\mathbf{A}}$ is bounded by

$$||\mathbf{A} - \tilde{\mathbf{A}}||_F^2 = \sum_{i=k+1}^{r} \sigma_i^2, \tag{2.4}$$

and is basically the squared sum of singular values omitted by the compact SVD. We usually set $k = rank(\mathbf{A})$ in the thesis, when using economy-sized SVD.

The function $\|\cdot\|_F : \mathbf{M}^{n \times d} \to \mathbb{R}$ in Eq. (2.4) is called Frobenius norm and is a matrix norm to measure the *size* of a matrix [49]. The norm is important in many matrix applications and occurs in the chapters 3, 4 and 5. Given $\mathbf{A} \in \mathbf{M}^{n \times m}$ the Frobenius norm is defined as

$$\|\mathbf{A}\|_F = |tr(\mathbf{A}\mathbf{A}^T)|^{\frac{1}{2}} = \left(\sum_{i=1}^{n} \sum_{j=1}^{m} |a_{i,j}|^2 \right)^{\frac{1}{2}}. \tag{2.5}$$

The Frobenius norm is unitary invariant [49], hence for orthogonal $\mathbf{U} \in \mathbf{M}^{n\times n}, \mathbf{V} \in \mathbf{M}^{d\times d}$ the following holds

$$\|\mathbf{A}\|_F = \|\mathbf{UAV}\|_F = \|\mathbf{A}\|_F = \sqrt{\sum_i^d \sigma_i^2}, \tag{2.6}$$

where σ_i^2 are the squared singular values (eigenvalues) of $\mathbf{A}$.

Note that for clarity reasons, we will use the field of real numbers $\mathbb{R}$ with common matrix transpose $\mathbf{A}^T$ in the following. However, the description above also generalizes to other fields, e. g., complex values $\mathbb{C}$. Using the above definitions and $\mathbb{R}$ we can describe the variance of $\mathbf{A}$ in terms of the singular values because

$$Var(\mathbf{A}) = tr(\mathbf{AA}^*) = tr(\mathbf{U\Sigma V}^T\mathbf{V}^T\mathbf{\Sigma}^T\mathbf{U}^T) = tr(\mathbf{U\Sigma\Sigma}^T\mathbf{U}^T) = tr(\mathbf{\Sigma\Sigma}^T) = tr(\mathbf{\Sigma}^2), \tag{2.7}$$

while omitting statistical corrections. As a consequence, the singular values express the variance on the principal components of $\mathbf{A}$, playing a vital role in chapter 5.

2.2 NYSTRÖM APPROXIMATION

Usage. The Nyström approximation is used to provide an approximated solution in chapter 4. In particular, Sec. 4.1 introduces a subspace Domain Adaptation model and approximated variants of the model obtained by the Nyström decomposition are derived in Sec. 4.2 and Sec. 4.3.

The computational complexity of calculating kernels or eigen value decompositions (EVD) scales in the worst case with $\mathcal{O}(n^3)$ where n is the sample size [125]. Therefore, low-rank approximations and dimensionality reductions of data matrices are popular methods to speed up computational processes [11]. The Nyström approximation [11] is a reliable technique to approximate a kernel matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ by a low-rank representation, without computing the eigendecomposition of the whole matrix. In the context of this thesis, we define the term kernel as following:

Definition 2.2.1 (Kernel [18])**.** *A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ mapping two values x, x' from the input space $\mathcal{X}$ to $\mathbb{R}$ is a kernel. The term arises from the theory of integral operators, and in machine learning, kernel functions commonly express a similarity term between two values x, x'. A kernel function is said to be positive semi-definite (PSD) if*

$$\int k(x, x')f(x)f(x')d\mu(x)d\mu(x') \geq 0. \tag{2.8}$$

for all $f \in L_2(\mathcal{X}, \mu)$ where μ is a measure. A real kernel k is symmetric, i. e., $k(x, x') = k(x', x)$, and the corresponding kernel matrix $\mathbf{K} \in \mathbb{R}$ is a Gram matrix with $K_{i,j} = k(x_i, x_j)$. If a symmetric matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ satisfies $\mathbf{v}^T\mathbf{Kv} \geq 0$ for all vectors $\mathbf{v} \in \mathbb{R}^n$ the matrix is called positive semi-definite.

If we use the term covariance function and kernel synonymously we usually refer to a linear kernel $k(\mathbf{x}_i, \mathbf{x}_j^T) = \mathbf{x}_i \mathbf{x}_j^T : \mathbf{x}_i, \mathbf{x}_j \in \mathbf{X} \subseteq \mathbb{R}$ *with* $\mathbb{E}(\mathbf{X}) = 0$*. Further, all used kernel in this thesis are PSD.*

If definition 2.2.1 holds, $(\mathcal{X}, \mu)$ is a finite measure space and kernel $k(\cdot)$ is PSD, then $k(\cdot)$ fulfils the Mercer Theorem [18]. By the Mercer theorem, the eigenvalues of PSD kernels $k(\mathbf{x}, \mathbf{x}')$ are absolutely summable. Further, k can be expanded by orthonormal eigenfunctions f_i and non-negative eigenvalues λ_i with $\lambda_1 \geq \lambda_2 \geq \cdots \geq 0$ in the form

$$k(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i f_i(\mathbf{x}) f_i(\mathbf{x}'). \tag{2.9}$$

The eigenfunctions and eigenvalues of a kernel are defined as solutions of the integral equation

$$\int k(\mathbf{x}', \mathbf{x}) f_i(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} = \lambda_i f_i(\mathbf{x}'), \tag{2.10}$$

where $p(\mathbf{x})$ is a probability density over the input space. This integral can be approximated based on the Nyström technique by an i.i.d. sample $\{\mathbf{x}_k\}_{k=1}^{s}$ of size s from $p(\mathbf{x})$

$$\frac{1}{s} \sum_{k=1}^{s} k(\mathbf{x}', \mathbf{x}_k) f_i(\mathbf{x}_k) \approx \lambda_i f_i(\mathbf{x}'). \tag{2.11}$$

Using this approximation we denote with $\mathbf{K}^{(s)}$ the corresponding $s \times s$ Gram sub-matrix and get the corresponding matrix eigenproblem equation as

$$\frac{1}{s} \mathbf{K}^{(s)} \mathbf{U}^{(s)} = \mathbf{U}^{(s)} \mathbf{\Lambda}^{(s)}, \tag{2.12}$$

with $\mathbf{U}^{(s)} \in \mathbb{R}^{s \times s}$ is column orthonormal and $\mathbf{\Lambda}^{(s)}$ is a diagonal matrix. Now we can derive the approximations for the eigenfunctions and eigenvalues of the kernel k

$$\boldsymbol{\lambda_i} \approx \frac{\boldsymbol{\lambda}_i^{(s)} \cdot n}{s}, \quad f_i(\mathbf{x}') \approx \frac{\sqrt{s/n}}{\boldsymbol{\lambda}_i^{(s)}} \mathbf{k}_{x'} \mathbf{u}_i^{(s)}, \tag{2.13}$$

where $\mathbf{u}_i^{(s)}$ is the ith column of $\mathbf{U}^{(s)}$. Thus, we can approximate f_i at an arbitrary point $\mathbf{x}'$ as long as we know the vector $\mathbf{k}_{x'} = (k(\mathbf{x}_1, \mathbf{x}'), ..., k(\mathbf{x}_s, \mathbf{x}'))$. For a given $n \times n$ Gram matrix $\mathbf{K}$ one may randomly chose s rows and s columns. The corresponding indices are called landmarks and should be chosen such that the data distribution is sufficiently covered. Strategies how to chose the landmarks have recently been addressed in [38], [41], [51], [58]. The approximation is exact if the sample size is equal to the rank of the original matrix and the rows of the sample matrix are linear independent.

Nyström Matrix Form

The technique just reviewed can be simplified by rewriting it in matrix form [88], where scaling factors like in Eq. (2.13) are neglected. We will use the matrix formulation throughout the remaining paper. Again, given a Gram matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$, it can be decomposed to

$$\mathbf{K} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}, \tag{2.14}$$

with $\mathbf{A} \in \mathbb{R}^{s \times s}$, $\mathbf{B} \in \mathbb{R}^{s \times (n-s)}$, $\mathbf{C} \in \mathbb{R}^{(n-s) \times s}$ and $\mathbf{D} \in \mathbb{R}^{(n-s) \times (n-s)}$. The submatrix $\mathbf{A}$ is called the landmark matrix containing s randomly chosen rows and columns from $\mathbf{K}$ and has the EVD $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$ as in Eq. (2.12), where eigenvectors are $\mathbf{U} \in \mathbb{R}^{s \times s}$ and eigenvalues are on the diagonal of $\mathbf{\Lambda} \in \mathbb{R}^{s \times s}$. The remaining approximated eigenvectors $\hat{\mathbf{U}}$ of $\mathbf{K}$, with corresponding parts $\mathbf{C}$ or $\mathbf{B}^T$, are obtained by the Nyström method with $\hat{\mathbf{U}}\mathbf{\Lambda} = \mathbf{C}\mathbf{U}$ as in Eq. (2.13). Combining $\mathbf{U}$ and $\hat{\mathbf{U}}$ the *full* approximated eigenvectors of $\mathbf{K}$ are

$$\tilde{\mathbf{U}} = \begin{bmatrix} \mathbf{U} \\ \hat{\mathbf{U}} \end{bmatrix} = \begin{bmatrix} \mathbf{U} \\ \mathbf{C}\mathbf{U}\mathbf{\Lambda}^{-1} \end{bmatrix} \in \mathbb{R}^{n \times s}. \tag{2.15}$$

The matrix of eigenvectors of $\mathbf{K}$ can be inverted by computing (Note $\mathbf{C} = \mathbf{B}^T$):

$$\tilde{\mathbf{V}} = \begin{bmatrix} \mathbf{U}^{-1} & \mathbf{\Lambda}^{-1}\mathbf{U}^{-1}\mathbf{B} \end{bmatrix}. \tag{2.16}$$

Combining Eq. (2.15), Eq. (2.16) and $\mathbf{\Lambda}$, the matrix $\mathbf{K}$ is approximated by

$$\tilde{\mathbf{K}} = \tilde{\mathbf{U}}\mathbf{\Lambda}\tilde{\mathbf{V}} = \begin{bmatrix} \mathbf{U} \\ \mathbf{C}\mathbf{U}\mathbf{\Lambda}^{-1} \end{bmatrix} \mathbf{\Lambda} \begin{bmatrix} \mathbf{U}^{-1} & \mathbf{\Lambda}^{-1}\mathbf{U}^{-1}\mathbf{B} \end{bmatrix}. \tag{2.17}$$

The Nyström approximation error is given by the Frobenius Norm between ground truth and reconstructed matrices, i. e., $error_{ny} = \left\| \tilde{\mathbf{K}} - \mathbf{K} \right\|_F$. Note that the submatrix $\mathbf{D}$ in (2.14) is not required for the Nyström approximation.

Kernel Approximation

The Nyström approximation [11] speeds up kernel computations because kernel evaluation must not be done over all points, but only a fraction of it. Let $\mathbf{K} \in \mathbb{R}^{n \times n}$ be a Mercer kernel matrix with decomposition as in Eq. (2.14). Again we pick s samples with $s \ll n$, leading to $\mathbf{A}$ as defined before. An approximated kernel is constructed by combining Eq. (2.17) and (2.14).

$$\tilde{\mathbf{K}} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{C}\mathbf{A}^{-1}\mathbf{B} \end{bmatrix} = \begin{bmatrix} \mathbf{A} \\ \mathbf{C} \end{bmatrix} \mathbf{A}^{-1} \begin{bmatrix} \mathbf{A}\mathbf{B} \end{bmatrix} = \mathbf{K}_{n,s}\mathbf{K}_{s,s}^{-1}\mathbf{K}_{s,n}, \tag{2.18}$$

with $\mathbf{K}_{n,s}$ being a sub-matrix of $\mathbf{K}$ incorporating all n rows and s landmark columns and $\mathbf{K}_{s,s}$, as landmarks matrix. Based on the definition of symmetric kernel matrices, $\mathbf{K}_{s,n} = \mathbf{K}_{n,s}^T$ is valid and, therefore, only $\mathbf{K}_{n,s}$ and $\mathbf{K}_{s,s}$ must be computed.

Generalized Matrix Approximation

Despite its initial restriction to kernel matrices, recent research [88] expanded the Nyström technique to approximate a SVD. Nyström-SVD generalizes the concept of matrix decomposition with the consequence that respective matrices must not be square. Let $\mathbf{G} \in \mathbb{R}^{n \times d}$ be a rectangular matrix with decomposition as in Eq. (2.14). The SVD of the landmark matrix is given by $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ where $\mathbf{U}$ are left and $\mathbf{V}$ are right singular vectors. $\mathbf{\Sigma}$ are positive singular values. The decomposition matrices have the same size as above. Similarly to the EVD in section 2.2 the left and right singular vectors for the non-symmetric part $\mathbf{C}$ and $\mathbf{B}$ are obtained via Nyström techniques [88] and are defined as $\hat{\mathbf{U}} = \mathbf{C}\mathbf{V}\mathbf{\Sigma}^{-1}$ and $\hat{\mathbf{V}} = \mathbf{B}^T\mathbf{U}\mathbf{\Sigma}^{-1}$ respectively. Applying the same principals as for Nyström-EVD, $\mathbf{G}$ is approximated by

$$\tilde{\mathbf{G}} = \tilde{\mathbf{U}}\mathbf{\Sigma}\tilde{\mathbf{V}}^T = \begin{bmatrix} \mathbf{U} \\ \hat{\mathbf{U}} \end{bmatrix} \mathbf{\Sigma} \begin{bmatrix} \mathbf{V} & \hat{\mathbf{V}} \end{bmatrix} = \begin{bmatrix} \mathbf{U} \\ \mathbf{C}\mathbf{V}\mathbf{\Sigma}^{-1} \end{bmatrix} \mathbf{\Sigma} \begin{bmatrix} \mathbf{V} & \mathbf{\Sigma}^{-1}\mathbf{U}^T\mathbf{B} \end{bmatrix}. \tag{2.19}$$

Note that for non-gram matrices like $\mathbf{G}$, it is $\mathbf{C} \neq \mathbf{B}^T$. The general matrix approximation in Eq. (2.19), described in this section, is used in Sec. 4.2.

2.3 TRANSFER LEARNING AND DOMAIN ADAPTATION

Usage. Transfer Learning and Domain Adaptation is one of the two main building blocks and are used in most chapters of this thesis. In particular, the chapters 3, 4, and 5 contributing to the field. Note that methods discussed in this chapter are related to the methods in chapter 3 and 4. For chapter 5, we introduce the related work in the sections itself.

For Transfer Learning [39], [95], we consider a domain $\mathcal{D} = \{\mathbb{R}^d_{\mathcal{D}}, p(\mathcal{D}), f_{\mathcal{D}}\}$, where $\mathbb{R}^d_{\mathcal{D}}$ is a d-dimensional domain specific feature space with samples distributed i.i.d by $p(\mathcal{D})$. The function $f_{\mathcal{D}}$ is the domain specific task $f_{\mathcal{D}} = p(y|x)$ with task space $\mathcal{Y}_{\mathcal{D}}$. In the scope of this work the task is a classification problem with $\mathcal{Y} = \{0, \ldots, c\}$ as the label space. In practice, we use the dataset $\mathbf{D}_{\mathcal{D}} = \{\mathbf{X}_{\mathcal{D}}, \mathbf{Y}_{\mathcal{D}}\} = \{\mathbf{x}_i, y_i\}_{i=1}^n \overset{i.i.d.}{\sim} p(\mathcal{D})$ with $\mathbf{x}_i \in \mathbb{R}^d_{\mathcal{D}}$ and $y_i \in \mathcal{Y}_{\mathcal{D}}$ and use it synonymously for domain. In the scope of this work we define two domains: source and target with above mentioned compositions and datasets. With the above notation, we are able to define Transfer Learning:

Definition 2.3.1 (Transfer Learning [95]). *Given a source domain $\mathcal{D}_s = \{\mathbb{R}^d_{\mathcal{S}}, p(\mathcal{S}), f_{\mathcal{S}}\}$ and a target domain $\mathcal{D}_t = \{\mathbb{R}^d_{\mathcal{T}}, p(\mathcal{T}), f_{\mathcal{T}}\}$, Transfer Learning is the process of improving a target predictive functions $h_t(x)$ by using related information form $\mathcal{D}_s$, while one or more components within $\mathcal{D}_s$ are unequal to corresponding $\mathcal{D}_t$ components.*

From the above definition, we can define subcategories. Not all are covered in this thesis. However, it sharpens the view on the topic and delimits the terms necessary to ask and answer research questions precisely. Transfer Learning is roughly divided into following subcategories:

- Heterogenous Transfer Learning [95], i. e., $\mathbb{R}^d_{\mathcal{S}} \neq \mathbb{R}^d_{\mathcal{T}}$ and $p(\mathcal{S}) \neq p(\mathcal{T})$ and/or $f_{\mathcal{S}} \neq f_{\mathcal{T}}$ - explicitly not cover in this work.
- Homogeneous Transfer Learning [95], i. e., $\mathbb{R}^d_{\mathcal{S}} = \mathbb{R}^d_{\mathcal{T}}$ and $p(\mathcal{S}) \neq p(\mathcal{T})$ and/or $f_{\mathcal{S}} \neq f_{\mathcal{T}}$ - covered in this work.
- Domain Adaptation (DA), or Transductive Transfer Learning [153], [197], is currently the favored term for Homogeneous Transfer Learning in the literature. Covered in this thesis in chapter 3, 4 and 5. Note that Domain Invariant [111] in this context is synonymously used if the number of domains is two.
- Concept drift [61], in chapter 6, can be interpreted as a time-dependent and memory-restricted version of Homogeneous Transfer Learning.
- Unsupervised Transfer Learning [197], i. e., $\mathcal{Y}_s \neq \mathcal{Y}_t$ mismatch in label space and further: $\mathbb{R}^d_{\mathcal{S}} = \mathbb{R}^d_{\mathcal{T}}$ and/or $p(\mathcal{S}) \neq p(\mathcal{T})$ and/or $f_{\mathcal{S}} \neq f_{\mathcal{T}}$ - explicitly not cover in this work. Variants of this subcategory currently emerging via Partial DA [114], Open Set DA [98] or Universal DA [161].
- Multi-Domain Adaptation [144], [150], [156] with more than one source or target domain are explicitly not covered in this work.

We restrict ourselves and the outcome of the Domain Adaptation part of this thesis to the following conventions of Domain Adaptation: (i) we *always assume labeled source data and unlabeled target data at adaptation time*. The adaptation time is the execution time of the proposed algorithms. (ii) We call methods without deep networks shallow Domain Adaptation and, of course, deep Domain Adaptation with these nets. (iii) The term Domain Adaptation is used instead of Homogeneous Transfer Learning.

Related Work

Scope. This sections describes related work of shallow Domain Adaptation methods related to chapter 3 and 4. For related deep approaches see Sec. 5.1.

Instance Transfer Methods [26], [137], [155] try to align the marginal distribution by selecting some source and target data based on a relevance or divergence measure and directly re-weights data for training. It seems that this type of algorithm works best when the conditional probability is the same in the source and the target domain and only aligns marginal distribution divergences [95]. Recently, some extensions to deep instance transfer are made [115].

Feature Adaptation techniques are trying to find a common latent subspace for source and target domain to reduce distribution differences, such that the underlying structure of the data is preserved in the subspace. A baseline approach for feature adaptation is TCA [45]. TCA finds a suitable subspace transformation called transfer components via minimizing the Maximum Mean Discrepancy (MMD) in the Reproducing Kernel

Hilbert Space (RKHS). JDA [59] also considers MMD but incorporates class-dependent pseudo labels. These works considered a subspace projection based on a combined eigendecomposition for both domains, which fails to include domain-specific attributes into the subspace. The Joint Geometrical Subspace Alignment (JGSA) [113] tackled this issue by searching MMD based subspaces for the domains individually with a focus on minimizing MMD, intra-, and inter-class variance simultaneously. The work in [145] is similar to JGSA regarding adaptation mechanism. However, within the adaptation process, the pseudo labeling is enhanced by an instance transfer selection schema.

The Geodesic Flow Kernel (GFK) [47] models two domains on a Geodesic manifold in a suitable subspace and aligns the domains by finding the shortest path on the manifold. The just described feature adaptation methods first adapt the data and train a classifier afterward. The MEDA approach [133] simultaneously trains a given classifier with a Geodesic-based adaptation mechanism similar to JGSA and GFK and is therefore able to train hyperparameters within an adaptation iteration. However, these methods rely on kernels, are therefore restricted to kernel choice, and are computationally demanding. The proposed work relies on original space and uses only a snapshot of data for computational efficiency.

Least-Squares Adaptation is closely related to us and is different from the Feature Adaptation by explicitly defining variants of the Least-Squares (LS) optimization rather than relying on some other type of optimization. Both try to find a suitable feature transformation or subspace projectors to align both domains. For LS-Adaptation, the SA [57] computes a target subspace representation by directly modifying the correlation matrices of both domains. The method in [99] extends the SA approach by using a more efficient low-rank SVD, making it applicable to larger domains. The Landmarks Selection-based Subspace Alignment (LSSA) [69] is a successor of SA and selects only a subset of both domains near to domain borders to align these borders in the subspace explicitly. However, LSSA cannot capture the whole domain characteristic, and in supervised classification problems, the landmark sample is prone to omit class information. Our proposed work considers a Uniform and class-wise sampling strategy to capture the whole domain.

The work of Shao et al. [67] proposed that least-squares approaches, as above, are inappropriate for effective adaptation because the source and target data may not lay in a single subspace. The work in [192] prevents this by projecting both domains into the same common subspace, while a regularization term prevents ill-conditioned subspace projectors. In chapter 4, we explicitly *override* the orthogonal basis of the source domain with the target one. We model the source subspace domain as part of the target subspace, and subspace differences do not exist because both must lie in the same subspace by construction. Further, the projector is not ill-defined if the data is not. All these algorithms require some *unlabeled target data* to be available at training time. These Domain Adaptation solutions cannot be directly used as predictors but instead are wrappers for classification algorithms.

2.3.1 *Problem Intuition*

Overview. This section introduces visual guidance to develop an intuition of a Domain Adaptation problem, necessary to grasp the unneglectable differences between traditional machine learning and Domain Adaptation-based machine learning. (i) a data-driven understanding of the problem showing selected text and image examples of Domain Adaptation datasets to grasp the concept of real-world domain differences. (ii) the problem is visualized via two popular synthetic toy datasets to demonstrate the effect. The toy datasets are again used for the proof of concept in the methodical chapters.

Real-World Domains

The real-world datasets are usually designed to evaluate Domain Adaptation algorithms. The examples shown in this section are borrowed from these real-world Domain Adaptation datasets, which are the image dataset Office-31 [40] and the text dataset Newsgroup [65]. Note that [65] is not the original publisher of Newsgroup, but the first realization of Newsgroup as a Domain Adaptation problem. The details of the datasets are given for Newsgroup in Sec. 3.2.2 and for Office-31 in Sec. 5.4.2.

Object Classification Examples. The task of object classification over varying domains [135] is fairly simple for humans, which are superior to current state-of-the-art methods, easily verified by inspecting the objects presented in Fig. 2.1. For an extensive review, one might inspect a broader range of examples in Appendix 7. The rows separate the object classes, namely bike, and mug, while the columns separate the three domains webcam, digital camera, or amazon product images. These domains are varying in terms of object positioning, lightning, or camera quality. The image differences create the Domain Adaptation problem by altering the underlying distribution of the data, violating the assumption of traditional machine learning that training and validation follow the same distribution.

Learning on webcam (first column, first row) and testing on amazon (third column, first row) will, therefore, results in severe drops in prediction performance [175]. Additionally, preprocessing via popular data augmentations [154], like random rotations or Gaussian noise filters, to improve the generalization capabilities of a model, are still insufficient to bride domain differences [131], [132]. Therefore, explicit alignment of domains must take place in an adaptation process.

Although arbitrary rotations are insufficient for Domain Adaptation, we will see in Sec. 3.1, how geometric projectors explicitly formulated for minimizing domain differences can still yield a solution to the problem.

Text Classification Examples. Text datasets like Reuters [25][1] and Newsgroup[2] contain

1 https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization+collection
2 http://qwone.com/~jason/20Newsgroups/

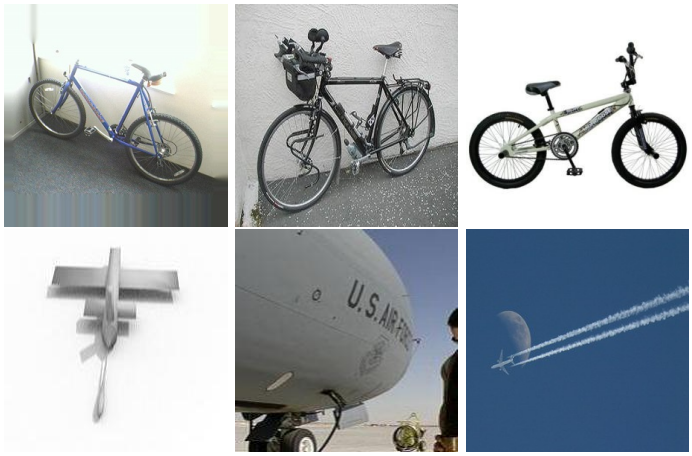
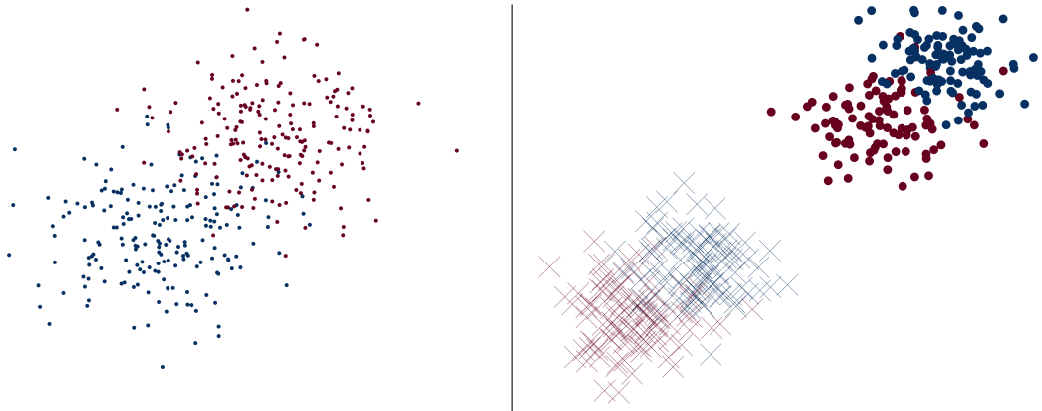


Figure 2.1: Examples from Office-31 [40] datasets in row one and VisDa [121] in row two. Columns show different domains and classes shown per row. Best viewed on computer display.

Example of Newsgroup-Graphics	**Example of Newsgroup-Hockey**
[...] I'm using RenderMan library on my NeXT but there is no documentation about NeXTSTEP version of RenderMan available. I can create very complicated scenes and render them using surface shaders, but I can not bring them to life by applying shadows and reflections. As far as I understand I have to define environmental and shadows maps to produce reflections [...]	[...] Implicitly you are assuming that goals scored against Winnipeg with Selanne on the ice can be blamed on him...Roger, he is a FORWARD. Winnipeg has a lousy defensive record anyway. Let's put it another way. John Cullen's +/- is terrible. What's your excuse for him? That his powerplay points don't count? Neither do Selanne' [...]

Table 2.1: Examples from two Domains from Newsgroup dataset [65].

various subcategories with many very specific terms to describe certain topics. Two snapshots from the Reuters categories *comp.graphics* and *rec.sports.hockey* are presented in Tab. 2.1 show these specific terms. This creates very distinct word distributions, lacking a broader context and therefore leading to models with poor generalization capabilities to other subcategories. Indeed, the classification of unseen subcategories from Reuters and Newsgroup is a Domain Adaptation problem, used to evaluate the proposed algorithms in Sec. 3.2 and Sec. 4.4. The *tf-idf* method is a common baseline representation [50], [79], [128], but extended representations are state of the art [165].



(a) Traditional Problem *(b)* Domain Adaptation Problem

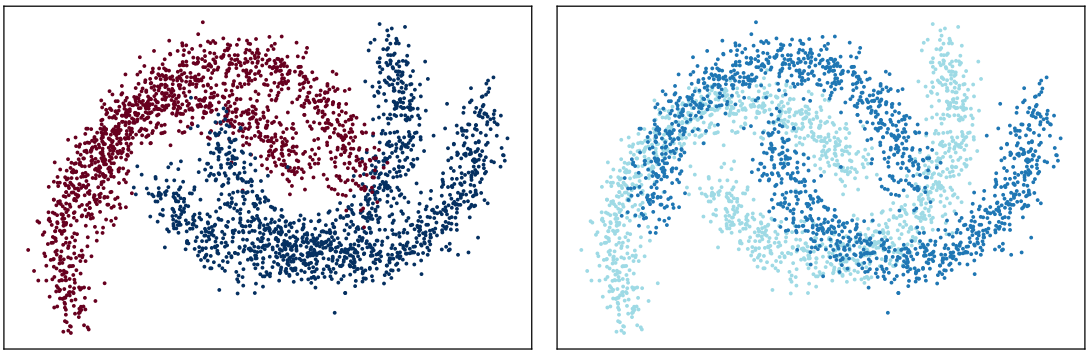
Figure 2.2: Toy example showing a comparison of a traditional classification task with two classes shown in red and blue. Left figure shows a traditional classification task with *one domain* and right side shows a Domain Adaptation classification task with *two domains*, which are indicated as shapes. Domain Adaptation aims to extract common information in one domain to help a model in another domain. Best viewed on computer display.

Synthetic Toy Datasets

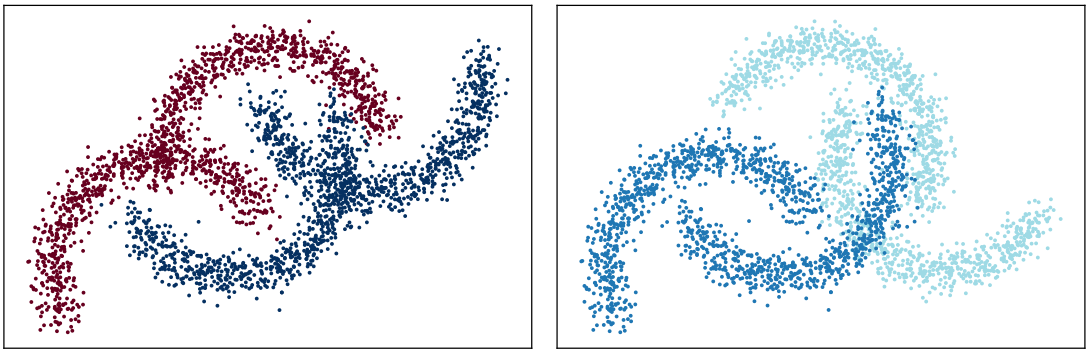
We develop a second level of intuition based on two common synthetic toy datasets to demonstrate the effect of a Domain Adaptation algorithm. The toy datasets are again used in the proof of concept sections of the methodical chapters.

Domain Clusters. The synthetic dataset [156][45] is solely used to demonstrate a concept rather than to evaluate the efficiency of the adaptation mechanics. In our case, we want to visualize the difference between traditional and Domain Adaptation problems. The domain clusters are shown in Fig. 2.2. The left plots in Fig. 2.2a demonstrate two Gaussian clusters without any change between source and target data. Both are inseparable, i. e., low domain separability. The right-hand side plots in Fig. 2.2b have two shifted Gaussian clusters to create a Domain Adaptation problem. A non-adaptive classifier trained on the 'x' data cannot achieve a performance better than random. We use the domain cluster dataset to demonstrate a concept in chapter 3.

Two Moons. First used introduced [44] and populated by [72] and still used for example in [177] with the purpose to evaluate the efficiency of the adaptation process. Fig. 2.3 shows variants of the two moon problem per row, while left columns are colored with class and right columns are colored with domain labels. The moons combine sinus and cosine functions with an additional shift and bias term to form an appropriate Domain Adaptation problem. The first row demonstrates the overlap of domains, while the second row shows a class overlap beyond domain borders. While the first one appears easier, both are equally difficult for Domain Adaptation because classers and domains



(a) Classes of Moon_1 *(b)* Domains of Moon_1



(c) Classes of Moon_2 *(d)* Classes of Moon_2

Figure 2.3: Toy examples of moon shaped functions as two class Domain Adaptation problem [72]. $\text{Moon}_{\#}$ represents a single Domain Adaptation problem. Classes are marked in red/blue and domains are illustrated in variants of blue. Best viewed on computer display.

are fairly well separated. One can use the given structure for the adaptation process. The two moon dataset shown in the first row is used to evaluate the proposed Domain Adaptation methods in chapter 3, 4 and 5.

Domain Adaptation Theory

Usage. The theory of Domain Adaptation will be used during the shallow Domain Adaptation chapters, namely 3 and 4, to explain which parts of the theory are affected by the proposed adaptation procedures. In chapter 4 and 5, we use the theory to bound the expected risk of the proposed model. Finally, we use the theory to explain the limitations of all proposed models. This section gives a brief introduction to it and describes the theory in terms of expected risk. For a detailed description please see [36], [89], [162].

The Domain Adaptation theory was initially populated by Ben-David et al. [36].

The whole purpose of the theory is to characterize the expected or empirical risk of a classifier $\varepsilon(f)$ in terms of the underlying Domain Adaptation problem on the affected source and target domain with their specific distributions $p(\mathcal{S})$ and $p(\mathcal{T})$. As in theorem 2.3.1, the theory assumes source and target data at training or adaptation time, while the source is labeled and the target one is unlabeled. The theory gives the relationship between the risk of the *same* classifier on the source domain $\varepsilon_s(f)$ and the target domain $\varepsilon_t(f)$ by constraining both domains by separability. The term separability is introduced via classification hypotheses and their specific hypothesis class to introduce the term formally. Afterward, we review bounds based on discrepancy measures and functions.

Hypotheses based Separability. Let h, h' be two hypothesis and $h, h' \in \mathcal{H}$ in the same hypothesis space $\mathcal{H}$ with VC dimension d. Further, we call the symmetric difference hypothesis space $\mathcal{H}\Delta\mathcal{H}$ for $\mathcal{H}$ the set of hypotheses obey following conditions

$$g \in \mathcal{H}\Delta\mathcal{H} \quad \text{and} \quad g(\mathbf{x}) = h(\mathbf{x}) \otimes h'(\mathbf{x}) \quad \text{for some} \quad h, h' \in \mathcal{H}, \tag{2.20}$$

where $\otimes$ is the XOR function and, therefore, g is the expression of disagreement of the two hypotheses h, h' and helps to understand the concept of hypothesis separability. Using the $\mathcal{H}\Delta\mathcal{H}$ we can express the notion of separability for two domains by the following measure

$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{S}, \mathcal{T}) = 2sup_{h,h' \in \mathcal{H}} \left| Pr_{\mathbf{x} \sim p(\mathcal{S})} \left[h(\mathbf{x}) \neq h'(\mathbf{x}) \right] - Pr_{\mathbf{x} \sim p(\mathcal{T})} \left[h(\mathbf{x}) \neq h'(\mathbf{x}) \right] \right| \tag{2.21}$$

which is the difference of probability in disagreement by two hypotheses on a sample from source and target domain respectively. If both domains are inseparable the $d_{\mathcal{H}\Delta\mathcal{H}}$ tends to zero and increases for non separable domains. First case is shown in Fig. 2.2a and second on in Fig. 2.2b. In further references, we call the $d_{\mathcal{H}\Delta\mathcal{H}}$ domain separability.

Let lambda be $\lambda = \varepsilon_s(f^*) + \varepsilon_t(f^*)$ be the ideal joint hypothesis for source and target, we are able to define the Domain Adaptation bound in the following.

Theorem 2.3.1 (Hypotheses based Domain Adaptation Bound [36])**.** *Let $\mathcal{U}_s, \mathcal{U}_t$ are unlabeled random i.i.d. samples of size n drawn from $p(\mathcal{S}), p(\mathcal{T})$ respectively. And let $\mathcal{H}$ be a hypothesis space with VC dimension d, then for any $\delta \in (0, 1)$ and probability with at least $1 - \delta$ for every $f, h, h' \in \mathcal{H}$:*

$$\varepsilon_t(f) \leq \varepsilon_s(f) + d_{\mathcal{H}\Delta\mathcal{H}}\left(p(\mathcal{S}), p(\mathcal{T})\right) + \lambda \tag{2.22}$$

$$\leq \varepsilon_s(f) + \tilde{d}_{\mathcal{H}\Delta\mathcal{H}}(U_s, U_t) + \lambda + 4\sqrt{\frac{2d\log(2n) + \log(\frac{2}{\delta})}{n}}, \tag{2.23}$$

where $\tilde{d}_{\mathcal{H}\Delta\mathcal{H}}$ is the empirical expression of $d_{\mathcal{H}\Delta\mathcal{H}}$.

The bound in Eq. (2.22) is also extended to the empirical risk [36]. However, the above is sufficient for our purpose to get a glimpse of what the theory is about. It tells

us that if two existing hypotheses h, h' can separate the domains *and* the optimum of f on joint error is large, we can expect a large error on the target domain even if there is a very low source risk of f. The same applies in reverse. If h, h' are unable to separate, a low λ and the risk on the source is low, we can expect a low target risk. For an introduction to the Vapnik–Chervonenkis (VC) Dimension please see [34, p.238].

Discrepancy based Separability. The Maximum Mean Discrepancy MMD [48] is a useful measure to quantify distribution differences. Let $f : \mathcal{X} \to \mathbb{R}$ be a function within class $\mathcal{F}$, a RKHS $\mathcal{H}_k$ with $\|f\|_{\mathcal{H}_k} \leq 1$ and two probability Borel measures p, q, the MMD is defined as [89]:

$$d_k(p,q) = \sup_{f \in \mathcal{F}} \left[\mathbb{E}_{x \sim p}[f(x)] - \mathbb{E}_{y \sim q}[f(y)] \right] = \left\| \mu_{x \sim p}\left[\phi(x)\right] - \mu_{y \sim q}\left[\phi(y)\right] \right\|_{\mathcal{H}_k}, \tag{2.24}$$

where $\mu(x) = \mathbb{E}_p[\phi(x)]$ and $\phi(x) = k(x, \cdot)$ is a valid kernel function as in definition 2.2.1. The empirical estimate is straightforward to compute using the unbiased squared MMD defined as [48]:

$$\hat{d}_k^2(\mathcal{U}_s, \mathcal{U}_t) = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} k(\mathbf{x}_i^s, \mathbf{x}_j^s) + \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} k(\mathbf{x}_i^t, \mathbf{x}_j^t) - \frac{2}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} k(\mathbf{x}_i^s, \mathbf{x}_j^t), \tag{2.25}$$

where $\mathbf{x}_i^s, \mathbf{x}_j^s \in \mathcal{U}_s$ and $\mathbf{x}_i^t, \mathbf{x}_j^t \in \mathcal{U}_t$ with sample size n. For an introduction to RKHS please see [34, p.167]. Recent work [89] showed that the MMD yields an upper bound for the expected target error in a similar fashion as in theorem 2.3.1. This allows us to describe domain separability via the MMD. The following theorem defines the properties of the bound and will be used in Sec. 4.3:

Theorem 2.3.2 (Discrepancy based Domain Adaptation Bound [89])**.** *Let $\mathcal{U}_s, \mathcal{U}_t$ are unlabeled random i.i.d. samples of size n drawn from $p(\mathcal{S}), p(\mathcal{T})$ respectively. And let $\|f\|_{\mathcal{H}_k}, \|h\|_{\mathcal{H}_k} \leq 1$ in an RKHS $\mathcal{H}_k$ then for any $\delta \in (0,1)$ and probability with at least $1 - \delta$ for every $f, h, h' \in \mathcal{H}$:*

$$\varepsilon_t(f) \leq \varepsilon_s(f) + d_k(p(\mathcal{S}), p(\mathcal{T})) + \lambda. \tag{2.26}$$

$$\leq \varepsilon_s(f) + \hat{d}_k(\mathcal{U}_s, \mathcal{U}_t) + \frac{2}{n} \mathbb{E}_{p(\mathcal{S})} \left[\sqrt{tr(\mathbf{K}_s)} \right] + \frac{2}{n} \mathbb{E}_{p(\mathcal{T})} \left[\sqrt{tr(\mathbf{K}_t)} \right] \tag{2.27}$$

$$+ 2\sqrt{\frac{log(\frac{2}{\delta})}{2n}} + \lambda. \tag{2.28}$$

where $d_k(p(\mathcal{S}), p(\mathcal{T}))$ is the MMD and $\hat{d}_k(\mathcal{U}_s, \mathcal{U}_t)$ is the empirical estimate of MMD. K_s and K_t are the kernels of the samples respectively.

Function based Separability. The work of Zhao et al. [162] extends the above theory [36] shown in theorem 2.3.1. The work is particular useful in this thesis because it allows us

to describe the limitation of Domain Adaptation algorithms by the domain labeling functions. Let

$$min\{\mathbb{E}_{p(\mathcal{S})}[|f_s - f_t|], \mathbb{E}_{p(\mathcal{T})}[|f_s - f_t|]\} \tag{2.29}$$

be the expected divergence between the optimal labeling function of source f_s and target f_t w.r.t the source and target marginal distributions. Define the following

$$d_{\tilde{\mathcal{H}}}(p(\mathcal{S}), p(\mathcal{T})) = \sup_{\tilde{h} \in \tilde{\mathcal{H}}} |Pr_{p(\mathcal{S})}(\tilde{h} = 1) - Pr_{p(\mathcal{T})}(\tilde{h} = 1)| \tag{2.30}$$

as the disagreement of hypothesis $\tilde{h}$ on the source and target distribution given

$$\tilde{\mathcal{H}} = \{sgn(|h(\mathbf{x}) - h'(\mathbf{x})| - z) \mid h, h' \in \mathcal{H}, 0 \leq z \leq 1\}, \tag{2.31}$$

where $\mathcal{H}$ is again a hypothesis class with finite VC dimension d. Note that Eq. (2.21) is a special case of Eq. (2.30) with $z = 0$. The extension provides a new Domain Adaptation bound:

Theorem 2.3.3 (Function based Domain Adaptation Bound [162]). *Let $\mathcal{S} = \langle p(\mathcal{S}), f_s \rangle$ and $\mathcal{T} = \langle p(\mathcal{T}), f_t \rangle$ be source and target domain, respectively. And let for any function class $\mathcal{H} \subseteq [0,1]^{\mathcal{X}}$ with $f, h \in \mathcal{H}$ then the following holds*

$$\varepsilon_t(f) \leq \varepsilon_s(f) + d_{\tilde{\mathcal{H}}}(p(\mathcal{S}), p(\mathcal{T})) + min\{\mathbb{E}_{p(\mathcal{S})}[|f_s - f_t|], \mathbb{E}_{p(\mathcal{T})}[|f_s - f_t|]\}. \tag{2.32}$$

The interpretation of the bound is similar to the one above. Eq. (2.32) enables us to describe a data-driven divergence between the labeling functions of two domains rather than a hypothesis perspective. This enables us to describe the limitations as follows. If a Domain Adaptation method has no adaptation mechanism for minimizing the labeling functions, we should consider data with the same labeling functions for successful adaptation. Note that Eq. (2.21) and Eq. (2.30) are conveniently referred to as the difference in marginal distribution [162] and play a vital role in Domain Adaptation very well observable in related work.

2.4 ROBUST SOFT LEARNING VECTOR QUANTIZATION

Usage. The Robust Soft Learning Vector Quantitation (RSLVQ) is the main classification model in chapter 6, where it is extended to deal with various types of Concept Drift streams.

The family of LVQ algorithms is a learning scheme of prototypes representing class regions [66]. The prototypes have a geometric representation and provide a simple interpretation and classification scheme. The prototypes are attracted and repelled within the learning process depending on the class of given samples minimizing the error function. Note that the first version of LVQ is a heuristic-based approach. However,

currently the Generalized-LVQ (GLVQ) [5], or probabilistic approaches like RSLVQ [12] are state of the art with differentiable error functions [110].

The Robust Soft Learning Vector Quantization [12] is a probabilistic prototype-based classification algorithm and capable of online learning. Let $\mathbf{D} = \{(\mathbf{x_i}, y_i) \in \mathbb{R}^d \times \{1, \ldots, C\}\}_{i=1}^n$ be a labeled dataset as classification task. The RSLVQ assumes that $\mathbf{D}$ can be represented as class-dependent Gaussian mixture model and approximates this mixture by a set of m prototypes $\Theta = \{(\boldsymbol{\theta}_j, y_j) \in \mathbb{R}^d \times \{1, \ldots, C\}\}_{j=1}^m$, where each prototype represents a multi-variate Gaussian model, i. e., $\mathcal{N}(\boldsymbol{\theta}_j, \Sigma)$ with $\Sigma = \mathbf{I}\sigma^2$ and $\mathbf{I}$ as identity matrix and variance σ^2. Further, $\boldsymbol{\theta}_j$ is a d-dimensional prototype representing the mean of a Gaussian mixture component with a variance of σ^2, which is assumed to be the same for every component. The goal of RSLVQ algorithms is to learn prototypes representing the class-dependent distribution, i. e., a corresponding class sample $\mathbf{x}_i$ should be mapped to the correct class or Gaussian mixture based on the highest probability.

The RSLVQ [12] algorithm minimizes the objective function

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} ls(\mathbf{x}_i, y_i|\Theta) \quad \text{with} \quad ls(\mathbf{x}_i, y_i|\Theta) = \frac{p(\mathbf{x}_i, \bar{y}_i|\Theta)}{p(\mathbf{x}_i|\Theta)}. \tag{2.33}$$

Where $p(\mathbf{x}_i, \bar{y}_i|\Theta)$ is the probability density function that $\mathbf{x}_i$ is generated by the mixture model of any of the different classes and $p(\mathbf{x}_i|\Theta)$ is the overall probability density function of $\mathbf{x}_i$ given Θ. In other words, $\bar{y}_i$ is every label which is not the ground truth label y_i of x_i. At time step t, these probabilities are computed by

$$ls_t(\mathbf{x}_t, y_t|\Theta) = \frac{p(\mathbf{x}_t, \bar{y}_t|\Theta)}{p(\mathbf{x}_t|\Theta)} = \sum_{j:c_j \neq y_t} p(j|\mathbf{x}_t). \tag{2.34}$$

Where j is the j-th prototype in Θ and c_j is the corresponding label. $p(j|\mathbf{x})$ is the probability that $\mathbf{x}$ is generated by the component j with

$$p(j|\mathbf{x}) = \frac{p(\mathbf{x}|j)p(j)}{p(\mathbf{x})} = \frac{exp\Big(-\frac{\|\mathbf{x}-\boldsymbol{\theta}_j^2\|}{2\sigma^2}\Big)}{\sum_{k=1}^{m} exp\Big(-\frac{\|\mathbf{x}-\boldsymbol{\theta}_k^2\|}{2\sigma^2}\Big)}. \tag{2.35}$$

Based on this, the gradient [12] for the prototype update step at time t is computed by

$$g_t = \begin{cases} -p(j|\mathbf{x}_t) ls_t(\mathbf{x}_t - \boldsymbol{\theta}_t), & if\ c_j = y_t, \\ p(j|\mathbf{x}_t)(1 - ls_t)(\mathbf{x}_t - \boldsymbol{\theta}_t), & if\ c_j \neq y_t. \end{cases} \tag{2.36}$$

Note that ls_t abbreviated for Eq. (2.34). The normalization term in Eq. (2.33) is not computed at the update step and, therefore, the RSLVQ is feasible for potentially infinite streams. The RSLVQ predicts a given sample point $\mathbf{x}$ by selecting the nearest prototype

$\boldsymbol{\theta}_q$ according to the Gaussian kernel

$$q = \min_i exp\Big(- \frac{\|\mathbf{x} - \boldsymbol{\theta}^2\|}{2\sigma^2} \Big), \tag{2.37}$$

and assigning the corresponding label of $\boldsymbol{\theta}_q$ to $\mathbf{x}$. The σ is the width of the Gaussian kernel and together with the number of prototypes, are the only tuneable parameters in the RSLVQ. For a more comprehensive derivation of RSLVQ with momentum SGD, see [12], [168].

3 SHALLOW GEOMETRIC DOMAIN ADAPTATION

Summary: Common shallow Domain Adaptation algorithms are based on a simple intuition to solve the domain differences but end up relying on complex computations to do so. They introduce further assumptions on the data, making the solutions restricted to an arbitrary set of mathematical constraints, such as kernel or subspace choice or other sets of hyperparameters. That introduces two new problems: The need for computational exhausting hyperparameter search and the methods are restricted to this set of mathematical restrictions necessary to make the solution work.

This chapter builds a Domain Adaptation algorithm based on our geometric intuition, which employs the same simple solution principal but is hyperparameter-free without the need to apply further mathematical restrictions on the data. While the geometric intuition also constraints the data to a certain extent, the derived solution is more straightforward to understand, more comfortable applying, and very competitive on the benchmarks.

Publications: This chapter is based on the following publications.

- C. Raab and F.-M. Schleif, 'Sparse Transfer Classification for Text Documents,' in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 11117 LNAI, F. Trollmann and A.-Y. Turhan, Eds. Springer International Publishing, 2018, pp. 169–181, doi. 10.1007/978-3-030-00111-7_15.
- C. Raab and F.-M. Schleif, 'Transfer Learning extensions for the probabilistic classification vector machine,' Neurocomputing, vol. 397, pp. 320–330, Jul. 2020, doi. 10.1016/j.neucom.2019.09.104.

Source Code: https://github.com/ChristophRaab/thesis/tree/main/sda/.

A common way to tackle the Domain Adaptation problem used by shallow Domain Adaptation algorithms [45], [134] is aligning domains in kernel space. It is assumed that aligning both kernel matrices is implicitly aligning the domain distributions. Related work [57], [67] extends the assumption by the claim that aligning feature matrices of both domains also tackling the problem of distribution differences. We assume the reader is familiar with the concept of Domain Adaptation from Sec. 2.3. Following the intuition and the principle of domain separability also introduced in Sec. 2.3, we simplify the Domain Adaptation problem as separated Gaussian clusters in the same feature space. Thereby, we assume with similar class conditional distributions but different marginal distributions.

An intuitive answer to the problem just introduced would be to use a sequence of rotations and translations to make the domains similar. In relation to the Domain Adaptation theory [36], it means we first learn a suitable domain representation $\mathcal{S}$ and $\mathcal{T}$ and obtain a hypothesis *h after* the adaptation process. Hence, the proposals in this chapter strive to minimize domain separability. See Sec. 2.3.1 for an introduction to the theory.

The idea is appealing since it would follow visual intuition, and we must not find an appropriate kernel function resolving expensive computations. In the scope of this chapter, we will therefore deal with the question: **RQ1: Can we perform Domain Adaptation with geometric transformations?** By providing an answer, we advance the field of shallow Domain Adaptation by the **following contributions:**

- A shallow Domain Adaptation algorithm that aligns the aforementioned geometric domain differences in 3.1.
- An augmentation strategy to align both domain sample sizes.
- The derived algorithm is analyzed in terms of the source to target approximation error.

The proposed solution is evaluated on common Domain Adaptation benchmarks with a model analysis is provided in Sec. 3.2. A summary and discussion are given at the end of a chapter. The chapter starts with the model. For an introduction to Domain Adaptation and related work please see Sec. 2.3 and 2.3.1.

3.1 MODEL

Overview. This section contains the description of a Domain Adaptation algorithm called Geometric Domain Adaptation (GDA). The section starts with the derivation of the method. Afterward, we introduce an augmentation schema to make the computation of the presented solution feasible. The section closes with the analysis of the time complexity and the properties of the transformed data in terms of approximation error.

Contribution. For the (GDA) approach, we rewrite the common least-squares formulation for Domain Adaptation. We will develop a brief geometric intuition, which helps us to solve the optimization problem in closed form and avoid developing theoretically and computationally complex algorithms while being competitive. The computations are made feasible by using a data augmentation schema.

Description

The primary assumption [79] in this chapter is that similar kernel or feature matrices imply similar underlying data distributions, reducing the distribution differences of

Domain Adaptation to a matrix alignment problem. It allows us to use the geometric intuition of domain separability, which are two domains, densely but separately clustered, in the feature space. A transformation procedure is sufficient for Domain Adaptation because both matrices are aligned - also in the geometric intuition. Hence, we will develop a Geometric Domain Adaptation (GDA) method, built upon the idea to approximate the source data matrix $\mathbf{X}_s$ with the target data matrix $\mathbf{X}_t$. The concept is to adapt the rotation and scaling properties of the target domain.

In shallow Domain Adaptation [57], [79], [99], we consider a labeled source dataset $\mathbf{D}_s = \{\mathbf{X}_s, Y_s\} = \{\mathbf{x}_i^s, y_i^s\}_{i=1}^n \overset{i.i.d.}{\sim} p(\mathcal{S})$ in the source domain $\mathcal{S}$ and an unlabeled target dataset $\mathbf{D}_t = \{\mathbf{X}_t\} = \{\mathbf{x}_j^t\}_{j=1}^m \overset{i.i.d.}{\sim} p(\mathcal{T})$ in the target domain $\mathcal{T}$ with same label space $\forall i, j : y_i, y_j \in \mathcal{Y}$. For a geometrically inspired Domain Adaptation algorithm, we search for left-sided transformation $\mathbf{M} : \mathbb{R}^n \to \mathbb{R}^n$ and a right-sided transformation $\mathbf{P} : \mathbb{R}^d \to \mathbb{R}^d$ to adapt source to target, i. e., $\mathbf{X}_s \sim \mathbf{X}_t$ in the original features space. To study the above Domain Adaptation problem we use the common least-square formulation [69], [92]. Extending it by the parameter matrices and assuming equally sized matrices the equation becomes

$$\min_{\mathbf{M},\mathbf{P}} \|\mathbf{M}\mathbf{X}_s\mathbf{P} - \mathbf{X}_t\|_F^2, \tag{3.1}$$

where $\mathbf{M} \in \mathbb{R}^{n\times n}$ and $\mathbf{P} \in \mathbb{R}^{d\times d}$ are transformation matrices drawing the data closer together.

A solution can easily be found analytically assuming $n = m$, summarized as follows: (i) standardize source and target to $\mathcal{N}(0, 1)$ to transform both domains to the origin, commonly used in Domain Adaptation [140]. We assume it will align marginal distributions in Euclidean space without considering label information. (ii) compute an (SVD) of source and target data, i. e., $\mathbf{X}_s = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ and $\mathbf{X}_t = \mathbf{L}\mathbf{T}\mathbf{R}^T$. (iii) Let us assume that $\mathbf{\Sigma} \sim \mathbf{T}$ and do not modify them during the next operations. This is not an unrealistic scenario due to the standardization in step one, reducing the scaling factor of singular values to the same range. (iv) obtain a solution for Eq. (3.1) by realizing that given assumption (iii) the optimization problem has a global optimum at

$$\begin{aligned}
\min_{\mathbf{M},\mathbf{P}} \|\mathbf{M}\mathbf{X}_s\mathbf{P} - \mathbf{X}_t\|_F^2 &= \left\|\mathbf{M}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{P} - \mathbf{L}\mathbf{T}\mathbf{R}^T\right\|_F^2 && (3.2)\\
&= \left\|\mathbf{L}\mathbf{U}^{-1}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}^{-1}\mathbf{R}^T - \mathbf{L}\mathbf{T}\mathbf{R}^T\right\|_F^2 && (3.3)\\
&= \left\|\mathbf{L}\mathbf{\Sigma}\mathbf{R}^T - \mathbf{L}\mathbf{T}\mathbf{R}^T\right\|_F^2 && (3.4)\\
&= \|\mathbf{\Sigma} - \mathbf{T}\|_F^2 && (3.5)
\end{aligned}$$

The solution is obtained by $\mathbf{M} = \mathbf{L}\mathbf{U}^{-1}$ and $\mathbf{P} = \mathbf{V}^{-1}\mathbf{R}^T$. Apply the transfer operation and approximate the source matrix by using target basis information for row and column span:

$$\bar{\mathbf{X}}_s = \mathbf{M}\mathbf{X}_s\mathbf{P} = \mathbf{L}\mathbf{U}^{-1}\mathbf{U}\mathbf{\Sigma}\mathbf{V}\mathbf{V}^{-1}\mathbf{R}^T = \mathbf{L}\mathbf{\Sigma}\mathbf{R}^T, \tag{3.6}$$

(a) Data not standardized | *(b)* Data after standardization | *(c)* Data after Geometric Domain Adaptation

Figure 3.1: The process of *Geometric Domain Adaptation* illustrated by a toy example with two domains. Class information is given by color (red/blue), and the domain is indicated by shape (domain one x, domain two $\cdot$). (a) the non-standardized data with a knowledge gap. (b) standardized features with $\mathcal{N}(0, 1)$. (c) Eq. (3.6) is applied, correcting the samples, i. e., shapes with the same color are aligned, and source data is used for learning a classification model. Best viewed on computer display.

with $\bar{\mathbf{X}}_s \in \mathbb{R}^{n \times d}$ as approximated source data, used for training. By this we assume for the Domain Adaptation problem $\bar{\mathbf{X}}_s \sim \mathbf{X}_t$ and because of it we are allowed to assume $p\mathcal{S}) \sim p(\mathcal{T})$ [57], [67].

Intuition

The left/right singular vectors are interpretable as rotations and singular values as scaling factors of the singular vectors based on underlying data. Hence, the above formulation yields a possible solution for geometric-based Domain Adaptation algorithms.

The four-step process is shown in figure 3.1 as geometrical interpretation demonstrated with a toy example created by Gaussian random sampling. The figure shows a synthetic dataset. Class one is labeled as red, and class two is labeled blue. The domains are given by shape $\cdot$ / x and identifying the source or target domain. First, the standardization draws the data to the origin. Second, applying our *Geometric Transfer* approach results in aligned domains, e. g., classes red and blue are separable without a shift from x to $\cdot$. Therefore, a classifier can be trained on x data and can be evaluated on $\cdot$ data.

Augmentation

The optimization problem in Eq. (3.1) has the assumption of equality sized domain matrices with sizes m and n and $n = m$. If the assumption is violated, both Eq. (3.2) and Eq. (3.6) will be invalid by definition.

Due to unlabeled target data at adaptation time, the source domain is augmented to match the target size if $n \neq m$. We limit the approach to text-based data because of the expression of class-wise topic spaces [22]. Hence, in this context, a topic space

is the set of all texts of the class samples. If the source matrix is larger, i. e. $n > m$, we assume the existence of a major and minor topic space, which is the domain with the larger and smaller sample size, respectively. By doing so, the major topic space is easily transformed to the minor topic space via class-wise application of *Latent Semantic Analysis* (LSA)[22], resulting in equally sized matrices. If $n < m$, we use class-wise Gaussian augmentation to enrich the source data to match the target data size. Note that the augmentation is designed not to produce any new semantic information to the classification problem. The pseudocode of the augmentation is summarized in algorithm 1 with $\boldsymbol{\mu}_c = \frac{1}{|c|}\sum_{y(\mathbf{x}_i^s)=c}\mathbf{x}_i^s$ is class-wise mean, $\boldsymbol{\sigma}_c = \frac{1}{|c|}\sum_{y(\mathbf{x}_i^s)=c}(\mathbf{x}_i^s - \boldsymbol{\mu}_c)^2$ is class-wise variance. This makes the approach and in particular Eq. (3.6) computable. Note that we will see simpler and intuitive augmentation strategies in chapter 4.

Algorithm 1 Source Domain Augmentation

procedure
Input: $\mathbf{X}_s; \mathbf{X}_t$ as n sized source and m sized target set; $\mathbf{Y}_s$ as n sized source label vector;
Output: Augmented source data matrix $\mathbf{X}_s$ with labels $\mathbf{Y}_s$
    $n_{max} = \frac{m}{|\mathcal{Y}|}$
    for each $c \in \mathbf{Y}_s$ **do**
        $n_{cur} = |\{\mathbf{x}_i^s | y(\mathbf{x}_i^s) = c,\ \forall \mathbf{x}_i^s \in \mathbf{X}_s\}|$
        if $n_{cur} > n_{max}$ **then**
            $\{\mathbf{X}_s^c, \mathbf{Y}_s^c\} = \{\mathbf{x}^s, y_s | \mathbf{x}^s \in \mathbf{X}_s \cap y_s = c\}$
            $\{\mathbf{X}_s, \mathbf{Y}_s\} = \{\mathbf{X}_s, \mathbf{Y}_s\} \setminus \{\mathbf{X}_s^c, \mathbf{Y}_s^c\}$
            $\mathbf{X}_s^c = PCA(\mathbf{X}_s^{cT}; n_{max})$ ▷ According to [22]
            $\mathbf{Y}_s^c = \{c\}^{n_{max}}$
            $\{\mathbf{X}_s, \mathbf{Y}_s\} = \{\mathbf{X}_s, \mathbf{Y}_s\} \cup \{\mathbf{X}_s^c, \mathbf{Y}_s^c\}$
        else
            $d = |n_{max} - n_{cur}|$
            $\mathbf{X}_s \cup \{\mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c)\}^d$
            $\mathbf{Y}_s \cup \{c\}^d$
        end if
    end for
end procedure

Properties

The procedure of (GDA), together with the augmentation procedure, is shown in algorithm 1. It is an asymmetric transfer approach [95] because the source domain is transformed to the target one. Further, it is transductive [39] and does need unlabeled target data at adaptation time. An obvious advantage of (GDA) is that it has no parameters and needs no parameter tuning.

The computational complexity is caused by two (SVD)s and $|\mathcal{Y}|$ eigendecompositions for class-wise (LSA) if $n \neq m$. This results in the overall complexity of $\mathcal{O}(min(n,m,d)^2)$ where n/m is source / target data matrix size. In the practical implementation, we use economy-sized (SVD).

The construction of new source data relies only on singular values from original source data and singular vectors are taken from target set. Hence, the approximation of the target domain source domain as in Eq. (3.6) is formalized by

$$E_{GDA} = ||\mathbf{\Sigma} - \mathbf{T}||_F^2 \tag{3.7}$$

Proof.

$$E_{GDA} = ||\bar{\mathbf{X}}_s - \mathbf{X}_t||_F^2 = ||\mathbf{L\Sigma R}^T - \mathbf{LTR}^T||_F^2 = ||\mathbf{L}(\mathbf{\Sigma} - \mathbf{T})\mathbf{R}^T||_F^2. \tag{3.8}$$

Because the Frobenius norm is unitarily invariant [49] and $\mathbf{L}, \mathbf{R}$ are orthogonal matrices the following holds:

$$E_{GDA} = ||\mathbf{L}(\mathbf{\Sigma} - \mathbf{T})\mathbf{R}||_F^2 = ||\mathbf{\Sigma} - \mathbf{T}||_F^2. \tag{3.9}$$

□

The proof shows that all affine matrix transformations by the singular vectors causing rotations or scaling are omitted, making the resulting features invariant to the domain-specific transformation. Further, the proof implies that the difference in domains is captured by the Frobenius norm of the singular values. By the statistical meaning of the singular values [94], the difference in domains is the difference in feature variance.

Algorithm 2 Geometric Domain Adaptation (GDA)

1: **procedure**
2: **Input:** $\mathbf{X}_s; \mathbf{X}_t$ as n sized source and m sized target set; $\mathbf{Y_s}$ as n sized source label vector;
3: **Output:** Adapted source data $\bar{\mathbf{X}}_s, \mathbf{Y}_s^a$
4: $\mathbf{X}_s^n, \mathbf{X}_t^n = standardization(\mathbf{X}_s, \mathbf{X}_t)$ ▷ Effect as in Fig. 3.1b
5: $\mathbf{X}_s^a, \mathbf{Y}_s^a = augmentation(\mathbf{X}_s^n, \mathbf{Y}_s)$ ▷ According to Alg. 1
6: $\mathbf{\Sigma} = SVD(\mathbf{X}_s^a)$
7: $\mathbf{L}, \mathbf{R} = SVD(\mathbf{X}_t)$
8: $\bar{\mathbf{X}}_s = \mathbf{L\Sigma R}^T$ ▷ According to Eq. (3.6)
9: **end procedure**

3.2 EXPERIMENTS

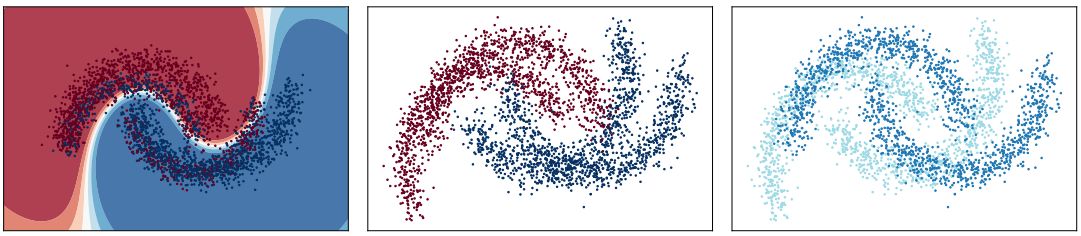
Overview. We follow the experimental design typical for shallow Domain Adaptation focusing on text datasets [26], [65], [79]. The experiments contain three main parts: **(i)** a

proof of concept validating (GDA) mechanics on the two moon problem in Sec. 3.2.1. **(ii)** exhausting empirical evaluation of (GDA) on two text-Domain Adaptation datasets in Sec. 3.2.4. **(iii)** qualitative evaluation of model behavior against a competitive approach in Sec. 3.2.5. To avoid duplications, we evaluate the time consumption and quantitative model complexity together with the remaining shallow Domain Adaptation models in Sec. 4.4.6 and 4.4.8. Detailed descriptions of the datasets, competitive methods and their parameters as well as the experimental setup are given in Sec. 3.2.2 and Sec. 3.2.3.

3.2.1 *Proof of Concept*

The mechanics of (GDA) are illustrated on the two moon problem shown in Fig. 3.2. The first plot shows the data after (GDA) transformation with ground truth labeling while the background is colored with the prediction confidence of a classifier trained after (GDA) takes place. The second and third plots show original data with class and domain labels, respectively.

The alignment of both domains is visible because the half moons with respect to the domain are aligned together. The difference in shift is reduced by applying standardization, and the rotational differences initially causing the two moon problem are reduced by (GDA). This can be confirmed by comparing the moons with same color in Fig. 3.2a and 3.2b. The adaptation draws the moons of the same color together. However, the approach does not incorporate class information. Therefore, the blue class protrudes somewhat into the red class.



(a) GDA Representation with classifier confidence

(b) Input Representation Class Labels

(c) Input Representation Domain Labels

Figure 3.2: First plot shows the moon problem after (GDA) transformation with a classifiers confidence as background, while data is colored based on ground truth labels. Second plot shows the original two moon problem with ground truth labels and third plot shows original data with domain labels. The (GDA) adaptation can be observed at the moons of the same color, as they are closer together as initially generated shown in Fig. 3.2b.

3.2.2 *Datasets*

A crucial characteristic of the datasets for Domain Adaptation is that domains for source and target are different but related. This relation exists because the source and target classes have the same top category or source. For an overview of the category types see Tab. 3.1. The classes themselves are subcategories or subsets. The study consists of the text benchmark datasets Reuters [25] and 20-Newsgroup [65].

Summary. An overview of the text datasets is shown in Tab. 3.1. The datasets are structured into top categories and subcategories. These top categories are regarded as labels, and the subcategories are used for training and testing. The variation of subcategories between training and testing creates domain differences. The difference to image datasets is that in image datasets, the subcategories are labels, e. g., desk or mug, and the difference in the top category (source of images) between training and testing, e. g., *Caltech* to *Amazon*, creates the domain differences. An overview of the text datasets is given in Tab. 3.1. Note that the linked version of datasets with the same choice of subcategories should be used to reproduce the results below. Regardless of the dataset, features have been standardized to standard mean and variance. **Reuters-21578**[1] [25]: A set of Reuters news-wire articles was collected in 1987 with a hierarchical structure given as top-categories and subcategories to organize the articles. The three top categories *Organization (Orgs)*, *Places* and *People* are used in our experiments. The category *Orgs* has 56 subcategories, *Places* has 176 and *People* has 269. In the category *Places*, all articles about the USA are removed, making the three nearly even distributed in terms of articles.

We follow the sampling scheme from [25], which will be discussed in the following. Note that the top categories, which are just mentioned, are the labels of the datasets.

All subcategories of a top category are randomly divided into two sub-subcategories with about the same number of articles. This selection is fixed for all experiments. For a top category O this creates two parts $o1$ and $o2$ of subcategories and for another top category P this creates the parts $p1$ and $p2$. The top category O is regarded as a class and P as another one.

The Domain Adaptation problem is created by using $o1$ and $p1$ for training, and $o2$ and $p2$ are used for testing. This is a two-class problem because of the two top categories, O, and P. Such a configuration is called O *vs* P. If the second part is used for training, i. e., $o2$ and $p2$, and the first part for testing, i. e., $o1$ and $p1$, it is regarded as P *vs* O. The individual subcategories have different distributions but are related by the top category, creating a change in distribution between training and testing.

Based on this six datasets are generated: **Orgs**→**People**, **People**→**Orgs**, **Orgs**→**Place**, **Place**→**Orgs**, **People**→**Place** and **Place**→**People**. The articles are converted to lower

1 http://www.daviddlewis.com/resources/testcollections/reuters21578

case, words are stemmed, and stopwords are removed. With the Document Frequency (DF)-Threshold of 3, the numbers of features are cut down. The features are generated with Term-Frequency Inverse-Document-Frequency (TFIDF). For a detailed choice of subcategories, see [25].

20-Newsgroup[2] [65]: The original collection has approximately 20000 text documents from 20 newsgroups. The four top categories are *comp*, *rec*, *talk* and *sci* and containing four subcategories each. We follow a data sampling scheme introduced by [79] and generate 216 cross domain datasets based on subcategories: Again the top categories are the labels, and the subcategories are varied between training and testing to create a Domain Adaptation problem.

Let C be a top category and $C1, C2, C3, C4$ are subcategories of C and another top category with K and $K1, K2, K3, K4$ are subcategories of K. A dataset is constructed by selecting two subcategories for each top-category, e. g., $C1$, $C2$, $K1$, and $K2$, for training and select another four, e. g., $C3$, $C4$, $K3$, and $K4$ for testing. The top categories C and K are respective classes.

For two top categories every permutation is used and therefore $C_4^2 \cdot K_4^2 = 36$ datasets are generated. By combining each top category with each other there are 216 dataset combinations. The datasets are summarized as mean per top category combination: **Comp**→**Rec**, **Comp**→**Sci**, **Comp**→**Talk**, **Rec**→**Sci**, **Rec**→**Talk**, **Sci**→**Talk**. The Domain Adaptation problem is created by training and testing on different subcategories analogy to Reuters. This version of *20-Newsgroup* has 25804 TF-IDF features within 15033 documents [79].

3.2.3 *Implementation Details*

Competitive Methods. We compare the (GDA) approach against Transfer Component Analysis (**(TCA)**) [45], Joint Distribution Adaptation (**(JDA)**) [59], Geodesic Flow Kernel (**(GFK)**) [47], Subspace Alignment (**SA**) [57], and Transfer Kernel Learning (**(TKL)**) [79]. The baseline and underlying classification algorithm of the methods is the (SVM).

Parameters. The proposed methods use the (SVM) as the underlying classification model with $C = 10$ and the (RBF) kernel with kernel width of $\sigma = 1$. (TKL) has the eigenvalue damping factor set to 2. The (JDA) subspace is set to 100, for (GFK) to 40, for (TCA) to 60, for SA to 5. The (JDA) regularization is set to 1. The (GDA) approach has no hyperparameters.

Experimental Setup. We follow the experimental setup of [79] and show mean accuracy over the 216 Newsgroup datasets but with 2-fold sampling over two runs summarized to top categories. Further, we show the mean accuracy over five random runs over the six Reuters datasets.

2 http://qwone.com/~jason/20Newsgroups/

Top Category (Names)	Subcategory	#Samples	#Features	#Classes
Comp	comp.graphics	970	25804	2
	comp.os.ms-windows.misc	963		
	comp.sys.ibm.pc.hardware	979		
	comp.sys.mac.hardware	958		
Rec	rec.autos	987		
	rec.motorcycles	993		
	rec.sport.baseball	991		
	rec.sport.hokey	997		
Sci	sci.crypt	989		
	sci.electronics	984		
	sci.med	987		
	sci.space	985		
Talk	talk.politics.guns	909		
	talk.politics.mideast	940		
	talk.politics.misc	774		
	talk.religion.misc	627		
Orgs	56 subcategories	1237	4771	2
People	269 subcategories	1208		
Places	176 subcategories	1016		

Table 3.1: Dataset characteristics of text Domain Adaptation datasets containing top-categories and corresponding subcategories, numbers of samples, features and labels. Horizontal line separates dataset in 20-Newsgroup (upper half) and Reuters (lower half) [79]. At Reuters there are many subcategories, therefore we only show the number of subcategories.

3.2.4 *Performance Results*

The experimental results are shown in accuracy for Newsgroup in Tab. 3.2 and Reuters in Tab. 3.3. For Newsgroup, the (GDA) approach is by far the best approach. It outperforms the remaining competitive approaches. The good performance validates the initial geometric intuition and the subsequent derivation of the (GDA) method and demonstrates its use for Domain Adaptation. The performance gain of (SVM) with (GDA) is roughly a fifth. At the Reuters dataset, the (GDA) approach is outperformed by SA. The SA is the only competitive solution because (GDA) outperforms the remaining ones while the difference between (GDA) and SA is less than five percent. Combining the (SVM) and (GDA) raises the accuracy by about a quarter to vanilla (SVM). This again confirms the usefulness of the geometric derivation.

(GDA) is hyperparameter-free, while the only competitive algorithm needs the tuning of the subspace parameter. Hence, the approach is easy to apply and provides very competitive performance on two text datasets, making it favorable to compared solutions.

Dataset	(SVM)	(TCA)	(JDA)	(GFK)	SA	(TKL)	(GDA)
Comp→Rec	77.6	78.9	83.1	75.1	78.5	95.3	**99.7**
Comp→Sci	71.1	62.0	75.5	64.1	80.2	87.4	**99.7**
Comp→Talk	84.4	75.0	87.7	83.8	91.1	96.6	**96.9**
Rec→Sci	69.3	79.6	79.0	64.4	81.1	86.9	**99.9**
Rec→Talk	74.5	86.6	82.0	72.9	79.7	88.6	**96.7**
Sci→Talk	70.9	77.6	70.5	64.2	76.0	85.2	**96.8**
Avg.	74.6	76.6	79.6	70.7	81.1	90.0	**98.3**

Table 3.2: Mean **accuracy** of shallow DA methods on **Newsgroup** text dataset.

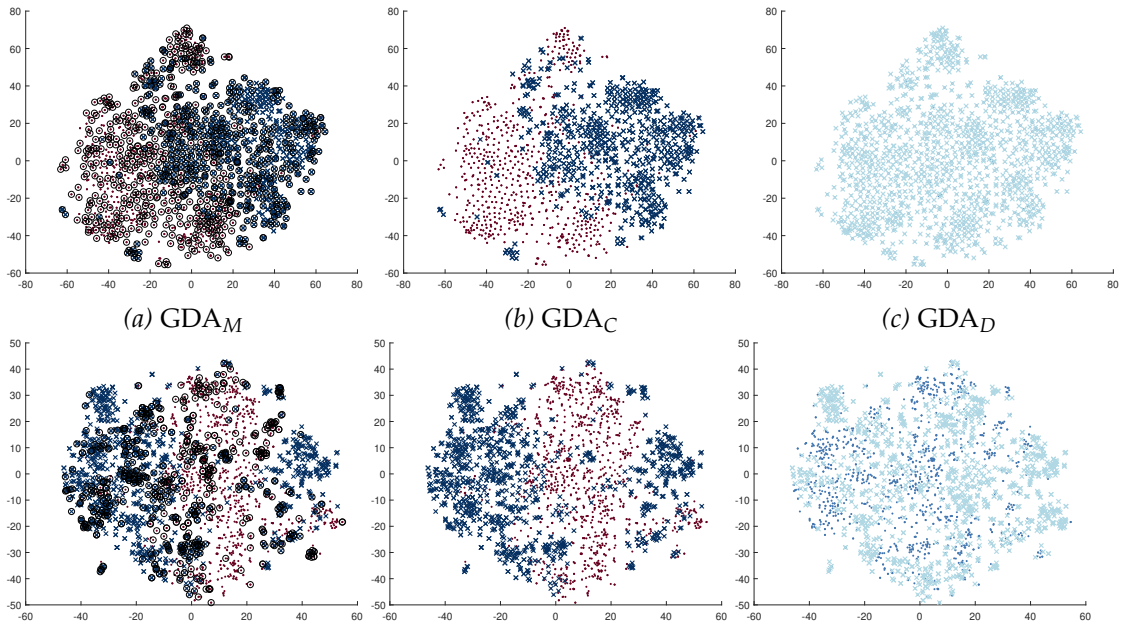
Dataset	(SVM)	(TCA)	(JDA)	(GFK)	SA	(TKL)	(GDA)
Orgs→People	78.1	79.5	76.6	75.3	**99.9**	80.7	97.8
People→Orgs	79.2	82.7	80.0	71.6	**99.9**	87.2	97.2
Orgs→Place	69.2	72.9	70.0	60.5	**97.3**	70.3	94.7
Place→Orgs	66.3	71.1	65.6	61.5	**97.2**	81.7	94.4
People→Place	55.7	57.4	57.0	57.5	**97.4**	70.5	94.8
Place→People	57.4	48.9	60.7	56.2	**97.4**	66.6	90.5
Avg.	67.7	68.7	68.3	63.8	**98.1**	77.3	94.9

Table 3.3: Mean **accuracy** of shallow DA methods on **Reuters** text dataset.

3.2.5 *Model Analysis*

The (GDA) and (TKL)[79], trained on **Orgs→People**, are plotted with TSNE in Fig. 3.3. The first column shows the two dimensional representation of both Orgs and People, where the black circles mark a support vector, while a red dot marks a source example and a blue x shows a target example. The second column is for clarity and shows the same representation with the same coloring. The third column shows the data with domain labels. To avoid overlap, we refer the reader to Sec 4.4.6 for a quantitative overview of the model complexity.

The mean accuracy of (GDA) on this dataset is 97.8% indicating the adaptation success. We can report that (GDA) can create inseparable domains and causing the distribution to appear very similar. Further, the blue class is roughly clustered into a single large area with only minor exceptions, explaining the good performance of (GDA) - especially against (TKL). The (SVM) can represent both classes very well. However, the complexity of the model is also more extensive than the (TKL) (SVM) model. The pairwise distances appear very similar at the (GDA) plot, making it hard for the (SVM) to capture the class boundaries with a sparser model, yielding a potential explanation for the more complex model.



(a) GDA$_M$ *(b)* GDA$_C$ *(c)* GDA$_D$

(d) TKL$_M$ *(e)* TKL$_C$ *(f)* TKL$_D$

Figure 3.3: TSNE [29] representation of Reuters **Orgs**→**People** data of the algorithms (TKL) [79] and (GDA). <Name>$_M$ shows the support vectors in black circles of (SVM) trained on the features generated by <Name>, while data is colored by ground truth labels. <Name>$_C$ and <Name>$_D$ show the same representation colored with class and domain labels for better visibility. (GDA) adapts source to target successfully with a test accuracy of 97.8%. Best viewed on computer display.

3.3 DISCUSSION

We presented the combination of two geometric transformations reinterpreted as Domain Adaptation algorithm, answering the research questions from the beginning. It follows the intuition of solving the Domain Adaptation problem by rearranging the source domain to behave like the target one. The transformations needed to solve the task are obtained by combining the singular vectors of source and target. The adapted source domain is close to the target domain and is also invariant to the own rotations and scaling caused by source singular vectors. To make the necessary computations feasible, we introduced a class-sensitive augmentation schema to align domain matrix sizes. The approach is hyperparameter-free and is very competitive to related shallow methods.

Limitations. The geometric intuition is solution and limitation at the same time. It solely concentrates on aligning marginal data distributions without considering differences in feature variances or local class structures. This limits the adaptation capacities of the approach to problems where the underlying class conditional probability distribution

and the feature variances of both domains are similar. If only the feature distributions are different, the method has a high accuracy expectation, demonstrated above.

Further, the proposed solution requires a full eigendecomposition over all available data and is prone to domain specific noise even after adaption. In the next chapter, the proposed optimization problem is modified, and by providing a solution to it, we tackle the just mentioned issues by requiring only a snapshot of data. A future outlook is discussed at the end of the next chapter.

4 SHALLOW SUBSPACE DOMAIN ADAPTATION

Summary: Subspace learning provides a natural way for low-dimensional data representations to focus on relevant feature characteristics. These methods also allow learning invariant features in terms of Domain Adaptation. The resulting solutions are easy to apply while requiring less data as competitive non-subspace approaches. However, existing methods project data from different domains in a common or target domain subspace without bounding the degree of domain alignment.

This chapter will propose a subspace algorithm projecting the data into a common low-dimensional space and characterizing the domain differences in this space by their singular values. Moreover, we do so while having all advantages mentioned above: easy and data-type independent application, low number of hyperparameters, and requiring less data combined in a single solution.

Publications: This chapter is based on the following publications.

- C. Raab and F.-M. Schleif, 'Transfer Learning extensions for the probabilistic classification vector machine,' Neurocomputing, vol. 397, pp. 320–330, Jul. 2020, doi. 10.1016/j.neucom.2019.09.104.
- C. Raab and F.-M. Schleif, 'Low-Rank Subspace Override for Unsupervised Domain Adaptation,' in KI 2020: Advances in Artificial Intelligence, U. Schmid, F. Klügl, and D. Wolter, Eds. Cham: Springer International Publishing, 2020, pp. 132–147, doi. 10.1007/978-3-030-58285-2_10.

Source Code: https://github.com/ChristophRaab/thesis/tree/main/sda/.

Supervised classification models are restricted to the so-called source domain - the given underlying probability distribution of the training data. Consequently, these models are unable to generalize well to other domains denoted as the target, occurring naturally in many applications [87], [135]. Domain Adaptation methods are used to adapt a domain sample or a trained model to the desired domain. However, these techniques suffer either from being restricted to a particular task, such as deep vision adaptation [116] or text classification, described in the previous chapter. Further, they require much computational time and data [117], which is not always guaranteed, have complex parameterization [134], or expensive optimization procedures [157]. The limitation is present for deep and shallow Domain Adaptation models.

Subspace Domain Adaptation models, on the other hand, provide a way to omit domain-specific content by employing low-rank techniques [141], which are easy to

parametrize [57] and do not require large portions of target data [69]. Moreover, the inspected models are shallow without needing millions of parameters. They are applicable regardless of the downstream tasks such as text classification or object detection, given that the data are sufficiently preprocessed. Thereby, shallow adaptation models provide a representation for both domains suitable for a classification model. In relation to the Domain Adaptation theory [36], it means we first learn a suitable domain representation for $\mathcal{S}$, and $\mathcal{T}$ and obtain a hypothesis *h after* the adaptation process. Hence, the proposals in this chapter strive to minimize domain separability. See Sec. 2.3.1 for an introduction to the theory.

We learn subspace projectors in the quest for domain invariant subspaces, fulfilling the above constraints on the Domain Adaptation task. While related methods [57] provide a bounded source projector in terms of L_2 difference to the target one, they are not limit the resulting subspace in terms of domain differences. However, necessary for quantifying the still existing differences of the domains in the subspace. In the scope of this chapter, we will therefore deal with the question: **RQ2: Can subspace projectors yield a bounded subspace for Domain Adaptation?** By providing an answer, we advance the field of shallow subspace Domain Adaptation by the **following contributions:**

- A subspace projector bounding the source and target subspace to the difference in spectra (Sec. 4.1).
- The derived approach is discussed in the light of the Domain Adaptation theory [36] (Sec. 4.3).
- The solution is extended to need only a snapshot of data by employing a low-rank technique, focusing on descriptive data characteristics (Sec. 4.2 and 4.3).
- The projector is obtained via closed-form, overriding domain structures, resulting in a fast solution easy to parametrize.

The presented idea is evaluated in Sec. 4.4 on various Domain Adaptation tasks such as text and image classification against state-of-the-art Domain Adaptation approaches and achieves remarkable performance across all tasks. A summary and discussion about the limitations, together with a future outlook, are provided at the end of the chapter. The chapter starts with the first model without introducing Domain Adaptation and related work because of the introduction in Sec. 2.3 and 2.3.1.

4.1 MODEL

Overview. This section contains the description of a shallow subspace algorithm called Subspace Override (SO). In the following, the method is derived from the Least-Squares

perspective. The section closes with a discussion of the complexity and the subspace properties of the approach.

Contribution. Related subspace Domain Adaptation approaches utilize the PCA for Domain Adaptation. However, the computational and time requirements of these algorithms are by far too high compared to the performance gain. The obtained subspace is not bounded in terms of the domain characteristics, and there is no guarantee that the source and target domain are in the same subspace. The contribution tackles the issues by projecting both domains into the same bounded subspace while outperforming competitive shallow Domain Adaptation methods.

Description

The task of Domain Adaptation is to align distribution differences with the goal that the domain distributions will be similar afterward. As in prior work [57], [67], [69], [77], [79], [99], we assume that similar matrices will lead to similar distributions. Hence, we strive for aligning the domain data matrices in a suitable subspace and model the source data to be part of the target data, and therefore it *must* be in the same (single) subspace. For transforming domain data into the subspace, we need a subspace projector $\mathbf{M} : \mathbb{R}^n \rightarrow \mathbb{R}^m$. In the following, we derive a suitable projector and the characteristics a projector needs to provide a bounded subspace.

In shallow Domain Adaptation [57], [79], [99], we consider a labeled source dataset $\mathbf{D}_s = \{\mathbf{X}_s, Y_s\} = \{\mathbf{x}_i^s, y_i^s\}_{i=1}^n \overset{i.i.d.}{\sim} p(\mathcal{S})$ in the source domain $\mathcal{S}$ and an unlabeled target dataset $\mathbf{D}_t = \{\mathbf{X}_t\} = \{\mathbf{x}_j^t\}_{j=1}^m \overset{i.i.d.}{\sim} p(\mathcal{T})$ in the target domain $\mathcal{T}$ with same label space $\forall i, j : y_i, y_j \in \mathcal{Y}$. To draw both domains closer together, represented by their respective samples $\mathbf{X}_s$ and $\mathbf{X}_t$, consider the following optimization function

$$\min_{\mathbf{M}} ||\mathbf{M}\mathbf{X}_s - \mathbf{X}_t||_F^2, \tag{4.1}$$

$$s.t. \;\; \mathbf{M}\mathbf{M}^T = \mathbf{I}. \tag{4.2}$$

The goal is to learn $\mathbf{M}$ to adapt $\mathbf{X}_s$ to the target domain. Further, we make sure that the obtained projection operator is an orthogonal basis. The formulation is quite unusual for subspace learning because the parameter matrix appears on the wrong side of the equation. However, we will see the advantages of this formulation in the following description.

The optimization problem has two flaws; first, matrix sizes must be equal. We tackle the problem via an augmentation strategy later on in this chapter. Second, Eq. (4.1) prevents effective Domain Adaptation, because the transformation $\mathbf{M}$ may project the data in different spaces [67]. However, if we model $\mathbf{M}$ to be directly related to the target domain, the projection operator will be domain invariant. To obtain this kind of solution for the problem in Eq. (4.1), we rewrite it such that source data is part of the target subspace.

Let us consider the relationship between singular- and eigendecomposition and rewrite the PCA in terms of SVD. Given the rectangular target data matrix $\mathbf{X}_t \in \mathbb{R}^{m \times d}$ we can rewrite the eigendecomposition to

$$\mathbf{X}_t^T \mathbf{X}_t = (\mathbf{R}\mathbf{T}^T\mathbf{L}^T)(\mathbf{L}\mathbf{T}\mathbf{R}^T) = \mathbf{R}\mathbf{\Sigma}^2\mathbf{R}^T, \tag{4.3}$$

with $\mathbf{T} \in \mathbb{R}^{m \times d}$ as singular values and $\mathbf{L} \in \mathbb{R}^{m \times m}$ are singular vectors of $\mathbf{X}_t$. Further, $\mathbf{T}^2 = \mathbf{T}^T\mathbf{T} \in \mathbb{R}^{m \times d}$ as eigenvalues and $\mathbf{R} \in \mathbb{R}^{d \times d}$ as eigenvectors of $\mathbf{X}_t^T\mathbf{X}_t$. A low rank solution and a reduction of dimensionality is integrated into the new data matrix by sorting $\mathbf{T}$ and $\mathbf{R}$ in descending order with respect to $\mathbf{T}$ and choosing only the biggest l[1] eigenvalues and corresponding eigenvectors

$$\mathbf{X}_t^l = \mathbf{X}_t\mathbf{R}^l = \mathbf{L}^l\mathbf{T}^l{\mathbf{R}^l}^T\mathbf{R}^l = \mathbf{L}^l\mathbf{T}^l \in \mathbb{R}^{m \times l}, \tag{4.4}$$

with $\mathbf{L}^l \in \mathbb{R}^{m \times l}$ and $\mathbf{T}^l \in \mathbb{R}^{l \times l}$ and $\mathbf{R}^l \in \mathbb{R}^{d \times l}$. $\mathbf{X}_t^l$ is the reduced target matrix and only the most relevant data w.r.t. to variance is kept. In Eq. (4.3) a linear covariance or kernel is used, but non-linear kernels like the RBF kernel could be integrated as well.

With the insights of Eq. (4.3) and Eq. (4.4), we rewrite the optimization problem in Eq. (4.1) to a low-rank subspace version and state the **main optimization problem**:

$$\begin{aligned} \min_{\mathbf{M}} \; & ||\mathbf{M}\mathbf{U}^l\mathbf{\Sigma}^l - \mathbf{L}^l\mathbf{T}^l||_F^2, \\ & s.t. \;\; \mathbf{M}\mathbf{M}^T = \mathbf{I}. \end{aligned} \tag{4.5}$$

Based on domain relatedness and standardization techniques, we assume that singular values are similar, i. e., $\mathbf{\Sigma}^l \simeq \mathbf{T}^l$ and fix them. Naturally, this assumption does not always hold. See Sec. 4.3 for a discussion. *If they are fixed, then* the optimal solution to Eq. (4.5) is easily obtained by solving the linear equation and obtain the solution $\mathbf{M} = \mathbf{L}^l{\mathbf{U}^l}^T$. By applying $\mathbf{M}$ to Eq. (4.5) the source data becomes

$$\mathbf{X}_s^l = \mathbf{M}\mathbf{U}^l\mathbf{\Sigma}^l = \mathbf{L}^l{\mathbf{U}^l}^T\mathbf{U}^l\mathbf{\Sigma}^l = \mathbf{L}^l\mathbf{\Sigma}^l \in \mathbb{R}^{m \times l} \tag{4.6}$$

and is used for training an invariant classifier. The resulting model can be evaluated on $\mathbf{X}_t^l = \mathbf{L}^l\mathbf{T}^l \in \mathbb{R}^{m \times l}$. This overrides the source basis and prevents the source subspace to be arbitrarily different from the target due to the affiliation to the target space. The solution also fulfils the constrains of orthogonality because $\mathbf{M}$ is orthogonal due to the orthogonal $\mathbf{L}^l$ and ${\mathbf{U}^l}^T$. In particular, Eq. (4.6) projects the source data onto the principal components of the subspace basis of $\mathbf{X}_t$. If data matrices $\mathbf{X}_t$ and $\mathbf{X}_s$ are standardized, the geometric interpretation is a rotation of source data w.r.t to angles of the target basis. Hence, we solely adapt the marginal distributions of the domains. We call this procedure Subspace Override (SO).

1 The number of subspace dimensions l is the only hyperparameter and is optimized via grid search.

Gaussian Augmentation

The problem of different-sized matrices is not new. In Sec. 3.1, sample sizes of data matrices must also be aligned. We did this by assuming a latent topic space [22] in the smaller sample matrix and projected the samples of the larger one into this latent space according to class affiliation. However, this procedure has two main disadvantages. First, the Latent-Semantic-Analysis (LSA) [22] only applies to text data. Second, the LSA has to be computed for every class, which can be expensive.

For the just introduced SO approach, the matrix size alignment is not necessary as seen in Eq. (4.6) and Eq. (4.17), because differences in size are aligned during transformation. However, the original dataset $\mathbf{X}_s$ has an $n \times 1$ sized label vector with $n \neq m$, which does not correspond to $\tilde{\mathbf{X}}_s$. This label-vector should not be transformed into the new size because semantic label information does not correspond with transformed data. Hence, sample sizes must still be the same, i. e., $m = n$, but is not required by the definition in Eq. (4.6). We propose a data augmentation strategy for solving different sample sizes, *applied before* doing knowledge transfer. Data augmentation is standard in machine or deep learning and has various applications [74], [86]. However, source and target data should have a reasonable size to encode domain knowledge properly.

In general, there are two cases, first $n < m$, meaning there is not enough source data. This is augmented via sampling from a class-wise multivariate Gaussian distribution $\mathcal{N}$ harmonizing the number of samples per class of source data. The other case is $n > m$ and is solved by Uniform random removal of source data from the classes with too many samples. The approach reduces source data to size m. This is somewhat counter-intuitive because one does not want to reduce the source set. However, we have no class information of the target set at training time, and we would be guessing class labels of target data when adding new artificial samples. The data augmentation strategy is summarized in pseudocode 3, where $\boldsymbol{\mu}_c = \frac{1}{|c|} \sum_{y(\mathbf{x}_i^s)=c} \mathbf{x}_i^s$ is class-wise mean, $\boldsymbol{\sigma}_c = \frac{1}{|c|} \sum_{y(\mathbf{x}_i^s)=c} (\mathbf{x}_i^s - \boldsymbol{\mu}_c)^2$ is class-wise variance. The function $y(\cdot)$ maps a source sample to the ground truth label $c \in \mathcal{Y}$ and $|c|$ is the number of class sample occurrences. The Gaussian Augmentation will also be used in the remaining sections of this chapters.

Properties

The pseudocode of SO, together with the Gaussian augmentation, is shown in algorithm 4. The SO algorithm consists of two economy sized SVD with $\mathcal{O}(2min(n, m, d)^2)$, two matrix multiplication with again $\mathcal{O}(2min(n, m, d)^2)$ plus standardization, making an overall complexity of $\mathcal{O}(min(n, m, d)^2)$.

According to [95], it is an asymmetric transfer approach because the approach projects the data into the target subspace. Further, it is transductive [39] and does need unlabeled target data. The orthogonal projector $\mathbf{M}$ yields a bounded subspace for affected domains as detailed in the following:

Algorithm 3 Gaussian Augmentation (GA)

procedure
Input: $\mathbf{X}_s$ as n sized source matrix, $\mathbf{X}_t$ as m sized target matrix; $\mathbf{Y}_s$ as n sized source label vector;
Output: Augmented source data matrix $\mathbf{X}_s^a$ with labels $\mathbf{Y}_s^a$
$n_{max} = m/|\mathcal{Y}|$
for each $c \in \mathcal{Y}$ **do**
$n_{cur} = |\{\mathbf{x}_i^s | y(\mathbf{x}_i^s) = c, \ \forall \mathbf{x}_i^s \in \mathbf{X}_s\}|$
$d = |n_{max} - n_{cur}|$
if $n_{cur} > n_{max}$ **then**
$\{\mathbf{X}_s, \mathbf{Y}_s\} \setminus \{\mathbf{x}^s, y_s | \mathbf{x}^s \in \mathbf{X}_s \cap p(\mathbf{x}^s) = \frac{1}{n_{cur}} \cap y_s = c\}^d$
else
$\mathbf{X}_s \cup \{\mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c)\}^d$
$\mathbf{Y}_s \cup \{c\}^d$
end if
end for
return $\mathbf{X}_s, \mathbf{Y}_s$ as $\mathbf{X}_s^a, \mathbf{Y}_s^a$
end procedure

Algorithm 4 Subspace Override (SO)

procedure
Input: $\mathbf{X}_s$ as n sized source matrix; $\mathbf{X}_t$ as m sized target matrix; $\mathbf{Y}_s$ as n sized source label vector; l as number of subspace dim;
Output: Adapted source data $\mathbf{X}_s^l, \mathbf{Y}_s^a$ and subspace target matrix $\mathbf{X}_t^l$
$[\mathbf{X}_s^n, \mathbf{X}_t^n] = standardization(\mathbf{X}_s, \mathbf{X}_t)$ ▷ $\mathbf{X}_s, \mathbf{X}_t \sim \mathcal{N}(0, 1)$
$\mathbf{X}_s^a, \mathbf{Y}_s^a = GA(\mathbf{X}_s^n, \mathbf{Y}_s)$ ▷ According to Alg. 3 in Sec. 4.1
$\mathbf{\Sigma} = SVD(\mathbf{X}_s)$;
$\mathbf{L}, \mathbf{T}, \mathbf{R} = SVD(\mathbf{X}_t)$;
$\mathbf{X}_s^l = \mathbf{L}^l \mathbf{\Sigma}^l$ ▷ According to Eq. (4.6)
$\mathbf{X}_t^l = \mathbf{L}^l \mathbf{T}^l$ ▷ According to Eq. (4.4)
return $\mathbf{X}_s^l, \mathbf{Y}_s^a, \mathbf{X}_t^l$
end procedure

Theorem 4.1.1. *Given two rectangular matrices $\mathbf{X}_t, \mathbf{X}_s \in \mathbb{R}^{n \times d}$ with $n, d > 1$ and rank of $\mathbf{X}_t$ and $\mathbf{X}_s > 1$. The squared Frobenius norm $\left\|\mathbf{X}_s^l - \mathbf{X}_t^l\right\|_F^2$ in the subspace $\mathbb{R}^l$ induced by orthogonal matrix $\mathbf{M} \in \mathbb{R}^{n \times l}$ with $\mathbf{M}^T\mathbf{M} = \mathbf{I}$ is bounded by*

$$E_{SO} = \left\|\mathbf{X}_s^l - \mathbf{X}_t^l\right\|_F^2 \leq \sum_{i=1}^{l+1}(\sigma_i(\mathbf{X}_s) - \sigma_i(\mathbf{X}_t))^2 \leq \|\mathbf{X}_s - \mathbf{X}_t\|_F^2 . \tag{4.7}$$

Proof. Following [49], the squared Frobenius norm of a matrix difference between two matrices can be bounded by

$$\sum_{i=1}^{q}(\sigma_i(\mathbf{X}_s) - \sigma_i(\mathbf{X}_t))^2 \leq \|\mathbf{X}_s - \mathbf{X}_t\|_F^2 , \tag{4.8}$$

where $q = min(n, d)$ and $\sigma_i(\cdot)$ is the i-th singular value of the respective matrix in descending order. However, the subspace matrices $\mathbf{X}_s^l$ and $\mathbf{X}_t^l$ are a special case due to the subspace override of the projector $\mathbf{M} = \mathbf{L}^l{\mathbf{U}^l}^T$, because

$$\begin{aligned}
\left\|\mathbf{X}_s^l - \mathbf{X}_t^l\right\|_F^2 &= \left\|\mathbf{M}\mathbf{U}^l\mathbf{\Sigma}^l - \mathbf{L}^l\mathbf{T}^l\right\|_F^2 = \left\|\mathbf{L}^l\mathbf{\Sigma}^l - \mathbf{L}^l\mathbf{T}^l\right\|_F^2 && (4.9)\\
&= \left\|\mathbf{L}^l\mathbf{\Sigma}^l\right\|_F^2 + \left\|\mathbf{L}^l\mathbf{T}^l\right\|_F^2 - 2tr({\mathbf{\Sigma}^l}^T{\mathbf{L}^l}^T\mathbf{L}^l\mathbf{T}^l) && (4.10)\\
&= \left\|\mathbf{\Sigma}^l\right\|_F^2 + \left\|\mathbf{T}^l\right\|_F^2 - 2tr({\mathbf{\Sigma}^l}^T\mathbf{T}^l) && (4.11)\\
&= \sum_{i=1}^{l}\sigma_i^2(\mathbf{X}_s^l) + \sum_{i=1}^{l}\sigma_i^2(\mathbf{X}_t^l) - 2\sum_{i=1}^{l}(\sigma_i(\mathbf{X}_s^l)\cdot\sigma_i(\mathbf{X}_t^l)) && (4.12)\\
&= \sum_{i=1}^{l}(\sigma_i(\mathbf{X}_s^l) - \sigma_i(\mathbf{X}_t^l))^2. && (4.13)
\end{aligned}$$

□

The important fact in the right part of Eq. (4.10) and (4.11) is that we do not rely on the bound of the Frobenius inner product as in the proof for Eq. (4.8) [49, p. 459], because ${\mathbf{L}^l}^T\mathbf{L}^l = \mathbf{I}$. Therefore, we can directly compute the Frobenius inner product of the the diagonal matrices $\mathbf{\Sigma}^l$ and $\mathbf{T}^l$, which is simply the sum of the product of the singular values. Consequently follows for $l+1$ and $(\sigma_{l+1}(\mathbf{X}_s) - \sigma_{l+1}(\mathbf{X}_t))^2 \neq 0$,

$$\left\|\mathbf{X}_s^l - \mathbf{X}_t^l\right\|_F^2 \leq \sum_{i=1}^{l+1}(\sigma_i(\mathbf{X}_s) - \sigma_i(\mathbf{X}_t))^2 < \sum_{i=1}^{q}(\sigma_i(\mathbf{X}_s) - \sigma_i(\mathbf{X}_t))^2 \leq \|\mathbf{X}_s - \mathbf{X}_t\|_F^2 , \tag{4.14}$$

where again $q = min(n, d)$ and $1 < l < q$. The bound applies independently from the projector and allows a direct expression of domain differences in feature variance.

4.2 NYSTRÖM MODEL

Overview. This section contains the description of a Nyström based formulation of the SO approach, subsequently called Nyström Subspace Override (NSO). The procedure is introduced in the following, while its properties are discussed at the end of the section.

Contribution. We propose a Nyström based version of the SO approach, which approximates the subspace and the singular values via Nyström. This version has three main improvements: (i) reduction of computational complexity via Nyström, (ii) implicit dimensionality reduction by reducing sample size requirements of SO, and (iii) noise reduction of values that are close to zero with no large numerical impact.

Description

We assume again that if the source and target datasets are similar, then they follow similar distributions, i. e., if $\mathbf{X}_s \simeq \mathbf{X}_t$ then $p(\mathcal{S}) \simeq p(\mathcal{T})$. If so, the resulting domain kernels will also have similar kernel distributions, which is required if a kernel-based classification model is learned in a Domain Adaptation scenario [79].

The above SO approach creates a bounded subspace via $\mathbf{M}$ and further projects source data into the same subspace as the target one, useful for successful Domain Adaptation [67]. However, two drawbacks are present: The approach needs a complete SVD of both domain datasets, which can be very expensive for large high dimensional datasets, see Newsgroup in Sec. 3.2.2 for an example. Further, the domain-related noise of both domains is also projected into the subspace. For tackling the issues just mentioned, we introduce the Nyström approximation of SO in the following.

For clarity, the following notation will overlap with the previous section but keep things simple. We assume the reader is familiar with Nyström-SVD techniques. Otherwise, the reader may consider Sec. 2.2 for an introduction to the Nyström approximation.

In short, the Nyström SVD technique is a low-rank approximation which decomposes a given matrix $\mathbf{K} \in \mathbb{R}^{n \times d}$ into the constitution

$$\mathbf{K} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{F} \end{bmatrix}, \tag{4.15}$$

with $\mathbf{A} \in \mathbb{R}^{l \times l}$, $\mathbf{B} \in \mathbb{R}^{l \times (d-l)}$, $\mathbf{C} \in \mathbb{R}^{(n-l) \times l}$ and $\mathbf{F} \in \mathbb{R}^{(n-l) \times (d-l)}$. The matrix $\mathbf{A}$ contains the random samples called the landmark matrix. Given $\mathbf{K}$, the singular value decomposition $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, and $\mathbf{C}$, the full SVD of $\mathbf{K}$ is reconstructable, which is similar to the following approach.

Consider $\mathbf{X}_s$ and $\mathbf{X}_t$ with the decomposition as in Eq. (4.15). For a Nyström SVD, we sample from both matrices l rows/columns obtaining landmarks matrices $\mathbf{A}_s = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \in \mathbb{R}^{l \times l}$ and $\mathbf{A}_t = \mathbf{L}\mathbf{T}\mathbf{R}^T \in \mathbb{R}^{l \times l}$. The target data is projected into the subspace as

in Eq. (4.4) via the Nyström technique and keeps only the most relevant data structures via

$$\tilde{\mathbf{X}}_t^l = \tilde{\mathbf{L}}\mathbf{T} = \begin{bmatrix} \mathbf{L} \\ \hat{\mathbf{L}} \end{bmatrix} \mathbf{T} = \begin{bmatrix} \mathbf{L} \\ \mathbf{C}_t\mathbf{R}\mathbf{T}^{-1} \end{bmatrix} \mathbf{T} \in \mathbb{R}^{m \times l}. \tag{4.16}$$

Analogously, the source data could be approximated by $\mathbf{X}_s^l = \tilde{\mathbf{U}}\mathbf{\Sigma} \in \mathbb{R}^{n \times l}$. The Nyström technique is also used to approximate the solution to the optimization problem with $\mathbf{M} = \tilde{\mathbf{L}}\tilde{\mathbf{U}}^{-1}$ and to project the source data into the target subspace via

$$\tilde{\mathbf{X}}_s^l = \mathbf{M}\tilde{\mathbf{U}}\mathbf{\Sigma} = \tilde{\mathbf{L}}\tilde{\mathbf{U}}^{-1}\tilde{\mathbf{U}}\mathbf{\Sigma} = \tilde{\mathbf{L}}\mathbf{\Sigma} \in \mathbb{R}^{m \times l}. \tag{4.17}$$

Hence, it is sufficient to only compute a SVD of $\mathbf{A}_t$ and $\mathbf{A}_s$ instead of $\mathbf{X}_t$ and $\mathbf{X}_s$ with $l \ll m, d, n$ and therefore is considerably lower in computational complexity.

By definition of the Nyström approximation, it is $\tilde{\mathbf{U}}\tilde{\mathbf{U}}^T = \tilde{\mathbf{L}}\tilde{\mathbf{L}}^{-1} = \mathbf{I}$ and $\tilde{\mathbf{L}}$ is an orthogonal basis. Therefore, the subspace projections are orthogonal transformations and fulfill the conditions of Eq. (4.5).

Besides small sample requirements, the major advantage of using the approximated low-rank solution in favor of the optimal solution is that singular values close to zero are set to zero, reducing the noise of the data in the subspace. Therefore the approach focuses on intrinsic data characteristics, which should lead to better classification performance.

Subsequently, this approach is denoted as Nyström Subspace Override (NSO). The matrix $\mathbf{X}_s^l$ is used for source, and $\mathbf{X}_t^l$ is used for testing. The pseudocode is shown Alg. 5.

Algorithm 5 Nyström Subspace Override (NSO)

procedure
Input: $\mathbf{X}_s$ as n sized source matrix; $\mathbf{X}_t$ as m sized target matrix; $\mathbf{Y}_s$ as n sized source label vector; l as number of landmarks parameter;
Output: Adapted source data $\tilde{\mathbf{X}}_s^l\mathbf{Y}_s^a$; approximated target matrix $\tilde{\mathbf{X}}_t^l$
    $\mathbf{X}_s^n, \mathbf{X}_t^n = standardization(\mathbf{X}_s, \mathbf{X}_t)$ ▷ $\mathbf{X}_s, \mathbf{X}_t \sim \mathcal{N}(0,1)$
    $\mathbf{X}_s^a, \mathbf{Y}_s^a = GA(\mathbf{X}_s^n, \mathbf{Y})$ ▷ According to Alg. 3 in Sec. 4.1
    $\mathbf{A}_s = matrix\_decomposition(\mathbf{X}_s^a, l)$ ▷ According to Eq. (2.14)
    $\mathbf{A}_t, \mathbf{C}_t = matrix\_decomposition(\mathbf{X}_t^n, l)$ ▷ According to Eq. (2.14)
    $\mathbf{\Sigma} = SVD(\mathbf{A}_s)$; ▷ Singular Values of landmark matrix of $\mathbf{X}_s$
    $\mathbf{L}, \mathbf{T} = SVD(\mathbf{A}_t)$; ▷ SVD of landmark matrix of $\mathbf{X}_t$
    $\tilde{\mathbf{L}} = \begin{bmatrix} \mathbf{L} & \mathbf{C}_t\mathbf{R}\mathbf{T}^{-1} \end{bmatrix}^T$ ▷ According to Eq. (4.16)
    $\tilde{\mathbf{X}}_t^l = \tilde{\mathbf{L}}\mathbf{T}$ ▷ According to Eq. (4.16)
    $\tilde{\mathbf{X}}_s^l = \tilde{\mathbf{L}}\mathbf{\Sigma}$ ▷ According to Eq. (4.17)
    return $\tilde{\mathbf{X}}_s^l, \mathbf{Y}_s^a, \tilde{\mathbf{X}}_t^l$
end procedure

Properties

The computational complexity of NSO is given by the complexity of Nyström-SVD and the calculation of the respective landmark matrices $\mathbf{A}_s$ and $\mathbf{A}_t$ with complexity $\mathcal{O}(l^2)$. The matrix inversion of diagonal matrix $\mathbf{T}^{-1}$ in Eq. (4.16) can be neglected. Remaining matrix multiplications are $\mathcal{O}(nl^2)$, contributing to an overall complexity of NSO, which is $\mathcal{O}(l^2)$ with $l \ll \{n, m\}$. This makes NSO the fastest subspace Domain Adaptation solution in terms of computational complexity compared to SO and to *compared methods in Sec. 4.4.8*.

We showed that NSO is a valid PCA approximation by Eq. (4.4). It follows by definition of SVD that $\tilde{\mathbf{L}}_s \tilde{\mathbf{L}}_s^T = \tilde{\mathbf{L}} \tilde{\mathbf{L}}^{-1} = \mathbf{I}$ and $\tilde{\mathbf{L}}$ is an orthogonal basis. Therefore, Eq. (4.16) and Eq. (4.17) are orthogonal transformations. In particular Eq. (4.17) transforms the source data into the target subspace and projects it onto the principal components of $\mathbf{X}_t$. As just detailed, the subspace projector $\mathbf{M} = \tilde{\mathbf{L}} \tilde{\mathbf{U}}^{-1}$ fulfils the requirements for theorem 4.1.1 and therefore yields a bounded subspace in the same fashion.

4.3 CLASS-INFORMED MODEL

Overview. This section contains the description of a class-informed formulation of the NSO approach, subsequently called class-informed Nyström Subspace Override (cNSO). The reasoning and the mechanics are detailed in the following, while the properties are discussed at the end of the section.

Contribution. The above NSO algorithm samples Uniform over the source domain. Related work [125] reveals better source domain approximation w.r.t classes, integrated into the NSO method. The sampling has an unintended scaling effect on the data described and tackled in the remaining part. The properties of the approach and the relation of all three approaches to the Domain Adaptation theory are discussed afterward.

Description

We assume again that $\mathbf{X}_s \simeq \mathbf{X}_t$ achieves $p(\mathcal{S}) \simeq p(\mathcal{T})$, which is the goal for the subsequently proposed model. The standard technique to create Nyström landmark matrices is to sample uniformly or find clusters in the data [51]. In supervised classification with more than two classes, class-wise sampling should be utilized to properly include class-depending attributes of a matrix into the approximation [125].

However, a decomposition as in Eq. (4.15), required for Nyström-SVD, is intractable with class-wise sampling because respective matrices are non-square. Let $\mathbf{X}_s \in \mathbb{R}^{n \times d}$ with $n \neq d$ and landmark indices $I = \{i_1, \ldots, i_l\}$, if $n > d$ at least one $i_j > d$ is undefined. Therefore, we sample rows class-wise and obtain $\mathbf{A}_s^d \in \mathbb{R}^{l \times d}$ instead of $\mathbf{A}_s \in \mathbb{R}^{l \times l}$, making it possible to sample from the whole range of source data. The

sampling from target data $\mathbf{X}_t$ is done uniformly row-wise because of missing class information. The resulting singular value decompositions are utilized for successive Nyström approximations:

$$\mathbf{A}_t^d = \mathbf{L}^d \mathbf{T}^d \mathbf{R}^{dT} \quad \text{and} \quad \mathbf{A}_s^d = \mathbf{U}^d \mathbf{\Sigma}^d \mathbf{V}^{dT}. \tag{4.18}$$

However, the possible numerical range of $\mathbf{\Sigma}^d, \mathbf{T}^d$ versus $\mathbf{\Sigma}, \mathbf{T}$ is naturally not the same, which we show in the following using the Gerschgorin Theorem [16]. For this paragraph, we omit the nomenclature for source and target because it applies to both. The Gerschgorin theorem provides a geometric structure to bound eigenvalues to so-called discs for complex square matrices, but also generalizes to none complex square matrices. The work of [3] expands the Gerschgorin circles to so-called Gerschgorin type circles for singular values: Given the matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ with $n, d \geq 1$, the singular values of $\mathbf{X}$ are in the range of

$$s_i = \{p_i \pm |r_i|\}, \quad i = 1, \ldots, n. \tag{4.19}$$

Where $p_i = |x_{ii}|$ and the range r_i is defined as

$$r_i = \max\left\{ \sum_{j=1, j \neq i}^{d} |x_{ij}|, \sum_{j=1, j \neq i}^{n} |x_{ji}| \right\}, \quad i = 1, \ldots, n. \tag{4.20}$$

Let $\mathbf{A}^d \in \mathbb{R}^{l \times d} \subseteq \mathbf{X}$ and , $\mathbf{A} \in \mathbb{R}^{l \times l} \subseteq \mathbf{X}$ be two random sample matrices from $\mathbf{X}$ respectively. Denote $\mathbf{X}_d \in \mathbb{R}^{l \times (d-l)}$ as submatrix from $\mathbf{X}$ beginning at l_{th} column of $\mathbf{X}$ and assume that $\mathbf{X}_d \neq \mathbf{O}$, then the following holds:

$$r_i^d \neq r_i \quad \forall i \in \{1, \ldots, l\}, \tag{4.21}$$

where r_i^d, r_i are the Gerschgorin type bounds of the i_{th} row of $\mathbf{A}^d, \mathbf{A}$ respectively.

As a consequence of Eq. (4.21), it is not an unrealistic scenario to assume that approximated matrices $\mathbf{X}_{(\cdot)}^l$ are scaled different by $\mathbf{\Sigma}^d, \mathbf{T}^d$ and accurate scaling of the singular vectors as expected cannot be guaranteed. Therefore, we apply a post-processing correction and standardize the approximated matrices to transform the data back to zero mean and variance one. The singular vectors also have an approximation error. However, both subspace projections are based on the same transformation matrix, hence making an identical error, and as a result, the error should not affect the classification. The pseudocode of cNSO is shown in Alg. 6.

Properties

If the number of landmarks l is smaller than the number of dimensions d then, cNSO has the same computational complexity as NSO, which is $\mathcal{O}(l^2)$. When a large number of landmarks is necessary that $d < l$, then the complexity is $\mathcal{O}(d^2)$. This makes cNSO theoretically slower than NSO. However, our parameter analysis (Sec. 4.4.7) reveals

Algorithm 6 Class-wise Nyström Subspace Override (cNSO)

procedure
Input: $\mathbf{X}_s$ as n sized source matrix; $\mathbf{X}_t$ as m sized target matrix; $\mathbf{Y}_s$ as n sized source label vector; l as number of landmarks parameter.
Output: Adapted source data $\tilde{\mathbf{X}}_s^l, \mathbf{Y}_s^a$; approximated target matrix $\tilde{\mathbf{X}}_t^l$;
    $\mathbf{X}_s^a, \mathbf{Y}_s^a = GA(\mathbf{X}_s^n, \mathbf{Y}_s)$ ▷ According to Alg. 3 in Sec. 4.1
    $n_c = \frac{l}{|\mathcal{Y}|}$
    for each $c \in Y$ **do** ▷ Class-wise row sampling
        $c_{cur} = |\{\mathbf{x}^s | y(\mathbf{x}^s) = c, \ \forall \mathbf{x}^s \in \mathbf{X}_s^a\}|$
        $\mathbf{A}_s^d = \{\mathbf{A}_s^d\} \cup \{\mathbf{x}^s = \{x_s^i\}_{i=1}^d | \mathbf{x}^s \in \mathbf{X}_s^a \cap p(\mathbf{x}^s) = \frac{1}{n_{cur}} \cap y(\mathbf{x}^s) = c\}^{n_c}$
    end for
    $\mathbf{A}_t^d, \mathbf{C}_t$ = decomposition($\mathbf{X}_t$,l) ▷ Eq. (4.15)
    $\mathbf{\Sigma}^d = SVD(\mathbf{A}_s^d)$;
    $\mathbf{L}^d, \mathbf{T}^d, \mathbf{R}^d = SVD(\mathbf{A}_t^d)$;
    $\tilde{\mathbf{L}} = \begin{bmatrix} \mathbf{L}^d & \mathbf{C}_t \mathbf{R}^d \mathbf{T}^{d^{-1}} \end{bmatrix}^T$ ▷ Eq. (4.16)
    $\tilde{\mathbf{X}}_t^l = \tilde{\mathbf{L}} \mathbf{T}^d$ ▷ Eq. (4.16)
    $\tilde{\mathbf{X}}_s^l = \tilde{\mathbf{L}} \mathbf{\Sigma}^d$ ▷ Eq. (4.17)
    $\tilde{\mathbf{X}}_s^l, \tilde{\mathbf{X}}_t^l$=standardization($\tilde{\mathbf{X}}_s^l$,$\tilde{\mathbf{X}}_t^l$) ▷ Effect as in Fig. 4.1
    return $\tilde{\mathbf{X}}_s^l, \mathbf{Y}_s^a, \tilde{\mathbf{X}}_t^l$
end procedure

that such a large number of landmarks is rarely necessary, and one can rely on NSO complexity in practice.

Out-of-Sample Extension. For unseen target/source samples, e. g., $\mathbf{x} \in \mathbb{R}^d$, the model is expandable, analogously to Eq. (4.16). Based on (4.4), a subspace projection via (approximated) right singular vectors is also valid. Hence, a sample can be projected into the subspace via

$$\mathbf{x}^l = \mathbf{x} \tilde{\mathbf{V}}_t^T = x \begin{bmatrix} \mathbf{R} & \mathbf{T}^{-1} \mathbf{U}^T \mathbf{B}_t \end{bmatrix} \tag{4.22}$$

and be evaluated by an arbitrary classifier learned in the subspace. The out of sample extension is expandable to SO and NSO.

Domain Approximation Error. The difference between the source and target domain after cNSO, i. e., approximation error of source by target domain is again bounded by

$$E_{NSO} = \left\| \tilde{\mathbf{X}}_s^l - \tilde{\mathbf{X}}_t^l \right\|_F^2 \leq \sum_{i=1}^{l+1} (\sigma_i(\mathbf{X}_s) - \sigma_i(\mathbf{X}_t))^2 \leq \|\mathbf{X}_s - \mathbf{X}_t\|_F^2 . \tag{4.23}$$

Where $\sigma_i(\cdot)$ is the i-th singular value in descending order of $\mathbf{X}_s$ and $\mathbf{X}_t$ respectively and $1 < l < min(n, d)$. The proof is analog to Theorem 4.1.1. As in prior LS approaches [57], [69], [92], we want cNSO to minimize the difference between the source and target data. In Eq. (4.23), it is shown that cNSO has a lower norm than the original data and proves

that the matrices are aligned during cNSO, making them numerically more similar. However, similar matrices do not necessarily indicate a good classification performance in accuracy by an arbitrary classifier in a Domain Adaptation setting, which we will see in the following.

Relation to Domain Adaptation Theory. For this paragraph, we assume the reader is familiar with the Domain Adaptation theory (Sec. 2.3.1), especially with theorem 2.3.1 and lemma 2.3.2. The methods proposed in this chapter, namely, SO, NSO, and cNSO, approximate the source data via target subspace. Hence, they minimize domain separability and yield an upper bound for the expected target error shown in the following.

Theorem 4.3.1. *Let $\mathbf{X}_s^l \sim p(\mathcal{S}), \mathbf{X}_t^l \sim p(\mathcal{T})$ in as random i.i.d samples in $\mathbb{R}^{n\times l}$ with rank l and $\|\mathbf{x}\| \leq 1 : \forall \mathbf{x} \in \mathbf{X}_s^l, \mathbf{X}_t^l$. Let $\mathcal{H}$ be a hypothesis space, then for any $\delta \in (0,1)$ and probability with at least $1-\delta$ for every $f, h, h' \in \mathcal{H}$:*

$$\varepsilon_t(f) \leq \varepsilon_s(f) + \left(\frac{2}{n^2} + \frac{1}{\sqrt{n}}\right)\left[\sqrt{\sum_{i=1}^{l}\sigma_i^2(\mathbf{X}_s^l)} + \sqrt{\sum_{i=1}^{l}\sigma_i^2(\mathbf{X}_t^l)}\right] \tag{4.24}$$

$$- \frac{2}{\sqrt{n}}\sqrt{\sum_{i=1}^{l}\sigma_i(\mathbf{X}_s^l)\sigma_i(\mathbf{X}_t^l)} + 2\sqrt{\frac{log(\frac{2}{\delta})}{2n}} + \lambda. \tag{4.25}$$

Proof. First, we observe that $\hat{d}_k(\mathcal{U}_s, \mathcal{U}_t) \leq \hat{d}_k^2(\mathcal{U}_s, \mathcal{U}_t)^{\frac{1}{2}}$ [48]. By theorem 2.3.2, we see that we only need to prove that the eigenspectra yields a valid bound for the squared estimate of MMD. We choose the linear kernel $k(\mathbf{x}, \mathbf{y}) = \mathbf{x}\mathbf{y}^T$ and to fullfil the requirements from definition 2.3.2 for MMD, we limit $\|\mathbf{x}\| \leq 1 : \forall \mathbf{x} \in \mathbf{X}_s^l, \mathbf{X}_t^l$. The sum of a linear kernel necessary for computing $\hat{d}_k^2(\mathbf{X}, \mathbf{X})$ can be upper bounded by the l_1 matrix norm given by

$$\sum_{i=1}^{n}\sum_{j=1}^{n} k(\mathbf{x}_i, \mathbf{x}_j) \leq n \left\|\mathbf{X}\mathbf{X}^T\right\|_1 = n \cdot max_{1\leq i\leq n}\sum_{i=j}^{n} |\mathbf{x}_i\mathbf{x}_j^T|. \tag{4.26}$$

By the relationship of the matrix norms, i. e., $\|\mathbf{X}\|_1 \leq \sqrt{n}\|\mathbf{X}\|_2 \leq \sqrt{n}\|\mathbf{X}\|_F$ the following holds

$$\sum_{i=1}^{n}\sum_{j=1}^{n} k(\mathbf{x}_i, \mathbf{x}_j) \leq n \left\|\mathbf{X}\mathbf{X}^T\right\|_1 \leq n^{1.5} \left\|\mathbf{X}\mathbf{X}^T\right\|_F. \tag{4.27}$$

Using Eq. (4.27) and again under the assumption of $k(\mathbf{x}, \mathbf{y}) = \mathbf{x}\mathbf{y}^T$, $\hat{d}_k^2$ has the upper bound

$$\hat{d}_k^2(\mathbf{X}_s^l, \mathbf{X}_t^l) = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} k(\mathbf{x}_i^s, \mathbf{x}_j^s) + \frac{1}{n^2}\sum_{i=1}^{n}\sum_{i=1}^{n} k(\mathbf{x}_i^t, \mathbf{x}_j^t) - \frac{2}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} k(\mathbf{x}_i^s, \mathbf{x}_j^t), \tag{4.28}$$

$$\leq \frac{n^{1.5}}{n^2}\left[\left\|\mathbf{X}_s\mathbf{X}_s^T\right\|_F + \left\|\mathbf{X}_t\mathbf{X}_t^T\right\|_F - 2\left\|\mathbf{X}_s\mathbf{X}_t^T\right\|_F\right]. \tag{4.29}$$

Because $\mathbf{X}_s^l = \mathbf{L}^l\mathbf{\Sigma}^l$,$\mathbf{X}_t^l = \mathbf{L}^l\mathbf{T}^l$ and $\mathbf{L}^l$ is orthogonal the term is simplified to

$$\hat{d}_k^2(\mathbf{X}_s^l, \mathbf{X}_t^l) \leq \frac{1}{\sqrt{n}} \left[\left\| \mathbf{X}_s \mathbf{X}_s^T \right\|_F + \left\| \mathbf{X}_t \mathbf{X}_t^T \right\|_F - 2 \left\| \mathbf{X}_s \mathbf{X}_t^T \right\|_F \right] \tag{4.30}$$

$$\leq \frac{1}{\sqrt{n}} \left[\left\| \mathbf{\Sigma}^l {\mathbf{\Sigma}^l}^T \right\|_F + \left\| \mathbf{T}^l {\mathbf{T}^l}^T \right\| - 2 \left\| \mathbf{\Sigma}^l {\mathbf{T}^l}^T \right\|_F \right] \tag{4.31}$$

$$\leq \frac{1}{\sqrt{n}} \left[\sqrt{\sum_{i=1}^{l} \sigma_i^2(\mathbf{X}_s^l)} + \sqrt{\sum_{i=1}^{l} \sigma_i^2(\mathbf{X}_t^l)} - 2\sqrt{\sum_{i=1}^{l} \sigma_i(\mathbf{X}_s^l) \cdot \sigma_i(\mathbf{X}_t^l)} \right]. \tag{4.32}$$

Using the same principle we rewrite the following term borrowed from theorem 2.3.2 to

$$\frac{2}{n}\mathbb{E}_{p(\mathcal{S})} \left[\sqrt{tr(\mathbf{K}_S)} \right] + \frac{2}{n}\mathbb{E}_{p(\mathcal{T})} \left[\sqrt{tr(\mathbf{K}_T)} \right] = \frac{2}{n^2} \sqrt{\sum_{i=1}^{l} \sigma_i^2(\mathbf{X}_s^l)} + \frac{2}{n^2} \sqrt{\sum_{i=1}^{l} \sigma_i^2(\mathbf{X}_t^l)}. \tag{4.33}$$

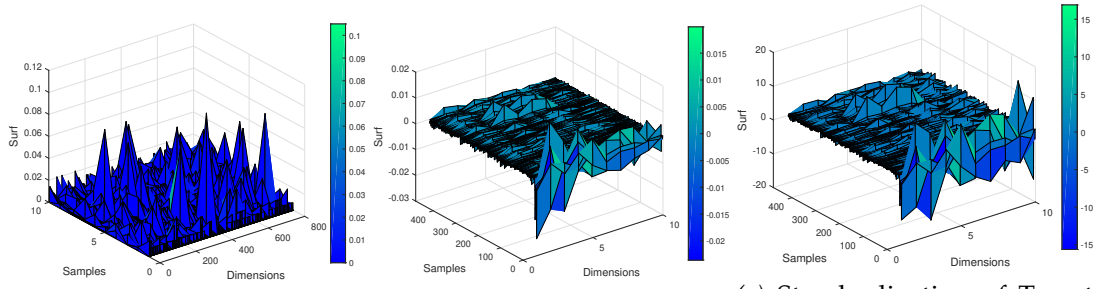
Putting all together and further simplifying completes the proof. □

The theorem 4.3.1 bounds MMD and, with this, the domain separability of source and target. If the MMD is zero, then both domains are almost indistinguishable [79]. By using the SO variants, we control the MMD by the source and target spectra. The size of the spectra is controlled by l. Therefore, l influences the expected error of the classifier f on the target domain in terms of theorem 4.3.1.

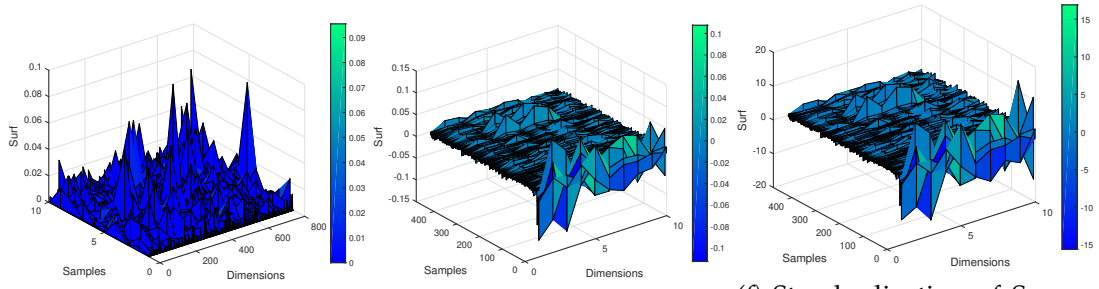
However, as introduced in Sec. 2.3.1, $\lambda = \varepsilon_s(f^*) + \varepsilon_t(f^*)$ is the joint optimal hypothesis f^* on source and target. As pointed out in [162], the minimization of the domain separability to absolutely zero will not lead to perfect prediction results. If λ is large, then the disagreement of source and target labeling functions is large, which could even counterplay low domain separability [36]. Hence, we can observe a minimization of spectra given source and target by the algorithms accordingly to their optimum. However, the performance is also bounded by the disagreement of labeling functions. Therefore, we can not expect zero error on the Domain Adaptation datasets by minimizing the difference in spectra to zero. The plots in Sec. 4.4.7 of Reuters and Office-Caltech Surf show the just mentioned limitation and the reached optimum of the subspace override variants very clearly.

4.4 EXPERIMENTS

Overview. We follow the experimental design typical for shallow Domain Adaptation algorithms on standard benchmark datasets [26], [47], [59], [65], [79]. The experiments contain three main parts: **(i)** a proof of concept validating the mechanics of cNSO on real-world data in Sec. 4.4.1. **(ii)** exhausting experimental validation of the methods on four Domain Adaptation datasets in Sec. 4.4.4. **(iii)** starting from Sec. 4.4.6 until the end of the experimental section, we analyze the properties of SO, NSO, and cNSO,



(a) Target Samples *(b)* Target in Subspace *(c)* Standardization of Target Subspace



(d) Source Samples *(e)* Source in Subspace *(f)* Standardization of Source Subspace

Figure 4.1: Process of *Nyström Subspace Override* with ten landmark applied to *Caltech→Amazon* image dataset encoded with SURF features as a surface plot. Sampling is done one sample per class per domain with a total of 10 per domain. Best viewed on computer display.

including model complexity, parameter behavior, ablation study, and time consumption. Detailed descriptions of the datasets, competitive methods and their parameters as well as the experimental setup are given in Sec 4.4.2 and Sec. 4.4.3.

4.4.1 *Proof of Concept*

In a two-dimensional space and omitting the dimensionality reduction for clarity, the mechanics of SO and NSO appear very similar to GDA in Sec. 3.2.1. Therefore we omit it here. The main result is a transformation of source domain moons to the targets domain moon positions, while the class information is omitted, with the consequence of an overlap of the classes into a moon with a different class. The mechanics of class informed Nyström Subspace Override (cNSO) are illustrated step by step in Fig. 4.1 on the Caltech→Amazon SURF dataset: (i) the first column visualizes the samples of Nyström to create the approximated set of subspace projection operators. (ii) the second column shows the data after the subspace projection. The similarity in structure but dissimilarity in scaling, as discussed in Sec. 4.3, is visible. (iii) the last column shows the data after applying post-correction, leading to a high similarity afterward.

Dataset	Subsets	#Samples	#Features	#Classes
Caltech-256	Caltech (C)	1123	Surf Histograms (800) | Decaf (4096)	10
Office	Amazon (A)	958		
	DSLR (D)	295		
	Webcam(W)	157		

Table 4.1: Dataset characteristics of image Domain Adaptation benchmarks containing datasets and corresponding subsets, numbers of samples, features and classes.

4.4.2 *Dataset*

Summary. The study consists of the benchmark datasets Reuters [25], 20-Newsgroup [65] and Caltech-Office [102]. Reuters and 20-Newsgroup are already described in Sec. 3.2.2. A summary of image and text datasets is shown in Tab. 4.1 and Tab. 3.1. The image datasets are detailed in the following.

Caltech-256- Office[2] [102]: Caltech (*C*) is an extensive dataset of images and initially contains 30607 images within 257 categories. However, in this setting, only 1123 images are used to be related to the Office dataset. We adopt the sampling scheme from [102]. The Office dataset is a collection of images drawn from three sources, which are from *Amazon (A)*, digital SLR camera *DSLR (D)* and *webcam (W)*. They vary regarding camera, light situation, and size but having 31 object categories, e. g., computer or printer, in common. Duplicates and images with more than 15 similar Scale Invariant Feature Transform (SIFT) in common are removed. To get an overall collection of the four image sets, which are considered as domains, categories with the same description are taken. From the Caltech and Office dataset, ten similar categories are extracted: backpack, touring-bike, calculator, head-phones, computer-keyboard, laptop-101, computer-monitor, computer mouse, coffee-mug, and projector. They are the class labels from one to ten.
With this, a classifier should be trained in the source domain, e. g., on projector images (Class One) from amazon (Domain A), and should be able to classify the target image from Caltech (Domain C) to the corresponding image category. The final feature extraction is done with SURF and encoded with 800-bin histograms. We used Decaf features with 4096 dimensions for evaluating shallow Domain Adaptation methods against deep Domain Adaptation methods. Finally, the twelve combination of domain datasets are: **C**→**A**, **C**→**W**, **C**→**D**, **A**→**W**, **A**→**C**, **A**→**D**, **W**→**C**, **W**→**A**, **W**→**D**, **D**→**C**, **D**→**A**, **D**→**W**. They are designed to be trained and tested against each other by the ten labels [102]. An overview of the image dataset is given in Tab. 4.1.

2 https://github.com/ChristophRaab/Office_Caltech_DA_Dataset

4.4.3 *Implementation Details*

Competitive Methods. The proposed SO, NSO, and cNSO algorithms are compared against the same methods as in the previous chapter, namely, Transfer Component Analysis (**TCA**) [45], Joint Distribution Adaptation (**JDA**) [59], Geodesic Flow Kernel (**GFK**) [47] and Subspace Alignment (**SA**) [57]. We replace the augmentation strategy of Geometric Domain Adaptation (Chap. 3) with Gaussian augmentation (**GDA**$_{GA}$), making it semantically valid for using non-text data and to compare against the SO-variants. Further competitive methods are Correlation Alignment (**CORAL**) [92], Cascaded Geometric Covariance Alignment (**CGCA**) [148], Scatter Component Analysis (**SCA**) [100], Easy Transfer Learning (**EasyTL**) [157], Joint Geometrical and Statistical Alignment (**JGSA**) [113] and Manifold Embedded Distribution Alignment (**MEDA**) [134]. Based on the suggestion of the authors, the vanilla Support Vector Machine (SVM) and K-Nearest Neighbors are underlying classifiers. Further, we compare against the following deep Domain Adaptation networks: Joint Adaptation Network (**JAN**) [103], Domain Adaptation Network (**DAN**) [78], Deep Domain Confusion (**DDC**) [68] and Correlation Alignment Network (**CORAL**) [93]. The baseline model for shallow methods is the SVM, and for deep methods, it is the Alexnet.

Parameters. The proposed methods use the SVM as the underlying classification model with $C = 10$ together with the RBF kernel and kernel width of $\sigma = 1$. We set the number of landmarks/subspaces to 500 for text and 50/125 at SURF and decaf features for our proposed methods, respectively. The subspace parameter of JDA is set to 40, for SA to 100, for TCA to 60, for GFK to 40, for MEDA to 20, and for JGSA to 30. The JDA regularization is set to 1 while JGSA s regularization is set to 0.1. All parameters are optimized for providing the best performance over all datasets. The remaining parameters of the approaches, including neural networks, are taken from the respective publications.

Experimental Setup. Unlike the experiments in the previous chapter (Sec. 3.2), we also compare the proposed approaches against deep learning methods on image datasets. For a fair evaluation, we implement the typical experimental design [78] for deep Domain Adaptation, using all available source and target data for the adaptation process for all datasets. The results are based on 5 random runs. However, the Newsgroup dataset is very high dimensional. Therefore, we reduced the dataset by a 2-fold split with two random runs per dataset combination to keep computations feasible.

4.4.4 *Performance Results*

The results are shown per dataset separately. The results on Newsgroup in Tab. 4.2, Reuters in Tab. 4.3, OC with Surf features in Tab. 4.4, OC with decaf and deep DA methods in Tab. 4.5. For clarity, we omitted the performance of SO and NSO in the main performance evaluation because we assume that cNSO is the best representative

Dataset	SVM	TCA	JDA	GFK	SA	CORAL	CGCA	SCA	EasyTL	JGSA	MEDA	cNSO (ours)
Comp→Rec	77.6	78.9	83.1	75.1	78.5	79.4	84.0	56.1	42.2	88.3	49.1	**90.2**
Comp→Sci	71.1	62.0	75.5	64.1	80.2	71.8	73.2	72.4	25.2	78.4	49.2	**98.4**
Comp→Talk	84.4	75.0	87.7	83.8	91.1	90.5	87.0	89.5	41.1	91.2	54.4	**96.7**
Rec→Sci	69.3	79.6	79.0	64.4	81.1	75.0	74.0	71.4	33.8	80.5	50.0	**99.0**
Rec→Talk	74.5	86.6	82.0	72.9	79.7	81.6	77.3	78.3	41.6	80.8	55.0	**96.4**
Sci→Talk	70.9	77.6	70.5	64.2	76.0	74.2	69.0	72.2	41.1	77.7	53.7	**96.4**
Avg.	74.6	76.6	79.6	70.7	81.1	78.7	77.4	73.3	37.5	82.8	51.9	**96.2**

Table 4.2: Mean **accuracy** of shallow DA methods on **Newsgroup** text dataset.

Dataset	SVM	TCA	JDA	GFK	SA	CORAL	CGCA	SCA	EasyTL	JGSA	MEDA	cNSO (ours)
Orgs→People	78.1	79.5	76.6	75.3	**99.9**	77.5	78.0	77.8	39.2	76.5	48.0	99.6
People→Orgs	79.2	82.7	80.0	71.6	**99.9**	78.2	78.6	79.8	37.9	74.2	47.3	98.5
Orgs→Place	69.2	72.9	70.0	60.5	97.3	70.3	70.1	69.8	28.9	72.2	43.2	**98.6**
Place→Orgs	66.3	71.1	65.6	61.5	**97.2**	66.5	67.7	65.3	27.0	64.4	41.4	**97.2**
People→Place	55.7	57.4	57.0	57.5	**97.4**	57.8	57.0	57.3	22.4	52.6	40.9	**97.4**
Place→People	57.4	48.9	60.7	56.2	**97.4**	56.3	54.4	58.2	18.3	55.5	38.5	**97.4**
Avg.	67.7	68.7	68.3	63.8	**98.1**	67.7	67.6	68.0	28.9	65.9	43.2	**98.1**

Table 4.3: Mean **accuracy** of shallow DA methods on **Reuters** text dataset.

Dataset	SVM	TCA	JDA	GFK	SA	CORAL	CGCA	SCA	EasyTL	JGSA	MEDA	cNSO (ours)
C→A	53.1	53.9	55.2	41.8	52.2	52.1	54.1	33.1	50.1	51.8	56.5	**88.5**
C→W	41.7	42.4	46.8	40.7	18.3	38.6	43.1	24.9	49.5	46.1	53.9	**81.0**
C→D	47.8	46.5	49.7	39.5	15.9	36.3	37.6	33.1	48.4	44.6	50.3	**79.0**
A→C	41.7	45.4	43.5	39.0	60.0	45.1	44.9	26.3	43.0	39.7	43.9	**61.5**
A→W	31.9	37.6	44.4	36.9	29.2	44.4	43.9	27.6	40.7	46.1	53.2	**81.0**
A→D	44.6	40.1	31.2	33.1	28.0	39.5	36.3	25.5	38.9	47.8	45.9	**79.0**
W→C	21.2	31.2	31.5	27.4	23.2	33.7	33.8	15.6	29.7	30.2	34.2	**63.5**
W→A	27.6	34.7	31.7	31.2	29.5	35.9	37.6	21.1	35.2	40.0	42.7	**95.8**
W→D	78.3	83.4	92.4	82.8	78.3	86.6	88.5	41.4	77.1	**91.1**	88.5	79.0
D→C	26.5	36.2	32.6	27.2	21.9	33.9	35.4	17.2	31.3	30.3	34.8	**66.6**
D→A	26.2	37.1	36.7	30.9	26.5	37.7	38.9	17.2	31.9	38.2	40.6	**93.1**
D→W	52.5	83.1	88.5	71.9	**89.8**	84.7	87.1	32.5	69.5	91.5	87.5	83.1
Avg.	41.1	47.6	48.7	41.9	39.4	47.4	48.4	26.3	45.4	49.8	52.7	**79.3**

Table 4.4: Mean **accuracy** of shallow DA on **Caltech-Office** with SURF features.

	Traditional Methods												Deep Domain Adaptation				
Dataset	SVM	TCA	JDA	GFK	SA	CORAL	CGCA	SCA	EasyTL	JGSA	MEDA	NSO	Alexnet	DDC-MMD	JAN	DAN	Deep-CORAL
C→A	90.6	90.2	92.4	85.6	92.0	91.5	90.1	48.0	90.2	92.1	**93.5**	88.9	92.5	92.5	93.4	92.9	92.8
C→W	79.0	78.3	81.7	76.6	73.2	78.6	75.9	35.3	76.9	86.4	**93.6**	81.3	74.8	74.9	85.0	86.6	84.3
C→D	83.4	89.8	87.3	82.8	79.0	84.7	85.4	46.1	81.5	92.4	**93.0**	79.0	74.9	74.8	83.0	82.6	78.1
A→C	81.9	81.2	82.7	76.6	83.8	83.2	81.6	43.0	81.7	85.1	**87.5**	61.6	85.3	84.9	84.1	84.1	80.0
A→W	74.2	78.0	72.9	67.8	77.3	75.9	71.2	36.5	74.2	79.0	**88.1**	81.2	65.1	65.2	85.5	84.5	84.3
A→D	80.9	80.9	79.6	73.9	81.5	81.5	74.8	43.6	84.7	79.6	**91.1**	79.0	78.0	75.8	83.3	85.4	65.6
W→C	63.0	69.5	74.0	61.1	76.0	67.9	73.7	27.9	66.3	84.9	**88.3**	63.6	70.9	69.8	78.4	78.6	60.8
W→A	73.8	74.6	79.7	71.2	86.1	76.0	80.5	29.8	73.6	90.3	93.1	**96.2**	80.0	77.6	84.5	83.4	73.6
W→D	**100.0**	**100.0**	**100.0**	**100.0**	98.7	**100.0**	**100.0**	51.1	98.1	**100.0**	**100.0**	79.0	98.8	98.8	99.7	99.5	99.4
D→C	52.7	68.8	80.2	61.2	75.9	68.0	75.5	24.2	69.1	85.0	**87.1**	66.7	77.3	77.8	79.6	78.1	66.5
D→A	62.5	79.7	88.9	69.5	87.3	77.2	86.9	26.2	76.3	91.9	**93.2**	92.8	82.8	82.3	84.4	85.1	77.4
D→W	89.8	97.6	99.3	98.6	95.6	98.3	99.0	33.7	93.9	99.7	**99.0**	83.1	99.0	98.8	98.7	98.6	99.0
Avg.	77.7	82.4	84.9	77.1	83.9	81.9	82.9	37.1	80.5	88.9	**92.3**	79.4	81.6	81.1	86.6	86.6	80.1

Table 4.5: Mean **accuracy** of shallow and deep DA on **Caltech-Office** with Decaf and original images (Deep Learning approaches), respectively.

of the proposed algorithms, which is verifiable at Sec. 4.4.5 showing an Ablation study involving all proposals.

To summarize, our NSO algorithm is the best on Reuters and Newsgroup data. The only competitive algorithm is SA on Reuters data with similar results to ours. SA is also an LS subspace approach. However, SA is outperformed by NSO at Newsgroup. NSO demonstrates its usefulness for large sparse matrices that are given at these datasets. At the OC-Surf dataset, the NSO outperforms given many datasets and has the best mean accuracy. Only at OC-Decaf features, NSO is midfield in performance, but it is still competitive. We assume that the Decaf features are very dense feature matrices in terms of descriptive information, even if the singular values are small. Therefore, the low-rank approximation is contra-productive.

The intriguing part of this evaluation comes with the cross-task evaluation. While SA is very good at Reuters and Newsgroup, it has bad performance on OC datasets. While MEDA and JGSA have poor performance at Reuters and Newsgroup, they are good at OC datasets. Our NSO approach is in three out of four tasks, the recommendable choice showing convincing task-independent performance. In Fig. 4.3, the parameter sensitivity is shown and demonstrates that the parameterization (number of landmarks) of NSO is stable for various tasks, simple to optimize, and supports the Nyström error expectation.

4.4.5 *Ablation Study*

We inspect the performance contribution of the variants of SO and the GDA. First, the exact solution to the optimization problem is called Subspace Override (SO). The approximation with Uniform sampling, described in Sec. 4.2, is evaluated to study the impact of class-wise sampling (Sec. 4.3) on the performance.

The results are given in Tab. 4.6 and show that the Nyström approximation yields the best performance independent of the sampling strategy. This comes from approximating the subspace projection, where small values are likely to be zero, reducing noise further. It shows that the post-transformation correction works as expected at cNSO by being in the same performance range as the SO-variant.

Dataset	GDA_{Ga}	SO	NSO	cNSO
Reuters	95.0	94.8	**97.6**	**98.1**
Newsgroup	98.2	93.0	96.1	**96.2**
CO - Surf	79.2	**79.3**	79.1	**79.3**
CO - Decaf	79.3	79.2	**79.4**	**79.4**
Avg.	87.9	86.2	88.1	**88.3**

Table 4.6: Component evaluation of GDA and SO-variants in mean.

The performance of GDA with Gaussian augmentation outperforms the SO approach. The subspace variants do not necessarily provide a better domain invariance by providing similar mechanics, while the low-rank approximation of the subspace does it. The reason is already discussed and is related to noise reduction in the subspace. Overall, as proposed, the class-wise NSO is recommended because it performances best in the ablation study.

4.4.6 *Model Analysis*

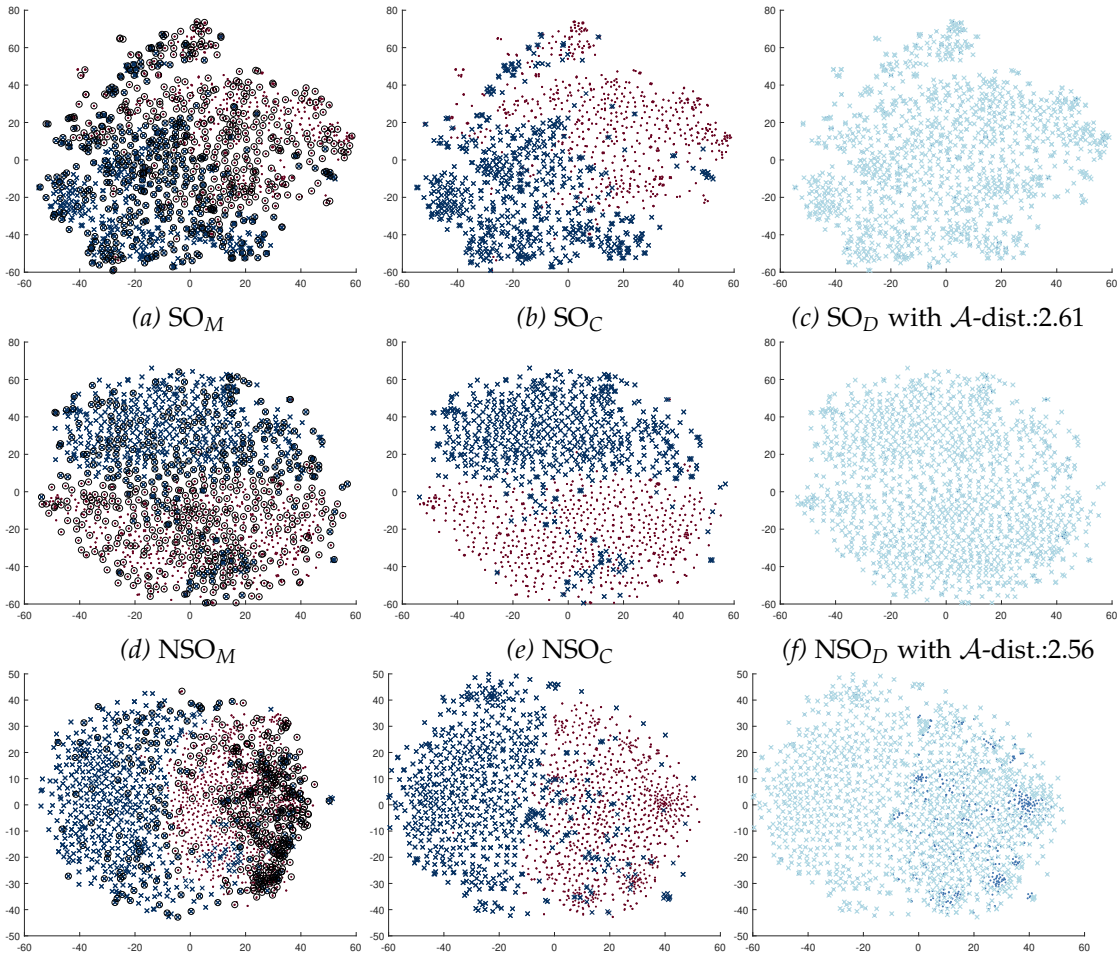
Qualitative. The SO, NSO and cNSO models, trained on **Orgs** and evaluated on **People**, are plotted with TSNE [29] in Fig. 4.2. The first column shows the two-dimensional representation of both Orgs and People, where the black circles mark a support vector, while a red dot "·" marks a source example and a blue "x" shows a target example. The second column is for clarity and shows the same representation with the same coloring. The third column shows the data with domain labels. Further, for every domain plot, we provide the $\mathcal{A}$-distance. The $\mathcal{A}$-Distance [24] is defined as $\mathcal{A} = 2(2 - 1\hat{\varepsilon})$, where $\hat{\varepsilon}$ is the error of a trained domain classifier and is a measure for domain separability. It quantifies the inability of a classifier to distinguish the source and target domain - in this case, the SVM.

The average classification accuracy on the Reuters dataset is 94.8, 97.6, and 98.1, respectively. Hence, we are allowed to assume that the models shown in the plot can achieve high accuracy. However, the models seem very complex, with a large number of support vectors. The SO-variant is weakest at separating both classes, while the Nyström variants can create a decision boundary visibility with the bare eye.

Comparing NSO and cNSO, we can verify that the class-informed model yields a better approximation by obtaining clusters naturally occurring in the data. Both solutions are mapping the source data consequently into the target subspace. However, by preserving these clusters, the intrinsic structure of the source data is kept. Therefore, it seems reasonable that cNSO performs better than NSO and explains the simpler model obtainable using cNSO.

The $\mathcal{A}$-distance is largest at cNSO, while at SO and NSO, the distance is insignificantly different. This means the cNSO is worst at creating an invariant representation where source and target domain are indistinguishable. However, as already mentioned above, the cNSO is best at preserving the data structure, and therefore, the obtained performance is higher. Theoretical reasoning for having a higher $\mathcal{A}$-distance with less classification error as competitive approaches is provided at the end of Sec. 4.3.

Quantitative. We measured the model complexity with the number of model vectors, e. g., support vectors. Our experimental results are shown as mean summarized per dataset group in Tab. 4.7. Note that for a fair comparison, the underlying SVM always has the same parameters. The presented approaches have significantly more complex



(a) SO_M *(b)* SO_C *(c)* SO_D with $\mathcal{A}$-dist.:2.61

(d) NSO_M *(e)* NSO_C *(f)* NSO_D with $\mathcal{A}$-dist.:2.56

(g) $cNSO_M$ *(h)* $cNSO_C$ *(i)* $cNSO_D$ with $\mathcal{A}$-dist.:3.51

Figure 4.2: TSNE [29] representation of Reuters Org→People data of the algorithms SO, NSO and cNSO. <Name>$_M$ shows the support vectors in black circles of SVM trained on the features generated by <Name>, while data is colored by ground truth labels. <Name>$_C$ and <Name>$_D$ show the same representation colored with class and domain labels for better visibility. Plots are optimized minimizing visual artifacts. Standardization at cNSO is also omitted to reduce artifacts. Best viewed on computer display.

Dataset	SVM	SA	GDA_{GA}	SO	NSO	cNSO
Reuters	441.9	**100.9**	1029.8	1030.1	986.2	990.4
Newsgroup	1846.4	**302.6**	3558.8	1794.5	2450.0	2448.4
OC - S	284.5	**238.4**	626.6	625.2	626.0	626.0
OC - D	545.4	**243.6**	608.0	611.5	602.9	601.1
Avg.	779.5	**221.4**	1455.8	1015.3	1166.3	1044.2

Table 4.7: Mean model vectors of a classifier for Reuters, Image and Newsgroup datasets.

models than competitive approaches. The average number of support vectors of the SVM is 779.5, while the SO variants have more than 1000 model vectors. The related SA model is even simpler than vanilla SVM.

The override and scaling procedure cause the drastic increase of model complexity: pairwise distances are very similar for domains and classes, observable in the above plots. Consequently, the SVM needs more support vectors to provide a decision boundary for modeling outliers or complex structures. Comparing cNSO and NSO, we can report that cNSO has, on average, the least complex model. As described above, it can better capture the intrinsic structure of the data, and the SVM needs fewer support vectors to express the decision boundary. Comparing all proposed models, SO provides the least complex one while the GDA_{GA} is the most complex. Both variants use regular SVD. However, the subspace data is easier to describe than in the original feature space.

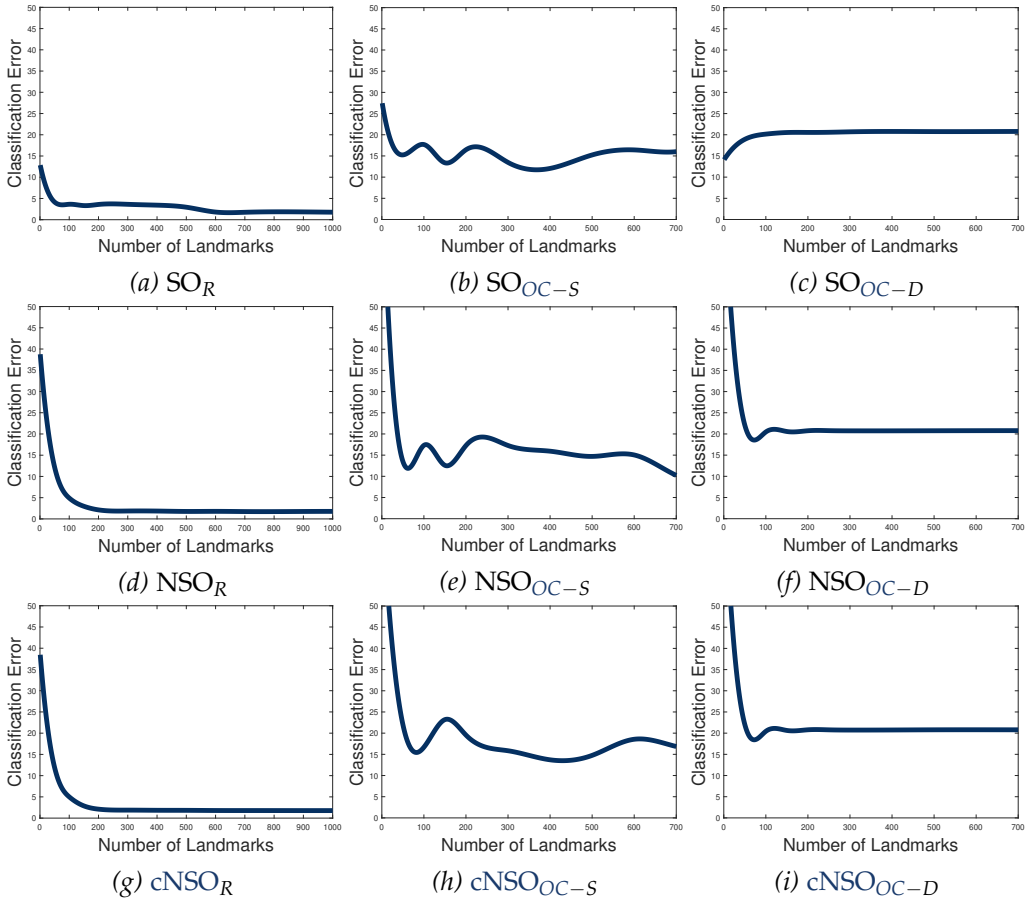
4.4.7 *Parameter Analysis*

The sensitivity of the number of landmarks/subspace-dimension on mean prediction error as the only parameter of Subspace Override (SO), Nyström Subspace Override (NSO), and the class-informed NSO (cNSO) is demonstrated in Fig. 4.3. The comparison is made given Reuters and Office-Caltech Surf and Decaf datasets.

Overall, the SO approach can work with a low number of dimensions. At the same time, the Nyström-variants need at least 100-200 samples to work correctly and is more stable in terms of varying performance as the Nyström approaches. In the following, we call this effect cold start.

At Reuters (R) and Office-Caltech Decaf (OC-D), the cold-start of SO is observable, followed by an alignment of performance of all three approaches. The behavior supports the Nyström error expectation of approximating the real rank of a matrix and low-rank approximation mechanics. Generally, the more information is kept, the better the classification performance until a performance saturation is reached.

At Office-Caltech Surf (OC-S), the results between SO and Nyström variants are different. All three methods fluctuate in performance between roughly 50 to 300



(a) SO_R *(b)* SO_{OC-S} *(c)* SO_{OC-D}

(d) NSO_R *(e)* NSO_{OC-S} *(f)* NSO_{OC-D}

(g) $cNSO_R$ *(h)* $cNSO_{OC-S}$ *(i)* $cNSO_{OC-D}$

Figure 4.3: Relationship between the number of landmarks and mean classification error. <Name>$_R$ shows the performance on Reuters for algorithm name. <Name>$_{OC-S}$,<Name>$_{OC-D}$ shows performance on Office-Caltch with Surf and Decaf features respectively given algorithm name.

dimensions and are more stable in the higher dimensions. A potential explanation could be that noise and redundancies are sampled, leading to a matrix approximation not representing the classes properly. If more characteristic features are sampled and present at classification time, a classifier performs better.

Note that the plots Reuters and OC-D also show the limitations and their optimums of the subspace override variants in terms of the Domain Adaptation theory very clearly, discussed in the properties of cNSO in Sec. 4.3.

Dataset	TCA	JDA	GFK	SA	CORAL	CGCA	SCA	JGSA	MEDA	GDA_{GA}	SO	cNSO
Newsgroup	21.4	4.8	214.4	59.7	705.8	11977.0	59.0	3637.0	3447.0	28.7	70.6	**2.64**
Reuters	6.5	1.5	2.6	3.0	15.4	225.6	14.8	122.1	53.2	1.6	2.3	**0.6**
CO - Surf	3.2	0.9	0.6	0.7	0.4	6.4	12.2	10.8	6.3	0.3	0.6	**0.2**
CO - Decaf	1.8	0.4	1.1	1.3	10.6	99.8	10.3	79.8	45.0	0.5	2.7	**0.2**
Overall	8.2	1.9	54.7	16.2	183.1	3077.2	24.1	962.4	887.9	7.8	19.1	0.9
CO_2 in kg	0.034	0.009	0.271	0.079	0.901	15.186	0.102	4.689	4.389	0.037	0.093	0.004

Table 4.8: Mean computational time in seconds with $C0_2$ emissions of subspace DA methods.

4.4.8 *Time and Ecological Analysis*

The mean time results of the subspace DA methods in seconds are shown in Tab. 4.8. The deep DA methods are not presented as they are unrivaled to the shallow methods. Note that we reduced the sample size of Newsgroup to 25 % of the original size for the time experiments because of the long computational times of some methods.

The experiments show that our NSO approach is task-independent, the fastest algorithm. Compared to recent MEDA, JGSA , and CGCA, the NSO approach needs substantially less time. The related SA approach is also fast, but as theoretically derived, the override of a subspace basis approximated by Nyström leads to a boost in computational performance. In contrast to the proposed GDA and SO approach, cNSO is a magnitude faster, while SO is the slowest. Both SO and GDA rely on an SVD of source and target. However, the reason why SO is slower as GDA is an implementation detail. The GDA is implemented via economy-sized SVD. The SO approach computes the standard SVD to allow the parameter l to be every possible number of subspaces. Therefore SO is naturally slower than GDA. The effect is not present at cNSO because of the Nyström approximation.

The mean amount of CO2 emissions in *kg* are shown in the last line of Tab. 4.8. We followed [143] to compute the results and used the mean running time per Domain Adaptation dataset with the number of runs specified in 4.4.3. The results show that the emissions of cNSO are significantly lower than remaining competitive approaches especially compared to CGCA, JGSA , and MEDA. These approaches contribute to the trend of larger models with longer runtime resulting in larger CO_2 footprints. In summary, the cNSO approach is competitive and efficient and, therefore, should be favored concerning Green AI [188].

4.5 DISCUSSION

We proposed a low-rank domain approximation algorithm called Nyström Subspace Override. It overrides the source basis with the target basis, designed as a domain invariant subspace projector. Due to the affiliation of the projector to the target space, we make sure that both domains lie in the same subspace and can bound domain

differences according to the differences in spectra. The obtained projectors yield an answer for the research question asked at the beginning of the chapter. The proposed approach requires only a subset of domain data from both domains and provides a subspace variant of the Domain Adaptation-related least-squares problem. The Nyström based projection, paired with smart class-wise sampling, showed its reliability and robustness in this study. Validated on common Domain Adaptation tasks and data, it showed a convincing performance. Additionally, NSO has the lowest computational complexity and time consumption than discussed solutions, making the approach favorable for Green AI.

One of the main findings in this chapter is that the difference in eigenspectra, which are the variances on the principal components, bound the domain separability. On the other hand, minimizing the variance of two domains is the moment-matching task. In the next chapter, we bring both topics together and formulate a moment-matching loss function based on the difference of the eigenspectra. Further, we describe how to minimize such a loss with a neural network effectively.

Limitations. We did a short review of the proposed solutions in the light of the Domain Adaptation theory. As pointed out, we only minimize the differences between the marginal distributions, but domain shifts express themselves also in the labeling functions. Consequently, the obtained solution will work very well if and only if the marginal distributions change. If the labeling function, meaning the conditional class probability, becomes a too drastic shift, the methods will fail. When using it, one should keep this in mind. This problem is indeed common for shallow and deep Domain Adaptation, as it also affects CORAL, SA, and TCA and will be the focus of further studies.

Future Outlook. As already mentioned, tackling the differences of labeling distributions will be a future topic and will get more attention from the research community in the next years. Newer Methods [180], [186], [190] already use local structures by modeling metrics and centroids on manifolds to capture the intrinsic labeling of the data. While deep methods share the same problem, they gain more attention, because they simultaneously learn a hypothesis. It turned out to be helpful [78], and shallow methods with simultaneous training and adaptation will be a topic, as already started with [134]. In the early 2010s, we saw a lot of shallow Domain Adaptation papers. However, the significant number has already declined and continues to do so. More and more deep Domain Adaptation methods will be published for two reasons: First, deep learning is easier to prototype, requiring just a standard network, and ideas are easily implemented into existing frameworks. Second, but more importantly, is that using the same adaptation mechanism together with deep learning achieves ad-hoc better results as shallow methods [45], [78]. This, of course, has its reason in the drastic increase of parameters. However currently, better performance is regarded more valuable than a sparse model. The discussion above represents one of the primary research challenges of shallow Domain Adaptation: being competitive to deep learning models while keeping

their sparse solutions.

5 DEEP SPECTRAL DOMAIN ADAPTATION

Summary: The spectral decomposition is well known for its application in low-rank approximation and dimensionality reduction. Besides these applications, the statistical interpretation of singular values provides a valuable tool to express statistical moments on principal components, potentially spanning over multiple feature dimensions. Thus, naturally representing intrinsic structures, making them valuable for moment matching in deep Domain Adaptation.

This chapter will formulate a singular value-based moment matching framework for deep Domain Adaptation and provide analysis for the developed solution. The solution can manipulate feature characteristics and statistical moments simultaneously in favor of Domain Adaptation and is extendable to adversarial domain learning. The result is a network architecture able to outperform current state-of-the-art Domain Adaptation solutions.

Publications: This chapter is based on the following publications.

- C. Raab, P. Meier, and F. Schleif, 'Domain Invariant Representations with Deep Spectral Alignment,' in 28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2020, no. February, pp. 509–514, url. esann-2020-31.pdf.
- C. Raab, P. Väth, P. Meier, and F.-M. Schleif, 'Bridging Adversarial and Statistical Domain Transfer via Spectral Adaptation Networks,' in Proceedings of the Asian Conference on Computer Vision (ACCV), 2021, pp. 457–473, doi. 10.1007/978-3-030-69535-4_28

Source Code: https://github.com/ChristophRaab/thesis/tree/main/dsda/.

The ability to learn sophisticated functions and non-trivial data distributions are some of the main advantages of deep networks. In recent years, this capability has led to a drastic increase in classification accuracy in computer vision [84] and natural language processing [136], making them state-of-the-art models in these areas. The network architectures tend to overfit the given training distribution while showing poor generalization on related distributions. Especially in real application scenarios, the training and test domains are different, and the networks cannot generalize well to the test distribution [93].

A current solution to this problem is to over-generalize the networks by training on massive datasets and then specialize these networks by fine-tuning them to the desired

target domain [135]. However, this solution assumes that there is a sufficient amount of labeled data in the target domain, but in many cases, this cannot be guaranteed [68]. So if fine-tuning is not feasible, deep Domain Adaptation techniques are used to learn a network that achieves good classification results in the target domain [130]. Deep Domain Adaptation makes the following assumptions: (i) there exists a pre-trained over-generalized network. (ii) a small amount of labeled data in a source domain. (iii) a small amount of unlabelled data in the target domain [85]. The goal is to learn a network that achieves good classification results and, at the same time, reduces the differences between the distributions of the source and target domains to apply it in the target domain. In relation to the Domain Adaptation theory [36] and with the above assumptions, we *simultaneously* learn a hypothesis h, minimizing empirical source risk, together with a suitable domain representation $\mathcal{S}$ and $\mathcal{T}$. Hence, the proposals in this chapter strive to minimize domain separability and classifier risk. See Sec. 2.3.1 for an introduction to the theory.

During the training process, the networks learn discriminative features for the classification task and simultaneously learn an invariant representation by minimizing a statistical discrepancy between two or more domains [78], [93], [111]. Statistical adaptation [142] is usually integrated as a regularization term into the network. To some extent, these methods can be interpreted as minimizing the discrepancy between one or more (higher) central moments of the domains [111]. The obtained representation should neglect source-specific domain characteristics such as light and camera settings in an image classification task. The adaptation of moments over domains is called moment matching [76], as a subcategory of statistical adaptation.

The main problem of this techniques [78], [93], [111], [142] is the use of moment matching to align the specific moments of the feature dimension over two domains. This procedure neglects variances spanning over multiple dimensions. It cannot catch the span because every single moment of a feature is aligned on its own. Further, there is no guarantee that these adaptations align the variance spanning over multiple features. On the other hand, the spectral decomposition [17] providing eigen- and singular values, which is linked by definition to variances of the principal axes suiting this purpose and, therefore, will play a vital role in this chapter.

Further, statistical adaptation networks, or moment matching techniques, are naturally restricted in creating invariant features concerning the chosen discrepancy measure. In contrast, domain adversarial neural networks (DANN) [72] consists of a classifier network and a domain classifier on top of the feature extractor, i. e., bottleneck output. The learning process is a min-max game related to GANs [63]. The network feature extractor tries to fool the domain classifier (discriminator) by learning an adversarial representation expected to be invariant to the source and target domain. Supported by the Domain Adaptation theory [36], minimizing the domain classifier loss and reverse propagating the resulting gradient to the feature extractor facilitates learning a transferable representation.

Recent work [142] revealed that DANN type networks focus too much on easily transferable features associated with large singular values, neglecting discriminative features assigned to mid-size singular values. Further, the lower numerical part, i. e., small source singular values, is dedicated to the domain-specific information [141] of the primary learning domain. These results fit very well to a moment matching regularization via singular values as described above, providing a natural way for statistical adaptation and enhancing adversarial learning simultaneously by integrating minor changes in the original framework.

Based on the reasoning just introduced, we will provide an answer to the following question in the course of this chapter: **RQ3: Can spectral decomposition provide a moment matching framework for Domain Adaptation?** By providing an answer to this question, we advance the field of statistical and adversarial Domain Adaptation by the **following contributions:**

- Formulating of the Spectral Loss (SL) as a principal component moment matching regularization technique (Sec. 5.2)
- Extending the proposed Spectra Loss (SL) by relevance weighting and domain adversarial learning (Sec. 5.3)
- Statistical reasoning of singular values in the context of moment matching and adversarial learning in the respective sections.
- Providing an extensive demonstration of the singular value decomposition calculus with the application to the proposed losses in the respective sections.
- The network proposed in Sec. 5.3 is analyzed in the light of the most recent developments of the Domain Adaptation theory [162].

We validate our proposal in an exhausting empirical evaluation and properties analysis in Sec. 5.4. A summary and a discussion about the limitations followed by an outlook is provided at the end in Sec. 5.5. We start the chapter in the following by introducing the background of deep Domain Adaptation, providing notation and most recent and related work.

5.1 BACKGROUND AND RELATED WORK

Deep Domain Adaptation. In deep Domain Adaptation [72], [93], [103], [111], we consider a labeled source dataset $\mathbf{D}_s = \{\mathbf{X}_s, \mathbf{Y}_s\} = \{\mathbf{x}_i^s, y_i^s\}_{i=1}^n \overset{i.i.d.}{\sim} p(\mathcal{S})$ in the source domain $\mathcal{S}$ and an unlabeled target dataset $\mathbf{D}_t = \{\mathbf{X}_t\} = \{\mathbf{x}_j^t\}_{j=1}^m \overset{i.i.d.}{\sim} p(\mathcal{T})$ in the target domain $\mathcal{T}$ with same label space $\forall i, j : y_i, y_j \in \mathcal{Y}$ but different distributions $p(\mathcal{S}) \neq p(\mathcal{T})$. The overall goal is (still) to learn a classifier model, but additionally, it

should generalize to a related target domain. The input feature space $\mathcal{X}$ is the initial representation of the source and target, i. e., $\mathbf{X}_s, \mathbf{X}_t \in \mathcal{X}$.

Initially, we consider a neural network $g : \mathcal{X} \to \mathcal{Y}$ with the parameters θ and given $\mathbf{X}_s$, minimizing a classification loss - most often the cross-entropy $\mathcal{L}_y(g(\mathbf{x}^s;\theta), y^s) = -\sum_{i\in\mathcal{Y}} y_i^s log(g(\mathbf{x}^s;\theta)_i)$. The risk of the network is $\varepsilon[\mathcal{L}_y(g(\mathbf{x}^s;\theta), y^s)]$ and during learning, the empirical risk approximates the expected risk by

$$\min_{\theta} \mathbb{E}[\mathcal{L}_y(g(\mathbf{X}_s;\theta), \mathbf{Y}_s)]. \tag{5.1}$$

The network architecture is composed of multiple hidden layers followed by an output or classification layer. Consider $g(\mathbf{X}_s;\theta)_l = a(f(\mathbf{X}_s;\theta)_l)_l$ as the layer l with an activation function $a(\cdot)_l$ and parameter layer $f(\cdot)_l$ for the given source data and $g(\mathbf{X}_t;\theta)_l$ for the target data analogously. Recent network architectures roughly distinguish between the categories of statistical adaptation [78], [93], [103], [111] and adversarial adaptation [116], [118], [127], [142], [146].

Statistical Adaptation. Related to the model proposed in Sec 5.2, statistical adaptation methods use one or more higher layers, i. e., the fully connected layers of the network, to adapt the output distributions of the (hidden) layers $g(\mathbf{X}_s;\theta)_l$ and $g(\mathbf{X}_t;\theta)_l$[111]. This leads to very individualized approaches. To measure the difference between the output distributions of the network, a divergence measure $dist : g(\mathbf{X}_s;\theta)_l \times g(\mathbf{X}_t;\theta)_l \to \mathbb{R}_0^+$ is addedd to the objective function:

$$\min_{\theta} \mathbb{E}[\mathcal{L}_y(g(\mathbf{X}_s;\theta), \mathbf{Y}_s)] + \eta \cdot dist(g(\mathbf{X}_s;\theta)_l, g(\mathbf{X}_t;\theta)_l). \tag{5.2}$$

The dissimilarity measure is used as a regularization. The parameter $\eta \in [0, \infty)$ controls the trade-off between aligning the statistical divergence and minimizing the classification objective. By setting a high η, the distributions are very closely aligned with each other, but the feature representations are degenerated, causing a high loss [68].

In the following, we will discuss prior related work, minimizing the statistical properties [135] of source and target distributions of the output of some layer. A commonly used dissimilarity measure is the Maximum Mean Discrepancy (MMD) [48], which is the difference in mean of two matrices in a reproducing kernel Hilbert space (RKHS).

In [62], the gradient of the MMD with respect to the parameters is used additionally for the parameter updates. The work in [68] uses the MMD to directly propagate the update step w.r.t data, causing the parameters to change accordingly. The original MMD considers only the prior distribution, while in [103] a joint-MMD was proposed to minimize the conditional distribution discrepancy. The minimization of MMD in the proposed networks can be seen as an alignment of statistical moments given a particular kernel, e. g., RBF-Kernel, of the two domains [76].

The authors of [111] proposed the Central Moment Discrepancy for Domain Adaptation, which strives for explicitly minimizing higher central moments. The CORAL

loss [93], minimizing the difference of the full covariance matrices between two domains. Our proposal is a particular case of CORAL. By aligning *only* the singular spectra of the domains, we align the diagonal of covariances as a side effect.

However, our SL explicitly minimizes a diagonal matrix, which is easier to learn, making it favorable over CORAL. As we will see in Sec. 5.2, minimizing the spectral alignment can minimize the second central moment between domains. Further, we do not rely on a particular kernel matrix nor kernel function, but any positive semi-definite (PSD) kernel can be used. Due to this flexibility, it is also relatively easy to extend, e. g., relevance weighting, as we will see in Sec. 5.3.

Adversarial Adaptation. Related to the model extension in Sec 5.3 where we combine *statistical* and *adversarial* adaptation, we now introduce adversarial learning: let the b_{th} layer of the network be the bottleneck layer and consider the network from the first to the b_{th} layer as the feature extractor $f : \mathcal{X} \to \mathcal{F}$ with parameters θ_f. From the $b + 1_{th}$ layer to the output, let $g : \mathcal{F} \to \mathcal{Y}$ be the classifier network with parameters θ_g, where $\mathcal{F}$ is a latent feature space and $\mathcal{Y}$ is the label space. Usually $b \leq l$ for the domain regularization layer. Additionally, let $d : \mathcal{F} \to \mathcal{C} = \{-1, 1\}$ be a domain classifier with parameters θ_d, predicting the domain of samples.

Adversarial Domain Adaptation yields to minimize the loss of $d(\cdot)$, by propagating the reversed gradients from $d(\cdot)$ to $f(\cdot)$ and trying to confuse $d(\cdot)$ [72]. The Gradient-Reversal-Layer (GRL) is defined as

$$R(x; \lambda) = x \quad \text{and} \quad \frac{\partial R}{\partial x} = -\lambda \mathbf{I}, \tag{5.3}$$

where $\mathbf{I}$ is the identity matrix. The invariant representation is achieved at the saddle point of

$$\min_{\theta_f, \theta_g, \theta_d} \mathbb{E}[\mathcal{L}_y(g(f(\mathbf{X}_s; \theta_f); \theta_g), \mathbf{Y}_s)] + \mathbb{E}[\mathcal{L}_d(d(R[f(\mathbf{X}; \theta_f), \lambda]; \theta_d), \mathbf{Y}_d)] \tag{5.4}$$

where $\mathbf{Y}_d = [\mathbf{1}n, -\mathbf{1}m]$ are the domain labels $\{1, -1\}$ of source size n and target size m and $\mathbf{X} = [\mathbf{X}_s, \mathbf{X}_t]$. Using Eq. (5.3) and Eq. (5.4), the parameters of the feature extractor, domain discriminator and label classifier are updated via

$$\theta_f := \theta_f - \zeta \left(\frac{\partial \mathcal{L}_y i}{\partial \theta_f} - \lambda \frac{\partial \mathcal{L}_d^i}{\partial \theta_f}) \right), \tag{5.5}$$

$$\theta_g := \theta_g - \zeta \frac{\partial \mathcal{L}_y^i}{\partial \theta_g}, \tag{5.6}$$

$$\theta_d := \theta_d - \zeta \frac{\partial \mathcal{L}_d^i}{\partial \theta_d}, \tag{5.7}$$

$$\tag{5.8}$$

where i denotes the i_{th} random batch, ζ is the learning rate within the Stochastic Gradient Descent (SGD). A network using the GRL and the network structure defined above is called a Domain Adversarial Neural Network (DANN) [72].

The vanilla DANN [72] implements the cross-entropy for $\mathcal{L}_d$. Other authors [127] used the Wasserstein distance in $\mathcal{L}_d$ because of the intuitive expression of distribution differences [97]. DANN-type networks have also been extended to *normalized* Wasserstein distances [138].

The Conditional Domain Adaptation Network (CDAN) [118] enriches bottleneck features with class conditional confidence via multi-linear mapping of bottleneck features and classification confidence. The mapping is defined as the tensor multiplication $T : \mathcal{Y} \times \mathcal{F} \rightarrow \mathcal{F}_c$, where the classifier output $\mathbf{G} \in \mathbb{R}^{n^i \times c \times 1}$ extends the bottleneck feature matrix $\mathbf{F} \in \mathbb{R}^{n^i \times 1 \times d_f}$ with

$$T(\mathbf{G}, \mathbf{F}) = \mathbf{G} \otimes \mathbf{F} = \mathbf{T} \in \mathbb{R}^{n^i \times c \times d_f}, \tag{5.9}$$

where d_f is the output dimension of the b_{th} layer, c is the number of classes and n^i is the sample size of the i_{th} random batch. The result of the map is flattened resulting in $\mathbf{T} \in \mathbb{R}^{n^i \times (c \cdot d_f)}$ and fed into the discriminator, i. e., $d(T(\cdot))$. CDAN is the baseline in related networks and is extended in this work due to its good performance. The SDAN [193] integrates the Spectral Normalization (SN) [119] to obtain 1-Lipschitz continuous gradients. SN is also a building block in our network, but SDAN does no statistical alignment during learning.

None of the above adversarial networks explicitly define $dist(\cdot)$ in Eq. (5.4) in the same way as statistical deep learning does it in Eq. (5.2). In this sense, the Batch-Spectral-Penalization (BSP) network is related to us shrinking the first k singular values, given features from $f(\cdot)$, to lower the influence of easily transferable features.

However, our ASAN network explicitly defines $dist(\cdot)$ to align the spectra, which is crucial to be less dependent on the source feature spectrum. In [152], an element-wise comparison of distributions in the label space is implemented and [116] where $l < b$, aligning distributions in low-level filters. Our proposed loss directly modifies adversarial and statistical characteristics in one loss, making it superior to discussed approaches.

5.2 MODEL

Overview. This section contains the description of a moment matching technique based on singular values, which is called Spectral loss (SL) or $\mathcal{L}_S$ in equations. The combination into a neural network is called Deep Spectral Network (DSN). The architectural overview of the DSN network is given in Fig. 5.1. Extended statistical reasoning based on the relationship between singular values and the variance is given in at the beginning of

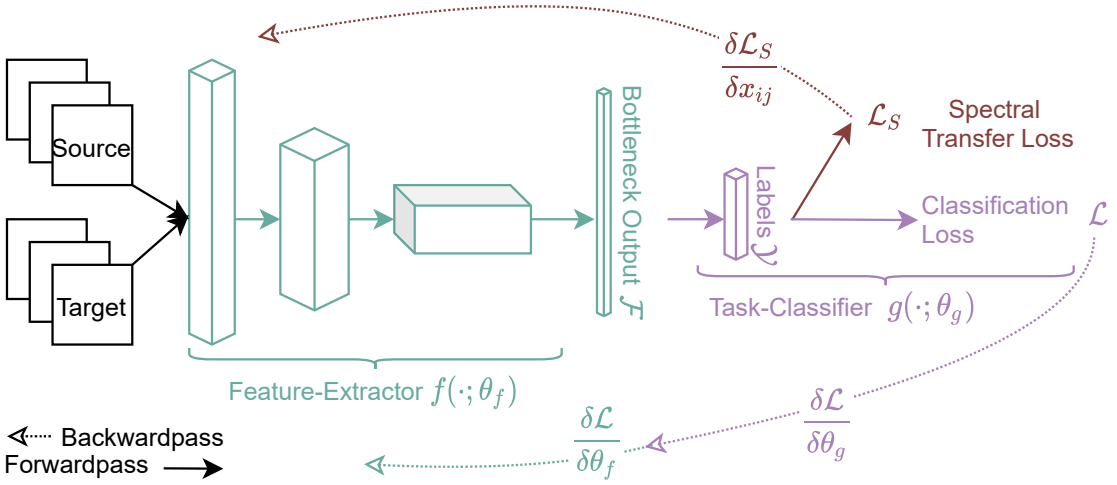


Figure 5.1: Architectural overview of the proposed DSN model, extending a CNN by the $\mathcal{L}_S$ in the label space.

this section. Afterward, the primary contribution of this section is introduced, namely the Spectral Loss and the intuition behind it. The gradient analysis and the learning procedure finish the section.

Contribution. Compared to prior moment matching techniques [78], [93], [103], [111], our model has the following advantages: Variances of principal components instead of feature variances are aligned. It is hyper-parameter-free and, therefore, easier to apply while maintaining competitive performance, verified in the experiments in Sec. 5.4.4. Finally, it does not depend on the choice of a particular kernel function like [78] does.

Reasoning: Statistical Properties of Singular Values

Given the background in Sec 5.1, let $\mathbf{X}_s$ and $\mathbf{X}_t$ be the output of the source and target from layer b or more precise the output of $f(\cdot)$ called bottleneck features. The Singular Value Decomposition (SVD) of these outputs is given with $\mathbf{X}_s = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ and $\mathbf{X}_t = \mathbf{L}\mathbf{T}\mathbf{R}^T$. Here $\mathbf{R}, \mathbf{V} \in \mathbb{R}^{d\times d}$, $\mathbf{U} \in \mathbb{R}^{n\times n}$ and $\mathbf{L} \in \mathbb{R}^{m\times m}$ are matrices. Further, $\mathbf{U}, \mathbf{L}, \mathbf{V}$ and $\mathbf{R}$ are column-orthogonal. $\mathbf{\Sigma}$ is a $n \times d$ matrix wherein all entries $\sigma_{ij} = 0$ iff $i \neq j$ and $\mathbf{T}$ is a $m \times d$ matrix wherein all entries $t_{ij} = 0$ iff $i \neq j$. Furthermore, by σ_e we denote the e-th column singular value in $\mathbf{\Sigma}$.

For a linear covariance function, we can decompose the respective kernel with the eigen value decomposition and the SVD into

$$\mathbf{K} = \mathbf{C}\mathbf{D}\mathbf{C}^{-1} = \mathbf{X}^T\mathbf{X} = (\mathbf{V}\mathbf{\Sigma}\mathbf{U}^T)(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T) = \mathbf{C}\mathbf{\Sigma}^2\mathbf{C}^{-1}, \tag{5.10}$$

where $\mathbf{C}$ are the eigenvectors (right singular vectors of $\mathbf{X}$) and $\mathbf{D}$ are the eigenvalues of $\mathbf{K}$. The singular values $\mathbf{\Sigma}$ of $\mathbf{X}$ are the square root eigenvalues of $\mathbf{K}$. Accordingly, the entries of the diagonal of $\mathbf{D}, \mathbf{\Sigma}^2$ give the variance of the columns of $\mathbf{K}$.

Spectral Loss

Assuming that the expected values $\mathbb{E}_s(\mathbf{X}_s) = 0, \mathbb{E}_t(\mathbf{X}_t) = 0$, we got $tr(\mathbf{\Sigma}^2) = tr(\mathbf{X}\mathbf{X}^T)$. Hence, minimizing the difference between $\mathbf{\Sigma}$ and $\mathbf{T}$ is roughly the same as minimizing the differences on the diagonal of the covariance matrices. The diagonal entries of the covariance matrix contain the feature variance. Therefore, minimizing the Spectral Loss is the same as minimizing the second central moment. The relation allows us to interpret the Spectral Loss as moment matching regularization comparable to [78], [93], [111]. Note that regularization of the Maximum Mean Discrepancy (MMD) is also considered as a moment matching technique [78].

Subsequently, we assume the same batch sizes for both domains during training, i. e., $n = m$. Using the Spectral Loss for Domain Adaptation, we implement for the discrepancy measure $dist(\cdot)$ in Eq. (5.2) the Spectral Loss

$$\mathcal{L}_S = \frac{1}{n}||\mathbf{\Sigma} - \mathbf{T}||_F^2, \tag{5.11}$$

to extend the classification loss. Where $\frac{1}{n}$ is a scaling constant and $||\cdot||_F^2$ denotes the squared Frobenius norm.

Minimizing Eq. (5.11) will align the spectra of source and target. Accordingly, the variances of the outputs in layer l are aligned to each other, and a domain invariant representation for source and target is learned. The loss can be integrated into any layer as a regularization term or simultaneously used in multiple layers, as suggested by [103]. Here we use the proposed approach in the last layer.

Intuition

Alignment in the classification layer as described in this section is rather unintuitive. A natural expression of deep Domain Adaptation is the alignment of bottleneck source and target features because of the conceptual separation between feature extractor and classifier. In the just introduced SL, we explicitly align the classification output of the network, similar to [152], [195].

Recap that in 5.1, we defined $g(\mathbf{X}_s;\theta)_l$ as the output of the l_{th} layer and let us view the output of the network for source and target as two distinct probability distributions $p_{\mathcal{S}}(y|x;\theta)$,$p_{\mathcal{T}}(y|x;\theta)$ with their empirical samples $\mathbf{P}_s = g(\mathbf{X}_s)$,$\mathbf{P}_t = g(\mathbf{X}_t)$ respectively. Further, let μ_2^{Σ}, μ_2^T the variance or second central moment on the principal components on $p_{\mathcal{S}}, p_{\mathcal{T}}$. Because both distributions depend on θ, we learn the following via SGD:

$$\min_{\theta} tr(\Sigma^2 - \mathbf{T}^2) = tr(\mu_2^{\Sigma} - \mu_2^T) = tr(var(\mathbf{P}_s) - var(\mathbf{P}_t)), \tag{5.12}$$

$$if \;\; \mathbb{E}(\mathbf{X}_s), \mathbb{E}(\mathbf{X}_t) = 0, \tag{5.13}$$

which makes it a moment-matching approach. By learning SL we aligning the spread of the output of $p_{\mathcal{S}}$ and $p_{\mathcal{T}}$, which is the variance of class predictions given source

and target samples w.r.t. θ. Hence, we learn a network whose output is invariant of source and target. Note that we choose the squared Frobenius norm due to its practical advantages in formulating the gradient.

Derivative

In the following, the derivative of Eq. (5.11) with respect to the data is given, which allows the optimization of the spectral parameters. Data-driven derivations are common in deep Domain Adaptation at bottleneck or classification layer [76], [93], [103], [111], [152].

First, we show that the derivative exists and where it is defined with respect to the source data, i. e., $x_{ij} \in \mathbf{X}_s$. The obtained solution can be applied straightforward to target data. Hence, the derivative of $\mathcal{L}_S$ with respect to target data is directly formulated without discussing the properties. The context of notation will be clear in the respective formulations. For computing $\frac{\partial \mathcal{L}_S}{\partial x_{ij}}$, it is easy to see that the partial derivative of $||\mathbf{\Sigma} - \mathbf{T}||_F^2$ has the same form like $||\mathbf{\Sigma}||_F^2$, because $\mathbf{T}$ is constant and the problem reduces to compute the partial derivative of all $\sigma_e \in \mathbf{\Sigma}$ with respect to x_{ij}.

Singular Value Derivative. We follow the solution of [8] and analyze

$$\frac{\partial \mathbf{X}_s}{\partial x_{ij}} = \frac{\partial (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)}{\partial x_{ij}} = \frac{\partial \mathbf{U}}{\partial x_{ij}}\mathbf{\Sigma}\mathbf{V}^T + \mathbf{U}\frac{\partial \mathbf{\Sigma}}{\partial x_{ij}}\mathbf{V}^T + \mathbf{U}\mathbf{\Sigma}\frac{\partial \mathbf{V}^T}{\partial x_{ij}}. \tag{5.14}$$

Multiplying with $\mathbf{U}^T$ from the left and with $\mathbf{V}$ from the right we get

$$\mathbf{U}^T\frac{\partial \mathbf{X}_s}{\partial x_{ij}}\mathbf{V} = \mathbf{U}^T\frac{\partial \mathbf{U}}{\partial x_{ij}}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V} + \mathbf{U}^T\mathbf{U}\frac{\partial \mathbf{\Sigma}}{\partial x_{ij}}\mathbf{V}^T\mathbf{V} + \mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\frac{\partial V^T}{\partial x_{ij}}\mathbf{V}. \tag{5.15}$$

Because $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, we have $\mathbf{U}^T\mathbf{U} = \mathbf{I}_n$ and $\mathbf{V}^T\mathbf{V} = \mathbf{I}_d$ with identity matrices $\mathbf{I}$, so Eq. (5.15) becomes

$$\begin{aligned}\mathbf{U}^T\frac{\partial \mathbf{X}_s}{\partial x_{ij}}\mathbf{V} &= \mathbf{U}^T\frac{\partial \mathbf{U}}{\partial x_{ij}}\mathbf{\Sigma} + \frac{\partial \mathbf{\Sigma}}{\partial x_{ij}} + \mathbf{\Sigma}\frac{\partial \mathbf{V}^T}{\partial x_{ij}}\mathbf{V} \\ \mathbf{U}^T\frac{\partial \mathbf{X}_s}{\partial x_{ij}}\mathbf{V} &= \mathbf{\Omega}_U^{ij}\mathbf{\Sigma} + \frac{\partial \mathbf{\Sigma}}{\partial x_{ij}} + \mathbf{\Sigma}\mathbf{\Omega}_V^{ij},\end{aligned} \tag{5.16}$$

where $\mathbf{\Omega}_U^{ij} = \mathbf{U}^T\frac{\partial \mathbf{U}}{\partial x_{ij}}$ and $\mathbf{\Omega}_V^{ij} = \frac{\partial \mathbf{V}^T}{\partial x_{ij}}\mathbf{V}$ depend on x_{ij}. Furthermore, if we multiply both sides of the definition of $\mathbf{\Omega}_U^{ij}$ from the left with $\mathbf{U}$, we get

$$\mathbf{U}\mathbf{\Omega}_U^{ij} = \mathbf{U}\mathbf{U}^T\frac{\partial \mathbf{U}}{\partial x_{ij}} = \frac{\partial \mathbf{U}}{\partial x_{ij}}. \tag{5.17}$$

In other words, by computing the entries of the matrix $\mathbf{\Omega}_U^{ij}$, we get the partial derivative of $\mathbf{U}$ with respect to x_{ij}. If we differentiate $\mathbf{U}^T\mathbf{U} = \mathbf{I}_n$ with respect to x_{ij} we get:

$$\frac{\partial (\mathbf{U}^T\mathbf{U})}{\partial x_{ij}} = 0 \Leftrightarrow \mathbf{\Omega}_U^{ij} + (\mathbf{\Omega}_U^{ij})^T = 0. \tag{5.18}$$

So clearly $\mathbf{\Omega}_U^{ij}$ and analogously $\mathbf{\Omega}_V^{ij}$ are anti-symmetric matrices and all their diagonal elements are 0. Because $\mathbf{\Sigma}$ and $\mathbf{T}$ have only nonzero elements σ_{ii} and t_{ii}, all diagonal entries of the products $\mathbf{\Omega}_U^{ij}\mathbf{\Sigma}$ and $\mathbf{\Sigma}\mathbf{\Omega}_V^{ij}$ are 0. Also, $\frac{\partial \mathbf{X}_s}{\partial x_{ij}} = 0$, for all entries x_{el} with $(e,l) \neq (i,j)$ and is 1 otherwise. Using this, we get that the diagonal elements of the matrices on the left and right side of Eq. (5.16), become

$$\frac{\partial \sigma_e}{\partial x_{ij}} = u_{ie} v_{je}. \tag{5.19}$$

On the other hand if we evaluate the off-diagonal elements of the matrices in Eq. (5.16), we get the linear system

$$\begin{aligned} \sigma_l \mathbf{\Omega}_{U_{el}}^{ij} + \sigma_e \mathbf{\Omega}_{V_{el}}^{ij} &= u_{ie} v_{jl} \\ \sigma_e \mathbf{\Omega}_{U_{el}}^{ij} + \sigma_l \mathbf{\Omega}_{V_{el}}^{ij} &= -u_{il} v_{je}. \end{aligned} \tag{5.20}$$

If all singular values σ_e, σ_l are different, i. e., for $e \neq l$, this system has the unique solution

$$\mathbf{\Omega}_{U_{el}}^{ij} = \frac{\sigma_l u_{ie} v_{jl} + \sigma_e u_{il} v_{je}}{\sigma_l^2 - \sigma_e^2} \quad \textit{and} \quad \mathbf{\Omega}_{V_{el}}^{ij} = \frac{\sigma_l u_{il} v_{je} + \sigma_e u_{ie} v_{jl}}{\sigma_e^2 - \sigma_l^2}. \tag{5.21}$$

This can be used to compute all entries $\mathbf{\Omega}_U^{ij}$ and $\mathbf{\Omega}_V^{ij}$ and hence with Eq. (5.17) the sought-after derivatives $\frac{\partial \mathbf{U}}{\partial x_{ij}}$. In the case that the singular value decomposition of $\mathbf{X}_s$ the matrix $\mathbf{\Sigma}$ has singular values $\sigma_e = \sigma_l$ for $e \neq l$, it is shown in [8] how $\mathbf{\Omega}_U^{ij}$ can be computed with a least squares approach.

Source Derivative. Combining Eq. (5.11) and Eq. (5.16) with the computed matrix $\mathbf{\Omega}_U^{ij}$ we have shown, that the derivative of $\mathcal{L}_S$ is given by the derivative of σ_e w.r.t. x_{ij}. Hence, the derivative of the loss function with respect to source data, i. e., $x_{ij}^s := x_{ij} \in \mathbf{X}_s$, is

$$\begin{aligned} \frac{\partial \mathcal{L}_S}{\partial x_{ij}^s} &= \frac{1}{n} \frac{\partial \sigma_e}{\partial x_{ij}^s} \left[\sqrt{\Sigma_{e=1}^n (\sigma_e - t_e)^2} \right]^2 = \frac{2}{n} \mathbf{\Sigma}_{e=1}^n (\sigma_e - t_e) \frac{\partial \sigma_e}{\partial x_{ij}^s} \\ &= \frac{2}{n} \Sigma_{e=1}^n (\sigma_e - t_e) \cdot u_{ie} v_{je}. \end{aligned} \tag{5.22}$$

Let $\mathbf{u}_i \in \mathbb{R}^{1 \times n}$ be the i_{th} left source singular row vector from $\mathbf{U}$ with the first n entries, $\mathbf{V}_n \in \mathbb{R}^{d \times n}$ the right source singular matrix with the first n columns and using Eq. (5.22), the gradient for a sample $\mathbf{x}_i^s \in \mathbf{X}_s$ with $d < n$ is

$$\frac{\partial \mathcal{L}_S}{\partial \mathbf{x}_i^s} = \frac{2}{n} (\mathbf{\Sigma} - \mathbf{T}) (\mathbf{u}_i \odot \mathbf{V}_n)^T, \tag{5.23}$$

where $\odot$ is the Hadamard matrix product. Further, we consider $(\mathbf{\Sigma} - \mathbf{T}) \in \mathbb{R}^{1 \times d}$ as vector. Note that this derivative assumes that samples size is smaller than feature

dimension, which is no unrealistic assumption, because of small batch sizes shown in Sec. 5.4.3.

Target Derivative. Remember that the target singular value decomposition is $\mathbf{X}_t = \mathbf{LTR}^T$, then it is easy to see that the derivative with respect to the target data, i. e., $x_{ij}^t \in \mathbf{X}_t$, is

$$\frac{\partial \mathcal{L}_S}{\partial x_{ij}^t} = \frac{1}{n}\frac{\partial \sigma_e}{\partial x_{ij}^t}\left[\sqrt{\Sigma_{e=1}^n(\sigma_e - t_e)^2}\right]^2 = -\frac{2}{n}\Sigma_{e=1}^n(\sigma_e - t_e)\frac{\partial t_e}{\partial x_{ij}^t} \tag{5.24}$$

$$= -\frac{2}{n}\Sigma_{e=1}^n(\sigma_e - t_e)\cdot l_{ie}r_{je}. \tag{5.25}$$

Again for a specific target sample $\mathbf{x}_i^t$ the gradient for $n < d$ is defined as

$$\frac{\partial \mathcal{L}_S}{\partial \mathbf{x}_i^t} = -\frac{2}{n}(\mathbf{\Sigma} - \mathbf{T})(\mathbf{l}_i \odot \mathbf{R}_n)^T, \tag{5.26}$$

where, similar as above, $\mathbf{l}_i \in \mathbf{L}$ is the i_{th} left *target* singular row vector with first n elements and $\mathbf{R}_n \in \mathbb{R}^{d\times n}$ the right target singular matrix with the first e columns.

The derivatives of $\mathcal{L}_S$ has a global optimum for $\mathbf{\Sigma} - \mathbf{T} = 0$, while the function is non-convex. Source and target derivatives are pointing in opposite directions leading by definition to a lower spectral loss.

Learning Procedure

Combining the cross-entropy classification loss from Eq. (5.1) and SL from Eq. (5.11), the overall DSN loss is

$$\mathcal{L} = \mathcal{L}_y + \eta \cdot \mathcal{L}_S, \tag{5.27}$$

here η[1] is a trade-off parameter for SL. Both loss functions combined as $\mathcal{L}$ can be minimized simultaneously by training the DSN with stochastic gradient descent via

$$\min_\theta \mathbb{E}[\mathcal{L}_y(g(\mathbf{X}_i^s;\theta), \mathbf{y}_i^s)] + \frac{\lambda}{n}||\mathbf{\Sigma}^i - \mathbf{T}^i||_F^2, \tag{5.28}$$

where $\mathbf{X}_i^s$ and $\mathbf{y}_s^i$ are the i_{th} random source mini batch. Further, $\mathbf{\Sigma}^i$ and $\mathbf{T}^i$ are singular values of the output of the neural network $g(\mathbf{X}_i^s)$ and $g(\mathbf{X}_i^t)$ for a random source and target mini batch respectively.

1 The η parameter is usually optimized via grid search.

5.3 ADVERSARIAL RELEVANCE MODEL

Overview. This section presents an extension of the Spectral Loss, which we call Relevance Spectral Loss (RSL) or $\mathcal{L}_{RS}$ in equations. We call the combination of RSL, CDAN [118] and Spectral Normalization (SN) [119] as Adversarial Spectral Adaptation Network (ASAN). The architectural overview of ASAN is given in Fig. 5.2. The reasoning between singular values and source specific information is discussed at the beginning of the chapter. Afterward, the main proposals, namely the modified RSL plus gradient analysis combined with the Spectral Normalization [119], are presented. The section is finished with the learning procedure, an analysis of theoretical properties, and time complexity.

Contribution. The main disadvantage of the loss presented in Sec 5.2 is that all principal component variances are assumed as equally important for the adaptation task. This is neither true for low-rank approximation [57], shallow Domain Adaptation like Transfer Component Analysis (TCA) [45] nor for deep Domain Adaptation [142]. Therefore, we extend the loss by shrinkage of singular values with source-specific content assumed in the lower numerical ranges of the source spectra concerning a DANN-type network [142]. Implementing the shrinkage mechanism is the same as a relevance reweighting of the principal component variances, prioritizing minimal source influences.
Delimitation. RSL is implemented in the bottleneck layer, different from SL in 5.2 because shrinkage of bottleneck feature is a form of low-rank approximation, and both domain discriminator and classifier use these features. Implementation at the classifier layer would exclude the domain discriminator from the loss.

Reasoning: Relationship of Feature Characteristics and Singular Values

Given $\mathbf{\Sigma}$ and $\mathbf{T}$ produced by a DANN [72] network, each spectrum associated with their features is separable into three areas. Large singular values represent features with high transferability due to high principal angles between their associated subspaces. Features with mid-size singular values contain class discriminative information for learning classification tasks [142]. Small singular values interfere with the generalization ability of a network because of domain-specific features. It is observed by shrinking small source singular values in the fine-tuning process of neural networks leading to low principal angles between the corresponding source and target singular vectors and better generalization [141].

Combining [141], [142] from above and the statistical perspective (Sec. 5.2) with a CNN, singular values express the variance of the feature covariance matrix and are associated with the variance of filter outputs. Therefore, large singular values correspond to high variance, resulting in transferability due to Uniform filter activations in both domains. Mid-size singular values are associated with filters producing class-discriminative information with more class-specific activations. Small singular values

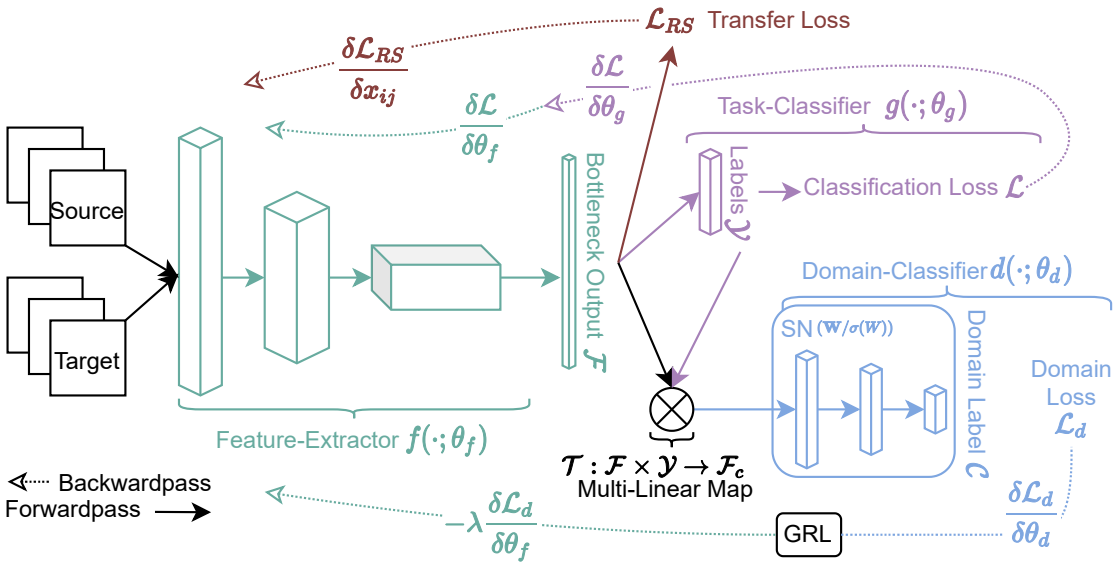


Figure 5.2: Architectural overview of our proposed ASAN model, extending the CDAN [118] network by our $\mathcal{L}_{RS}$ and Spectral Normalization [119] (SN). GRL is the Gradient-Reversal-Layer [72].

express domain-specific features due to the low expressiveness of filters over domain borders.

By aligning the spectra, overly emphasized features, represented by large singular values, are neglected due to the singular values similarities, while the discriminative features are also aligned. Aligning the statistics (Sec. 5.2) and shrink domain-specific signals [141] enables rich adaptation without the need for devastating transferable features [142], which may is counter-intuitive in adaptation tasks.

Relevance Spectral Alignment

The approach [142] of shrinking the k highest or smallest k singular values from both domains for Domain Adaptation comes with the drawback that they are still related to the source spectrum. In some sense, it is counter-intuitive to shrink the influence of highly transferable features in the adaptation task. However, by aligning the most significant singular values, the network does not rely on one spectrum. The expressed variances of the features are the result of two domains.

Following the same reasoning, the classification task is enhanced by not relying on the description of one but the alignment of two spectra. Only the domain-specific contents should be shrunk due to low expressiveness over domain borders [141], which we consider as relevance weighing of domain-specific features. Subsequently, we assume the same batch sizes for both domains during training, i. e., $n = m$. Finally, we define

for the discrepancy measure $dist(\cdot)$ in Eq. (5.2) the proposed Relevance Spectral loss as

$$\mathcal{L}_{RS} = ||(\mathbf{\Sigma}_n - \mathbf{T}_n) + (\mathbf{\Sigma}_k^2 - \mathbf{T}_k)||_F^2 \tag{5.29}$$

where $\mathbf{\Sigma}_n$ and $\mathbf{T}_n$ are the largest $n-k$ singular values and $\mathbf{\Sigma}_k$ and $\mathbf{T}_k$ are the smallest k singular values respectively. Further, $||\cdot||_F^2$ is the squared Frobenius norm. We follow the fine-tuning perspective of [141] and actively shrink only the k smallest source singular values. However, we will show in the following that the respective target singular values are also minimized during learning. The loss can be integrated into any layer as a regularization term or simultaneously used in multiple layers, as suggested by [103]. Here we use our proposed approach in the bottleneck layer.

$\mathcal{L}_{RS}$ has three main benefits. The network aligns the source and target spectrum during learning and depends not only on the source spectrum in $\mathcal{F}$. Assuming the properties as in Sec. 5.2, the network minimizes the difference of the diagonal entries of covariance matrices related to [93], [111], which adapts the second central moment. Domain-specific information [141] in the last k source singular values is minimized during the adversarial adaptation process. Simultaneously, discriminative and transferable features [142] in the largest $n-k$ are encouraged to be large and similar over domain borders.

To summarize, $\mathcal{L}_{RS}$ bridges statistical and enhances adversarial adaptation formulated in one loss, making ASAN favorable in simplicity and theoretical understanding.

Derivative

Recap that although the base network is an domain adversarial network, the RSL is again a statistical adaptation loss with data-driven gradients [93], [103], [111]. Therefore, we follow the suggestions and learn the RSL given the source $\frac{\partial \mathcal{L}_{RS}}{\partial \mathbf{x}_s}$ and target data $\frac{\partial \mathcal{L}_{RS}}{\partial \mathbf{x}_t}$. Given the relationship of singular and eigen values in Eq. (5.10) and following [8], [30], the derivative of a singular value σ and a squared singular value σ^2 w.r.t a sample point is

$$\frac{\partial \sigma_e}{\partial x_{ij}} = u_{ie} v_{je}, \qquad \frac{\partial \sigma_e^2}{\partial x_{ij}} = v_{ie} v_{je}, \tag{5.30}$$

where u_{ie} and v_{je} are the components of the left and right singular vectors of $\mathbf{X}$. The derivative of the e_{th} singular value is given by the j_{th} feature of the i_{th} data-sample and is defined for any $\sigma_i \neq \sigma_j$. For a detailed description of the derivative of singular values see Sec. 5.2([8]) and for eigenvalues see [30].

Source Derivative. Looking once more at Eq. (5.29), the loss needs a derivative for each domain. For the derivative, we assume more feature dimensions as samples, i. e., $n < d$. The source-based derivative of $\mathcal{L}_{RS}$ is given by the derivative of σ_e w.r.t. the source data

x_{ij}^s, therefore

$$\frac{\partial \mathcal{L}_{RSL}}{\partial x_{ij}^s} = \frac{\partial \sigma_e}{\partial x_{ij}^s} \sqrt{\Sigma_{e=1}^{n-k}(\sigma_e - t_e)^2 + \Sigma_{i=k+1}^{n}(\sigma_e^2 - t_e)^2}^2 \tag{5.31}$$

$$= 2\Sigma_{e=1}^{n-k}(\sigma_e - t_e)\frac{\partial \sigma_e}{\partial x_{ij}^s} + 2\Sigma_{e=k+1}^{n}(\sigma_e^2 - t_e)\frac{\partial \sigma_e^2}{\partial x_{ij}^s} \tag{5.32}$$

$$= 2\big(\Sigma_{e=1}^{n-k}(\sigma_e - t_e) \cdot |u_{ie}v_{je}| + \Sigma_{e=k+1}^{n}(\sigma_e^2 - t_e) \cdot |v_{ie}v_{je}|\big). \tag{5.33}$$

For using the derivative for parameter updates, we show gradient for a single sample $\mathbf{x}_s^i$:

$$\frac{\partial \mathcal{L}_{RSL}}{\partial \mathbf{x}_i^s} = 2((\mathbf{\Sigma}_n - \mathbf{T}_n)|\mathbf{u}_i^n \odot \mathbf{V}_n|^T + (\mathbf{\Sigma}_k^2 - \mathbf{T}_k)|\mathbf{v}_i^k \odot \mathbf{V}_k|^T), \tag{5.34}$$

where singular value matrices as defined above, $\mathbf{u}_i^n$ is the i_{th} left source singular rows vector with the first $n-k$ entries, $\mathbf{V}_n \in \mathbb{R}^{d \times n-k}$ containing the first $n-k$ columns of $\mathbf{V}$. $\mathbf{v}_i^k$ is the i_{th} right source singular row vector containing the last k entries of $\mathbf{V}$ and $\mathbf{V}_k \in \mathbb{R}^{d \times k}$ contains the last k columns of $\mathbf{V}$

Target Derivative. Let l_{ie} and r_{je} be the singular vector components of the target data. The target-based derivative of $\mathcal{L}_{RS}$ is analogously given by t_k w.r.t target data x_{ij}^t as

$$\frac{\partial \mathcal{L}_{RSL}}{\partial x_{ij}^t} = -2\big(\Sigma_{e=1}^{n-k}(\sigma_e - t_e) \cdot |l_{ie}r_{je}| + \Sigma_{e=k+1}^{n}(\sigma_e^2 - t_e) \cdot |l_{ie}r_{je}|\big). \tag{5.35}$$

Again, for using the formula for parameter updates, the sample wise gradient is

$$\frac{\partial \mathcal{L}_{RSL}}{\partial \mathbf{x}_i^t} = -2((\mathbf{\Sigma}_n - \mathbf{T}_n)|\mathbf{l}_i^n \odot \mathbf{R}_n|^T + (\mathbf{\Sigma}_k^2 - \mathbf{T}_k)|\mathbf{l}_i^k \odot \mathbf{R}_k|^T), \tag{5.36}$$

where $\mathbf{l}_i^n$ is the i_{th} left source singular rows vector with the first $n-k$ entries and $\mathbf{l}_i^k$ from the same singular vector but the last k entries. $\mathbf{R}_n, \mathbf{R}_k$ are defined analogously to the source definitions.

Discussion. The derivatives give some interesting insights: The absolute value of the singular-vector components is not a derivative result but was added afterward to avoid sign flipping of singular-vectors [28]. Sign flipping leads to wrongly directed gradients. Both derivations point in opposite directions leading to equilibrium at $\mathcal{L}_{RS} = 0$ because every iteration makes a step towards the respective other domain.

The difference between the $n-k$ largest singular values from the source and target is regularized by the component-wise correlation of the row and column spaces given a source or target data point. This prevents a too drastic focus on one of the spaces. The smallest k source singular values are regularized by the component-wise correlation of right singular vectors given feature x_{ij}. Consequently, the smallest k target singular values are mitigated while the target spectrum adapts to the source spectrum.

Stabilize Discriminator Gradients

The domain classifier $d(\cdot)$ is related to the discriminator in GANs [193]. The GAN discriminator suffers from unstable gradients and saturation of training due to perfect predictions [96]. The occurrence of these problems is also possible in training the domain classifier.

We integrate Spectral Normalization [119] in the domain classifier network to tackle this problem by regularizing the gradients to be 1-Lipschitz [119] and avoid the situation where the gradients coming from $d(\cdot)$ are devastating gradients from $\mathcal{L}_{RS}$. Let $\boldsymbol{\theta}$ be parameters in an arbitrary layer of $d(\cdot)$, then SN [119] has the form

$$\boldsymbol{\theta}_{sn} = \frac{\boldsymbol{\theta}}{\sigma(\boldsymbol{\theta})}, \tag{5.37}$$

where $\sigma(\boldsymbol{\theta})$ is the largest singular value of the parameters $\boldsymbol{\theta}$ and $||\boldsymbol{\theta}_{sn}||_{lip} \leq 1$. We implement SN [119] into every layer of the domain classifier, and SN takes place after the forward pass and before gradient propagation. In the following, we call the combination of $\mathcal{L}_{RS}$, SN and CDAN-baseline the *Adversarial Spectral Adaptation Network*. See Fig. 5.2 for an architectural overview.

Learning Procedure

The overall loss of ASAN is composed of the CDAN loss in Eq. (5.4), containing cross-entropy classification loss and the cross entropy domain discriminator loss, together with RSL:

$$\mathcal{L} = \mathcal{L}_y + \mathcal{L}_d + \eta \cdot \mathcal{L}_{RS}, \tag{5.38}$$

where η is the trade-off parameter for RSL. The ASAN loss is again minimized by training the parts of ASAN, namely the feature extractor, domain discriminator and label classifier, simultaneously by stochastic gradient descent:

$$\min_{\theta_f, \theta_g, \theta_d} \mathbb{E}[\mathcal{L}_y(g(f(\mathbf{X}_s^i; \theta_f); \theta_g), \mathbf{Y}_s^i)] \tag{5.39}$$

$$+\mathbb{E}[\mathcal{L}_d(d(R(\mathbf{T}, \lambda); \theta_d), \mathbf{Y}_d^i)] \tag{5.40}$$

$$+\eta \cdot ||(\boldsymbol{\Sigma}_n^i - \mathbf{T}_n^i) + (\boldsymbol{\Sigma}_k^{i\,2} - \mathbf{T}_k^i)||_F^2, \tag{5.41}$$

where i denotes the i_{ih} random batch of source and target data, and $\mathbf{T} = g(f(\mathbf{X}^i; \theta_f) \otimes \theta_g)$, $f(\mathbf{X}^i; \theta_f) \in \mathbb{R}^{n^i \times (c \cdot d_f)}$ is the multi-linear map. Further, $\mathbf{Y}_d^i$ is a label vector in range $\{1, -1\}$ with size of source and target batch and $\mathbf{X}^i = [\mathbf{X}_s^i, \mathbf{X}_t^i]$. Note that SN as in Sec. 5.3 is applied to every parameter matrix in the discriminator after forward pass. Unlike DSN in Sec. 5.2, at ASAN the data for the source and target spectra are obtained from the output of the feature extractor $f(\mathbf{X}_s^i; \theta_f)$ and $f(\mathbf{X}_t^i; \theta_f)$.

Theoretical Properties

For clarity, the following notation will be consistent with the notation of Sec. 2.3.1. We assume the reader is familiar with the Domain Adaptation theory [36], [162]. Otherwise, the reader may consider Sec. 2.3.1 for a brief introduction to it.

The analysis of ASAN relies on the work of Zhao et al. [162], which extends the Domain Adaptation theory of Ben-David et al. [36]. Recap that the theory provides the bound

$$\varepsilon_t(g) \leq \varepsilon_s(g) + d_{\tilde{\mathcal{H}}}(p(\mathcal{S}), p(\mathcal{T})) + min\{\mathbb{E}_{p(\mathcal{S})}[|f_s - f_t|], \mathbb{E}_{p(\mathcal{T})}[|f_s - f_t|]\}, \tag{5.42}$$

where $\varepsilon_s(g), \varepsilon_t(g)$ is the expected risk of a classifier g on source and target domain, $d_{\tilde{\mathcal{H}}}$ as the disagreement of the joint hypothesis $\tilde{h}$ on the source, and target domain while $\tilde{\mathcal{H}} = \{sgn(|h(\mathbf{x}) - h'(\mathbf{x})| - z) \,|\, h, h' \in \mathcal{H}, 0 \leq z \leq 1\}$, where $\mathcal{H}$ is a hypothesis class. The last term is the expected divergence between the optimal labeling function of source f_s and target f_t.

Assuming a fixed representation obtained from $f(\cdot)$, it is shown for CDAN that learning $d(\cdot)$ yields an upper bound [118], i. e.,

$$d_{\tilde{\mathcal{H}}}(p(\mathcal{S}), p(\mathcal{T})) \leq \sup |\varepsilon(d)|, \tag{5.43}$$

where $z = 0$. This is done under the assumption that the hypothesis class $\mathcal{H}_d$ of $d(\cdot)$ is rich enough to contain $d_{\tilde{\mathcal{H}}}$, i. e., $d_{\tilde{\mathcal{H}}} \subseteq \mathcal{H}_d$ for $z = 0$. This is not an unrealistic scenario, since we are able to choose $d(\cdot)$ as a multi-layer perceptron approximating any function [72], [118]. Following this reasoning, we assume that $\mathcal{H}_d$ is also rich enough that $d_{\tilde{\mathcal{H}}} \subseteq \mathcal{H}_d$ for $0 \leq z \leq 1$ and we bound Eq. (5.42) by

$$\varepsilon_t(g) \leq \varepsilon_s(g) + \sup |\varepsilon(d)| + min\{\mathbb{E}_{p(\mathcal{S})}[|f_s - f_t|], \mathbb{E}_{p(\mathcal{T})}[|f_s - f_t|]\}. \tag{5.44}$$

See [72], [118], [162] for technical details about the proof. Hence, minimizing $\sup |\varepsilon(d)|$ by learning $d(\cdot; \theta_d)$, influenced by learning $\mathcal{L}_{RS}$, yields an upper bound for the risk of $g(\cdot)$ on the target domain, i. e., $f(\cdot)$ learns an invariant representation. In particular, the last term of Eq. (5.44), i. e., $min\{\cdot\}$, is the limitation of ASAN, since it does not approximate the labeling functions. Therefore, the performance of DANN-type networks is limited to differences in ground truth labeling of source and target, see Fig. 1 in [162].

Time Complexity

The SVD required for $\mathcal{L}_{RS}$ is $\mathcal{O}(min(p, d_f)^2)$ where p is the batch size and d_f is the dimension of the bottleneck space $\mathcal{F}$. The input space is usually high dimensional due to RGB image data, e. g., $d_{\S} = 3 \times 224 \times 224 = 150.528$ in Resnet50 [84]. Given the computational complexity of convolution [75] and $d_f = 256 \ll d_x$ in ASAN, the computation of SVD at the bottleneck layer does not increase the complexity class of the network. Further, SN uses a *modified* power iteration converging in one iteration [119]. In practice, the time requirements of our extensions are neglectable in comparison to the Resnet50 training time.

5.4 EXPERIMENTS

Overview. We provide the experimental validation of the DSN and ASAN architecture on standard benchmark datasets. The experiments contain four main parts: **(i)** a proof of concept verifying the mechanics of the architectures is given in Sec. 5.4.1. **(ii)** performance analysis in terms of similarity given various moment matching approaches including the proposed DSN is shown in Sec. 5.4.4. **(iii)** evaluation of ASAN in the primary performance overview against state-of-the-art Domain Adaptation networks is presented in Sec. 5.4.5. **(iv)** starting from Sec. 5.4.6 until the end of the experimental section, we analyze the properties of ASAN. Detailed descriptions of datasets, network architecture, competitive methods, experimental setup and parameters are given in Sec. 5.4.2 and Sec. 5.4.3. For selected examples for the described dataset see Sec. 7 in Appendix 7.

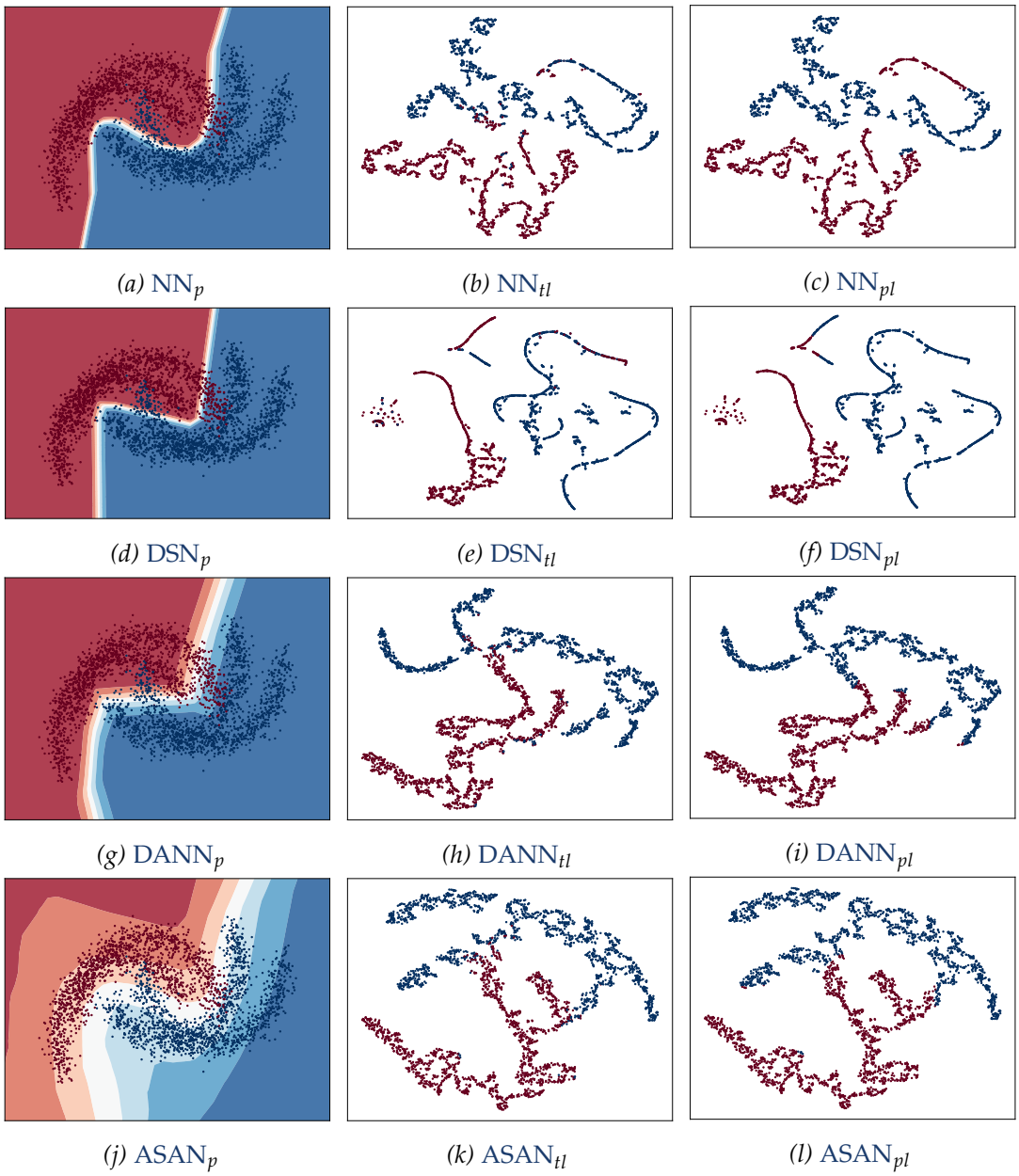
5.4.1 *Proof of Concept*

The proof of concept plot is shown in Fig. 5.3 and demonstrates the behavior of vanilla neural network without any adaptation (NN), DSN, DANN [73] and ASAN per row on the two moon problem [73]. The columns show different levels of representation and data coloring.

The first column, named <Name>$_p$, shows the confidence boundary of the method <Name> with ground truth data coloring. Inspecting the first column, the key difference between ASAN and the remaining models is the confidence levels at domain borders, transitioning from the blue to the red dots. This effect is precisely the intent of relevance-based shrinking of the smallest numerical singular values. The remaining models are too confident at the domain borders, which results in false classification and less accuracy.

The second row shows a TSNE [29] plot of bottleneck features from <Name> with data coloring based on ground truth classes. The last row shows the same representation but model predictions color data. The rows are based on the ablation of adversarial learning and spectral loss: (i) NN, (ii) NN+SL (DSN), (iii) NN+Adversarial (DANN), (iv) NN+Adversarial+RSL (ASAN). For a fair comparison, we used DANN as an adversarial technique rather than CDAN [117], as formulated in Sec. 5.1.The DSN performs better as NN because of the domain alignment observable at the right side kink of the boundary. However, overconfidence is especially present for both NN and DSN, showing the need for using the shrinkage mechanism as formulated in RSL. Comparing DANN and ASAN, we see that the confidence boundary is less confident at NN and DSN domain borders. However, without RSL, the network is unable to effectively lowering its confidence at the domain boundary.

Comparing the second and third columns, we see that ASAN creates the most



(a) NN_p (b) NN_{tl} (c) NN_{pl}

(d) DSN_p (e) DSN_{tl} (f) DSN_{pl}

(g) $DANN_p$ (h) $DANN_{tl}$ (i) $DANN_{pl}$

(j) $ASAN_p$ (k) $ASAN_{tl}$ (l) $ASAN_{pl}$

Figure 5.3: Row-wise demonstration of vanilla neural network (NN), DSN, DANN [73] and ASAN on the two moon problem. <Name>$_p$ shows the confidence boundary of the classifier <Name>, while data is colored based on ground truth labels. The second and third columns show the TSNE [29] of the bottleneck features of the networks. Plots <Name>$_{tl}$ and <Name>$_{pl}$ show the outputs of the bottleneck space. Data is colored based on ground truth labels at *tl* plots and predicted labels *pl* of the network <Name>. Best viewed on computer display.

compact and smooth surface. While DANN also creates smooth surfaces, it cannot correctly classify data points at class boundaries due to the over-confidence at domain boundaries. The plots demonstrate the effect of adversarial domain learning used by ASAN and DANN, verified by the bottleneck features of NN and DSN. Both create representations with mixed classes and create artifacts visible at the right side of NN plots and the left side of DSN plots. Note that TSNE parameters are tuned for the best visual representation concerning all plots.

In summary, this section demonstrates the benefits of SL over vanilla NN because of the domain alignments. Further, we see that SL needs the relevance extension for regularizing over-confidence. Finally, we see the effect of adversarial learning by merging both domains, and that RSL has a positive effect on statistical and adversarial alignment.

5.4.2 *Datasets*

Office-31 [40] is an image data set with 4652 photographs, each assigned to one of the 31 classes. The dataset is divided into the three domains Amazon (**A**), **D**igital **S**ingle-**L**ens **R**eflex camera (**D**) and Webcam (**W**). The Domain Adaptation task is to learn on one domain and test on another. The shift between the domains results from differences in surroundings and camera characteristics. The tasks are **A**→**W**, **D**→**W**, **W**→**D**, **A**→**D**, **D**→**A** and **W**→**A**.

Image-Clef is another image dataset and was released in 2014 as part of the ImageClef Domain Adaptation challenge. It contains 12 common classes from domains Caltech-256 (**C**), ImageNet ILSVRC2012 (**I**), and PAS-CALVOC2012 (**P**) with an total of 1.800 photos. The test setting, again similar to Office-31, is **I**→**P**, **P**→**I**, **I**→**C**, **C**→**I**, **C**→**P** and **P**→**C**.

Office-Home [109] is more comprehensive and more difficult than Office-31 with 65 classes and 15.500 images in total. The dataset domains are Art (**A**), containing painting and sketches, Clipart (**C**), Product (**P**), containing product images without background, and real-world (**R**), containing objects from regular cameras. The test setting for Domain Adaptation is **A**→**C**, **A**→**P**, **A**→**R**, **C**→**A**, **C**→**P**, **C**→**R**, **P**→**A**, **P**→**C**, **P**→**R**, **R**→**A**, **R**→**C** and **R**→**P**.

5.4.3 *Implementation Details*

Architecture. Following the architectural style of CDAN+E (+E : Entropy reweighting)[118], the Resnet50 bottleneck network [84] pre-trained on Imagenet represents the feature extractor $f(\cdot)$. The classifier $g(\cdot)$ on top of $f(\cdot)$ is a fully connected network matching $\mathcal{F}$ to the task depended label space $\mathcal{Y}$, e. g., with 31 dimensions for Office-31. The classifier loss is cross-entropy. The domain classifier $d(\cdot)$ has three fully connected layers, while the first two have RELU activations and dropout, and the last has sigmoid

activation. The input of the discriminator $d(\cdot)$ is the result of the multi-linear map $T(f,g) = f(\mathbf{X}) \otimes g(\mathbf{X})$ [118]. The loss of $d(\cdot)$ is binary-cross-entropy. In the adaptation process, the whole network is fine-tuned on the Domain Adaptation task. Beyond that, we extend the following: all discriminator layers are regularized with SN [119]. The proposed RSL is computed from the source and target bottleneck features and propagated to the feature extractor.

Competitive Methods. We compare our ASAN against the following recent Domain Adaptation networks networks: Domain Adversarial Neural Network (**DANN**) [72], Conditional Domain Adversarial Network (**CDAN**) [118], Batch Spectral Penalization (**BSP**) [142], Spectral Normalized CDAN (**SDAN**) [193], Joint Adaptation Network (**JAN**) [103], Stepwise Adaptive Feature Norm (**SAFN**) [159], Enhanced Transport Distance (**ETN**) [174]. Note that the authors of ETN have not provided the standard deviation in their results. But we still want to show the performance against very recent works. Further our DSN is compared against the moment matching networks Domain Adaptation Network (**DAN**) [78], **JAN** (as above), Deep Domain Confusion (**DDC**) [68] and Correlation Alignment Network (**CORAL**) [93].

Experimental Setup. We follow the standard study protocol [103] for deep Domain Adaptation and use all available labeled source and unlabeled target data. The mean and standard deviation result from three random runs using the best-reported target accuracy per training process. All approaches use the same feature extractor. Reported results of former work are copied in the result tables 5.2, 5.3, and 5.4 if the experimental designs are the same. The classifier and discriminator are trained from scratch with a learning rate of ten times the feature extractors' learning rate. The batch size is set to 32.

Parameters. The ASAN hyperparameters are optimized as in [42] on the Office-31 dataset and set to $k = 11$ and $\eta = 10e^{-3}$ for all datasets. Sec. 5.4.10 gives more details about the tuning and behavior of k. All parameters are trained via mini-batch SGD with a momentum of 0.9. The initial learning rate $\zeta_0 = 0.001$ is modified by a progress based adjustment of $\zeta_p = \zeta_0(1 + \alpha \cdot p)^{-\beta}$, where $\alpha = 10$, $\beta = 0.75$, and $0 \leq p \leq 1$ depending on the training process as suggested by [72]. The λ parameter for the discriminator contribution to the overall loss is progressively increased from 0 to 1 based on $(1 - exp(-\delta \cdot x))/(1 + exp(-\delta \cdot x))$, with $\delta = 10$ as suggested for adversarial architectures [118].

5.4.4 *Performance Results: Moment Matching*

The evaluation analyzes the degree of similarity between moment matching regularization and our Spectral Loss in the DSN. We explicitly used the standard hyperparameters without specific tuning to demonstrate the advantage of parameter-free methods. The study setup is as described in Sec. 5.4.3, but for a fair comparison, we implement our loss and all competitive losses in the classification layer of the network. Hence, the loss

Dataset	**A→W**	**D→W**	**W→D**	**A→D**	**D→A**	**W→A**	Avg.
Resnet [84]	68.4±0.2	96.7±0.1	99.3±0.1	68.9±0.2	62.5±0.3	60.7±0.3	76.1
DDC [68]	72.2±1.7	71.3±2.0	68.5±2.9	89.4±4.3	65.5±2.5	90.4±5.3	76.2
DAN [78]	**79.7±2.6**	70.8±4.1	**71.1±1.7**	88.1±2.8	65.3±2.3	84.7±3.1	76.6
JAN [103]	77.2±1.1	70.9±3.6	73.0±1.1	**93.6±2.9**	63.6±3.6	85.1±4.7	77.2
CORAL [93]	71.7±1.4	**71.8±4.0**	69.6±0.8	98.3±0.8	63.1±4.5	91.7±4.5	77.7
DSN (ours)	75.6±2.1	71.5±2.4	**71.1±2.3**	**93.6±3.9**	**65.6±7.1**	**92.8±2.6**	**78.4**

Table 5.1: Mean prediction **accuracy** with standard deviation of **moment matching networks** on the **Office-31** dataset over three random runs.

is propagated through all layers of the Resnet network. We used 50 epochs to retrain the networks and $\lambda = 0.5$ for all networks. Results are reported in Tab. 5.1 in accuracy (0-100%).

The best performing network is our DSN, but the performance gap in accuracy from ours is, on average, only about 1%. Because CORAL is worse than DSN in the performance experiments, we conclude that minimizing the Spectral Loss contributes more to the adaptation process of the network than minimizing the difference of a covariance matrix. Further, it shows that matching the variance of the principal axes of source and target data contributes more to the adaptation process than matching the feature variance of the domains.

The minimization of non-linear kernel measures, as in DDC or DAN, does not necessarily lead to better adaptation capacities. In fact, without hyperparameter tuning, DDC and DAN perform just slightly better than Resnet. The tuning of the DSN is not expensive because it has no tunable parameters besides the regularization parameter, which makes it easier to apply than other approaches.

To summarize, the Spectral Loss regularization is a favorable moment matching technique due to no hyperparameters and comparable performance to other moment matching techniques. In the next section, we follow the state-of-the-art and implement the RSL, which is an extension of SL, into bottleneck layers of the Conditional Adversarial Network [118], and achieve better prediction performance as state-of-the-art networks.

5.4.5 *Performance Results: State of the Art*

We report the experiments for Office-31 in Tab. 5.2, for Image-Clef in Tab. 5.3, and Office-Home in Tab. 5.4 as accuracy (0-100%). Overall, the ASAN architecture outperforms the compared algorithms in two out of three datasets while having the overall best mean performance.

The results are obtained by optimizing the parameters only on the Office-31 dataset.

This shows the robustness of the performance of our ASAN in terms of parameter sensitivity across changing tasks, making it a stable approach and easily applicable to related real-world scenarios.

At Office-31, ASAN reports the best performance at four out of six comparisons while showing the best mean performance. The second-best performing algorithm is SDAN, which also relies on CDAN and SN. However, due to no additional alignment, our ASAN is superior to SDAN by learning a bottleneck space, aligning both domain spectra. Further, by creating an invariant representation via shrinking the first k singular values, the BSP approach seems not to be competitive with ASANs spectral alignment (RSL).

At Image-Clef, ASAN is only second-best in performance. However, better than related CDAN and SDAN. This suggests that learning our proposed RSL within ASAN improves CDAN, leading to better performance than related methods. At Office-Home, ASAN demonstrates outstanding performance by outperforming in ten out of twelve tasks. The results are comparable with the results of Office-31. The ASAN is best, ETN is second, and SDAN is third in mean performance. Again, RSL, combined with SN, is superior to related approaches such as BSP, SDAN, or CDAN and, further, outperforms very recent benchmark performances of ETN.

Office-31 and Office-Home experiments demonstrate the ASANs parameter robustness: the parameters optimized on Office-31 are used in different but related tasks and show robust generalization capacities. Robust parametrization and excellent performance make our ASAN favorable.

5.4.6 *Convergence and Spectral Analysis*

We report the convergence behavior of our RSL within the ASAN architecture compared to related networks in Fig. 5.4. The data is obtained by training on **A**→**W** from Office-31 dataset.The $\mathcal{A}$-Distance [24] is defined as $\mathcal{A} = 2(2 - 1\hat{\varepsilon})$, where $\hat{\varepsilon}$ is the error of the

Dataset	**A→W**	**D→W**	**W→D**	**A→D**	**D→A**	**W→A**	Avg.
Resnet [84]	68.4±0.2	96.7±0.1	99.3±0.1	68.9±0.2	62.5±0.3	60.7±0.3	76.1
DANN (2015) [72]	82.0±0.4	96.9±0.2	99.1±0.1	79.7±0.4	68.2±0.4	67.4±0.5	82.2
JAN (2017) [103]	85.4±0.3	97.4±0.2	99.8±0.2	84.7±0.3	68.6±0.3	70.0±0.4	84.3
CDAN (2018) [118]	93.1±0.2	98.2±0.2	**100±0**	89.8±0.3	70.1±0.4	68.0±0.4	86.6
BSP (2019) [142]	93.3±0.2	98.2±0.2	**100±0**	93.0±0.2	73.6±0.3	72.6±0.3	88.5
SDAN (2020) [193]	95.3±0.2	98.9±0.1	**100±0**	**94.7±0.3**	72.6±0.2	71.7±0.2	88.9
ETN (2020) [174]	92.1	**100.0**	**100.0**	88.0	71.0	67.8	86.2
ASAN (ours)	**95.6±0.4**	98.8±0.2	**100±0**	94.4±0.9	**74.7±0.3**	**74.0±0.9**	**90.0**

Table 5.2: Mean prediction **accuracy** with standard deviation on the **Office-31** dataset over three random runs.

Dataset	**I→P**	**P→I**	**I→C**	**C→I**	**C→P**	**P→C**	Avg.
Resnet [84]	74.8±0.3	83.9±0.1	91.5±0.3	78.0±0.2	65.5±0.3	91.2±0.3	80.7
DANN (2015) [72]	75.0±0.6	86.0±0.3	96.2±0.4	87.0±0.5	74.3±0.5	91.5±0.6	85.0
JAN (2017) [103]	76.8±0.4	88.0±0.2	94.7±0.2	89.5±0.3	74.2±0.3	91.7±0.3	85.8
CDAN (2018) [118]	77.7±0.3	90.7±0.2	97.7±0.3	91.3±0.3	74.2±0.2	94.3±0.3	87.7
SAFN (2019) [159]	78.0±0.4	91.7±0.4	96.2±0.1	91.1±0.6	77.0±0.2	94.7±0.1	88.1
SDAN (2020) [193]	78.1±0.2	91.5±0.2	97.5±0.2	92.1±0.3	76.6±0.3	95.0±0.1	88.4
ETN (2020) [174]	**81.0**	91.7	**97.9**	**93.3**	**79.5**	**95.0**	**89.7**
ASAN (ours)	78.9±0.4	**92.3±0.5**	97.4±0.5	92.1±0.3	76.4±0.7	94.4±0.2	88.6

Table 5.3: Mean prediction **accuracy** with standard deviation on the **Image-clef** dataset over three random runs.

Dataset	**A→C**	**A→P**	**A→R**	**C→A**	**C→P**	**C→R**	**P→A**	**P→C**	**P→R**	**R→A**	**R→C**	**R→P**	Avg.
Resnet [84]	34.9	50.0	58.0	37.4	41.9	46.2	38.5	31.2	60.4	53.9	41.2	59.9	46.1
DANN [72]	45.6	59.3	70.1	47.0	58.5	60.9	46.1	43.7	68.5	63.2	51.8	76.8	57.6
JAN [103]	45.9	61.2	68.9	50.4	59.7	61.0	45.8	43.4	70.3	63.9	52.4	76.8	58.3
CDAN [118]	49.0	69.3	74.5	54.4	66.0	68.4	55.6	48.3	75.9	68.4	55.4	80.5	63.8
BSP [142]	52.0	68.6	76.1	58.0	70.3	70.2	58.6	50.2	77.6	72.2	59.3	81.9	66.3
SDAN [193]	52.0	72.0	76.3	59.4	71.7	72.6	58.6	52.0	79.2	71.6	58.1	82.8	67.1
ETN [174]	51.3	71.9	**85.7**	57.6	69.2	**73.7**	57.8	51.2	79.3	70.2	57.5	82.1	67.3
ASAN	**53.6**	**73.0**	77.0	**62.1**	**73.9**	72.6	**61.6**	**52.8**	**79.8**	**73.3**	**60.2**	**83.6**	**68.6**

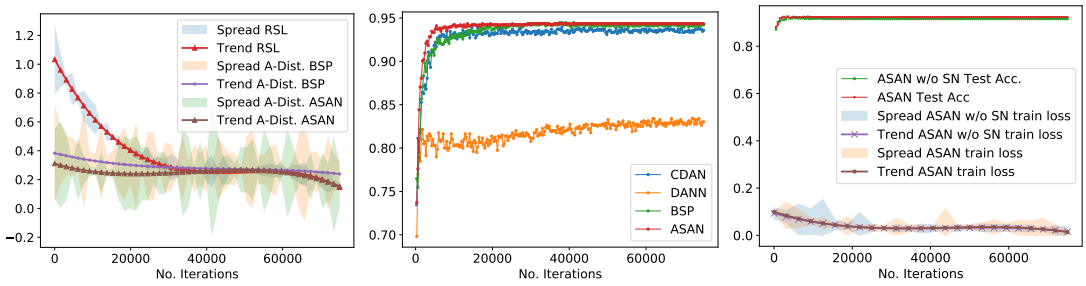
Table 5.4: Mean prediction **accuracy** on the **Office-Home** dataset over three random runs.

trained domain classifier. The $\mathcal{A}$-Distance is related to the $d_{\tilde{\mathcal{H}}}(\mathcal{S}, \mathcal{T})$ in Eq. (5.43) and measures the domain classifiers inability to distinguish the source and target domain. In contrast, a low $\mathcal{A}$-Distance is an indicator for an invariant representation [24].

The proposed RSL and the $\mathcal{A}$-Distance of our ASAN and BSP are shown in 5.4a, which we compare due to the commonality of manipulating the feature spectra. The interpolated lines (red, purple, brown) show the overall learning trend, while the colored areas (blue, orange, green) show the fluctuation during learning. We observe that our network learns a better invariant representation via an almost all-time lower $\mathcal{A}$-Distance by not relying on only one spectrum. The plot shows that spectral differences represented by RSL are effectively reduced. Interestingly, the trend curves of the $\mathcal{A}$-Distance of ASAN and the RSL are similar in shape, allowing the presumption that learning RSL is related to learning an invariant representation, i. e., minimizing the $\mathcal{A}$-Distance.

Figure 5.4b represents the target accuracy during learning. It is observable that the ASAN network converges very fast in a higher target accuracy than related approaches. Further, the saturation is very stable and practically does not change once reached. This behavior of ASAN is related to the Spectral Normalization by giving well-defined gradients back to the feature extractor.

This assumption is verified in Fig. 5.4c, where the learning and evaluation process



(a) A-Distance & RSL *(b)* Target Acc. *(c)* Source loss & Test Acc.

Figure 5.4: Learning process of ASAN compared to related networks over time given **A**→**W** images from Office-31 dataset. Best viewed on computer display.

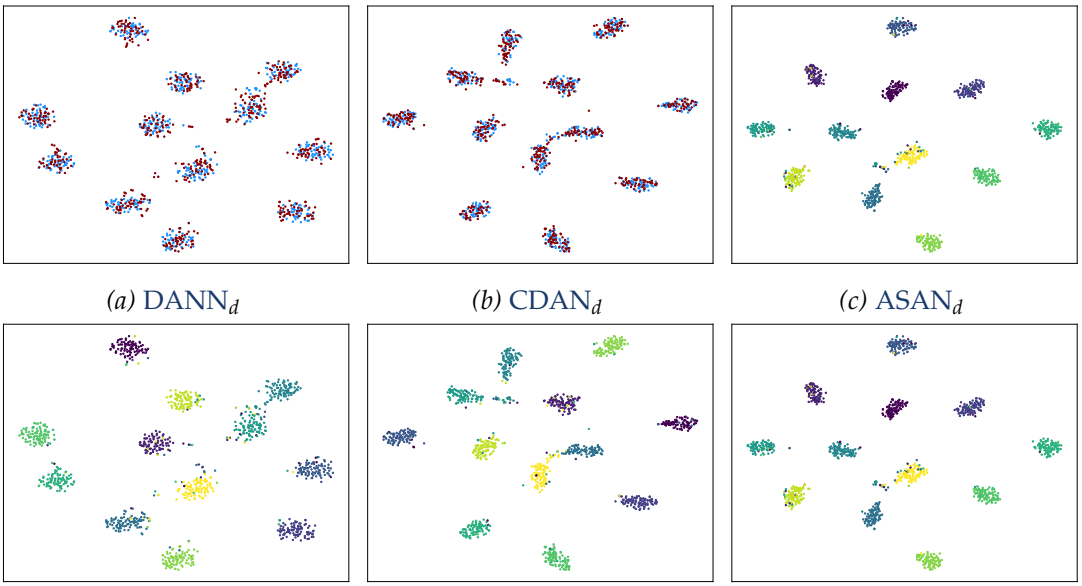
of ASAN itself and ASAN without Spectral Normalization (ASAN w/o SN) is plotted. The target accuracy of ASAN (red) remains stable while the accuracy of ASAN w/o SN (green) has a decline in target accuracy after reaching the best performance similar to ASAN. In contrast, ASAN remains stable at high accuracy. The trends of train loss (brown for ASAN and purple for ASAN w/o SN) show an almost all-time lower learning loss of ASAN. The fluctuation of the train losses shows that ASAN (blue) is more stable than ASAN w/o SN (orange) and, most of the time, is lower in value.

5.4.7 *Feature Analysis*

We evaluate the empirical feature representation of the bottleneck features given **A**→**W** images from the Office-31 dataset. The result is reported in Fig. 5.5 based on TSNE [29]. The plot is split into two parts: the top row (Fig. 5.5a - 5.5c) is a scatter plot of the bottleneck features of trained DANN, CDAN, and ASAN colored with ground truth domain labels. Blue shows the source, and red shows the target domain.

ASAN shows the superiority of creating a domain invariant representation by almost perfectly assigning all red points to blue clusters compared to CDAN and DANN. The bottom row (Fig. 5.5d - 5.5f) shows the same representation but with classification labels. The class-label plots show that ASAN representations are easily classifiable by a neural network.

However, some points are still located in the wrong cluster, representing the limitation of ASAN described in Sec. 5.3. The ASAN, DANN, and CDAN do not approximate the label distribution of the target during learning. Therefore, the target accuracy of ASAN is bounded by the distribution differences of label distributions [162]. As a result, the label distribution difference is directly related to the remaining miss-classified samples. However, as shown in the performance evaluation (Sec. 5.4.5), ASAN performs considerably better than the remaining networks and is, therefore, the preferable choice. Additional results and analysis on different datasets are shown in Appendix 7.



(a) DANN$_d$ *(b)* CDAN$_d$ *(c)* ASAN$_d$

(d) DANN$_c$ *(e)* CDAN$_c$ *(f)* ASAN$_c$

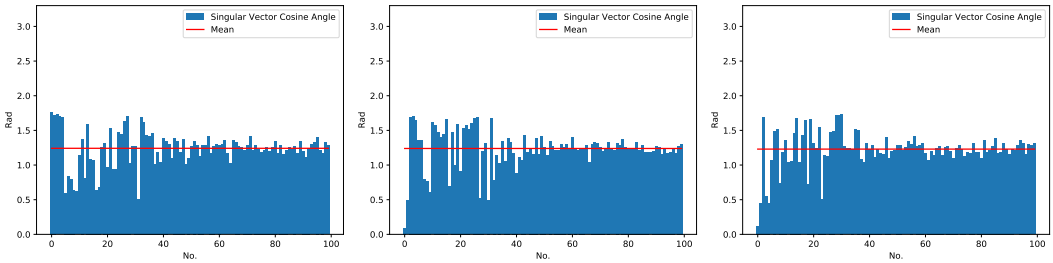
Figure 5.5: TSNE [29] of the bottleneck features of selected networks given **A**→**W** images from Office-31 dataset. <Name>$_d$ and <Name>$_c$ show the outputs with ground truth domain and classification labels, respectively. For the first row, blue shows the source, and red shows the target domain. Best viewed on computer display.

5.4.8 *Subspace Compatibility Analysis*

The transferability of features from source to target domain associated with the singular values σ_i and t_i is measurable by the cosine angle of associated singular vectors $\mathbf{v}_i$ and $\mathbf{r}_i$. While a high angle represents good transferability, this characteristic is associated with the most significant singular values in DANN [72]-type networks. This leads to devastating signals from mid-size singular values containing necessary discriminative information for a classification task [142].

We follow the suggestion of [142] and analyze the cosine angle of the trained DANN [72], CDAN [118], and our proposed ASAN to measure the transferability. The results are presented in Fig. 5.6 and show the cosine angle in radian (RAD) of the singular vectors of the associated singular values between the source and target data in descending order with respect to the magnitude of the singular values. The data is obtained using **A**→**W** images from Office-31 [40] dataset.

The plot indicates that DANN [72], as already stated in [142], has a too high focus on easily transferable features due to the high cosine similarity of the singular vector of the largest singular vectors compared to the rest of the spectrum. This makes DANN [72] vulnerable to neglect discriminative features [142]. The CDAN [118] model is more



(a) RAD of DANN[72] Bottleneck-Feature Subspace *(b)* RAD of CDAN[118] Bottleneck-Feature Subspace *(c)* RAD of ASAN Bottleneck-Feature Subspace

Figure 5.6: Cosine angle in RAD of right singular vectors associated to the first 100 largest singular values between the source and target bottleneck feature output of the trained DANN [72], CDAN [118], and our ASAN. The data is obtained using **A**→**W** images from the Office-31 [40] dataset. The red line shows the mean RAD The plot indicates a better distribution of subspace similarities between the singular vectors of associated large and mid-size singular values of ASAN features. Best viewed on a computer display.

prone to this problem, showing a low angle on the first singular vectors.

However, the ASAN model has a better overall distribution of cosine angles and distributes transferability over the analyzed range of the spectra. The red line shows the mean value, which is very similar in all three architectures, supporting the claim that our ASAN distributes transferability over the spectrum more successfully than compared approaches. Following the interpretation of [142], the ASAN model should, therefore, better express discriminative features invariant to the domains. This finding reflects the performance results in Sec. 5.4.5 by observing a better mean accuracy presented in the results section.

5.4.9 *Ablation Study*

In this section, we present the ablation study results to verify the effectiveness of our approach shown in Tab. 5.5. The building block of our ASAN network is the CDAN base network. The Spectral-Normalization is integrated into the discriminator network, and our proposed Relevance Spectral Loss is learned based on bottleneck features. All parts are evaluated separately on the Office-31 dataset with the same experimental setup as described above.

The results show that the CDAN as base-networks has the worst performance due to no additional adaptation techniques besides the inherent adaptation of CDAN. Integrating SN into CDAN helps stabilize the discriminator gradients, and thereby, CDAN learns a better invariant representation. Learning our RSL with CDAN (CDAN+RSL), we can report that the result is marginally better than CDAN+SN. While not stabilizing

Dataset	**A→W**	**D→W**	**W→D**	**A→D**	**D→A**	**W→A**	Avg.
CDAN	93.1±0.2	98.2±0.2	**100±0**	89.8±0.3	70.1±0.4	68.0±0.4	86.6
CDAN + SN	95.3±0.2	98.9±0.1	**100±0**	**94.7±0.3**	72.6±0.2	71.7±0.2	88.9
CDAN + RSL	95.1±0.0	**99.1±0.0**	**100±0**	93.0±0.0	73.8±0.0	73.2±0.0	89.0
ASAN (CDAN+RSL+SN)	**95.6±0.4**	98.8±0.2	**100±0**	94.4±0.9	**74.7±0.3**	**74.0±0.9**	**90.0**

Table 5.5: Result of the ablation study as mean prediction **accuracy** with standard deviation on the **Office-31** dataset over three random runs.

the gradients, the RSL aligns the spectra of the source and target and, therefore, has increased performance compared to CDAN.

Finally, combining RSL, SN, and CDAN, namely ASAN, leads to the best results. It combines the advantages of the just discussed techniques: the feature extractor receives stable gradients from the discriminator while the spectra are aligned.

5.4.10 *Hyperparameter Tuning and Behavior*

The influence in the prediction performance of the only hyperparameter k of ASAN over four datasets is shown in Fig. 5.7.

As already introduced, k specifies the smallest source singular values group size, which are shrunk during learning. Again, because we inspect the smallest part of the source spectrum, the tuning of this parameter is different from standard hyperparameter tuning: First, we define a range of spectral indices that we define as small, e. g., the smallest 20 singular values. Next, the hyperparameter is optimized via random search [46] in the Domain Adaptation setting [42], and finally, based on the best result, k is chosen accordingly.

We can report that the best value is $k = 11$, and as the plot indicates, minimizing the smallest parts of the spectra almost every time leads to better results. At *Art vs Clipart* the effect is significant, with a performance boost of over 20%. Our results confirm the results of [141] that the smallest part of the source spectrum contains domain-specific information and is at risk of harming the adaptation process.

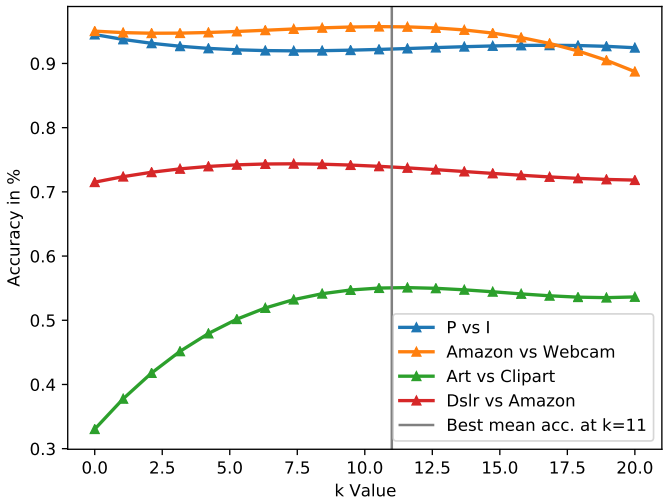


Figure 5.7: Influence of the hyperparameter k on the prediction performance of ASAN in accuracy with the best mean performance at $k = 11$. Best viewed on a computer display.

5.5 DISCUSSION

At the beginning of the chapter, we asked if spectral decomposition can yield a framework for moment matching. The Spectral Loss provides an answer to this question. Due to the connection to the feature variances of eigenvalues, it is made relatively easy. However, the result is very remarkable because the alignment of square rooted eigenvalues, namely singular values, provide by definition a direct option to align these statistics along the principal components of batch data. These principal components occur more naturally and do not rely on user-specified feature dimensions. While the proposed Spectral Loss within the Deep Spectral Network is similar in performance, we showed the ease of use in the experimental part of this chapter.

Moreover, as pointed out in recent studies, careful manipulation of singular values affects the feature characteristics favorable for domain adversarial adaptation. Consequently, those singular values allow a combined enhancement of adversarial and statistical adaptation in a regularization schema called Relevance Spectral Alignment. In combination with the CDAN and the Spectral Normalization, the Adversarial Spectral Adaptation Network (ASAN) is assembled. The extensive analysis explains that ASAN should outperform current state-of-the-art approaches. The experimental part of the chapter ultimately provides empirical evidence for the outstanding performance, robustness, and ease of application.

Limitations. The Domain Adaptation theory provides a framework for performance guarantees and, more importantly, describes the limitation of ASAN. ASAN does not tackle the expected divergence between labeling functions of source and target domain.

Further, ASAN makes neither an assumption about local data structures nor an attempt to approximate these structures, which is indeed an intuition behind the just mentioned label divergences.

Further Directions. The introduced limitations apply to many Domain Adaptation approaches. We are already seeing, and in the coming years, we will see developments that address these limitations. In mid 2021, there are already the first approaches using simple metric learning to directly manipulate local structures [164], [173], [186], [191]. Further, local structures will be captured via (self-supervised) prototypical representations or point clouds [176], [196]. To sum up, the Domain Adaptation theory will be developed further, which already started [189] and based on these assumptions, rethinking of Entropy in terms of Domain Adaptation will take place [194].

6 NON-STATIONARY ONLINE PROTOTYPE LEARNING

Summary: The Robust Soft Learning Vector Quantization (RSLVQ) is a prototype-based classification model assuming a generative Gaussian Mixture Model for underlying data. The model is by definition capable of online learning, making it theoretically suitable for stream classification. However, streams are affected by Concept Drift, changing data distributions, and leading to poor prediction performance of the plain model. In this chapter, the RSLVQ is extended by a Concept Drift detector and an adaptation mechanism. The concept adaptation is implemented by dividing the problem into active and passive depending on the drift type. The contributions lead to a significant performance boost in prediction performance and provide a stable model during Concept Drift.

Publications: This chapter is based on the following publications.

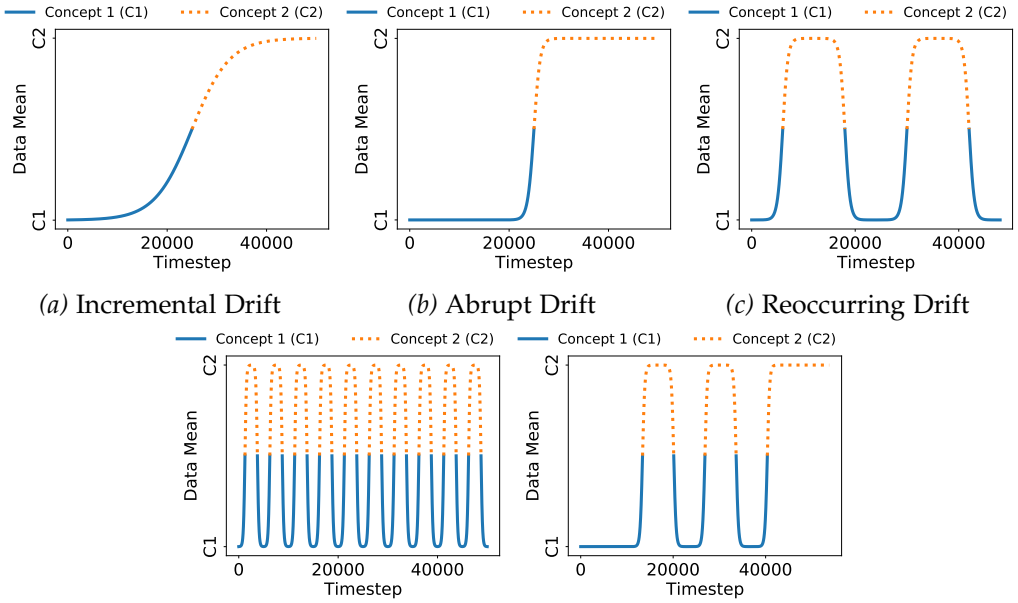
- C. Raab, M. Heusinger, and F. M. Schleif, "Reactive Soft Prototype Computing for frequent reoccurring Concept Drift," in 27th European Symposium on Artificial Neural Networks, ESANN 2019, Bruges, Belgium, April 24-26, 2019, 2019, pp. 437–442, url. esann-2019-33.pdf.
- C. Raab, M. Heusinger, and F.-M. Schleif, "Reactive Soft Prototype Computing for Concept Drift Streams," Neurocomputing, vol. 416, pp. 340–351, Nov. 2020, doi. 10.1016/j.neucom.2019.11.111.
- M. Heusinger, C. Raab, and F.-M. Schleif, "Passive Concept Drift handling via variations of learning vector quantization," Neural Comput. Appl., vol. 6, Aug. 2020, doi. 10.1007/s00521-020-05242-6.

Source Code: https://github.com/ChristophRaab/thesis/tree/main/rrslvq/.

The primary assumption in the previous chapters is to have data from two domains, namely source and target data, during learning. Moreover, Domain Adaptation and Transfer Learning, in general, do not assume limited data accessibility. For example, batches of data arrive do not arrive at different time steps and cannot be processed simultaneously. It is possible to inspect every single data point multiple times regardless of domain and time restrictions. However, recent years demonstrated a rapidly increasing amount of data generated by technologies like social media or sensor data [139]. In these cases, data is *streamed*, meaning data is generated within a time sequence and published without an explicit intention to organize or store [108] it. The procedure exceeds the memory and processing capabilities of analyzing systems by far, making it impossible to collect and store all needed data and afterward processing it multiple

times. Hence, with the methods introduced in chapters 3, 4, and 5, processing these streams nor adapting to potential distribution differences is impossible because prior data is no longer available and running times are too long.

Based on the above, we have to introduce certain assumptions on the learning model to execute common tasks like regression or classification on data streams: The whole dataset is not storable. A significant amount is neither known nor accessible at learning time. The time complexity per learning step must be approximately linear to be able to follow the stream. An appropriate data storage system is optional but without the intent to store a complete dataset. Finally, streams are often affected by a change of underlying class distributions known as Concept Drift. This results in drops in the prediction performance of prior models, making them unusable. The detection and handling of these events is one key area in the field of streaming research. Drifts appear differently through speed, intensity, and frequency, i. e., incremental, abrupt, gradual, or reoccurring shown in Fig. 6.1 [61]. Every sub-figure shows the concept of a particular drift given as data mean. The shapes mark the dominating concept at a given time step. The vertical axis shows the overall data mean and the transition from one concept to another.



(a) Incremental Drift *(b)* Abrupt Drift *(c)* Reoccurring Drift

(d) Frequent Reoccurring *(e)* Gradual Drift

Figure 6.1: Different types of drifts, one per sub-figure and conceptually illustrated as data mean. The colors mark the dominant concept at a given time step. The vertical axis shows the data mean and the transition from one concept to another. Given the time axis, the transition speed is given, which reaches from very slow at incremental to very fast at frequent reoccurring drift. The figures are inspired by [61].

Streaming algorithms are fulfilling the assumed restrictions from above. Hence, they are designed to process data as fast as they arrive and online. In recent years many models such as trees [9], k-nearest neighbors [105], or ensemble models [23] are successfully adapted to stream classification. However, the task of Concept Drift handling remains still open.

The first major problem is the high number of different Concept Drift types, making it challenging to design an algorithm able to handle all types of drift. The second problem is the stability-plasticity dilemma [61]. It appears by handling an occurring Concept Drift, expressing the trade-off between incorporating new knowledge into models (plasticity) and preserve prior knowledge (stability). The dilemma requires a learned model to adapt to a new concept within a time period or decide which prior learned concept is required. The model performance can worsen within this transition time because the model is still adapting to the new concept.

Prototype-based learning is considered more robust and better interpretable as concepts such as SVMs or neural networks used in the chapters before [110]. Moreover, they are favorable for streams because of the online learning ability available by definition of online stochastic gradient descent (SGD) [129]. The Robust Soft Learning Vector Quantization (RSLVQ) [12] is a probabilistic prototype classification model and expresses predictions of a data stream sample as ensemble predictions by the prototypes. In RSLVQ, a set of prototypes approximates the assumed generative Gaussian mixture model, while every prototype represents an associated Gaussian distribution. The model is learned via online stochastic gradient descent. However, the original RSLVQ can neither detect nor adapt its current model to any new concept in a decent amount of time. Because first, there is a lack of mechanics for detecting, and second, adaptation times via SGD can be too long depending on the drift type and intensity. For example, they are illustrated by the Concept Drift in Fig. 6.1b.

Therefore, in the scope of this chapter, we deal with the question naturally arising from the previous discussion: **RQ4: Can we extend probabilistic prototype models to handle Concept Drift?** In the context of this thesis, handling refers to detection *and* adaptation. By providing an answer to this question, we advance the field of prototype-based stream classification by the **following contributions:**

- We modify the Kolmogorov-Smirnov [43] (KS) test to be appropriate for Concept Drift detection in data streams.
- A prototype insertion and re-learning procedure for active drift adaptation.
- Online momentum-based gradient descent for passive adaptation.
- Introduction of frequent reoccurring Concept Drift as new drift scenario.
- Analysis of the observed stability during drift provided by the ensemble of detection and adaptation policies.

An extensive empirical performance study including prediction performance, time, and memory requirements during Concept Drift handling is provided in Sec. 6.3. A discussion and outlook are provided at the end in Sec. 6.4. We start the chapter by introducing the background of stream classification and Concept Drift, followed by related stream classification models, Concept Drift detectors, and adaptation approaches.

6.1 BACKGROUND AND RELATED WORK

Stream Classification. In supervised classification a stream with potentially infinite length is a sequence $S = \{s_1, \dots, s_t, \dots\}$ of tuples $s_i = \{\mathbf{x_i}, y_i\}$, arriving one s_t at time t. A stream classifier predicts labels $y_t \in \mathcal{Y} = \{1, \dots, C\}$ of unseen data $\mathbf{x}_t \in \mathbb{R}^d$ by prior model $\hat{y} = h_{t-1}(\mathbf{x}_t)$, with $y_i \in \mathcal{Y}$, and includes this tuple into the model afterwards $h_t = learn(h_{t-1}, s_t)$. This requires algorithms to predict and learn at any time.

Most stream classifiers extend basic online variants of Naive Bayes, k-Nearest Neighbors, or classification trees - or use them as baselines. The OzaBagging [23] is an ensemble schema for effectively combining multiple baseline models. The Hoeffding Tree is a tree [9] with online learning using the Hoeffding bound [1] to determine the necessity to split a tree into new leaves. The bound makes claims about the needed sample sizes for an appropriate approximation of the ideal tree. These attributes make the learning of the tree very fast. However, OzaBagging and Hoeffding Tree assume a stationary distribution, which is violated, as we will see in the following.

Concept Drift. A stream classifier must cope with non-stationary environments, which change through the occurrence of a drift in the underlying concept. The so-called Concept Drift is the change of joint distributions of a set of samples and corresponding labels between two points in time by

$$\exists \mathbf{x} : p(\mathbf{x}_{t-1}, y_{t-1}) \neq p(\mathbf{x}_t, y_t). \tag{6.1}$$

Various drift types occur in streaming data and cause the concept to change from one time step to another. Again, Fig. 6.1 gives a short overview of drift types [61]. Note that we can rewrite Eq. (6.1) to

$$\exists \mathbf{x} : p(y_{t-1}|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}) \neq p(y_t|\mathbf{x}_t)p_t(\mathbf{x}_t). \tag{6.2}$$

If only the prior distribution $p(\mathbf{x})$ changes for two points in time, it is called virtual drift. We assume that a change in this prior distribution is always present at real Concept Drift [61]. Therefore, we assume with the proposed detector in Sec. 6.2 to identify every Concept Drift through monitoring $p(\mathbf{x})$. This is also an essential assumption for Concept Drift detectors based on statistical tests since they do not observe the conditional class probability but the prior distribution.

The frequent reoccurring drift is shown in Fig. 6.1d and has not yet been considered in the literature but is particularly impressive. Stream classification algorithms are

usually tested on streams with low drift occurrences per setting [101]. However, in real-world settings like robotics or autonomous driving, the concept can change very often due to changing external conditions like lighting or weather [13], [60]. Therefore, we also use stream generators with frequent reoccurring drifts.

Concept Drift Detectors. Concept Drift detectors are trying to detect a change in streams either by monitoring the distribution of the streams or the performance of a classifier with respect to some benchmark, e. g., accuracy. A popular approach for monitoring the prediction accuracy of a classifier is the Adaptive Sliding Window (ADWIN) [20], and it assumes that if a change in performance is detected, the concept has changed. ADWIN identifies changes in distributions using a window W and splits W into two adaptive subwindows, and compares underlying statistics. The main window grows as no change is detected and shrinks if a change between subwindow statistics is detected. The change is recognized via the Hoeffding Bound [20].

Another Concept Drift detector that monitors performance values is the Drift Detection Method (DDM) [13]. It assumes a Binomial distribution of performance values assuming the context is stable. Thereby, DDM monitors current performance plus standard deviation and reports a warning or a detection level based on its confidence. The Early Drift Detection Method (EDDM) is statistically similar to DDM and compares current performance with the maximum achieved performance to obtain faster warning and detection levels [19]. Other variants are built upon DDM to improve single aspects of it [21], [71]. The main problem with the approaches above is the monitoring of the performance of a classifier. A goal of a detector must be to identify any change as soon as possible. Therefore, a detector should monitor the data distribution rather than performance values, which we will implement by a feature-wise Kolmogorov-Smirnoff [43] test in Sec. 6.2.

Related methods similar to our approach identify drift not only but mainly via statistical tests on distributions of streams. In [90], two windows are maintained by a randomized search tree, keeping recent data and the last concept. The KS-Test detects Concept Drift between windows. The approach simplifies the KS-Test, similar to our approach. However, it requires a table of critical values for the application. Further research is done in [81], where one window stores a snapshot of the concept since last drift and another one stores a recently surveyed concept. However, the detector is supervised and is not able to detect Concept Drift independent of conditional class probabilities.

Passive Concept Adaptation. Concept Drift handling techniques can be roughly divided into active and passive [104]. The passive ones have no specific detection strategy, hence continually updating the model without awareness of Concept Drift. The SAMKNN approach [105] uses online KNN as baseline algorithms and maintains a short term memory (STM) and a long term memory (LTM). The approach minimizes the error of KNN on STM and compresses error-increasing data into the LTM. The STM is thereby the assumed current concept, and the LTM is a representation of former concepts. An

ensemble schema makes a final decision.

In this chapter, we use momentum-based gradient descent for passively adapting to slow drifts, e. g., incremental drifts. While the Hoeffding Tree initially has no formulation for Concept Drift, the algorithm passively adopts the new concept very fast because of the efficiency of building new trees with the Hoeffding bound. Therefore it should also be mentioned as a passive adaptation algorithm.

Active Concept Adaptation. Active adaptation changes a model noticeable. The Adaptive Random Forest (ARF) [101] classifier maintains an ensemble of online random forest variants. A Concept Drift detector monitors every tree performance, and drift is detected if a drop in performance is noticed. The adaptation strategy replaces outdated trees with new ones, which are constantly trained in the background with assumed current concepts. Similar to ARF [101], the ADWIN algorithm from above is easily combined with stream baselines or baseline ensembles, e. g., OzaBaggingADWIN [32] or HoeffdingAdoptiveTree [31]. Both use ADWIN to actively update the content of the sliding window and the classification model.

A common way to adapting a Learning Vector Quantization (LVQ) model actively is prototype adaptation or insertion [82]. In [200], a new prototype is inserted as the mean of a set of misclassified examples. In [80] one sample is chosen from a selected set of samples yielding a new prototype if the classification error is minimized. A near-mean technique is proposed by [82], using a specific sample as a new prototype, which has the smallest Euclidean distance to the mean of a set of misclassified examples.

Besides great results in the respective domains, they share the problem of assuming the existence of all classes in a given batch of samples or apply asymmetric prototype insertion. Due to the composition of streams, with potentially unbalanced classes [70], this assumption is not guaranteed. In this work, we propose a prototype adaptation strategy that can cope with missing classes in a batch of stream data. In the last years, we already see Concept Drift classifiers implementing both active and passive adaptation [149]. In the following chapter, we will implement active and passive adaptation for the RSLVQ.

6.2 MODEL

Overview. The section details the proposed extensions to make the Robust Soft Learning Vector Quantization (RSLVQ) [12] able to handle Concept Drift. The integrated Concept Drift detector is called Kolmogorov-Smirnoff-Windowing (KSWIN). The resulting model is called Reactive RSLVQ (RRSLVQ). First, we introduce the memory management and the Concept Drift detector. Followed by the active and passive adaptation strategy. The section finishes with an analysis of the properties of RRSLVQ and the behavior during a drift.

Contribution. The RSLVQ [12] is a probabilistic prototype-based learning model, applying the concept of Gaussian Mixture models to prototype-based classification [5]. Using the RSLVQ is a straightforward way for probabilistic stream classification. As already introduced, however, the RSLVQ is not per se able to detect and adapt to Concept Drift. The RSLVQ is extended by a Concept Drift detector based on the Kolmogorov-Smirnoff-test. If a drift is detected, an active prototype insertion is initiated. Based on momentum-based gradient descent, the passive adaptation strategy is integrated into the model, constantly adapting to slow Concept Drifts missed by the detector.

Concept Drift Detection

Before applying a drift detector, a memory strategy must be defined. The sliding window Ψ (Fig. 6.2) keeps n recent points from the stream. It pushes incoming data to the top and removing the oldest one from the bottom. We use the Kolmogorov-Smirnov test as a Concept Drift detector, which expects two data windows. Therefore, we create two windows out of the sliding window Ψ. The first windows $R = \{\mathbf{x}_i \in \Psi\}_{i=n-r+1}^{n}$ has the most recent data points from Ψ. We define most recent as the r newest arrived data points. The second window W is created by sampling uniformly from the non-recent part of Ψ by

$$W = \{\mathbf{x}_i \in \Psi | i \leq n - r \wedge p(\mathbf{x}) = \mathcal{U}(\mathbf{x}_i | 1, n - r)\} \tag{6.3}$$

with $|W| = |R| = r$ and $\mathcal{U}(\mathbf{x}_i|1, n-r) = \frac{1}{n-r}$ is the Uniform distribution. By this, we do not make any assumptions of distribution but assume to represent the concept of the non-recent data. The memory strategy with the sliding window is summarized in Fig. 6.2. For subsequent adaptation steps, we also store the label y_i to a given sample $\mathbf{x}_i$.

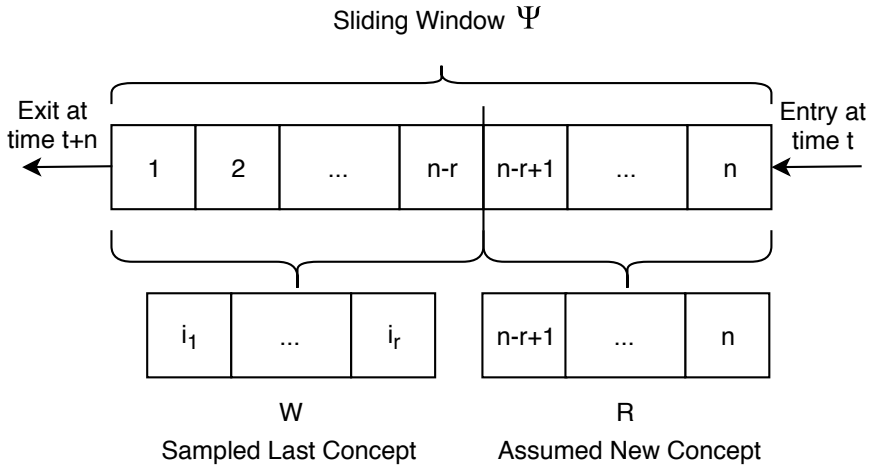


Figure 6.2: The memory strategy within the Concept Drift detector. It stores n samples in a sliding window. At every time step t, r samples are picked Uniform and are compared against the newest r samples of the window.

By using the above sampling scheme for W, there is a limitation in Concept Drift detection regarding some streams. The worst case is to take a sample from a recurring stream at certain intervals and always get the same concept of the changing stream. Thus the distribution seems to be the same from window R and W because the detector always picks at the same rate of concept change. However, there is a Concept Drift ongoing. This is due to the large window and the Uniform picking. Other probability distributions for the sampling scheme should lead to a different result, but with familiar problems and some Concept Drifts will be missed because the sampling procedure approximates the window Ψ. We choose the Uniform distribution because it has no parameters and is fair in sampling from unknown distributions.

The Kolmogorov-Smirnov [43] test is a non-parametric test, accepting one dimensional data without any assumptions of underlying distributions. It compares the absolute distance $dist_{w,r}$ between two empirical cumulative distributions F_W and F_R by

$$dist_{w,r} = \sup_x |F_W(x) - F_R(x)|, \tag{6.4}$$

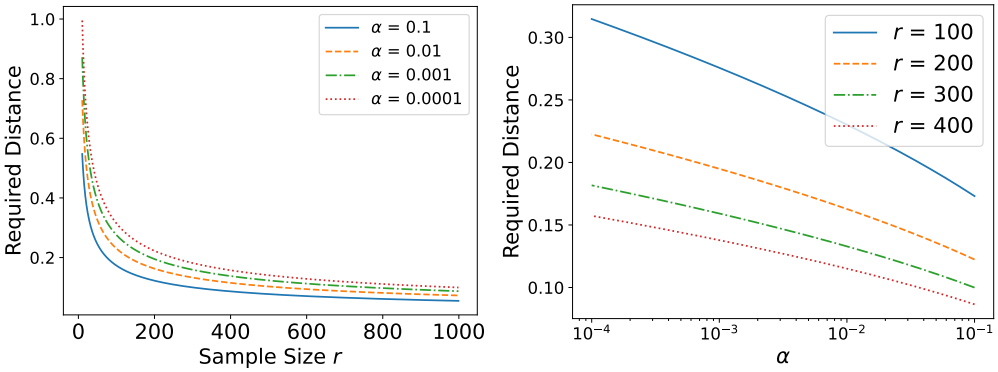
where $F_{(\cdot)}(x) = \frac{1}{n}\sum_{i=1}^{n} I_{[-\infty,x]}(x_i)$ and $I_{[-\infty,x]}(x_i)$ is an indicator function which is one if $x_i \leq x$ and zero otherwise. Note that $sup(x)$ is the smallest required x for a condition to be valid. If this lower bound of maximum distance, i. e., $dist_{w,r}$, is greater than the test statistic, then the null hypothesis is rejected, with significance level α. For two subwindows W, R with the same size, the test is reduced to

$$dist_{w,r} > c(\alpha)\sqrt{\frac{n+r}{nr}} = \sqrt{-\frac{1}{2}\ln\left(\frac{\alpha}{2}\right)}\sqrt{\frac{n+r}{nr}} \overset{\text{(n=r)}}{=} \sqrt{-\frac{\ln\left(\frac{\alpha}{2}\right)}{r}}, \tag{6.5}$$

where α is the confidence level, r is the size of the window R and n is the size of the window w. $c(\alpha)$ is the critical value of the test with respect to the confidence level α, which can either be looked up for standard test sizes or approximated by $\sqrt{-\frac{1}{2}\ln\left(\frac{\alpha}{2}\right)}$ [56]. Due to the restriction to one-dimensional distributions, the test in Eq. (6.5) is applied to all dimensions, therefore at any point t, d tests must be done due to $\mathbb{R}^d$. With this extension and based on Eq. (6.5), the following can be stated:

Lemma 6.2.1. *Given* $\mathbf{X}_t$ *and* $\mathbf{X}_{t-1}$ *with* $\mathbf{X}_l = \{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^d$, $p(\mathbf{X}) = \prod_{k=1}^{d} p(\mathbf{x}^{(k)})$, $\forall \mathbf{x}^{(k)} \sim$ *i.i.d and their cumulative distributions* F_t,F_{t-1}, *KS-Test detects a change between* $p(\mathbf{X}_t)$ *and* $p(\mathbf{X}_{t-1})$, *i. e.,* $\exists \mathbf{x}^{(k)} : p(\mathbf{x}_t^{(k)}) \neq p(\mathbf{x}_{t-1}^{(k)})$, *with probability* $1-\alpha$, *if the difference in empirical distribution is at least* $\sqrt{\frac{-ln\left(\frac{\alpha}{2}\right)}{r}}$.

To implement Lemma 6.2.1, we set $\mathbf{X}_t = R$ and $\mathbf{X}_{t-1} = W$. For a reasonable choice of window size r, the influence of changing the window size and the confidence level of the KS-Test is demonstrated in Fig. 6.3. The left figure shows the influence of the window size, and the right figure, the confidence level on the required distance. By



(a) Window Size (Log scale) *(b)* Confidence Level (Log Scale)

Figure 6.3: Plot of the parameter sensitivity to the required distance (Eq. (6.5)) of the KS-Test in log scale on the x-axes. The left figure shows the sample size and the right figure, shows the confidence level against the required distance needed for a Concept Drift detection. Both figures show sensitivity towards the window size and especially with a tiny window the test becomes more insensitive to small differences in distribution.

decreasing the confidence level α, *dist* increases, and with increasing window size r, *dist* decreases. Hence, they behave competitively w.r.t. *dist* value. In a streaming context with potentially unlimited data, the KS-Test will be too sensitive, given a large window size, with many false positives. This motivates the choice of a relatively small window size $r = 30$ while being statistically valid at the same time.

When it comes to many statistical tests and multiple hypothesis testing, false positives occur due to random chance. By applying the Bonferroni-Dunn correction [27], we reduce this effect by obtaining a new alpha with $\alpha^* = \lfloor \frac{\alpha}{r} \rfloor = \frac{0.05}{60} = 0.0001$. Because α^* affects Eq. (6.5), a small alpha increases the required distance.

Both a small window size and the corrected alpha should avoid false positives. However, both actions do not entirely avoid false signals but making the KS-Test more insensitive. Combined with the proposed prototype adaptation strategy, we will see that those false positives are not critical. In the following, the KS-Test detector is called KSWIN.

Active Adaptation

For the active adaptation, we assume a set of m prototypes $\Theta = \{(\boldsymbol{\theta}_j, y_j) \in \mathbb{R}^d \times \{1, \ldots, C\}\}_{j=1}^m$, where each prototype represents a multi-variate Gaussian model, i. e., $\mathcal{N}(\boldsymbol{\theta}_j, \Sigma)$ with $\Sigma = \mathbf{I}\sigma^2$ and $\mathbf{I}$ as identity matrix of class y_i and variance σ^2. For an extended description of RSLVQ, we refer the interested reader to Sec. 2.4.

The adaptation strategy adapts actively to abrupt or gradual Concept Drift. We allow missing classes in R since streams have unbalanced classes, and class-wise adaptation

can be infeasible [70]. If a drift is detected as in Sec. 6.2, window R represents a new context and Θ represents the last context. According to KSWIN, the current set of prototypes Θ has approximated an outdated distribution.

Hence, we propose the following two-step adaptation strategy. The first step is to replace the complete set Θ of prototypes with m new prototypes. Note that the number of classes remains the same. A good starting point for replacement by means of approximating an unknown mixture model by Gaussian mixture is the mean of points [12] regardless of class affiliation. Therefore, the new prototypes become

$$\boldsymbol{\theta}_j^{(new)} = \frac{1}{r} \sum_{\mathbf{x} \in R} \mathbf{x}, \quad \forall j = 1, \ldots, m. \tag{6.6}$$

The next step is to train all prototypes on all $\{\mathbf{x}_i, y_i\} \in R$. See Eq. (6.3) and Fig. 6.2 for details about R. This modifies prototypes and draws them to the same class points, and pushes different class prototypes away. If there are missing classes in R, the algorithm pushes the prototypes away from points with a different class.

The adaptation procedure is visualized in Fig. 6.4. It shows the adaptation of two prototypes with different classes as mean in 6.4a and the optimization afterward in Fig. 6.4b in a two-dimensional feature space. Data is generated with MIXED generator stream from Scikit-Multiflow [120].

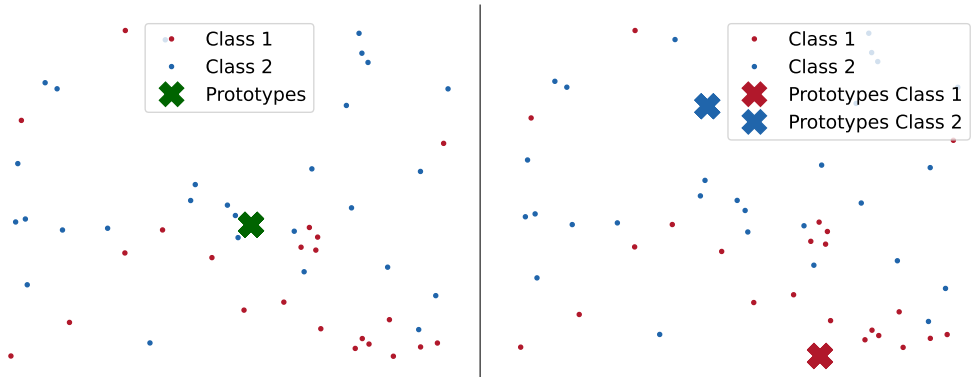
Note that at the end of this section, we will see that adaptation on R leads to a lower error if the distributions are different enough instead of taking no action. Therefore, false positives occurring by the limitation of the Concept Drift detector are not critical because every replacement always leads to lower error and does no harm. Hence, we now assume that we can adapt to fast or abrupt Concept Drift actively.

Passive Adaptation

We assume a set of prototypes as in the activate adaptation above. For an extended description of RSLVQ, we refer the interested reader to Sec. 2.4.

As pointed out recently [163], momentum-based gradient descent is a reliable strategy to passively adapt Concept Drift given LVQ classifiers for certain Concept Drifts. Hence, for the passive adaptation, we introduce momentum-based gradient descent and, in particular, show how to implement the Adadelta procedure for the RSLVQ. Adadelta, among others, is favorable for streaming because the approach has no learning rate, while the introduced hyperparameter is easy to set.

Momentum SGD. The vanilla SGD has already been modified and applied to various fields, e. g., deep learning [112], to reduce common problems inherit of the respective field. These problems include among others, sensitivity to steep imbalanced valleys, i. e., areas where the cost surface curves are much more steeply in one dimension than in another [4], common around local optima. Momentum [6] damps the oscillations of



(a) Insertion Step *(b)* Optimization Step

Figure 6.4: Sketch of the active prototype adaptation strategy on given window R with $|R| = 50$ for more details. First, insertion of m prototypes as mean over all samples in the left figure, marked as a red diamond. Second after optimization, where green/blue crosses/dots are representing class prototypes. Best viewed on computer displays.

gradients in these valleys and, therefore, helps to accelerate SGD optimizer to descent into the relevant direction. It does this by adding a fraction γ to the previous update vector $\mathbf{v}_{t-1}$ of the previous time step $t-1$ to the current update vector $\mathbf{v}_t$ by

$$\mathbf{v}_t = \gamma \mathbf{v}_{t-1} + \eta \nabla_\theta J(\boldsymbol{\theta}), \tag{6.7}$$

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \mathbf{v}_t. \tag{6.8}$$

where $\nabla_\theta J(\boldsymbol{\theta})$ is the gradient of the objective function J with respect to the parameter θ at learning step t and η is the learning rate. The momentum term γ is usually set to 0.9 [64], [91], but can be seen as a hyperparameter, which is optimized, e. g., via grid search in a range of $[0.9, 0.99]$ [52], [91].

These momentum-based algorithms diverge to a local optimum much faster and sometimes reach better optima than SGD [91]. Since momentum-based algorithms make larger steps per iteration, they should adapt faster to new concepts. Thus, we implemented this idea into the RSLVQ and replaced the gradient update rule with Adadelta.

Adadelta. While other momentum-based SGDs accumulate all previously squared gradients for controlling the parameter update via denominator, the Adadelta [52] approach restricts the window of accumulated past gradients to some fixed size w. The idea is to restrict the denominator of Adadelta to be a local estimate rather than converging to infinity. This makes continual momentum-based learning possible even for vast streams.

Instead of storing all w squared gradients inefficiently, the sum of gradients is recursively defined as a decaying average of all past squared gradients instead. The

running average $E[\mathbf{g}^2]_t$ at time step t of the squared gradient $\mathbf{g}^2$ then depends (as a fraction γ similarly to the momentum term) only on the previous average and the current gradient

$$E[\mathbf{g}^2]_t = \gamma E[\mathbf{g}^2]_{t-1} + (1-\gamma)\mathbf{g}_t^2. \tag{6.9}$$

For further computations, we also define for the parameter updates $\Delta\boldsymbol{\theta}$, the exponentially decaying average over the squared parameter updates as

$$E[\Delta\boldsymbol{\theta}^2]_t = \gamma E[\Delta\boldsymbol{\theta}^2]_{t-1} + (1-\gamma)\Delta\boldsymbol{\theta}_t^2. \tag{6.10}$$

Next, let the root mean squared (RMS) of parameter update be

$$RMS[\Delta\boldsymbol{\theta}]_t = \sqrt{E[\Delta\boldsymbol{\theta}^2]_t + \epsilon}, \tag{6.11}$$

where ε is a constant to better condition the denominator [52]. The update rule of Adadelta is defined by

$$\Delta\boldsymbol{\theta}_t = -\frac{\eta}{RMS[\mathbf{g}]_t}\mathbf{g}_t. \tag{6.12}$$

However, if every quantity has a theoretical unit measure, the update rule does not relate to the parameters but the gradient unit. For providing the same units for the update rule as Hessian methods do, the learning rate η is replaced by $RMS[\Delta\boldsymbol{\theta}]_{t-1}$ and as a result the final Adadelta parameter equation is

$$\Delta\boldsymbol{\theta}_t = -\frac{RMS[\Delta\boldsymbol{\theta}]_{t-1}}{RMS[\mathbf{g}]_t}\mathbf{g}_t \tag{6.13}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \Delta\boldsymbol{\theta}_t. \tag{6.14}$$

While this fraction comes in handy to correct units, it approximates the diagonal of the hessian matrix, assuming a diagonal curvature [52], which is also positive - necessary for not switching the gradient.

Implementation. To exchange the SGD learning of the RSLVQ with the AdaDelta algorithm, the learning rule of the RSLVQ is replaced by the update rule provided in Eq. (6.13). The posterior probabilities are calculated the same way as in the RSLVQ_{SGD} and are computed over the objective function of the RSLVQ by

$$g_t = \begin{cases} -p(j|\mathbf{x}_t)ls_t(\mathbf{x}_t - \boldsymbol{\theta}_t), & if\ c_j = y_t, \\ p(j|\mathbf{x}_t)(1-ls_t)(\mathbf{x}_t - \boldsymbol{\theta}_t), & if\ c_j \neq y_t, \end{cases} \tag{6.15}$$

where $p(j|\mathbf{x})$ is the posterior probability of $\mathbf{x}$ is generated by the j_{th} prototype and ls_t is the probability that the x was generated by any component with different class.

In the next step, the running average of past squared gradients $E[\mathbf{g}^2]$ is updated by Eq. (6.9). Now the gradient update at time step t can be calculated by

$$\Delta\boldsymbol{\theta}_t = -\frac{\sqrt{E[\Delta\boldsymbol{\theta}^2]_{t-1} + \epsilon}}{\sqrt{E[\mathbf{g}^2]_t + \epsilon}}\mathbf{g}_t. \tag{6.16}$$

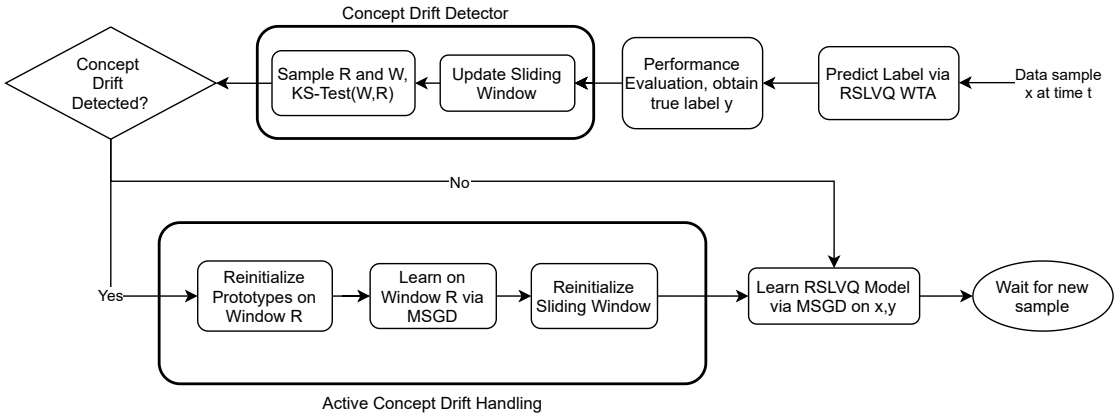


Figure 6.5: Overview of all components of the RRSLVQ model. Note MSGD is short for momentum-based SGD. WTA is the winner takes all rule. R is the recent window introduced within the description of the Concept Drift detector. KS-Test is the Kolmogorov-Smirnoff test.

Afterward, the prototype update $\Delta\boldsymbol{\theta}_t$ is applied via gradient descent as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \Delta\boldsymbol{\theta}_t. \tag{6.17}$$

The squared parameter updates $E[\Delta\boldsymbol{\theta}^2]$ are stored by Eq. (6.10) for further iterations.

With this, we eliminated the learning rate but introduced the γ parameter. We omit the need for tuning the γ by following [52] and set it to $\gamma = 0.9$. We assume in the following that the momentum-based RSLVQ is able to passively adapt to slow Concept Drift while at the same time reducing the complexity of the model in terms of parametrization.

Properties

The RRSLVQ optimized via online momentum gradient descent has the time complexity of $\mathcal{O}(m)$ at given time t for m prototypes without Concept Drift. The Concept Drift handling has the complexity $\mathcal{O}(r \cdot m)$. The KS-Test can be implemented in log-linear time [90].

The demand for memory depends on the number of prototypes m and sliding window size n. Therefore, the model needs at maximum $m \times d$ and $n \times d$ as real float numbers in memory, which accumulates in the complexity of $\mathcal{O}(d \cdot (m+n)) = \mathcal{O}(d \cdot (m+300))$ with $n = 300$. For using RRSLVQ in embedded systems with restrictive memory, we follow the *any memory framework* by [61, p. 6] and we suggest setting the number of prototypes plus the window length to a maximum of $d \cdot (m+300) < k$ with k as maximum float memory storage capacity. The procedure of RRSLVQ is shown in pseudocode 7. A schematic overview of all components is given in Fig. 6.5.

Algorithm 7 Reactive Robust Soft Learning Vector Quantization (RRSLVQ)

procedure INITIALIZATION
Input: m as number of prototypes; σ as width of Gaussian kernel;
γ as decay-factor; r as sampling size; α as confidence-level.
Output: $h_0(\mathbf{x}; \Theta, \gamma, \alpha, r)$ as initialized model
    Create prototypes $\Theta = \{\boldsymbol{\theta}_j\}_{j=1}^{m} \sim \mathcal{N}(\mathbf{0}, \Sigma = \mathbf{I}\sigma^2)$
end procedure

procedure PREDICTION
Input: $h_{t-1}(\mathbf{x})$ as RRSLVQ model; $\mathbf{x}_t \in \mathbb{R}^d$ sample point arrived at time t.
Output: Predicted label $\hat{y}_t$ of sample $\mathbf{x}_t$.
    Compute nearest prototype $\boldsymbol{\theta}_q$ to $\mathbf{x}_t$ (Eq. (2.37))
    Assign predicted label $\hat{y}$ based on the neared prototype $\boldsymbol{\theta}_q$
end procedure

procedure LEARNING
Input: $h_{t-1}(\mathbf{x})$ as RRSLVQ model from previous time step; $\{\mathbf{x}_t, y_t\}$
    as labeled data at time step t.
Output: $h_t(\mathbf{x})$ as updated model.
    Update sliding window (According to Fig. 6.2)
    Create sampling window W and R (Eq. (6.3) and Fig. 6.2)
    Compute d distribution differences $dist_{w,r}$ between $R, W \in \mathbb{R}^d$ (Eq. (6.4))
    if $\exists d_k > \sqrt{-\frac{\ln\left(\frac{\alpha}{2}\right)}{r}}$ (CD-Detection Eq. (6.5)) **then**
        $\boldsymbol{\theta}_j^{(new)} = \frac{1}{r}\sum_{\mathbf{x}\in R}\mathbf{x}, \quad \forall j = 1, \ldots, m$
        for $\forall \mathbf{x}_i, y_i \in R$ **do**
            Step 28 to 33
        end for
        $\Psi = R$ (Replace Sliding Window Ψ with last concept R)
    end if
    Use $\{\mathbf{x}_t, y_t\}$ for training:
    for $\boldsymbol{\theta}_j : j = 1 \rightarrow m$ **do**
        Compute posterior probabilities (Eq. (2.35))
        Compute gradient g_t for $\boldsymbol{\theta}_j$ (Eq. (6.15))
        Compute past gradient and parameter updates (Eq. (6.9) and (6.10))
        Update prototype $\boldsymbol{\theta}_j = \boldsymbol{\theta}_j - \Delta\boldsymbol{\theta}_t$ (Eq. (6.17))
    end for
end procedure

Stability during Concept Drift

The stability during drift will be shown in the following, and we start with the introduction of notation and background. Let $h : \mathcal{X} \to \mathcal{Y}$ on a given sample $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$, where $\mathcal{Y}$ is the label space, be a hypothesis. The loss of a hypothesis $l(h(x)) \in \mathbb{R}_0^+$ is measuring the performance of a given set of labeled data sampled from the concept $C = p(x, y) = p(y|x)p(x)$. The expected risk $\varepsilon(h(x))$ and the empirical risk $\hat{\varepsilon}(h(x))$ of the classifier $h(x)$ are defined [37] as

$$\begin{aligned} \varepsilon(h(x)) &= \mathbb{E}[l(h(x), y)] \\ &= \int l(h(x), y)p(y|x)p(x)dxdy \leq \hat{\varepsilon}(x) = \frac{1}{n}\sum_{i=1}^{n} l(h(x), y), \end{aligned} \tag{6.18}$$

by computing the empirical risk with n samples.

Theorem 6.2.2. *Given two hypotheses $h_1(x)$ and $h_2(x)$ with loss $l(h_{(\cdot)}(x))$, trained on two different concepts C_1 and C_2 with c classes each and corresponding risks $\varepsilon_{(\cdot)}(h_{(\cdot)})$. The empirical risk $\varepsilon_2(h_1(x))$ on C_2 can be bounded by*

$$\hat{\varepsilon}_2(h_2(x)) \leq 1 - \frac{1}{c} \leq \delta\varepsilon_2(h_1(x)) \leq \hat{\varepsilon}_2(h_1(x)), \tag{6.19}$$

under the assumption that there is a linear relationship $\delta = \frac{p_2(x)}{p_1(x)}$ with $\delta \in \mathbb{R}^+$ between the prior of the concepts.

Proof. First, consider the risk of the first classifier on concept one by

$$\varepsilon_1(h_1(x)) = \mathbb{E}[l(h_1(x), y)] = \int l(h_1(x), y)p_1(y|x)p_1(x)dxdy, \tag{6.20}$$

where ε_1 is the risk corresponding to concept C_1. Applying $h_1(x)$ to concept two leads to the expected risk

$$\varepsilon_2(h_1(x)) = \int l(h_1(x), y)p_2(y|x)p_2(x)dxdy. \tag{6.21}$$

Similar as in [37], we assume the relationship between the two concepts by $p_2(x) = \delta p_1(x)$, e. g., two variants of Gaussian distributions, then we can rewrite the risk to

$$\begin{aligned} \varepsilon_2(h_1(x)) &= \int l(h_1(x), y)p_2(y|x)\delta p_1(x)dxdy \\ &= \delta \int l(h_1(x), y)p_2(y|x)p_1(x)dxdy. \end{aligned} \tag{6.22}$$

The expected risk $\varepsilon_2(h_2(x))$ is analog to Eq. (6.21). If the difference in prior distribution between the two concepts is large enough, then δ becomes large. Hence, for large δ, the expected risk of $\varepsilon_2(h_1(x))$ becomes larger than random guessing, which is $1 - \frac{1}{c}$ for

c classes. Because $h_2(x)$ is trained on concept two, we expect that the empirical risk $\hat{\varepsilon}_2(h_2(x))$ in the worst-case is random and therefore

$$\hat{\varepsilon}_2(h_2(x)) \leq 1 - \frac{1}{c} \leq \delta\varepsilon_2(h_1(x)) \leq \hat{\varepsilon}_2(h_1(x)). \tag{6.23}$$

□

This can be implemented as $h_1(x)$ being the RSLVQ model at time $t-1$ and $h_2(x) = h(R)$ at time t after the adaptation procedure.

This has the following consequences: (i) the performance of the adapted model yields a smaller loss compared to the model from $t-1$ if large enough differences between windows are presents, which the KSWIN detects described in Lemma 6.2.1 given a small r and α. (ii) if a Concept Drift is falsely detected (false-positive), then all prototypes are reinitialized and trained on the most recent window R, which is precisely the last concept. Therefore, the false positives of KSWIN do not harm. (iii) the classification result is not negatively affected even if KSWIN detects a change in prior distribution within a concept, which is not real Concept Drift, for the same reason as in (ii). The effect is demonstrated in Fig. 6.7, showing that δ is always sufficiently large so that $h(R)$ achieves better performance on the MIXED Concept Drift stream.

6.3 EXPERIMENTS

Overview. The experimental section of the chapter consists of three parts. **(i)** analysis of the KSWIN detection performance as an independent approach compared to other Concept Drift detectors in Sec. 6.3.4. **(ii)** performance evaluation of the RSLVQ paired with the KSWIN detector against state-of-the-art models in Sec. 6.3.5. **(iii)** time and memory consumption during Concept Drift stream processing in Sec. 6.3.8 and Sec. 6.3.6. Detailed descriptions of datasets, competitive methods and the parameters are provided in Sec. 6.3.1 and Sec. 6.3.2. The performance metrics suitable for evaluation classifiers in stream settings are detailed in 6.3.3. The stream setup, including time steps, batch size, and Concept Drift occurrence, is introduced before the individual performance evaluations.

6.3.1 *Datasets*

We use six stream generators (synthetic data) and six real-world datasets. We follow the study design of [70], but additional, we simulate frequent gradual/abrupt reoccurring drift with streams having abrupt or gradual drift. This type of drift stream introduces reoccurring drift with a certain frequency. The frequency is specified in Sec. 6.3.4 and Sec. 6.3.5 respectively. An overview of the streams is given in Tab. 6.1. The synthetic and the real-world datasets are described in detail in Appendix 7.

Stream	# Instances	# Features	Type	Drift	# Classes
SEA_a	1,000,000	3	Synthetic	A	2
SEA_g	1,000,000	3	Synthetic	G	2
$MIXED_a$	1,000,000	4	Synthetic	A	2
$MIXED_g$	1,000,000	4	Synthetic	G	2
RTG	1,000,000	10	Synthetic	N	2
RBF_m	1,000,000	10	Synthetic	I_m	5
RBF_f	1,000,000	10	Synthetic	I_f	5
HYPER	1,000,000	10	Synthetic	I_f	2
POKER	829,201	11	Real	-	10
GMSC	120,269	11	Real	-	2
ELEC	45,312	8	Real	-	2
COV	581,012	27	Real	-	7
SQRE	200,000	2	Real	-	4
WTHR	18,160	8	Real	-	2

Table 6.1: Configuration of the data sets (A: Abrupt Drift, G: Gradual Drift, I_m: Moderate Incremental Drift, I_f: Fast Incremental Drift and N: No Drift)

6.3.2 *Implementation Details*

Competitive Methods. The KSWIN detector is tested against other commonly used detectors, Adaptive Sliding Window (**ADWIN**) [20], Drift-Detection Method (**DDM**) [13] and Early Drift Detection Method (**EDDM**) [19]. Further, KSWIN combined with reactive RSLVQ (RRSLVQ) is evaluated against the Hoeffding Adaptive Tree (**HAT**) [31], OzaBagging [23], Adaptive Random Forest (**ARF**) [101], Self Adjusted Memory KNN (**SAMKNN**) [105] and Robust Soft Learning Vector Quantization (**RSLVQ**) [12] as the baseline.

Parameters. The parameters for the memory management of KSWIN are $(r = 30, n = 300)$ and $\alpha = 0.0001$ for the Kolmogorov-Smirnoff test. The ADWIN parameter is $\alpha = 0.002$. The DDM algorithm has a minimum test set size preset to 30 and a detection threshold set to 3. The window size of EDDM is also set to 30 with a detection threshold of $\beta = 0.95$. The parameters of the remaining algorithms are set to standard settings obtainable at [120].

Experimental Setup. The length of the streams is fixed to $t_{max} = 1,000,000$. We use the interleaved test-then-train accuracy and the kappa statistics as described below. At t_{max}, both scores are accumulated and become the overall accuracy/kappa scores shown in the subsequential evaluations. The remaining experimental details are described individually for evaluating Concept Drift detectors in Sec. 6.3.4 and stream classifiers in Sec. 6.3.5.

6.3.3 *Performance Metrics*

Due to the stream setting with the accompanying assumptions, the standard procedure for evaluating machine learning models can not be directly applied as in the previous chapters. We implement the following suggested [70] metrics for an appropriate evaluation:

Interleaved Test-Then-Train. In a stream setting, prediction performance is measured with interleaved test-then-train accuracy [70], [104]. It is the moving average accuracy including the current sequence of data at time t by

$$A(S) = \frac{1}{t} \sum_{i=1}^{t} 1(h_{i-1}(\mathbf{x}_i) = y_i). \tag{6.24}$$

Where $1(\cdot)$ is an indicator function and is one for correct classification and zero otherwise. Each example can be used to test the model before it is used for training, and from this, the accuracy can be incrementally updated. When intentionally performed in this order, the model is always being tested on examples it has not seen. This scheme has the advantage that no holdout set is needed for testing, making maximum use of the available data. It also ensures a smooth plot of accuracy over time, as each example will become increasingly less significant to the overall average. The disadvantage of the metric is that it makes it difficult to accurately separate and measure training and testing times. Also, the true accuracy that an algorithm can achieve at a given point is unbalanced. Algorithms will be punished for early mistakes, regardless of the level of accuracy they can eventually deliver. However, this effect will diminish over time [35]

Kappa Statistics. We also report Kappa statistics [70], which measures classifiers performance w.r.t. class imbalance:

$$\kappa = \frac{p_0 - p_c}{1 - p_c}, \tag{6.25}$$

where p_0 is the classifiers pre-sequential accuracy and p_c is the probability, that a chance classifier makes a correct prediction. If the classifier is always correct, then $\kappa = 1$. If the predictions coincide with the correct predictions as often as those of the chance classifier, then $\kappa = 0$. The kappa statistic is computable for any sequence $\mathbf{x}_t$ at time t and is in the range of $-1 \geq \kappa \geq 1$.

6.3.4 *Performance Results: Concept Drift Detectors*

Setup. Each tested stream has 100,000 time steps with a batch size of ten per time step and in summary, there are one million samples per stream. Every time step of occurring drifts is compared against the detectors' predicted time step, either zero for no Concept Drift and one if drift is detected. In summary, there are ten standard Concept Drift

Detectors	True Negative	False Positive	False Negative	True Positive
KSWIN	748,956	51,040	379,345	20,659
ADWIN	799,823	173	399,933	71
EDDM	799,941	55	399,971	33
DDM	799,984	12	400,004	0

Table 6.2: Confusion matrix of Concept Drift detectors showing that KSWIN detects the most drifts but also has most false positives. It is tested on standard concept streams (8) and frequent reoccurring versions (4) of them. Overall, there are 400,004 Concept Drifts within 120,000,000 time steps. Each of the eight standard streams has one drift, and each of the frequent reoccurring Concept Drift streams has 99,999 drifts.

streams with one Concept Drift per stream and six frequent reoccurring Concept Drift streams with 99,999 occurrences of Concept Drift per stream. The results of the Concept Drift detectors are split up into two parts. For both parts, the detectors test every data dimension separately and not the performance values of classifiers.

Results Sensitivity. The results of the first part of the evaluation are given in Tab. 6.2. It shows a confusion matrix of the detectors tested on the listed Concept Drift stream generators given in Tab. 6.1 plus the frequent reoccurring versions. The KSWIN algorithm detects the Concept Drifts and has, by far, the most true-positives but also the most false-positive. The detection accuracy of KSWIN is roughly five percent, while the detection rate of the remaining detectors is roughly 0.001%. To summarize, Concept Drift detection per dimension on stream data is somewhat unreliable, especially if the number of Concept Drifts is high. Most of the time, the detectors are just missing Concept Drift because of insensitivity. The KSWIN detector is an improvement to the state-of-the-art detectors by means of detection rate. The false-positive rate should be tackled in future work. However, as shown in the prototype adaptation strategy in Sec. 6.2 and shown in the following experiments, where KSWIN is paired with the Naive Bayes classifier, false positives are not critical.

Results Accuracy. The second part is shown in Tab. 6.3 and gives the prediction performance of the Naive Bayes classifier combined with a given detector. If a detector notices Concept Drift, the current underlying learning model of the detector is discarded, and a new one is learned with the current batch of data. This experiment ensures that every detector uses the same underlying classifier, and the classifier does not influence the performance. Inspecting each drift separately, the KSWIN algorithm is only best at four out of 16 streams. However, KSWIN has the best mean performance. The second best algorithm is ADWIN. The results at MIXED are interesting because the algorithms that do not recognize the Concept Drift are not able to switch to the new concept. EDDM and DDM are particularly affected by this and also ADWIN at frequent reoccurring MIXED. However, apart from MIXED, the performance of the detectors on the frequently reoccurring drift streams is not much worse compared to the remaining streams.

Streams	KSWIN	ADWIN [20]	EDDM [19]	DDM [13]
MIXED$_a$	86.5	**90.2**	50.6	45.0
MIXED$_g$	**86.4**	50.0	50.0	50.0
Hyperplane	58.0	**60.9**	58.9	58.8
RTG	63.3	**70.5**	63.2	66.6
RBF$_f$	**65.9**	53.0	49.5	50.0
RBF$_m$	64.6	**65.1**	50.3	60.2
SEA$_a$	84.0	**89.1**	70.4	88.8
SEA$_g$	83.3	83.0	67.9	**84.0**
Standard Mean	**74.6**	70.9	58.5	64.4
MIXED$_{ra}$	**78.3**	49.0	49.0	49.0
MIXED$_{rg}$	**77.7**	49.2	49.0	49.8
SEA$_{ra}$	83.9	**87.7**	81.0	87.5
SEA$_{rg}$	83.0	**86.9**	80.6	84.7
Reoccurring Mean	**82.2**	71.5	67.9	70.7
Overall Mean	**78.4**	71.2	63.2	67.5

Table 6.3: Interleaved mean accuracy of Concept Drift detectors with Naive Bayes as the underlying classifier. Tested on standard Concept Drift streams and the frequent reoccurring versions of them separated in two parts. The results are overall similar comparing prediction performance, but in the mean, the KSWIN detector outperforms the remaining detectors. Winner marked bold. a = abrupt, f = incremental fast, m = incremental medium, g = gradual, ra = abrupt frequent reoccurring, rg = gradual frequent reoccurring

6.3.5 *Performance Results: Stream Classifiers*

Setup. In this setting, the frequent reoccurring drifts starting at sample 2000 and, further, every 1000 samples after the last drift is finished. The width of gradual drift is 1000. For every non-frequent reoccurring stream, we are adding abrupt/gradual Concept Drift at position 500,000, while gradual drift has a width of 50,000.

Results Accuracy. The results of the prediction performance of Concept Drift classifiers on the data streams are presented in Tab. 6.4. Our approach shows a boost in performance to baseline RSLVQ and is comparable to other Concept Drift classifiers like HAT and OZA. The RSLVQ is the worst classifier on the tested real-world datasets, mostly due to the worse COV and SQRE performance. The RRSLVQ performs about eight percent better when looking at the mean of the real-world datasets. However, it is still worse than the other classifiers, especially SAMKNN, which performed best. At COV, where RSLVQ had an accuracy of 36 %, RRSLVQ improved the prediction to an accuracy of 73 %, which is approximately equal to the ARF performance.

On the artificial data streams, the RSLVQ performs worst with a mean accuracy of 69 %. The RRSLVQ does a better job on the synthetic Concept Drift streams due to the

Streams	ARF [101]	SAMKNN [104]	HAT [31]	RSLVQ [12]	RRSLVQ	OZA [23]
COV	69.9	**89.4**	82.2	36.5	72.8	69.1
ELEC	**85.6**	72.5	77.5	62.2	64.5	74.0
POKER	**80.7**	79.5	64.5	72.1	57.8	79.0
WTHR	73.9	77.8	67.7	65.02	67.0	**78.3**
GMSC	**93.0**	92.7	88.0	78.0	92.1	92.0
SQRE	51.1	**96.6**	73.0	33.4	59.9	44.4
REAL MEAN	75.7	**84.8**	75.5	57.9	68.4	75.6
MIXED_a	93.4	**99.0**	87.0	73.3	89.3	97.5
MIXED_g	91.2	**99.0**	87.0	73.3	89.4	97.5
Hyperplane	50.9	56.0	62.2	52.5	**62.1**	57.3
RTG	58.3	70.1	65.8	57.7	**80.8**	65.6
RBF_f	74.6	**93.7**	62.4	59.4	66.5	93.5
RBF_m	79.8	93.9	73.4	59.9	68.0	**95.6**
SEA_a	88.7	89.0	83.6	80.8	**89.2**	88.1
SEA_g	87.6	88.7	83.0	80.3	**90.0**	87.5
MIXED_{ra}	84.9	**85.4**	52.9	73.2	77.9	54.6
MIXED_{rg}	**88.2**	85.4	52.9	72.3	77.9	54.7
SEA_{ra}	86.7	87.8	83.1	80.2	88.6	**94.7**
SEA_{rg}	85.1	88.1	82.7	80.5	89.1	**94.7**
ARTIFICIAL MEAN	80.8	**86.3**	73.0	69.0	80.7	81.8
OVERALL MEAN	79.1	**85.8**	73.8	65.3	74.6	79.7

Table 6.4: Interleaved mean **accuracy on data streams**. Moving average of accuracy on one million samples. Winner marked bold. a = abrupt, f = incremental fast, m = incremental medium, g = gradual, ra = abrupt frequent reoccurring, rg = gradual frequent reoccurring

drift detector and thus has better accuracy than RSLVQ and HAT with 81 %. Again, SAMKNN performed best with an accuracy of 86 %. Overall, RRSLVQ performed approximately equal to OZA with an accuracy of 81 %, which is a performance increase of twelve percent compared to the baseline RSLVQ.

Results Kappa. The results of the streaming experiment measured by Kappa statistics are shown in Tab. 6.5. The improved performance of RRSLVQ compared to RSLVQ is also given w.r.t. Kappa. This means that RRSLVQ also performs better on imbalanced data and has no tendency only to predict one label. Especially on COVTYPE, where RSLVQ could not separate the classes of the dataset, RRSLVQ showed a Kappa score of 0.46. The improved score is not only related to frequent reoccurring drift. It is also given when using other types of drift. On the Hyperplane generator, RRSLVQ performed best and is the only algorithm with a Kappa score > 0.2. This is because the distribution of the stream changes naturally very often and thus needs a sensible Concept Drift detector. However, the Kappa score is still very low on this dataset. Given the performance of the baseline classifier, the RRSLVQ does a good job of improving the RSLVQ on Concept Drift streams.

On the real-world streams, the Kappa score is also better, but the overall improve-

Streams	ARF [101]	SAMKNN [104]	HAT [31]	RSLVQ [12]	RRSLVQ	OZA [23]
COV	0.43	**0.83**	0.71	0.0	0.46	0.82
ELEC	**0.67**	0.43	0.53	0.22	0.26	0.46
POKER	**0.67**	0.63	0.37	0.50	0.35	0.61
WTHR	0.34	0.44	0.26	0.27	0.27	**0.47**
GMSC	**0.08**	0.0	0.0	-0.01	0.0	0.0
SQRE	0.35	**0.95**	0.64	0.11	0.21	0.21
REAL MEAN	0.43	**0.55**	0.42	0.18	0.26	0.43
MIXED_a	0.87	**0.98**	0.74	0.46	0.81	0.95
MIXED_g	0.82	**0.98**	0.74	0.46	0.81	0.95
Hyperplane	0.01	0.12	0.24	0.05	**0.26**	0.15
RTG	0.16	**0.34**	0.27	0.13	0.26	0.17
RBF_{if}	0.47	**0.87**	0.24	0.19	0.25	0.87
RBF_{im}	0.59	0.88	0.43	0.16	0.28	**0.91**
SEA_a	**0.75**	0.73	0.60	0.54	0.74	0.71
SEA_g	0.72	0.76	0.64	0.59	**0.79**	0.74
MIXED_{ra}	0.70	**0.71**	0.06	0.30	0.44	0.09
MIXED_{rg}	**0.76**	0.71	0.06	0.30	0.43	0.09
SEA_{ra}	0.70	0.71	0.61	0.58	0.73	**0.89**
SEA_{rg}	0.66	0.74	0.63	0.58	0.76	**0.89**
ARTIFICIAL MEAN	0.60	**0.71**	0.44	0.36	0.55	0.62
OVERALL MEAN	0.54	**0.65**	0.43	0.30	0.46	0.55

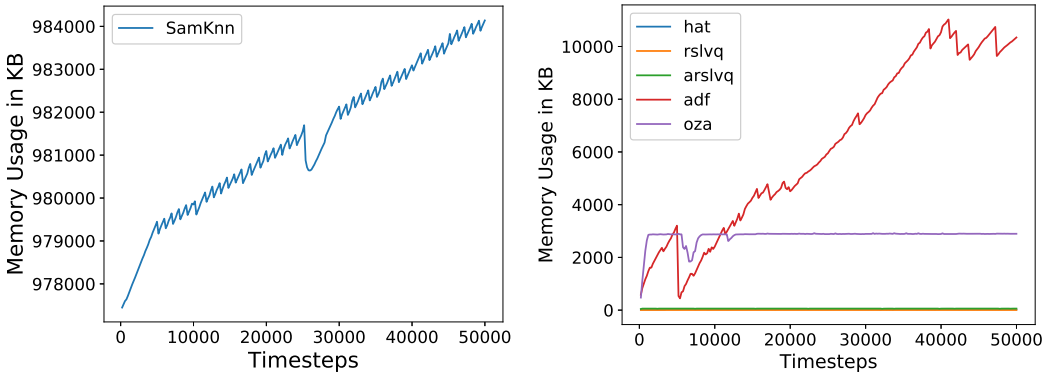
Table 6.5: Interleaved test-then-train **Kappa statistics on data streams.** Moving average of Kappa on one million samples. Winner marked bold. a = abrupt, if = incremental fast, im = incremental medium, g = gradual, ra = frequent reoccurring

ment is small. This is affected by the fact that RSLVQ did a better job at distinguishing the ten classes of the POKER dataset. Overall, SAMKNN performed best with a score of 0.65, while RRSLVQ performed very similar to HAT at 0.45.

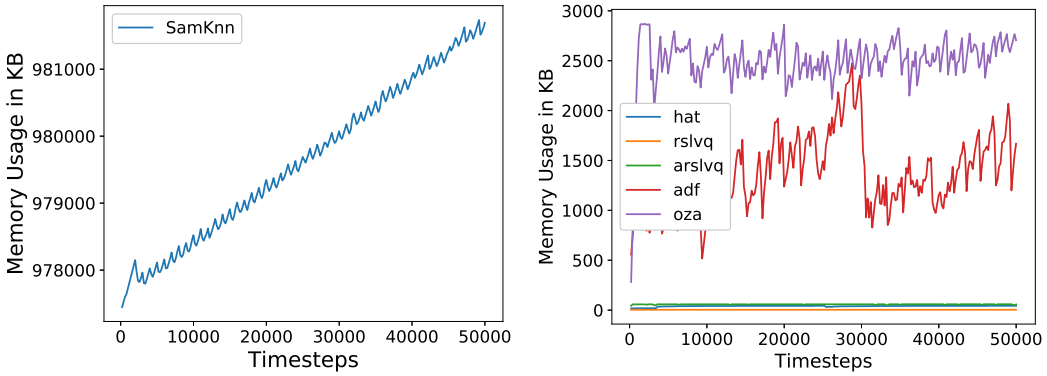
The advantages of the RRSLVQ are stability in Concept Drift and constant model size and time. This makes processing stream data feasible on a limited technical device. The prototypes provide an interpretable model, which is a missing feature in competing algorithms. The prediction performance is comparable to OZA and HAT. Moreover, comparing the prediction results in Tab. 6.4 to the time and memory consumption in Fig. 6.8 and 6.6 in the following sections, we will see the trade-off relation between prediction performance and model size and time.

6.3.6 *Memory Analysis*

An overview of memory consumption during stream processing is shown in Fig. 6.6. It shows that the RSLVQ variants and HAT have very low memory requirements and are stable in memory consumption during the drift. Further, we measured linear memory consumption. This validates Sec. 6.2. The stable consumption is not provided by the



(a) Memory SAMKNN MIXED$_a$ *(b)* Memory Others MIXED$_a$



(c) Memory SAMKNN MIXED$_{ra}$ *(d)* Memory Others MIXED$_{ra}$

Figure 6.6: Memory usage of stream classifiers tested on MIXED stream with abrupt drift (top) and frequent abrupt reoccurring drift (bottom). SAMKNN is plotted separately due to scaling issues.

remaining classifiers, and at the detection of drifts, the memory usage fluctuates. Note that this section applies the same setup as Sec. 6.3.5.

6.3.7 *Stability Analysis*

The RRSLVQ provides stable performance during drift and high adaptation rate shown in Fig. 6.7, superior w.r.t. other methods. Other variants of Concept Drift handling, like incremental prototype insertion or prototype replacement based on the ADWIN detector, are not providing stability during a drift. Hence, both parts, the KSWIN and the prototype adaptation strategy, are equally relevant for the stability of RRSLVQ.

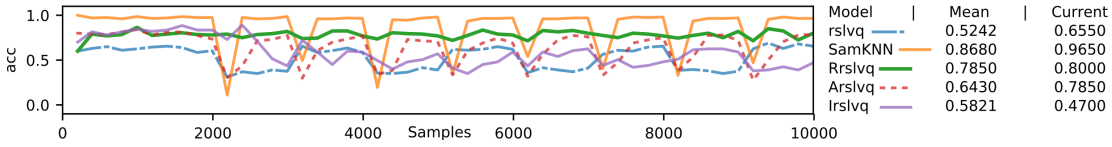
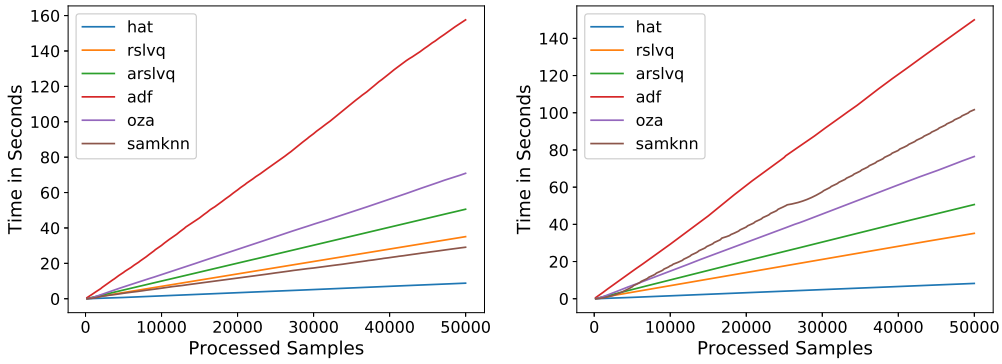


Figure 6.7: Performance of baseline RSLVQ, RRSLVQ, Incremental RSLVQ (Irslvq), ADWIN RSLVQ (Arslvq), and SAMKNN [104] on MIXED stream in SciKit-Multiflow [120]. The plot shows clear drops in the performance of non-RRSLVQ methods during abrupt Concept Drift. From point 2000, drift happens every 1000 points after the last drift is finished. The line shows accuracy over the last 200 samples. Best viewed on computer display.

6.3.8 *Time Analysis*

The time result is plotted in Fig. 6.8 as incremental time in seconds per processed sample. It also validates the time complexity of RRSLVQ and further shows that every other stream classifier also has linear time complexity. However, the RRSLVQ needs less total time than the ARF and OZA. The RSLVQ is slightly faster than the RRSLVQ because of the missing Concept Drift detector, non-momentum-based SGD, and prototype adaptation strategy. The HAT algorithm is the fastest. All in all, the classifiers do not need a significant amount of additional time to handle Concept Drift. Note that this section applies the same setup as Sec. 6.3.5.



(a) Incremental Time MIXED$_a$ *(b)* Incremental Time MIXED$_{ra}$

Figure 6.8: Incremental time in seconds per amount of samples processed on MIXED generator with abrupt drift (left) and frequent reoccurring drift (right). All tested methods are increasing linearly in time due to linear complexity per batch.

6.4 DISCUSSION

In the beginning, we asked the question of how to adapt the RSLVQ to Concept Drift streams. The proposed solution is integrating the Concept Drift detector KSWIN, combining the Kolmogorov-Smirnoff test and memory management. Further, we enhanced the RSLVQ via active prototype insertion and momentum-based gradient descent. The proposed RRSLVQ method is a significant improvement to the original RSLVQ. Especially in streams with high rates of drift, the RRSLVQ shows remarkable stability over time. KSWIN seems to detect occurring changes in stream data and supports the Concept Drift handling process with good indicators at a given time.

The KSWIN algorithm best detects drift at the given streaming data based on the experimental setting. The adaptation strategy paired with the momentum-based gradient descent is helpful to minimize performance losses during the drift. Compared to other stream approaches, the RRSLVQ provides a straightforward and interpretable model. The memory and time complexity is easy to bound and well-suited for embedding systems. The prediction performance is comparable with OzaBaggingADWIN [23] and Hoeffding Adaptive Tree [31].

Limitations. The improvements made do not only bring advantages. The sliding window uses Uniform sampling over the last concept, a common but trivial assumption with the potential risk of missing some Concept Drift by sampling around them. Further, dimension-wise testing makes the KSWIN detector inefficient and very sensitive to high-dimensional data streams. Finally, the proposed solutions require many hyperparameters like sampling size, Gaussian kernel width, and prototype set size. Even tuning during stream processing does not provide satisfactory solutions to the problem, making it hard to apply if standard parameters do not provide adequate performance.

Future Directions. The RRSLVQ is not the only stream classifier with a large number of hyperparameters. Other solutions like OzaBaggingADWIN [23], Adoptive Random Forest [101] or Hoeffding Adaptive Tree [31] share this drawback. Many hyperparameters constrain a solution to a stream holding certain assumptions, which may not hold even in synthetic streams. In the next years, we will see the development of parameter-free methods, which has already started, considering methods like Dynamic Adaptive Window Independence Drift Detector (DAWIDD) [171].
The insensitivity and inefficiency of current Concept Drift detectors will be tackled. Current feature based detectors are faster in detecting Concept Drift than performance based detectors. However, current sample-testing procedures rely on dimension-wise testing, making the approach very slow. We already see the development of new models, naturally able to detect high-dimensional Concept Drift streams. While not the most recent idea [126], the community has just grasped the concept of high dimensional Concept Drift streams. In the next years, we will see more developments like coreset-detectors [170], or one-class-detectors [166], which are by definition of the underlying mechanics able to deal with high dimensional Concept Drift streams.

7 CONCLUSION AND OUTLOOK

Non-Stationary environments played a vital role in this dissertation. First, we assumed finite datasets samples from a labeled and an unlabeled domain plus the availability of both at learning and adaptation time. In chapter 3, we saw how the intuition developed in chapter 2 is guiding us to develop a geometric Domain Adaptation algorithm (GDA) by finding a Least-Squares solution to match augmented source and target samples. The obtained domain is invariant to domain-specific orthogonal transformations, and, therefore, it is precisely the difference between the domain singular values. While reasonably easy, it provides the roadmap for the following chapters. In chapter 4, we used the same Least-Squares problem with two main differences: first, we directly formulated the optimization problem in the respective domain subspaces. Second, we found a closed-form solution by obtaining a subspace projector modifying the source data to be part of the target data. We called the approach Subspace Override (SO) and showed that the differences of singular values from a domain subspace bound the obtained domains, making it more adaptive than GDA. SO is extended to Uniform Nyström (NSO) and class-wise Nyström (cNSO). While the latter outperforms competitive methods over various tasks, it is easily parameterized and the fastest. Concerning the Domain Adaptation theory, all the above solutions obtain a representation, focusing on lower domain separability, while a classifier is trained afterward.

Deep Domain Adaptation simultaneously minimizes a classification loss and domain separability by adding a domain adaptive regularization within training. In chapter 5, we used the above findings and directly minimized the difference of singular values from two domains given random batches called Spectral Loss (SL). The relationship of singular values to statistical moments allows direct minimization of statistical moments on the principal axes. SL wrongly treats all singular values equally important while outperforming recent moment-matching networks. The proposed relevance-based Spectral Loss (RSL) assumes that the smallest singular values are less relevant for the adaptation and shrinks them. Combing domain adversarial learning, RSL, and Spectral-Normalization, we obtain the Adversarial Spectral Adaptation Network (ASAN). It outperforms competitive networks while having robust parameterization and providing guarantees for expected separability.

By assuming an infinite dataset, which is analyzed only by a short, temporal snapshot of a dataset, we model a data stream scenario. If the stream is affected by Concept Drift, then the subject is changed to Concept Drift stream classification. We introduced the RSLVQ for stream classification and, in combination with Kolmogorov-Smirnoff, obtained a model able to detect drift with active and passive adaptation procedures called Reactive RSLVQ (RRSLVQ). The obtained model provides a bound for adaptability during a drift, and theoretical and empirical evidence confirms stable performance

and fast concept adaptation while competitive to recent solutions.

Limitations. We have shown throughout this thesis that various Domain Adaptation ideas lead to competitive and outperforming approaches. However, all have certain limitations in adaptation. It is the only term of the Domain Adaptation theory left out and not tackled by all methods: the expected divergence between labeling functions. In other words, the difference in class conditional distributions of source and target domain. While we surely changed these functions by the above data modifications, we have neither a theoretical nor practical guarantee that these changes provide a reasonable solution to the problem. Many recent Domain Adaptation algorithms share this limitation.

Contributed methods are separable either in shallow or deep Domain Adaptation. While shallows methods are faster and task-independent by having a sparse model, they cannot compete with the predictive performance of current deep networks. The latter model, however, is unrivaled larger with a longer training time. Hence, both categories counterplay each other. The purpose of Domain Adaptation is ultimately to increase the performance of a predictive function in an application domain; however, one has to choose a Domain Adaptation technique that either comes with time and memory or performance restrictions.

The contributed stream classification model, namely RRSLVQ, has two main limitations: (i) It has many parameters for the Concept Drift detector and the RSLVQ model. The tuning of the parameters is non-trivial and time-consuming in a stream setting. Hence, the standard parameters are used, constraining the data stream with certain assumptions, theoretically derived. Most certainly, these settings will not hold for every arbitrary stream. If the constraints are violated, we can not expect accustomed performance and stability. (ii) the dimension-wise Concept Drift detection misses a drift that occurs across multiple dimensions because the change in a single feature dimension may be too small to be noticed. Further, the high number of tests certainly will lead to false testing. In contrast, performance stability is ensured in the event of an incorrect test - however, the stability relies on a reasonable hyperparameter choice.

Future Outlook. Current stream classifiers are evaluated on streams that by no means address current challenges occurring in real-world data streams, namely a fast change of appearing Concept Drifts. Moreover, current algorithms are not evaluated on high-dimensional data streams with a high number of classes. With a non-trivial parametrization, it is already shown that current approaches struggle to achieve substantial performance on such streams. New publications in the field point in the direction of non-parametric Concept Drift detectors or apply techniques to deal with high-dimensional data rather than dimension-wise procedures.

For Domain Adaptation, first and foremost, the expected divergence of labeling functions will be addressed. The central concept used by shallow or deep techniques will be remarkably similar: incorporating local structures, metric or manifold learning,

and abandoning the somewhat unreliable pseudo labeling techniques based on the confidence of a classifier in the middle of the adaptation process. The thesis contributions motivate a connection between singular values and domain separability, leading to a broader application of the Domain Adaptation theory. Simultaneously, the entropy and connected divergence measures will be reinterpreted to tackle the above-mentioned expected difference in label functions.

Last but not least, the community is currently addressing many new subcategories. Hence, the research area of Domain Adaptation seems to be renamed to closed-set Domain Adaptation, assuming the same source and target classes. The number of contributions to variants such as partial, open set, or universal Domain Adaptation, assuming missing, partly, or completely unknown classes in the target domain, is already increasing. Finally, the general data protection regulation (GDPR[1]) of the European Union poses new challenges on data availability, currently addressed by the community through source-free Domain Adaptation [199] - basically the adaptation of an already trained source model. Other emerging technologies addressing the GDPR are federated learning [160] and impose new opportunities for distributed private learning, particularly interesting for Domain Adaptation [181].

The current volume of new publications in the research areas discussed, especially for Domain Adaptation, is very high. The above limitations and outlooks describing a generalized point of view, and at the time of writing this thesis, the first publications addressing these issues have already been published. By overcoming the above problems and after more than 20 years of research, the techniques of Concept Drift and Domain Adaptation will be mature enough to be used productively by people who are not particularly experienced in these areas, or as it is called in other fields - democratization of artificial intelligence.

1 https://gdpr.eu

PUBLICATIONS IN THE CONTEXT OF THIS THESIS

CONFERENCE ARTICLES

[122] C. Raab and F.-M. Schleif, "Sparse Transfer Classification for Text Documents," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, F. Trollmann and A.-Y. Turhan, Eds., vol. 11117 LNAI, Springer International Publishing, 2018, pp. 169–181.

[123] ——, "Transfer learning for the probabilistic classification vector machine," in *Proceedings of the Seventh Workshop on Conformal and Probabilistic Prediction and Applications*, A. Gammerman, V. Vovk, Z. Luo, E. Smirnov, and R. Peeters, Eds., ser. Proceedings of Machine Learning Research, vol. 91, PMLR, 2018, pp. 187–200.

[124] F.-M. Schleif, C. Raab, and P. Tino, "Sparsification of Indefinite Learning Models," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11004 LNCS, 2018, pp. 173–183, ISBN: 9783319977843.

[151] C. Raab, M. Heusinger, and F. M. Schleif, "Reactive Soft Prototype Computing for frequent reoccurring Concept Drift," in *27th European Symposium on Artificial Neural Networks, ESANN 2019, Bruges, Belgium, April 24-26, 2019*, 2019, pp. 437–442.

[167] M. Heusinger, C. Raab, and F.-M. Schleif, "Analyzing Dynamic Social Media Data via Random Projection - A New Challenge for Stream Classifiers," in *2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, IEEE, May 2020, pp. 1–8.

[168] ——, "Passive Concept Drift Handling via Momentum Based Robust Soft Learning Vector Quantization," in *Advances in Intelligent Systems and Computing*, A. Vellido, K. Gibert, C. Angulo, and J. D. Martín Guerrero, Eds., vol. 976, Cham: Springer International Publishing, 2020, pp. 200–209.

[178] M. Münch, C. Raab, M. Biehl, and F.-M. Schleif, "Structure Preserving Encoding of Non-euclidean Similarity Data," in *Proceedings of the 9th International Conference on Pattern Recognition Applications and Methods*, SCITEPRESS - Science and Technology Publications, 2020, pp. 43–51.

[183] C. Raab, P. Meier, and F.-M. Schleif, "Domain Invariant Representations with Deep Spectral Alignment," in *28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2020, pp. 509–514.

[184] C. Raab and F.-M. Schleif, "Low-Rank Subspace Override for Unsupervised Domain Adaptation," in *KI 2020: Advances in Artificial Intelligence*, U. Schmid, F. Klügl, and D. Wolter, Eds., Cham: Springer International Publishing, 2020, pp. 132–147, **Best Paper Award**.

[198] C. Raab, P. Väth, P. Meier, and F.-M. Schleif, "Bridging Adversarial and Statistical Domain Transfer via Spectral Adaptation Networks," in *Proceedings of the Asian Conference on Computer Vision (ACCV)*, Nov. 2021, pp. 457–473.

JOURNAL ARTICLES

[169] M. Heusinger, C. Raab, and F.-M. Schleif, "Passive concept drift handling via variations of learning vector quantization," *Neural Computing and Applications*, vol. 6, Aug. 2020.

[179] M. Münch, C. Raab, M. Biehl, and F.-m. Schleif, "Data-Driven Supervised Learning for Life Science Data," *Frontiers in Applied Mathematics and Statistics*, vol. 6, no. November, pp. 1–15, Nov. 2020.

[182] C. Raab, M. Heusinger, and F.-M. Schleif, "Reactive Soft Prototype Computing for Concept Drift Streams," *Neurocomputing*, vol. 416, pp. 340–351, Nov. 2020.

[185] C. Raab and F.-M. Schleif, "Transfer learning extensions for the probabilistic classification vector machine," *Neurocomputing*, vol. 397, pp. 320–330, Jul. 2020.

[187] F.-M. Schleif, C. Raab, and P. Tino, "Sparsification of core set models in non-metric supervised learning," *Pattern Recognition Letters*, vol. 129, pp. 1–7, Jan. 2020.

REFERENCES

[1] W. Hoeffding, "Probability Inequalities for Sums of Bounded Random Variables," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 13–30, Mar. 1963.

[2] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*, ser. The Wadsworth statistics/probability series. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software, 1984.

[3] L. Qi, "Some simple estimates for singular values of a matrix," *Linear Algebra and Its Applications*, vol. 56, pp. 105–119, 1984.

[4] R. S. Sutton, "Two problems with backpropagation and other steepest-descent learning procedures for networks," in *Proc. 8th annual conf. cognitive science society*, Hillsdale, NJ: Erlbaum, 1986, pp. 823–831.

[5] A. Sato and K. Yamada, "Generalized Learning Vector Quantization," *Advances in Neural Information Processing Systems, NIPS*, no. 8, pp. 423–429, 1995.

[6] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural Networks*, vol. 12, no. 1, pp. 145–151, 1999.

[7] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '00*, 2000, pp. 71–80, ISBN: 1581132336.

[8] T. Papadopoulo and M. I. A. Lourakis, "Estimating the Jacobian of the Singular Value Decomposition: Theory and Applications," in *Computer Vision - ECCV 2000*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 554–570, ISBN: 978-3-540-45054-2.

[9] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '01*, vol. 28, New York, New York, USA: ACM Press, 2001, pp. 97–106, ISBN: 158113391X.

[10] N. C. Oza and S. Russell, "Experimental comparisons of online and batch versions of bagging and boosting," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '01*, New York, New York, USA: ACM Press, 2001, pp. 359–364, ISBN: 158113391X.

[11] C. Williams and M. W. Seeger, "Using the Nystrom Method to Speed Up Kernel Machines," in *NIPS Proceedings*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds., vol. 13, MIT Press, 2001, pp. 682–688, ISBN: 9788578110796. arXiv: arXiv:1011.1669v3.

[12] S. Seo, M. Bode, and K. Obermayer, "Soft nearest prototype classification," *IEEE Transactions on Neural Networks*, vol. 14, no. 2, pp. 390–398, 2003.

[13] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with Drift Detection," in *Advances in Artificial Intelligence – SBIA 2004*, A. L. C. Bazzan and S. Labidi, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 286–295, ISBN: 978-3-540-28645-5.

[14] D. Kifer, S. Ben-David, and J. Gehrke, "Detecting change in data streams," in *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, ser. VLDB '04, VLDB Endowment, 2004, pp. 180–191, ISBN: 0120884690.

[15] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Proceedings of the seventh {ACM} {SIGKDD} international conference on Knowledge discovery and data mining, San Francisco, CA, USA, August 26-29, 2001*, 2004, pp. 377–382.

[16] R. S. Varga, *Geršgorin and His Circles*, ser. Springer Series in Computational Mathematics. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, vol. 36, ISBN: 978-3-540-21100-6.

[17] N. J. Higham, "Computing the Polar Decomposition—with Applications," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 4, pp. 1160–1174, 2005.

[18] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005, ISBN: 026218253X.

[19] M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, and R. Morales-Bueno, "Early Drift Detection Method," *4th ECML PKDD International Workshop on Knowledge Discovery from Data Streams*, vol. 6, pp. 77–86, 2006.

[20] A. Bifet and R. Gavaldà, "Kalman Filters and Adaptive Windows for Learning in Data Streams," in *Discovery Science*, L. Todorovski, N. Lavrač, and K. P. Jantke, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 29–40, ISBN: 978-3-540-46493-8.

[21] J. Gama and G. Castillo, "Learning with Local Drift Detection," in *Advanced Data Mining and Applications*, X. Li, O. R. Zaïane, and Z. Li, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 42–55, ISBN: 978-3-540-37026-0.

[22] T. K. Landauer, "Latent Semantic Analysis," *Encyclopedia of Cognitive Science*, vol. 3, no. 11, p. 4356, 2006.

[23] N. Oza, "Online Bagging and Boosting," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Waikoloa, Hawaii, USA, October 10-12, 2005*, vol. 3, IEEE, 2006, pp. 2340–2345, ISBN: 0-7803-9298-1. arXiv: /citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.32.8889 [http:].

[24] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira, "Analysis of Representations for Domain Adaptation," in *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. C. Platt, and T. Hoffman, Eds., MIT Press, 2007, pp. 137–144.

[25] W. Dai, G.-R. Xue, Q. Yang, and Y. Yu, "Co-clustering based classification for out-of-domain documents," in *Proceedings of the 13th {ACM} {SIGKDD} International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007*, 2007, p. 210.

[26] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proceedings of the 24th international conference on Machine learning - ICML '07*, New York, New York, USA: ACM Press, 2007, pp. 193–200, ISBN: 9781595937933.

[27] N. Salkind, *Encyclopedia of Measurement and Statistics*, N. Salkind, Ed. 2455 Teller Road, Thousand Oaks California 91320 United States of America: Sage Publications, Inc., 2007, pp. 103–107, ISBN: 9781412916110.

[28] R. Bro, E. Acar, and T. G. Kolda, "Resolving the sign ambiguity in the singular value decomposition," *Journal of Chemometrics*, vol. 22, no. 2, pp. 135–140, Feb. 2008.

[29] Laurens van der Maaten and H. Geoffrey E., "Visualizing Data using t-SNE," *Journal of Machine Learning Research*, vol. 164, no. 2210, p. 10, 2008.

[30] K. B. Petersen and M. S. Pedersen, *The Matrix Cookbook*. Technical University of Denmark, 2008.

[31] A. Bifet and R. Gavaldà, "Adaptive Learning from Evolving Data Streams," in *Advances in Intelligent Data Analysis VIII*, N. M. Adams, C. Robardet, A. Siebes, and J.-F. Boulicaut, Eds., ser. Lecture Notes in Computer Science, vol. 5772, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 249–260, ISBN: 978-3-642-03914-0.

[32] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, "New ensemble methods for evolving data streams," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*, New York, New York, USA: ACM Press, 2009, p. 139, ISBN: 9781605584959.

[33] R. Elwell and R. Polikar, "Incremental learning in nonstationary environments with controlled forgetting," in *2009 International Joint Conference on Neural Networks*, IEEE, Jun. 2009, pp. 771–778, ISBN: 978-1-4244-3548-7.

[34] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY: Springer New York, 2009, ISBN: 978-0-387-84857-0.

[35] V. Attar, P. Sinha, and K. Wankhade, "A fast and light classifier for data streams," *Evolving Systems*, vol. 1, no. 3, pp. 199–207, 2010.

[36] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine Learning*, vol. 79, no. 1-2, pp. 151–175, May 2010.

[37] A. Cornuéjols, "On-Line Learning: Where Are We So Far?" In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6202 LNAI, 2010, pp. 129–147, ISBN: 3642163912.

[38] K. De Brabanter, J. De Brabanter, J. Suykens, and B. De Moor, "Optimized fixed-size kernel models for large data sets," *Computational Statistics & Data Analysis*, vol. 54, no. 6, pp. 1484–1504, Jun. 2010.

[39] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[40] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting Visual Category Models to New Domains," in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos, and N. Paragios, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 213–226, ISBN: 978-3-642-15561-1.

[41] K. Zhang and J. T. Kwok, "Clustered Nyström method for large scale manifold learning and dimension reduction," *IEEE Transactions on Neural Networks*, vol. 21, no. 10, pp. 1576–1587, 2010.

[42] E. Zhong, W. Fan, Q. Yang, O. Verscheure, and J. Ren, "Cross validation framework to choose amongst models and datasets for transfer learning," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6323 LNAI, no. PART 3, pp. 547–562, 2010.

[43] R. H. C. Lopes, "Kolmogorov-Smirnov Test," in *International Encyclopedia of Statistical Science*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 718–720, ISBN: 978-3-642-04898-2.

[44] E. Morvant, A. Habrard, and S. Ayache, "On the usefulness of similarity based projection spaces for transfer learning," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7005 LNCS, pp. 1–16, 2011.

[45] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2011.

[46] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[47] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2066–2073, ISBN: 9781467312264.

[48] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A Kernel Two-sample Test," *J. Mach. Learn. Res.*, vol. 13, pp. 723–773, 2012.

[49] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[50] W. Jiang and F.-l. Chung, "Transfer Spectral Clustering," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, PART 2, vol. 7524 LNAI, 2012, pp. 789–803, ISBN: 9783642334856.

[51] A. Talwalkar, S. Kumar, and M. Mohri, "Sampling Methods for the Nyström Method," *Journal of Machine Learning Research*, vol. 13, pp. 981–1006, 2012.

[52] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method," *Proceedings of the 12th annual conference on Genetic and evolutionary computation - GECCO '10*, p. 103, Dec. 2012. arXiv: 1212.5701.

[53] A. Bifet and R. Gavaldà, "Learning from Time-Changing Data with Adaptive Windowing," *Proceedings of the 2007 SIAM International Conference on Data Mining*, pp. 443–448, 2013.

[54] A. Bifet, B. Pfahringer, J. Read, and G. Holmes, "Efficient data stream classification via probabilistic adaptive windows," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, ser. SAC '13, New York, NY, USA: ACM, 2013, p. 801, ISBN: 978-1-4503-1656-9.

[55] A. Cline and I. Dhillon, *Handbook of Linear Algebra*, L. Hogben, Ed. Chapman and Hall/CRC, Nov. 2013, pp. 1027–1039, ISBN: 9780429185533.

[56] A. Fallis, "Probability & Mathematical Statistics," *Journal of Chemical Information and Modeling*, vol. 53, no. 9, pp. 1689–1699, 2013. arXiv: arXiv:1011.1669v3.

[57] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2960–2967, 2013.

[58] A. Gittens and M. W. Mahoney, "Revisiting the Nystrom Method for Improved Large-Scale Machine Learning," *Journal of Machine Learning Research*, vol. 17, 117:1–117:65, Mar. 2013. arXiv: 1303.1849.

[59] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2200–2207, 2013.

[60] K. Wankhade, T. Hasan, and R. Thool, "A survey: Approaches for handling evolving data streams," in *Proceedings - 2013 International Conference on Communication Systems and Network Technologies, CSNT 2013*, Apr. 2013, pp. 621–625.

[61] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, pp. 1–37, 2014. arXiv: 1010.4784.

[62] M. Ghifary, W. B. Kleijn, and M. Zhang, "Domain Adaptive Neural Networks for Object Recognition," in *PRICAI 2014: Trends in Artificial Intelligence*, D.-N. Pham and S.-B. Park, Eds., vol. 35, Cham: Springer International Publishing, 2014, pp. 898–904, ISBN: 978-3-319-13560-1.

[63] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2014, pp. 2672–2680.

[64] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15, Dec. 2014. arXiv: 1412.6980.

[65] M. Long, J. Wang, G. Ding, S. J. Pan, and P. S. Yu, "Adaptation regularization: A general framework for transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 5, pp. 1076–1089, 2014. arXiv: PAI.

[66] D. Nova and P. A. Estévez, "A review of learning vector quantization classifiers," *Neural Computing and Applications*, vol. 25, no. 3-4, pp. 511–524, 2014. arXiv: 1509.07093.

[67] M. Shao, D. Kit, and Y. Fu, "Generalized transfer subspace learning through low-rank constraint," *International Journal of Computer Vision*, vol. 109, no. 1-2, pp. 74–93, 2014.

[68] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep Domain Confusion: Maximizing for Domain Invariance," *arXiv*, 2014. arXiv: 1412.3474.

[69] R. Aljundi, R. Emonet, D. Muselet, and M. Sebban, "Landmarks-based kernelized subspace alignment for unsupervised domain adaptation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 07-12-June, IEEE, Jun. 2015, pp. 56–63, ISBN: 978-1-4673-6964-0.

[70] A. Bifet, G. de Francisci Morales, J. Read, G. Holmes, and B. Pfahringer, "Efficient Online Evaluation of Big Data Stream Classifiers," *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 59–68, 2015.

[71] I. Frias-Blanco, J. del Campo-Avila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-Diaz, and Y. Caballero-Mota, "Online and Non-Parametric Drift Detection Methods Based on Hoeffding's Bounds," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 3, pp. 810–823, Mar. 2015.

[72] 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*, 2015, pp. 1180–1189.

[73] ——, "Unsupervised Domain Adaptation by Backpropagation," in *Proceedings of the 32nd International Conference on Machine Learning, {ICML} 2015, Lille, France, 6-11 July 2015*, 2015, pp. 1180–1189.

[74] S. Hauberg, O. Freifeld, A. B. L. Larsen, J. W. Fisher, and L. K. Hansen, "Dreaming More Data: Class-dependent Distributions over Diffeomorphisms for Learned Data Augmentation," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, {AISTATS} 2016, Cadiz, Spain, May 9-11, 2016*, 2015, pp. 342–350. arXiv: 1510.02795.

[75] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 15, IEEE, Jun. 2015, pp. 5353–5360, ISBN: 978-1-4673-6964-0. arXiv: arXiv:1412.1710v1.

[76] Y. Li, K. Swersky, and R. Zemel, "Generative Moment Matching Networks," in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, Eds., ser. Proceedings of Machine Learning Research, vol. 37, Lille, France: PMLR, 2015, pp. 1718–1727.

[77] P. Liu, P. Yang, K. Huang, and T. Tan, "Uniform low-rank representation for unsupervised visual domain adaptation," in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, Nov. 2015, pp. 216–220.

[78] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning Transferable Features with Deep Adaptation Networks," in *Proceedings of the 32nd International Conference on Machine Learning*, F. Bach and D. Blei, Eds., ser. Proceedings of Machine Learning Research, vol. 37, Lille, France: PMLR, 2015, pp. 97–105. arXiv: 1502.02791.

[79] M. Long, J. Wang, J. Sun, and P. S. Yu, "Domain invariant transfer kernel learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 6, pp. 1519–1532, 2015.

[80] V. Losing, B. Hammer, and H. Wersing, "Interactive online learning for obstacle classification on a mobile robot," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2015-Septe, no. 2, 2015.

[81] C. Salperwyck, M. Boullé, and V. Lemaire, "Concept drift detection using supervised bivariate grids," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2015-Septe, 2015.

[82] J. Climer and M. J. Mendenhall, "Dynamic Prototype Addition in Generalized Learning Vector Quantization," in *Advances in Intelligent Systems and Computing*, ser. Advances in Intelligent Systems and Computing, E. Merényi, M. J. Mendenhall, and P. O'Driscoll, Eds., vol. 428, Cham: Springer International Publishing, 2016, pp. 355–368, ISBN: 978-3-319-28517-7.

[83] M. Cordts, M. Omran, S. Ramos, *et al.*, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2016-Decem, IEEE, Jun. 2016, pp. 3213–3223, ISBN: 978-1-4673-8851-1. arXiv: 1604.01685.

[84] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, 2016, pp. 770–778, ISBN: 9781467388504.

[85] M. Long, J. Wang, Y. Cao, J. Sun, and P. S. Yu, "Deep learning of transferable representation for scalable domain adaptation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 8, pp. 2027–2040, 2016.

[86] M. Melucci, "Relevance Feedback Algorithms Inspired by Quantum Detection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 4, pp. 1022–1034, 2016. arXiv: 1611.03530.

[87] L. Mou, Z. Meng, R. Yan, *et al.*, "How Transferable are Neural Networks in NLP Applications?" In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, J. Su, X. Carreras, and K. Duh, Eds., The Association for Computational Linguistics, 2016, pp. 479–489, ISBN: 978-1-945626-25-8.

[88] A. Nemtsov, A. Averbuch, and A. Schclar, "Matrix compression using the Nyström method," *Intelligent Data Analysis*, vol. 20, no. 5, pp. 997–1019, May 2016. arXiv: 1305.0203.

[89] I. Redko and Y. Bennani, "Non-negative embedding for fully unsupervised domain adaptation," *Pattern Recognition Letters*, vol. 77, pp. 35–41, Jul. 2016, ISSN: 01678655.

[90] D. M. dos Reis, P. Flach, S. Matwin, and G. Batista, "Fast Unsupervised Online Drift Detection Using Incremental Kolmogorov-Smirnov Test," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, vol. 22, no. 1, pp. 1545–1554, 2016.

[91] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv*, 2016. arXiv: 1609.04747.

[92] B. Sun, J. Feng, and K. Saenko, "Return of Frustratingly Easy Domain Adaptation," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, {USA.}*, 2016, pp. 2058–2065.

[93] B. Sun and K. Saenko, "Deep CORAL: Correlation Alignment for Deep Domain Adaptation," in *Computer Vision – ECCV 2016 Workshops*, G. Hua and H. Jégou, Eds., Cham: Springer International Publishing, 2016, pp. 443–450, ISBN: 978-3-319-49409-8. arXiv: arXiv:1607.01719v1.

[94] R. Vidal, Y. Ma, and S. Sastry, *Generalized Principal Component Analysis*, ser. Interdisciplinary Applied Mathematics. New York, NY: Springer New York, 2016, vol. 40, ISBN: 978-0-387-87810-2.

[95] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big Data*, vol. 3, no. 1, p. 9, 2016.

[96] M. Arjovsky and L. Bottou, "Towards Principled Methods for Training Generative Adversarial Networks," in *5th International Conference on Learning Representations, {ICLR} 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[97] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein Generative Adversarial Networks," in *Proceedings of the 34th International Conference on Machine Learning*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of Machine Learning Research, vol. 70, International Convention Centre, Sydney, Australia: PMLR, 2017, pp. 214–223.

[98] P. P. Busto and J. Gall, "Open Set Domain Adaptation," in *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2017, pp. 754–763, ISBN: 978-1-5386-1032-9.

[99] N. Elhadji-Ille-Gado, E. Grall-Maes, and M. Kharouf, "Transfer Learning for Large Scale Data Using Subspace Alignment," in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, vol. 2018-Janua, IEEE, Dec. 2017, pp. 1006–1010, ISBN: 978-1-5386-1418-1.

[100] M. Ghifary, D. Balduzzi, W. B. Kleijn, and M. Zhang, "Scatter Component Analysis: A Unified Framework for Domain Adaptation and Domain Generalization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 7, pp. 1414–1430, Jul. 2017. arXiv: 1510.04373.

[101] H. M. Gomes, A. Bifet, J. Read, *et al.*, "Adaptive random forests for evolving data stream classification," *Machine Learning*, vol. 106, no. 9-10, pp. 1469–1495, 2017.

[102] B. Gong, K. Grauman, and F. Sha, "Geodesic Flow Kernel and Landmarks: Kernel Methods for Unsupervised Domain Adaptation," in *Advances in Computer Vision and Pattern Recognition*, 9783319583464, 2017, pp. 59–79, ISBN: 9783319583471.

[103] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep Transfer Learning with Joint Adaptation Networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17, JMLR.org, 2017, pp. 2208–2217.

[104] V. Losing, B. Hammer, and H. Wersing, "KNN classifier with self adjusting memory for heterogeneous concept drift," *Proceedings - IEEE International Conference on Data Mining, ICDM*, vol. 1, pp. 291–300, 2017.

[105] ——, "Self-adjusting memory: How to deal with diverse drift types," *IJCAI International Joint Conference on Artificial Intelligence*, pp. 4899–4903, 2017.

[106] S. Nikolaidis, S. Nath, A. D. Procaccia, and S. Srinivasa, "Game-Theoretic Modeling of Human Adaptation in Human-Robot Collaboration," in *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, New York, NY, USA: ACM, Mar. 2017, pp. 323–331, ISBN: 9781450343367.

[107] C. Prahm, B. Paassen, A. Schulz, B. Hammer, and O. Aszmann, "Transfer Learning for Rapid Re-calibration of a Myoelectric Prosthesis After Electrode Shift," in *Converging Clinical and Engineering Research on Neurorehabilitation II*, J. Ibáñez, J. González-Vargas, J. M. Azorín, M. Akay, and J. L. Pons, Eds., Cham: Springer International Publishing, 2017, pp. 153–157, ISBN: 978-3-319-46669-9.

[108] J. C. Velten, R. Arif, and D. Moehring, "Managing Disclosure through Social Media: How Snapchat is Shaking Boundaries of Privacy Perceptions," *The Journal of Social Media in Society*, vol. 6, no. 1, pp. 220–250, 2017.

[109] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, "Deep Hashing Network for Unsupervised Domain Adaptation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2017-Janua, IEEE, Jul. 2017, pp. 5385–5394, ISBN: 978-1-5386-0457-1. arXiv: 1706.07522.

[110] T. Villmann, A. Bohnsack, and M. Kaden, "Can learning vector quantization be an alternative to SVM and deep learning? - Recent trends and advanced variants of learning vector quantization for classification learning," *Journal of Artificial Intelligence and Soft Computing Research*, vol. 7, no. 1, pp. 65–81, 2017.

[111] W. Zellinger, T. Grubinger, E. Lughofer, T. Natschläger, and S. Saminger-Platz, "Central Moment Discrepancy (CMD) for Domain-Invariant Representation Learning," in *5th International Conference on Learning Representations, {ICLR} 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[112] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *5th International Conference on Learning Representations, {ICLR} 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[113] J. Zhang, W. Li, and P. Ogunbona, "Joint Geometrical and Statistical Alignment for Visual Domain Adaptation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jul. 2017, pp. 5150–5158, ISBN: 978-1-5386-0457-1.

[114] Z. Cao, L. Ma, M. Long, and J. Wang, "Partial Adversarial Domain Adaptation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11212 LNCS, Springer International Publishing, Aug. 2018, pp. 139–155, ISBN: 9783030012366. arXiv: 1808.04205.

[115] Q. Chen, Y. Liu, Z. Wang, I. Wassell, and K. Chetty, "Re-Weighted Adversarial Adaptation Network for Unsupervised Domain Adaptation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[116] J. Hoffman, E. Tzeng, T. Park, *et al.*, "CyCADA: Cycle-Consistent Adversarial Domain Adaptation," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, Stockholmsmässan, Stockholm Sweden: PMLR, 2018, pp. 1989–1998.

[117] M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Conditional Adversarial Domain Adaptation," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, 2018, pp. 1647–1657.

[118] ——, "Conditional Adversarial Domain Adaptation," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, 2018, pp. 1647–1657.

[119] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral Normalization for Generative Adversarial Networks," in *International Conference on Learning Representations*, 2018.

[120] J. Montiel, J. Read, A. Bifet, and T. Abdessalem, "Scikit-Multiflow: A Multi-output Streaming Framework," *Journal of Machine Learning Research*, vol. 19, no. 72, pp. 1–5, 2018. arXiv: 1807.04662.

[121] X. Peng, B. Usman, N. Kaushik, D. Wang, J. Hoffman, and K. Saenko, "VisDA: A Synthetic-to-Real Benchmark for Visual Domain Adaptation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, vol. 2018-June, IEEE, Jun. 2018, pp. 2134–2139, ISBN: 978-1-5386-6100-0.

[122] C. Raab and F.-M. Schleif, "Sparse Transfer Classification for Text Documents," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, F. Trollmann and A.-Y. Turhan, Eds., vol. 11117 LNAI, Springer International Publishing, 2018, pp. 169–181.

[123] ——, "Transfer learning for the probabilistic classification vector machine," in *Proceedings of the Seventh Workshop on Conformal and Probabilistic Prediction and Applications*, A. Gammerman, V. Vovk, Z. Luo, E. Smirnov, and R. Peeters, Eds., ser. Proceedings of Machine Learning Research, vol. 91, PMLR, 2018, pp. 187–200.

[124] F.-M. Schleif, C. Raab, and P. Tino, "Sparsification of Indefinite Learning Models," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11004 LNCS, 2018, pp. 173–183, ISBN: 9783319977843.

[125] F.-M. M. Schleif, A. Gisbrecht, P. Tiňo, and P. Tino, "Supervised low rank indefinite kernel approximation using minimum enclosing balls," *Neurocomputing*, vol. 318, pp. 213–226, 2018.

[126] J. Shao, F. Huang, Q. Yang, and G. Luo, "Robust Prototype-Based Learning on Data Streams," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 5, pp. 978–991, 2018.

[127] J. Shen, Y. Qu, W. Zhang, and Y. Yu, "Wasserstein Distance Guided Representation Learning for Domain Adaptation," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New*, S. A. McIlraith and K. Q. Weinberger, Eds., AAAI Press, 2018, pp. 4058–4065.

[128] H. J. Song and S. B. Park, "An adapted surrogate kernel for classification under covariate shift," *Applied Soft Computing Journal*, vol. 69, pp. 435–442, 2018.

[129] M. Straat, F. Abadi, C. Göpfert, B. Hammer, and M. Biehl, "Statistical Mechanics of On-Line Learning Under Concept Drift," *Entropy*, vol. 20, no. 10, p. 775, Oct. 2018.

[130] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A Survey on Deep Transfer Learning," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, V. Krurková, Y. Manolopoulos, B. Hammer, L. Iliadis, and I. Maglogiannis, Eds., vol. 11141 LNCS, Cham: Springer International Publishing, 2018, pp. 270–279, ISBN: 9783030014230.

[131] R. Volpi, P. Morerio, S. Savarese, and V. Murino, "Adversarial Feature Augmentation for Unsupervised Domain Adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2018, pp. 5495–5504, ISBN: 978-1-5386-6420-9. arXiv: 1711.08561.

[132] R. Volpi, H. Namkoong, O. Sener, J. C. Duchi, V. Murino, and S. Savarese, "Generalizing to Unseen Domains via Adversarial Data Augmentation," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., Curran Associates, Inc., 2018, pp. 5334–5344.

[133] J. Wang, W. Feng, Y. Chen, H. Yu, M. Huang, and P. S. Yu, "Visual Domain Adaptation with Manifold Embedded Distribution Alignment," in *2018 ACM Multimedia Conference on Multimedia Conference - MM '18*, ser. MM '18, New York, New York, USA: ACM Press, 2018, pp. 402–410, ISBN: 9781450356657.

[134] ——, "Visual Domain Adaptation with Manifold Embedded Distribution Alignment," in *2018 ACM Multimedia Conference on Multimedia Conference - MM '18*, ser. MM '18, New York, New York, USA: ACM Press, 2018, pp. 402–410, ISBN: 9781450356657.

[135] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, Oct. 2018. arXiv: 1911.02685.

[136] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent Trends in Deep Learning Based Natural Language Processing [Review Article]," *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55–75, Aug. 2018.

[137] S. Zhou, E. Smirnov, G. Schoenmakers, R. Peeters, and T. Jiang, "Conformal feature-selection wrappers for instance transfer," in *Proceedings of the Seventh Workshop on Conformal and Probabilistic Prediction and Applications*, A. Gammerman, V. Vovk, Z. Luo, E. Smirnov, and R. Peeters, Eds., ser. Proceedings of Machine Learning Research, vol. 91, PMLR, 2018, pp. 96–113.

[138] Y. Balaji, R. Chellappa, and S. Feizi, "Normalized Wasserstein for Mixture Distributions With Applications in Adversarial Learning and Domain Adaptation," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2019, pp. 6499–6507, ISBN: 978-1-7281-4803-8.

[139] M. Carnein and H. Trautmann, "Optimizing Data Stream Representation: An Extensive Survey on Stream Clustering Algorithms," *Business & Information Systems Engineering*, vol. 61, no. 3, pp. 277–297, Jun. 2019.

[140] W.-G. Chang, T. You, S. Seo, S. Kwak, and B. Han, "Domain-Specific Batch Normalization for Unsupervised Domain Adaptation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2019-June, IEEE, Jun. 2019, pp. 7346–7354, ISBN: 978-1-7281-3293-8. arXiv: 1906.03950.

[141] X. Chen, S. Wang, B. Fu, M. Long, and J. Wang, "Catastrophic Forgetting Meets Negative Transfer: Batch Spectral Shrinkage for Safe Transfer Learning," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 1908–1918.

[142] X. Chen, S. Wang, M. Long, and J. Wang, "Transferability vs. Discriminability: Batch Spectral Penalization for Adversarial Domain Adaptation," in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, Long Beach, California, USA: PMLR, 2019, pp. 1081–1090.

[143] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres, "Quantifying the Carbon Emissions of Machine Learning," Oct. 2019, ISSN: 15577317. arXiv: 1910.09700.

[144] J. Lee, P. Sattigeri, and G. Wornell, "Learning New Tricks From Old Dogs: Multi-Source Transfer Learning From Pre-Trained Networks," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 4372–4382.

[145] J. Li, M. Jing, K. Lu, L. Zhu, and H. T. Shen, "Locality Preserving Joint Transfer for Domain Adaptation," *IEEE Transactions on Image Processing*, vol. 28, no. 12, pp. 6103–6115, Dec. 2019. arXiv: 1906.07441.

[146] S. Li, C. H. Liu, B. Xie, L. Su, Z. Ding, and G. Huang, "Joint Adversarial Domain Adaptation," in *Proceedings of the 27th ACM International Conference on Multimedia*, New York, NY, USA: ACM, Oct. 2019, pp. 729–737, ISBN: 9781450368896.

[147] H. Liu, M. Long, J. Wang, and M. Jordan, "Transferable Adversarial Training: A General Approach to Adapting Deep Classifiers," in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, Long Beach, California, USA: PMLR, 2019, pp. 4013–4022.

[148] S. Mahadevan, B. Mishra, and S. Ghosh, "A Unified Framework for Domain Adaptation Using Metric Learning on Manifolds," in *Machine Learning and Knowledge Discovery in Databases*, M. Berlingerio, F. Bonchi, T. Gärtner, N. Hurley, and G. Ifrim, Eds., Cham: Springer International Publishing, 2019, pp. 1–17, ISBN: 978-3-030-10928-8. arXiv: arXiv:1804.10834v1.

[149] J. Montiel, A. Bifet, V. Losing, J. Read, and T. Abdessalem, "Learning Fast and Slow: A Unified Batch/Stream Framework," *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018*, pp. 1065–1072, 2019.

[150] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment Matching for Multi-Source Domain Adaptation," in *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[151] C. Raab, M. Heusinger, and F. M. Schleif, "Reactive Soft Prototype Computing for frequent reoccurring Concept Drift," in *27th European Symposium on Artificial Neural Networks, ESANN 2019, Bruges, Belgium, April 24-26, 2019*, 2019, pp. 437–442.

[152] S. Rakshit, U. Chaudhuri, B. Banerjee, and S. Chaudhuri, "Class Consistency Driven Unsupervised Deep Adversarial Domain Adaptation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, IEEE, Jun. 2019, pp. 657–666, ISBN: 978-1-7281-2506-0.

[153] I. Redko, E. Morvant, A. Habrard, M. Sebban, and Y. Bennani, *Advances in Domain Adaption Theory*. Elsevier, Aug. 2019, ISBN: 9781785482366.

[154] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, Dec. 2019.

[155] B. Wang, J. Mendez, M. Cai, and E. Eaton, "Transfer Learning via Minimizing the Performance Gap Between Domains," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 10 644–10 654.

[156] H. Wang, W. Yang, Z. Lin, and Y. Yu, "TMDA: Task-Specific Multi-source Domain Adaptation via Clustering Embedded Adversarial Training," in *2019 IEEE International Conference on Data Mining (ICDM)*, vol. 2019-Novem, IEEE, Nov. 2019, pp. 1372–1377, ISBN: 978-1-7281-4604-1.

[157] J. Wang, Y. Chen, H. Yu, M. Huang, and Q. Yang, "Easy Transfer Learning By Exploiting Intra-Domain Structures," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, Jul. 2019, pp. 1210–1215, ISBN: 978-1-5386-9552-4.

[158] X. Wang, Y. Jin, M. Long, J. Wang, and M. I. Jordan, "Transferable Normalization: Towards Improving Transferability of Deep Neural Networks," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 1951–1961.

[159] R. Xu, G. Li, J. Yang, and L. Lin, "Larger Norm More Transferable: An Adaptive Feature Norm Approach for Unsupervised Domain Adaptation," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, vol. 2019-Octob, IEEE, Oct. 2019, pp. 1426–1435, ISBN: 978-1-7281-4803-8.

[160] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated Machine Learning," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1–19, Feb. 2019. arXiv: 1902.04885.

[161] K. You, M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Universal Domain Adaptation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[162] H. Zhao, R. T. D. Combes, K. Zhang, and G. Gordon, "On Learning Invariant Representations for Domain Adaptation," in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, Long Beach, California, USA: PMLR, 2019, pp. 7523–7532.

[163] M. Biehl, F. Abadi, C. Göpfert, and B. Hammer, "Prototype-Based Classifiers in the Presence of Concept Drift: A Modelling Framework," in *Advances in Intelligent Systems and Computing*, vol. 976, 2020, pp. 210–221, ISBN: 9783030196417.

[164] W. Deng, L. Zheng, Y. Sun, and J. Jiao, "Rethinking Triplet Loss for Domain Adaptation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 8, pp. 1–1, 2020.

[165] D. Ghosal, D. Hazarika, A. Roy, N. Majumder, R. Mihalcea, and S. Poria, "KinGDOM: Knowledge-Guided DOMain Adaptation for Sentiment Analysis," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2020, pp. 3198–3210. arXiv: 2005.00791.

[166] Ö. Gözüaçık and F. Can, "Concept learning using one-class classifiers for implicit drift detection in evolving data streams," *Artificial Intelligence Review*, no. 0123456789, Nov. 2020.

[167] M. Heusinger, C. Raab, and F.-M. Schleif, "Analyzing Dynamic Social Media Data via Random Projection - A New Challenge for Stream Classifiers," in *2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, IEEE, May 2020, pp. 1–8.

[168] ——, "Passive Concept Drift Handling via Momentum Based Robust Soft Learning Vector Quantization," in *Advances in Intelligent Systems and Computing*, A. Vellido, K. Gibert, C. Angulo, and J. D. Martín Guerrero, Eds., vol. 976, Cham: Springer International Publishing, 2020, pp. 200–209.

[169] ——, "Passive concept drift handling via variations of learning vector quantization," *Neural Computing and Applications*, vol. 6, Aug. 2020.

[170] M. Heusinger and F.-M. Schleif, "Reactive Concept Drift Detection Using Coresets Over Sliding Windows," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, Dec. 2020, pp. 1350–1355, ISBN: 978-1-7281-2547-3.

[171] F. Hinder, A. Artelt, and B. Hammer, "Towards non-parametric drift detection via Dynamic Adapting Window Independence Drift Detection (DAWIDD)," in *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[172] A. Kumar, T. Ma, and P. Liang, "Understanding Self-Training for Gradual Domain Adaptation," in *Proceedings of the 37th International Conference on Machine Learning*, Feb. 2020. arXiv: 2002.11361.

[173] I. H. Laradji and R. Babanezhad, "M-ADDA: Unsupervised Domain Adaptation with Deep Metric Learning," in *Domain Adaptation for Visual Understanding*, R. Singh, M. Vatsa, V. M. Patel, and N. Ratha, Eds., Cham: Springer International Publishing, 2020, pp. 17–31, ISBN: 978-3-030-30671-7.

[174] M. Li, Y.-M. Zhai, Y.-W. Luo, P.-F. Ge, and C.-X. Ren, "Enhanced Transport Distance for Unsupervised Domain Adaptation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 13 936–13 944.

[175] P. Li, Z. Ni, X. Zhu, J. Song, and W. Wu, "Optimal Transport with Dimensionality Reduction for Domain Adaptation," *Symmetry*, vol. 12, no. 12, p. 1994, Dec. 2020.

[176] J. Liang, D. Hu, and J. Feng, "Do We Really Need to Access the Source Data? Source Hypothesis Transfer for Unsupervised Domain Adaptation," *Proceedings of the 37th International Conference on Machine Learning*, Feb. 2020. arXiv: 2002.08546.

[177] Z. Lu, Y. Yang, X. Zhu, C. Liu, Y.-Z. Song, and T. Xiang, "Stochastic Classifiers for Unsupervised Domain Adaptation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[178] M. Münch, C. Raab, M. Biehl, and F.-M. Schleif, "Structure Preserving Encoding of Non-euclidean Similarity Data," in *Proceedings of the 9th International Conference on Pattern Recognition Applications and Methods*, SCITEPRESS - Science and Technology Publications, 2020, pp. 43–51.

[179] M. Münch, C. Raab, M. Biehl, and F.-m. Schleif, "Data-Driven Supervised Learning for Life Science Data," *Frontiers in Applied Mathematics and Statistics*, vol. 6, no. November, pp. 1–15, Nov. 2020.

[180] S. Noori Saray and J. Tahmoresnezhad, "Joint distinct subspace learning and unsupervised transfer classification for visual domain adaptation," *Signal, Image and Video Processing*, no. iii, Jul. 2020.

[181] X. Peng, Z. Huang, Y. Zhu, and K. Saenko, "Federated Adversarial Domain Adaptation," in *International Conference on Learning Representations*, 2020.

[182] C. Raab, M. Heusinger, and F.-M. Schleif, "Reactive Soft Prototype Computing for Concept Drift Streams," *Neurocomputing*, vol. 416, pp. 340–351, Nov. 2020.

[183] C. Raab, P. Meier, and F.-M. Schleif, "Domain Invariant Representations with Deep Spectral Alignment," in *28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2020, pp. 509–514.

[184] C. Raab and F.-M. Schleif, "Low-Rank Subspace Override for Unsupervised Domain Adaptation," in *KI 2020: Advances in Artificial Intelligence*, U. Schmid, F. Klügl, and D. Wolter, Eds., Cham: Springer International Publishing, 2020, pp. 132–147, **Best Paper Award**.

[185] ——, "Transfer learning extensions for the probabilistic classification vector machine," *Neurocomputing*, vol. 397, pp. 320–330, Jul. 2020.

[186] R. K. Sanodiya, A. Mathew, J. Mathew, and M. Khushi, "Statistical and Geometrical Alignment using Metric Learning in Domain Adaptation," in *2020 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Jul. 2020, pp. 1–8, ISBN: 978-1-7281-6926-2. arXiv: 2006.09882.

[187] F.-M. Schleif, C. Raab, and P. Tino, "Sparsification of core set models in non-metric supervised learning," *Pattern Recognition Letters*, vol. 129, pp. 1–7, Jan. 2020.

[188] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green AI," *Communications of the ACM*, vol. 63, no. 12, pp. 54–63, 2020, ISSN: 15577317. arXiv: 1907.10597.

[189] R. Tachet, H. Zhao, Y.-X. Wang, and G. Gordon, "Domain Adaptation with Conditional Distribution Matching and Generalized Label Shift," *arXiv*, no. NeurIPS, Mar. 2020. arXiv: 2003.04475.

[190] L. Tian, Y. Tang, L. Hu, Z. Ren, and W. Zhang, "Domain Adaptation by Class Centroid Matching and Local Manifold Self-Learning," *IEEE Transactions on Image Processing*, vol. 29, pp. 9703–9718, Mar. 2020. arXiv: 2003.09391.

[191] S. Wang and L. Zhang, "Self-adaptive Re-weighted Adversarial Domain Adaptation," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, California: International Joint Conferences on Artificial Intelligence Organization, Jul. 2020, pp. 3181–3187, ISBN: 978-0-9992411-6-5. arXiv: 2006.00223.

[192] L. Yang, M. Men, Y. Xue, and P. Zhong, "Low-rank representation-based regularized subspace learning method for unsupervised domain adaptation," *Multimedia Tools and Applications*, vol. 79, no. 3-4, pp. 3031–3047, Jan. 2020.

[193] L. Zhao and Y. Liu, "Spectral Normalization for Domain Adaptation," *Information*, vol. 11, no. 2, Jan. 2020.

[194] S. Zhao, M. Gong, T. Liu, H. Fu, and D. Tao, "Domain Generalization via Entropy Regularization," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 16 096–16 107.

[195] Y. Zuo, H. Yao, and C. Xu, "Category-Level Adversarial Self-Ensembling for Domain Adaptation," in *2020 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, Jul. 2020, pp. 1–6, ISBN: 978-1-7281-1331-9.

[196] I. Achituve, H. Maron, and G. Chechik, "Self-Supervised Learning for Domain Adaptation on Point Clouds," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021, pp. 123–133.

[197] N. Agarwal, A. Sondhi, K. Chopra, and G. Singh, "Transfer Learning: Survey and Classification," in *Advances in Intelligent Systems and Computing*, Ml, vol. 1168, Springer Singapore, 2021, pp. 145–155, ISBN: 9789811553448.

[198] C. Raab, P. Väth, P. Meier, and F.-M. Schleif, "Bridging Adversarial and Statistical Domain Transfer via Spectral Adaptation Networks," in *Proceedings of the Asian Conference on Computer Vision (ACCV)*, Nov. 2021, pp. 457–473.

[199] H.-W. Yeh, B. Yang, P. C. Yuen, and T. Harada, "SoFA: Source-Data-Free Feature Alignment for Unsupervised Domain Adaptation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021, pp. 474–483.

[200] A. Zell, R. Hübner, T. Korb, *et al.*, "Snns: An efficient simulator for neural nets," in *MASCOTS '93: Proceedings of the International Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems*, pp. 343–346, ISBN: 1-56555-018-8.

ACRONYMS

ADWIN Adaptive Windowing. 99, 100, 111, 113, 114, 117–119

ASAN Adversarial Spectral Adaptation Networks. 70, 76–78, 80–94, 121

BSP Batch Spectral Penalization. 70, 85, 87, 88

CDAN Conditional Domain Adversarial Network. 70, 76, 77, 80–82, 84, 85, 87–93

CGCA Cascaded Geometric Covariance Alignment. 55, 56, 62

CNN Convolutional Neural Network. 76

cNSO Class-aware Nyström Subspace Override. 48–53, 55–62, 121

CORAL Correlation Alginment. 55, 56, 62, 63, 68, 69, 85, 86

COV Forest Cover Type Dataset. 111, 114–116

DA Domain Adaptation. 13, 35, 55, 56, 62

DAN Domain Adaptation Network. 55, 56, 85, 86

DANN Domain Adversarial Network. 66, 67, 70, 76, 81–85, 87–91

DDC Deep Domain Confusion. 55, 56, 85, 86

DDM Dirft-Detection Method. 99, 111, 113, 114

DSN Domain Spectral Network. 70, 71, 75, 80, 82–86

EasyTL Easy Transfer Learning. 55, 56

EDDM Early Dirft-Detection Method. 99, 111, 113, 114

ELEC Electricity Dataset. 111, 115, 116

ETN Enhanced Transport Distance. 85, 87, 88

EVD Eigen Value Decomposition. 9, 11, 12

GAN Generative Adversarial Network. 66, 80

GDA Geometric Domain Adaptation. 26, 27, 29–31, 33–36, 53, 55, 57, 58, 60, 62, 121

GDPR General Data Protection Regularization. 123

GFK Geodesic Flow Kernel. 14, 33, 35, 55, 56

GLVQ Generalized Learning Vector Quantization. 22

GMSC Give Me Some Credit Generator. 111, 115, 116

GRL Gradient Reversal Layer. 69, 70, 77

HAT Hoeffding Adaptive Tree. 111, 114–116, 118

JAN Joint Adaptation Network. 55, 56, 85–88

JDA Joint Distribution Adaptation. 14, 33, 35, 55, 56, 62

JGSA Joint Geometrical and Statistical Alignment. 14, 55–57, 62

KNN k-Nearest Neighbours. 99, 111

KS Kolmogorov Sminroff. 97, 99, 102, 103, 107

KSWIN Kolmogorov Sminroff Sliding Window. 100, 103, 104, 110, 111, 113, 114, 117, 119

LS Least-Squares. 14, 50, 57

LSA Latent Semantic Analysis. 29, 30, 43

LSSA Landmarks Selection-based Subspace Alignment. 14

LTM Long Term Memory. 99

LVQ Learning Vector Quantization. 4, 21, 100, 104

MEDA Manifold Embedding Distribution Alignment. 14, 55–57, 62

MMD Maximum Mean Discrepancy. 2, 13, 14, 20, 51, 52, 56, 68, 72

NN Neural Network. 82–84

NSO Nyström Subspace Override. 46–53, 55–60, 62, 63, 121

OC Office-Caltech. 55, 57, 60, 61

OZA Oza Bagging Ensemble Classifier.. 114–116, 118

PCA Principal Component Analysis. 2, 41, 42, 48

PSD Positive Semi-Definite. 9, 10, 69

RAD Radian. 90, 91

RBF Radial Basis Function. 33, 42, 55, 68, 111, 114–116

RKHS Reproducing Kernel Hilbert Space. 14, 20, 68

RMS Root Mean Square. 106

RRSLVQ Reactive Robust Soft learning Vector Quantization. 100, 107, 108, 111, 114–119, 121, 122

RSL Relevance Spectral Loss. 76, 78–80, 82, 84–89, 91, 92, 121

RSLVQ Robust Soft learning Vector Quantization. 4, 21–23, 95, 97, 100, 101, 103–107, 110, 111, 114–116, 118, 119, 121, 122

RTG Random Tree Generator. 111, 114–116

SA Subspace Alignment. 14, 33, 55–57, 60, 62, 63

SAFN Stepwise Adaptive Feature Norm. 85, 88

SAMKNN Self-Adjusting Memory-KNN. 99, 111, 114–118

SCA Scatter Component Analysis. 55, 56, 62

SDAN Spectral Normalized Conditional Domain Adversarial Network. 70, 85, 87, 88

SEA Streaming Ensemble Algorithm. 111, 114–116

SGD Stochastic Gradient Descent. 23, 70, 72, 85, 97, 104–107, 118

SIFT Scale Invariant Feature Transform. 54

SL Spectral Loss. 67, 69, 70, 72, 75, 76, 82, 84, 86, 121

SN Spectral Normalization. 70, 76, 77, 80, 81, 85, 87, 92

SO Subspace Override. 3, 40, 42, 43, 46, 48, 50–53, 55, 57–60, 62, 121

SQRE Moving Squares Dataset. 111, 114–116

STM Short Term Memory. 99

SURF Speeded Up Robust Feature. 53–56

SVD Singular Value Decomposition. 7, 8, 12, 14, 27, 30, 42, 43, 46–48, 60, 62, 71, 81

SVM Support Vector Machine. 33–36, 55, 56, 58–60, 97

TCA Transfer Component Analysis. 13, 33, 35, 55, 56, 62, 63, 76

TKL Transfer Kernel Learning. 33, 35, 36

TSNE T-Distributed Stochastic Neighbor Embedding. 35, 36, 58, 59, 82–84, 89, 90, 152

VC Vapnik–Chervonenkis. 19–21

WTA Winner Takes All. 107

WTHR Weather Dataset. 111, 115, 116

APPENDIX: ADDITIONAL RESULTS OF ADVERSARIAL SPECTRAL ADAPTATION NETWORKS

This appendix contains additional results of chapter 5 not included in the main document for clarity. This appendix demonstrates that the proposed attributes of Adversarial Spectral Adaptation Networks hold for multiple datasets. Further, some examples of the Domain Adaptation datasets are shown. The contents of this part of the appendix are: (**i**) Additional performance results on the VisDa [121] dataset. (**ii**) Additional results of the convergence and spectral analysis. (**iii**) Additional results of the feature analysis. (**iv**) The last section shows example images taken from the benchmark datasets used in the experiments.

Note that the techniques used for the presented results are the same as the techniques used to obtain the main results of this thesis. However, they are applied to different datasets to emphasize the usefulness and efficiency of the approach beyond the experiments shown in the main document.

ADDITIONAL EXPERIMENTAL RESULTS ON VISDA

This section introduces the VisDa dataset [121] and discusses the performance results of ASAN compared to other networks.

The VisDa [121] dataset is one of the largest Domain Adaptation benchmark datasets with over 280,000 images divided into the three subcategories training, validation, and testing. The task is to train on 152,397 synthetic images, such as concepts of planes or cars, and validate the obtained network on 55,388 real images. Both training and validation datasets consist of twelve classes with a minimum of approximately 6,000 examples and a maximum of approximately 16,000 class examples in the training domain. Some selected sample images are shown in Tab. 5.

The shown results are obtained using the same experimental setup as in the main

Method (ResNet-50)	Dataset Synthetic → Real	Method (ResNet-101)	Dataset Synthetic → Real
CDAN [141]	70.0	CDAN [141]	73.7
TAT [147]	71.9	BSP [142]	75.9
TransNorm [158]	71.6	SAFN [159]	**76.1**
ASAN	**74.0**	ASAN	74.7

Table 1: **Mean** prediction **accuracy** with standard deviation on the **VisDa** dataset over three random runs. Results separated by baseline network ResNet-50 and ResNet-101 .

part (see Sec. 5.4 in the main part for details) and shown in Tab. 1. While using the ResNet-50 baseline, we can obtain very competitive results and a performance improvement to the CDAN-baseline.

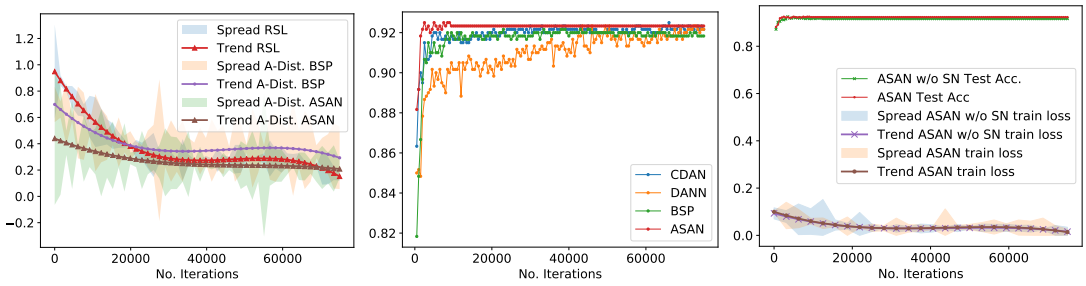
However, using ResNet-101, we are not able to report such a performance improvement. There are two main reasons for this: first, the shrinkage parameter k is optimized using real images and discards domain-specific information given on rich, detailed images. However, this richness is not present in synthetic images and leads to a drastic reduction of domain-specific information. Second, the parameter is optimized on the ResNet-50 bottleneck space while used in ResNet-101 with a different bottleneck space composition. Therefore, the occurrence of *domain-specific* in the source spectrum is different, which leads to a wrong application of the shrinkage mechanism. In future work, the k parameter will be optimized on the ResNet-101 network.

ADDITIONAL RESULTS OF CONVERGENCE AND SPECTRAL ANALYSIS

This section presents an additional analysis of convergence and spectral properties of the ASAN network against related methods. The data is obtained by learning and evaluating on $\mathbf{P}\rightarrow\mathbf{I}$ from the Image-Clef dataset.

The plot in Fig. 1a shows the performance of the proposed Relevance Spectral Loss (RSL) against BSP [142]. As in the main part, we rely on the $\mathcal{A}$-Distance [14], [24] for the quality analysis of the invariant representation. The $\mathcal{A}$-Distance is defined as $\mathcal{A} = 2(2 - 1\hat{\varepsilon})$, where $\hat{\varepsilon}$ is the error of the trained domain classifier. The results confirm the results presented in the main proposal: the $\mathcal{A}$-Distance trend of ASAN (brown) is overall lower in value than the trend of BSP [142] (purple), allowing the statement that ASAN learns a better invariant representation faster than BSP [142]. The fluctuation around both trend lines, green for ASAN and orange for BSP [142], confirms this statement by having a spread generally oriented to lower $\mathcal{A}$-Distances values compared to BSP [142]. Additionally, the RSL is effectively reduced by ASAN during learning. This confirms that the similarity of spectra of both domains in the bottleneck space $\mathcal{F}$ is aligned, which is why ASAN offers a better invariant representation. Note that by the definition of the $\mathcal{A}$-Distance, negative values are possible [14], [24], indicating that the domain classifier has a lower performance than random guessing favorable for Domain Adaptation because the source and target domain data are completely indistinguishable for the domain classifier.

The stability properties of the proposed ASAN in the main part are also confirmed given different datasets. The plot is presented in Fig. 1b and shows that ASAN has the best overall accuracy. Further, the plots show that ASANs crucial advantage, besides the performance, is the convergence in an optimum, providing higher accuracy than others. Once achieved, the optimum does not change afterward, while BSP [142], CDAN [118], and DANN [72] are fluctuating during the continuous learning process and, in the worse case, descend in performance.



(a) A-Distance & RSL *(b)* Target Acc. *(c)* Source loss & Test Acc.

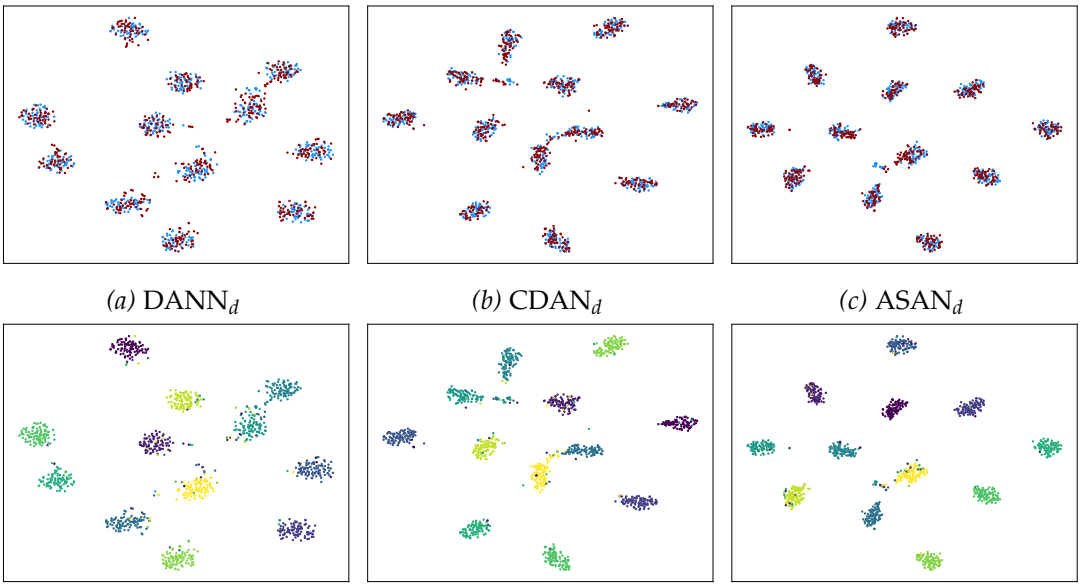
Figure 1: Learning process of our ASAN compared to related networks over time given **P**→**I** images from Image-Clef dataset. Best viewed on computer display.

The plot in Fig. 1c shows that our proposed RSL adaptation-loss is substantially responsible for achieving high accuracy. Simultaneously, Spectral Normalization [119] supports the adaptation process via stability during training as the ASAN and ASAN w/o (without Spectral Normalization [119]) showing overall similar learning. The ASAN w/o achieves similar accuracy from an early stopping perspective. However, the Spectral Normalization [119] process within ASAN stabilizes the network at a high optimum. The combination is leading to the overall best prediction performance. The latter can again be confirmed by comparing the accuracy of Fig. 1b and Fig. 1c and in the experiments section in chapter 5, where ASAN performs considerably better than SDAN [193]. SDAN consists of the same base-network and Spectral Normalization [119] without further spectral alignments necessary for rich Domain Adaptation, as presented with our ASAN model.

ADDITIONAL RESULTS OF FEATURE ANALYSIS

This section presents an additional analysis of feature properties of the ASAN network against related methods. The data is obtained by learning and evaluating on **P**→**I** from the Image-Clef dataset.

The result is presented in Fig. 2 and is split into two parts: the top row (Fig. 2a - 2c) is a scatter plot of the bottleneck features of trained DANN [72], CDAN [118], and our ASAN colored with ground truth domain labels. Blue shows the source, and red shows the target domain. ASAN shows the superiority of creating a domain invariant representation by learning more compact clusters and observable separates the obtained clusters better compared to DANN [72] and CDAN [118]. The bottom row (Fig. 2d - 2f) shows the same representation but with classification labels. Due to dense and separated clusters, the obtained representation is easily classified by a neural network. However, again, the main proposal limitation also takes part in the plot but is the same for DANN [72] and CDAN [118]. The limitation is the lack of approximating the joint distribution of labels and features, leading to some samples being assigned to the



(a) DANN$_d$ *(b)* CDAN$_d$ *(c)* ASAN$_d$

(d) DANN$_c$ *(e)* CDAN$_c$ *(f)* ASAN$_c$

Figure 2: TSNE [29] of bottleneck features of selected networks given **P**→**I** images from the Image-Clef dataset. <Name>$_d$ and <Name>$_c$ show the outputs with ground truth domain and classification labels respectively. For the first row, blue shows the source, and red shows the target domain. Best viewed on computer display.

wrong cluster. See Sec. 5.3 for a discussion. Nevertheless, this effect is much weaker at ASAN than at CDAN [118] and DANN [72].

EXAMPLES FROM BENCHMARK DATASETS

To provide an overview of the dataset characteristics and their differences among each other, we present some selected examples of the datasets Office-31 [40] in Tab. 2, Image-Clef in Tab. 3, Office-Home [109] in Tab. 4 and VisDa [121] in Tab. 5. With this, the focus of the sampling process lies in showing the difficulties between the dataset domains.

Intuitively, it is easy to see that the Office-31 [40] dataset is the easiest among all four datasets. This is also confirmed in the experiments section of the main proposal. While Office-31 [40] contains objects from different angles and light settings, it is still observable that these images belong to the same class. The Image-Clef dataset is most difficult. However, the algorithms still show a consistently good performance. We assume that the main reason for the good performance is that the photographs are still made in some sense in the real world. However, when it comes to Office-Home [109], we observe that some domains like Art show drastic changes in the appearances (style and presentation) of the images to the domain Real-World. Also, in comparison to Product,

Table 2: Example images from the Office-31 [40] datasets. The objects shown capture the domain shift via difficult examples. Row caption is class name and column caption is domain name.

Table 3: Example images from the Image-Clef datasets. The objects shown capture the domain shift via difficult examples. Row caption is class name and column caption is domain name.

there are partially some broad domain shifts, which is the overall reason why all tested algorithms have worse performance on Office-Home [109] in comparison to Office-31 [40] and Image-Clef. On the other hand, the VisDa [121] dataset creates an entirely different scenario because the objects in the train images are synthetic. In contrast, the validation dataset contains real images with partly non-trivial representations of an object. However, due to the small class size and large sample size, the networks are usually better on VisDa [121] as on Office-Home [109].

Table 4: Example images from the Office-Home [109] datasets. The objects shown capture the domain shift via difficult examples. Row caption is class name and column caption is domain name.

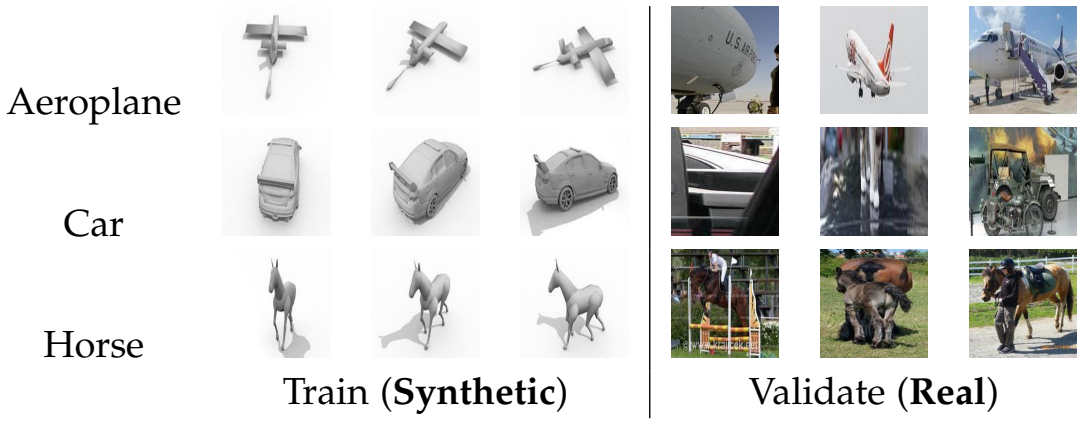


Table 5: Example images from the VisDa [121] datasets. The objects shown capture the domain shift via difficult examples. Row caption is class name and column caption is domain name.

APPENDIX: STREAMING DATASET DESCRIPTIONS

In this appendix, the data streams are described. It is split up into synthetic data streams and real-world streams. The synthetic stream generators are potentially infinite, and drift can be implemented by changing the generating function. A common approach to create drift is to invert the function.

SYNTHETIC STREAM GENERATORS

LED. The LED data set simulates both abrupt and gradual drifts based on the used generator. The generator was first introduced in [2]. This data set yields instances with 24 Boolean features, 17 of which are irrelevant. The remaining seven features correspond to each segment of a seven-segment LED display. The goal is to predict the digit displayed on the LED display, where each feature has a ten % chance of being inverted. Drifts occur in the data set by swapping relevant with irrelevant features. The first drift swaps three features and is continuously replaced with an abrupt drift that swaps seven features. LED_g simulates one gradual drift, while LED_a simulates three abrupt drifts.

SEA. The SEA generator is generating a data stream with abrupt Concept Drift, first described by Street and Kim in [15]. It produces data streams with three continuous attributes (f_1, f_2, f_3). The range of values that each attribute can assume lies between zero and ten. Only the first two attributes (f_1, f_2) are relevant, i. e., f_3 does not influence the class value determination. New instances are obtained by randomly setting a point in a two-dimensional space, such that these dimensions correspond to f_1 and f_2. This two-dimensional space is split into four blocks, each corresponding to one of four different functions. In each block, a point belongs to class 1 if $f_1 + f_2 \leq \theta$ and to class 0 otherwise. The threshold θ is used to split instances between classes 0 and 1, assuming the values $\theta = 8$ (block 1), $\theta = 9$ (block 2), $\theta = 7$ (block 3), and $\theta = 9.5$ (block 4). Two important features are the possibility to balance classes, which means the class distribution will tend to a Uniform one, and the possibility to add noise, which will, according to some probability, change the chosen label for a data instance. In this experiment, the SEA generator is used with ten percent noise in the data stream. SEA_g simulates one gradual drift, while SEA_a simulates an abrupt drift.

MIXED The MIXED Generator creates a binary classification stream. The stream consists of four features, two Boolean attributes v, w and two numeric attributes x and y between $[0; 1]$. The label is positive if two out of three conditions are satisfied

$$v = true, w = true, y < 0.5 + 0.3 \sin(3\pi x). \tag{1}$$

After each Concept Drift, the classification is reversed [13].

RTG. The Random Tree Generator (RTG) is based on a random tree that splits features at random and sets labels to its leafs [7]. After the tree is built, new instances are obtained through uniformly distributed random values for each attribute. According to the attribute values of an instance, the leaf reached after a traverse of the tree determines its class value. RTG allows customizing the number of nominal and numeric attributes, as well as the number of classes. In our experiments, we did not simulate drifts for the RTG data set. Since the concepts are generated and classified according to a tree structure, it should favor decision tree learners in theory.

RBF. This generator produces data sets through the Radial Basis Function (RBF) [120]. This generator creates several centroids, having a random central position, and associates them with a standard deviation value, a weight, and a class label. A centroid is randomly selected, while centroids with higher weights have more chances to be selected. The new instance input values are set according to a random direction chosen to offset the centroid. The displacement is randomly drawn from a Gaussian distribution according to the standard deviation associated with the given centroid. Incremental drift is introduced by moving centroids at a continuous rate, effectively causing new instances to belong to new centroids with a potential class change. Both $\mathrm{RBF_m}$ and $\mathrm{RBF_f}$ were parametrized with 50 centroids while always implementing drift. $\mathrm{RBF_m}$ simulates a moderate incremental drift (speed of change set to 0.0001) while $\mathrm{RBF_f}$ simulates a faster incremental drift (speed of change set to 0.001).

HYPER. The HYPER data set simulates an incremental drift, and it was generated based on the hyperplane generator [9]. A hyperplane is a flat, $n-1$ dimensional subset of that space that divides it into two separate parts. It is possible to change a hyperplane orientation and position by slightly changing its relative size of the weights w_i. This generator can be used to simulate time-changing concepts by varying the values of its weights as the stream progresses [120]. HYPER was parametrized with ten attributes and a magnitude of change of 0.001. Also, ten percent noise was added.

REAL-WORLD STREAMS

The real-world streams are finite in length, and Concept Drift from a statistical standpoint is unknown. However, some streams have scenarios, which change by nature, like COVTYPE.

GMSC. The Give Me Some Credit (GMSC) data set[2] is a credit scoring data set where the objective is to decide whether a loan should be allowed or not. This decision is essential for banks since erroneous loans lead to the risk of default and unnecessary expenses on future lawsuits. The data set contains historical data on 150,000 borrowers, each described by ten attributes.

2 https://www.kaggle.com/c/GiveMeSomeCredit

Electricity. The Electricity data set[3] was collected from the Australian New South Wales Electricity Market, where prices are not fixed. These prices are affected by the demand and supply of the market itself and set every five minutes. The Electricity data set contains 45,312 instances, where class labels identify the changes in the price (two possible classes: up or down) relative to a moving average of the last 24 hours. An important aspect of this data set is that it exhibits temporal dependencies [13].

Poker-Hand. The Poker-Hand data set consists of 1,000,000 instances and eleven attributes. Each record of the Poker-Hand data set is an example of a hand consisting of five playing cards drawn from a standard deck of 52. Each card is described using two attributes (suit and rank) for ten predictive attributes. There is one class attribute that describes the *Poker Hand*. This data set has no drift in its original form since the poker hand definitions do not change, and the instances are randomly generated. Thus, the version presented in [54] is used, in which virtual drift is introduced via sorting the instances by rank and suit. Duplicate hands were removed.

Forest Cover Type. Forest Cover Type (COVTYPE) is a data set, which is often used to benchmark stream mining algorithms [10], [54]. The dataset assigns cartographic variables like elevation, soil type, slope, etc., of 30 square meter cells to different forest cover types. It only includes forests with minimal human-caused disturbances so that the contained cover types are primarily a result of ecological processes.

Airlines. The task on the Airlines dataset is to predict whether a planned flight will be delayed or not. It contains the airport of departure and arrival, as well as the airline and time-related features. The delay is only represented as a binary attribute. The dataset is used in the form as the MOA framework [120] provides it.

SQRE. The Moving Squares dataset has been used in [104], and consists of four equally distant separated squared Uniform distributions which are moving in a horizontal direction with constant speed. Whenever the leading square reaches a predefined boundary, the direction is inverted. Each square represents a different class. The unique selling point of the streams is the predefined time horizon of 120 examples before old instances may start to overlap current ones. Thus, the dataset should be useful for dynamic sliding window approaches, allowing testing whether the size is adjusted accordingly.

Weather (WTHR). As the name indicates, the weather stream dataset records eight different features such as temperature, pressure, wind speed, etc., from 1949-1999 measured at the Offutt Air Force Base in Bellevue, Nebraska. It is a binary classification problem with the target to predict whether it is going to rain on a certain day or not [104]. The dataset has 18.159 records with a class imbalance towards rain of 69%. The dataset was first introduced by [33].

3 https://www.openml.org/d/151