

Submitted by
Marius-Constantin Dinu
11728074

Submitted at
**Institute for Machine
Learning**

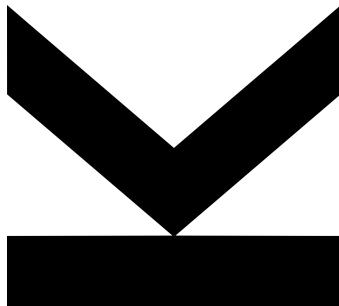
Thesis Supervisor / First
Evaluator
Sepp Hochreiter

Second Evaluator
Gary Marcus

Co-Supervisor
Werner Zellinger

July 2024

Parameter Choice and Neuro-Symbolic Approaches for Deep Domain- Invariant Learning



Doctoral Thesis
to obtain the academic degree of
Doktor der technischen Wissenschaften
in the Doctoral Program
Technische Wissenschaften

Acknowledgments

My academic journey has been supported by countless individuals — researchers advancing our field, family, colleagues, and friends. Though too numerous to name, their contributions are deeply appreciated.

Firstly, I am deeply indebted to Sepp Hochreiter for establishing this institute, which has significantly shaped my life. His dedication to sharing his knowledge and experience with me has been invaluable, and his expertise continues to inspire me. Words cannot fully capture my appreciation.

Similarly, I express my heartfelt thanks to Werner Zellinger and Sergei Pereverzyev. Werner, in particular, has been a vital mentor during the latter part of my studies. His insightful discussions and advice have been incredibly beneficial. Likewise, Sergei's contributions have been significant, and I've learned a great deal from his work and insights.

I extend my gratitude to my colleagues and friends at the Institute for Machine Learning at Johannes Kepler University. I cherish the engaging discussions, the supportive and friendly environment, the lasting friendships, the exciting conference experiences, and the countless enjoyable moments we've shared. Their support has been critical in navigating a field that is both promising and increasingly challenging, competitive, and sometimes discouraging to pursue a PhD in. I am grateful to Gary Marcus, whose work and our discussions significantly inspired and contributed to the ideas in this thesis. I am also thankful to all my co-authors and peers, whom I thoroughly enjoyed collaborating with. Special mentions go to Claudiu Leoveanu-Condrei, Markus Hofmarcher, Markus Holzleitner, Andreas Radler, Kajetan Schweighofer, José Arjona-Medina and Lukas Gruber. Furthermore, I am grateful to Michael Laux, whose profound wisdom and knowledge, despite brief interactions, offered invaluable guidance and inspiration that will shape my work and perspective.

Finally, I want to express my deepest gratitude to my parents, friends, family, and especially Laura. Your unwavering support has been my cornerstone, without which I could not have pursued my academic goals.

Abstract

As artificial intelligence (AI) systems advance, we move towards broad AI: systems capable of performing well on diverse tasks, understanding context, and adapting rapidly to new scenarios. A central challenge for broad AI systems is to generalize over tasks in related domains and being robust to distribution shifts. Neuro-symbolic (NeSy) AI bridges the gap between symbolic and sub-symbolic paradigms to address these challenges. This enables us to create adaptable, generalizable, and more interpretable systems. These characteristics are fundamental for broad AI systems, which aim to solve a wide range of tasks across various domains. Consequently, the development of broad AI requires advancements in domain adaptation (DA), enabling models trained on source domains to effectively generalize to unseen target domains. Traditional approaches often rely on parameter optimization, fine-tuning, and representation learning methods, which can be impractical due to high costs and risks of catastrophic forgetting. In contrast, NeSy AI systems use multiple models and methods to generalize to unseen domains and maintain performance across varying conditions. However, building scalable and generalizable hybrid systems remains challenging.

In this work, we analyze common DA and NeSy approaches with the focus on deep domain-invariant learning. This analysis extends to real-world challenges, such as adapting to continuously changing domains, handling missing modalities, and dealing with large domain gaps between source and target domains. We showcase state-of-the-art model-selection and parameter choice methods for scenarios with limited samples where gradient-based optimization is feasible. For cases where model tuning is infeasible, we introduce domain-specific adaptations without gradient-based updates, connecting in-context learning to domain-invariant learning. This approach presents NeSy methodologies that enable adaptation in previously challenging situations. Therefore, my work establishes a framework for scalable and generalizable broad AI systems applicable across various problem settings. Furthermore, it demonstrates how symbolic reasoning and large language models (LLMs) can build universal computational graphs that generalize across domains and problems, contributing to more adaptable, generalizable, and interpretable AI approaches for real-world applications.

Marius-Constantin Dinu

PUBLICATIONS

- [1] **M.-C. Dinu**, C. Leoveanu-Condrei, M. Holzleitner, W. Zellinger, and S. Hochreiter, “SymbolicAI: A framework for logic-based approaches combining generative models and solvers”, in *Conference on Lifelong Learning Agents, PMLR*, 2024.
- [2] **M.-C. Dinu**, C. Leoveanu-Condrei, M. Holzleitner, W. Zellinger, and S. Hochreiter, “SymbolicAI: A framework for logic-based approaches combining generative models and solvers”, in *GenAI4DM Workshop at The Twelfth International Conference on Learning Representations*, 2024.
- [3] A. Patel, M. Hofmarcher, C. Leoveanu-Condrei, **M.-C. Dinu**, C. Callison-Burch, and S. Hochreiter, “Large language models can self-improve at web agent tasks”, in *Advances in Neural Information Processing Systems (under review)*, 2024.
- [4] **M.-C. Dinu**, M. Holzleitner, M. Beck, H. D. Nguyen, A. Huber, H. Eghbal-zadeh, B. A. Moser, S. Pereverzyev, S. Hochreiter, and W. Zellinger, “Addressing parameter choice issues in unsupervised domain adaptation by aggregation”, in *International Conference on Learning Representations*, 2023.
- [5] **M.-C. Dinu***, M. Hofmarcher*, V. P. Patil, M. Dorfer, P. M. Blies, J. Brandstetter, J. Arjona-Medina, and S. S. Hochreiter, “XAI and strategy extraction via reward redistribution”, in *xxAI - Beyond Explainable Artificial Intelligence: State-of-the-Art and Future Challenges*, ser. Lecture Notes in Artificial Intelligence, vol. LNAI 13200, Springer International Publishing, May 2022.
- [6] V. Patil*, M. Hofmarcher*, **M.-C. Dinu**, M. Dorfer, P. M. Blies, J. Brandstetter, J. Arjona-Medina, and S. Hochreiter, “Align-RUDDER: Learning from few demonstrations by reward redistribution”, in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 162, PMLR, Jul. 2022, pp. 17531–17572.
- [7] K. Schweighofer*, A. Radler*, **M.-C. Dinu***, M. Hofmarcher, V. Patil, A. Bitto-Nemling, H. Eghbal-Zadeh, and S. Hochreiter, “A dataset perspective on offline reinforcement learning”, in *First Conference on Lifelong Learning Agents*, Aug. 2022.
- [8] R. Siripurapu, V. P. Patil, K. Schweighofer, **M.-C. Dinu**, T. Schmied, L. E. F. Diez, M. Holzleitner, H. Eghbal-Zadeh, M. K. Kopp, and S. Hochreiter, “InfODist: Online distillation with informative rewards improves generalization in curriculum learning”, in *Deep Reinforcement Learning Workshop NeurIPS*, 2022.
- [9] C. A. Steinparz, T. Schmied, F. Paischer, **M.-C. Dinu**, V. P. Patil, A. Bitto-Nemling, H. Eghbal-zadeh, and S. Hochreiter, “Reactive exploration to cope with non-stationarity in lifelong reinforcement learning”, in *Conference on Lifelong Learning Agents*, PMLR, 2022, pp. 441–469.
- [10] K. Schweighofer, M. Hofmarcher, **M.-C. Dinu**, P. Renz, A. Bitto-Nemling, V. P. Patil, and S. Hochreiter, “Understanding the effects of dataset characteristics on offline reinforcement learning”, in *Deep Reinforcement Learning Workshop NeurIPS 2021*, Dec. 2021.
- [11] W. Zellinger, N. Shepeleva, **M.-C. Dinu**, H. Eghbal-zadeh, H. D. Nguyen, B. Nessler, S. Pereverzyev, and B. A. Moser, “The balancing principle for parameter choice in distance-regularized domain adaptation”, in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34, Curran Associates, Inc., 2021, pp. 20798–20811.

* Equal contribution.

- [12] M. Holzleitner, J. A. Arjona-Medina, **M.-C. Dinu**, A. Vall, L. Gruber, and S. Hochreiter, “A two time-scale update rule ensuring convergence of episodic reinforcement learning algorithms at the example of rudder”, *NeurIPS Optimization Foundations for Reinforcement Learning Workshop*, 2019.

TEACHING

- **Lecturer** at Johannes Kepler University Linz
Exercise in Deep Reinforcement Learning (365.250) Summer 2021
- **Lecturer** at Johannes Kepler University Linz
Exercise in Deep Reinforcement Learning (365.250) Summer 2022
- **Lecturer** at Johannes Kepler University Linz
Exercise in Deep Reinforcement Learning (365.250) Summer 2023

Contents

1	Introduction	1
1.1	Domain Shift, Domain Adaptation and Domain-Invariant Learning	4
1.2	Model Selection and Parameter Choice Methods for Unsupervised Domain Adaptation	6
1.3	In-Context Learning	7
1.4	A Neuro-Symbolic Perspective on Large Language Models	22
1.5	List of Publications	24
2	Selected Publications	27
2.1	The balancing principle for parameter choice in distance-regularized domain adaptation	28
2.2	Addressing Parameter Choice Issues in Unsupervised Domain Adaptation by Aggregation	56
2.3	<i>SymbolicAI</i> : A framework for logic-based approaches combining generative models and solvers	108
3	Conclusion and Outlook	155
3.1	Future Work	155
A	Glossary	167

Chapter 1

Introduction

As artificial intelligence (AI) systems advance, we are moving towards *broad AI* capable of performing well on diverse tasks, understanding context, and adapting rapidly to new scenarios. Broad AI, as conceptualized by Hochreiter (2022), represents a significant advancement over current narrow AI systems. It aims to create sophisticated and adaptive systems characterized by enhanced capabilities in knowledge transfer and interaction, adaptability and robustness, abstraction and advanced reasoning, and efficiency. These systems are designed to quickly adapt to new situations, customers, products, processes, workflows, or sensory inputs, addressing the limitations of current deep learning approaches which often underperform in real-world applications due to their data-hungry nature and limited transfer capabilities. Key challenges in developing such broad AI systems include distribution shifts and generalization. Addressing these challenges is crucial for creating AI systems that can solve a wide range of tasks across various domains and overcome the limitations that currently hinder the widespread adoption of AI in critical real-world scenarios. Neuro-symbolic (NeSy) AI emerges as a promising approach to tackle these challenges by bridging the gap between symbolic and sub-symbolic paradigms of AI (Marcus, 2001, 2020). It combines the strengths of both approaches while mitigating their individual weaknesses. Symbolic AI excels in logical reasoning, interpretability, and explicit knowledge representation, but struggles with adaptability and learning from raw data. Sub-symbolic AI, exemplified by deep neural networks, excels in pattern recognition and learning from large datasets, but often lacks interpretability and struggles with abstract reasoning. NeSy AI aims to create synergy between these approaches by integrating neural learning with symbolic knowledge and reasoning (Garcez et al., 2008, 2015; Besold et al., 2017; Garcez et al., 2019; Garcez and Lamb, 2020; Lamb et al., 2020; Hamilton et al., 2022; Yu et al., 2023). This integration is challenging due to the fundamentally different nature of these paradigms, which has historically limited their combination. However, recent advances in machine learning —

specifically due to advances made by large language models (LLMs) (Vaswani et al., 2017; Wei et al., 2022; Singhal et al., 2023; Schick et al., 2023) — and knowledge representation (Augusto, 2022) have made it possible to create hybrid systems that leverage the strengths of both approaches. LLMs provide a comprehensive natural language understanding and generalization capability, and enable a seamless combination of logical reasoning with statistical approaches. NeSy AI systems can potentially offer improved adaptability, robustness, generalizability, and interpretability compared to purely symbolic or sub-symbolic systems. These characteristics are essential for developing broad AI systems, which are designed to solve a wide range of tasks across various domains and adapt to new challenges without extensive re-training or fine-tuning. Therefore, to create broad AI systems, it is essential to advance the field of DA and, in particular, domain-invariant learning.

Domain adaptation (DA) focuses on training a model on data from a set of source domains to help generalize the model to unseen target domains (Widmer and Kubat, 1996; Ben-David et al., 2010; Liu et al., 2022). DA addresses the domain shift between the source and target domains, which is useful in scenarios where labeled data is scarce or expensive to obtain, making it impractical to re-train models from scratch for each new domain. Domain-invariant learning is a DA methodology that aims to learn shared features that are robust and consistent across different domains. Core techniques to domain-invariant learning include fine-tuning (Sugiyama et al., 2007; You et al., 2019; Zellinger et al., 2021), representation learning (Tzeng et al., 2014; Ganin et al., 2016; Sun et al., 2017; Zellinger et al., 2017; Chen et al., 2020; Rahman et al., 2020; Liu and Xue, 2021; Zhu et al., 2021), and novel parameter choice methods (Sugiyama et al., 2007, 2012; Saito et al., 2021; Zellinger et al., 2021; Musgrave et al., 2021). Parameter choice methods involve selecting optimal hyperparameters, regularization techniques, and model architectures to effectively transfer knowledge from source to target domains. Fine-tuning involves adjusting pre-trained models to adapt to new domains, while representation learning enables the learning of transferable features or embeddings that capture domain-invariant information. In the context of DA, both approaches aim to allow models to adapt to new domains by utilizing shared representations, while minimizing the need for extensive retraining.

However, several problems appear in practice: fine-tuning models that exceed 100 billion parameters (Brown et al., 2020; Google, 2023; Touvron et al., 2023; Jiang et al., 2023) is often too computationally expensive or restricted due to proprietary limitations on accessing model weights. Additionally, the rate of change in real-world data is often intractable. Therefore, these approaches are not always feasible or applicable. Furthermore, representation learning approaches lack interpretability and suffer from *catastrophic forgetting* (Hebb, 1949; Carpenter and Grossberg, 1987; McCloskey and Cohen, 1989; Mermilliod et al., 2013; Ehret et al., 2020), when the model is exposed

to new domains, which stems from the underlying shift in data distribution across domains. Catastrophic forgetting refers to a model forgetting previously learned information upon learning new information, particularly in sequential learning tasks.

In contrast to traditional single-model approaches, NeSy AI systems leverage multiple models and methods to address domain-specific requirements and generalize across varying conditions. They can select between parametric, non-parametric and symbolic methods that incorporate symbolic reasoning up to powerful pattern recognition abilities, structured knowledge bases, and search engines. This diverse toolkit enables more adaptable and interpretable methodologies. A hybrid paradigm also enables the formulation of higher-level abstractions and logical expressions through dedicated symbolic approaches. It incorporates *learning-for-reasoning* methods to treat the learning aspect as an accelerator for reasoning, in which deep neural networks are employed to reduce the search space for symbolic systems (Silver et al., 2016, 2017b,a; Qu and Tang, 2019; Schrittwieser et al., 2020). *Reasoning-for-learning* that views reasoning as a way to regularize learning, in which symbolic knowledge acts as a guiding constraint that oversees machine learning tasks (Hu et al., 2016; Xu et al., 2018), and *learning-reasoning* to enable a symbiotic relationship between learning and reasoning to boost problem-solving capabilities (Donadello et al., 2017; Manhaeve et al., 2018; Mao et al., 2019; Ellis, 2023). While NeSy AI systems offer numerous advantages, their implementation presents significant challenges. In practice, building such hybrid systems is challenging due to the inherent complexity of integrating disparate symbolic and neural components, as well as the difficulty in designing interfaces that allow seamless interaction between these components. This complexity often results in ad-hoc implementations that scale poorly and don't generalize across a variety of tasks and domains. The lack of standardized frameworks for NeSy integration further exacerbates these issues, leading to solutions that are often specific to particular problem domains and difficult to adapt to new scenarios.

Motivated by these limitations and the potential of NeSy AI, my work analyzes common DA and NeSy approaches, with a particular focus on deep domain-invariant learning. This analysis is conducted from two complementary perspectives: First, we address DA issues faced when dealing with restricted sample sizes and where gradient-based optimization is feasible, and propose novel model-selection and parameter choice methods (Zellinger et al., 2021; Dinu et al., 2023). Second, we establish a connection between in-context learning (Vaswani et al., 2017) — the ability of LLMs to adapt to new tasks through prompting — and domain-invariant learning. This approach focuses on NeSy methodologies in situations where tuning models is infeasible, introducing domain-specific adaptations without gradient-based updates (Dinu, 2022; Dinu et al., 2024a,b).

As a result, this work summarizes the following contributions:

- novel model-selection and parameter choice methods for unsupervised domain adaptation,
- a novel NeSy approach based on LLMs in situations where fine-tuning models is infeasible,
- a scalable and generalizable framework for broad AI systems, which enable domain-specific adaptations,
- novel benchmarks for parameter choice methodologies and NeSy methodologies based on LLMs,
- how symbolic reasoning and LLMs can be connected to build universal computational graphs that generalize across domains and problems.

Therefore, our research contributes to the development of more transferable, robust, and interpretable AI approaches that meet the demands of real-world applications, such as robotics, personalized healthcare systems, and business intelligence platforms (Tesla, 2022; NVIDIA, 2024).

1.1 Domain Shift, Domain Adaptation and Domain-Invariant Learning

The challenges of domain shift and the scarcity of labeled data present significant constraints to traditional deep learning approaches. To illustrate, in the automotive industry, models designed for classification and segmentation tasks are initially trained in a simulated setting, potentially with full access to labeled data, and aimed at being transferred to real-world applications. Despite the profusion of unlabeled real-world data, for instance, video recordings under various environmental conditions (Tesla, 2022), this task remains challenging due to intricate domain shifts between the simulated data and the target domain (Dosovitskiy et al., 2017).

The domain shift refers to the changes in the data distribution between the source domain (distribution p) where the model is trained, and the target domain (distribution q) where the model is applied. This shift can manifest in various forms, such as changes in feature distribution $p_{\mathcal{X}} \neq q_{\mathcal{X}}$, label distribution $p_{\mathcal{Y}} \neq q_{\mathcal{Y}}$, and conditional distribution between labels given features $p_{\mathcal{Y}|\mathcal{X}} \neq q_{\mathcal{Y}|\mathcal{X}}$.

To address these challenges systematically, we turn to the field of unsupervised domain adaptation (UDA). UDA provides a formal framework for transferring knowledge from a labeled source domain to an unlabeled target domain, directly tackling the issues of domain shift and label scarcity. In the following section, we present the mathematical foundations of UDA, which will serve as the basis for our subsequent analysis and proposed methods.

Unsupervised Domain Adaptation Let $\mathcal{X} \subset \mathbb{R}^{d_1}$ be a compact *input space* and $\mathcal{Y} \subset \mathbb{R}^{d_2}$ be a compact *label space*. Following Ben-David et al. (2010), we consider two datasets: A *source dataset* $(\mathbf{x}, \mathbf{y}) = ((x_1, y_1), \dots, (x_n, y_n)) \in (\mathcal{X} \times \mathcal{Y})^n$ independently drawn according to some source distribution (probability measure) p on $\mathcal{X} \times \mathcal{Y}$ and an unlabeled *target* dataset $\mathbf{x}' = (x'_1, \dots, x'_m) \in \mathcal{X}^m$ with elements independently drawn according to the marginal $q_{\mathcal{X}}$. The marginal distribution of p on \mathcal{X} is analogously denoted as $p_{\mathcal{X}}$. The goal is to find $f : \mathcal{X} \rightarrow \mathcal{Y}$ with small *target risk* $\mathcal{R}_q(f)$. For simplicity, we consider the *expected target risk* $\mathcal{R}_q(f) = \int_{\mathcal{X} \times \mathcal{Y}} \|f(x) - y\|_{\mathcal{Y}}^2 dq(x, y)$ with least squares loss. From this formulation, we can see that UDA aims to adapt a model trained on a labeled source domain to perform well on a new, unlabeled target domain, effectively addressing the challenges of domain shift and the absence of labeled data in the target domain (Ben-David et al., 2010; Liu et al., 2022).

Domain-invariant learning is a general principle to construct algorithms for DA that focuses on learning a feature extractor $\varphi : \mathcal{X} \rightarrow \mathbb{R}^d$ that maps input data $x \in \mathcal{X}$ to a shared d -dimensional *representation space \mathbb{R}^d with $d \geq 1$. The goal is to obtain representations $\varphi(x)$ that are similar across both the source domain and the target domain. More specifically, we aim to learn φ such that the associated probability distribution of $\varphi(X_p)$ and $\varphi(X_q)$ are as close as possible, where X_p and X_q are random variables with distribution p and q , respectively, often measured using distribution distance metrics or divergences such as Maximum Mean Discrepancy (Gretton et al., 2012), Wasserstein distance and Central Moment Discrepancy (Zellinger et al., 2017). See Figure 1.1 for a visual illustration of domain-invariant learning.*

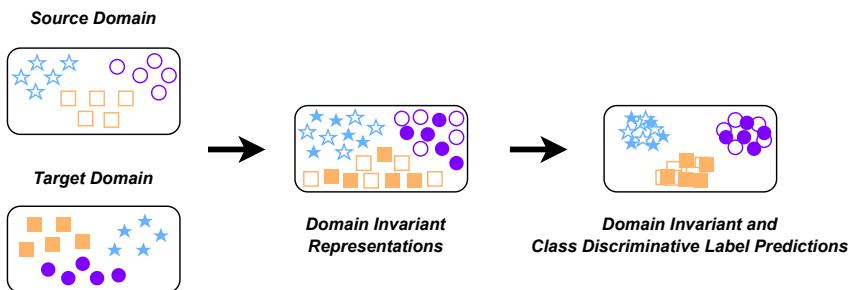


Figure 1.1: Illustration of domain-invariant learning. Source and target domain features are transformed into domain-invariant features, which are then made class discriminative. This enables the model to distinguish between different classes while maintaining consistent performance across various domains.

1.2 Model Selection and Parameter Choice Methods for Unsupervised Domain Adaptation

Implementing effective UDA methods requires careful selection of model architectures and hyperparameters. This section examines approaches to model selection and parameter choice that aim to optimize UDA performance across diverse domains, focusing on how these methods can be applied to various UDA techniques such as moment matching (Zellinger et al., 2017; Gretton et al., 2012), divergence-based (Li et al., 2020), or discriminative approaches (Ganin et al., 2016), among others.

Parameter Choice Problem Typical approaches to parameter choice in UDA first involve training a sequence of m models with different hyperparameter settings. This often includes modifying parameters such as the learning rate or weighting factors of loss terms. A current best practice for a second step is then to apply a model selection method to obtain the best-performing model with the minimum risk on the target domain. However, the state-of-the-art model selection methods (Sugiyama et al., 2007; You et al., 2019; Zellinger et al., 2021) can only perform as well as the best model within the sequence and disregard the $m - 1$ trained models, in spite of the computational power and time invested for training. Furthermore, the choice of models and parameters often hinges on the task-specific characteristics of the dataset, necessitating dedicated experimentation.

Parameter Choice Contribution In our work (Dinu et al., 2023), we go beyond selecting a single model from the sequence of models by computing a weighted aggregation of all models within the sequence.

More precisely, we calculate an ensemble $f^{\text{agg}} = \sum_{k=1}^m c_k f_k$ with minimal target risk $\mathcal{R}_q(f^{\text{agg}}) = \min_{c_1, \dots, c_m \in \mathbb{R}^m} \mathcal{R}_q(\sum_{k=1}^m c_k f_k)$. Here f_k denotes the models within the sequence and c_k the aggregation coefficients. The optimal aggregation risk is less than or equal to the risk of the best model obtained by model selection, i.e., $\mathcal{R}_q(f^{\text{agg}}) \leq \mathcal{R}_q(f^{\text{sel}})$, because this model is included within the sequence of aggregated models. Using vector-valued least squares, the vector of optimal coefficients can be calculated as follows (see Section 2.2):

$$c = G^{-1}g = \arg \min_{(c_1, \dots, c_m) \in \mathbb{R}^m} \int_X \left\| \sum_{k=1}^m c_k f_k(x) - f_q(x) \right\|_Y^2 dq_{\mathcal{X}}(x), \quad (1.1)$$

where $f_q(x) = \int_Y y dq(y|x)$ is the Bayesian predictor (the model with minimal risk on the target domain). The Gram matrix G consists of scalar products between the predictions of the individual models within the sequence, i.e. that $G = (\int_X \langle f_k(x), f_u(x) \rangle_Y dq_{\mathcal{X}}(x))_{k,u=1}^m$, and $g = (\int_X \langle f_q(x), f_k(x) \rangle_Y dq_{\mathcal{X}}(x))_{k=1}^m$. Since calculating f_q requires access to the

labels of the target domain, to overcome this problem, under the assumptions of 1) a covariate shift and 2) a bounded density ratio between the marginals of the source distribution p and target distribution q , so that $\beta(x) = \frac{dq_x}{dp_x}(x) \in [0, B]$ with some constant $B > 0$, we arrive at the following result:

$$g = \left(\int_X \langle f_p(x), f_k(x) \rangle_Y \beta(x) d\mu_X(x) \right)_{k=1}^m. \quad (1.2)$$

Equation (1.2) computes the expectation over scalar products between the sequence of models and the Bayesian predictor f_p weighted by the density ratio $\beta(x)$ to adjust for the domain shift.

To realize the computations of Equation (1.1) algorithmically, we employ the following two steps: First, we estimate the density ratio $\beta(x)$ (see e.g. Sugiyama et al. (2012) for a collection of possible methods). Second, we calculate the empirical estimators for the Gram matrix $\tilde{G} = (\frac{1}{t} \sum_{i=1}^t \langle f_k(x'_i), f_u(x'_i) \rangle_Y)_{k,u=1}^m$ and vector $\tilde{g} = (\frac{1}{s} \sum_{i=1}^s \beta(x_i) \langle y_i, f_k(x_i) \rangle_Y)_{k=1}^m$ based on our datasets, and gain aggregation coefficients $\tilde{c} = \tilde{G}^{-1} \tilde{g}$.

Our method defines the state of the art in parameter choice for UDA, as it outperforms the current best practices in model selection with theoretical guarantees. We also performed large-scale empirical evaluations on a text-based domain (AmazonReviews (McAuley, 2021)), an image-based domain (DomainNet (Peng et al., 2018)), and four time-series datasets (Ragab et al., 2023), featuring a heterogeneity dataset for human activity recognition, electroencephalogram recordings of sleep activities, as well as smartphone and smartwatch activity and biometrics datasets.

1.3 In-Context Learning

Large language models (LLMs) have recently demonstrated promising capabilities (Kojima et al., 2022; Huang et al., 2022; Ruis et al., 2022; Weng et al., 2023; Fang et al., 2024). State-of-the-art models typically exceed 10 billion parameters and exhibit basic reasoning capabilities through in-context learning (Brown et al., 2020), as seen with Chain-of-Thought (Wei et al., 2022) and Tree-of-Thought prompting (Yao et al., 2023).

In-context learning is an *emergent phenomenon* in LLMs that occurs after extensive pretraining on vast datasets. The models have been trained on countless domains and tasks, and their in-context learning capabilities enable them to generalize across various tasks and domains. Unlike traditional learning paradigms that rely on gradient-based updates to adjust model parameters, in-context learning leverages the model’s ability to adapt to specific tasks without changing its underlying weights. We illustrate this with the following example.

Example 1. *Prompt-based Instructions:*

A typical in-context learning prompt is constructed based on an instruction, e.g., "Perform a fuzzy comparison between the following statements and determine if they are approximately equal or not:", some contextual patterns as few-shot examples:

```
1 1 = "ONE" => True,
2 6.0 = "SIX" => True,
3 1 = "two" => False,
4 "five" = 5 => True,
5 ten = 10 => True,
6 4 = "Three" => False,
7 "four" = "FOUR" => True,
8 ...
```

and the new query statement which we want the LLM to complete:

```
1 "eight" = 8 =>
```

The LLM leverages the provided examples to create associations between numbers and words, allowing it to interpret the fuzzy comparison between the presented values and extrapolate to similar examples. This enables the LLM to infer that "eight" and "8" are approximately equal in this context, leading it to predict "True" as the correct outcome for the new query statement.

Large Language Models To begin our analysis, we first consider a simple prompt: "Hello, my world!", and its possible tokenization:

[He", "l", "l", "o", " ", "my", "w", "o", "r", "ld", "!"]

The vocabulary \mathcal{V} might then include these tokens and many others:

$$\mathcal{V} = \{\text{"He"}, \text{"l"}, \text{"o"}, \text{"w"}, \text{"r"}, \text{"ld"}, \text{"!"}, \text{" "}, \text{"my"}, \text{"us"}, \text{"of"}, \dots\}$$

This example illustrates how a sentence is converted into tokens, and how the vocabulary contains whole words, subword units and characters.

More formally, let \mathcal{V} be a finite set representing the vocabulary of the language model. We then define \mathcal{V}^* as the set of all possible token sequences (including the empty sequence) of elements from the finite set of tokens \mathcal{V} . We call an element $x \in \mathcal{X} \subset \mathcal{V}^*$ a *sequences of tokens* from the *input space* and an element $y \in \mathcal{Y} \subset \mathcal{V}^*$ a sequence of tokens from the *output space*.

For our analysis, we first define the subspace $\mathcal{P} \subset \mathcal{X} \subset \mathcal{V}^*$ as the space of all meaningful prompts and subspaces $\mathcal{I} \subset \mathcal{X}$, $\mathcal{C} \subset \mathcal{X} \times \mathcal{Y}$, and $\mathcal{S} \subset \mathcal{X}$ representing the instruction space, context space, and query space, respectively. We then define the prompt input $x \in \mathcal{P} = \mathcal{I} \times \mathcal{C} \times \mathcal{S}$ as a tuple $x = (i, c, s)$, where $i \in \mathcal{I}$, $c \in \mathcal{C}$, and $s \in \mathcal{S}$. Now, let $f : \mathcal{P} \rightarrow \mathcal{Y}$ represent the LLM which predicts outputs with desired targets $y = f(x) \in \mathcal{Y}$.

In-Context Learning and Domain-Invariant Learning In the context of domain-invariant learning, we consider the prompt domain \mathcal{P} , where samples are drawn from two probability distributions over this domain: a source distribution $p(i, c, s)$ and a target distribution $q(i, c, s)$. The basic intuition is that an LLM f learns representations that are invariant across different contexts, enabling generalization from source samples to target samples. In-context learning then minimizes the expected target risk $\mathcal{R}_q(f)$ not by explicitly updating the model parameters, but through implicit adaptation based on the provided context examples, which is often referred to as *prompt engineering* (Nori et al., 2023).

The in-context learning approach can then be summarized as follows:

1. **Encoding:** Given an embedding function $\rho : \mathcal{V} \rightarrow \mathbb{R}^d$ with $d \geq 1$ we encode the instructions $i \in \mathcal{I}$, context $c \in \mathcal{C}$, and query statement $s \in \mathcal{S}$ into vector representations:

$$\begin{aligned}\mathbf{i}_{1:m} &= (\rho(i_1), \rho(i_2), \dots, \rho(i_m)) \in \mathbb{R}^{m \times d} = (\rho(i_j))_{j=1}^m, \\ \mathbf{c}_{1:k} &= (\rho(c_1), \rho(c_2), \dots, \rho(c_k)) \in \mathbb{R}^{k \times d} = (\rho(c_j))_{j=1}^k, \\ \mathbf{s}_{1:n} &= (\rho(s_1), \rho(s_2), \dots, \rho(s_n)) \in \mathbb{R}^{n \times d} = (\rho(s_j))_{j=1}^n.\end{aligned}$$

2. **Concatenation:** The embedded vectors are concatenated into a single sequence (with lengths k_1 and k_2 of the two context sequences):

$$\mathbf{z}_{1:j=m+k_1+k_2+n}^1 = [\mathbf{i}_{1:m}, \mathbf{c}_{1:k_1+k_2}, \mathbf{s}_{1:n}] = (z_i^1)_{i=1}^{j=m+k_1+k_2+n}.$$

3. **Transformations:** Transformation functions φ_{trans}^l (e.g., attention mechanisms, feed-forward neural networks, normalization, etc.) are applied across different hidden layers with $1 < l < L$, such that the latent sequences $\mathbf{z}_{1:j}^{l-1}$ produces the subsequent latent representation $\mathbf{z}_{1:j}^l$ with latent dimension d_l :

$$\mathbf{z}_{1:j}^l = \varphi_{\text{trans}}^l(\mathbf{z}_{1:j}^{l-1}) \in \mathbb{R}^{j \times d_l}.$$

4. **Decoding:** The last latent representation $\mathbf{z}_{1:j}^L \in \mathbb{R}^{d_L}$ is usually aggregating the sequences before it is then decoded to obtain the output $y' \in \mathcal{V}$:

$$y' = \varphi_{\text{dec}}(\mathbf{z}_{1:j}^L) \in \mathcal{V},$$

where $\varphi_{\text{dec}} : \mathbb{R}^{d_L} \rightarrow \mathcal{V}$ maps from the latent space to a token in the output space.

5. **Auto-regression:** In this step, the prediction of the decoder is usually appended to the input, and the process repeats from steps 1 to 4 until halting criteria, such as a stop token, are encountered. This auto-regressive process generates the predicted sequence $\hat{y} \in \mathcal{V}^*$.

We view in-context learning as deeply rooted in DA and related to domain-invariant learning, since LLMs seem to learn shared feature representations $\varphi_{\text{trans}}^l(\cdot)$ that generalize across diverse tasks and domains without explicit retraining. This is because LLMs are trained on a large corpus of data that includes various prompt-to-prediction pairs from different domains. This training enables an LLMs to form domain-invariant representations that generalize across different tasks, even to those they have not specifically been trained for. In our paper Dinu et al. (2024a), we accredit this to the associations formed between different domains during training based on the instructions and contextual sequences for problem-specific tasks.

We draw this conclusion based on the evidence of analyzing the memorization and learning capabilities of LLMs (Carlini et al., 2021; Min et al., 2022; Hartmann et al., 2023; Oswald et al., 2023; Duan et al., 2024), polysemanticity and capacity in neural networks (Scherlis et al., 2022; Zhao et al., 2024), and semantic feature spaces (Wolfram, 2023). To illustrate, Figure 1.2 presents a UMAP projection (McInnes et al., 2018) of the latent space, visualizing the semantic relationships between four different domains based on their embedded representations in the LLM. We show the embedding activations of the last layer of the open-source GPT-Neo 1.3 billion parameter model (Black et al., 2021) for the respective domains. Here, *domain* represents a specific data distribution of token sequences. These domains are denoted as *Mathematical*, *Programming*, *Natural Language*, and *Random* domain. Each dataset includes examples representative of the data distribution of token sequences for the respective domains.

However, we also see an overlap between the first three domains, since they have semantically related features, such as " $a = b$ " and " $a \text{ equals } b$ ". This is possible because during training, the model developed a semantic mapping that captures cross-domain correspondences and contextual synonyms. In-context learning prompts then interpolate between these feature representations by establishing new in-context associations between sequences and, consequently, between data distributions, thereby enforcing domain-invariant predictions. The power of in-context learning lies in its ability to generalize representations across various tasks and domains by leveraging semantically related features for context-specific adaptations of data distributions.

Structure of In-Context Learning for Domain Adaptation LLMs are usually trained on associated pairs with instructions $i \in \mathcal{I}$, labeled contexts $c \in \mathcal{C}$ sampled w.r.t. to their marginal source and target distribution respectively, and unlabeled query statements $s \in \mathcal{S}$. The contexts include expected labeled outcomes $y \in \mathcal{Y}$ that create a bridge between domains, and help the model understand relationships between sequences of tokens. Specifically, we denote the following elements:

- $\mathbf{i} = (i_u)_{u=1}^m$ denotes the instruction coming from the marginal distribu-

UMAP Projection (Latent Embedding)

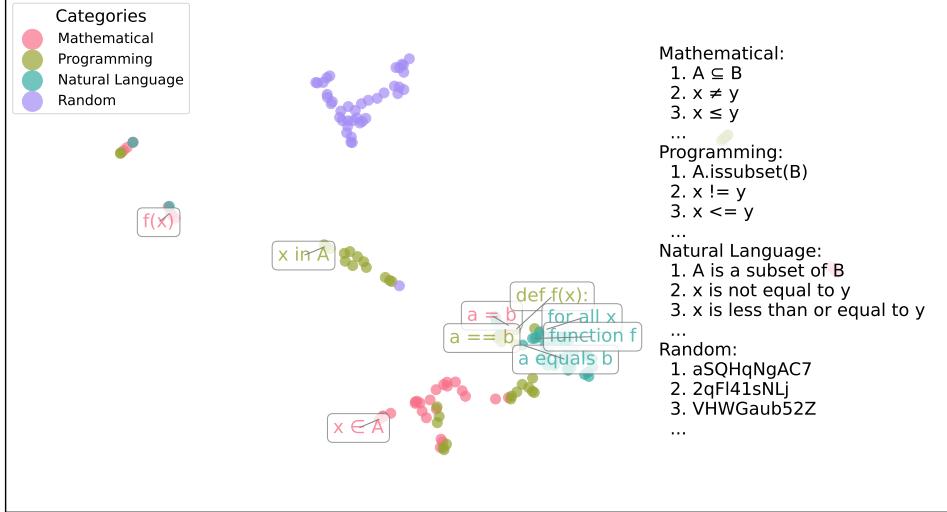


Figure 1.2: UMAP projection from the latent embeddings of four domains before the classification layer of a GPT-Neo 1.3 billion parameters LLM. The domains are *Mathematical*, *Programming*, *Natural Language* and *Random*. Each domain includes 40 samples in their respective domain-specific formulation. We see semantically similar concepts cluster together across domains, highlighting the overlap between domains. The *Random* domain is furthest apart, since the embedded sequences do not share semantic overlap with the other domains.

tion $p_{\mathcal{I}}$, where each i_u represents a token in the instruction sequence.

- $\mathbf{C}_p = \{\mathbf{c}_p^v = [(c_{v,t})_{t=1}^{l_{v_1}}, (y_{v,w})_{w=1}^{h_{v_1}}]\}_{v=1}^{k_1}$ represents source domain examples, where $c_{v,t}$ is the t -th token of the v -th context sequence of length l_{v_1} , and $y_{v,w}$ is the w -th token of the corresponding label sequence of length h_{v_1} , coming from the marginal distribution p_C .
- $\mathbf{C}_q = \{\mathbf{c}_q^v = [(c_{v,t})_{t=1}^{l_{v_2}}, (y_{v,w})_{w=1}^{h_{v_2}}]\}_{v=1}^{k_2}$ represents target domain examples, where $c_{v,t}$ is the t -th token of the v -th context sequence of length l_{v_2} , and $y_{v,w}$ is the w -th token of the corresponding label sequence of length h_{v_2} , coming from the marginal distribution q_C .
- $\mathbf{s} = (s_j)_{j=1}^n$ is an unlabeled query statement coming from the marginal distribution q_S , where each s_j represents a token in the query sequence.

For example, in a sentiment analysis task across different product categories, \mathbf{i} might be "Classify the sentiment as positive or negative:", \mathbf{c}_p could include labeled examples like ("This book was a page-turner!", "very positive") and ("The plot was predictable", "somewhat negative") from the

book domain, \mathbf{c}_q could include ("The battery life is impressive", "extremely positive") and ("The screen cracks easily", "very negative") from the electronics domain, and \mathbf{s} could be an unlabeled review "My new book arrived with a crack on its cover" to classify.

We then formulate the input to the model as $\mathbf{z}^1 = \varphi_{\text{enc}}(\mathbf{i}, \mathbf{c}_p, \mathbf{c}_q, \mathbf{s}; \rho)$, where $\varphi_{\text{enc}} : \mathcal{I}^m \times \mathcal{C}^{k_p} \times \mathcal{C}^{k_q} \times \mathcal{S}^n \rightarrow \mathbb{R}^{j \times d}$ represents the encoding and concatenation function based on the embedding model ρ (see in-context learning approach step 1. and 2.), with k_p and k_q the respective lengths for the context variables and j the total sequence length. The function φ_{enc} utilizes two domains, and outputs a latent embedding for the input of subsequent layers φ_{trans}^l . We omit the sequence length indices for brevity of notation.

Problem Statement The aim is to form non-trivial in-context associations that enable the model to predict $\hat{y} = y \in \mathcal{Y}$, and formulate in-context learning with the principles of UDA that leverages few labeled data pairs from both domains to learn associations and generalize to unlabeled query statements.

By learning from these in-context associations during training, the LLM is conditioned to associate concepts that are related, and therefore can learn domain-invariant latent representations. This process aims to minimize the discrepancy between distributions of source and target components in the latent space. Specifically, for any layer index l , the transformation φ_{trans}^l should map the source component \mathbf{z}_p^{l-1} and the target component \mathbf{z}_q^{l-1} of \mathbf{z}^{l-1} to similar regions in the latent space, such that $\varphi_{\text{trans}}^l(\mathbf{z}_p^{l-1}) \approx \varphi_{\text{trans}}^l(\mathbf{z}_q^{l-1})$.

We obtain \mathbf{z}_p^1 by applying the encoding function to the source context sequences only:

$$\mathbf{z}_p^1 = \varphi_{\text{enc}}(\mathbf{0}, \mathbf{c}_p, \mathbf{0}, \mathbf{0}; \rho), \quad (1.3)$$

where $\mathbf{0}$ represents a placeholder (such as a zero vector or a special masking token) used in place of the instruction, target context, and query components. Similarly, for the target component:

$$\mathbf{z}_q^1 = \varphi_{\text{enc}}(\mathbf{0}, \mathbf{0}, \mathbf{c}_q, \mathbf{0}; \rho). \quad (1.4)$$

This formulation explicitly shows how we isolate the source and target contexts in the encoding process, allowing us to analyze their representations separately in the subsequent transformations.

Implicit Contextual Associations Such associations are often formed implicitly through token sequences that have specific properties attached to them, i.e., the "equals" statement is related to "=" or "the same" which allows for substitutions, and are innate to many domains such as mathematics or programming. Therefore, the models profit from many such contextual examples, which also enables us to evaluate unconventional queries as shown in the following observations.

Let us examine several examples to illustrate the capabilities of two state-of-the-art language models, Claude 3.5 Sonnet and GPT-4 (OpenAI, 2023; Anthropic, 2024), both of which were trained using in-context learning.

Example 2. Consider the following arithmetic query:

1 $\textcircled{9} + \textcircled{2} - \textcircled{2} * \textcircled{5} =$

This query, when interpreted by a model, returns a similar prediction as shown below:

1 Let's solve this step-by-step, following the order of operations (PEMDAS: Parentheses, Exponents, Multiplication/Division, Addition/Subtraction):
2
3 First, let's identify the numbers:
4 $\textcircled{9} = 9$
5 $\textcircled{2} = 2$
6 $\textcircled{5} = 5$
7 Now, our equation looks like this:
8 $9 + 2 - 2 * 5 =$
9 We start with multiplication:
10 $2 * 5 = 10$
11 Now our equation is:
12 $9 + 2 - 10 =$
13 We perform addition and subtraction from left to right:
14 $9 + 2 = 11$
15 $11 - 10 = 1$
16
17 Therefore:
18 $\textcircled{9} + \textcircled{2} - \textcircled{2} * \textcircled{5} = 1$
19 The final answer is 1.

The model predicts the answer 1, which is semantically correct for a human reader. However, it is interesting to observe that the model interprets the symbols $\textcircled{9}$, $\textcircled{2}$, $\textcircled{5}$, and $+$ as numbers 9, 2, 5, and the $+$ operator, respectively. These symbols are entirely different Unicode encodings and are mapped to different embeddings, yet the model successfully associates them with their corresponding mathematical representations.

Let us now explore how this capability extends to other symbols.

Example 3. Consider an operation between two weather symbols:

1 $\text{☀} + \text{🌧} =$

When evaluated, the models return a similar prediction as follows:

1 $\text{☀} + \text{🌧} = \text{🌈}$
2 The combination of a sun symbol (☀) and a rain symbol (🌧) typically represents a rainbow (🌈). This is because rainbows occur in nature when sunlight and rain are present simultaneously ...

In this case, not only are the symbols ☀, 🌧, and 🌈 associated with the words sun, rain, and rainbow, respectively, but we also observe that the + and = operators follow the same semantics as in the arithmetic example above. This showcases a model’s ability to blend different domains — in this case, weather phenomena and basic arithmetic operations — to produce a meaningful interpretation.

However, it is important to note that not every query is parsable, as demonstrated in the following example.

Example 4. Consider a query using the Braille alphabet with numbers and the + operator:

$$1 \ 1 \ + \ \bullet \ =$$

In this case, the Claude model struggles to associate the number 1 with the two dots to infer a semantic operation and predict a number:

-
- 1 To solve this, we need to understand what the symbol : represents.
 - 2 This symbol is a Braille character, but its numerical value isn’t universally defined in the same way as the earlier emoji numbers were. Without additional context about how ...
-

This example illustrates the limitations of the model’s ability to make associations across different symbol systems, especially when the relationship between the symbols is not explicitly defined or commonly used.

Now, to showcase the capabilities of in-context learning, we will add two examples to help the model understand how to interpret the query.

Example 5. Consider the adapted query using Braille alphabet with numbers and the + operator, and two added in-context examples:

$$1 \ 2 \ + \ \bullet \ = \ 3$$

$$2 \ 1 \ + \ \bullet\bullet \ = \ 5$$

3

$$4 \ 1 \ + \ \bullet \ =$$

In this case, both models interpret the context and associate the Unicode encoding with the in-context examples to deduce the correct answer:

1 Let’s approach this step-by-step:

2

3 From the first equation, we can deduce:

4 $2 \ + \ \bullet \ = \ 3$

5 $\bullet \ = \ 3 \ - \ 2 \ = \ 1$

```

6 From the second equation:
7 1 + :: = 5
8 :: = 5 - 1 = 4
9 Now, we need to solve:
10 1 + : = ?
11 We can see a pattern forming. The Braille symbols seem to
    represent numbers:
12 • represents 1
13 : would likely represent 2
14 :: represents 4
15 Therefore, the equation 1 + : would be equivalent to 1 + 2
16
17 So, the answer is:
18 1 + : = 3
19 Would you like me to explain any part of this reasoning in more
    detail?

```

Note that the `::` symbol represents the number 4, not 3, since it is the fourth entry in the encoding table of the Braille sequence.

The last example shows that, with only two provided examples, the models are able to associate the domains of numbers with the Braille alphabet and compute the expected prediction (even though the association for the number 4 for a human reader is non-intuitive).

In practice, these associations are often more subtle, since they occur in permutations of prompts $\mathcal{X} \times \mathcal{Y}$ that have no explicitly separated instruction, context, query statement and label declarations. To illustrate this concept of domain blending within in-context learning, let's examine a final, more complex example.

Example 6. We present the models with a prompt that combines two disparate domains: Python programming and Shakespearean literature. The prompt reads: "Write a simple Python function to check if a file exists in Shakespeare style". This unusual combination challenges the model to seamlessly integrate knowledge from both disjoint domains.

The resulting prediction demonstrates the model's ability to interpret novel in-context associations without additional context samples:

```

1 import os
2
3 def doth_file_exist(file_path):
4     """
5     To exist, or not to exist, that is the question:
6     Whether 'tis nobler in the mind to suffer

```

```

7     ...
8     And lose the name of action.--Soft you now,
9     The fair file!--Nymph, in thy orisons
10    Be all my sins remember'd.
11
12    Args:
13        file_path (str): The path whose existence is to be verified.
14
15    Returns:
16        bool: True if the file exists, False otherwise.
17        """
18        return os.path.exists(file_path)
19
20    file_path = "path/to/your/file.txt"
21    if doth_file_exist(file_path):
22        print("Verily, the file exists!")
23    else:
24        print("Alas, the file does not exist.")

```

The model predicts tokens from both the Python programming and Shakespearean literature distributions, showcasing the model’s ability to blend seemingly disparate domains, even though such combinations are highly unlikely in real-world scenarios.

In the last example, the instruction is defined within the sequence of *Write a ... in ... style*. The source and target domain context is present in an abstract form as *Python programming* and *Shakespeare* respectively. The query statement is implicitly provided during inference, as LLMs typically employ special tokens to delineate user prompts from model-generated responses. Although, the models learned during training to represent both domains independently, due to semantic overlap between tokens, at inference both models are able to interpolate between the domain representations and predict output sequences coming from the joint distribution.

In-Context Associations and Attention To understand the latent representations $\varphi_{\text{trans}}^l(\cdot)$ in an LLM and how they could form domain-invariant features, we also examine the token frequency and embedding distributions, and the self-attention mechanism of modern *Transformer* architectures based on the GPT-Neo model. In Figure 1.3 we show the token frequency per domain and the embeddings after the encoding phase and before the pre-classification latent layer. The first row shows the absolute frequency of tokens per domain, with the y-axis normalized by the 99-percentile of the maximum value obtained across all domains. The x-axis shows only the used unique token IDs across all four domains, concatenated in order, to enable comparison of the token frequencies. The second and third row show the normalized embedding distribution for the encoded tokens after the tokenization and encoding phase and the latent embeddings before the classification layer, respectively. Despite the initial differences in token frequencies

and distributions in the lower layers, the normalization and transformation processes within the LLM architecture result in centered values (around zero) and aligned embedding distributions across domains, setting the premise for achieving domain-invariant representations as illustrated in the third row.

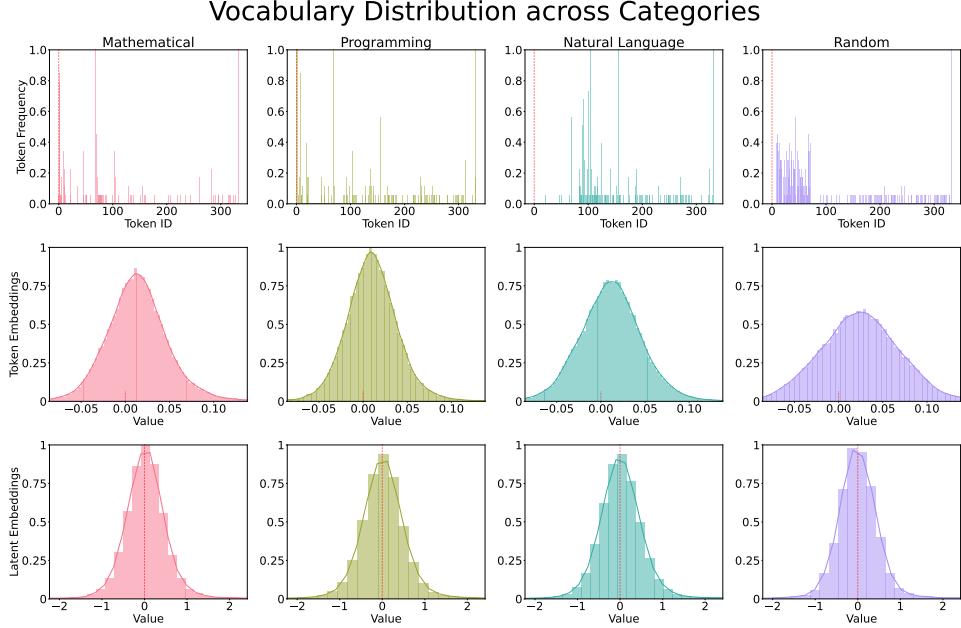


Figure 1.3: We show the distribution of the tokens across four domains, namely *Mathematical*, *Programming*, *Natural Language* and *Random* domain. The first row shows on the y-axis the 99-percentile normalized frequency of tokens. The x-axis shows the used unique tokens across all four domains, concatenated in order. The second and third row show the normalized embedding distribution for the encoded tokens after the tokenization and encoding phase and the latent embeddings before the classification layer, respectively.

The self-attention mechanism is an associative memory (Ramsauer et al., 2020) that computes a scaled dot-product between query (\mathbf{Q}) (unrelated to the previously mentioned query statement of the input sequence), key (\mathbf{K}), and value (\mathbf{V}) matrices. These matrices are linear projections of the input sequence represented as a matrix $\mathbf{z} \in \mathbb{R}^{j \times d}$, where j is the sequence length and d is the embedding dimension:

$$\begin{aligned}\mathbf{Q} &= \mathbf{z}\mathbf{W}_Q, \quad \mathbf{W}_Q \in \mathbb{R}^{d \times d_k}, \\ \mathbf{K} &= \mathbf{z}\mathbf{W}_K, \quad \mathbf{W}_K \in \mathbb{R}^{d \times d_k}, \\ \mathbf{V} &= \mathbf{z}\mathbf{W}_V, \quad \mathbf{W}_V \in \mathbb{R}^{d \times d_v},\end{aligned}\tag{1.5}$$

where \mathbf{W}_Q , \mathbf{W}_K , and \mathbf{W}_V are learnable weight matrices for the queries,

keys, and values, respectively. The matrix multiplication $\mathbf{z}\mathbf{W}$ projects each token embedding in \mathbf{z} to a new space, creating the corresponding \mathbf{Q} , \mathbf{K} , or \mathbf{V} representation. Following Vaswani et al. (2017), we typically set $d_k = d_v = d/h = 64$, where d_k is the dimensionality of the key vectors, and $h = 8$ is the number of attention heads in a multi-head attention setting.

The scaled dot-product is commonly known as attention scores, which we refer to as our transformation function $\varphi_{\text{trans}}(\cdot)$, and is computed as follows:

$$\varphi_{\text{trans}}(\mathbf{z}) := \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}, \quad (1.6)$$

where a scaling factor $\sqrt{d_k}$ is introduced to mitigate the effect of large dot products in high-dimensional spaces. This scaling helps maintain stable gradients during training, particularly for larger values of d_k , and prevents the softmax function from producing small gradients in regions where the input is large in magnitude.

The self-attention function computes a weighted sum of the rows of the value matrix \mathbf{V} (each row being a value vector), with weights derived from the similarity between corresponding rows of the query matrix \mathbf{Q} and columns of the key matrix \mathbf{K}^\top . Here, self-attention refers to the fact that $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ all share the same sequence matrix \mathbf{z} . It's worth noting that in other attention settings, different sequence representations can be used for each component. The resulting attention output $\varphi_{\text{trans}}(\cdot)$ associates information from different parts of the input sequence, capturing the dependencies and relationships between tokens. For networks with multiple layered transformations $\varphi_{\text{trans}}^{l>1}$, the output is usually normalized before passed on to subsequent layers. The respective matrices $\mathbf{W}_Q^l, \mathbf{W}_K^l, \mathbf{W}_V^l$ learn these associations for domain-specific tasks and generalize to the semantics based on the instructions and contextual patterns. This generalization mechanism then later allows the model to extrapolate semantics across diverse domains when provided with few-shot examples, effectively enabling DA through the formation of context-dependent associations in the attention layers.

Domain-Invariance in Language Models In the previous sections, we have seen LLMs able of forming domain-invariant predictions based on our empirical observations, and accredited this to associative patterns learned during training. We now conclude our analysis by defining domain-invariance in an LLM setting, and propose a measurable relationship between semantic similarity of domains and the convergence of their embedded representations. This relationship forms the basis for future work on quantifying the models' capabilities for cross-domain generalization.

We note that in a general case an LLM is a function that maps $f_\theta : \mathcal{P} \rightarrow \mathcal{V}^*$ with parameters $\theta \in \Theta$, which produces a sequence of outputs based on the prompt. This is due to the auto-regressive setting of predicting and

sampling the next token and appending it to the input, which is repeated until the halting criteria such as a stop token terminates the sequence. In this setting, a decoder typically samples according to some sampling strategy (i.e. using a temperature parameter, *beam search*, etc.) to obtain the most likely sequence of token. However, for simplicity the following definition will focus only on a special case where we omit the auto-regressive process and assume that the context $\mathcal{C} \subset \mathcal{X} \times \mathcal{V}$ considers the output space equal to the vocabulary, and the LLM predicts only the next token \mathcal{V} . Furthermore, to define the domain invariance as shown in Example 1, we assume that "True" and "False" are tokens and part of the vocabulary and then formalize the definition as follows:

Let \mathcal{V} be the vocabulary of tokens of an LLM and $\mathcal{P} \subset \mathcal{X} \subset \mathcal{V}^*$ be the prompt space. Define $\mathcal{P} = \mathcal{I}^m \times \mathcal{C}^{k_p} \times \mathcal{C}^{k_q} \times \mathcal{S}^n$ with instruction space $i \in \mathcal{I} \subset \mathcal{X}$, context space $c \in \mathcal{C} \subset \mathcal{X} \times \mathcal{V}$, and query space $s \in \mathcal{S} \subset \mathcal{X}$. Let $p(i, c, s)$ and $q(i, c, s)$ denote the joint distributions for the source and target domains respectively, with p_c and q_c being their respective marginal distributions over the context space. The sequence dimensions m, k_p, k_q, n are each ≥ 1 , the sequence length is $j = m + k_p + k_q + n$, and the first latent dimension is $d_1 \geq 1$.

Now, let an LLM be a function $f_\theta : \mathcal{P} \rightarrow \mathcal{V}$ with parameters $\theta \in \Theta$, with transformation layers $\varphi_{\text{trans}}^l : \mathbb{R}^{j \times d_{l-1}} \rightarrow \mathbb{R}^{j \times d_l}$ and dimensions d_l for $1 < l < L$, and the last transformation layer $\varphi_{\text{trans}}^L : \mathbb{R}^{j \times d_{L-1}} \rightarrow \mathbb{R}^{d_L}$ before the decoder $\varphi_{\text{dec}} : \mathbb{R}^{d_L} \rightarrow \mathcal{V}$, where the output dimension $d_L = |\mathcal{V}| \geq 1$ is equal to the vocabulary size.

Definition 1 (Domain-Invariant Language Models). *We say an LLM has domain-invariant properties if, for all pairs of contextual samples with equivalent semantics, represented by $\mathbf{c}_p \equiv \mathbf{c}_q$, where $\mathbf{c}_p \in \mathbf{C}_p$ comes from the marginal distribution p_c and the semantically equivalent $\mathbf{c}_q \in \mathbf{C}_q$ comes from the marginal distribution q_c , the argmax (greedy sampling decoder) of their outputs from the last transformation layer φ_{trans}^L is equal. Formally:*

$$\forall (\mathbf{z}_p^L, \mathbf{z}_q^L) \in \mathbf{Z}_p^L \times \mathbf{Z}_q^L \text{ with } \mathbf{z}_p^L = \varphi^{1:L}(\mathbf{c}_p), \mathbf{z}_q^L = \varphi^{1:L}(\mathbf{c}_q), \text{ and } \mathbf{c}_p \equiv \mathbf{c}_q : \\ \underset{i \in \{1, \dots, |\mathcal{V}|\}}{\text{argmax}} [\mathbf{z}_p^L]_i = \underset{i \in \{1, \dots, |\mathcal{V}|\}}{\text{argmax}} [\mathbf{z}_q^L]_i, \quad (1.7)$$

where \mathbf{z}_p^L and \mathbf{z}_q^L are the outputs of the last transformation layer derived from \mathbf{c}_p and \mathbf{c}_q respectively, with $\mathbf{z}_p^1 = \varphi_{\text{enc}}(\mathbf{0}, \mathbf{c}_p, \mathbf{0}, \mathbf{0}; \rho)$ and $\mathbf{z}_q^1 = \varphi_{\text{enc}}(\mathbf{0}, \mathbf{0}, \mathbf{c}_q, \mathbf{0}; \rho)$. Here, $\varphi_{\text{enc}} : \mathcal{P} \rightarrow \mathbb{R}^{j \times d_1}$ is the embedding and concatenation function, $\rho : \mathcal{V} \rightarrow \mathbb{R}^{d_1}$ is an embedding function, and $\mathbf{0}$ denotes the mask tokens for the components. The latent representations $\mathbf{Z}_p^l = \{\mathbf{z}_{p,u}^l = \varphi^{1:l}(\mathbf{c}_{p,u})\}_{u=1}^{h_p}$ and $\mathbf{Z}_q^l = \{\mathbf{z}_{q,u}^l = \varphi^{1:l}(\mathbf{c}_{q,u})\}_{u=1}^{h_q}$ come from the respective domains with $h_p \geq 1$ and $h_q \geq 1$ samples, by applying the composition function up to layer index l , where $\varphi^{1:l} = \varphi_{\text{trans}}^l \circ \varphi_{\text{trans}}^{l-1} \circ \dots \circ \varphi_{\text{trans}}^2 \circ \varphi_{\text{trans}}^1 \circ \varphi_{\text{enc}}$.

The semantic equivalence symbol \equiv refers to selecting consistently equivalent samples from the respective sets, e.g., the number "1" and string "one", which are then pairwise embedded, transformed, and compared. We then stipulate that for all these pairwise latent representations, their argmax must be equal, ensuring they map to the same next token. The determination of semantic equivalence is incumbent upon the modeling process of the distributions and is an integral part of the training process design. This definition illustrates how the phenomenon of cross-domain generalization in LLMs can emerge when a sufficiently large and diverse training data corpus is used. The definition of domain-invariant properties in LLMs and the semantic equivalence can be viewed as a language-based extension of the *Curry–Howard correspondence*, which originally establishes a direct relationship between computer programs and mathematical proofs (Howard, 1980).

Domain-Invariant Associations While Definition (1) provides a strict criterion for domain invariance, in practice, we often have only a degree of similarity between domain representations, especially in an auto-regressive generative process. To quantify this similarity, we introduce a semantic distance measure that captures the closeness of representations of the model's layers, and provide an ϵ -close domain-invariance approximation.

Let \mathcal{F}_Θ be the class of LLMs with ϵ -close domain-invariant associations:

$$\mathcal{F}_\Theta = \{f_\theta : \mathcal{P} \rightarrow \mathcal{Y} \mid \theta \in \Theta, f_\theta \text{ with } \epsilon\text{-close domain-invariant associations}\}$$

For a model $f_\theta \in \mathcal{F}_\Theta$, we define the semantic distance between domains represented by \mathbf{C}_p and \mathbf{C}_q as:

$$d_{\text{sem},\theta}(\mathbf{C}_p, \mathbf{C}_q) = \min_{1 \leq l \leq L} \max_{\substack{\mathbf{z}_p^l \in \mathbf{Z}_p^l \\ \mathbf{z}_q^l \in \mathbf{Z}_q^l}} \|\mathbf{z}_p^l - \mathbf{z}_q^l\|_2, \quad (1.8)$$

where \mathbf{z}_p^l and \mathbf{z}_q^l are the input representations for the respective layer index l based on \mathbf{C}_p and \mathbf{C}_q following Definition (1).

We say that two domains with contextual samples \mathbf{C}_p and \mathbf{C}_q have ϵ -close domain-invariant associations if $d_{\text{sem},\theta}(\mathbf{C}_p, \mathbf{C}_q) \leq \epsilon$, thereby setting an upper bound on the difference in representations measured by $d_{\text{sem},\theta}$. The semantic distance provides an approximation of the degree of domain invariance formed through associations, with smaller distances indicating greater similarity between domain representations. It also captures the idea that as the distance between the transformed representations of samples from two domains decreases, the semantic distance between these domains also decreases. The semantic distance $d_{\text{sem},\theta}$ therefore quantifies this relationship and can be applied to a variety of domains, considering factors such as semantic overlap, structural similarity, and task-specific relevance.

In practice, the argmax is not applied; instead, more sophisticated auto-regressive sampling techniques are used. Therefore, it suffices that the last

layer representations \mathbf{z}_p^L and \mathbf{z}_q^L are close enough to semantically map to a similar distribution of representations. This is supported by empirical evidence that suggests the existence of semantic *meaning* spaces or a *semantic law of motion*, as described by Stephen Wolfram (Wolfram, 2023). The ϵ -close domain-invariant association offers a practical relaxation when perfect invariance is unattainable. This approach provides a quantifiable measure of similarity between domain representations. In an auto-regressive setting, it can leverage the semantic *closeness* between tokens to guide the semantic space towards the desired outcome.

Furthermore, if we assume that all transformation functions $\varphi_{\text{trans}}^\pi$ of the model f_θ with $\pi > l$ are Lipschitz continuous with Lipschitz constant K_π , then for all $\mathbf{z}_p^L, \mathbf{z}_q^L$ based on \mathbf{C}_p and \mathbf{C}_q coming from the respective marginal distributions, the upper bound on the difference in representations in the last layer is given by:

$$\forall (\mathbf{z}_p^L, \mathbf{z}_q^L) \in \mathbf{Z}_p^L \times \mathbf{Z}_q^L : \|\mathbf{z}_p^L - \mathbf{z}_q^L\|_2 \leq \epsilon \prod_{\pi=l+1}^L K_\pi. \quad (1.9)$$

See also the Lipschitz constant of self-attention (Kim et al., 2021).

Hypothesis on Domain-Invariant Associations The described setting now provides a framework to formulize the relationship between in-context learning and domain-invariant representation learning in LLMs. Based on this, we propose the following hypothesis:

Hypothesis 1. *For an LLM $f_\theta \in \mathcal{F}_\Theta$ with ϵ -close domain-invariant associations the following applies: Given any pair of samples from domains \mathbf{C}_p (source) and \mathbf{C}_q (target) with their respective latent representations, if their semantic distance $d_{\text{sem},\theta}(\mathbf{C}_p, \mathbf{C}_q) \leq \epsilon$ for some $\epsilon \geq 0$, then:*

$$\Delta(q_{Y|X}, q_{f_\theta(X)}) \leq \Delta(p_{Y|X}, p_{f_\theta(X)}) + \delta(\epsilon), \quad (1.10)$$

where:

- X is a random variable taking values in the input space \mathcal{X} ,
- Y is a random variable taking values in the output space \mathcal{Y} ,
- Δ is a discrepancy measure between probability distributions,
- $q_{Y|X}$ and $p_{Y|X}$ are the conditional distributions of Y given X for the target and source domains respectively,
- $q_{f_\theta(X)}$ and $p_{f_\theta(X)}$ are the distributions of the model's outputs $f_\theta(X)$ in the target and source domains respectively,
- $\delta(\epsilon)$ is a function that approaches zero as ϵ approaches zero.

This hypothesis formalizes the concept that when the semantic distance between samples of a source domain and a target domain is small, the model’s discrepancy between its output distribution and the true output distribution on the target domain is bounded by its discrepancy on the source domain plus a small error term. This error term decreases as the semantic distance decreases.

In turn, this also highlights the model’s limitations: when a query extrapolates beyond the model’s domain capabilities, we obtain meaningless and potentially *hallucinated* results (Huang et al., 2023, 2024), as the model attempts to fill knowledge gaps with its existing, but irrelevant, capabilities. To some degree, this limitation is often obfuscated by blending the generative process with reinforcement learning fine-tuning from human feedback (Ouyang et al., 2022; Kaufmann et al., 2024). However, recent publications have begun to critically examine these aspects (Hicks et al., 2024). Despite these and other open challenges, the proposed intersection of domain-invariant learning and LLMs, along with the utilization of their operator-like patterns, offers potential for grounding prompt engineering in a formal framework and advancing NeSy approaches, particularly in bridging symbolic reasoning with the adaptive capabilities of neural networks.

1.4 A Neuro-Symbolic Perspective on Large Language Models

Building on the in-context learning capabilities of large language models (LLMs) and their ability to interpret in-context associations, we can define task-specific prompts that interpret various instructions and contextual patterns, and combine them in a neuro-symbolic (NeSy) approach. These prompts leverage the operator-like patterns learned by the model, such as `equals`, `is like`, and `from ... to`, to solve task-specific requirements without the necessity to update model parameters. This NeSy approach connects various domains via a language-based interface, utilizing the model’s capacity to recognize and apply semantic patterns across domains.

We therefore consider LLMs suitable as *semantic parsers* (Dinu et al., 2024a) that can take in various expressions (which are operation descriptions defining how to manipulate strings), interpret them using learned associations, and predict the most probable statistical outcome. These expressions can be composed into complex functions, forming a multi-step generative process that leverages a model’s domain-invariant capabilities.

The language-based interface is well-suited for a NeSy approach to interface with classical symbolic methods. We can interchange between solvers and various tools that can verify our predicted outcomes, taking advantage of the model’s ability to bridge symbolic and sub-symbolic paradigms. This NeSy approach allows for more structured and rule-based approaches, where

the model not only predicts statistical associations but also adheres to defined symbolic rules and expressions. This enhances our ability to verify results by combining the strengths of both neural and symbolic components.

SymbolicAI Framework In our work Dinu et al. (2024a), we propose a neuro-symbolic approach called *SymbolicAI*. The framework is designed to bridge the gap between symbolic reasoning and neural network capabilities, particularly with the help of LLMs.

At its core, SymbolicAI performs mappings that take in various symbols and produce symbols as output, where symbols are sequences of tokens drawn from a finite vocabulary \mathcal{V} . These symbols can be of any length and structure, allowing for great flexibility in representation. The SymbolicAI framework uses these symbols to form more complex expressions, and uses in-context learning to adapt the generative process based on the given instructions, contexts and query statements. A key concept in this framework is the idea of expressions and their polymorphic contexts. An expression is an operation that defines transformations of symbols to create new symbols. This can be as simple as concatenating two symbols or as complex as applying mathematical operations to them. When this operation is evaluated, it produces a new symbol based on specific rules. For example, one rule might define how to perform arithmetic operations on symbols representing numbers and strings. To illustrate this, consider an example where we want to combine the symbols "eight" and "8". We could define a rule that interprets this as addition, resulting in a new symbol "16" as the prediction.

The framework is grounded in formal language theory and grammars, building on the work of linguists and computer scientists Chomsky and Newell (Chomsky, 1959; Newell and Simon, 1972; Hays, 2003). Our NeSy approach uses symbolic methods to define sets of rules for relationships between objects, evoke causal structures, and perform formal evaluations. This allows us to define task-specific operations that query LLMs to guide their predictions and verify the results with other tools and solvers. An LLM can interpret instructions to form associative patterns with the respective data and return a statistically probable answer. Our framework then combines such operations into more complex expressions. These expressions guide the generative process of the LLM and enable efficient domain-specific adaptations even under continual domain changes without the necessity of parameter updates. SymbolicAI thus provides a systematic way to represent and manipulate symbolic information and control generative processes. Our NeSy approach is well suited for enabling enhanced reasoning capabilities and building broad AI systems that verify the results of generated content such as that obtained through LLMs, while still leveraging their powerful language understanding and generalization abilities. This can lead to more robust and versatile AI systems capable of handling a wider range of tasks, setting the

foundation for automated AI workflows. Moreover, the symbolic component of NeSy systems can address challenges of computational irreducibility often encountered in complex tasks, allowing for more transparent, interpretable, and discernible results in scenarios where purely neural approaches might lead to unpredictable or difficult-to-verify outcomes.

Computational Graphs, Explainability and Evaluation At the core of the proposed framework is its ability to construct complex computational graphs with symbolic and sub-symbolic approaches that use various expressions. These computational graphs guide the generative process through the in-context learning capabilities of LLMs via functional methods that build, evaluate and execute prompt instructions and contextual patterns. These operations form more complex behaviors, adapting the application behavior based on the data and context.

As a key contribution, we propose a quality measure based on the Fréchet distance and its empirical *VERTEX* score that evaluates the performance of computational graphs with respect to reference distributions. Additionally, we propose a benchmark to evaluate various LLMs suitable for this NeSy approach.

1.5 List of Publications

In this section, the publications that are directly and more loosely connected to this work are listed, with short explanations of the publications and my roles in them. A “*” character indicates shared first authorship.

Directly connected publications. The following publications are directly connected to this work. All publications are full conference papers and have been peer-reviewed.

1. (Dinu et al., 2024a): **M.-C. Dinu**, C. Leoveanu-Condrei, M. Holzleitner, W. Zellinger, S. Hochreiter. SymbolicAI: A framework for logic-based approaches combining generative models and solvers. In *Third Conference on Lifelong Learning Agents*, PLMR, 2024.
2. (Dinu et al., 2024b) **M.-C. Dinu**, C. Leoveanu-Condrei, M. Holzleitner, W. Zellinger, S. Hochreiter. SymbolicAI: A framework for logic-based approaches combining generative models and solvers. GenAI4DM Workshop at *The Twelfth International Conference on Learning Representations*, 2024.
3. (Dinu et al., 2023): **M.-C. Dinu**, M. Holzleitner, M. Beck, H. D. Nguyen, A. Huber, H. Eghbal-zadeh, B. A. Moser, S. Pereverzyev, S. Hochreiter, W. Zellinger. Addressing parameter choice issues in

unsupervised domain adaptation by aggregation. In *International Conference on Learning Representations*, 2023.

4. (Zellinger et al., 2021): W. Zellinger, N. Shepeleva, **M.-C. Dinu**, H. Eghbal-zadeh, H. D. Nguyen, B. Nessler, S. Pereverzyev, B. A. Moser. The balancing principle for parameter choice in distance-regularized domain adaptation. In *Advances in Neural Information Processing Systems*, 2021.

(Dinu et al., 2023) is the main publication of our work on parameter choice issues for unsupervised domain adaptation. My main contributions to this work are the implementation of the method, writing of the paper, and development of the benchmark and baseline models. This includes the design and implementation of the IWA model and its training for the domain adaptation experiments, the realization of the baselines, as well as the investigation of approaches for evaluating the aggregation correlation coefficients of the IWA models. (Dinu et al., 2024a) is my most comprehensive publication and focuses on NeSy approaches. My contributions are the conception and implementation of the main approach, as well as the writing of the paper, development of the framework, the proposal and implementation of the VERTEX score, the creation of the benchmark and main parts of the formal derivations.

Loosely connected publications. The following publications are loosely connected to this work. All publications in this section have been peer-reviewed, except for Patel et al. (2024) which is still under review at NeurIPS 2024.

1. (Patel et al., 2024): A. Patel, M. Hofmarcher, C. Leoveanu-Condrei, **M.-C. Dinu**, C. Callison-Burch, S. Hochreiter. Large Language Models Can Self-Improve At Web Agent Tasks. *Advances in Neural Information Processing Systems* (under review), 2024.
2. (Dinu* et al., 2022): **M.-C. Dinu***, M. Hofmarcher*, V. P. Patil, M. Dorfer, P. M. Blies, J. Brandstetter, J. A. Arjona-Medina, S. Hochreiter. XAI and Strategy Extraction via Reward Redistribution. *International Workshop on Extending Explainable AI Beyond Deep Models and Classifiers*. Springer International Publishing, page 177-205, 2022.
3. (Schweighofer* et al., 2022): K. Schweighofer*, A. Radler*, **M.-C. Dinu***, M. Hofmarcher, V. Prakash Patil, A. Bitto-Nemling, H. Eghbal-zadeh, S. Hochreiter. A dataset perspective on offline reinforcement learning. *First Conference on Lifelong Learning Agents*, PMLR, page 470-517, 2022.

4. (Patil* et al., 2022): V. P. Patil*, M. Hofmarcher*, **M.-C. Dinu**, M. Dorfer, P. M. Blies, J. Brandstetter, J. A. Arjona-Medina, S. Hochreiter. Align-rudder: Learning from few demonstrations by reward redistribution. Proceedings of the 39th International Conference on Machine Learning, volume 39, PMLR, page 162:17531-17572, 2022.
5. (Steinparz et al., 2022): C. A. Steinparz, T. Schmied, F. Paischer, **M.-C. Dinu**, V. P. Patil, A. Bitto-Nemling, H. Eghbal-zadeh, S. Hochreiter. Reactive exploration to cope with non-stationarity in lifelong reinforcement learning. First Conference on Lifelong Learning Agents, PMLR, page 441-469, 2022.
6. (Siripurapu et al., 2022): R. Siripurapu, V. P. Patil, K. Schweighofer, **M.-C. Dinu**, T. Schmied, L. E. F. Diez, M. Holzleitner, H. Eghbal-Zadeh, M. K. Kopp, S. Hochreiter. InfODist: Online distillation with Informative rewards improves generalization in Curriculum Learning. Deep Reinforcement Learning Workshop NeurIPS, 2022.
7. (Schweighofer et al., 2021): K. Schweighofer, M. Hofmarcher, **M.-C. Dinu**, P. Renz, A. Bitto-Nemling, V. Patil, S. Hochreiter. Understanding the effects of dataset characteristics on offline reinforcement learning. Deep RL Workshop NeurIPS, 2021.
8. (Holzleitner et al., 2019): M. Holzleitner, J. A. Arjona-Medina, **M.-C. Dinu**, A. Vall, L. Gruber, S. Hochreiter. A Two Time-Scale Update Rule Ensuring Convergence of Episodic Reinforcement Learning Algorithms at the Example of RUDDER. NeurIPS Optimization Foundations for Reinforcement Learning Workshop, 2019.

Chapter 2

Selected Publications

In this chapter, our main publications of this cumulative thesis are presented: (Zellinger et al., 2021) in Section 2.1, (Dinu et al., 2023) in Section 2.2, and (Dinu et al., 2024b) in Section 2.3. The full list of publications and my contributions are discussed in Section 1.5.

The papers shown in this chapter are shrunk to fit the page layout and otherwise reprinted as originally published with permission from the corresponding publishers. Copyright for the original publications is held by the respective copyright holders. For the full supplemental materials of (Zellinger et al., 2021), (Dinu et al., 2023) and (Dinu et al., 2024a) please refer to the respective papers at `proceedings.neurips.cc`, and `proceedings.mlr.press`.

2.1 The balancing principle for parameter choice in distance-regularized domain adaptation

The balancing principle for parameter choice in distance-regularized domain adaptation

Werner Zellinger^{1,*} Natalia Shepeleva¹ Marius-Constantin Dinu^{2,3}

Hamid Eghbal-zadeh^{4,5} Duc Hoan Nguyen⁶ Bernhard Nessler²

Sergei V. Pereverzyev⁶ Bernhard A. Moser¹

¹Software Competence Center Hagenberg GmbH

²Institute for Machine Learning, Johannes Kepler University Linz

³Dynatrace Research

⁴Institute of Computational Perception, Johannes Kepler University Linz

⁵LIT AI Lab, Johannes Kepler University Linz

⁶Johann Radon Institute for Computational and Applied Mathematics, Austrian Academy of Sciences

*werner.zellinger@scch.at

Abstract

We address the unsolved algorithm design problem of choosing a justified regularization parameter in unsupervised domain adaptation. This problem is intriguing as no labels are available in the target domain. Our approach starts with the observation that the widely-used method of minimizing the source error, penalized by a distance measure between source and target feature representations, shares characteristics with regularized ill-posed inverse problems. Regularization parameters in inverse problems are optimally chosen by the fundamental *principle of balancing* approximation and sampling errors. We use this principle to balance learning errors and domain distance in a target error bound. As a result, we obtain a theoretically justified rule for the choice of the regularization parameter. In contrast to the state of the art, our approach allows source and target distributions with disjoint supports. An empirical comparative study on benchmark datasets underpins the performance of our approach.

1 Introduction

Domain adaptation uses the knowledge in a *source* domain to improve the performance of an algorithm on a related *target* domain [1]. In particular, domain adaptation tackles domain shifts in machine learning applications: Medical diagnostic systems should be adapted to new physical human variations; Industrial quality inspection systems should be accurate for new products; Self-driving cars should be able to adapt to new geographical environments and weather conditions. In this work, we focus on *unsupervised domain adaptation* where labels are only available in the source domain.

There are mainly two types of approaches for unsupervised domain adaptation: *importance weighting* [2, 3, 4, 5, 6, 7] and *feature representation learning* [8, 9, 10, 11, 12, 13]. In this work, we focus on feature representation learning which goes beyond classical importance weighting by allowing a target distribution with support outside of the source distribution. The core idea behind feature representation learning approaches is to map the data into a new feature space where the source and target

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

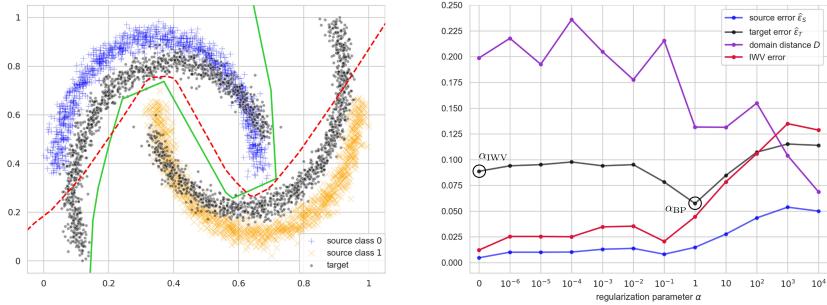


Figure 1: Unsupervised domain adaptation on Transformed Moons. Left: Target data (black dots) partially outside of the support of the source data (blue +, orange \times). The common assumption of bounded density ratio is violated in large regions. In contrast, all our assumptions are satisfied. Our method identifies the best parameter of a domain adaptation algorithm [16] (green solid) which improves training on source data only (red dashed). Right: Regularization parameter (x-axis) which penalizes a distance [25] (purple) leading to models with different source error (blue) and target error (black). Importance weighted validation (IWV) shows the smallest error (red) for models without domain adaptation ($\alpha_{IWV} = 0$). In contrast, our approach identifies the optimal parameter ($\alpha_{BP} = 1$).

data representations appear similar, and where enough information is preserved for prediction [14]. The similarity is often realized by *regularization using distance measures* between source and target representations [15, 16, 17, 18, 19, 20, 21]. However, the performance of such methods crucially depends on the choice of the regularization parameter which penalizes the distance. The problem we investigate in this work is to choose this parameter, which is sophisticated without any target labels.

While remarkable theoretical results have been achieved which quantify the generalization ability of domain adaptation models [8, 22, 19, 20, 21], the choice of the regularization parameter which is crucial for finding such models has not systematically been addressed. Even though some parameter choice strategies exist, they are either purely heuristically driven or very limited by their assumptions [23]. Typical approaches are fixing the regularization parameters [12], minimizing the source error [16], balancing the source error and a distance [17], multiplying a fixed weighting parameter (e.g. 1 in [16]) by a heuristic schedule value that increases during training, or, (importance) weighting the input samples by the ratio between target and source density [3, 24, 23]. One common problem shared among all these approaches is that they all can fail if the density ratio is unbounded. Such unbounded density ratio is typical for many of the high dimensional problems considered in machine learning [19], e.g. see Figure 1. Besides the aforementioned issues, the lack of principled strategies for parameter choice causes misinterpretations in the ranking of domain adaptation methods which are traditionally compared by performance, while often relying on different parameter choice strategies.

In this work, we propose a principled method for choosing distance-penalizing parameters of feature representation learning approaches for unsupervised domain adaptation. Our approach starts with the observation that the distance-regularization setting of domain adaptation shares characteristics with regularized ill-posed inverse problems (see Table 1). In inverse problems, the regularization parameter can be optimally chosen by the fundamental *balancing principle* which optimizes an approximation-sample (bias-variance) trade-off [26, 27, 28]. We apply this principle for balancing *domain distance* and *learning errors* of target error bounds. In particular, we approach the problem of non-computable terms in the target error bound by a new algorithmic criterion for approximating the value of balance. We call our method the Balancing Principle for Domain Adaptation (BPDA).

The BPDA is general in the sense that it can be applied based on different target error bounds, e.g. on [8, 22, 19, 20, 21]. To the best of our knowledge, the BPDA is the first principled method for parameter choice in unsupervised domain adaptation that allows an unbounded ratio between target and source density. We provide a bound on the generalization error of the best model corresponding to the parameter chosen by the BPDA. Finally, we empirically investigate the behavior of the BPDA based on two target error bounds, different domain adaptation methods and benchmark datasets. Our

results show that the BPDA outperforms or is on par with the state of the art on the problem of choosing the regularization parameter, on several domain adaptation methods; applied on different datasets.

2 Summary of results

Notation Let $\mathcal{X} \subset \mathbb{R}^n$ be an *input space* and \mathcal{Y} be a discrete *label space*. Following the classical setting of unsupervised domain adaptation [25], we consider two datasets: A *source dataset* $(\mathbf{x}, \mathbf{y}) = ((x_1, l_S(x_1)), \dots, (x_s, l_S(x_s))) \in (\mathcal{X} \times \mathcal{Y})^s$ with inputs x_1, \dots, x_s independently drawn according to some source distribution (Borel probability measure) p_S on \mathcal{X} and labeled according to some *labeling function*¹ $l_S : \mathcal{X} \rightarrow \mathcal{Y}$, and, an unlabeled *target dataset* $\mathbf{x}' = (x'_1, \dots, x'_t) \in \mathcal{X}^t$ with elements independently drawn according to some target distribution p_T on \mathcal{X} . Throughout this work, we focus on loss functions $L : \mathcal{Y} \times \mathcal{Y} \rightarrow [0, \infty)$ which satisfy $L(y, y) = 0$. For example consider the 0-1 loss $L(y_1, y_2) := \mathbb{1}[y_1 \neq y_2]$, where $\mathbb{1}[P]$ is 1 iff the predicate P is true and 0 otherwise, and the quadratic loss function $L(y_1, y_2) := |y_1 - y_2|^2$. We denote the source error by $\varepsilon_S(f) = \varepsilon_S(f, l_S)$ with *cross-error* defined as $\varepsilon_S(f, g) := \mathbb{E}_{x \sim p_S}[L(f(x), g(x))]$ and its empirical sample estimate by $\hat{\varepsilon}_S(f) = \hat{\varepsilon}_S(f, l_S)$ with $\hat{\varepsilon}_S(f, g) := \sum_{i=1}^s L(f(x_i), g(x_i))$. We denote the analogously defined target error by $\varepsilon_T(f)$, target cross-error by $\varepsilon_T(f, g)$ and its empirical sample estimate by $\hat{\varepsilon}_T(f)$ with empirical cross-error $\hat{\varepsilon}_T(f, g)$. Throughout this work, we focus on target cross-errors $\varepsilon_T(f, g)$ which satisfy the triangle inequality.

Learning setup In this work, we focus on feature representation learning algorithms for domain adaptation. These approaches aim at finding two learning models: A *representation mapping* $\phi \in \Phi \subset \{\phi : \mathcal{X} \rightarrow \mathcal{R}\}$ into some representation space $\mathcal{R} \subset \mathbb{R}^m$ and a *classifier* $g \in \mathcal{G} \subset \{g : \mathcal{R} \rightarrow \mathcal{Y}\}$. Loosely speaking, the aim is to find a mapping ϕ under which the source representations $\phi(\mathbf{x}) := (\phi(x_1), \dots, \phi(x_s))$ and the target representations $\phi(\mathbf{x}') := (\phi(x'_1), \dots, \phi(x'_t))$ appear similar, and, at the same time, enough information is preserved for prediction [14] by $g(x)$. A common approach to realize this aim is to solve the following objective function [19]

$$\min_{g \in \mathcal{G}, \phi \in \Phi} \hat{\varepsilon}_S(g \circ \phi) + \alpha \cdot d(\phi(\mathbf{x}), \phi(\mathbf{x}')) \quad (1)$$

where d is a distance measure between source and target representations and $\alpha \in [0, \infty)$ is a parameter². Good choices for d in Eq. (1) have been identified to be the Wasserstein distance [29, 30], the Maximum Mean Discrepancy [31, 32], moment distances [17, 33, 18, 34, 35, 36], adversarially learned distances [16, 37] and other measures of divergence [38, 39, 19, 20].

Problem For some $\alpha \in [0, \infty)$, let $g_\alpha \circ \phi_\alpha$ denote the minimizer of Eq. (1). Given an increasing sequence of parameters $\alpha_1, \dots, \alpha_w \in [0, \infty)$ with $\alpha_1 = 0$, the problem studied in this work is to choose the parameter α in the sequence $\alpha_1, \dots, \alpha_w$ with the lowest target error $\varepsilon_T(g_\alpha \circ \phi_\alpha)$.

Approach Our approach consists in minimizing a target error bound which satisfies the form

$$\varepsilon_T(g_\alpha \circ \phi_\alpha) \leq D(\alpha) + E(\alpha) \quad (2)$$

where $D(\alpha)$ gives a notion of *domain distance* (cf. [25]) by quantifying a distance between source and target data representations and $E(\alpha)$ comprises different *learning errors*. We assume that $E(\alpha)$ is bounded by some constant $B > 0$. The general form in Eq. (2) is satisfied by many error bounds [8, 22, 20, 21, 34] which all can be taken as a basis for our approach (more detailed examples are provided in Section 3 and Section 4). One problem that complicates the minimization of these error bounds is that all of them contain terms that are not computable due to the lack of target data. That is, $E(\alpha)$ cannot be directly estimated. The BPDA overcomes this problem by a new criterion for estimating the value of balance between the normalized terms $E(\alpha)$ and $D(\alpha)$. The BPDA is detailed in Algorithm 1.

Properties of Algorithm 1 The BPDA has the following striking properties.

- The BPDA is a general procedure which can be instantiated by any error bound of the form in Eq. (2). See Section 4 and Section 5 for its application based on two different target error bounds [25, 20].

¹For simplicity, we use labeling functions instead of the more general concept of conditional distributions.

²For simplicity we omit further regularization of ϕ and g .

- In contrast to state-of-the-art methods, the BPDA does not assume a target labeling function l_T that is equal to the source labeling function l_S (covariate-shift assumption) and it does not assume that the ratio between target and source density is bounded (sample selection bias assumption). See the supplementary material for a discussion of covariate-shift violations.
- The learning model $g_{\alpha_{\text{BP}}} \circ \phi_{\alpha_{\text{BP}}}$ identified by the BPDA satisfies a generalization bound, see Section 4. If the learning errors term $E(\alpha)$ is non-decreasing, then the target error of $g_{\alpha_{\text{BP}}} \circ \phi_{\alpha_{\text{BP}}}$ is only a constant factor away from the minimum $\min_{\alpha \in [0, \infty)} D(\alpha) + E(\alpha)$ of the instantiation bound in Eq. (2).

Algorithm 1: Balancing principle for domain adaptation (BPDA)

Input : Increasing sequence of parameters $\alpha_1, \dots, \alpha_w \in [0, \infty)$ with $\alpha_1 = 0$ and minimizers $f_1 := g_{\alpha_1} \circ \phi_{\alpha_1}, \dots, f_w := g_{\alpha_w} \circ \phi_{\alpha_w}$ of Eq. (1).

Output : Parameter α_{BP} solving the problem above.

Initialization: $S = \{\}$

for $i = 1, \dots, w$ **do**

| Compute empirical cross-error $\hat{\varepsilon}_T(f_i, f_j)$ and domain distance $D(\alpha_j)$ for all $j = 1, \dots, i - 1$.

| **if** $\hat{\varepsilon}_T(f_i, f_j) \leq D(\alpha_j) \left(2 + \frac{2B}{D(0)}\right)$ **for all** $j = 1, \dots, i - 1$ **then**

| | $S := S \cup \{\alpha_i\}$

| **end**

end

return : $\alpha_{\text{BP}} := \max S$

Besides the properties above, the BPDA outperforms or is on par with the state of the art, on the problem of choosing the regularization parameter, on several domain adaptation methods; applied on different datasets, see Section 5.

Related work Approaches which follow Eq. (1) are sometimes interpreted as learning *domain-invariant* representation. Note that the *minimization* of $d(\phi(\mathbf{x}), \phi(\mathbf{x}'))$ in Eq. (1) to achieve $\phi(\mathbf{x}) = \phi(\mathbf{x}')$ differs from the conception as *regularization* [40, 41]. In fact, minimization of the distance means unjustified over-penalization which might lead to deteriorated performance [19]. Interestingly, our interpretation as regularization problem opens up a powerful toolbox of mathematical techniques. Our approach takes up the technique of balancing stability and approximation in regularized ill-posed inverse problems. The balancing principle has its origins in [26] devoted to nonparametric regression estimation and has been introduced in the context of ill-posed problems [27] and in supervised learning with kernels [42]. Following this line of research, we propose to apply the mathematical techniques underlying the balancing principle in the context of domain adaptation. The most related principled parameter choice methods in the context of unsupervised domain adaptation are importance-weighted cross-validation [3] and its extensions [24, 23]. In contrast to these methods, our method is not restricted by the assumption of a bounded ratio between target and source density. One empirically driven method which is related to ours is [17, 43] which aims at balancing the source error and a distance between source and target data representations. However, this method is not theoretically justified as it ignores the minimal combined error of a classifier on representations as defined in [8]. Nevertheless, if the minimal combined error is negligibly small, our results provide a theoretical explanation of the principles underlying the success of [17, 43]. Our method relies on bounds on the target error such as e.g. [8, 22, 19, 20, 21, 34].

3 Preliminaries

Sampling error bound Throughout this work, we generically denote an upper bound on the sampling error by $\eta_{t, \mathcal{G}, \delta} \in [0, \infty)$, which is assumed to hold true with probability at least $1 - \delta$:

$$|\varepsilon_T(f, g) - \hat{\varepsilon}_T(f, g)| \leq \eta_{t, \mathcal{G}, \delta} \quad (3)$$

The bound $\eta_{t, \mathcal{G}, \delta}$ depends on the sample size t , the function class \mathcal{G} and the constant δ , and, it is assumed to satisfy $\eta_{t, \mathcal{G}, \delta} \rightarrow 0$ for $t \rightarrow \infty$.

Two accompanying target error bounds In the seminal works [8, 25] binary classification $\mathcal{Y} := \{0, 1\}$ with 0-1 loss is considered and it is shown that the following bound holds for all symmetric function classes \mathcal{G} ($g \in \mathcal{G} \implies 1 - g \in \mathcal{G}$), $\phi \in \Phi$, $g \in \mathcal{G}$ and datasets of equal size $s = t$ with probability at least $1 - \delta$:

$$\varepsilon_T(g \circ \phi) \leq \varepsilon_S(g \circ \phi) + \lambda_{\mathcal{G}}(\phi) + \widehat{d}_{\mathcal{G}\Delta\mathcal{G}}(\phi(\mathbf{x}), \phi(\mathbf{x}')) + \eta_{t,\mathcal{G},\delta} \quad (4)$$

where

$$\widehat{d}_{\mathcal{G}\Delta\mathcal{G}}(\mathbf{x}, \mathbf{x}') := 2 \left(1 - \min_{f, f' \in \mathcal{G}} \left[\frac{1}{s} \sum_{i=1}^s \mathbb{1}[f(x_i) = f'(x_i)] + \frac{1}{s} \sum_{i=1}^s \mathbb{1}[f(x'_i) \neq f'(x'_i)] \right] \right) \quad (5)$$

is the empirical $\mathcal{G}\Delta\mathcal{G}$ -divergence [44, 25] and $\lambda_{\mathcal{G}}(\phi) := \inf_{f \in \mathcal{G}} (\varepsilon_S(f \circ \phi) + \varepsilon_T(f \circ \phi))$ is the minimum possible *combined error* determined by the application of \mathcal{G} . Using Eq. (3), the source error can be further upper bounded by the empirical source error. However, the term $\lambda_{\mathcal{G}}(\phi)$ cannot be estimated based on given datasets as it depends on the unknown labeling function l_T .

In [20], the bound in Eq. (4) is generalized to multiple classes $\mathcal{Y} := \{1, \dots, k\}$ and scoring functions $\mathcal{H} \subset \{h : \mathcal{R} \rightarrow \mathbb{R}^k\}$, where the output on each dimension indicates the confidence of prediction. For some $h \in \mathcal{H}$, let us denote by $g_h : \mathcal{R} \rightarrow \mathcal{Y}$, $x \mapsto \arg \max_{i \in \{1, \dots, k\}} h^{(i)}(\phi(x))$ with $h^{(i)}(z)$ being the i -th component of $h(z)$. Let further ε_T denote the target error based on 0-1 loss. Then, with probability at least $1 - \delta$

$$\varepsilon_T(g_h \circ \phi) \leq \varepsilon_S^{(\rho)}(h \circ \phi) + \lambda_{\mathcal{H}}(\phi) + \widehat{d}_{h,\mathcal{H}}^{(\rho)}(\phi(\mathbf{x}), \phi(\mathbf{x}')) + \eta_{s+t,\mathcal{H},\delta} \quad (6)$$

where $\varepsilon_S^{(\rho)}(h) := \int_{\mathcal{X}} \Lambda_{\rho} \circ \rho_h(x, l_S) d\rho_h(x)$ with $\rho_h(x, l_S) := 1/2(h^{(l_S(x))}(x) - \max_{y \neq l_S(x)} h^{(y)}(x))$ and $\Lambda_{\rho}(x)$ being $1 - x/\rho$ if $0 \leq x < \rho$, 0 if $\rho \leq x$ and 1 otherwise. Further, the target terms T, p_T, l_T are defined analogously to S, p_S, l_S , respectively; the combined error is defined by $\lambda_{\mathcal{H}}(\phi) := \inf_{h \in \mathcal{H}} (\varepsilon_S^{(\rho)}(h \circ \phi) + \varepsilon_T^{(\rho)}(h))$, and the *empirical margin disparity discrepancy* (MDD) is given by

$$\widehat{d}_{h,\mathcal{H}}^{(\rho)}(\mathbf{x}', \mathbf{x}) := \max_{h' \in \mathcal{H}} \left[\frac{1}{s} \sum_{i=1}^s \Lambda_{\rho} \circ \rho_{h'}(x_i, g_h) - \frac{1}{t} \sum_{i=1}^t \Lambda_{\rho} \circ \rho_{h'}(x'_i, g_h) \right] \quad (7)$$

Similarly to Eq. (4), the source error can be further estimated by an empirical error and the combined error $\lambda_{\mathcal{H}}(\phi)$ cannot be estimated based on given data.

Balancing principle for regularized inverse problems Let \mathcal{H} and \mathcal{K} be two Hilbert spaces and $V : \mathcal{H} \rightarrow \mathcal{K}$ be a linear operator. Then, the linear *inverse problem* associated to a given *datum* $g \in \mathcal{K}$ is to find some function f satisfying $Vf = g$, see e.g. [45, 46, 40, 41] and references therein. In general, the above problem is ill-posed, i.e. a solution does either not exist, is not unique or does not depend continuously on g . Existence and uniqueness can be approached by using the the following minimizer as approximation of f

$$f_{\mathcal{H}} \in \arg \min_{f \in \mathcal{H}} \|Vf - g\|_{\mathcal{K}}^2$$

However, especially in the case of a noisy operator \widehat{V} and noisy data \widehat{g} , continuous dependency on data becomes an important issue which can be restored using Tikhonov regularization [45]

$$f_{\alpha} \in \arg \min_{f \in \mathcal{H}} \|\widehat{V}f - \widehat{g}\|_{\mathcal{K}}^2 + \alpha \|f\|_{\mathcal{H}}^2 \quad (8)$$

In many cases, probabilistic bounds on the error can be proven

$$\|f_{\alpha} - f_{\mathcal{H}}\|_{\mathcal{H}} \leq S(\alpha) + A(\alpha) \quad (9)$$

where $S(\alpha)$ is called *sampling error* made by considering noisy approximations \widehat{V} and \widehat{g} of V and g , respectively, and, $A(\alpha)$ is called *approximation error* originating from adding the regularizer $\|f\|_{\mathcal{H}}^2$. Commonly, $S(\alpha)$ decreases while $A(\alpha)$ increases with increasing α . We refer to [42] for detailed examples. The fact that $S(\alpha)$ decreases and $A(\alpha)$ increases motivates the so-called *balancing principle* which aims at computing α^* such that $S(\alpha^*) = A(\alpha^*)$ [26, 27, 42, 41, 28]. As a result, the balancing principle provides a procedure for approximating α^* without having access to the values of $A(\alpha)$. The balancing principle obtains optimal error rates in many settings [47, 48, 49, 42, 41, 28].

4 Balancing principle for domain adaptation

In the following, we present the mathematical foundations of Algorithm 1 in two steps. In a first step, in Subsection 4.1, we state our assumptions and detail our idea of choosing the value of balance between terms in a target error bound. In a second step, in Subsection 4.2, we propose an algorithmic criterion, the *balancing principle estimate* for approximating the theoretical choice of the balancing value without target labels. Based on this criterion, we explain why we expect Algorithm 1 to be accurate.

4.1 Balancing terms in a target error bound

Assumptions In the following, let $g_\alpha \circ \phi_\alpha$ be a minimizer of Eq. (1) and assume that a target error bound holds which is of the form given by Eq. (2). Our approach is based on the plausible assumptions (a) that the function $\alpha \mapsto E(\alpha)$ is continuous and bounded by some constant $B > 0$, and (b) that the function $\alpha \mapsto D(\alpha)$ is continuous, non-increasing and non-degenerate, i.e. $D(0) > 0$. We further make the technical assumption (c) that $\lim_{\alpha \rightarrow \infty} D(\alpha)/D(0) < \sup_{\alpha \in [0, \infty)} E(\alpha)/B$. Note that (c) is satisfied in most standard cases, where $D(\alpha) \rightarrow 0$ for $\alpha, t \rightarrow \infty$. For example consider the two accompanying target error bounds in Section 3 as discussed at the end of this section.

Bridging regularized inverse problems and domain adaptation Under the assumptions above, our domain adaptation learning setup in Section 2 shares characteristics with the setting of regularized inverse problems as described in Section 3. Indeed, the Tikhonov regularizer $\|f\|_{\mathcal{H}}^2$ in Eq. (8) is similarly applied as the distance-regularizer $d(\phi(\mathbf{x}), \phi(\mathbf{x}'))$ in Eq. (1). In addition, error bounds for inverse problems as given by Eq. (9) show a similar form as target error bounds following Eq. (2). The sampling error $S(\alpha)$ in Eq. (9) decreases similarly to the domain distance in Eq. (2) and the approximation error $A(\alpha)$ in Eq. (9) cannot be estimated similarly to the learning errors $E(\alpha)$ in Eq. (2). However, in the domain adaptation setting, the term $E(\alpha)$ does not necessarily increase. We approach this issue by considering the *least non-decreasing majorant* of $E(\alpha)$.

Definition 1 (Least non-decreasing majorant [50]). *The least non-decreasing majorant of $E(\alpha)$ is given by $\bar{E}(\alpha) := \sup_{\beta \in [0, \alpha]} E(\beta)$.*

Further upper bounding Eq. (2) by

$$\varepsilon_T(g_\alpha \circ \phi_\alpha) \leq D(\alpha) + E(\alpha) \leq D(\alpha) + \bar{E}(\alpha) \quad (10)$$

results in a form which satisfies all properties needed to apply the balancing principle, see Table 1.

Table 1: Correspondences between regularized inverse problems and domain adaptation which allow to apply the balancing principle. See Section 3 for details on the inverse problem setting.

Tikhonov-regularized inverse problem	Distance-regularized domain adaptation
$f_\alpha \in \arg \min_{f \in \mathcal{H}} \ \widehat{\nabla} f - \widehat{g}\ _{\mathcal{K}}^2 + \alpha \ f\ _{\mathcal{H}}^2$	$g_\alpha \circ \phi_\alpha \in \arg \min_{g \in \mathcal{G}, \phi \in \Phi} \widehat{\varepsilon}_S(g \circ \phi) + \alpha d(\phi(\mathbf{x}), \phi(\mathbf{x}'))$
$\ f_\alpha - f_{\mathcal{H}}\ _{\mathcal{H}} \leq S(\alpha) + A(\alpha)$	$\varepsilon_T(g_\alpha \circ \phi_\alpha) \leq D(\alpha) + E(\alpha)$
decreasing sampling error $S(\alpha)$	decreasing domain distance $D(\alpha)$
increasing approximation error $A(\alpha)$	bounded learning errors $E(\alpha)$
$A(\alpha)$ not estimable	$E(\alpha)$ not estimable
balance $A(\alpha^*) = S(\alpha^*)$	balance $\frac{D(\alpha^*)}{D(0)} = \frac{\bar{E}(\alpha^*)}{B}$

Balancing value for domain adaptation Having identified the shared characteristics between regularized inverse problems and domain adaptation, we now apply the techniques underlying the balancing principle to domain adaptation. We define the *balancing value* α^* as achieving

$$\frac{D(\alpha^*)}{D(0)} = \frac{\bar{E}(\alpha^*)}{B} \quad (11)$$

The normalizing factor $1/D(0)$ on the left-hand side of Eq. (11) and the factor $1/B$ ensure that the two curves $D(\alpha)/D(0)$ and $\bar{E}(\alpha)/B$ intersect. In particular, the existence of α^* follows from the

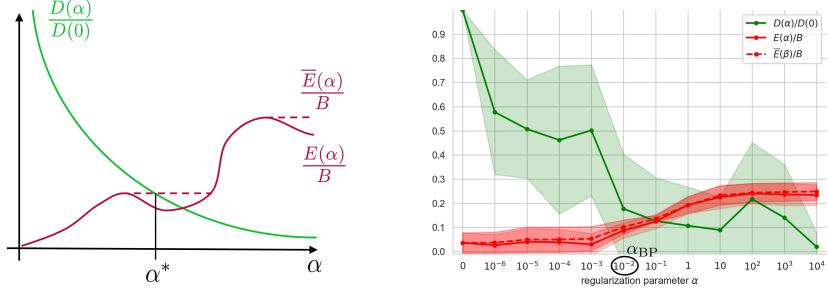


Figure 2: Left: The BPDA in Algorithm 1 overcomes the problem of the unknown learning errors term $E(\alpha)$ by approximating α^* which balances $D(\alpha)/D(0)$ (green) and the least non-decreasing majorant $\bar{E}(\alpha)/B$ (red dashed) of $E(\alpha)/B$ (red). Right: Average and standard deviation over 10 repetitions of estimated learning errors E and the domain distance D of the accompanying target error bound Eq. (4) for the models computed by Eq. (1) with the Maximum Mean Discrepancy [51] as distance. The BPDA chooses the value $\alpha_{\text{BP}} = 10^{-2}$ near the estimated balancing value.

assumptions (a)–(c) above. See Figure 2 for an illustration. Algorithm 1 approximates α^* . If $E(\alpha)$ is non-decreasing and the bound in Eq. (10) holds with equality, then the rate of the target error $\varepsilon_T(g_{\alpha^*} \circ \phi_{\alpha^*})$ is optimal, i.e. $\varepsilon_T(g_{\alpha^*} \circ \phi_{\alpha^*})$ is only a constant factor away from the optimum $\inf_{\alpha \in [0, \infty)} \varepsilon_T(g_\alpha \circ \phi_\alpha)$. See the supplementary material for a proof. This optimality property is shared with related regularization settings [47, 48, 49, 42, 28].

Two accompanying target error bounds Let us now discuss the reasoning above based on the two error bounds described in Section 3. First, recall the target error bound of [25] in Eq. (4). If we take $D(\alpha) := \hat{d}_{\mathcal{G}\Delta\mathcal{G}}(\phi_\alpha(\mathbf{x}), \phi_\alpha(\mathbf{x}')) + \eta_{t,\mathcal{G},\delta}$ and $E(\alpha) := \varepsilon_S(g_\alpha \circ \phi_\alpha) + \lambda_{\mathcal{G}}(\phi_\alpha)$, it is natural to assume $D(\alpha)$ to decrease with α , especially for adversarial approaches which penalize the empirical $\mathcal{G}\Delta\mathcal{G}$ -divergence $\hat{d}_{\mathcal{G}\Delta\mathcal{G}}(\phi(\mathbf{x}), \phi(\mathbf{x}'))$, see [16]. It also holds that $E(\alpha) \leq 3$ for all $\alpha \in [0, \infty)$. For the balancing value to exist for large sample size t , we need to verify that $D(\alpha)/D(0) \rightarrow 0$ for $t, \alpha \rightarrow \infty$. However, this is satisfied for most function classes Φ , since it can be assumed that the constant function $\phi : \mathbf{x} \mapsto c \in \mathbb{R}$ which achieves $\hat{d}_{\mathcal{G}\Delta\mathcal{G}}(\phi(\mathbf{x}), \phi(\mathbf{x}')) = 0$ is contained in Φ . That is, $\hat{d}_{\mathcal{G}\Delta\mathcal{G}}(\phi(\mathbf{x}), \phi(\mathbf{x}')) = 0$ can be achieved for $\alpha \rightarrow \infty$ and consequently $D(\alpha)/D(0) \rightarrow 0$ for $t, \alpha \rightarrow \infty$.

Consider now the bound in Eq. (6). By using $D(\alpha) := \hat{d}_{h,\mathcal{H}}^{(\rho)}(\phi_\alpha(\mathbf{x}), \phi_\alpha(\mathbf{x}')) + \eta_{s+t,\mathcal{H},\delta}$, $E(\alpha) := \varepsilon_S^{(\rho)}(h \circ \phi) + \lambda_{\mathcal{H}}(\phi)$ and $B = 3$, all assumptions above are naturally satisfied similarly to the target error bound in Eq. (4) and the balancing value exists. In Section 5, we empirically investigate the performance of our method based on our two accompanying target error bounds. However, let us first show how we can overcome the problem of the non-computable term $E(\alpha)$.

4.2 Approximation of balancing value

Unfortunately, $E(\alpha)$ in Eq. (1) usually contains information about the target labeling function l_T and α^* can therefore not be calculated directly.

Balancing principle estimate We propose the following criterion for estimating α^* which is realized by the BPDA in Algorithm 1.

Definition 2 (Balancing principle estimate). *Let $\alpha_1, \dots, \alpha_w, w \in \mathbb{N}$ with $\alpha_1 = 0$ be an increasing sequence of values in $[0, \infty)$ and denote by $f_i := g_{\alpha_i} \circ \phi_{\alpha_i}$. Then, the balancing principle estimate is*

$$\alpha_{\text{BP}} := \max \left\{ \alpha_i \mid \forall j \in \{1, \dots, i-1\} : \hat{\varepsilon}_T(f_i, f_j) \leq D(\alpha_j) \left(2 + \frac{2B}{D(0)} \right) + \eta_{t,\mathcal{G},\delta} \right\} \quad (12)$$

The balancing principle estimate in Definition 2 is based on checking an upper bound on the empirical cross-error $\hat{\varepsilon}_T(f_i, f_j)$ between two models f_i, f_j resulting from two different values α_i, α_j of the

regularization parameter, respectively. The empirical cross-error does not contain information about the unknown target labels $l_T(x'_1), \dots, l_T(x'_t)$ and can be computed based on available data. The main reason why we expect the balancing principle estimate α_{BP} to be near to the balancing value in Eq. (11) can be explained as follows.

Lemma 1. *Let $\delta \in (0, 1)$, $\alpha, \beta \in [0, \infty)$ and denote by $f_\alpha := g_\alpha \circ \phi_\alpha$. If $0 \leq \alpha \leq \beta \leq \alpha^*$ then the following holds with probability at least $1 - \delta$:*

$$\hat{\varepsilon}_T(f_\alpha, f_\beta) \leq D(\alpha) \left(2 + \frac{2B}{D(0)} \right) + \eta_{t, G, \delta} \quad (13)$$

Lemma 1 (see the supplementary material for its proof) shows that the inequality in Eq. (12) is satisfied if $\alpha_j \leq \alpha_i \leq \alpha^*$. This implies that $\alpha_i > \alpha^*$ if the criterion is violated for some $\alpha_j \leq \alpha_i$. Consequently, the maximum α_{BP} as defined in Eq. (12) from an increasing sequence $\alpha_1, \dots, \alpha_w$ which violates Eq. (13) for some $j \in \{1, \dots, i-1\}$ can be assumed to be near to α^* .

Generalization guarantee The model $g_{\alpha_{\text{BP}}} \circ \phi_{\alpha_{\text{BP}}}$ obtained by the BPDA in Algorithm 1 satisfies the following generalization guarantee.

Theorem 1. *Let $\delta \in (0, 1)$ and $\alpha_1, \dots, \alpha_w \in [0, \infty)$, $\alpha_1 = 0$ be an increasing sequence such that*

$$D(\alpha_l) \leq q \cdot D(\alpha_{l+1}) \quad (14)$$

for all $l \in \{1, \dots, w-1\}$ and some $q > 1$. Then, with probability at least $1 - \delta$

$$\varepsilon_T(g_{\alpha_{\text{BP}}} \circ \phi_{\alpha_{\text{BP}}}) \leq D(\alpha^*) \left(3 + \frac{3B}{D(0)} \right) q + \eta_{t, G, \delta} \quad (15)$$

Theorem 1 shows that the target error of the model $g_{\alpha_{\text{BP}}} \circ \phi_{\alpha_{\text{BP}}}$ identified by the BPDA has the same error rate as $D(\alpha^*)$ for $t \rightarrow \infty$. Moreover, if the optimum $\inf_{\alpha \in [0, \infty)} D(\alpha) + \bar{E}(\alpha)$ is achieved, then the error rate is optimal in the sense that $\varepsilon_T(g_{\alpha_{\text{BP}}} \circ \phi_{\alpha_{\text{BP}}})$ is only a constant factor worse than the minimum $\inf_{\alpha \in [0, \infty)} D(\alpha) + \bar{E}(\alpha)$. The constant factor is given by $(3 + 3B/D(0)) q \max\{D(0)/B, 1\}$. That is, the bound is larger for steeper D between two consecutive values for α (q is larger), and, it is larger for larger $D(0)/B$. The constant can be derived by bounding $D(\alpha^*)$ in Eq. (15) as detailed in the supplementary material. If $E(\alpha)$ is increasing and the instantiation bound in Eq. (2) is tight, i.e. it holds with equality, then the error $\varepsilon_T(g_{\alpha_{\text{BP}}} \circ \phi_{\alpha_{\text{BP}}})$ is only a constant factor worse than the optimal error $\inf_{\alpha \in [0, \infty)} \varepsilon_T(g_\alpha \circ \phi_\alpha)$.

It is well known that appropriate assumptions as given above are needed for successful domain adaptation [52, 19]. However, in practice, target error bounds are often not tight and optimality cannot be guaranteed. In the following section, we therefore investigate the performance of our method on benchmark datasets based on our two accompanying target error bounds.

5 Empirical evaluations

We empirically investigate the performance of our approach based on two target error bounds, two parameter selection methods, three datasets and different domain adaptation methods³.

5.1 Setup

Given a domain adaptation algorithm that follows Eq. (1), the goal is to identify the regularization parameter α from the sequence $0, 10^{-6}, 10^{-5}, \dots, 10^3, 10^4$ which leads to the smallest target error of the model learned by the algorithm.

Datasets We rely on one academic example which we call *Transformed Moons*. Transformed Moons shows a density ratio that is unbounded in large regions, see Figure 1. We also use the *Amazon Reviews* dataset [53]. This dataset contains text reviews from four domains: books (B), DVDs (D), electronics (E), and kitchen appliances (K). Reviews are encoded in 5000 dimensional feature vectors of bag-of-words unigrams and bigrams with binary labels: label 0 if the product is ranked by 1 to 3

³The source-code can be found at <https://github.com/Xpitfire/bpda>

stars, and label 1 if the product is ranked by 4 or 5 stars. From the four categories we obtain twelve domain adaptation tasks where each category serves once as source domain and once as target domain. We use the same data splits as previous works [54, 55, 16]. Thus, we have 2000 labeled source examples and 2000 unlabeled target examples for training, and between 3000 and 6000 examples for testing. Our third dataset is the *DomainNet-2019* dataset consisting of six different domains, each having 345 classes, and an average count of 288 images per class, i.e. around 0.6 million images [56]. However, our focus is not on large-scale domain adaptation, but rather on ranking model selection methods; hence, we propose a reduced version of the DomainNet-2019 dataset. In particular, we reduce the number of classes to five. We call our new dataset *MiniDomainNet*. See the supplementary material for the dataset statistics.

Balancing principle for domain adaptation We apply the BPDA in Algorithm 1 based on the two accompanying target error bounds described in Section 3. On the two-class datasets Transformed Moons and Amazon Reviews, we apply the BPDA based on Eq. (4). We set the domain distance $D(\alpha) := \hat{d}_{\mathcal{G}\Delta\mathcal{G}}(\phi_\alpha(\mathbf{x}), \phi_\alpha(\mathbf{x}')) + \eta_{t,\mathcal{G},\delta}$ and approximate its value by a classifier as proposed in [16, Subsection 3.2]. In our experiments on the multi-class dataset MiniDomainNet, we apply the BPDA based on Eq. (6). We define $D(\alpha) := \tilde{d}_{h,\mathcal{H}}^{(\rho)}(\phi_\alpha(\mathbf{x}), \phi_\alpha(\mathbf{x}')) + \eta_{s+t,\mathcal{H},\delta}$ and approximate its value as proposed in [20, Subsection 4.2]. In both variants of the BPDA, one for each bound, we follow the argument of [25, Subsection 7.2] to have enough unlabeled data to evaluate the bounds without considering the finite sample error term $\eta_{t,\mathcal{G},\delta}$. We repeat each domain adaptation model training several times. The if-statement in the BPDA in Algorithm 1 is considered violated, if there is a violation of the statement for at least one of the repetitions. For a fair comparison, the evaluations of IWV and DEV are also based on all repetitions. More precisely, for IWV and DEV, we choose the parameter with the lowest average importance weighted risk and lowest average DEV-risk, respectively, where the average is computed over all repetitions.

Parameter choice baselines We compare our parameter choice approach to four baselines. The first baseline is training on source data only (SO). The second baseline is the best target error (TB) and it serves as a lower bound for the error. The third baseline is *importance weighted validation* [3] (IWV). We follow [23] and use *held-out* validation, i.e. we hold out a part of the training data as validation set, and we compute the importance weights based on this validation set. We also follow [57] and [23, Subsection 4.3] to estimate the importance weight by a classifier trained to separate source from target data. The classifier is tuned separately for each task and dataset such that its validation misclassification error is at most 0.05. For MiniDomainNet, we compute the importance weight based on the features of the pre-trained ResNet-18 [58]. The fourth baseline is *deep embedded validation* [23] (DEV) which solves the unbounded variance issue in IWV. Following [23], DEV is applied on the features of the neural networks.

Domain adaptation methods In our experiments, we use three domain adaptation techniques. Domain-adversarial neural networks [16] (DANN), Maximum Mean Discrepancy [59, 15] (MMD) and Central Moment Discrepancy [18] (CMD). The details of all neural network architectures used, as well as the training strategy and hyperparameters are provided in the supplementary material.

Assumptions To evaluate the compliance of the assumptions in Subsection 4.1 for the algorithms CMD and MMD, we estimate E , the domain distance D and the least non-decreasing majorant \bar{E} on the Transformed Moons dataset. In particular, the term E is computed using the (in unsupervised domain adaptation unknown) target labels for training a classifier to minimize the minimum possible combined error in Eq. (4). We repeat the evaluations 10 times for different initializations of the domain adaptation model weights.

5.2 Results

Assumptions The terms E and D for MMD on Transformed Moons are shown in Figure 2 and, for CMD, in the supplementary material. The following observation (a)–(d) can be made: (a) E is bounded, the mean curves of $D(\alpha)$ tend to be non-increasing and the technical assumption $D(\alpha)/D(0) < \sup_{\alpha \in [0, \infty)} E(\alpha)/B$ is satisfied for $\alpha \geq 10^3$. That is, the mean curves tend to follow the assumptions listed in Section 4.1. (b) The mean curves \bar{E} and E tend to be similar. (c) The average parameters chosen by the balancing principle (see α_{BP} in Figure 2) are the maximum values for

which the mean curves of $\bar{E}(\alpha)/B$ are smaller than the mean curves of $D(\alpha)/D(0)$. That is, the BPDA described in Algorithm 1 tends to perform as expected. (d) There is a moderate trend towards a violation of the monotonicity assumption for D for CMD (see the supplementary material for the figure). However, the BPDA is (on average) robust w.r.t. this violation as it picks nearly the optimal value (see Table 2).

Comparative study Average values of our empirical evaluations on Transformed Moons and Amazon Review are summarized in Table 2. For the full tables on Transformed Moons, Amazon Reviews and MiniDomainNet, see the supplementary material. Although making no assumptions on the ratio between target and source density, the BPDA outperforms related parameter choice methods (IWV, DEV) on almost all tasks on Transformed Moons and Amazon Reviews, and, obtains competitive results on MiniDomainNet.

Table 2: Average target misclassification error with best values in bold. See the supplementary material for the full results for all datasets and domain adaptation tasks.

Method	Transformed Moons					Amazon Reviews				
	SO	IWV	DEV	BPDA	TB	SO	IWV	DEV	BPDA	TB
MMD	0.21	0.20	0.34	0.16	0.16	0.27	0.25	0.25	0.22	0.21
DANN	0.18	0.18	0.17	0.12	0.12	0.28	0.28	0.32	0.27	0.24
CMD	0.21	0.20	0.19	0.19	0.18	0.28	0.24	0.25	0.21	0.21
Avg.	0.20	0.19	0.23	0.16	0.15	0.28	0.26	0.27	0.23	0.22

6 Conclusion and future work

One widely-used technique for unsupervised domain adaptation is to map the data into a new feature space where the source and target data representations appear similar, and where enough information is preserved for prediction [14]. The similarity is often realized by minimizing the source error weighted by a distance measure between source and target representations. One common interpretation of this approach is to learn *domain-invariant* representations. However, there is a subtle difference between *distance-regularization* by penalizing the error minimization with a distance as above, and *distance-minimization* which results in domain-invariant representations. The latter can even lead to deteriorated performance [19]. In contrast, the interpretation as regularization problem opens up a powerful toolbox of mathematical techniques. We take up the technique of balancing stability and approximation in the regularization of ill-posed inverse problems, to tackle the problem of choosing the crucial regularization parameter in distance-regularized domain adaptation. Our approach overcomes the lack of target labels, it satisfies a generalization guarantee, and is (to the best of our knowledge) the first theoretically justified method that allows source and target distributions with disjoint supports. Finally, our approach outperforms or is on par, with the state of the art on the problem of choosing the regularization parameter, on several domain adaptation methods; applied on different datasets.

Broader impact

Many fields, such as manufacturing, personalized medicine or analytical chemistry, have to handle problems of domain shift together with issues of data limitations. These areas can profit from our research, as it provides a principled way of choosing a justified regularization parameter of unsupervised domain adaptation algorithms. Our method provides a high level of trust by applying mathematical techniques with guarantees originally developed in the area of ill-posed inverse problems. However, in domain adaptation in general, one critical point is the bias stored in the source data (e.g. past product, previous patient data, previous spectroscope), since knowledge in this data is used to improve the learning on the unlabeled target data (e.g. new product, new patient data, new spectroscope). If such a bias is present in one of the source domains, the predictions in new target domains might always suffer from this bias. We therefore suggest to not only look at new data in new target domains but more importantly consider dataset bias in already collected labeled source data.

Acknowledgments and Disclosure of Funding

We thank Markus Holzleitner for careful proofreading. We thank the anonymous reviewers for a constructive discussion which formed the basis for the supplementary material's section about limitations, risks and future developments. The research reported in this paper has been funded by the Federal Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK), the Federal Ministry for Digital and Economic Affairs (BMDW), and the Province of Upper Austria in the frame of the COMET–Competence Centers for Excellent Technologies Programme and the COMET Module S3AI managed by the Austrian Research Promotion Agency FFG. The LIT AI Lab is financed by the Federal State of Upper Austria. We further acknowledge the partial support of the FFG project AutoQual-I. The ELLIS Univ Linz, the LIT AI Lab and the Institute for Machine Learning, are supported by the Federal State Upper Austria. IARAI is supported by Here Technologies. We thank the projects AI-MOTION (LIT-2018-6-YOU-212), DeepToxGen (LIT-2017-3-YOU-003), AI-SNN (LIT-2018-6-YOU-214), DeepFlood (LIT-2019-8-YOU-213), Medical Cognitive Computing Center (MC3), INCONTROL-RL (FFG-881064), PRIMAL (FFG-873979), DL for GranularFlow (FFG-871302), AIRI FG 9-N (FWF-36284, FWF-36235), ELISE (H2020-ICT-2019-3 ID: 951847), AIDD (MSCA-ITN-2020 ID: 956832). We thank Janssen Pharmaceutica (MaDeSMart, HBC.2018.2287), Audi.JKU Deep Learning Center, TGW LOGISTICS GROUP GMBH, Silicon Austria Labs (SAL), FILL Gesellschaft mbH, Anyline GmbH, Google, ZF Friedrichshafen AG, Robert Bosch GmbH, UCB Biopharma SRL, Merck Healthcare KGaA, Verbund AG, TÜV Austria, and the NVIDIA Corporation. This material is based upon work supported by the Google Cloud Research Credits program with the award GCP19980904.

References

- [1] Sinno J Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.
- [2] Hideyoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.
- [3] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul V Buenau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in Neural Information Processing Systems*, pages 1433–1440, 2008.
- [4] Jiayuan Huang, Arthur Gretton, Karsten M Borgwardt, Bernhard Schölkopf, and Alex J Smola. Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems*, pages 601–608, 2007.
- [5] Corinna Cortes, Yishay Mansour, and Mehryar Mohri. Learning bounds for importance weighting. In *Advances in Neural Information Processing Systems*, pages 442–450, 2010.
- [6] Corinna Cortes, Mehryar Mohri, and Andrés M Medina. Adaptation based on generalized discrepancy. *Journal of Machine Learning Research*, 20(1):1–30, 2019.
- [7] ER Gizewski, L Mayer, BA Moser, DH Nguyen, S Pereverzyev Jr, SV Pereverzyev, N Shepeleva, and W Zellinger. On a regularization of unsupervised domain adaptation in RKHS. *RICAM-Reports*, 2021.
- [8] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems*, pages 137–144, 2007.
- [9] Hal Daumé III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, 2007.
- [10] Sinno J Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011.
- [11] Boqing Gong, Kristen Grauman, and Fei Sha. Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *Proceedings of the International Conference on Machine Learning*, pages 222–230, 2013.

- [12] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.
- [13] Mingming Gong, Kun Zhang, Tongliang Liu, Dacheng Tao, Clark Glymour, and Bernhard Schölkopf. Domain adaptation with conditional transferable components. In *Proceedings of the International Conference on Machine Learning*, pages 2839–2848, 2016.
- [14] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset Shift in Machine Learning*. The MIT Press, 2009.
- [15] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *Proceedings of the International Conference on Machine Learning*, pages 97–105, 2015.
- [16] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(Jan):1–35, 2016.
- [17] Baichen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *Proceedings of the European Conference on Computer Vision*, pages 443–450, 2016.
- [18] Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. Central moment discrepancy (cmd) for domain-invariant representation learning. *International Conference on Learning Representations*, 2017.
- [19] Fredrik D Johansson, David Sontag, and Rajesh Ranganath. Support and invertibility in domain-invariant representations. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 527–536, 2019.
- [20] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. Bridging theory and algorithm for domain adaptation. In *Proceedings of the International Conference on Machine Learning*, pages 7404–7413, 2019.
- [21] Victor Bouvier, Philippe Very, Clément Chastagnol, Myriam Tami, and Céline Hudelot. Robust domain adaptation: Representations, weights and inductive bias. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2020.
- [22] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. *Proceedings of the Annual Conference on Learning Theory*, 2009.
- [23] Kaichao You, Ximei Wang, Mingsheng Long, and Michael Jordan. Towards accurate model selection in deep unsupervised domain adaptation. In *Proceedings of the International Conference on Machine Learning*, pages 7124–7133, 2019.
- [24] Wouter M Kouw, Jesse H Krijthe, and Marco Loog. Robust importance-weighted cross-validation under sample selection bias. In *IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6. IEEE, 2019.
- [25] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, 2010.
- [26] OV Lepskii. On a problem of adaptive estimation in gaussian white noise. *Theory of Probability & Its Applications*, 35(3):454–466, 1991.
- [27] Alexander Goldenshluger and Sergei V Pereverzev. Adaptive estimation of linear functionals in hilbert scales from indirect white noise observations. *Probability Theory and Related Fields*, 118(2):169–186, 2000.
- [28] Shuai Lu, Peter Mathé, and Sergei V Pereverzev. Balancing principle in supervised learning for a general regularization scheme. *Applied and Computational Harmonic Analysis*, 48(1):123–148, 2020.

- [29] Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9):1853–1865, 2017.
- [30] Uri Shalit, Fredrik D Johansson, and David Sontag. Estimating individual treatment effect: generalization bounds and algorithms. In *Proceedings of the International Conference on Machine Learning*, pages 3076–3085, 2017.
- [31] Mahsa Baktashmotagh, Mehrtash T Harandi, Brian C Lovell, and Mathieu Salzmann. Unsupervised domain adaptation by domain invariant projection. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 769–776, 2013.
- [32] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In *Advances in Neural Information Processing Systems*, pages 136–144, 2016.
- [33] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6629–6640, 2017.
- [34] Werner Zellinger, Bernhard A Moser, and Susanne Saminger-Platz. On generalization in moment-based domain adaptation. *Annals of Mathematics and Artificial Intelligence*, 89(3):333–369, 2021.
- [35] Werner Zellinger, Bernhard A. Moser, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. Robust unsupervised domain adaptation for neural networks via moment alignment. *Information Sciences*, 483:174–191, May 2019.
- [36] Werner Zellinger and Bernhard A Moser. On the truncated Hausdorff moment problem under Sobolev regularity conditions. *Applied Mathematics and Computation*, 400:126057, 2021.
- [37] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.
- [38] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *Proceedings of the International Conference on Machine Learning*, pages 10–18, 2013.
- [39] Fuzhen Zhuang, Xiaohu Cheng, Ping Luo, Sinno Jialin Pan, and Qing He. Supervised representation learning: Transfer learning with deep autoencoders. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 4119–4125, 2015.
- [40] Heinz W Engl, Martin Hanke, and Andreas Neubauer. *Regularization of inverse problems*, volume 375. Springer Science & Business Media, 1996.
- [41] Shuai Lu and Sergei V Pereverzev. *Regularization theory for ill-posed problems: selected topics*, volume 58. Walter de Gruyter, 2013.
- [42] Ernesto De Vito, Sergei V Pereverzyev, and Lorenzo Rosasco. Adaptive kernel methods using the balancing principle. *Foundations of Computational Mathematics*, 10(4):455–479, 2010.
- [43] Ramin Nikzad-Langerodi, Werner Zellinger, Susanne Saminger-Platz, and Bernhard A Moser. Domain adaptation for regression under beer-lambert’s law. *Knowledge-Based Systems*, 210:106447, 2020.
- [44] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases-Volume 30*, pages 180–191. VLDB Endowment, 2004.
- [45] Andrei Nikolaevich Tikhonov. On the solution of ill-posed problems and the method of regularization. In *Doklady Akademii Nauk*, volume 151, pages 501–504, 1963.
- [46] CW Groetsch. The theory of tikhonov regularization for fredholm equations. *Boston Pitman Publication*, 1984.

- [47] Frank Bauer, Sergei Pereverzev, and Lorenzo Rosasco. On regularization algorithms in learning theory. *Journal of complexity*, 23(1):52–72, 2007.
- [48] Andrea Caponnetto and Ernesto De Vito. Optimal rates for the regularized least-squares algorithm. *Foundations of Computational Mathematics*, 7(3):331–368, 2007.
- [49] Steve Smale and Ding-Xuan Zhou. Learning theory estimates via integral operators and their approximations. *Constructive approximation*, 26(2):153–172, 2007.
- [50] Gord Sinnamon. Transferring monotonicity in weighted norm inequalities. *Collectanea Mathematica*, pages 181–216, 2003.
- [51] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(3):723–773, 2012.
- [52] Shai Ben-David, Tyler Lu, Teresa Luu, and Dávid Pál. Impossibility theorems for domain adaptation. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 129–136, 2010.
- [53] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 120–128, 2006.
- [54] Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. *Proceedings of the International Conference on Machine Learning*, pages 767–774, 2012.
- [55] Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. The variational fair auto encoder. *International Conference on Learning Representations*, 2016.
- [56] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1406–1415, 2019.
- [57] Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning for differing training and test distributions. In *Proceedings of the International Conference on Machine Learning*, pages 81–88, 2007.
- [58] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [59] Arthur Gretton, Karsten M Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. A kernel method for the two-sample-problem. In *Advances in Neural Information Processing Systems*, pages 513–520, 2006.

Supplementary material for: The balancing principle for parameter choice in distance-regularized domain adaptation

Werner Zellinger^{1,*} Natalia Shepeleva¹ Marius-Constantin Dinu^{2,3}

Hamid Eghbal-zadeh^{4,5} Duc Hoan Nguyen⁶ Bernhard Nessler²

Sergei V. Pereverzyev⁶ Bernhard A. Moser¹

¹Software Competence Center Hagenberg GmbH

²Institute for Machine Learning, Johannes Kepler University Linz

³Dynatrace Research

⁴Institute of Computational Perception, Johannes Kepler University Linz

⁵LIT AI Lab, Johannes Kepler University Linz

⁶Johann Radon Institute for Computational and Applied Mathematics, Austrian Academy of Sciences

*werner.zellinger@scch.at

1 Balancing value of target error bound

In this work, we assume that a target error bound is given which satisfies the form

$$\varepsilon_T(g_\alpha \circ \phi_\alpha) \leq D(\alpha) + E(\alpha) \quad (1)$$

such that (a) $\alpha \mapsto E(\alpha)$ is continuous and bounded by some constant $B > 0$, (b) $\alpha \mapsto D(\alpha) \geq 0$ is continuous, non-increasing and non-degenerate, i.e. $D(0) > 0$ and (c) $\lim_{\alpha \rightarrow \infty} \frac{D(\alpha)}{D(0)} < \sup_{\alpha \in [0, \infty)} \frac{E(\alpha)}{B}$. Under these assumptions, the balancing value α^* which achieves

$$\frac{D(\alpha^*)}{D(0)} = \frac{\overline{E}(\alpha^*)}{B} \quad (2)$$

exists, where $\overline{E}(\alpha)$ refers to the least non-decreasing majorant of $E(\alpha)$.

Definition 1 (Least non-decreasing majorant [1]). *The least non-decreasing majorant of $E(\alpha)$ is given by $\overline{E}(\alpha) := \sup_{\beta \in [0, \alpha]} E(\beta)$.*

Interestingly, the terms $D(\alpha^*)$ and $D(\alpha^*) + \overline{E}(\alpha^*)$ evaluated at the balancing value α^* are only a constant factor away from the minimizer $\inf_{\alpha \in [0, \infty)} D(\alpha) + \overline{E}(\alpha)$.

Lemma 0. *If $\inf_{\alpha \in [0, \infty)} D(\alpha) + \overline{E}(\alpha)$ is achieved, then*

$$D(\alpha^*) \leq \max \left\{ \frac{D(0)}{B}, 1 \right\} \inf_{\alpha \in [0, \infty)} D(\alpha) + \overline{E}(\alpha) \quad (3)$$

$$D(\alpha^*) + \overline{E}(\alpha^*) \leq 2 \max \left\{ \frac{D(0)}{B}, \frac{B}{D(0)} \right\} \inf_{\alpha \in [0, \infty)} D(\alpha) + \overline{E}(\alpha) \quad (4)$$

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

Lemma 0 proves, under certain assumptions, the optimality of the target error rate for the model $g_{\alpha^*} \circ \phi_{\alpha^*}$. More precisely, Eq. (4) implies that the error $\varepsilon_T(g_{\alpha^*} \circ \phi_{\alpha^*})$ is only a constant factor away from the optimum $\inf_{\alpha \in [0, \infty)} \varepsilon_T(g_\alpha \circ \phi_\alpha)$ if, it exists, $E(\alpha)$ is non-decreasing and Eq. (1) holds with equality.

Proof. Denote by $\alpha_{\text{opt}} \in [0, \infty)$ the value achieving the infimum of $D(\alpha) + \bar{E}(\alpha)$. If $\alpha_{\text{opt}} \leq \alpha^*$, then the definition of α^* and assumption (b) imply

$$\bar{E}(\alpha^*) \frac{D(0)}{B} \leq D(\alpha^*) \leq D(\alpha_{\text{opt}}) \leq \inf_{\alpha \in [0, \infty)} D(\alpha) + \bar{E}(\alpha) \quad (5)$$

If $\alpha_{\text{opt}} > \alpha^*$ then the definition of α^* and Definition 1 imply

$$D(\alpha^*) \frac{B}{D(0)} \leq \bar{E}(\alpha^*) \leq \bar{E}(\alpha_{\text{opt}}) \leq \inf_{\alpha \in [0, \infty)} D(\alpha) + \bar{E}(\alpha) \quad (6)$$

Combining Eq. (5) and Eq. (6) for $D(\alpha^*)$ gives

$$D(\alpha^*) \leq \max \left\{ \frac{D(0)}{B}, 1 \right\} \inf_{\alpha \in [0, \infty)} D(\alpha) + \bar{E}(\alpha) \quad (7)$$

Combining Eq. (5) and Eq. (6) for $\bar{E}(\alpha^*)$ gives

$$\bar{E}(\alpha^*) \leq \max \left\{ \frac{B}{D(0)}, 1 \right\} \inf_{\alpha \in [0, \infty)} D(\alpha) + \bar{E}(\alpha) \quad (8)$$

Summing Eq. (7) and Eq. (8) yields

$$\begin{aligned} D(\alpha^*) + \bar{E}(\alpha^*) &\leq \max \left\{ \frac{D(0)}{B}, 1 \right\} \inf_{\alpha \in [0, \infty)} D(\alpha) + \bar{E}(\alpha) \\ &\quad + \max \left\{ \frac{B}{D(0)}, 1 \right\} \inf_{\alpha \in [0, \infty)} D(\alpha) + \bar{E}(\alpha) \\ &\leq 2 \max \left\{ \frac{B}{D(0)}, \frac{D(0)}{B} \right\} \inf_{\alpha \in [0, \infty)} D(\alpha) + \bar{E}(\alpha) \end{aligned}$$

□

2 Criterion for approximating the balancing value

Recall that we assume that the target cross-errors satisfy some concentration inequality

$$|\varepsilon_T(f, g) - \hat{\varepsilon}_T(f, g)| \leq \eta_{t, \mathcal{F}, \delta} \quad (9)$$

which holds with probability at least $1 - \delta$ uniformly over all $f, g \in \mathcal{F}$ for some $\eta_{t, \mathcal{F}, \delta} \in \mathbb{R}$ such that $\eta_{t, \mathcal{F}, \delta} \rightarrow 0$ for $t \rightarrow \infty$. The main criterion used to define the balancing principle is as follows.

Lemma 1. Let $\delta \in (0, 1)$, $\alpha, \beta \in [0, \infty)$ and denote by $f_\alpha := g_\alpha \circ \phi_\alpha$. If $0 \leq \alpha \leq \beta \leq \alpha^*$ then the following holds with probability at least $1 - \delta$:

$$\hat{\varepsilon}_T(f_\alpha, f_\beta) \leq D(\alpha) \left(2 + \frac{2B}{D(0)} \right) + \eta_{t, \mathcal{G}, \delta} \quad (10)$$

Proof of Lemma 1. The following inequalities are all to be understood to hold with probability at least $1 - \delta$. For all $\alpha \leq \beta \leq \alpha^*$, Eq. (9) and the triangle inequality give

$$\begin{aligned} \hat{\varepsilon}_T(f_\alpha, f_\beta) &\leq \varepsilon_T(f_\alpha, f_\beta) + \eta_{t, \mathcal{G}, \delta} \\ &\leq \varepsilon_T(f_\alpha) + \varepsilon_T(f_\beta) + \eta_{t, \mathcal{G}, \delta} \end{aligned}$$

Using the instantiation bound of the balancing principle in Eq. (1) further implies that

$$\hat{\varepsilon}_T(f_\alpha, f_\beta) \leq D(\alpha) + E(\alpha) + D(\beta) + E(\beta) + \eta_{t, \mathcal{G}, \delta}$$

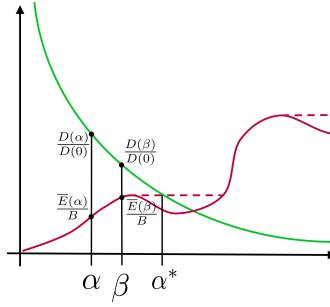


Figure 1: The proof of Lemma 1 is based on the monotonicity of $\frac{D(\alpha)}{D(0)}$ (green) and the monotonicity of the least non-decreasing majorant $\frac{\bar{E}(\alpha)}{B}$ (red dashed) of $\frac{E(\alpha)}{B}$ (red).

Definition 1 of the least non-decreasing majorant gives

$$\hat{\varepsilon}_T(f_\alpha, f_\beta) \leq D(\alpha) + \bar{E}(\alpha) + D(\beta) + \bar{E}(\beta) + \eta_{t,G,\delta}$$

Finally, we follow [2] and use the monotonicity of D and \bar{E} to obtain

$$\begin{aligned} \hat{\varepsilon}_T(f_\alpha, f_\beta) &\leq D(\alpha) + \frac{BD(\alpha)}{D(0)} + D(\beta) + \frac{BD(\beta)}{D(0)} + \eta_{t,G,\delta} \\ &\leq D(\alpha) \left(2 + \frac{2B}{D(0)} \right) + \eta_{t,G,\delta} \end{aligned}$$

Figure 1 provides a helpful illustration for the last two steps. \square

3 Generalization guarantee for balancing principle estimate

Our main theorem is stated as follows.

Theorem 1. Let $\delta \in (0, 1)$ and $\alpha_1, \dots, \alpha_w \in [0, \infty)$, $\alpha_1 = 0$ be an increasing sequence such that

$$D(\alpha_l) \leq q \cdot D(\alpha_{l+1}) \quad (11)$$

for all $l \in \{1, \dots, w-1\}$ and some $q > 1$. Then, with probability at least $1 - \delta$

$$\varepsilon_T(g_{\alpha_{\text{BP}}} \circ \phi_{\alpha_{\text{BP}}}) \leq D(\alpha^*) \left(3 + \frac{3B}{D(0)} \right) q + \eta_{t,G,\delta} \quad (12)$$

The following proof of Theorem 1 follows arguments from the principle of balancing stability and approximation in the theory of regularized ill-posed inverse problems. See Theorem 1 in [3] for a similar application to the adaptive choice of parameters in kernel regression.

Proof of Thm. 1. Let us denote by

$$\bar{\alpha} := \max \left\{ \alpha_i \mid \frac{\bar{E}(\alpha_i)}{B} \leq \frac{D(\alpha_i)}{D(0)}, i \in \{1, \dots, w\} \right\}$$

and by $f_\alpha := g_\alpha \circ \phi_\alpha$. From Eq. (10) we obtain for all $j \in \{1, \dots, w\}$ such that $\alpha_j \leq \bar{\alpha} \leq \alpha^*$ with probability at least $1 - \delta$

$$\varepsilon_T(f_{\alpha_j}, f_{\bar{\alpha}}) \leq D(\alpha_j) \left(2 + \frac{2B}{D(0)} \right) + \eta_{t,G,\delta}. \quad (13)$$

Note that $\bar{\alpha} \in \{\alpha_1, \dots, \alpha_w\}$ and that α_{BP} is the maximum of all $\alpha_i \in \{\alpha_1, \dots, \alpha_w\}$ satisfying

$$\varepsilon_T(f_{\alpha_i}, f_{\alpha_j}) \leq D(\alpha_j) \left(2 + \frac{2B}{D(0)} \right) + \eta_{t,G,\delta} \quad (14)$$

for all $j \in \{1, \dots, i-1\}$. It follows that $\bar{\alpha} \leq \alpha_{\text{BP}}$. Moreover, with probability at least $1 - \delta$,

$$\begin{aligned}\varepsilon_T(f_{\alpha_{\text{BP}}}) &\leq \varepsilon_T(f_{\alpha_{\text{BP}}}, f_{\bar{\alpha}}) + \varepsilon_T(f_{\bar{\alpha}}) \\ &\leq D(\bar{\alpha}) \left(2 + \frac{2B}{D(0)} \right) + \eta_{t, G, \delta} + \varepsilon_T(f_{\bar{\alpha}}) \\ &\leq D(\bar{\alpha}) \left(2 + \frac{2B}{D(0)} \right) + \eta_{t, G, \delta} + D(\bar{\alpha}) + E(\bar{\alpha}) \\ &\leq D(\bar{\alpha}) \left(2 + \frac{2B}{D(0)} \right) + \eta_{t, G, \delta} + D(\bar{\alpha}) + B \frac{D(\bar{\alpha})}{D(0)} \\ &= D(\bar{\alpha}) \left(3 + \frac{3B}{D(0)} \right) + \eta_{t, G, \delta}\end{aligned}$$

where we used the triangle inequality and Lemma 1 to prove the first two inequalities, followed by Eq. (1) and the same monotonicity argument as used in the proof of Lemma 1, see also Figure 1.

Finally, let l be such that $\bar{\alpha} =: \alpha_l \leq \alpha^* \leq \alpha_{l+1}$. Since D is non-increasing, we obtain $qD(\alpha^*) \geq qD(\alpha_{l+1})$ and, by assumption, $qD(\alpha_{l+1}) \geq D(\alpha_l) = D(\bar{\alpha})$. The final inequality is shown by recalling that $\bar{\alpha} \leq \alpha^*$. \square

4 MiniDomainNet dataset

The parameter selection methods for domain adaptation require to train several models with various parameters. Furthermore, evaluation of such methods include applying various domain adaptation techniques, which results in high computational demand when using large-scale datasets. In order to reduce the computational resources needed in this area, while keeping the difficulty of working with high-resolution images, and working on a problem with several domains, we fork a smaller version of the DomainNet dataset [4], which we call the *MiniDomainNet* dataset. MiniDomainNet makes research on the area of parameter selection for domain adaptation more accessible, by significantly reducing the computational needs, while providing a challenging, and sufficiently-large test bed for evaluating deep models.

The DomainNet dataset consists of approximately 0.6 million images divided into 6 domains (Quick-draw, Real, Clipart, Sketch, Infograph, and Painting), with each domain having 345 classes. The average count of images of DomainNet in each class, and across all domains is approx. 288. We curate the MiniDomainNet dataset from the DomainNet dataset as follows. We select the top-five largest classes in the training set of DomainNet, based on the highest average image-count per class across all domains. This selection process will result in a dataset with the largest amount of training data per class, which is ideal for training deep models.

In our experiments with MiniDomainNet, we follow a recommendation in [4], that uses a combined-source setting. To define our domain adaptation tasks, we select 5 out of the 6 domains and combine them into our *combined source* dataset. We use the remaining domain as our target dataset. By permuting all source combinations, we then define 6 domain adaptation tasks, which we refer to as combined-source datasets (CS, as denoted in Table 5).

In addition to providing the MiniDomainNet, we further address an issue regarding the currently available version of the DomainNet dataset. During our development process, we found that 10 files from the class 327 (t-shirt) in the painting domain sub-set, were missing in the file list of the training set (`painting_train.txt`). We provide a fix for this issue in our source code, by inserting the missing class references and their corresponding files. The fix can be found in the source code, in `data loaders/domainnet.py`.

5 Extended empirical evaluations

In this section, we provide details of our training setup, the computational resources used to conduct the experiments, the model selection procedures, and, our evaluation results.

5.1 Details for training

Transformed Moons On the Transformed Moons dataset, we use a feed-forward network with two fully-connected layers, with 16 nodes each, followed by ReLU non-linearity. The network is optimized by Adam [5] optimizer for 250 epochs, with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and the initial learning rate of 0.01, using a MultiStep scheduler which halved the learning rate in epochs 50, 100, and 150. To train proxy-A classifiers (required in [6]), we use 1 fully-connected layer with 16 nodes; and are trained with Adam optimizer for 200 epochs, with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and an initial learning rate of 0.01, and a MultiStep scheduler halving learning rate on epochs 50, 100, 150.

Amazon Reviews For Amazon Reviews, we follow [7] and use a feed-forward network three fully-connected layers, with 100 nodes each, and sigmoid non-linearity. The optimizer, learning rate, and scheduler are the same as in the Transformed Moons experiments (see above). We train each model for CMD and MMD experiments for 50 epochs and for DANN for 500. To estimate the $\mathcal{G}\Delta\mathcal{G}$ -divergence, we follow [7] and train a classifier for separating the source sample and the target sample. In particular we apply 2 fully-connected layers with 100 nodes each and use the Adam optimizer for 200 epochs, with $\beta_1 = 0.9$, $\beta_2 = 0.999$, the initial learning rate of 0.01, and, a MultiStep scheduler halving learning rate on epochs 50, 100, 150.

MiniDomainNet Following the pre-trained setup from [4], we use a frozen ResNet-18 backbone model which was trained on ImageNet [8], and operate subsequent computations on the 512 dimensional extracted features. To alleviate overfitting effects on pre-computed features, we perform data augmentation on each batch and forward the images through the backbone each time. We incorporate zero padding before resizing the images to 256×256 to avoid image distortions. Following the guidance for data augmentation techniques from [9], we perform random resized cropping to 224×224 with a random viewport between 70% and 100% of the original image, random horizontal flipping, color jittering of 0.25% on each RGB channel, and a ± 2 degree rotation. After the ResNet-18 backbone output, we add several projection layers, and define the domain adaptation layers on which we use the domain adaptation methods to align the representations. The first layers are defined as a common architecture across the different domain adaptation methods. Additional layers are further added for the classification networks, according to the requirements of the individual domain adaptation methods in CMD or MMD. The number of layers/neurons in the upper layers of our architecture have been tuned in order to achieve the best performance in the source-only setup. See Table 1 for a detailed description of the architecture used. We perform experiments on all 6 domain adaptation tasks as defined in 5.4 for each of the previously listed methods. All methods have been trained for 50 epochs with Adam optimizer, an initial learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a MultiStep learning rate scheduler, halving the learning rate after 15 and 35 epochs.

To apply the balancing principle, we require the training of an additional MDD classifier, see [10]), using the features of the adaptation layer from CMD and MMD, which is further used to calculate the MDD distance. The architecture of the MDD classifier is listed in Table 2. The MDD classifiers are trained with Adam optimizer, initial learning rate of 0.0001, and a MultiStep scheduler halving the learning rate after 15 and 25 epochs, and in total we run them for 35 epochs. For selecting the disparity parameters we followed the guidance from [11], and set $\gamma = 1.1$ in the MDD training loss, and $\rho = 0$ for calculating the MDD distance employed in BPDA.

5.2 Details for computational resources and source code

In experiments on Transformed Moons and Amazon Reviews, we used two HPC stations with in total 8xNVIDIA TITAN RTX 24GB, 4xIntel Xeon Scalable Processors Skylake Gold 6130 (2.10 GHz) and Ubuntu 18.04. All methods have been implemented in python using the *Pytorch library* [12]. We use *Scikit-learn library* [13] for evaluation measures and toy datasets, and the *TQDM library* [14], and *Tensorboard* [15] for keeping track of the progress of our experiments.

5.3 Details for model selection

Transformed Moons and Amazon Reviews IWV [16], DEV [11], and BPDA are used to choose the best parameter $\alpha \in \{0, 10^{-6}, 10^{-5}, \dots, 10^3, 10^4\}$, for three different distance-regularized domain adaptation methods, namely DANN [7], MMD [17, 18] and CMD [19]. The Transformed Moons and Amazon Reviews datasets contain only 2 classes; hence, BPDA is employed using the

Table 1: Architectural listing of all layers for training on the MiniDomainNet dataset.

Common Architecture		
	Layers	Values
Backbone Output Layer	ResNet-18 (Adaptive Average Pooling Layer)	512
Projection Layers	Fully-connected Layer	1024
	Batch Normalization 1D Layer	
	ReLU	
	Fully-connected Layer	1024
	Batch Normalization 1D Layer	
	ReLU Activation	
	Dropout Layer	0.5
	Fully-connected Layer	1024
	Batch Normalization 1D Layer	
	ReLU Activation	
Adaptation Layers	Dropout Layer	0.5
	Fully-connected Layer	1024
	Batch Normalization 1D Layer	
	ReLU Activation	
	Dropout Layer	0.5
	Fully-connected Layer	512
	Batch Normalization 1D Layer	
	ReLU Activation	
	CMD	
Class Output Layer	Fully-connected Layer	5
MMD		
Class Output Layer	Fully-connected Layer	5

Table 2: MDD classifier architecture for CMD and MMD.

MDD Classifier		
	Layers	Values
Backbone Output Layer	CMD/MMD-Method Adaptation Layer	512
Projection Layers	Fully-connected Layer	512
	Batch Normalization 1D Layer	
	ReLU	
	Fully-connected Layer	512
	Batch Normalization 1D Layer	
	ReLU Activation	
	Fully-connected Layer	512
	Batch Normalization 1D Layer	
	ReLU Activation	
	Class Output Layer	
Class Output Layer	Fully-connected Layer	5

bound introduced in [6]. All the methods (DANN, MMD and CMD) are repeated 10 times for each parameter α . The if-statement in the BPDA in Algorithm 1 is considered violated, if there is a violation of the statement for at least one of the repetitions. For a fair comparison, the evaluations of IWV and DEV are also based on 10 repetitions. More precisely, for IWV and DEV, we choose the parameter with the lowest average importance weighted risk and lowest average DEV-risk, respectively, where the average is computed over all 10 repetitions.

MiniDomainNet IWV, DEV, and BPDA are used to choose the best parameter $\alpha \in \{0, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$, for two different distance-regularized domain adaptation methods, namely MMD, and CMD. Since the MiniDomainNet dataset contains 5 classes, we use the bound proposed in [10] to instantiate the BPDA. The training procedure is the same as for Transformed Moons and Amazon Reviews.

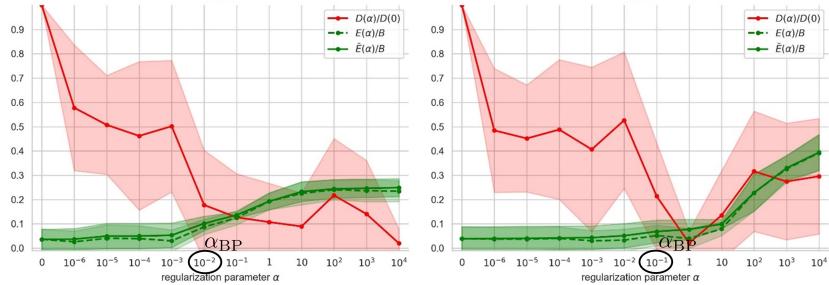


Figure 2: Average and standard deviation over 10 repetitions of estimated learning errors E (in unsupervised domain adaptation unknown) and the domain distance D of the accompanying target error bound [6] for distance-regularized domain adaptation models with the Maximum Mean Discrepancy [20] (left) and the Central Moment Discrepancy (right).

5.4 Results

This section provides empirical evidence for the compliance of the empirical settings with our assumptions made in Section 4.1 of the main document, and, gives results on the three aforementioned datasets, comparing our approach with the state of the art in parameter selection for domain adaptation.

Verification of assumptions Figure 2 shows the behaviour of the curves $D(\alpha)/D(0)$, $E(\alpha)/B$ and $\bar{E}(\alpha)/B$ for the two methods CMD and MMD and the Transformed Moons dataset. The following observations can be made.

- E is bounded, the mean curves of $D(\alpha)$ tend to be non-increasing and the technical assumption $D(\alpha)/D(0) < \sup_{\alpha \in [0, \infty)} E(\alpha)/B$ is satisfied for $\alpha \geq 10^3$. That is, the mean curves tend to follow the assumptions.
- The mean curves \bar{E} and E tend to be similar. That is, the risk which is described in the main document when considering label shift, does not apply.
- The average parameters chosen by the balancing principle (see α_{BP} in Figure) are the maximum values for which the mean curves of $E(\alpha)/B$ are smaller than the mean curves of $D(\alpha)/D(0)$. That is, the BPDA described in Algorithm 1 tends to perform as expected.
- There is a small trend towards a violation of the monotonicity assumption for D in the right sub-figure for CMD. However, the BPDA is (on average) robust w.r.t. this violation as it picks nearly the optimal value. The corresponding numbers can be found in Table 2.

Transformed Moons The results are provided in Table 3. It can be observed that BPDA achieves the lowest average classification error among all methods, over all domain adaptation techniques.

Table 3: Average target classification error (and standard deviation) for different regularization parameter choices on the Transformed Moons dataset. 10 repetitions with different random initialization of model weights are used to estimate the importance weighted risk, the DEV risk and the BPDA. The BPDA is computed using the bound in [6].

Method	SO	IWV	DEV	BPDA	TB
MMD	0.205 (± 0.025)	0.199 (± 0.031)	0.339 (± 0.065)	0.157 (± 0.069)	0.157 (± 0.069)
DANN	0.177 (± 0.032)	0.177 (± 0.032)	0.169 (± 0.075)	0.115 (± 0.098)	0.115 (± 0.098)
CMD	0.205 (± 0.026)	0.198 (± 0.022)	0.190 (± 0.051)	0.185 (± 0.039)	0.181 (± 0.038)
Avg.	0.196 (± 0.028)	0.191 (± 0.028)	0.232 (± 0.064)	0.152 (± 0.069)	0.151 (± 0.068)

Amazon Reviews Table 4 shows the results of three model selection methods IWV, DEV, and BPDA which are used to choose the best parameter α in the sequence of $0, 10^{-6}, 10^{-5}, \dots, 10^3, 10^4$,

for three different distance-regularized domain adaptation methods, namely DANN, MMD, and CMD. The 4 domains contained in Amazon Reviews are denoted in the tables as: books (B), DVDs (D), electronics (E), and kitchen appliances (K). As can be seen, our method achieves the lowest averaged classification error across all tasks using the MMD method. These results are consistent across all domain adaptation techniques.

MiniDomainNet Table 5 shows the results of the three model selection methods IWV, DEV, and BPDA. We omitted the experiments with DANN on the MiniDomainNet dataset due to our computational limits. The 6 domains in the MiniDomainNet are denoted in the tables as: Quickdraw (Q), Real (R), Sketch (S), Clipart (C), Infograph (I), and Painting (P). Since the source domain is always a combination of all the other domains except the target, we refer to the source as Combined Source (CS). As can be seen, our method achieves the lowest averaged classification error across all tasks using the CMD method. When using MMD, the three parameter selection methods perform very similar on average, with BPDA and DEV achieving the lowest average error across all tasks.

6 Discussion of risks and limitations

A constructive discussion with anonymous reviewers resulted in the following list of risks and limitations of the proposed BPDA method.

Label shift In this work, we do not assume a unique labeling function $l_S = l_T$ for source and target domain (covariate shift assumption), but l_S and l_T should be similar. In fact, even if the labeling functions are different $l_S \neq l_T$, the balancing value α^* can be well estimated by the balancing principle estimate α_{BP} as proven by Lemma 1. However, the quality of the balancing value α^* itself can be negatively affected by excessive label shift. For example, the function E can first increase strongly and then decrease (caused by label shift) which implies an increasing difference between $\bar{E}(\alpha)$ and $E(\alpha)$ for increasing α . In such situations, the target error of the model $g_{\alpha^*} \circ \phi_{\alpha^*}$ can be high and consequently also the one of the model $g_{\alpha_{BP}} \circ \phi_{\alpha_{BP}}$ identified by the BPDA. However, note that our experiments (see also Figure 2) indicate similar values for E and \bar{E} .

Loose instantiation bound Our model is agnostic w.r.t. the property that different target error bounds can be chosen as a basis. However, it is known in inverse problem literature [21, 3] that a loose bound can lead to a low performance of the balancing principle which also holds for the BPDA. This problem can be approached by choosing target error bounds that take into account the specific domain shift situation, e.g. the bound [22] is suitable for general domain shift scenarios.

Low performance of all models Situations exist which hinder distance-regularized domain adaptation methods to perform well. One such situation is excessive label shift as discussed above, see also [23] and references therein. Such scenarios can cause all models f_1, \dots, f_w to be inaccurate or unstable. The BPDA will select a model $f_i \in \{f_1, \dots, f_n\}$ with low target error $\varepsilon_T(f_i)$ compared to other models in the set $\{f_1, \dots, f_n\}$. Nevertheless, in such situations, f_i might have a high target error $\varepsilon_T(f_i)$ or it might be unstable. As a solution, distance-regularized domain adaptation methods can be applied with modifications, see e.g. [22, 24, 25, 26].

Focus on weight parameter Our theoretical guarantees and the high empirical performance come at the cost of focusing the selection process on the distance-penalizing parameter α . This is in contrast to other model selection methods, such as [16, 11, 27, 28, 29], which can select different types of parameters. This limitation can be approached by employing ideas from multipenalty regularization of inverse problems to combine the BPDA with related approaches [30]. One approach is to explore a grid of admissible values of several parameters by applying the balancing principle with respect to one of them and allowing others to take all corresponding grid values. As the result of such an application, one obtains a much reduced set of parameter combinations of interest and one can apply the balancing principle or related approaches w.r.t. the other parameters. See e.g. [31] for a recent application of this approach.

Acknowledgments and Disclosure of Funding

We thank Markus Holzleitner for careful proofreading. We thank the anonymous reviewers for a constructive discussion which formed the basis for the supplementary material's section about limitations, risks and future developments. The research reported in this paper has been funded by the Federal Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK), the Federal Ministry for Digital and Economic Affairs (BMDW), and the Province of Upper Austria in the frame of the COMET–Competence Centers for Excellent Technologies Programme and the COMET Module S3AI managed by the Austrian Research Promotion Agency FFG. The LIT AI Lab is financed by the Federal State of Upper Austria. We further acknowledge the partial support of the FFG project AutoQual-I. The ELLIS Unit Linz, the LIT AI Lab and the Institute for Machine Learning, are supported by the Federal State Upper Austria. IARAI is supported by Here Technologies. We thank the projects AI-MOTION (LIT-2018-6-YOU-212), DeepToxGen (LIT-2017-3-YOU-003), AI-SNN (LIT-2018-6-YOU-214), DeepFlood (LIT-2019-8-YOU-213), Medical Cognitive Computing Center (MC3), INCONTROL-RL (FFG-881064), PRIMAL (FFG-873979), S3AI (FFG-872172), DL for GranularFlow (FFG-871302), AIRI FG 9-N (FWF-36284, FWF-36235), ELISE (H2020-ICT-2019-3 ID: 951847), AIDD (MSCA-ITN-2020 ID: 956832). We thank Janssen Pharmaceutica (MaDeSmart, HBC.2018.2287), Audi.JKU Deep Learning Center, TGW LOGISTICS GROUP GMBH, Silicon Austria Labs (SAL), FILL Gesellschaft mbH, Anyline GmbH, Google, ZF Friedrichshafen AG, Robert Bosch GmbH, UCB Biopharma SRL, Merck Healthcare KGaA, Verbund AG, TÜV Austria, and the NVIDIA Corporation. This material is based upon work supported by the Google Cloud Research Credits program with the award GCP19980904.

Table 4: Average target classification error (and standard deviation) for different regularization parameter choices on the Amazon Reviews dataset. 10 repetitions with different random initialization of model weights are used to estimate the importance weighted risk, the DEV risk and the BPDA. The BPDA is computed using the bound in [6].

Task	MMD				
	SO	IWV	DEV	BPDA	TB
B→D	0.225 (± 0.004)	0.190 (± 0.004)	0.211 (± 0.005)	0.190 (± 0.004)	0.190 (± 0.004)
B→E	0.307 (± 0.010)	0.307 (± 0.010)	0.211 (± 0.005)	0.221 (± 0.008)	0.206 (± 0.012)
B→K	0.266 (± 0.004)	0.185 (± 0.009)	0.266 (± 0.004)	0.185 (± 0.009)	0.185 (± 0.009)
D→B	0.278 (± 0.008)	0.240 (± 0.007)	0.268 (± 0.006)	0.243 (± 0.006)	0.230 (± 0.007)
D→E	0.273 (± 0.004)	0.273 (± 0.007)	0.249 (± 0.007)	0.207 (± 0.004)	0.189 (± 0.008)
D→K	0.266 (± 0.004)	0.266 (± 0.004)	0.197 (± 0.006)	0.197 (± 0.006)	0.187 (± 0.007)
E→B	0.306 (± 0.003)	0.306 (± 0.003)	0.310 (± 0.007)	0.295 (± 0.013)	0.282 (± 0.014)
E→D	0.307 (± 0.007)	0.285 (± 0.006)	0.288 (± 0.009)	0.264 (± 0.016)	0.255 (± 0.020)
E→K	0.162 (± 0.004)	0.145 (± 0.003)	0.193 (± 0.004)	0.145 (± 0.003)	0.145 (± 0.003)
K→B	0.337 (± 0.007)	0.337 (± 0.007)	0.334 (± 0.006)	0.290 (± 0.010)	0.261 (± 0.010)
K→D	0.293 (± 0.005)	0.294 (± 0.007)	0.306 (± 0.007)	0.268 (± 0.010)	0.235 (± 0.014)
K→E	0.167 (± 0.002)	0.169 (± 0.004)	0.167 (± 0.002)	0.169 (± 0.004)	0.145 (± 0.002)
Avg.	0.266 (± 0.005)	0.249 (± 0.005)	0.250 (± 0.005)	0.223 (± 0.008)	0.209 (± 0.009)

Task	DANN				
	SO	IWV	DEV	BPDA	TB
B→D	0.228 (± 0.003)	0.220 (± 0.011)	0.509 (± 0.001)	0.233 (± 0.053)	0.220 (± 0.011)
B→E	0.322 (± 0.009)	0.327 (± 0.007)	0.498 (± 0.000)	0.313 (± 0.081)	0.235 (± 0.017)
B→K	0.276 (± 0.003)	0.296 (± 0.010)	0.272 (± 0.013)	0.247 (± 0.103)	0.219 (± 0.023)
D→B	0.290 (± 0.006)	0.290 (± 0.009)	0.253 (± 0.045)	0.253 (± 0.045)	0.245 (± 0.008)
D→E	0.284 (± 0.004)	0.274 (± 0.003)	0.299 (± 0.007)	0.252 (± 0.084)	0.221 (± 0.013)
D→K	0.270 (± 0.004)	0.300 (± 0.006)	0.303 (± 0.007)	0.217 (± 0.008)	0.217 (± 0.008)
E→B	0.312 (± 0.005)	0.310 (± 0.005)	0.312 (± 0.005)	0.372 (± 0.056)	0.310 (± 0.005)
E→D	0.317 (± 0.007)	0.313 (± 0.006)	0.313 (± 0.006)	0.327 (± 0.075)	0.277 (± 0.031)
E→K	0.170 (± 0.004)	0.170 (± 0.004)	0.170 (± 0.004)	0.172 (± 0.011)	0.170 (± 0.004)
K→B	0.345 (± 0.006)	0.337 (± 0.006)	0.338 (± 0.017)	0.314 (± 0.053)	0.314 (± 0.053)
K→D	0.313 (± 0.003)	0.360 (± 0.005)	0.360 (± 0.005)	0.298 (± 0.053)	0.296 (± 0.019)
K→E	0.174 (± 0.002)	0.183 (± 0.003)	0.221 (± 0.004)	0.194 (± 0.057)	0.172 (± 0.010)
Avg.	0.275 (± 0.005)	0.282 (± 0.006)	0.321 (± 0.010)	0.266 (± 0.057)	0.241 (± 0.017)

Task	CMD				
	SO	IWV	DEV	BPDA	TB
B→D	0.230 (± 0.011)	0.193 (± 0.006)	0.231 (± 0.007)	0.193 (± 0.006)	0.193 (± 0.006)
B→E	0.319 (± 0.013)	0.309 (± 0.010)	0.308 (± 0.009)	0.218 (± 0.011)	0.218 (± 0.011)
B→K	0.269 (± 0.005)	0.230 (± 0.007)	0.269 (± 0.006)	0.186 (± 0.010)	0.186 (± 0.010)
D→B	0.290 (± 0.015)	0.258 (± 0.009)	0.245 (± 0.008)	0.228 (± 0.006)	0.228 (± 0.006)
D→E	0.280 (± 0.009)	0.267 (± 0.007)	0.280 (± 0.006)	0.203 (± 0.007)	0.203 (± 0.007)
D→K	0.264 (± 0.004)	0.194 (± 0.006)	0.194 (± 0.006)	0.194 (± 0.006)	0.194 (± 0.006)
E→B	0.314 (± 0.009)	0.307 (± 0.006)	0.302 (± 0.005)	0.279 (± 0.010)	0.279 (± 0.010)
E→D	0.320 (± 0.020)	0.287 (± 0.006)	0.287 (± 0.006)	0.258 (± 0.014)	0.258 (± 0.014)
E→K	0.174 (± 0.013)	0.152 (± 0.005)	0.169 (± 0.006)	0.139 (± 0.005)	0.139 (± 0.005)
K→B	0.346 (± 0.022)	0.264 (± 0.007)	0.331 (± 0.006)	0.264 (± 0.007)	0.264 (± 0.007)
K→D	0.314 (± 0.013)	0.248 (± 0.006)	0.248 (± 0.006)	0.248 (± 0.006)	0.248 (± 0.006)
K→E	0.178 (± 0.007)	0.147 (± 0.004)	0.178 (± 0.007)	0.147 (± 0.004)	0.147 (± 0.004)
Avg.	0.275 (± 0.012)	0.238 (± 0.007)	0.254 (± 0.007)	0.213 (± 0.008)	0.213 (± 0.008)

Table 5: Average target classification error (and standard deviation) for different regularization parameter choices on the MiniDomainNet dataset. 3 repetitions with different random initialization of model weights are used to estimate the importance weighted risk, the DEV risk and the BPDA. The BPDA is computed using the bound in [10].

Task	MMD				
	SO	IWV	DEV	BPDA	TB
CS→Q	0.568(±0.007)	0.629(±0.023)	0.629(±0.023)	0.629(±0.023)	0.568(±0.007)
CS→R	0.068(±0.009)	0.102(±0.020)	0.102(±0.020)	0.098(±0.006)	0.068(±0.009)
CS→S	0.309(±0.010)	0.307(±0.001)	0.324(±0.021)	0.296(±0.012)	0.296(±0.012)
CS→C	0.246(±0.016)	0.264(±0.013)	0.264(±0.013)	0.264(±0.013)	0.246(±0.016)
CS→I	0.605(±0.012)	0.577(±0.004)	0.564(±0.001)	0.589(±0.021)	0.564(±0.001)
CS→P	0.178(±0.006)	0.212(±0.012)	0.202(±0.011)	0.214(±0.007)	0.178(±0.006)
Avg.	0.329(±0.010)	0.349(±0.012)	0.348(±0.015)	0.348(±0.014)	0.320(±0.009)

Task	CMD				
	SO	IWV	DEV	BPDA	TB
CS→Q	0.568(±0.007)	0.568(±0.007)	0.812(±0.000)	0.410(±0.008)	0.410(±0.008)
CS→R	0.068(±0.009)	0.068(±0.009)	0.841(±0.000)	0.100(±0.010)	0.068(±0.009)
CS→S	0.309(±0.010)	0.305(±0.012)	0.875(±0.000)	0.298(±0.005)	0.298(±0.005)
CS→C	0.246(±0.016)	0.257(±0.023)	0.850(±0.000)	0.282(±0.032)	0.246(±0.016)
CS→I	0.605(±0.012)	0.556(±0.031)	0.883(±0.000)	0.601(±0.016)	0.556(±0.031)
CS→P	0.178(±0.006)	0.246(±0.022)	0.986(±0.000)	0.293(±0.019)	0.178(±0.006)
Avg.	0.329(±0.010)	0.333(±0.017)	0.875(±0.000)	0.331(±0.015)	0.293(±0.013)

References

- [1] Gord Sinnamon. Transferring monotonicity in weighted norm inequalities. *Collectanea Mathematica*, pages 181–216, 2003.
- [2] OV Lepskii. On a problem of adaptive estimation in gaussian white noise. *Theory of Probability & Its Applications*, 35(3):454–466, 1991.
- [3] Ernesto De Vito, Sergei V Pereverzyev, and Lorenzo Rosasco. Adaptive kernel methods using the balancing principle. *Foundations of Computational Mathematics*, 10(4):455–479, 2010.
- [4] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1406–1415, 2019.
- [5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [6] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, 2010.
- [7] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(Jan):1–35, 2016.
- [8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.
- [9] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.
- [10] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. Bridging theory and algorithm for domain adaptation. In *Proceedings of the International Conference on Machine Learning*, pages 7404–7413, 2019.
- [11] Kaichao You, Ximei Wang, Mingsheng Long, and Michael Jordan. Towards accurate model selection in deep unsupervised domain adaptation. In *Proceedings of the International Conference on Machine Learning*, pages 7124–7133, 2019.
- [12] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.
- [13] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.
- [14] Casper O da Costa-Luis. tqdm: A fast, extensible progress meter for python and cli. *Journal of Open Source Software*, 4(37):1277, 2019.
- [15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [16] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul V Buenau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in Neural Information Processing Systems*, pages 1433–1440, 2008.

- [17] Arthur Gretton, Karsten M Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. A kernel method for the two-sample-problem. In *Advances in Neural Information Processing Systems*, pages 513–520, 2006.
- [18] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *Proceedings of the International Conference on Machine Learning*, pages 97–105, 2015.
- [19] Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. Central moment discrepancy (cmd) for domain-invariant representation learning. *International Conference on Learning Representations*, 2017.
- [20] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(3):723–773, 2012.
- [21] Shuai Lu and Sergei V Pereverzev. *Regularization theory for ill-posed problems: selected topics*, volume 58. Walter de Gruyter, 2013.
- [22] Remi Tachet des Combes, Han Zhao, Yu-Xiang Wang, and Geoffrey J Gordon. Domain adaptation with conditional distribution matching and generalized label shift. *Advances in Neural Information Processing Systems*, 33, 2020.
- [23] Han Zhao, Remi Tachet Des Combes, Kun Zhang, and Geoffrey Gordon. On learning invariant representations for domain adaptation. In *Proceedings of the International Conference on Machine Learning*, pages 7523–7532, 2019.
- [24] Kamyar Azizzadenesheli, Anqi Liu, Fanny Yang, and Animashree Anandkumar. Regularized learning for domain adaptation under label shifts. *International Conference on Learning Representations*, 2019.
- [25] Trung Le, Tuan Nguyen, Nhat Ho, Hung Bui, and Dinh Phung. Lamda: Label matching deep domain adaptation. In *Proceedings of the International Conference on Machine Learning*, pages 6043–6054, 2021.
- [26] Yifan Wu, Ezra Winston, Divyansh Kaushik, and Zachary Lipton. Domain adaptation with asymmetrically-relaxed distribution alignment. In *Proceedings of the International Conference on Machine Learning*, pages 6872–6881, 2019.
- [27] Wouter M Kouw, Jesse H Krijthe, and Marco Loog. Robust importance-weighted cross-validation under sample selection bias. In *IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6. IEEE, 2019.
- [28] Erheng Zhong, Wei Fan, Qiang Yang, Olivier Verscheure, and Jiagtao Ren. Cross validation framework to choose amongst models and datasets for transfer learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 547–562. Springer, 2010.
- [29] Lorenzo Bruzzone and Mattia Marconcini. Domain adaptation problems: A dasvm classification technique and a circular validation strategy. *IEEE transactions on pattern analysis and machine intelligence*, 32(5):770–787, 2009.
- [30] Massimo Fornasier, Valeriya Naumova, and Sergei V Pereverzyev. Parameter choice strategies for multipenalty regularization. *SIAM Journal on Numerical Analysis*, 52(4):1770–1794, 2014.
- [31] Mykola Krashchuk, Sergei Pereverzyev, Sergii V Siryk, and Nataliya Vasylyeva. Determination of the fractional order in semilinear subdiffusion equations. *Fractional Calculus and Applied Analysis*, 23(3):694–722, 2020.

2.2 Addressing Parameter Choice Issues in Unsupervised Domain Adaptation by Aggregation

ADDRESSING PARAMETER CHOICE ISSUES IN UNSUPERVISED DOMAIN ADAPTATION BY AGGREGATION

Marius-Constantin Dinu^{1,2} Markus Holzleitner¹ Maximilian Beck¹

Duc Hoan Nguyen⁵ Andrea Huber¹ Hamid Eghbal-zadeh¹ Bernhard A. Moser⁴

Sergei V. Pereverzyev⁵ Sepp Hochreiter^{1,3} Werner Zellinger⁵

¹ELLIS Unit Linz and LIT AI Lab, Institute for Machine Learning, Johannes Kepler University Linz

²Dynatrace Research

³Institute of Advanced Research in Artificial Intelligence

⁴Software Competence Center Hagenberg

⁵Johann Radon Institute for Computational and Applied Mathematics, Austrian Academy of Sciences

ABSTRACT

We study the problem of choosing algorithm hyper-parameters in unsupervised domain adaptation, i.e., with labeled data in a source domain and unlabeled data in a target domain, drawn from a different input distribution. We follow the strategy to compute several models using different hyper-parameters, and, to subsequently compute a linear aggregation of the models. While several heuristics exist that follow this strategy, methods are still missing that rely on thorough theories for bounding the target error. In this turn, we propose a method that extends weighted least squares to vector-valued functions, e.g., deep neural networks. We show that the target error of the proposed algorithm is asymptotically not worse than twice the error of the unknown optimal aggregation. We also perform a large scale empirical comparative study on several datasets, including text, images, electroencephalogram, body sensor signals and signals from mobile phones. Our method¹ outperforms deep embedded validation (DEV) and importance weighted validation (IWV) on all datasets, setting a new state-of-the-art performance for solving parameter choice issues in unsupervised domain adaptation with theoretical error guarantees. We further study several competitive heuristics, all outperforming IWV and DEV on at least five datasets. However, our method outperforms each heuristic on at least five of seven datasets.

1 INTRODUCTION

The goal of *unsupervised domain adaptation* is to learn a model on unlabeled data from a *target* input distribution using labeled data from a different *source* distribution (Pan & Yang, 2010; Ben-David et al., 2010). If this goal is achieved, medical diagnostic systems can successfully be trained on unlabeled images using labeled images with a different modality (Varsavsky et al., 2020; Zou et al., 2020); segmentation models for natural images can be learned using only labeled data from computer simulations Peng et al. (2018); natural language models can be learned from unlabeled biomedical abstracts by means of labeled data from financial journals (Blitzer et al., 2006); industrial quality inspection systems can be learned on unlabeled data from new products using data from related products (Jiao et al., 2019; Zellinger et al., 2020).

However, missing target labels combined with distribution shift makes parameter choice a hard problem (Sugiyama et al., 2007; You et al., 2019; Saito et al., 2021; Zellinger et al., 2021; Musgrave et al., 2021). Often, one ends up with a sequence of models, e.g., originating from different hyper-parameter configurations (Ben-David et al., 2007; Saenko et al., 2010; Ganin et al., 2016; Long et al.,

¹Large scale benchmark experiments are available at <https://github.com/Xpitfire/iwa>; dinu@ml.jku.at, werner.zellinger@ricam.oeaw.ac.at

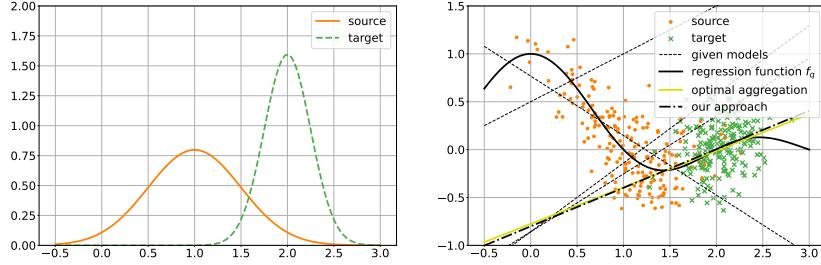


Figure 1: Unsupervised domain adaptation problem (Shimodaira, 2000; Sugiyama et al., 2007; You et al., 2019). Left: Source distribution (solid) and target distribution (dashed). Right: A sequence of different linear models (dashed) is used to find the optimal linear aggregation of the models (solid). Model selection methods (Sugiyama et al., 2007; Kouw et al., 2019; You et al., 2019; Zellinger et al., 2021) cannot outperform the best single model in the sequence, confidence values as used in Zou et al. (2018) are not available, and, approaches based on averages or tendencies of majorities of models (Saito et al., 2017) suffer from a high fraction of large-error-models in the sequence. In contrast, our approach (dotted-dashed) is nearly optimal. In addition, the model computed by our method provably approaches the optimal linear aggregation for increasing sample size. For further details on this example we refer to Section C in the Supplementary Material.

2015; Zellinger et al., 2017; Peng et al., 2019). In this work, we study the problem of constructing an optimal aggregation using all models in such a sequence. Our main motivation is that the error of such an optimal aggregation is clearly smaller than the error of the best single model in the sequence.

Although methods with mathematical error guarantees have been proposed to select the best model in the sequence (Sugiyama et al., 2007; Kouw et al., 2019; You et al., 2019; Zellinger et al., 2021), methods for learning aggregations of the models are either heuristics or their theory guarantees are limited by severe assumptions (cf. Wilson & Cook (2020)). Typical aggregation approaches are (a) to learn an aggregation on source data only (Nozza et al., 2016), (b) to learn an aggregation on a set of (unknown) labeled target examples (Xia et al., 2013; Dai et al., 2007; III & Marcu, 2006; Duan et al., 2012), (c) to learn an aggregation on target examples (pseudo-)labeled based on confidence measures of the given models (Zhou et al., 2021; Ahmed et al., 2022; Sun, 2012; Zou et al., 2018; Saito et al., 2017), (d) to aggregate the models based on data-structure specific transformations (Yang et al., 2012; Ha & Youn, 2021), and, (e) to use specific (possibly not available) knowledge about the given models, such as information obtained at different time-steps of its gradient-based optimization process (French et al., 2018; Laine & Aila, 2017; Tarvainen & Valpola, 2017; Athiwaratkun et al., 2019; Al-Stouhi & Reddy, 2011) or the information that the given models are trained on different (source) distributions (Hoffman et al., 2018; Rakshit et al., 2019; Xu et al., 2018; Kang et al., 2020; Zhang et al., 2015). One problem shared among all methods mentioned above is that they cannot guarantee a small error, even if the sample size grows to infinity. See Figure 1 for a simple illustrative example.

In this work, we propose (to the best of our knowledge) the first algorithm for computing aggregations of vector-valued models for unsupervised domain adaptation with target error guarantees. We extend the *importance weighted least squares algorithm* (Shimodaira, 2000) and corresponding recently proposed error bounds (Gizewski et al., 2022) to linear aggregations of vector-valued models. The importance weights are the values of an estimated ratio between target and source density evaluated at the examples. Every method for density-ratio estimation can be used as a basis for our approach, e.g. Sugiyama et al. (2012); Kanamori et al. (2012) and references therein. Our error bound proves that the target error of the computed aggregation is asymptotically at most twice the target error of the optimal aggregation.

In addition, we perform extensive empirical evaluations on several datasets with academic data (Transformed Moons), text data (Amazon Reviews (Blitzer et al., 2006)), images (MiniDomainNet (Peng et al., 2019; Zellinger et al., 2021)), electroencephalography signals (Sleep-EDF (Eldele et al., 2021; Goldberger et al., 2000)), body sensor signals (UCI-HAR (Anguita et al., 2013), WISDM (Kwapisz et al., 2011)), and, sensor signals from mobile phones and smart watches (HHAR (Stisen et al., 2015)).

We compute aggregations of models obtained from different hyper-parameter settings of 11 domain adaptation methods (e.g., DANN (Ganin et al., 2016) and Deep-Coral Sun & Saenko (2016)). Our method sets a new state of the art for methods with theoretical error guarantees, namely importance weighted validation (IWV) (Sugiyama et al., 2007) and deep embedded validation (DEV) (Kouw et al., 2019), on all datasets. We also study (1) classical least squares aggregation on source data only, (2) majority voting on target predictions, (3) averaging over model confidences, and (4) learning based on pseudo-labels. All of these heuristics outperform IWV and DEV on at least five of seven datasets, which is a result of independent interest. In contrast, our method outperforms each heuristic on at least five of seven datasets.

Our main contributions are summarized as follows:

- We propose the (to the best of our knowledge) first algorithm for ensemble learning of vector-valued models in (single-source) unsupervised domain adaptation that satisfies a non-trivial target error bound.
- We prove that the target error of our algorithm is asymptotically (for increasing sample sizes) at most twice the target error of the unknown optimal aggregation.
- We outperform IWV and DEV, and therefore set a new state-of-the-art performance for re-solving parameter choice issues under theoretical target error guarantees.
- We describe four heuristic baselines which all outperform IWV and DEV on at least five of seven datasets. Our method outperforms each heuristic on at least five of seven datasets.
- Our method tends to be more stable than others w.r.t. adding inaccurate models to the given sequence of models.

2 RELATED WORK

It is well known that aggregations of models in an ensemble often outperform individual models (Dong et al., 2020; Goodfellow et al., 2016). Traditional ensemble methods that have shown the advantage of aggregation are Boosting (Schapire, 1990; Breiman, 1998), Bootstrap Aggregating (bagging) (Breiman, 1994; 1996a) and Stacking (Wolpert, 1992; Breiman, 1996b). For example, averages of multiple models pre-trained on data from a distribution different from the target one have recently been shown to achieve state-of-the-art performance on ImageNet (Wortsman et al., 2022) and their good generalization properties can be related to flat minima (Hochreiter & Schmidhuber, 1994; 1997). However, most such methods don't take into account a present distribution shift.

Although some ensemble learning methods exist, which take into account a present distribution shift, in contrast to our work, they are either relying on labeled target data (Nozza et al., 2016; Xia et al., 2013; III & Marcu, 2006; Dai et al., 2007; Mayr et al., 2016), are restricted by fixing the aggregation weights to be the same (Razari & Samothrakis, 2019), make assumptions on the models in the sequence or the corresponding process for learning the models (Yang et al., 2012; Ha & Youn, 2021; French et al., 2018; Laine & Aila, 2017; Tarvainen & Valpola, 2017; Athiwaratkun et al., 2019; Al-Stouhi & Reddy, 2011; Hoffman et al., 2018; Rakshit et al., 2019; Xu et al., 2018; Kang et al., 2020; Zhang et al., 2015), or, learn an aggregation based on the heuristic approach of (pseudo-)labeling some target data based on confidence measures of models in the sequence (Zhou et al., 2021; Ahmed et al., 2022; Sun, 2012; Zou et al., 2018; Saito et al., 2017). Another crucial difference of all methods above is that none of these methods can guarantee a small target error in the general setting (distribution shift, vector valued models, different classes, single source domain) described above, even if the sample size grows to infinity.

Another branch of research are methods which aim at selecting the best model in the sequence. Although, such methods with error bounds have been proposed for the general setting above (Sugiyama et al., 2007; You et al., 2019; Zellinger et al., 2021), they cannot overcome a limited performance of the best model in the given sequence (cf. Figure 1 and Section 6 in the Supplementary Material of Zellinger et al. (2021)). In contrast, our method can outperform the best model in the sequence, and our empirical evaluations show that this is indeed the case in practical examples. A recent kernel-based algorithm for univariate regression, that is similar to ours, can be found in Gizewski et al. (2022). However, in contrast to Gizewski et al. (2022), our method allows a much more general form of vector-valued models which are not necessarily obtained from regularized kernel least squares, and, can therefore be applied to practical deep learning tasks.

Our work employs technical tools developed in Caponnetto & De Vito (2007; 2005). In fact, we extend Caponnetto & De Vito (2007; 2005) to deal with importance weighted least squares. Finally, it is important to note Huang et al. (2006), where a core Lemma of our proofs is proposed.

3 AGGREGATION BY IMPORTANCE WEIGHTED LEAST SQUARES

This section gives a summary of the main problem of this paper and our approach. For detailed assumptions and proofs, we refer to Section A of the Supplementary Material.

Notation and Setup Let $\mathcal{X} \subset \mathbb{R}^{d_1}$ be a compact *input space* and $\mathcal{Y} \subset \mathbb{R}^{d_2}$ be a compact *label space* with inner product $\langle \cdot, \cdot \rangle_{\mathcal{Y}}$ such that for the associated norm $\|y\|_{\mathcal{Y}} \leq y_0$ holds for all $y \in \mathcal{Y}$ and some $y_0 > 0$. Following Ben-David et al. (2010), we consider two datasets: A *source dataset* $(\mathbf{x}, \mathbf{y}) = ((x_1, y_1), \dots, (x_n, y_n)) \in (\mathcal{X} \times \mathcal{Y})^n$ independently drawn according to some source distribution (probability measure) p on $\mathcal{X} \times \mathcal{Y}$ and an unlabeled *target* dataset $\mathbf{x}' = (x'_1, \dots, x'_m) \in \mathcal{X}^m$ with elements independently drawn according to the marginal distribution² $q_{\mathcal{X}}$ of some target distribution q on $\mathcal{X} \times \mathcal{Y}$. The marginal distribution of p on \mathcal{X} is analogously denoted as $p_{\mathcal{X}}$. We further denote by $\mathcal{R}_q(f) = \int_{\mathcal{X} \times \mathcal{Y}} \|f(x) - y\|_{\mathcal{Y}}^2 dq(x, y)$ the *expected target risk* of a vector valued function $f : \mathcal{X} \rightarrow \mathcal{Y}$ w.r.t. the least squares loss.

Problem Given a set $f_1, \dots, f_l : \mathcal{X} \rightarrow \mathcal{Y}$ of models, the labeled source sample (\mathbf{x}, \mathbf{y}) and the unlabeled target sample \mathbf{x}' , the problem considered in this work is to find a model $f : \mathcal{X} \rightarrow \mathcal{Y}$ with a minimal target error $\mathcal{R}_q(f)$.

Main Assumptions We rely (a) on the *covariate shift* assumption that the source conditional distribution $p(y|x)$ equals the target conditional distribution $q(y|x)$, and, (b) on the *bounded density ratio* assumption that there is a function $\beta : \mathcal{X} \rightarrow [0, B]$ with $B > 0$ such that $dq_{\mathcal{X}}(x) = \beta(x) dp_{\mathcal{X}}(x)$.

Approach Our goal is to compute the linear aggregation $f = \sum_{i=1}^l c_i f_i$ for $c_1, \dots, c_l \in \mathbb{R}$ with minimal squared target risk $\mathcal{R}_q\left(\sum_{i=1}^l c_i f_i\right)$. Our approach relies on the fact that

$$\arg \min_{c_1, \dots, c_l \in \mathbb{R}} \mathcal{R}_q\left(\sum_{i=1}^l c_i f_i\right) = \arg \min_{c_1, \dots, c_l \in \mathbb{R}} \int_{\mathcal{X}} \left\| \sum_{i=1}^l c_i f_i(x) - f_q(x) \right\|_{\mathcal{Y}}^2 dq_{\mathcal{X}}(x) \quad (1)$$

for the *regression functions* given by $f_q(x) = \int_{\mathcal{Y}} y dq(y|x)$ ³, see e.g. Cucker & Smale (2002, Proposition 1). Unfortunately, the right hand side of Eq. (1) contains information about labels $f_q(x)$ which are not given in our setting of unsupervised domain adaptation. However, borrowing an idea from *importance sampling*, it is possible to estimate Eq. (1). More precisely, from the covariate shift assumption we get $f_p(x) = \int_{\mathcal{Y}} y dp(y|x) = f_q(x)$ and we can use the bounded density ratio β to obtain

$$\arg \min_{c_1, \dots, c_l \in \mathbb{R}} \mathcal{R}_q\left(\sum_{i=1}^l c_i f_i\right) = \arg \min_{c_1, \dots, c_l \in \mathbb{R}} \int_{\mathcal{X}} \beta(x) \left\| \sum_{i=1}^l c_i f_i(x) - f_p(x) \right\|_{\mathcal{Y}}^2 dp_{\mathcal{X}}(x) \quad (2)$$

which extends importance weighted least squares (Shimodaira, 2000; Kanamori et al., 2009) to linear aggregations $\sum_{i=1}^l c_i f_i$ of vector-valued functions f_1, \dots, f_l . The unique minimizer of Eq. (2) can be approximated based on available data analogously to classical least squares estimation as detailed in Algorithm 1. In the following, we call Algorithm 1 Importance Weighted Least Squares Linear Aggregation (IWA).

²The existence of the conditional probability density $q(y|x)$ with $q(x, y) = q(y|x)q_{\mathcal{X}}(x)$ is guaranteed by the fact that $\mathcal{X} \times \mathcal{Y}$ is Polish, i.e., a separable and complete metric space, c.f. Dudley (2002, Theorem 10.2.2.).

³ \mathcal{Y} -valued integrals are defined in the sense of Lebesgue-Bochner.

Relation to Model Selection The optimal aggregation $f^* := \arg \min_{c_1, \dots, c_l \in \mathbb{R}} \mathcal{R}_q \left(\sum_{i=1}^l c_i f_i \right)$ defined in Eq. (2) is clearly better than any single model selection since

$$\mathcal{R}_q(f^*) = \min_{c_1, \dots, c_l \in \mathbb{R}} \mathcal{R}_q \left(\sum_{i=1}^l c_i f_i \right) \leq \min_{c_1, \dots, c_l \in \{0, 1\}} \mathcal{R}_q \left(\sum_{i=1}^l c_i f_i \right) \leq \min_{f_1, \dots, f_l} \mathcal{R}_q(f_i). \quad (3)$$

However, the optimal aggregation f^* cannot be computed based on finite datasets and the next logical questions are about the accuracy of the approximation \tilde{f} in Algorithm 1.

Algorithm 1: Importance Weighted Least Squares Linear Aggregation (IWA).

Input : Set $f_1, \dots, f_l : \mathcal{X} \rightarrow \mathcal{Y}$ of models, labeled source sample (\mathbf{x}, \mathbf{y}) and unlabeled target sample \mathbf{x}' .

Output : Linear aggregation $\tilde{f} = \sum_{i=1}^l \tilde{c}_i f_i$ with weights $\tilde{c} = (\tilde{c}_1, \dots, \tilde{c}_l) \in \mathbb{R}^l$.

Step 1 Use unlabeled samples \mathbf{x} and \mathbf{x}' to approximate density ratio $\frac{d q_{\mathcal{X}}}{d p_{\mathcal{X}}}$ by some function $\beta : \mathcal{X} \rightarrow [0, B]$ using a classical algorithm, e.g. Sugiyama et al. (2012).

Step 2 Compute weight vector $\tilde{c} = \tilde{G}^{-1} \tilde{g}$ with empirical Gram matrix \tilde{G} and vector \tilde{g} defined by

$$\tilde{G} = \left(\frac{1}{m} \sum_{k=1}^m \langle f_i(x'_k), f_j(x'_k) \rangle_{\mathcal{Y}} \right)_{i,j=1}^l \quad \tilde{g} = \left(\frac{1}{n} \sum_{k=1}^n \beta(x_k) \langle y_k, f_i(x_k) \rangle_{\mathcal{Y}} \right)_{i=1}^l.$$

Return : Linear aggregation $\tilde{f} = \sum_{i=1}^l \tilde{c}_i f_i$.

4 TARGET ERROR BOUND FOR ALGORITHM 1

Let us start by introducing some further notation: $L^2(p)$ refers to the Lebesgue-Bochner space of functions from \mathcal{X} to \mathcal{Y} , associated to a measure p on \mathcal{X} with corresponding inner product $\langle \cdot, \cdot \rangle_{L^2(p)}$ (this space basically consists of all \mathcal{Y} -valued functions whose \mathcal{Y} -norms are square integrable with respect to the given measure p). Moreover, let us introduce the (positive semi-definite) Gram matrix $G = \left(\langle f_i, f_j \rangle_{L^2(q_{\mathcal{X}})} \right)_{i,j=1}^l$ and the vector $\bar{g} = \left(\langle \beta f_p, f_i \rangle_{L^2(p_{\mathcal{X}})} \right)_{i=1}^l$. We can assume that G is invertible (and thus positive definite), since otherwise some models are too similar to others and can be withdrawn from consideration (see Section D). Next, we recall that the minimizer of Eq. (2) is $c^* = (c_1^*, \dots, c_l^*) = G^{-1} \bar{g}$, see Lemma 4.

However, neither G nor the vector \bar{g} is accessible in practice, because there is no access to the target measure $q_{\mathcal{X}}$. Driven by the law of large numbers we try to approximate them by averages over our given data and therefore arrive at the formulas for \tilde{G} and \tilde{g} given in Algorithm 1. This leads to the approximation \tilde{f} . Up to this point, we were only considering an intuitive perspective on the problem setting, therefore, we will now formally discuss statements on the distance between the model \tilde{f} and the optimal linear model $f^* = \sum_{i=1}^l c_i^* f_i$, measured in terms of target risks, and how this distance behaves with increasing sample sizes. This is what we attempt with our main result:

Theorem 1. With probability $1 - \delta$ it holds that

$$\mathcal{R}_q(\tilde{f}) - \mathcal{R}_q(f_q) \leq 2(\mathcal{R}_q(f^*) - \mathcal{R}_q(f_q)) + C \left(\log \frac{1}{\delta} \right) (n^{-1} + m^{-1}) \quad (4)$$

for some coefficient $C > 0$ not depending on m, n and δ , and sufficiently large m and n .

Before we give an outline of the proof (see Section A), let us briefly comment on the main message of Algorithm 1. Observe, that (Cucker & Smale, 2002, Proposition 1) $\mathcal{R}_q(f) - \mathcal{R}_q(f_q) = \|f - f_q\|_{L^2(q_{\mathcal{X}})}^2$ can be interpreted as the total target error made by Algorithm 1, sometimes called *excess risk*. Indeed, in the deterministic setting of labeling functions, f_q equals the target labeling function and the excess risk equals the target error of Ben-David et al. (2010). Eq. (4) compares this error for the aggregation \tilde{f} , computed by Algorithm 1, to the error for the optimal aggregation f^* . Note that the error of the optimal aggregation f^* is unavoidable in the sense that it is determined by

the decision of searching for linear aggregations of f_1, \dots, f_l only. However, if the models f_1, \dots, f_l are sufficiently different, then this error can be expected to be small. Theorem 1 tells us that the error of \tilde{f} approaches the one of f^* with increasing target and source sample size. The rate of convergence is at least linear. Finally, we emphasize that Theorem 1 does not take into account the error of the density-ratio estimation. We refer to the recent work Gizewski et al. (2022), who, for the first time, included such error in the analysis of importance weighted least squares.

Let us now give a brief outline for the proof of Theorem 1. One key part concerns the existence of a Hilbert space \mathcal{H} with associated inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ (a reproducing kernel space of functions from $\mathcal{X} \rightarrow \mathcal{Y}$) which contains all given models f_1, \dots, f_l and the regression function $f_q = f_p$. The space \mathcal{H} can be constructed from any given models that are bounded and continuous functions. Furthermore, Algorithm 1 does not need any knowledge of \mathcal{H} , which is a modeling assumption only needed for the proofs, so that we can apply many arguments developed in Caponnetto & De Vito (2007; 2005). \mathcal{H} is also not necessarily generated by a prescribed kernel such as Gaussian or linear kernel, and, no further smoothness assumption is required, see Sections A and B in the Supplementary Material.

Moreover, in this setting one can express the excess risk as follows: $\mathcal{R}_q(f) - \mathcal{R}_q(f_q) = \|A(f - f_q)\|_{\mathcal{H}}^2$ for some bounded linear operator $A : \mathcal{H} \rightarrow \mathcal{H}$. This also allows us to formulate the entries of G and \bar{g} in terms of the inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ instead. Using properties related to the operators that appear in the construction of \mathcal{H} , in combination with Hoeffding-like concentration bounds in Hilbert spaces and bounds that measure, e.g., the deviation between empirical averages in source and target domain (as done in Gretton et al. (2006, Lemma 4)), we can quantify differences between the entries of G and \bar{G} (and \bar{g} and \tilde{g} respectively) in terms of n, m and δ . This leads to Eq. (4).

5 EMPIRICAL EVALUATIONS

We now empirically evaluate the performance of our approach compared to classical ensemble learning baselines and state-of-the-art model selection methods. Therefore, we structure our empirical evaluation as follows. First, we outline our experimental setup for unsupervised domain adaptation and introduce all domain adaptation methods for our analysis. Second, we describe the ensemble learning and model selection baselines, and third, we present the datasets used for our experiments. We then conclude with our results and a detailed discussion thereof.

5.1 EXPERIMENTAL SETUP

To assess the performance of our ensemble learning Algorithm 1 IWA, we perform numerous experiments with different domain adaptation algorithms on different datasets. By changing the hyper-parameters of each algorithm, we obtain, as results of applying these algorithms, sequences of models. The goal of our method is to find optimal models based on combinations of candidates from each sequence. As domain adaptation algorithms, we consider the AdaTime benchmark suite, and run our experiments on language, image, text and time-series data. This suite comprises a collection of 11 domain adaptation algorithms. We follow their evaluation setup and apply the following algorithms: Adversarial

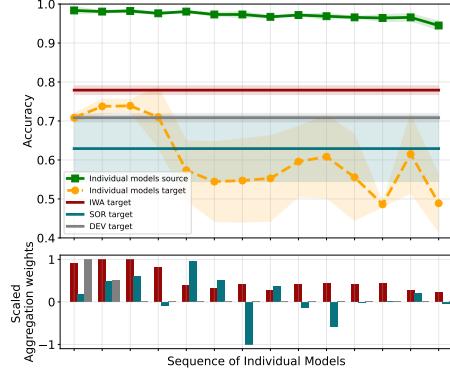


Figure 2: Top: Mean classification accuracy (y-axis) of our method (IWA), source-only regression (SOR), deep embedded validation (DEV) and individual models (green: source accuracy, orange: target accuracy) used in the aggregation for the HHAR dataset (Stisen et al., 2015) over 3 seeds. The individual models (x-axis) are trained with DIRT (Shu et al., 2018) for different hyper-parameter choices. Bottom: Scaled Aggregation weights (y-axis) for individual models (x-axis) computed by IWA, SOR and DEV (average over 3 seeds). Instead of searching for the best model in the sequence, IWA effectively uses all models in the sequence and obtains a performance not reachable by any procedure selecting only one model.

Spectral Kernel Matching (AdvSKM) (Liu & Xue, 2021), Deep Domain Confusion (DDC) (Tzeng et al., 2014), Correlation Alignment via Deep Neural Networks (Deep-Coral) (Sun et al., 2017), Central Moment Discrepancy (CMD) (Zellinger et al., 2017), Higher-order Moment Matching (HoMM) (Chen et al., 2020), Minimum Discrepancy Estimation for Deep Domain Adaptation (MMDA) (Rahman et al., 2020), Deep Subdomain Adaptation (DSAN) (Zhu et al., 2021), Domain-Adversarial Neural Networks (DANN) (Ganin et al., 2016), Conditional Adversarial Domain Adaptation (CDAN) (Long et al., 2018), A DIRT-T Approach to Unsupervised Domain Adaptation (DIRT) (Shu et al., 2018) and Convolutional deep Domain Adaptation model for Time-Series data (CoDATS) (Wilson et al., 2020). In addition to the sequence of models, IWA requires an estimate of the density ratio between source and target domain. To compute this quantity we follow (Bickel et al., 2007) and (You et al., 2019, Section 4.3), and, train a classifier discriminating between source and target data. The output of this classifier is then used to approximate the density ratio denoted as β in Algorithm 1. Overall, to compute the results in our tables we trained 16680 models over approximately a timeframe of 1500 GPU/hours using computation resources of NVIDIA P100 16GB GPUs.

For example, consider the top plot of Figure 2, where we compare the performance of Algorithm 1 to deep embedded validation (DEV) (You et al., 2019), a heuristics baseline source-only regression (SOR, see Section 5.2) and each individual model in the sequence. The bottom plot shows the scaled aggregation weights, i.e. how much each individual model contributes to the aggregated prediction of IWA, DEV, and SOR. In this example, the given sequence of models is obtained from applying the algorithm proposed in Shu et al. (2018) with different hyper-parameter choices to the Heterogeneity Human Activity Recognition dataset (Stisen et al., 2015). See Section D.3 in the Supplementary Material for the exact hyper-parameter values.

5.2 BASELINES

As representatives for the most prominent methods discussed in Section 1, we compare our method, IWA, to ensemble learning methods that use linear regression and majority voting as *heuristic* for model aggregation, and, model selection methods with *theoretical error guarantees*.

Heuristic Baselines The first baseline is majority voting on target data (TMV). It aggregates the predictions of all models by counting the overall class predictions and selects the class with the maximum prediction count as ensemble output. In addition, we implement three heuristic baselines which aggregate the vector-valued output, i.e. probabilities, of all classifiers using weights learned via linear regression. The final ensemble prediction is then made by selecting the class with the highest probability. The three heuristic regression baselines differ in the input used for the performed regression. Source-only regression (SOR) trains a regression model on classifier predictions (of the given models) and labels from the source domain only. Target majority voting regression (TMR) uses the same voting procedure as explained above to generate pseudo-labels on the target domain, which are then further used to train a linear regression model. In contrast, target confidence average regression (TCR) selects the highest average class probability over all classifiers to pseudo-label the target samples, which is then used for training the linear regression model.

Baselines with Theoretical Error Guarantees We compare IWA to the model selection methods importance weighted validation (IWW) (Sugiyama et al., 2007) and deep embedded validation (DEV) (You et al., 2019), which select models according to their (importance weighted) target risk. Both methods assume the knowledge of an estimated density ratio between target and source domains. In our experiments we follow Bickel et al. (2007); You et al. (2019) and estimate this ratio, by using a classifier that discriminates between source and target domain (see Supplementary Material Section D for more details).

5.3 DATASETS

We evaluate the previously mentioned methods according to a diverse set of datasets, including language, image and time-series data. All datasets have a train, evaluation and test split, with results only presented on the held-out test sets. For additional details we refer to Appendix C and D.

TransformedMoons This specific form of twinning moons is based on Zellinger et al. (2021). The source domain consists of two-dimensional input data points and their transformations to two opposing moon-shaped forms.

MiniDomainNet is a reduced version of DomainNet-2019 (Peng et al., 2019) consisting of six different image domains (Quickdraw, Real, Clipart, Sketch, Infograph, and Painting). In particular,

MiniDomainNet (Zellinger et al., 2021) reduces the number of classes of DomainNet-2019 to the top-five largest representatives in the training set of each class across all six domains.

AmazonReviews is based on Blitzer et al. (2006) and consists of text reviews from four domains: books, DVDs, electronics, and kitchen appliances. Reviews are encoded in feature vectors of bag-of-words unigrams and bigrams with binary labels indicating the rankings. From the four categories we obtain twelve domain adaptation tasks where each category serves once as source domain and once as target domain.

UCI-HAR The *Human Activity Recognition* (Anguita et al., 2013) dataset from the UC Irvine Repository contains data from three motion sensors (accelerometer, gyroscope and body-worn sensors) gathered using smartphones from 30 different subjects. It classifies their activities in several categories, namely, walking, walking upstairs, downstairs, standing, sitting, and lying down.

WISDM (Kwapisz et al., 2011) is a class-imbalanced dataset variant from collected accelerometer sensors, including GPS data, from 29 different subjects which are performing similar activities as in the UCI-HAR dataset.

HHAR The *Heterogeneity Human Activity Recognition* (Stisen et al., 2015) dataset investigate sensor-, device- and workload-specific heterogeneities using 36 smartphones and smartwatches, consisting of 13 different device models from four manufacturers.

Sleep-EDF The *Sleep Stage Classification* time-series setting aims to classify the electroencephalography (EEG) signals into five stages i.e., Wake (W), Non-Rapid Eye Movement stages (N1, N2, N3), and Rapid Eye Movement (REM). Analogous to Ragab et al. (2022); Eldele et al. (2021), we adopt the Sleep-EDF-20 dataset obtained from PhysioBank (Goldberger et al., 2000), which contains EEG readings from 20 healthy subjects.

We rely on the AdaTime benchmark suite (Ragab et al., 2022) in most evaluations. The four time-series datasets above are originally included there. We extend AdaTime to support the other discussed datasets as well, and extend its domain adaptation methods.

5.4 RESULTS

We separate the applied methods into two groups, namely *heuristic* and methods with *theoretical error guarantees*. All tables show accuracies of source-only (SO) and target-best (TB) models, where source-only denotes training without domain adaptation and target-best the best performing model obtained among all parameter settings. We highlight in bold the performance of the best performing method with theoretical error guarantees, and in italic the best performing heuristic. See Table 1 for results. Please find the full tables in the Supplementary Material Section D.

Outperformance of theoretically justified methods: On all datasets, our method outperforms IWV and DEV, setting a new state of the art for solving parameter choice issues under theoretical guarantees.

Outperformance of heuristics: It is interesting to note that each heuristic outperforms IWV and DEV on at least five of seven datasets. Moreover, every heuristic outperforms the (average) target best model (TB) in at least two cases, making it impossible for *any* model selection method to win in these cases. These facts highlight the quality of the predictions of our chosen heuristics. However, each heuristic is outperformed by our method on at least five of seven datasets.

Information in aggregation weights and robustness w.r.t. inaccurate models: It is interesting to observe that, in contrast to the other heuristic aggregation baselines, the aggregation weights c_1, \dots, c_l of our method tend to be larger for accurate models, see Section D.5. Another result is that our method tends to be less sensitive to a high number of inaccurate models than the baselines, see Section D.6. This serves as another reason for its high empirical performance.

6 CONCLUSION AND FUTURE WORK

We present a constructive theory-based method for approaching parameter choice issues in the setting of unsupervised domain adaptation. Its theoretical approach relies on the extension of weighted least squares to vector-valued functions. The resulting aggregation method distinguishes itself by a wide scope of admissible model classes without strong assumptions, e.g. support vector machines, decision trees and neural networks. A broad empirical comparative study on benchmark datasets for language,

images, body sensor signals and handy signals, underpins the theory-based optimality claim. It is left for future research to further refine the theory and its estimates, e.g., by exploiting concentration bounds from Gretton et al. (2006) or advanced density ratio estimators from Sugiyama et al. (2012).

Table 1: Mean and standard deviation (after \pm) of target classification accuracy on Amazon Reviews, Sleep-EDF, UCI-HAR, HHAR and WISDM datasets over three different random initialization of model weights and several domain adaptation tasks.

Amazon Reviews											
Method	Heuristic					Theoretical error guarantees					
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB		
HoMM	0.769(±0.009)	0.771(±0.010)	0.778(±0.010)	0.778(±0.011)	0.777(±0.010)	0.765(±0.011)	0.766(±0.011)	0.778(±0.010)	0.769(±0.012)		
AdvSKM	0.766(±0.012)	0.780(±0.009)	0.779(±0.010)	0.779(±0.010)	0.778(±0.011)	0.759(±0.012)	0.766(±0.012)	0.780(±0.009)	0.770(±0.012)		
DIRT	0.764(±0.009)	0.786(±0.008)	0.786(±0.010)	0.786(±0.008)	0.800(±0.008)	0.778(±0.022)	0.773(±0.056)	0.787(±0.008)	0.786(±0.009)		
DDC	0.766(±0.012)	0.779(±0.010)	0.780(±0.009)	0.779(±0.010)	0.778(±0.010)	0.767(±0.017)	0.768(±0.011)	0.780(±0.010)	0.770(±0.013)		
CMD	0.767(±0.012)	0.791(±0.009)	0.792(±0.009)	0.789(±0.010)	0.792(±0.010)	0.765(±0.010)	0.710(±0.015)	0.794(±0.009)	0.785(±0.009)		
MMDA	0.767(±0.011)	0.787(±0.011)	0.785(±0.011)	0.785(±0.011)	0.787(±0.012)	0.769(±0.011)	0.766(±0.011)	0.787(±0.011)	0.782(±0.011)		
CoDATS	0.767(±0.012)	0.781(±0.010)	0.784(±0.010)	0.784(±0.010)	0.784(±0.010)	0.759(±0.011)	0.759(±0.011)	0.784(±0.011)	0.776(±0.012)		
Deep-Coral	0.766(±0.012)	0.781(±0.009)	0.782(±0.009)	0.782(±0.009)	0.782(±0.009)	0.759(±0.016)	0.769(±0.037)	0.785(±0.009)	0.776(±0.013)		
CDAN	0.767(±0.012)	0.788(±0.010)	0.787(±0.009)	0.787(±0.010)	0.787(±0.011)	0.757(±0.011)	0.776(±0.014)	0.788(±0.010)	0.777(±0.011)		
DANN	0.767(±0.012)	0.796(±0.010)	0.792(±0.010)	0.793(±0.010)	0.800(±0.011)	0.776(±0.011)	0.778(±0.012)	0.797(±0.009)	0.798(±0.012)		
DSAN	0.769(±0.009)	0.796(±0.009)	0.792(±0.009)	0.791(±0.010)	0.800(±0.010)	0.779(±0.012)	0.763(±0.017)	0.795(±0.009)	0.789(±0.012)		
Avg.	0.767(±0.011)	0.787(±0.009)	0.786(±0.010)	0.786(±0.010)	0.789(±0.010)	0.772(±0.014)	0.764(±0.019)	0.788(±0.009)	0.781(±0.012)		
Sleep-EDF											
Method	Heuristic					Theoretical error guarantees					
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB		
HoMM	0.676(±0.036)	0.722(±0.017)	0.719(±0.023)	0.718(±0.021)	0.724(±0.032)	0.726(±0.046)	0.678(±0.035)	0.747(±0.025)	0.715(±0.047)		
AdvSKM	0.665(±0.058)	0.708(±0.023)	0.712(±0.027)	0.712(±0.027)	0.718(±0.030)	0.703(±0.052)	0.692(±0.038)	0.722(±0.025)	0.706(±0.054)		
DIRT	0.665(±0.059)	0.708(±0.024)	0.745(±0.027)	0.748(±0.027)	0.742(±0.031)	0.679(±0.045)	0.747(±0.037)	0.743(±0.037)			
DDC	0.646(±0.035)	0.717(±0.029)	0.719(±0.037)	0.721(±0.031)	0.726(±0.038)	0.690(±0.056)	0.690(±0.031)	0.724(±0.032)	0.721(±0.031)		
CMD	0.653(±0.057)	0.741(±0.022)	0.736(±0.016)	0.723(±0.020)	0.709(±0.015)	0.716(±0.052)	0.640(±0.068)	0.729(±0.018)	0.725(±0.053)		
MMDA	0.650(±0.051)	0.736(±0.014)	0.727(±0.021)	0.723(±0.018)	0.714(±0.028)	0.704(±0.033)	0.660(±0.034)	0.745(±0.031)	0.715(±0.042)		
CoDATS	0.672(±0.084)	0.738(±0.029)	0.739(±0.036)	0.736(±0.030)	0.723(±0.039)	0.683(±0.090)	0.690(±0.107)	0.744(±0.021)	0.715(±0.045)		
Deep-Coral	0.643(±0.049)	0.716(±0.018)	0.717(±0.028)	0.712(±0.027)	0.694(±0.032)	0.700(±0.053)	0.675(±0.077)	0.713(±0.021)	0.702(±0.070)		
CDAN	0.652(±0.056)	0.732(±0.016)	0.739(±0.024)	0.739(±0.024)	0.728(±0.029)	0.697(±0.031)	0.642(±0.065)	0.748(±0.019)	0.713(±0.045)		
DANN	0.641(±0.047)	0.722(±0.017)	0.723(±0.026)	0.721(±0.025)	0.714(±0.024)	0.687(±0.034)	0.644(±0.046)	0.724(±0.018)	0.710(±0.035)		
DSAN	0.653(±0.060)	0.738(±0.018)	0.740(±0.016)	0.732(±0.016)	0.728(±0.026)	0.712(±0.070)	0.589(±0.063)	0.757(±0.016)	0.700(±0.033)		
Avg.	0.655(±0.054)	0.729(±0.018)	0.729(±0.024)	0.725(±0.023)	0.717(±0.028)	0.700(±0.052)	0.660(±0.057)	0.737(±0.020)	0.712(±0.045)		
UCI-HAR											
Method	Heuristic					Theoretical error guarantees					
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB		
HoMM	0.782(±0.078)	0.832(±0.029)	0.818(±0.023)	0.818(±0.022)	0.788(±0.040)	0.800(±0.055)	0.800(±0.098)	0.846(±0.010)	0.851(±0.039)		
AdvSKM	0.724(±0.059)	0.701(±0.024)	0.800(±0.022)	0.810(±0.022)	0.768(±0.022)	0.707(±0.100)	0.711(±0.167)	0.809(±0.022)	0.811(±0.039)		
DIRT	0.782(±0.044)	0.919(±0.013)	0.907(±0.009)	0.898(±0.016)	0.756(±0.036)	0.807(±0.107)	0.808(±0.112)	0.900(±0.015)	0.928(±0.034)		
DDC	0.790(±0.061)	0.809(±0.019)	0.807(±0.026)	0.810(±0.017)	0.756(±0.048)	0.734(±0.070)	0.734(±0.109)	0.804(±0.028)	0.792(±0.043)		
CMD	0.788(±0.058)	0.869(±0.012)	0.849(±0.014)	0.839(±0.023)	0.731(±0.066)	0.834(±0.064)	0.812(±0.080)	0.842(±0.025)	0.888(±0.037)		
MMDA	0.785(±0.018)	0.819(±0.022)	0.812(±0.028)	0.800(±0.032)	0.759(±0.085)	0.767(±0.107)	0.807(±0.025)	0.840(±0.055)			
CoDATS	0.760(±0.037)	0.854(±0.020)	0.832(±0.024)	0.832(±0.026)	0.785(±0.057)	0.794(±0.078)	0.794(±0.078)	0.816(±0.016)	0.812(±0.012)		
Deep-Coral	0.756(±0.057)	0.809(±0.027)	0.805(±0.022)	0.808(±0.030)	0.804(±0.040)	0.798(±0.141)	0.798(±0.141)	0.805(±0.029)	0.809(±0.029)		
CDAN	0.756(±0.055)	0.842(±0.009)	0.813(±0.020)	0.840(±0.024)	0.802(±0.034)	0.781(±0.072)	0.687(±0.068)	0.846(±0.018)	0.852(±0.026)		
DANN	0.756(±0.026)	0.858(±0.016)	0.856(±0.033)	0.856(±0.033)	0.800(±0.057)	0.763(±0.032)	0.780(±0.043)	0.849(±0.023)	0.847(±0.007)		
DSAN	0.762(±0.032)	0.849(±0.023)	0.843(±0.033)	0.854(±0.025)	0.749(±0.065)	0.775(±0.043)	0.744(±0.035)	0.858(±0.023)	0.865(±0.038)		
Avg.	0.770(±0.046)	0.840(±0.017)	0.833(±0.023)	0.832(±0.024)	0.769(±0.060)	0.774(±0.070)	0.765(±0.090)	0.835(±0.020)	0.850(±0.020)		
HHAR											
Method	Heuristic					Theoretical error guarantees					
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB		
HoMM	0.739(±0.044)	0.757(±0.014)	0.759(±0.013)	0.759(±0.014)	0.700(±0.058)	0.720(±0.027)	0.733(±0.031)	0.759(±0.007)	0.764(±0.023)		
AdvSKM	0.718(±0.042)	0.749(±0.023)	0.742(±0.016)	0.748(±0.034)	0.676(±0.046)	0.730(±0.051)	0.728(±0.051)	0.752(±0.031)	0.749(±0.025)		
DIRT	0.728(±0.026)	0.803(±0.011)	0.792(±0.016)	0.803(±0.017)	0.796(±0.066)	0.743(±0.028)	0.739(±0.075)	0.816(±0.008)	0.820(±0.015)		
DDC	0.716(±0.063)	0.748(±0.014)	0.750(±0.009)	0.748(±0.007)	0.717(±0.075)	0.711(±0.048)	0.705(±0.066)	0.748(±0.012)	0.729(±0.027)		
CMD	0.738(±0.030)	0.781(±0.017)	0.781(±0.016)	0.780(±0.015)	0.698(±0.058)	0.720(±0.036)	0.731(±0.047)	0.780(±0.017)	0.785(±0.035)		
MMDA	0.738(±0.036)	0.773(±0.017)	0.771(±0.016)	0.770(±0.015)	0.698(±0.058)	0.728(±0.036)	0.731(±0.047)	0.780(±0.017)	0.785(±0.035)		
CoDATS	0.710(±0.030)	0.766(±0.023)	0.772(±0.024)	0.773(±0.020)	0.722(±0.064)	0.739(±0.028)	0.739(±0.040)	0.812(±0.009)	0.785(±0.039)		
Deep-Coral	0.745(±0.046)	0.766(±0.012)	0.762(±0.015)	0.766(±0.027)	0.681(±0.073)	0.754(±0.054)	0.758(±0.244)	0.764(±0.006)	0.776(±0.023)		
CDAN	0.728(±0.039)	0.762(±0.012)	0.758(±0.017)	0.764(±0.016)	0.765(±0.063)	0.774(±0.035)	0.775(±0.036)	0.816(±0.011)	0.790(±0.038)		
DANN	0.757(±0.057)	0.779(±0.012)	0.774(±0.009)	0.773(±0.011)	0.722(±0.103)	0.798(±0.041)	0.793(±0.045)	0.818(±0.009)	0.807(±0.020)		
DSAN	0.721(±0.053)	0.803(±0.010)	0.797(±0.014)	0.802(±0.007)	0.724(±0.065)	0.741(±0.033)	0.596(±0.031)	0.825(±0.008)	0.826(±0.046)		
Avg.	0.732(±0.042)	0.771(±0.015)	0.768(±0.017)	0.771(±0.018)	0.722(±0.068)	0.746(±0.037)	0.722(±0.063)	0.787(±0.012)	0.784(±0.028)		
WISDM											
Method	Heuristic					Theoretical error guarantees					
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB		
HoMM	0.753(±0.054)	0.741(±0.026)	0.738(±0.031)	0.739(±0.047)	0.775(±0.062)	0.753(±0.054)	0.740(±0.054)	0.728(±0.021)	0.774(±0.037)		
AdvSKM	0.747(±0.058)	0.771(±0.023)	0.751(±0.055)	0.757(±0.032)	0.742(±0.062)	0.747(±0.050)	0.747(±0.137)	0.777(±0.014)	0.779(±0.045)		
DIRT	0.756(±0.038)	0.774(±0.015)	0.773(±0.016)	0.777(±0.037)	0.771(±0.067)	0.749(±0.059)	0.770(±0.048)	0.780(±0.015)	0.774(±0.043)		
DDC	0.744(±0.059)	0.775(±0.015)	0.774(±0.023)	0.775(±0.027)	0.773(±0.068)	0.748(±0.052)	0.751(±0.049)	0.780(±0.015)	0.774(±0.043)		
CMD	0.710(±0.088)	0.772(±0.021)	0.765(±0.032)	0.767(±0.040)	0.728(±0.092)</						

ACKNOWLEDGMENTS

The ELLIS Unit Linz, the LIT AI Lab, and the Institute for Machine Learning are supported by the Federal State Upper Austria. IARAI is supported by Here Technologies. We thank the projects AIMOTION (LIT-2018-6-YOU-212), AI-SNN (LIT-2018-6-YOU-214), DeepFlood (LIT-2019-8-YOU-213), Medical Cognitive Computing Center (MC3), INCONTROL-RL (FFG-881064), PRIMAL (FFG-873979), S3AI (FFG-872172), DL for GranularFlow (FFG-871302), AIRI FG 9-N (FWF-36284, FWF-36235), and ELISE (H2020-ICT-2019-3 ID: 951847). We further thank Audi.JKU Deep Learning Center, TGW LOGISTICS GROUP GMBH, Silicon Austria Labs (SAL), FILL GmbH, Anyline GmbH, Google, ZF Friedrichshafen AG, Robert Bosch GmbH, UCB Biopharma SRL, Merck Healthcare KGaA, Verbund AG, TÜV Austria, Frauscher Sensonic, and the NVIDIA Corporation. The research reported in this paper has been funded by the Federal Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK), the Federal Ministry for Digital and Economic Affairs (BMDW), and the Province of Upper Austria in the frame of the COMET–Competence Centers for Excellent Technologies Programme and the COMET Module S3AI managed by the Austrian Research Promotion Agency FFG.

REFERENCES

- M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- W. Ahmed, P. Morerio, and V. Murino. Cleaning noisy labels by negative ensemble learning for source-free unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1616–1625, 2022.
- S. Al-Stouhi and C. K. Reddy. Adaptive boosting for transfer learning using dynamic updates. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 60–75. Springer, 2011.
- D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. *European Symposium on Artificial Neural Networks*, pp. 437–442, 2013.
- B. Athiwaratkun, M. Finzi, P. Izmailov, and A. G. Wilson. There are many consistent explanations of unlabeled data: Why you should average. *International Conference on Learning Representations (2019)*, 2019.
- S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems*, pp. 137–144, 2007.
- S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175, 2010.
- S. Bickel, M. Brückner, and T. Scheffer. Discriminative learning for differing training and test distributions. In *Proceedings of the 24th international conference on Machine learning*, pp. 81–88, 2007.
- A. Bietti and J. Mairal. Invariance and stability of deep convolutional representations. *Advances in neural information processing systems*, 30, 2017.
- A. Bietti and J. Mairal. Group invariance, stability to deformations, and complexity of deep convolutional representations. *The Journal of Machine Learning Research*, 20(1):876–924, 2019.
- L. Biewald. Experiment tracking with weights and biases, 2020. URL <https://www.wandb.com/>. Software available from wandb.com.

- J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pp. 120–128, 2006.
- L. Breiman. Bagging predictors. Technical Report 421, Department of Statistics, UC Berkeley, 1994.
- L. Breiman. Bagging predictors. *Machine Learning*, 26(2):123–140, 1996a.
- L. Breiman. Stacked regressions. *Machine Learning*, 24(1):49–64, 1996b.
- L. Breiman. Arcing classifier (with discussion and a rejoinder by the author). *The Annals of Statistics*, 26(3):801–849, 1998. doi: 10.1214/aos/1024691079.
- A. Caponnetto and E. De Vito. Risk bounds for regularized least-squares algorithm with operator-valued kernels. Technical report, CBCL paper 249/CSAIL-TR-2005-031, MIT, 2005.
- A. Caponnetto and E. De Vito. Optimal rates for the regularized least-squares algorithm. *Foundations of Computational Mathematics*, 7(3):331–368, 2007.
- C. Chen, Z. Fu, Z. Chen, S. Jin, Z. Cheng, X. Jin, and X.-S. Hua. Homm: Higher-order moment matching for unsupervised domain adaptation. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2020.
- M. Chen, Z. Xu, K. Weinberger, and F. Sha. Marginalized denoising autoencoders for domain adaptation. *Proceedings of the International Conference on Machine Learning*, pp. 767–774, 2012.
- F. Cucker and S. Smale. On the mathematical foundations of learning. *Bulletin of the American mathematical society*, 39(1):1–49, 2002.
- C. O. da Costa-Luis. Tqdm: A fast, extensible progress meter for python and cli. *Journal of Open Source Software*, 4(37):1277, 2019.
- W. Dai, Q. Yang, G. R. Xue, and Y. Yu. Boosting for transfer learning. In *Proceedings of the 24th International Conference on Machine Learning*, pp. 193–200, 2007.
- X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma. A survey on ensemble learning. *Frontiers of Computer Science*, 14(2):241–258, 2020.
- L. Duan, I. W. Tsang, and D. Xu. Domain transfer multiple kernel learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):465–479, 2012.
- R. M. Dudley. *Real analysis and probability*, volume 74. Cambridge University Press, 2002.
- E. Eldele, Z. Chen, C. Liu, M. Wu, C.-K. Kwok, X. Li, and C. Guan. An attention-based deep learning approach for sleep stage classification with single-channel eeg. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2021.
- A. Fermanian, P. Marion, J. P. Vert, and G. Biau. Framing rnm as a kernel method: A neural ode approach. *Advances in Neural Information Processing Systems*, 34, 2021.
- G. French, M. Mackiewicz, and M. Fisher. Self-ensembling for visual domain adaptation. *International Conference on Learning Representations*, 2018.
- Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(Jan):1–35, 2016.
- E. R. Gizewski, L. Mayer, B. A. Moser, D. H. Nguyen, S. Pereverzyev Jr, S. V. Pereverzyev, N. Shepeleva, and W. Zellinger. On a regularization of unsupervised domain adaptation in RKHS. *Applied and Computational Harmonic Analysis*, 57:201–227, 2022.
- A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals. *Circulation*, 101(23):215–220, 2000.

- I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.
- A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola. A kernel method for the two-sample-problem. In *Advances in Neural Information Processing Systems*, pp. 513–520, 2006.
- J. M. Ha and B. D. Youn. A health data map-based ensemble of deep domain adaptation under inhomogeneous operating conditions for fault diagnosis of a planetary gearbox. *IEEE Access*, 9: 79118–79127, 2021.
- K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- S. Hochreiter and J. Schmidhuber. Simplifying neural nets by discovering flat minima. *Advances in neural information processing systems*, 7, 1994.
- S. Hochreiter and J. Schmidhuber. Flat minima. *Neural computation*, 9(1):1–42, 1997.
- J. Hoffman, M. Mohri, and N. Zhang. Algorithms and theory for multiple-source adaptation. *Advances in Neural Information Processing Systems*, 31, 2018.
- J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, and A. Smola. Correcting sample selection bias by unlabeled data. *Advances in neural information processing systems*, 19, 2006.
- H. Daume III and D. Marcu. Domain adaptation for statistical classifiers. *Journal of artificial Intelligence research*, 26:101–126, 2006.
- J. Jiao, M. Zhao, J. Lin, and C. Ding. Classifier inconsistency-based domain adaptation network for partial transfer intelligent diagnosis. *IEEE Transactions on Industrial Informatics*, 16(9): 5965–5974, 2019.
- T. Kanamori, S. Hido, and M. Sugiyama. A least-squares approach to direct importance estimation. *The Journal of Machine Learning Research*, 10:1391–1445, 2009.
- T. Kanamori, T. Suzuki, and M. Sugiyama. Statistical analysis of kernel-based least-squares density-ratio estimation. *Machine Learning*, 86(3):335–367, 2012.
- G. Kang, L. Jiang, Y. Wei, Y. Yang, and A. G. Hauptmann. Contrastive adaptation network for single-and multi-source domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 2020.
- D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- W. M. Kouw, J. H. Krijthe, and M. Loog. Robust importance-weighted cross-validation under sample selection bias. In *IEEE International Workshop on Machine Learning for Signal Processing*, pp. 1–6. IEEE, 2019.
- A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.
- J. R. Kwapisz, G. M. Weiss, and S. A. Moore. Activity recognition using cell phone accelerometers. *Sigkdd Explorations*, 12(2):74–82, 2011.
- S. Laine and T. Aila. Temporal ensembling for semi-supervised learning. *International Conference on Learning Representations (ICLR)*, 2017.
- Q. Liu and H. Xue. Adversarial spectral kernel matching for unsupervised time series domain adaptation. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 30, 2021.
- M. Long, Y. Cao, J. Wang, and M. Jordan. Learning transferable features with deep adaptation networks. In *Proceedings of the International Conference on Machine Learning*, pp. 97–105, 2015.
- M. Long, Z. Cao, J. Wang, and M. I. Jordan. Conditional adversarial domain adaptation. *Advances in Neural Information Processing Systems (NeurIPS)*, 31, 2018.

- C. Louizos, K. Swersky, Y. Li, M. Welling, and R. Zemel. The variational fair auto encoder. *International Conference on Learning Representations*, 2016.
- C. Ma and L. Wu. The baron space and the flow-induced function spaces for neural network models. *Constructive Approximation*, 55(1):369–406, 2022.
- A. Mayr, G. Klambauer, T. Unterthiner, and S. Hochreiter. Deeptox: toxicity prediction using deep learning. *Frontiers in Environmental Science*, 3:80, 2016.
- K. Musgrave, S. Belongie, and S.-N. Lim. Unsupervised domain adaptation: A reality check. *arXiv preprint arXiv:2111.15672*, 2021.
- D. Nozza, E. Fersini, and E. Messina. Deep learning and ensemble methods for domain adaptation. In *IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 184–189. IEEE, 2016.
- S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.
- A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.
- X. Peng, B. Usmanand N. Kaushik and D. Wang, J. Hoffman, and K. Saenko. Visda: A synthetic-to-real benchmark for visual domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 2021–2026, 2018.
- X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1406–1415, 2019.
- S. V. Pereverzyev. *An Introduction to Artificial Intelligence based on Reproducing Kernel Hilbert Spaces*. Birkhäuser Cham, 2022.
- I. Pinelis. An approach to inequalities for the distributions of infinite-dimensional martingales. In *Probability in Banach Spaces, 8: Proceedings of the Eighth International Conference*, pp. 128–134. Springer, 1992.
- M. Ragab, E. Eldele, W. L. Tan, C.-S. Foo, Z. Chen, M. Wu, C.-K. Kwok, and X. Li. Adatime: A benchmarking suite for domain adaptation on time series data. *arXiv preprint arXiv:2203.08321*, 2022.
- M. M. Rahman, C. Fookes, M. Baktashmotagh, and S. Sridharan. On minimum discrepancy estimation for deep domain adaptation. *Domain Adaptation for Visual Understanding*, 2020.
- S. Rakshit, B. Banerjee, G. Roig, and S. Chaudhuri. Unsupervised multi-source domain adaptation driven by deep adversarial ensemble learning. In *German Conference on Pattern Recognition*, pp. 485–498. Springer, 2019.
- H. Razari and S. Samothrakis. Bagging adversarial neural networks for domain adaptation in non-stationary eeg. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7. IEEE, 2019.
- L. Rosasco, M. Belkin, and E. De Vito. On learning with integral operators. *Journal of Machine Learning Research*, 11(2), 2010.
- K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pp. 213–226. Springer, 2010.
- K. Saito, Y. Ushiku, and T. Harada. Asymmetric tri-training for unsupervised domain adaptation. In *International Conference on Machine Learning*, pp. 2988–2997. PMLR, 2017.

- K. Saito, D. Kim, P. Teterwak, S. Sclaroff, T. Darrell, and K. Saenko. Tune it the right way: Unsupervised validation of domain adaptation via soft neighborhood density. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9184–9193, 2021.
- R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.
- H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.
- C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.
- R. Shu, H. Bui, H. Narui, and S. Ermon. A dirt-t approach to unsupervised domain adaptation. *International Conference on Learning Representations (ICLR)*, 2018.
- A. Stisen, H. Blunck, S. Bhattacharya, T. S. Prentow, M. B. Kjærgaard, A. Dey, T. Sonne, and M. M. Jensen. Smart devices are different: Assessing and mitigating mobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, SenSys ’15, pp. 127–140, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336314. doi: 10.1145/2809695.2809718.
- G. Strang. *Linear algebra and its applications*. Orlando, FL, Academic Press, Inc., 1980.
- M. Sugiyama, M. Krauledat, and K. M. Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(5), 2007.
- M. Sugiyama, T. Suzuki, and T. Kanamori. *Density ratio estimation in machine learning*. Cambridge University Press, 2012.
- B. Sun and K. Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *Proceedings of the European Conference on Computer Vision*, pp. 443–450, 2016.
- B. Sun, J. Feng, and K. Saenko. Correlation alignment for unsupervised domain adaptation. *Domain Adaptation in Computer Vision Applications*, pp. 153–171, 2017.
- W. Tuand S. Sun. Dynamical ensemble learning with model-friendly classifiers for domain adaptation. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pp. 1181–1184. IEEE, 2012.
- A. Tarvainen and H. Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017.
- G. Teschl. Topics in Linear and Nonlinear Functional Analysis. *Amer. Math. Soc., Providence, to appear*, 2022a.
- G. Teschl. Topics in Real Analysis. *Amer. Math. Soc., Providence, to appear*, 2022b.
- E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.
- T. Varsavsky, M. Orbes-Arteaga, C. H. Sudre, M. S. Graham, P. Nachev, and M. J. Cardoso. Test-time unsupervised domain adaptation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 428–436. Springer, 2020.
- G. Wilson and D. J. Cook. A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 11(5):1–46, 2020.
- G. Wilson, J. R. Doppa, and D. J. Cook. Multi-source deep domain adaptation with weak supervision for time-series sensor data. *Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD)*, 2020.
- D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:214–259, 1992.

- M. Wortsman, G. Ilharco, S. Y. Gadre, R. Roelofs, R. Gontijo-Lopes, A. S. Morcos, H. Namkoong, A. Farhadi, Y. Carmon, S. Kornblith, and L. Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *arXiv preprint arXiv:2203.05482*, 2022.
- R. Xia, C. Zong, X. Hu, and E. Cambria. Feature ensemble plus sample selection: domain adaptation for sentiment classification. *IEEE Intelligent Systems*, 28(3):10–18, 2013.
- R. Xu, Z. Chen, W. Zuo, J. Yan, and L. Lin. Deep cocktail network: Multi-source unsupervised domain adaptation with category shift. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3964–3973, 2018.
- J. B. Yang, Q. Mao, Q. L. Xiang, I. W.-H. Tsangand K. M. A. Chai, and H. L. Chieu. Domain adaptation for coreference resolution: An adaptive ensemble approach. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 744–753, 2012.
- K. You, X. Wang, M. Long, and M. Jordan. Towards accurate model selection in deep unsupervised domain adaptation. In *Proceedings of the International Conference on Machine Learning*, pp. 7124–7133. PMLR, 2019.
- W. Zellinger, T. Grubinger, E. Lughofer, T. Natschläger, and S. Saminger-Platz. Central moment discrepancy (cmd) for domain-invariant representation learning. *International Conference on Learning Representations*, 2017.
- W. Zellinger, T. Grubinger, M. Zwick, E. Lughofer, H. Schöner, T. Natschläger, and S. Saminger-Platz. Multi-source transfer learning of time series in cyclical manufacturing. *Journal of Intelligent Manufacturing*, 31(3):777–787, 2020.
- W. Zellinger, N. Shepeleva, M.-C. Dinu, H. Eghbal-zadeh, H. Nguyen, B. Nessler, S. Pereverzyev, and B. A. Moser. The balancing principle for parameter choice in distance-regularized domain adaptation. *Advances in Neural Information Processing Systems*, 34, 2021.
- K. Zhang, M. Gong, and B. Schölkopf. Multi-source domain adaptation: A causal view. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.
- K. Zhou, Y. Yang, Y. Qiao, and T. Xiang. Domain adaptive ensemble learning. *IEEE Transactions on Image Processing*, 30:8008–8018, 2021.
- Y. Zhu, F. Zhuang, J. Wang, G. Ke, J. Chen, J. Bian, H. Xiong, and Q. He. Deep subdomain adaptation network for image classification. *IEEE Transactions on Neural Networks and Learning Systems*, 32(4):1713–1722, 2021.
- D. Zou, Q. Zhu, and P. Yan. Unsupervised domain adaptation with dual-scheme fusion network for medical image segmentation. In *IJCAI*, pp. 3291–3298, 2020.
- Y. Zou, Z. Yu, B. V. K. Kumar, and J. Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 289–305, 2018.

A NOTATION AND PROOF OF MAIN RESULT

The aim of this section is to give a full proof of our main result, Theorem 1 in the main paper. We start by introducing and summarizing the notation and the required concepts from functional analysis and measure theory, so that we can state and prove the required lemmas.

Summary of Notation

- *Spaces:* input space $\mathcal{X} \subset \mathbb{R}^{d_1}$ and label space \mathcal{Y} with inner product $\langle \cdot, \cdot \rangle_{\mathcal{Y}}$. \mathcal{Y} is assumed to be a separable Hilbert space such that for the associated norm $\|y\|_{\mathcal{Y}} \leq y_0$ holds for all $y \in \mathcal{Y}$ and some $y_0 > 0$. Note that this setting is more general than the one from the main text, where we assumed $\mathcal{Y} \subset \mathbb{R}^{d_2}$ (the simplification in the main text improves readability and respectation of space limits).
- *Datasets and Distributions:* Source data set: $(\mathbf{x}, \mathbf{y}) = ((x_1, y_1), \dots, (x_n, y_n)) \in (\mathcal{X} \times \mathcal{Y})^n$ independently drawn according to source distribution p on $\mathcal{X} \times \mathcal{Y}$ and an unlabeled *target* dataset $\mathbf{x}' = (x'_1, \dots, x'_m) \in \mathcal{X}^m$ independently drawn according marginal distribution $q_{\mathcal{X}}$ of target distribution q on $\mathcal{X} \times \mathcal{Y}$ (the corresponding marginal distribution of p on \mathcal{X} is similarly denoted as $p_{\mathcal{X}}$).
- *Source Risk:* $\mathcal{R}_p(f) = \int_{\mathcal{X} \times \mathcal{Y}} \|f(x) - y\|_{\mathcal{Y}}^2 d(p(x, y))$.
- *Source Regression function* $f_p(x) = \int_{\mathcal{Y}} y d(p(y|x))$. (Vector valued) integral in the sense of Lebesgue-Bochner.
- *Target Risk:* $\mathcal{R}_q(f) = \int_{\mathcal{X} \times \mathcal{Y}} \|f(x) - y\|_{\mathcal{Y}}^2 d(q(x, y))$
- *Target Regression function* $f_q(x) = \int_{\mathcal{Y}} y d(q(y|x))$. (Vector valued) integral in the sense of Lebesgue-Bochner.

Problem

- *Given:* sequence $f_1, \dots, f_l : \mathcal{X} \rightarrow \mathcal{Y}$ of models, source sample (\mathbf{x}, \mathbf{y}) and unlabeled target sample \mathbf{x}'
- *Aim:* find aggregation $f = \sum_{i=1}^l c_i f_i$ with minimal $\mathcal{R}_q(f)$.

Main Assumptions

- *covariate shift:* $p(y|x) = q(y|x)$ and thus $f_p = f_q$.
- *bounded density ratio:* there is $\beta : \mathcal{X} \rightarrow [0, B]$ such that $dq_{\mathcal{X}}(x) = \beta(x) d p_{\mathcal{X}}(x)$.

Existence of the associated conditional probability measures is guaranteed by the fact that $\mathcal{X} \times \mathcal{Y}$ is Polish (a separable and complete metric space), c.f. Dudley (2002, Theorem 10.2.2.).

Notation from functional analysis/operator theory Let \mathbf{U} and \mathbf{V} denote separable Hilbert spaces (i.e. they admit countable orthonormal bases) with associated inner products $\langle \cdot, \cdot \rangle_{\mathbf{U}}$ (or $\langle \cdot, \cdot \rangle_{\mathbf{V}}$, respectively). Let us briefly recall some notions from functional analysis that we need in order to set up our theory. There are lots of standard references on these aspects, e.g. Teschl (2022a) and Teschl (2022b):

- $\mathcal{L}(\mathbf{U}, \mathbf{V})$: space of bounded linear operators $\mathbf{U} \rightarrow \mathbf{V}$ with uniform norm $\|\cdot\|_{\mathcal{L}(\mathbf{U}, \mathbf{V})}$. $\mathcal{L}(\mathbf{U})$: space of bounded linear operators $\mathbf{U} \rightarrow \mathbf{U}$.
- For $A \in \mathcal{L}(\mathbf{U}, \mathbf{V})$, its *adjoint* is denoted by $A^* \in \mathcal{L}(\mathbf{V}, \mathbf{U})$ (and uniquely defined by the equation $\langle Au, v \rangle_{\mathbf{V}} = \langle u, A^*v \rangle_{\mathbf{U}}$ for any $u \in \mathbf{U}, v \in \mathbf{V}$).
- If $A \in \mathcal{L}(\mathbf{U})$ and $A = A^*$: A is called *self-adjoint*.
- If $A \in \mathcal{L}(\mathbf{U})$ is self adjoint and $\langle Au, u \rangle_{\mathbf{U}} \geq 0$ for any $u \in \mathbf{U}$, then A is called *positive*. Equivalently: there exists (unique) bounded and self-adjoint $B := \sqrt{A} \in \mathcal{L}(\mathbf{U})$ such that $B^2 = A$.

- *Trace* of an operator $A \in \mathcal{L}(\mathbf{U})$: $\text{Tr}(A) = \sum_k \langle Ae_k, e_k \rangle_{\mathbf{U}}$ for any orthonormal basis $(e_k)_{k=1}^{\infty}$ of \mathbf{U} (independent of choice of basis). If $\text{Tr}(A) < \infty$: A is called *trace class*.
- $\mathcal{L}_2(\mathbf{U})$: separable Hilbert space of *Hilbert-Schmidt operators* on \mathbf{U} with scalar product $\langle A, B \rangle_{\mathcal{L}_2(\mathbf{U})} = \text{Tr}(B^* A)$ and norm $\|A\|_{\mathcal{L}_2(\mathbf{U})} = \sqrt{\text{Tr}(A^* A)} \geq \|A\|_{\mathcal{L}(\mathbf{U})}$.
- $A : \mathbf{U} \rightarrow \mathbf{V}$ is called *Hilbert-Schmidt*, if $A^* A$ is trace class. Also here: $\|A\|_{\mathcal{L}(\mathbf{U}, \mathbf{V})} \leq \sqrt{\text{Tr}(A^* A)}$
- For (probability) measure q on \mathcal{X} (or \mathcal{Y}) and appropriate functions $F : \mathcal{X} \rightarrow \mathbf{U}$ (e.g. strongly measurable and $\|F\|_{\mathbf{U}}$ is integrable wrt. q) we denote the usual (\mathbf{U} -valued) *Bochner integral* of F as $\int_{\mathcal{X}} F(x) dq(x)$. We denote the associated L^p -spaces by $L^p(\mathcal{X}, q, \mathbf{U})$, or $L^p(q)$ for short, if the associated spaces are clear from the context.

Assumptions on models We assume that the regression function $f^* = f_p = f_q$ as well as the models f_1, \dots, f_l belong to a *hypothesis space* $\mathcal{H} \subseteq C(\mathcal{X}, \mathcal{Y}) \subseteq L^2(p_{\mathcal{X}}) \cap L^2(q_{\mathcal{X}})$, where $C(\mathcal{X}, \mathcal{Y})$ denotes the space of bounded continuous functions $\mathcal{X} \rightarrow \mathcal{Y}$. The space \mathcal{H} should satisfy the following assumptions, which are discussed in much greater detail in Caponnetto & De Vito (2007) and Caponnetto & De Vito (2005):

Hypothesis 1. (Caponnetto & De Vito, 2007) *The space \mathcal{H} is a separable Hilbert space of functions $f : \mathcal{X} \rightarrow \mathcal{Y}$ such that:*

- For all $x \in \mathcal{X}$ there is a Hilbert-Schmidt operator $K_x : \mathcal{Y} \rightarrow \mathcal{H}$ satisfying

$$f(x) = K_x^* f, \quad f \in \mathcal{H}, \quad (5)$$

- The function from $\mathcal{X} \times \mathcal{X}$ to \mathbb{R}

$$(x, t) \mapsto \langle K_t v, K_x w \rangle_{\mathcal{H}} \text{ is measurable } \forall v, w \in \mathcal{Y}; \quad (6)$$

- There is $\kappa > 0$ such that

$$\text{Tr}(K_x^* K_x) \leq \kappa, \quad \forall x \in \mathcal{X}. \quad (7)$$

Moreover we assume that the norms $\|f_k\|_{\mathcal{H}}, k = 1, 2, \dots, l$, are under our control, such that we can put a threshold $\gamma_l > 0$ and consider $\|f_k\|_{\mathcal{H}} \leq \gamma_l$.

Further useful observations Then we have

$$K_t^* K_x = K(t, x) \in \mathcal{L}_2(\mathcal{Y}) \quad \forall x, t \in \mathcal{X}. \quad (8)$$

Given $x \in \mathcal{X}$ the operator

$$T_x = K_x K_x^* \in \mathcal{L}_2(\mathcal{H}), \quad (9)$$

is a positive Hilbert-Schmidt operator and (9) ensures

$$\|T_x\|_{\mathcal{L}(\mathcal{H})} \leq \|T_x\|_{\mathcal{L}_2(\mathcal{H})} = \|K(x, x)\|_{\mathcal{L}_2(\mathcal{Y})} \leq \kappa. \quad (10)$$

Let $T_{q_{\mathcal{X}}} : \mathcal{H} \rightarrow \mathcal{H}$ be

$$T_{q_{\mathcal{X}}} = \int_{\mathcal{X}} T_x dq_{\mathcal{X}}(x),$$

where the integral converges in $\mathcal{L}_2(\mathcal{H})$ to a positive trace class operator with

$$\|T_{q_{\mathcal{X}}}\|_{\mathcal{L}(\mathcal{H})} \leq \|T_{q_{\mathcal{X}}}\|_{\mathcal{L}_2(\mathcal{H})} \leq \text{Tr}(T_{q_{\mathcal{X}}}) = \int_{\mathcal{X}} \text{Tr}(T_x) dq_{\mathcal{X}}(x) \leq \kappa. \quad (11)$$

Following Proposition 1 in Caponnetto & De Vito (2007), we have the minimizers f_q of expected risk \mathcal{R}_q are the solution of the following equation:

$$T_{q_{\mathcal{X}}} f_q = g,$$

where

$$g = \int_{\mathcal{X}} K_x f_q(x) dq_{\mathcal{X}}(x) \in \mathcal{H},$$

with integral converging in \mathcal{H} .

Next we define the operators

$$\begin{aligned} T_{\mathbf{x}'} &= \frac{1}{m} \sum_{j=1}^m K_{x'_j} K_{x'_j}^*, \\ T_{\mathbf{x}, \beta} &= \frac{1}{n} \sum_{i=1}^n \beta(x_i) K_{x_i} K_{x_i}^*, \\ g_{\mathbf{x}, \mathbf{y}, \beta} &= \frac{1}{n} \sum_{i=1}^n \beta(x_i) K_{x_i} y_i. \end{aligned}$$

In the sequel we adopt the convention that C denotes a generic positive coefficient, which can vary from appearance to appearance and may only depend on basic parameter such as $p_{\mathcal{X}}, q_{\mathcal{X}}, \kappa, B, y_0$ and others introduced below, but not on n, m and error probability $\delta > 0$.

We will need the following statements.

Lemma 1. *With probability at least $1 - \delta$ we have*

$$\|T_{q_{\mathcal{X}}} - T_{\mathbf{x}'}\|_{\mathcal{L}(\mathcal{H})} \leq \|T_{q_{\mathcal{X}}} - T_{\mathbf{x}'}\|_{\mathcal{L}_2(\mathcal{H})} \leq C \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) m^{-\frac{1}{2}}, \quad (12)$$

$$\|T_{\mathbf{x}'} - T_{\mathbf{x}, \beta}\|_{\mathcal{L}(\mathcal{H})} \leq C \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) \left(n^{-\frac{1}{2}} + m^{-\frac{1}{2}} \right), \quad (13)$$

$$\|T_{\mathbf{x}, \beta} f^* - g_{\mathbf{x}, \mathbf{y}, \beta}\|_{\mathcal{H}} \leq C \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) n^{-\frac{1}{2}}, \quad (14)$$

where $C > 0$ does not depend on n, m and δ .

The proof of Lemma 1 is based on Lemma 4 of Huang et al. (2006), which we formulate in our notations as follows

Lemma 2. ((Huang et al., 2006)) *Let ϕ be a map from \mathbf{U} to \mathbf{U} such that $\|\phi(x)\|_{\mathbf{U}} \leq R$ for all $x \in \mathcal{X}$. Then with probability at least $1 - \delta$ it holds*

$$\left\| \frac{1}{m} \sum_{j=1}^m \phi(x'_j) - \frac{1}{n} \sum_{i=1}^n \beta(x_i) \phi(x_i) \right\|_{\mathbf{U}} \leq \left(1 + \sqrt{2 \log \frac{2}{\delta}} \right) R \sqrt{\frac{B^2}{n} + \frac{1}{m}}.$$

Moreover, we will need a concentration inequality that follows from Pinelis (1992), see also Rosasco et al. (2010).

Lemma 3 (Concentration lemma). *If $\xi_1, \xi_2, \dots, \xi_n$ are zero mean independent random variables with values in a separable Hilbert space \mathbf{U} , and for some $D > 0$ one has $\|\xi_i\|_{\mathbf{U}} \leq D$, $i = 1, 2, \dots, n$, then the following bound*

$$\left\| \frac{1}{n} \sum_{i=1}^n \xi_i \right\|_{\mathbf{U}} \leq \frac{D \sqrt{2 \log \frac{2}{\delta}}}{\sqrt{n}}$$

holds true with probability at least $1 - \delta$.

Proof of Lemma 1.

Let us start by proving (12) by introducing the map $\xi : \mathcal{X} \rightarrow \mathcal{L}_2(\mathcal{H})$ as $\xi(x) = K_x K_x^* - T_{q_{\mathcal{X}}}$. From (10) and (11) it follows that

$$\|\xi(x)\|_{\mathcal{L}_2(\mathcal{H})} \leq \|K_x K_x^*\|_{\mathcal{L}_2(\mathcal{H})} + \|T_{q_{\mathcal{X}}}\|_{\mathcal{L}_2(\mathcal{H})} \leq 2\kappa.$$

Moreover, we have

$$\int_{\mathcal{X}} \xi(x) dq_{\mathcal{X}}(x) = \int_{\mathcal{X}} K_x K_x^* dq_{\mathcal{X}}(x) - T_{q_{\mathcal{X}}} = 0.$$

Therefore, for $x'_j, j = 1, 2, \dots, m$, drawn i.i.d from the marginal probability measure $q_{\mathcal{X}}$, the corresponding operators $\xi_j = \xi(x'_j)$ can be treated as zero mean independent random variables in $\mathcal{L}_2(\mathcal{H})$, such that the condition of Concentration lemma are satisfied with $D = 2\kappa$, and

$$\|T_{\mathbf{x}'} - T_{q_{\mathcal{X}}}\|_{\mathcal{L}_2(\mathcal{H})} = \left\| \frac{1}{m} \sum_{j=1}^m K_{x'_j} K_{x'_j}^* - T_{q_{\mathcal{X}}} \right\|_{\mathcal{L}_2(\mathcal{H})} = \left\| \frac{1}{m} \sum_{j=1}^m \xi_j \right\|_{\mathcal{L}_2(\mathcal{H})} \leq \frac{2\kappa \sqrt{2 \log \frac{2}{\delta}}}{\sqrt{m}}.$$

To obtain (13), for any $f \in \mathcal{H}$ we define a map $\phi = \phi_f : \mathcal{X} \rightarrow \mathcal{H}$ as $\phi_f(x) = K_x K_x^* f$. It clear that

$$\|\phi_f(x)\|_{\mathcal{H}} = \|K_x K_x^*\|_{\mathcal{L}(\mathcal{H})} \|f\|_{\mathcal{H}} \leq \kappa \|f\|_{\mathcal{H}}.$$

Therefore, for the map $\phi = \phi_f$ the condition of the above Lemma 2 is satisfied with $R = \kappa \|f\|_{\mathcal{H}}$. Then directly from that lemma for any $f \in \mathcal{H}$ we have

$$\begin{aligned} \|T_{\mathbf{x}'} f - T_{\mathbf{x}, \beta} f\|_{\mathcal{H}} &= \left\| \frac{1}{m} \sum_{j=1}^m \phi_f(x'_j) - \frac{1}{n} \sum_{i=1}^n \beta(x_i) \phi_f(x_i) \right\|_{\mathcal{H}} \\ &\leq \left(1 + \sqrt{2 \log \frac{2}{\delta}} \right) \left(\sqrt{\frac{B^2}{n}} + \frac{1}{m} \right) \kappa \|f\|_{\mathcal{H}} \\ &\leq C \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) \left(m^{-\frac{1}{2}} + n^{-\frac{1}{2}} \right) \|f\|_{\mathcal{H}}, \end{aligned}$$

that proves (13).

Consider now the map $F : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{H}$ defined by

$$F(x, y) = \beta(x) K_x (f_p(x) - y).$$

Recall that $\|K_x\|_{\mathcal{L}(\mathcal{Y}, \mathcal{H})} \leq \sqrt{\text{Tr}(K_x^* K_x)} \leq \sqrt{\kappa}$. Then we obtain:

$$\|F(x, y)\|_{\mathcal{H}} \leq \|K_x\|_{\mathcal{L}(\mathcal{Y}, \mathcal{H})} \left\| \int_{\mathcal{Y}} y' dp(y'|x) - y \right\|_{\mathcal{Y}} |\beta(x)| \leq 2y_0 B \sqrt{\kappa}.$$

Moreover, for $p(x, y) = p(y|x)p_{\mathcal{X}}(x)$ we have

$$\int_{\mathcal{X} \times \mathcal{Y}} F(x, y) dp(x, y) = \int_{\mathcal{X}} K_x \beta(x) \int_{\mathcal{Y}} \left(\int_{\mathcal{Y}} y' dp(y'|x) - y \right) dp(y|x) dp_{\mathcal{X}}(x) = 0,$$

such that for $(x_i, y_i), i = 1, 2, \dots, n$, drawn i.i.d from the measure $p(x, y)$ the corresponding values $F_i = F(x_i, y_i)$ are zero mean independent random variables in \mathcal{H} . Then for the just defined $F_i = \beta(x_i) K_{x_i} (f_q(x_i) - y_i)$ the conditions of Lemma 3 are satisfied with $D = 2y_0 B \sqrt{\kappa}$, such that

$$\begin{aligned} \left\| \frac{1}{n} \sum_{i=1}^n F_i \right\|_{\mathcal{H}} &= \left\| \frac{1}{n} \sum_{i=1}^n \beta(x_i) K_{x_i} (f_q(x_i) - y_i) \right\|_{\mathcal{H}} \\ &= \left\| \sum_{i=1}^n \beta(x_i) K_{x_i} K_{x_i}^* f_q - \sum_{i=1}^n \beta(x_i) K_{x_i} y_i \right\|_{\mathcal{H}} \\ &= \|T_{\mathbf{x}, \beta} f_q - g_{\mathbf{x}, \mathbf{y}, \beta}\|_{\mathcal{H}} \leq \frac{2y_0 B \sqrt{\kappa} \sqrt{2 \log \frac{2}{\delta}}}{\sqrt{n}}. \end{aligned}$$

This bound gives us (14).

Aggregation for vector-valued functions Next we construct a new approximant in the form of a linear combination of approximants f_1, f_2, \dots, f_l , computed for all tried parameter values. The linear combination of the approximants is computed as

$$f = \sum_{k=1}^l c_k f_k. \quad (15)$$

Since f_1, f_2, \dots, f_l belong to RKHS \mathcal{H} , it is clear that $f \in \mathcal{H}$. Now we want to argue on how close we can get to f_q . Following Proposition 1 in Caponnetto & De Vito (2007), we have

$$\mathcal{R}_q(f) - \mathcal{R}_q(f_q) = \|f - f_q\|_{L^2(q_X)}^2 = \left\| \sqrt{T_{q_X}}(f - f_q) \right\|_{\mathcal{H}}^2. \quad (16)$$

Next we observe that the best approximation f^* of the target regression function f_q by linear combinations corresponds to the vector $c^* = (c_1^*, \dots, c_l^*)$ of ideal coefficients in (15) that solves the linear system $Gc^* = \bar{g}$ with the Gram matrix $G = (\langle \sqrt{T_{q_X}} f_k, \sqrt{T_{q_X}} f_u \rangle_{\mathcal{H}})_{k,u=1}^l$ and the right-hand side vector $\bar{g} = (\langle \sqrt{T_{q_X}} f_q, \sqrt{T_{q_X}} f_k \rangle_{\mathcal{H}})_{k=1}^l$. Let us provide a prove of this short observation in the next lemma. Note that the entries G and \bar{g} can equivalently also be formulated in terms of $\langle \cdot, \cdot \rangle_{L^2(q_X)}$, as done in the main text. We are going to use this formulation in the next lemma in order to be compatible with the main text (switching to the inner products in terms of \mathcal{H} would not change the argument of the proof at all):

Lemma 4. *The best $L^2(q_X)$ -approximation f^* of the target regression function f_q by linear combinations corresponds to the vector $c^* = (c_1^*, \dots, c_l^*) = G^{-1}\bar{g}$.*

Proof. Let us denote (16) by $f(c)$ and rewrite this expression appropriately:

$$f(c) = \sum_{i,j=1}^l c_i c_j \langle f_i, f_j \rangle_{L^2(q_X)} - 2 \sum_{i=1}^l c_i \langle f_i, f_q \rangle_{L^2(q_X)} + \langle f_q, f_q \rangle_{L^2(q_X)}.$$

Taking the derivative with respect to c_i yield:

$$\frac{\partial f(c)}{\partial c_i} = 2 \left(\sum_{j=1}^l c_j \langle f_i, f_j \rangle_{L^2(q_X)} - \langle f_i, f_q \rangle_{L^2(q_X)} \right).$$

Setting these derivatives to zero (for all $i \in \{1, \dots, l\}$) gives the claimed equation. Noting that the Hessian is equal to $2G$ (and thus positive-definite) ensures that c^* is a global minimum of f . \square

But, of course, neither Gram matrix G nor the vector \bar{g} is accessible, because there is no access to the target measure q_X , so we switch to the empirical counterparts \tilde{G} and \tilde{g} .

Then the following lemma is helpful to gain some information on the error made by the empirical average:

Lemma 5. *With probability $1 - \delta$ we have*

$$\begin{aligned} \left| \left\langle \sqrt{T_{q_X}} f_u, \sqrt{T_{q_X}} f_k \right\rangle_{\mathcal{H}} - \frac{1}{m} \sum_{j=1}^m \langle f_k(x'_j), f_u(x'_j) \rangle_{\mathcal{Y}} \right| &\leq C \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) m^{-\frac{1}{2}}, \\ \left| \left\langle \sqrt{T_{q_X}} f_k, \sqrt{T_{q_X}} f_q \right\rangle_{\mathcal{H}} - \frac{1}{n} \sum_{i=1}^n \beta(x_i) \langle f_k(x_i), f_q(x_i) \rangle_{\mathcal{Y}} \right| &\leq C \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) (n^{-\frac{1}{2}} + m^{-\frac{1}{2}}), \end{aligned}$$

where $C > 0$ does not depend on n, m and δ .

Proof. Keeping in mind that $f_q, f_k \in \mathcal{H}$ we have

$$\begin{aligned} \left\langle \sqrt{T_{q_X}} f_u, \sqrt{T_{q_X}} f_k \right\rangle_{\mathcal{H}} &= \langle T_{\mathbf{x}'} f_k, f_u \rangle_{\mathcal{H}} + \langle (T_{q_X} - T_{\mathbf{x}'}) f_u, f_k \rangle_{\mathcal{H}} \\ &= \left\langle \frac{1}{m} \sum_{j=1}^m K_{x'_j} K_{x'_j}^* f_k, f_u \right\rangle_{\mathcal{H}} + \langle (T_{q_X} - T_{\mathbf{x}'}) f_u, f_k \rangle_{\mathcal{H}} \\ &= \frac{1}{m} \sum_{j=1}^m \left\langle K_{x'_j}^* f_k, K_{x'_j}^* f_u \right\rangle_{\mathcal{Y}} + \langle (T_{q_X} - T_{\mathbf{x}'}) f_u, f_k \rangle_{\mathcal{H}} \\ &= \frac{1}{m} \sum_{j=1}^m \langle f_k(x'_j), f_u(x'_j) \rangle_{\mathcal{Y}} + \langle (T_{q_X} - T_{\mathbf{x}'}) f_u, f_k \rangle_{\mathcal{H}}. \end{aligned}$$

Moreover, from (12) with probability $1 - \delta$ we have that

$$\left| \langle (T_{q_X} - T_{\mathbf{x}'}) f_u, f_k \rangle_{\mathcal{H}} \right| \leq C \|f_u\|_{\mathcal{H}} \|f_k\|_{\mathcal{H}} \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) m^{-\frac{1}{2}}.$$

Then

$$\left| \left\langle \sqrt{T_{q_X}} f_u, \sqrt{T_{q_X}} f_k \right\rangle_{\mathcal{H}} - \frac{1}{m} \sum_{j=1}^m \langle f_k(x'_j), f_u(x'_j) \rangle_{\mathcal{Y}} \right| \leq C \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) m^{-\frac{1}{2}}.$$

Now, we prove the second statement in Lemma 5. We have

$$\begin{aligned} \left\langle \sqrt{T_{q_X}} f_k, \sqrt{T_{q_X}} f_q \right\rangle_{\mathcal{H}} &= \langle f_k, T_{q_X} f_q \rangle_{\mathcal{H}} = \langle f_k, T_{q_X} f_q - g_{\mathbf{x}, \mathbf{y}, \beta} \rangle_{\mathcal{H}} + \langle f_k, g_{\mathbf{x}, \mathbf{y}, \beta} \rangle_{\mathcal{H}} \\ &= \frac{1}{n} \sum_{i=1}^n \beta(x_i) \langle f_k, K_{x_i} y_i \rangle_{\mathcal{H}} + \langle f_k, T_{q_X} f_q - g_{\mathbf{x}, \mathbf{y}, \beta} \rangle_{\mathcal{H}} \\ &= \frac{1}{n} \sum_{i=1}^n \beta(x_i) \langle K_{x_i}^* f_k, y_i \rangle_{\mathcal{Y}} + \langle f_k, T_{q_X} f_q - g_{\mathbf{x}, \mathbf{y}, \beta} \rangle_{\mathcal{H}} \\ &= \frac{1}{n} \sum_{i=1}^n \beta(x_i) \langle f_k(x_i), y_i \rangle_{\mathcal{Y}} + \langle f_k, T_{q_X} f_q - g_{\mathbf{x}, \mathbf{y}, \beta} \rangle_{\mathcal{H}}. \end{aligned}$$

From Lemma 1, with probability $1 - \delta$ we have

$$\begin{aligned} &\|T_{q_X} f_q - g_{\mathbf{x}, \mathbf{y}, \beta}\|_{\mathcal{H}} \\ &\leq \|T_{q_X} f_q - T_{\mathbf{x}'} f_q\|_{\mathcal{H}} + \|T_{\mathbf{x}'} f_q - g_{\mathbf{x}, \mathbf{y}, \beta}\|_{\mathcal{H}} \\ &\leq \|T_{q_X} f_q - T_{\mathbf{x}'} f_q\|_{\mathcal{H}} + \|T_{\mathbf{x}'} f_q - T_{\mathbf{x}, \beta} f_q\|_{\mathcal{H}} + \|T_{\mathbf{x}, \beta} f_q - g_{\mathbf{x}, \mathbf{y}, \beta}\|_{\mathcal{H}} \\ &\leq C \|f_q\|_{\mathcal{H}} \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) m^{-\frac{1}{2}} + C \|f_q\|_{\mathcal{H}} \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) (n^{-\frac{1}{2}} + m^{-\frac{1}{2}}) + \|T_{\mathbf{x}, \beta} f_q - g_{\mathbf{x}, \mathbf{y}, \beta}\|_{\mathcal{H}} \\ &\leq C \|f_q\|_{\mathcal{H}} \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) m^{-\frac{1}{2}} + C \|f_q\|_{\mathcal{H}} \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) (n^{-\frac{1}{2}} + m^{-\frac{1}{2}}) + C \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) n^{-\frac{1}{2}}. \end{aligned}$$

Then

$$\langle f_k, T_{q_X} f_q - g_{\mathbf{x}, \mathbf{y}, \beta} \rangle_{\mathcal{H}} \leq C \|f_k\|_{\mathcal{H}} \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) (n^{-\frac{1}{2}} + m^{-\frac{1}{2}}).$$

Therefore,

$$\left| \left\langle \sqrt{T_{q_X}} f_k, \sqrt{T_{q_X}} f_q \right\rangle_{\mathcal{H}} - \frac{1}{n} \sum_{i=1}^n \beta(x_i) \langle f_k(x_i), y_i \rangle_{\mathcal{Y}} \right| \leq C \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) (n^{-\frac{1}{2}} + m^{-\frac{1}{2}}).$$

□

Towards our main generalization bound Next we use similar arguments as in Theorem 4 from Gizewski et al. (2022) to obtain our main result, Theorem 1. Lemma 5 suggests to approximate G and \bar{g} by their empirical counterparts:

$$\tilde{G} = \left(\frac{1}{m} \sum_{j=1}^m \langle f_k(x'_j), f_u(x'_j) \rangle_{\mathcal{Y}} \right)_{k,u=1}^l, \quad (17)$$

$$\tilde{g} = \left(\frac{1}{n} \sum_{i=1}^n \beta(x_i) \langle y_i, f_k(x_i) \rangle_{\mathcal{Y}} \right)_{k=1}^l \quad (18)$$

which can be effectively computed from data samples. Moreover, again from Lemma 5 we can argue that with probability $1 - \delta$ it holds:

$$\|\bar{g} - \tilde{g}\|_{\mathbb{R}^l} \leq C \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) (n^{-\frac{1}{2}} + m^{-\frac{1}{2}}), \quad (19)$$

$$\|G - \tilde{G}\|_{\mathcal{L}(\mathbb{R}^l)} \leq C \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) m^{-\frac{1}{2}}. \quad (20)$$

With the matrix \tilde{G} at hand one can easily check whether or not it is well-conditioned and \tilde{G}^{-1} exists (otherwise one needs to get rid of models with similar performance). Thus the norms $\|\tilde{G}\|_{\mathcal{L}(\mathbb{R}^l)}$ and $\|\tilde{G}^{-1}\|_{\mathcal{L}(\mathbb{R}^l)}$ can be bounded independently of m and n , due to the fact that all their entries can be bounded as follows (we only do the calculation for the entries of \tilde{G}):

$$\begin{aligned} |\tilde{G}_{k,u}| &\leq \frac{1}{m} \sum_{j=1}^m \left| \langle f_k(x'_j), f_u(x'_j) \rangle_{\mathcal{Y}} \right| = \frac{1}{m} \sum_{j=1}^m \left| \langle K_{x'_j}^* f_k, K_{x'_j}^* f_u \rangle_{\mathcal{Y}} \right| \\ &= \frac{1}{m} \sum_{j=1}^m \left| \langle K_{x'_j} K_{x'_j}^* f_k, f_u \rangle_{\mathcal{H}} \right| = \frac{1}{m} \sum_{j=1}^m \left| \langle T_{x'_j} f_k, f_u \rangle_{\mathcal{H}} \right| \\ &\leq \frac{1}{m} \sum_{j=1}^m \|T_{x'_j}\|_{\mathcal{L}(\mathcal{H})} \|f_k\|_{\mathcal{H}} \|f_u\|_{\mathcal{H}} \leq \kappa \gamma_l^2, \end{aligned}$$

where we used the reproducing property (5) to obtain the equality in the first line and (10) for the last inequality. Now assume that m is so large that with probability $1 - \delta$ we have

$$\|G - \tilde{G}\|_{\mathcal{L}(\mathbb{R}^l)} < \frac{1}{\|\tilde{G}^{-1}\|_{\mathcal{L}(\mathbb{R}^l)}}. \quad (21)$$

Moreover we can use the following simple manipulation:

$$G^{-1} = \tilde{G}^{-1}(G\tilde{G}^{-1})^{-1} = \tilde{G}^{-1}(I - (I - G\tilde{G}^{-1}))^{-1} = \tilde{G}^{-1}(I - (\tilde{G} - G)\tilde{G}^{-1})^{-1}.$$

Then (21) ensures that the Neumann series for $(I - (\tilde{G} - G)\tilde{G}^{-1})^{-1}$ converges and we obtain the following bound:

$$\|G^{-1}\|_{\mathcal{L}(\mathbb{R}^l)} \leq \frac{\|\tilde{G}^{-1}\|_{\mathcal{L}(\mathbb{R}^l)}}{1 - \|\tilde{G}^{-1}\|_{\mathcal{L}(\mathbb{R}^l)} \|G - \tilde{G}\|_{\mathcal{L}(\mathbb{R}^l)}} = O(1). \quad (22)$$

Now we are in the position to prove our main generalization bound (4) for unsupervised domain adaptation:

Proof of Theorem 1. We have already discussed that the coefficients in the best approximation f^* to f_q are given by $c^* = (c_1^*, c_2^*, \dots, c_l^*) = G^{-1}\bar{g}$. Since:

$$G^{-1}(\tilde{g} - \bar{g}) + G^{-1}(G - \tilde{G})\tilde{c} = G^{-1}\tilde{g} - c^* + \tilde{c} - G^{-1}\tilde{g} = \tilde{c} - c^*$$

then from (19)–(22) with probability $1 - \delta$ we have

$$\begin{aligned}\|\tilde{c} - c^*\|_{\mathbb{R}^l} &\leq \|G^{-1}\|_{\mathcal{L}(\mathbb{R}^l)} (\|\tilde{g} - \bar{g}\|_{\mathbb{R}^l} + \|G - \tilde{G}\|_{\mathcal{L}(\mathbb{R}^l)} \|\tilde{c}\|_{\mathbb{R}^l}) \\ &\leq C \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) (n^{-\frac{1}{2}} + m^{-\frac{1}{2}}).\end{aligned}\quad (23)$$

Moreover:

$$\begin{aligned}\mathcal{R}_q(\tilde{f}) - \mathcal{R}_q(f_q) &= \left\| \sqrt{T_{qX}}(\tilde{f} - f_q) \right\|_{\mathcal{H}}^2 \\ &\leq \left(\left\| \sqrt{T_{qX}}(f^* - f_q) \right\|_{\mathcal{H}} + \left\| \sqrt{T_{qX}}(\tilde{f} - f^*) \right\|_{\mathcal{H}} \right)^2 \\ &\leq 2 \left\| \sqrt{T_{qX}}(f^* - f_q) \right\|_{\mathcal{H}}^2 + 2 \left\| \sqrt{T_{qX}}(\tilde{f} - f^*) \right\|_{\mathcal{H}}^2 \\ &= 2(\mathcal{R}_q(f^*) - \mathcal{R}_q(f_q)) + 2 \left\| \sqrt{T_{qX}}(\tilde{f} - f^*) \right\|_{\mathcal{H}}^2 \\ &\leq 2(\mathcal{R}_q(f^*) - \mathcal{R}_q(f_q)) + 2 \left(\sum_{k=1}^l |c_k^* - \tilde{c}_k| \left\| \sqrt{T_{qX}}f_k \right\|_{\mathcal{H}} \right)^2 \\ &\leq 2(\mathcal{R}_q(f^*) - \mathcal{R}_q(f_q)) + 2l \|c^* - \tilde{c}\|_{\mathbb{R}^l}^2 \max_k \left\| \sqrt{T_{qX}}f_k \right\|_{\mathcal{H}}^2 \\ &\leq 2(\mathcal{R}_q(f^*) - \mathcal{R}_q(f_q)) + 2 \left\| \sqrt{T_{qX}} \right\|_{\mathcal{L}(\mathcal{H})}^2 l \gamma_l^2 \|c^* - \tilde{c}\|_{\mathbb{R}^l}^2,\end{aligned}\quad (24)$$

The statement of the theorem follows now from (23)–(24) (using again the inequality $(a + b)^2 \leq 2(a^2 + b^2)$). \square

On the dependence of the error bound on the number l of models An interesting question is, how the bound in Eq. (4) depends on l . To this end, let us have a look at the second term in the last line in Eq. (24): $\left\| \sqrt{T_{qX}} \right\|_{\mathcal{L}}$ analyzes a sampling operator, thus does not depend on the number of models, same goes for γ_l , which is just a uniform bound on all our models. To analyze $\|c^* - \tilde{c}\|_{\mathbb{R}^l}^2$, let us have a look at the individual factors in the first inequality of Eq. (23). To not overload notation, $C > 0$ is used here for any absolute constant that is independent of l, m, n and δ .

- $\|G - \tilde{G}\|_{\mathcal{L}(\mathbb{R}^l)}$: The individual entries of $G - \tilde{G}$ are (in absolute values) bounded by Lemma 5. The proof arguments only involve norm bounds on the associated sampling operators, the uniform bound γ_l on all the models and the bound B on β , thus the absolute constant C there is independent of l . By the definition of the matrix Frobenius norm, we thus get

$$\|G - \tilde{G}\|_{\mathcal{L}(\mathbb{R}^l)} \leq Cl \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) (n^{-\frac{1}{2}} + m^{-\frac{1}{2}}) \quad (25)$$

- $\|\tilde{g} - \bar{g}\|_{\mathbb{R}^l}$: Similar arguments as before lead to $\|\tilde{g} - \bar{g}\|_{\mathcal{L}(\mathbb{R}^l)} \leq C\sqrt{l} \left(\log^{\frac{1}{2}} \frac{1}{\delta} \right) (n^{-\frac{1}{2}} + m^{-\frac{1}{2}})$
- $\|G^{-1}\|_{\mathcal{L}(\mathbb{R}^l)}$: It is natural to assume that there is some constant $c \geq \|\tilde{G}^{-1}\|_{\mathcal{L}(\mathbb{R}^l)}$. Otherwise, we can, e.g., orthogonalize our models and coefficients without changing the aggregation, but with reducing the conditioning number (i.e., with reducing $\|\tilde{G}^{-1}\|$). It is also natural to assume that m and n are large enough such that $l(n^{-\frac{1}{2}} + m^{-\frac{1}{2}}) < \frac{1}{2c}$. Then applying Eq. (25) to Eq. (22), we can deduce that $\|G^{-1}\|_{\mathcal{L}(\mathbb{R}^l)} \leq 2c$.
- $\|\tilde{c}\|_{\mathbb{R}^l}$: This quantity can also be assumed to be known independently of l , since it is given by our data only.

Combining the previous points gives us $\|c^* - \tilde{c}\|_{\mathbb{R}^l}^2 \leq Cl^2 (\log \frac{1}{\delta}) (n^{-1} + m^{-1})$ which finally leads to the refined bound

$$\mathcal{R}_q(\tilde{f}) - \mathcal{R}_q(f_q) \leq 2(\mathcal{R}_q(f^*) - \mathcal{R}_q(f_q)) + Cl^3 \left(\log \frac{1}{\delta} \right) (n^{-1} + m^{-1})$$

for sufficiently large l, m and n and error probability $\delta > 0$.

B CONSTRUCTION OF FUNCTION SPACES

Let us give a short discussion on the construction of our required function space mentioned in the previous Section A, the reproducing kernel space \mathcal{H} . As mentioned already in the main text, the explicit knowledge of \mathcal{H} is not required, we just need to rely on its existence. First, any of our models f can be regarded as an element of some reproducing kernel space (RKHS) $\hat{\mathcal{H}}$ satisfying the assumptions 1. This is immediate if $f : \mathcal{X} \rightarrow \mathbb{R}$ is a real valued continuous function and we take $k(x, y) = f(x)f(y)$ as the associated reproducing kernel. In the case $f : \mathcal{X} \rightarrow \mathcal{Y}$ and \mathcal{Y} is finite dimensional, it is not hard to see that a similar construction is possible, as this case can again be boiled down to the construction of a kernel with real-valued output, see e.g. Remark 1 in Caponnetto & De Vito (2007) for details.

Overall we end up with a finite sequence of spaces $(\mathcal{H}_k)_{k=1}^{l+1}$ of functions living on the same domain \mathcal{X} (we have $l+1$ as we also take into account the regression function), and the existence of a RKHS containing all given models and the regression function is not a real restriction. For example, in case of real valued functions, this assumption is automatically satisfied, as linear combinations of functions with the same domain which stem from a finite sequence of RKHSs belong to an RKHS. This follows from a classical result by N. Aronszajn and R. Godement, see e.g. Pereverzyev (2022, Theorem 1.4.).

There is also ongoing research on constructing function spaces (and especially associated reproducing kernels) for families of neural networks that are used in applications, see e.g. Ma & Wu (2022) for ReLU networks, Fermanian et al. (2021) for recurrent networks and Bietti & Mairal (2017; 2019) for convolutional neural networks. Incorporating these into our work may lead to refined generalization bounds that also reflect the nature of our models. We leave the details open for future work.

C DATASETS

This section provides an overview over all applied datasets from language, image, and time-series domains.

Illustrative example: For the illustrative example (Figure 1 in the main paper) we rely on the following setting, taken from Shimodaira (2000); Sugiyama et al. (2007); You et al. (2019): The data points are labelled with $y = \frac{\sin(\pi x)}{\pi x}$ with random noise sampled from the normal distribution $\mathcal{N}(0, (\frac{1}{4})^2)$. Moreover $p_x \sim \mathcal{N}(1, \frac{1}{4})$, $q_x \sim \mathcal{N}(2, (\frac{1}{4})^2)$. The density ratio β can be computed analytically and is bounded. We aggregate several linear models with our approach and compare it to the optimal linear model, whose coefficients have been evaluated using a computer algebra system.

Academic Dataset We rely on the Transformed Moons dataset (Zellinger et al., 2021), allowing us to visualize and address low-dimensional input data. The dataset consists of two-dimensional input data points forming two classes with a “moon-shaped” support. The shift from source to target domain is simulated by a transformation in input space as depicted in Figure 3. The results are shown in the following table:

Language Dataset To evaluate our method on a language task, we rely on the Amazon Reviews (Blitzer et al., 2006) dataset. This dataset consists of text reviews from four domains: books (B), DVDs (D), electronics (E), and kitchen appliances (K). Reviews are encoded in 5000 dimensional feature vectors of bag-of-words unigrams and bigrams with binary labels: label 0 if the product is ranked by 1 to 3 stars, and label 1 if the product is ranked by 4 or 5 stars. From the four categories, we obtain twelve domain adaptation tasks, where each category serves once as source domain and once

Table 2: Mean and standard deviation (after \pm) of target classification accuracy on Transformed Moons dataset over three different random initialization of model weights and 11 domain adaptation methods.

Method	Transformed Moons								
	Heuristic			Theoretical error guarantees					
SO	TMV	TMR	TCR	SOR	IVW	DEV	DVA (ours)	TB	
HoMM	0.994(± 0.003)	0.994(± 0.000)	1.000(± 0.003)	1.000(± 0.003)	0.985(± 0.012)	0.994(± 0.003)	0.994(± 0.003)	0.992(± 0.000)	0.998(± 0.000)
AdvSKM	0.985(± 0.000)	1.000(± 0.000)	0.998(± 0.000)	0.998(± 0.000)	0.987(± 0.014)	0.985(± 0.000)	0.985(± 0.000)	1.000(± 0.000)	1.000(± 0.000)
DIRT	0.964(± 0.009)	0.827(± 0.018)	0.830(± 0.017)	0.981(± 0.003)	0.990(± 0.006)	0.964(± 0.009)	0.869(± 0.009)	0.983(± 0.003)	0.966(± 0.000)
DDC	0.994(± 0.023)	0.998(± 0.000)	0.996(± 0.000)	0.996(± 0.000)	0.981(± 0.009)	0.994(± 0.023)	0.994(± 0.023)	0.998(± 0.000)	0.998(± 0.000)
CMD	0.990(± 0.006)	0.956(± 0.000)	0.964(± 0.000)	0.996(± 0.000)	0.994(± 0.006)	0.990(± 0.006)	0.990(± 0.006)	0.996(± 0.000)	1.000(± 0.000)
MMDA	0.990(± 0.006)	1.000(± 0.003)	1.000(± 0.003)	1.000(± 0.003)	0.996(± 0.009)	0.990(± 0.006)	0.990(± 0.006)	1.000(± 0.003)	1.000(± 0.003)
CoDAS	0.979(± 0.002)	1.000(± 0.002)	1.000(± 0.002)	1.000(± 0.002)	0.989(± 0.016)	0.989(± 0.016)	0.979(± 0.016)	1.000(± 0.000)	1.000(± 0.005)
Deep-Coral	1.000(± 0.000)	0.985(± 0.009)	1.000(± 0.000)	1.000(± 0.000)	1.000(± 0.000)	1.000(± 0.000)			
CDAN	1.000(± 0.000)	0.985(± 0.026)	1.000(± 0.000)	1.000(± 0.000)	1.000(± 0.000)	1.000(± 0.000)			
DANN	0.994(± 0.006)	1.000(± 0.000)	1.000(± 0.000)	1.000(± 0.000)	0.994(± 0.016)	0.994(± 0.006)	0.994(± 0.006)	1.000(± 0.003)	1.000(± 0.003)
DSAN	0.990(± 0.009)	1.000(± 0.000)	1.000(± 0.000)	1.000(± 0.000)	1.000(± 0.003)	0.990(± 0.009)	0.990(± 0.009)	1.000(± 0.000)	1.000(± 0.000)
Avg.	0.989(± 0.008)	0.980(± 0.006)	0.981(± 0.007)	0.997(± 0.002)	0.989(± 0.010)	0.989(± 0.008)	0.981(± 0.022)	0.997(± 0.002)	0.997(± 0.005)

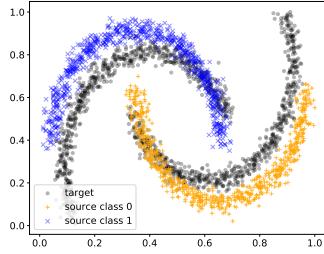


Figure 3: Transformed Moons dataset. Source data is depicted as blue + and orange \times . Target data points are shown as black dots.

as target domain (e.g., see Table 15). We follow similar data splits as previous works (Chen et al., 2012; Louizos et al., 2016; Ganin et al., 2016). In particular, we use 4000 labeled source examples and 4000 unlabeled target examples for training, and over 1000 examples for testing.

Image Dataset Our third dataset is MiniDomainNet, which is based on the DomainNet-2019 dataset (Peng et al., 2019) consisting of six different image domains (Quickdraw: Q, Real: R, Clipart: C, Sketch: S, Infograph: I, and Painting: P). We follow Zellinger et al. (2021) and rely on the reduced version of DomainNet-2019, referred to as MiniDomainNet, which reduces the number of classes to the top-five largest representatives in the training set across all six domains. To further improve computation time, we rely on a ImageNet (Krizhevsky et al., 2012) pre-trained ResNet-18 (He et al., 2016) backbone. Therefore, we assume that the backbone has learned lower-level filters suitable for the “Real” image category, and we only need to adapt to the remaining five domains (e.g., Clipart, Sketch). This results in five domain adaptation tasks.

Time-Series Dataset We based our time-series experiments on the four datasets included in the AdaTime benchmark suite (Ragab et al., 2022), which consists of UCI-HAR, WISDM, HHAR, and Sleep-EDF. The suite includes four representative datasets spanning 20 cross-domain real-world scenarios, i.e., human activity recognition and sleep stage classification. The first dataset is the *Human Activity Recognition* (HAR) (Anguita et al., 2013) dataset from the UC Irvine Repository denoted as UCI-HAR, which contains data from three motion sensors (accelerometer, gyroscope and body-worn sensors) gathered using smartphones from 30 different subjects. It classifies their activities in several categories, namely, walking, walking upstairs, downstairs, standing, sitting, and lying down. The WISDM (Kwapisz et al., 2011) dataset is a class-imbalanced variant from collected accelerometer sensors, including GPS data, from 29 different subjects which are performing similar activities as in the UCI-HAR dataset. The *Heterogeneity Human Activity Recognition* (HHAR) (Stisen et al., 2015) dataset investigate sensor-, device- and workload-specific heterogeneities using 36 smartphones and smartwatches, consisting of 13 different device models from four manufacturers. Finally, the *Sleep Stage Classification* time-series setting aims to classify the electroencephalography

(EEG) signals into five stages i.e., Wake (W), Non-Rapid Eye Movement stages (N1, N2, N3), and Rapid Eye Movement (REM). Analogous to Ragab et al. (2022); Eldele et al. (2021), we adopt the Sleep-EDF-20 dataset obtained from PhysioBank (Goldberger et al., 2000), which contains EEG readings from 20 healthy subjects. For all datasets, each subject is treated as an own domain, and adopt from a source subject to a target subject.

D EXPERIMENTAL SETUP

This section is meant to provide further details on the overall computational setting of our experiments. We start by giving an overview on the used computational resources for the specific datasets and the implementation tools. Next, we describe the network architectures for the individual datasets in greater detail. In the third subsection we elaborate on the construction of our models, and the fourth subsection is devoted to matrix inversion. Finally, in the last subsection, we describe the detailed empirical results and give the complete tables.

D.1 COMPUTATIONAL RESOURCES AND IMPLEMENTATIONS

Overall, to compute the results in our tables, we trained 16680 models with an approximate computational budget of 1500 GPU/hours on one high-performance computing station using 8×NVIDIA P100 16GB, 512GB RAM, 40 Cores Xeon(R) CPU E5-2698 v4 @ 2.20GHz on CentOS Linux 7.

Transformed Moons: 11 methods × 14 parameters × 1 domain adaptation tasks × 3 seeds + 3 density estimator classifier = 465 trained models

Amazon Reviews: 11 methods × 14 parameters × 12 domain adaptation tasks × 3 seeds + 12 × 3 density estimator classifier = 5580 trained models

MiniDomainNet: 11 methods × 8 parameters × 5 domain adaptation tasks × 3 seeds + 5 × 3 density estimator classifier = 1335 trained models

UCI-HAR: 11 methods × 14 parameters × 5 domain adaptation tasks × 3 seeds + 5 × 3 density estimator classifier = 2325 trained models

Sleep-EDF: 11 methods × 14 parameters × 5 domain adaptation tasks × 3 seeds + 5 × 3 density estimator classifier = 2325

HHAR: 11 methods × 14 parameters × 5 domain adaptation tasks × 3 seeds + 5 × 3 density estimator classifier = 2325 trained models

WISDM: 11 methods × 14 parameters × 5 domain adaptation tasks × 3 seeds + 5 × 3 density estimator classifier = 2325 trained models

In Total: $465 + 5580 + 1335 + 4 \times 2325 = 16680$ trained models

All methods have been implemented in Python using the *Pytorch* (Paszke et al., 2017, BSD license) library. For monitoring the runs we used *Weights & Biases* (Biewald, 2020, MIT license). We use *Scikit-learn* (Pedregosa et al., 2011) library for evaluation measures and toy datasets, and the *TQDM* (da Costa-Luis, 2019) library, and *Tensorboard* (Abadi et al., 2015) for keeping track of the progress of our experiments. We built parts of our implementation on the codebase of Zellinger et al. (2021, MIT License) and Ragab et al. (2022, MIT License).

D.2 ARCHITECTURES AND TRAINING SETUP

In this subsection, we provide details on the model architectures and the training setup for every dataset. Our base architectures are based on the AdaTime benchmark suite, which is a large-scale evaluation of domain adaptation algorithms on time-series data. We extended the benchmark suite to support 11 state-of-the-art model architectures on multiple dataset types ranging from language, image to time-series data, addressed by Transformed Moons, Amazon Reviews, MiniDomainNet and the four time-series datasets (UCI-HAR, WISDM, HHAR, and Sleep-EDF) spanning in total 38 cross-domain real-world scenarios.

Transformed Moons For the Transformed Moons dataset we use two sequential blocks with fully-connected layers, 1D-BatchNorm, ReLU activation functions and Dropout. The full architecture specification can be found in Table 3. The domain classifier (density ratio estimator) uses the same architecture. We train the class prediction models for 50 epochs and the domain classifier for 80 epochs with learning rate 0.001, weight decay 0.0001 and batchsize 128 using the Adam optimizer (Kingma & Ba, 2014). We share the same base architecture and training setup across every domain adaption method (e.g., DANN, HoMM, CMD). Additional hyper-parameters are reported in Table 9.

Table 3: Model architecture for the Transformed Moons dataset. The values for neural network layers correspond to the number of output units.

Architecture		
	Layers	Values
	Input units	2
MLP Block 1	Fully-connected Layer	128
	Batch Normalization 1D Layer	128
	ReLU Activation	
	Dropout	
MLP Block 2	Fully-connected Layer	128
	Batch Normalization 1D Layer	128
	ReLU Activation	
	Dropout	
	Fully-connected Layer	128
Methods		
See Table 7 and Table 8 for details.		

Amazon Reviews For the Amazon Reviews dataset we use two sequential blocks with fully-connected layers, 1D-BatchNorm, ReLU activation function and Dropout, analogous to the setup for Transformed Moons. We also use the same architecture for the domain classifier. We train the class prediction models for 50 epochs and the domain classifier for 80 epochs with learning rate 0.001, weight decay 0.0001 and batchsize 128 using the Adam optimizer (Kingma & Ba, 2014). We share the same base architecture and training setup across every domain adaption method (e.g., DANN, HoMM, CMD). Additional hyper-parameters are reported in Table 9.

Table 4: Model architecture for the Amazon Reviews dataset. The values for neural network layers correspond to the number of output units.

Architecture		
	Layers	Values
	Input units	5000
MLP Block 1	Fully-connected Layer	128
	Batch Normalization 1D Layer	128
	ReLU Activation	
	Dropout	
MLP Block 2	Fully-connected Layer	128
	Batch Normalization 1D Layer	128
	ReLU Activation	
	Dropout	
	Fully-connected Layer	128
Methods		
See Table 7 and Table 8 for details.		

MiniDomainNet Following the pre-trained setup from Peng et al. (2019), we use a frozen ResNet-18 backbone model which was trained on ImageNet, and operate subsequent computations on the

512 dimensional extracted features. To alleviate overfitting effects on pre-computed features, we perform data augmentation on the images of each batch and forward each batch through the backbone. We incorporate zero padding before resizing the images to 256x256 to avoid image distortions. Furthermore, in alignment with data augmentation techniques from Shorten & Khoshgoftaar (2019), we perform random resized cropping to 224x224 with a random viewport between 70% and 100% of the original image, random horizontal flipping, color jittering of 0.25% on each RGB channel, and a ± 2 degree rotation.

After the ResNet-18 backbone output, we add a projection layer, and define the domain adaptation layers on which we use the domain adaptation methods to align the representations. The backbone and projection layers are defined as a common architecture across the different domain adaptation methods. Additional layers are further added for the classification networks, according to the requirements of the individual domain adaptation methods (e.g., CMD, HoMM). The number of layers/neurons in the upper layers of our architecture have been tuned in order to achieve the best performance in the source-only setup. See Table 5 for a detailed description of the architecture used. We perform experiments on all 5 domain adaptation tasks as defined in Section C for each of the previously listed methods, and with 3 repetitions based on different random weights initialization. All class prediction models have been trained for 60 epochs and domain classifiers for 100 epochs with Adam optimizer, a learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, batchsize of 128 and weight decay of 0.0001. Additional hyper-parameters are reported in Table 9.

Table 5: Model architecture for the MiniDomainNet dataset. The values for neural network layers correspond to the number of output units.

Architecture		
Layers	Values	
Backbone Output Layer	ResNet-18 (Adaptive Average Pooling Layer)	512
Fully-connected Layer		128
Methods		
See Table 7 and Table 8 for details.		

AdaTime Unless stated otherwise, we follow the implementation and hyper-parameter settings as reported in Ragab et al. (2022). We extended the AdaTime suite to comprise a collection of 11 domain adaptation algorithms. We learned all domain adaptations models according to the following approaches (see also Table 6, Table 7 and Table 8): Deep Domain Confusion (DDC) (Tzeng et al., 2014), Correlation Alignment via Deep Neural Networks (Deep-Coral) (Sun et al., 2017), Higher-order Moment Matching (HoMM) (Chen et al., 2020), Minimum Discrepancy Estimation for Deep Domain Adaptation (MMDA) (Rahman et al., 2020), Central Moment Discrepancy (CMD) (Zellinger et al., 2017), Deep Subdomain Adaptation (DSAN) (Zhu et al., 2021), Domain-Adversarial Neural Networks (DANN) (Ganin et al., 2016), Conditional Adversarial Domain Adaptation (CDAN) (Long et al., 2018), A DIRT-T Approach to Unsupervised Domain Adaptation (DIRT) (Shu et al., 2018), Convolutional deep Domain Adaptation model for Time-Series data (CoDATS) (Wilson et al., 2020), and Adversarial Spectral Kernel Matching (AdvSKM) (Liu & Xue, 2021). The backbone architecture of all models is a 1D-CNN network. It consists of three CNN blocks and each block has a 1D convolutional layer, followed by 1D batch normalization layer, ReLU activation function, 1D max pooling and Dropout. In the first block, the kernel size of the convolutional layer is set according to the dataset as reported in Ragab et al. (2022). After the convolutional blocks, we apply an 1D adaptive pooling layer. All methods are trained for 100 epochs on all datasets. The batch size is 32, except for Sleep-EDF, where we use batch size of 128. All models are trained with Adam optimizer (Kingma & Ba, 2014) and weight decay of 10^{-4} . Additional hyper-parameters are reported in Table 10.

D.3 MODEL SEQUENCE

Our algorithm, IWA, constructs an ensemble from a sequence of different classifiers, e.g. obtained from a sequence of possible hyper-parameter configurations in domain adaptation algorithms. To obtain this sequence of models, we train multiple models for every domain adaptation task across all datasets with different hyper-parameter choices. For the experiments on the language, im-

Table 6: Model backbone for the AdaTime suite. Kernel size, stride, output channels of the convolutional layers are dataset dependent and are chosen according to Ragab et al. (2022).

Architecture	
	Layers
Conv Block 1	Convolutional 1D Layer Batch Normalization 1D Layer ReLU Activation Max Pooling 1D Layer Dropout
Conv Block 2	Convolutional 1D Layer Batch Normalization 1D Layer ReLU Activation Max Pooling 1D Layer Dropout
Conv Block 3	Convolutional 1D Layer Batch Normalization 1D Layer ReLU Activation Max Pooling 1D Layer Dropout
	Adaptive Pooling 1D Layer
Methods	
See Table 7 and Table 8 for details.	

Table 7: Model architecture for the AdaTime dataset. Layer hyper-parameters are dataset dependent and are chosen according to Ragab et al. (2022).

Method Architectures (Part 1)	
DANN	
Class Output Head	Fully-connected Layer
Domain Classifier Head	Fully-connected Layer ReLU Activation Fully-connected Layer ReLU Activation Fully-connected Layer
DeepCoral	
Class Output Head	Fully-connected Layer
DDC	
Class Output Head	Fully-connected Layer
HoMM	
Class Output Head	Fully-connected Layer
CoDATS	
Class Output Head	Fully-connected Layer ReLU Activation Fully-connected Layer ReLU Activation Fully-connected Layer
DSAN	
Class Output Head	Fully-connected Layer

age and academic dataset, the values of the hyper-parameters are shown in Table 9. For the time-series datasets we use the best hyper-parameters in Ragab et al. (2022), and, to obtain a good sequence of values. For all settings except MiniDomainnet, we multiply each parameter by

Table 8: Model architecture for the AdaTime dataset. Hyper-parameters are dataset dependent and are chosen according to Ragab et al. (2022).

Method Architectures (Part 2)	
AdvSKM	
Class Output Head	Fully-connected Layer
AdvSKM Embedder 1	Fully-connected Layer
	Fully-connected Layer
	Batch Normalization 1D Layer
	Cosine Activation
	Fully-connected Layer
	Fully-connected Layer
	Batch Normalization 1D Layer
	Cosine Activation
AdvSKM Embedder 2	Fully-connected Layer
	Fully-connected Layer
	Batch Normalization 1D Layer
	ReLU Activation
	Fully-connected Layer
	Fully-connected Layer
	Batch Normalization 1D Layer
	ReLU Activation
MMDA	
Class Output Head	Fully-connected Layer
CMD	
Class Output Head	Fully-connected Layer
CDAN	
Class Output Head	Fully-connected Layer
Domain Classifier Head	Fully-connected Layer
	ReLU Activation
	Fully-connected Layer
	ReLU Activation
	Fully-connected Layer
DIRT	
Class Output Head	Fully-connected Layer
Domain Classifier Head	Fully-connected Layer
	ReLU Activation
	Fully-connected Layer
	ReLU Activation
	Fully-connected Layer

$\lambda \in \{0, 0.0001, 0.001, 0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.5, 2, 5, 10\}$. In this way, we generate a sequence of 14 hyper-parameter choices. Due to computational limitations, in MiniDomainNet we use $\lambda \in \{0, 0.0001, 0.001, 0.01, 0.1, 1, 5, 10\}$. All values are listed in Table 9 and Table 10.

Table 9: Domain adaptation hyper-parameter sequences for experiments on the datasets Transformed Moons, AmazonReviews, and MiniDomainNet. We multiply each hyper-paramter with a set of scaling factors $\lambda \in \{0, 0.0001, 0.001, 0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.5, 2, 5, 10\}$ to obtain a sequence. Due to computational limitations, for MiniDomainNet we use only $\lambda \in \{0, 0.0001, 0.001, 0.01, 0.1, 1, 5, 10\}$.

Method	Hyper-parameter	Datasets		
		Transformed Moons	Amazon Reviews	MiniDomainNet
DANN	Classification loss weight	0.9603	0.9603	0.9603
	Domain loss weight	$\lambda \times 0.9238$	$\lambda \times 0.9238$	$\lambda \times 0.9238$
DeepCoral	Classification loss weight	0.05931	0.05931	0.05931
	Coral loss weight	$\lambda \times 8.452$	$\lambda \times 8.452$	$\lambda \times 8.452$
DDC	Classification loss weight	0.1593	0.1593	0.1593
	MMD loss weight	$\lambda \times 0.2048$	$\lambda \times 0.2048$	$\lambda \times 0.2048$
CMD	Classification loss weight	0.96	0.96	0.96
	CMD loss weight	$\lambda \times 5.52$	$\lambda \times 5.52$	$\lambda \times 5.52$
HoMM	Classification loss weight	0.2429	0.2429	0.2429
	Higher-order-MMD loss weight	$\lambda \times 0.9824$	$\lambda \times 0.9824$	$\lambda \times 0.9824$
CoDATS	Classification loss weight	0.5416	0.5416	0.5416
	Adversarial loss weight	$\lambda \times 0.5582$	$\lambda \times 0.5582$	$\lambda \times 0.5582$
DSAN	Classification loss weight	0.4133	0.4133	0.4133
	Local MMD loss weight	$\lambda \times 0.16$	$\lambda \times 0.16$	$\lambda \times 0.16$
AdvSKM	Classification loss weight	0.4637	0.4637	0.4637
	Adversarial MMD loss weight	$\lambda \times 0.1511$	$\lambda \times 0.1511$	$\lambda \times 0.1511$
MMDA	Classification loss weight	0.9505	0.9505	0.9505
	MMD loss weight	$\lambda \times 0.5476$	$\lambda \times 0.5476$	$\lambda \times 0.5476$
	Conditional loss weight	$\lambda \times 0.5167$	$\lambda \times 0.5167$	$\lambda \times 0.5167$
	Coral loss weight	$\lambda \times 0.5838$	$\lambda \times 0.5838$	$\lambda \times 0.5838$
CDAN	Classification loss weight	0.6636	0.6636	0.6636
	Adversarial loss weight	$\lambda \times 0.1954$	$\lambda \times 0.1954$	$\lambda \times 0.1954$
	Conditional loss weight	$\lambda \times 0.0124$	$\lambda \times 0.0124$	$\lambda \times 0.0124$
DIRT	Classification loss weight	0.9752	0.9752	0.9752
	Adversarial loss weight	$\lambda \times 0.3892$	$\lambda \times 0.3892$	$\lambda \times 0.3892$
	Conditional loss weight	$\lambda \times 0.09228$	$\lambda \times 0.09228$	$\lambda \times 0.09228$
	Virtual adversarial loss weight	$\lambda \times 0.1947$	$\lambda \times 0.1947$	$\lambda \times 0.1947$

Table 10: Domain adaptation hyper-parameters for experiments on the time-series data. We multiply each hyper-parameter with a set of scaling factors $\lambda \in \{0, 0.0001, 0.001, 0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.5, 2, 5, 10\}$ to obtain a sequence.

Method	Hyper-parameter	Datasets			
		UCI-HAR	Sleep-EDF	WISDM	HHAR
DANN	Classification loss weight	9.74	8.3	5.613	0.9603
	Domain loss weight	$\lambda \times 5.43$	$\lambda \times 0.324$	$\lambda \times 1.857$	$\lambda \times 0.9238$
DeepCoral	Classification loss weight	8.67	9.39	8.876	0.05931
	Coral loss weight	$\lambda \times 0.44$	$\lambda \times 0.19$	$\lambda \times 5.56$	$\lambda \times 8.452$
DDC	Classification loss weight	6.24	2.951	7.01	0.1593
	MMD loss weight	$\lambda \times 6.36$	$\lambda \times 8.923$	$\lambda \times 7.595$	$\lambda \times 0.2048$
CMD	Classification loss weight	0.96	0.96	0.96	0.96
	CMD loss weight	$\lambda \times 5.52$	$\lambda \times 5.52$	$\lambda \times 5.52$	$\lambda \times 5.52$
HoMM	Classification loss weight	2.15	0.197	0.1913	0.2429
	Higher-order-MMD loss weight	$\lambda \times 9.13$	$\lambda \times 1.102$	$\lambda \times 4.239$	$\lambda \times 0.9824$
CoDATS	Classification loss weight	6.21	9.239	7.187	0.5416
	Adversarial loss weight	$\lambda \times 1.72$	$\lambda \times 1.342$	$\lambda \times 6.439$	$\lambda \times 0.5582$
DSAN	Classification loss weight	1.76	6.713	0.1	0.4133
	Local MMD loss weight	$\lambda \times 1.59$	$\lambda \times 6.708$	$\lambda \times 0.1$	$\lambda \times 0.16$
AdvSKM	Classification loss weight	3.05	2.5	3.05	0.4637
	Adversarial MMD loss weight	$\lambda \times 2.876$	$\lambda \times 2.5$	$\lambda \times 2.876$	$\lambda \times 0.1511$
MMDA	Classification loss weight	6.13	4.48	0.1	0.9505
	MMD loss weight	$\lambda \times 2.37$	$\lambda \times 5.951$	$\lambda \times 0.1$	$\lambda \times 0.5476$
	Conditional loss weight	$\lambda \times 7.16$	$\lambda \times 6.13$	$\lambda \times 0.4753$	$\lambda \times 0.5167$
	Coral loss weight	$\lambda \times 8.63$	$\lambda \times 3.36$	$\lambda \times 0.1$	$\lambda \times 0.5838$
CDAN	Classification loss weight	5.19	6.803	9.54	0.6636
	Adversarial loss weight	$\lambda \times 2.91$	$\lambda \times 4.726$	$\lambda \times 3.283$	$\lambda \times 0.1954$
	Conditional loss weight	$\lambda \times 1.73$	$\lambda \times 1.307$	$\lambda \times 0.1$	$\lambda \times 0.0124$
DIRT	Classification loss weight	7.0	9.183	0.1	0.9752
	Adversarial loss weight	$\lambda \times 4.51$	$\lambda \times 7.411$	$\lambda \times 0.1$	$\lambda \times 0.3892$
	Conditional loss weight	$\lambda \times 0.79$	$\lambda \times 2.564$	$\lambda \times 0.1$	$\lambda \times 0.09228$
	Virtual adversarial loss weight	$\lambda \times 9.31$	$\lambda \times 3.583$	$\lambda \times 0.1$	$\lambda \times 0.1947$

D.4 MATRIX INVERSION

Matrix inversion is a well-known numerical task, especially in cases of limited computing precision and ill-conditioned matrices. In our case, similar models in the given sequence can cause numerical instability due to limited compute precision. That is, occasionally a tabula rasa inversion of the matrix \tilde{G} in Algorithm 1 is numerically unstable. Various standard approaches can be applied to handle this common issue, including the exclusion of similar models and various regularization techniques. In our computational setup, we rely on the Python routine `numpy.linalg.pinv`, which is based on the eigendecomposition of \tilde{G} (coinciding with the singular value decomposition in our case due to positive-definiteness) and an eigenvalue-based regularization based on a threshold value $rcond$ for small eigenvalues, see Strang (1980, pages 138–140) for details. The choice of $rcond$ depends on the scale of the Gram matrix and can therefore be chosen by source data only. Based on evaluating our method on source data only (target domain is fixed to be source domain) on several choices for $rcond$, we obtain a stable choice for $rcond$ of 10^{-1} for all datasets.

D.5 CORRELATION ANALYSIS: TARGET ACCURACY OF INDIVIDUAL MODELS VS. AGGREGATION WEIGHT

Figure 2 in the main paper suggests that there is a positive correlation between the target accuracy of each individual model (orange dashed line, Figure 2 top) and the respective aggregation weight (red bars, Figure 2 bottom), if the linear aggregation of the models is computed by our method IWA. In the following, we analyze whether this trend holds throughout all other experiments.

More precisely, for a given sequence f_1, \dots, f_l of models, we compute the Pearson correlation coefficient between the aggregation weights $\tilde{c}_1, \dots, \tilde{c}_l$ (see Algorithm 1) and the corresponding target accuracies $\frac{1}{t} \sum_{i=1}^t \mathbf{1}[y_i = f_1(x_i)], \dots, \frac{1}{t} \sum_{i=1}^t \mathbf{1}[y_i = f_l(x_i)]$ for the target test data $(x_1, y_1), \dots, (x_t, y_t)$, where $\mathbf{1}[P] = 1$ iff P is true and $\mathbf{1}[P] = 0$ otherwise. In this context, a positive correlation coefficient means that the aggregation algorithm assigns a higher weight to models performing better on target samples. We calculate these coefficients for our method IWA and the other linear regression baselines SOR, TCR, and TMR for all domain adaptation methods across all datasets and cross-domain scenarios. Note that for this analysis we cannot compare to the other baseline TMV, as the count-based aggregation by majority voting does not involve the computation of aggregation weights.

In Figure 4 and 6 we compare the resulting correlation coefficient distribution of our method IWA to the ones for the heuristic baselines SOR, TCR, and TMR. Figure 5 compares the correlation coefficients for IWA on different datasets.

We find that IWA shows a stronger positive correlation than other methods, between a model’s target accuracy and its aggregation weight.

D.6 SENSITIVITY ANALYSIS: EFFECT OF ADDING INACCURATE MODELS

We study the sensitivity of our method with respect to adding inaccurate models in the given sequence of models. We define an inaccurate model as having a target accuracy lower than 80% of the target accuracy of the model computed without domain adaptation (SO). In particular, we add +10, +50 and +100 inaccurate models to the given sequences of models. One inaccurate model is constructed as follows: First, a model is chosen uniformly at random from the given sequence of models. Second, the outputs of the chosen model are corrupted by adding, to half of the elements of its vector-valued output, random Gaussian noise with zero-mean and unit variance.

In Figure 7, we show the median performance over all domain adaptation methods for each dataset. We see, that the performance of our method (IWA) is not very sensitive w.r.t. an increase in the number of inaccurate models. This is in contrast to the heuristics (e.g. TMV, TMR or TCR) and model selection method DEV, which show a high sensitivity concerning adding inaccurate models. SOR also shows stable results; it is, however, clearly outperformed by our method in five out of six datasets. We conclude, that IWA is not only overall the best performing method, but also the most stable choice concerning inaccurate models in the given sequence.

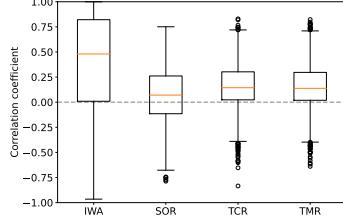


Figure 4: Boxplots of the correlation coefficients of IWA and the linear regression heuristic baselines SOR, TCR, and TMR over all datasets.

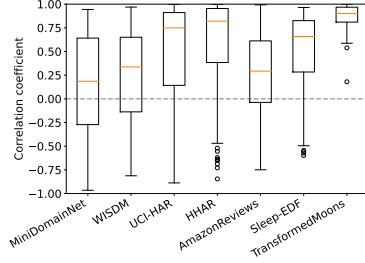


Figure 5: Boxplots of the correlation coefficients of IWA for each dataset.

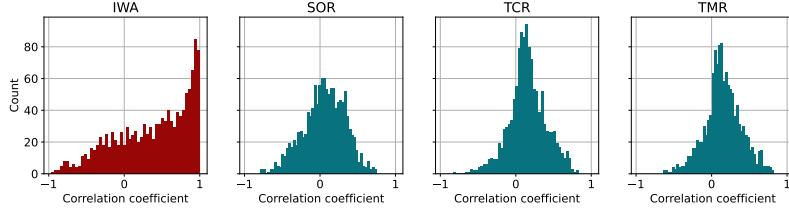


Figure 6: Histogram of the correlation coefficients of IWA and the linear regression heuristic baselines SOR, TCR and TMR over all datasets (showing the same data as Figure 4). In comparison to the heuristic baselines, IWA shows a stronger positive correlation between a model's target accuracy and its aggregation weight.

D.7 DETAILED EMPIRICAL RESULTS

In this section, we add all result tables for the datasets described in the main paper. Table 15, Table 16 and Table 17 show all domain adaptation tasks for the Amazon Review dataset. Table 13 shows all domain adaptation tasks for the MiniDomainNet experiments. Table 21, Table 22, Table 23, Table 24, Table 25, Table 26, Table 27, and Table 28 show all domain adaptation task results for the time-series datasets.

Baselines As addressed in the main paper, our method, IWA, is compared to ensemble learning methods that use linear regression and majority voting as *heuristic* for model aggregation, and, model selection methods with *theoretical error guarantees*. The heuristic baselines are majority voting on target data (TMV), source-only regression (SOR), target majority voting regression (TMR), target confidence average regression (TCR). The model selection methods with theoretical error guarantees are importance weighted validation (IWF) (Sugiyama et al., 2007) and deep embedded validation (DEV) (Kouw et al., 2019). The tables also provide a column for source-only (SO) performance and target-best (TB) performance. We highlight in bold the performance of the best performing method with theoretical error guarantees, and in italic the best performing heuristic.

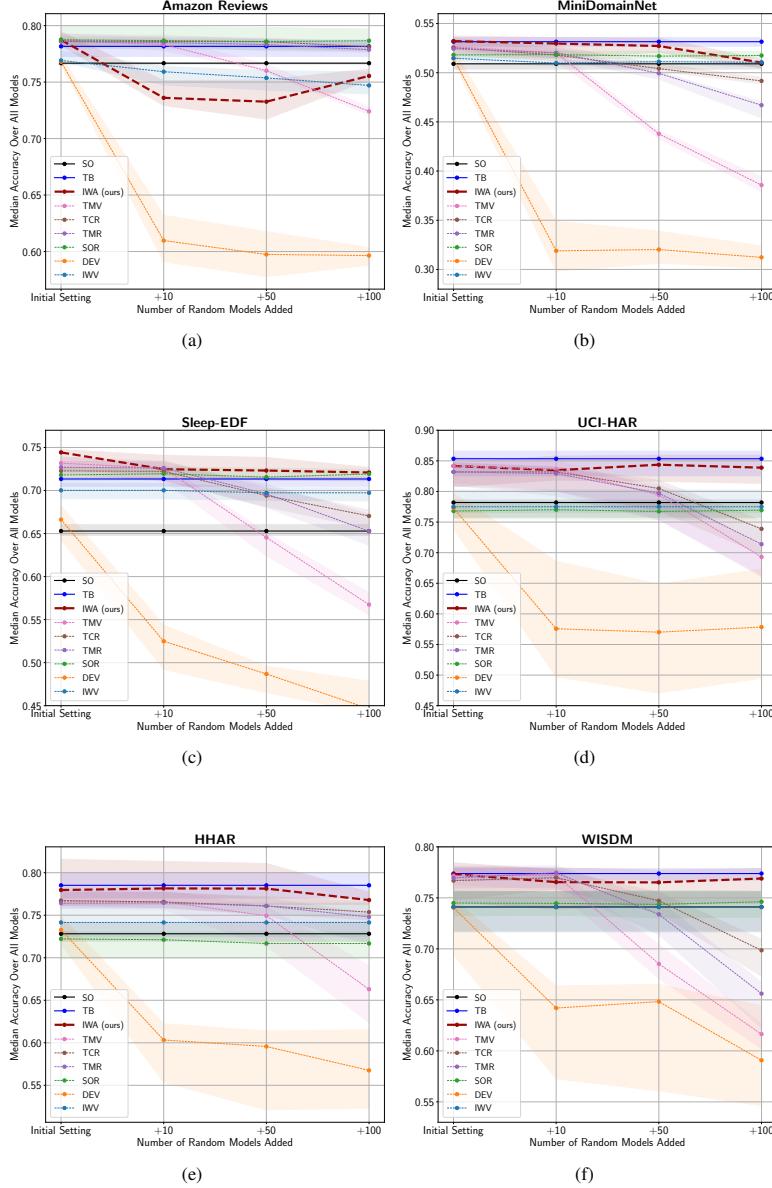


Figure 7: Sensitivity of methods for parameter choice issues w.r.t. adding inaccurate models in the given sequence of models; separate for each dataset (a-f), averaged over all domain adaptation methods, source-target pairs, and random seeds. Horizontal axes: Number of inaccurate models added to the initial sequence of models. Vertical axes: Target accuracy. Solid lines indicate median and shaded area indicate 50% confidence intervals.

D.7.1 DETAILED SUMMARY RESULTS

Table 11: Average target accuracies (and average standard deviations) for all 7 datasets (e.g., Sleep-EDF, MiniDomainNet, Amazon Reviews) taken over several domain adaptation tasks (e.g., 5 on Sleep-EDF, 5 on MiniDomainNet, 12 on Amazon Reviews), 11 domain adaptation methods (e.g., DANN, HoMM, CMD) and 3 repetitions with different random initialization of model weights. The input sequences of the approaches (e.g., DEV, IWA) consist of neural networks computed by runs of the domain adaptation methods with different hyper-parameters (e.g., 8 different values of λ for DANN).

Dataset	Heuristic					Theoretical error guarantees				TB
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)		
Transformed Moons	0.989(±0.008)	0.980(±0.006)	0.981(±0.007)	0.997(±0.002)	0.989(±0.010)	0.989(±0.008)	0.981(±0.022)	0.997(±0.002)	0.997(±0.005)	
Amazon Reviews	0.767(±0.011)	0.787(±0.009)	0.786(±0.010)	0.786(±0.010)	0.789(±0.010)	0.772(±0.014)	0.764(±0.019)	0.788(±0.009)	0.781(±0.012)	
MiniDomainNet	0.507(±0.022)	0.526(±0.011)	0.525(±0.014)	0.526(±0.013)	0.519(±0.012)	0.513(±0.022)	0.515(±0.028)	0.531(±0.011)	0.534(±0.022)	
Sleep-EDF	0.655(±0.054)	0.729(±0.018)	0.729(±0.024)	0.725(±0.023)	0.717(±0.028)	0.700(±0.052)	0.660(±0.057)	0.737(±0.020)	0.712(±0.045)	
UCI-HAR	0.770(±0.046)	0.840(±0.017)	0.833(±0.023)	0.832(±0.024)	0.769(±0.060)	0.774(±0.070)	0.765(±0.090)	0.835(±0.020)	0.850(±0.029)	
HHR	0.732(±0.042)	0.771(±0.015)	0.768(±0.017)	0.771(±0.018)	0.722(±0.068)	0.746(±0.037)	0.722(±0.063)	0.787(±0.012)	0.784(±0.028)	
WISDM	0.736(±0.050)	0.768(±0.027)	0.768(±0.036)	0.765(±0.037)	0.737(±0.062)	0.736(±0.052)	0.726(±0.077)	0.764(±0.025)	0.771(±0.046)	

Table 12: Mean and standard deviation (after \pm) of target classification accuracy on Amazon Reviews dataset over three different random initialization of model weights and 12 domain adaptation tasks.

Method	Amazon Reviews					Theoretical error guarantees				TB
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)		
HoMM	0.760(±0.009)	0.777(±0.010)	0.778(±0.010)	0.778(±0.011)	0.777(±0.010)	0.765(±0.011)	0.766(±0.011)	0.778(±0.010)	0.769(±0.012)	
AdvHoMM	0.766(±0.012)	0.789(±0.009)	0.789(±0.010)	0.778(±0.008)	0.788(±0.011)	0.778(±0.022)	0.777(±0.056)	0.787(±0.008)	0.770(±0.009)	
DIRT	0.764(±0.009)	0.786(±0.008)	0.786(±0.009)	0.786(±0.008)	0.789(±0.008)	0.778(±0.024)	0.777(±0.048)	0.787(±0.008)	0.777(±0.013)	
DDC	0.766(±0.012)	0.779(±0.010)	0.780(±0.009)	0.779(±0.010)	0.778(±0.010)	0.767(±0.010)	0.768(±0.011)	0.780(±0.010)	0.770(±0.013)	
CMD	0.767(±0.012)	0.791(±0.009)	0.792(±0.009)	0.789(±0.010)	0.792(±0.010)	0.765(±0.015)	0.710(±0.015)	0.794(±0.009)	0.785(±0.009)	
MMDA	0.767(±0.011)	0.787(±0.011)	0.785(±0.010)	0.785(±0.010)	0.787(±0.012)	0.769(±0.011)	0.766(±0.010)	0.787(±0.011)	0.782(±0.011)	
CoDATS	0.766(±0.013)	0.795(±0.009)	0.793(±0.010)	0.794(±0.012)	0.799(±0.010)	0.779(±0.016)	0.773(±0.020)	0.796(±0.009)	0.791(±0.015)	
Deep-Coral	0.767(±0.012)	0.784(±0.010)	0.784(±0.010)	0.783(±0.010)	0.783(±0.010)	0.769(±0.011)	0.769(±0.011)	0.785(±0.013)	0.776(±0.013)	
CDAN	0.767(±0.012)	0.796(±0.010)	0.787(±0.009)	0.787(±0.010)	0.787(±0.010)	0.763(±0.011)	0.763(±0.011)	0.788(±0.010)	0.776(±0.013)	
DANN	0.767(±0.012)	0.796(±0.010)	0.792(±0.010)	0.793(±0.010)	0.793(±0.010)	0.800(±0.011)	0.776(±0.011)	0.797(±0.012)	0.798(±0.012)	
DSAN	0.769(±0.009)	0.796(±0.009)	0.792(±0.009)	0.791(±0.010)	0.800(±0.010)	0.779(±0.012)	0.763(±0.017)	0.795(±0.009)	0.789(±0.012)	
Avg.	0.767(±0.011)	0.787(±0.009)	0.786(±0.010)	0.786(±0.010)	0.789(±0.010)	0.772(±0.014)	0.764(±0.019)	0.788(±0.009)	0.781(±0.012)	

Table 13: Mean and standard deviation (after \pm) of target classification accuracy on MiniDomainNet dataset over three different random initialization of model weights and five domain adaptation tasks.

Method	MiniDomainNet					Theoretical error guarantees				TB
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)		
HoMM	0.509(±0.018)	0.526(±0.009)	0.524(±0.010)	0.520(±0.010)	0.514(±0.015)	0.521(±0.018)	0.511(±0.018)	0.531(±0.007)	0.523(±0.020)	
AdvSKM	0.509(±0.015)	0.516(±0.011)	0.514(±0.013)	0.514(±0.017)	0.514(±0.019)	0.515(±0.022)	0.512(±0.022)	0.524(±0.017)	0.522(±0.017)	
DIRT	0.499(±0.022)	0.514(±0.006)	0.515(±0.009)	0.508(±0.006)	0.507(±0.018)	0.493(±0.028)	0.498(±0.033)	0.519(±0.006)	0.525(±0.020)	
DDC	0.510(±0.032)	0.514(±0.012)	0.512(±0.009)	0.511(±0.019)	0.514(±0.015)	0.511(±0.028)	0.512(±0.043)	0.516(±0.015)	0.521(±0.019)	
CMD	0.509(±0.019)	0.528(±0.011)	0.531(±0.016)	0.531(±0.016)	0.522(±0.013)	0.518(±0.025)	0.490(±0.037)	0.533(±0.009)	0.533(±0.024)	
MMDA	0.509(±0.021)	0.524(±0.004)	0.522(±0.005)	0.521(±0.007)	0.519(±0.012)	0.517(±0.008)	0.524(±0.006)	0.527(±0.009)	0.531(±0.029)	
CoDATS	0.502(±0.026)	0.535(±0.026)	0.533(±0.027)	0.536(±0.028)	0.525(±0.015)	0.518(±0.031)	0.524(±0.051)	0.536(±0.025)	0.529(±0.042)	
Deep-Coral	0.507(±0.022)	0.528(±0.009)	0.534(±0.011)	0.534(±0.012)	0.529(±0.011)	0.514(±0.028)	0.524(±0.034)	0.533(±0.012)	0.533(±0.020)	
CDAN	0.514(±0.009)	0.530(±0.013)	0.532(±0.011)	0.529(±0.009)	0.530(±0.005)	0.531(±0.011)	0.529(±0.011)	0.533(±0.010)	0.534(±0.017)	
DANN	0.496(±0.022)	0.530(±0.011)	0.532(±0.011)	0.530(±0.005)	0.536(±0.011)	0.519(±0.015)	0.515(±0.013)	0.541(±0.006)	0.532(±0.014)	
DSAN	0.509(±0.022)	0.537(±0.011)	0.534(±0.013)	0.547(±0.012)	0.525(±0.013)	0.513(±0.018)	0.535(±0.023)	0.546(±0.007)	0.563(±0.013)	
Avg.	0.507(±0.022)	0.526(±0.011)	0.525(±0.014)	0.526(±0.013)	0.518(±0.012)	0.513(±0.022)	0.515(±0.028)	0.531(±0.011)	0.534(±0.022)	

Table 14: Mean and standard deviation (after \pm) of target classification accuracy on four time series datasets over three different random initialization of model weights and five domain adaptation tasks.

Sleep-EDF									
Method	Heuristic					Theoretical error guarantees			
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
HoMM	0.676(± 0.036)	0.722(± 0.017)	0.719(± 0.023)	0.718(± 0.021)	0.724(± 0.023)	0.726(± 0.046)	0.678(± 0.035)	0.724(± 0.025)	0.715(± 0.047)
AdvSKM	0.665(± 0.058)	0.708(± 0.023)	0.712(± 0.027)	0.712(± 0.032)	0.718(± 0.030)	0.703(± 0.069)	0.692(± 0.038)	0.722(± 0.025)	0.706(± 0.054)
DIRT	0.656(± 0.058)	0.743(± 0.009)	0.745(± 0.012)	0.748(± 0.019)	0.742(± 0.031)	0.679(± 0.038)	0.686(± 0.061)	0.749(± 0.010)	0.728(± 0.037)
DDC	0.646(± 0.035)	0.717(± 0.029)	0.721(± 0.037)	0.712(± 0.031)	0.695(± 0.020)	0.694(± 0.056)	0.666(± 0.031)	0.724(± 0.012)	0.704(± 0.031)
CMD	0.646(± 0.057)	0.710(± 0.022)	0.730(± 0.022)	0.728(± 0.021)	0.709(± 0.015)	0.716(± 0.052)	0.725(± 0.061)	0.724(± 0.012)	0.725(± 0.053)
MMDA	0.650(± 0.035)	0.720(± 0.014)	0.724(± 0.021)	0.728(± 0.018)	0.720(± 0.008)	0.718(± 0.054)	0.690(± 0.031)	0.724(± 0.012)	0.715(± 0.042)
CoDATS	0.672(± 0.084)	0.738(± 0.036)	0.739(± 0.030)	0.721(± 0.029)	0.724(± 0.029)	0.682(± 0.090)	0.690(± 0.107)	0.744(± 0.021)	0.715(± 0.045)
Deep-Coral	0.643(± 0.049)	0.716(± 0.018)	0.717(± 0.028)	0.712(± 0.027)	0.694(± 0.032)	0.700(± 0.053)	0.675(± 0.077)	0.713(± 0.021)	0.702(± 0.070)
CDAN	0.652(± 0.056)	0.732(± 0.016)	0.739(± 0.024)	0.739(± 0.018)	0.728(± 0.029)	0.697(± 0.031)	0.642(± 0.065)	0.748(± 0.019)	0.713(± 0.045)
DANN	0.641(± 0.047)	0.722(± 0.017)	0.723(± 0.026)	0.721(± 0.025)	0.714(± 0.024)	0.687(± 0.034)	0.644(± 0.046)	0.724(± 0.018)	0.710(± 0.035)
DSAN	0.653(± 0.060)	0.748(± 0.008)	0.740(± 0.016)	0.732(± 0.016)	0.728(± 0.026)	0.712(± 0.070)	0.589(± 0.063)	0.757(± 0.016)	0.700(± 0.033)
Avg.	0.655(± 0.054)	0.729(± 0.018)	0.729(± 0.024)	0.725(± 0.023)	0.717(± 0.028)	0.700(± 0.052)	0.660(± 0.057)	0.737(± 0.020)	0.712(± 0.045)

UCI-HAR									
Method	Heuristic					Theoretical error guarantees			
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
HoMM	0.782(± 0.078)	0.833(± 0.020)	0.818(± 0.023)	0.818(± 0.022)	0.783(± 0.040)	0.809(± 0.058)	0.800(± 0.098)	0.826(± 0.010)	0.854(± 0.039)
AdvSKM	0.724(± 0.059)	0.791(± 0.021)	0.806(± 0.022)	0.804(± 0.022)	0.768(± 0.042)	0.750(± 0.042)	0.711(± 0.167)	0.826(± 0.022)	0.811(± 0.062)
DIRT	0.728(± 0.029)	0.790(± 0.010)	0.802(± 0.010)	0.800(± 0.010)	0.789(± 0.037)	0.790(± 0.077)	0.743(± 0.107)	0.800(± 0.010)	0.792(± 0.034)
DDC	0.700(± 0.061)	0.806(± 0.019)	0.807(± 0.026)	0.807(± 0.019)	0.750(± 0.108)	0.724(± 0.060)	0.734(± 0.109)	0.804(± 0.026)	0.820(± 0.015)
CMD	0.788(± 0.058)	0.869(± 0.019)	0.849(± 0.014)	0.839(± 0.023)	0.731(± 0.066)	0.804(± 0.064)	0.812(± 0.080)	0.842(± 0.025)	0.888(± 0.037)
MMDA	0.785(± 0.018)	0.819(± 0.022)	0.812(± 0.028)	0.800(± 0.032)	0.759(± 0.085)	0.733(± 0.073)	0.767(± 0.107)	0.807(± 0.025)	0.840(± 0.055)
CoDATS	0.760(± 0.037)	0.854(± 0.022)	0.832(± 0.027)	0.832(± 0.026)	0.785(± 0.057)	0.801(± 0.079)	0.794(± 0.078)	0.846(± 0.016)	0.867(± 0.012)
Deep-Coral	0.790(± 0.045)	0.810(± 0.007)	0.800(± 0.022)	0.800(± 0.020)	0.771(± 0.030)	0.768(± 0.047)	0.773(± 0.087)	0.808(± 0.016)	0.806(± 0.022)
CDAN	0.750(± 0.055)	0.842(± 0.009)	0.844(± 0.020)	0.840(± 0.034)	0.802(± 0.065)	0.786(± 0.072)	0.777(± 0.117)	0.816(± 0.018)	0.853(± 0.026)
DANN	0.756(± 0.051)	0.850(± 0.019)	0.849(± 0.030)	0.849(± 0.057)	0.836(± 0.082)	0.780(± 0.043)	0.840(± 0.077)	0.840(± 0.023)	0.865(± 0.038)
DSAN	0.762(± 0.032)	0.849(± 0.023)	0.843(± 0.033)	0.854(± 0.025)	0.749(± 0.065)	0.775(± 0.043)	0.744(± 0.035)	0.858(± 0.023)	0.865(± 0.028)
Avg.	0.770(± 0.046)	0.840(± 0.017)	0.833(± 0.023)	0.832(± 0.024)	0.769(± 0.060)	0.774(± 0.070)	0.765(± 0.090)	0.835(± 0.020)	0.850(± 0.029)

HHAR									
Method	Heuristic					Theoretical error guarantees			
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
HoMM	0.739(± 0.044)	0.757(± 0.014)	0.759(± 0.013)	0.750(± 0.011)	0.700(± 0.058)	0.720(± 0.027)	0.733(± 0.031)	0.759(± 0.007)	0.764(± 0.023)
AdvSKM	0.718(± 0.042)	0.749(± 0.027)	0.742(± 0.022)	0.748(± 0.034)	0.676(± 0.046)	0.730(± 0.051)	0.728(± 0.051)	0.752(± 0.031)	0.749(± 0.025)
DIRT	0.728(± 0.029)	0.743(± 0.011)	0.792(± 0.018)	0.803(± 0.017)	0.796(± 0.066)	0.743(± 0.028)	0.739(± 0.075)	0.816(± 0.008)	0.820(± 0.015)
DDC	0.716(± 0.063)	0.748(± 0.014)	0.750(± 0.009)	0.748(± 0.007)	0.717(± 0.075)	0.711(± 0.048)	0.705(± 0.066)	0.748(± 0.012)	0.729(± 0.027)
CMD	0.748(± 0.027)	0.760(± 0.014)	0.764(± 0.009)	0.764(± 0.007)	0.737(± 0.100)	0.775(± 0.031)	0.643(± 0.031)	0.766(± 0.016)	0.794(± 0.023)
MMDA	0.750(± 0.035)	0.783(± 0.017)	0.781(± 0.016)	0.780(± 0.015)	0.698(± 0.064)	0.719(± 0.036)	0.731(± 0.047)	0.785(± 0.017)	0.785(± 0.035)
CoDATS	0.710(± 0.039)	0.766(± 0.021)	0.762(± 0.015)	0.766(± 0.017)	0.681(± 0.073)	0.754(± 0.054)	0.758(± 0.244)	0.764(± 0.006)	0.776(± 0.023)
Deep-Coral	0.745(± 0.046)	0.766(± 0.012)	0.762(± 0.015)	0.766(± 0.017)	0.681(± 0.073)	0.754(± 0.054)	0.758(± 0.244)	0.764(± 0.006)	0.776(± 0.023)
CDAN	0.728(± 0.039)	0.762(± 0.012)	0.758(± 0.017)	0.764(± 0.016)	0.765(± 0.063)	0.774(± 0.035)	0.775(± 0.036)	0.816(± 0.011)	0.790(± 0.038)
DANN	0.757(± 0.057)	0.779(± 0.012)	0.774(± 0.009)	0.773(± 0.011)	0.722(± 0.103)	0.798(± 0.041)	0.793(± 0.045)	0.818(± 0.009)	0.807(± 0.020)
DSAN	0.721(± 0.053)	0.803(± 0.010)	0.797(± 0.014)	0.802(± 0.007)	0.724(± 0.065)	0.741(± 0.033)	0.596(± 0.031)	0.825(± 0.008)	0.826(± 0.046)
Avg.	0.732(± 0.042)	0.771(± 0.015)	0.768(± 0.017)	0.771(± 0.018)	0.722(± 0.068)	0.746(± 0.037)	0.722(± 0.063)	0.787(± 0.012)	0.784(± 0.028)

WISDM									
Method	Heuristic					Theoretical error guarantees			
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB
HoMM	0.753(± 0.054)	0.741(± 0.026)	0.738(± 0.031)	0.739(± 0.047)	0.775(± 0.062)	0.753(± 0.054)	0.740(± 0.054)	0.728(± 0.021)	0.774(± 0.037)
AdvSKM	0.747(± 0.050)	0.771(± 0.043)	0.781(± 0.055)	0.779(± 0.035)	0.742(± 0.062)	0.747(± 0.050)	0.747(± 0.135)	0.777(± 0.031)	0.779(± 0.041)
DIRT	0.738(± 0.038)	0.792(± 0.015)	0.797(± 0.019)	0.797(± 0.037)	0.756(± 0.071)	0.738(± 0.059)	0.797(± 0.059)	0.798	

D.7.2 DETAILED AMAZON REVIEWS RESULTS

Table 15: Mean and standard deviation (after \pm) of target classification accuracy on Amazon Reviews (Part 1) over 3 repetitions with different random initialization of model weights.

Task	HoMM				Theoretical error guarantees				TB
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	
B \rightarrow D	0.778(± 0.01)	0.793(± 0.008)	0.793(± 0.008)	0.795(± 0.002)	0.795(± 0.008)	0.782(± 0.021)	0.782(± 0.021)	0.794(± 0.007)	0.789(± 0.008)
B \rightarrow E	0.745(± 0.014)	0.759(± 0.013)	0.759(± 0.013)	0.761(± 0.012)	0.763(± 0.012)	0.747(± 0.005)	0.749(± 0.025)	0.761(± 0.012)	0.758(± 0.012)
B \rightarrow K	0.777(± 0.002)	0.792(± 0.018)	0.791(± 0.018)	0.782(± 0.020)	0.783(± 0.013)	0.788(± 0.003)	0.778(± 0.002)	0.777(± 0.015)	0.776(± 0.003)
D \rightarrow E	0.770(± 0.006)	0.791(± 0.006)	0.791(± 0.006)	0.794(± 0.002)	0.788(± 0.003)	0.778(± 0.002)	0.777(± 0.015)	0.791(± 0.003)	0.784(± 0.003)
D \rightarrow E	0.774(± 0.012)	0.791(± 0.003)	0.781(± 0.003)	0.779(± 0.004)	0.778(± 0.004)	0.764(± 0.017)	0.769(± 0.009)	0.781(± 0.004)	0.774(± 0.003)
D \rightarrow K	0.778(± 0.015)	0.791(± 0.016)	0.791(± 0.016)	0.791(± 0.017)	0.791(± 0.017)	0.781(± 0.011)	0.782(± 0.011)	0.789(± 0.016)	0.786(± 0.017)
D \rightarrow K	0.730(± 0.003)	0.731(± 0.016)	0.731(± 0.016)	0.732(± 0.016)	0.732(± 0.016)	0.731(± 0.017)	0.731(± 0.017)	0.719(± 0.005)	0.735(± 0.011)
E \rightarrow B	0.702(± 0.026)	0.7(± 0.02)	0.7(± 0.02)	0.701(± 0.019)	0.699(± 0.021)	0.690(± 0.011)	0.692(± 0.014)	0.698(± 0.011)	0.703(± 0.03)
E \rightarrow D	0.733(± 0.003)	0.733(± 0.006)	0.728(± 0.011)	0.728(± 0.011)	0.729(± 0.003)	0.729(± 0.003)			
E \rightarrow K	0.862(± 0.009)	0.874(± 0.008)	0.874(± 0.008)	0.876(± 0.009)	0.873(± 0.008)	0.856(± 0.011)	0.858(± 0.012)	0.873(± 0.009)	0.863(± 0.009)
K \rightarrow B	0.715(± 0.008)	0.721(± 0.004)	0.714(± 0.016)	0.708(± 0.003)	0.722(± 0.007)	0.717(± 0.007)			
K \rightarrow D	0.745(± 0.011)	0.752(± 0.01)	0.752(± 0.01)	0.747(± 0.013)	0.755(± 0.008)	0.735(± 0.018)	0.742(± 0.022)	0.749(± 0.006)	0.745(± 0.008)
K \rightarrow E	0.844(± 0.009)	0.854(± 0.008)	0.854(± 0.008)	0.855(± 0.005)	0.857(± 0.009)	0.845(± 0.005)	0.844(± 0.006)	0.856(± 0.006)	0.849(± 0.009)
Avg.	0.769(± 0.012)	0.778(± 0.01)	0.778(± 0.01)	0.778(± 0.01)	0.777(± 0.011)	0.765(± 0.011)	0.766(± 0.012)	0.778(± 0.009)	0.774(± 0.011)

Task	AdvSKM				Theoretical error guarantees				TB
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	
B \rightarrow D	0.784(± 0.004)	0.79(± 0.003)	0.79(± 0.003)	0.79(± 0.003)	0.79(± 0.003)	0.777(± 0.008)	0.778(± 0.008)	0.794(± 0.005)	0.79(± 0.005)
B \rightarrow E	0.754(± 0.009)	0.761(± 0.014)	0.761(± 0.014)	0.763(± 0.012)	0.762(± 0.011)	0.754(± 0.003)	0.751(± 0.007)	0.761(± 0.012)	0.756(± 0.012)
B \rightarrow K	0.769(± 0.021)	0.784(± 0.019)	0.784(± 0.019)	0.782(± 0.021)	0.78(± 0.016)	0.771(± 0.012)	0.764(± 0.016)	0.782(± 0.017)	0.774(± 0.019)
D \rightarrow B	0.779(± 0.004)	0.795(± 0.007)	0.795(± 0.007)	0.794(± 0.007)	0.792(± 0.003)	0.785(± 0.007)	0.78(± 0.011)	0.796(± 0.009)	0.789(± 0.009)
D \rightarrow E	0.766(± 0.013)	0.785(± 0.006)	0.785(± 0.006)	0.784(± 0.008)	0.777(± 0.005)	0.773(± 0.005)	0.769(± 0.009)	0.785(± 0.005)	0.776(± 0.002)
D \rightarrow K	0.777(± 0.015)	0.791(± 0.019)	0.791(± 0.019)	0.791(± 0.015)	0.791(± 0.015)	0.791(± 0.013)	0.791(± 0.013)	0.791(± 0.013)	0.789(± 0.013)
D \rightarrow K	0.777(± 0.015)	0.791(± 0.022)	0.791(± 0.022)	0.792(± 0.019)	0.791(± 0.019)	0.791(± 0.022)	0.795(± 0.022)	0.792(± 0.022)	0.791(± 0.021)
E \rightarrow D	0.714(± 0.005)	0.724(± 0.002)	0.734(± 0.002)	0.737(± 0.003)	0.733(± 0.008)	0.725(± 0.004)	0.725(± 0.004)	0.732(± 0.006)	0.737(± 0.011)
E \rightarrow K	0.858(± 0.004)	0.874(± 0.01)	0.874(± 0.01)	0.874(± 0.012)	0.876(± 0.011)	0.866(± 0.012)	0.866(± 0.012)	0.875(± 0.013)	0.865(± 0.01)
K \rightarrow B	0.713(± 0.002)	0.723(± 0.006)	0.723(± 0.006)	0.724(± 0.009)	0.722(± 0.004)	0.708(± 0.014)	0.705(± 0.011)	0.726(± 0.006)	0.718(± 0.006)
K \rightarrow D	0.741(± 0.01)	0.752(± 0.012)	0.752(± 0.012)	0.751(± 0.013)	0.752(± 0.017)	0.75(± 0.019)	0.742(± 0.008)	0.749(± 0.012)	0.756(± 0.014)
K \rightarrow E	0.848(± 0.008)	0.857(± 0.007)	0.857(± 0.007)	0.858(± 0.008)	0.856(± 0.008)	0.846(± 0.012)	0.847(± 0.007)	0.858(± 0.008)	0.85(± 0.008)
Avg.	0.766(± 0.009)	0.779(± 0.01)	0.779(± 0.01)	0.779(± 0.011)	0.778(± 0.01)	0.769(± 0.011)	0.766(± 0.011)	0.78(± 0.01)	0.775(± 0.011)

Task	DIRT				Theoretical error guarantees				TB
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	
B \rightarrow D	0.777(± 0.01)	0.816(± 0.008)	0.816(± 0.008)	0.817(± 0.008)	0.813(± 0.007)	0.791(± 0.012)	0.787(± 0.032)	0.813(± 0.004)	0.809(± 0.008)
B \rightarrow E	0.756(± 0.013)	0.792(± 0.006)	0.792(± 0.006)	0.792(± 0.01)	0.782(± 0.008)	0.758(± 0.011)	0.758(± 0.011)	0.788(± 0.008)	0.804(± 0.005)
B \rightarrow K	0.777(± 0.013)	0.815(± 0.003)	0.815(± 0.003)	0.814(± 0.005)	0.813(± 0.006)	0.793(± 0.009)	0.782(± 0.025)	0.814(± 0.003)	0.814(± 0.003)
D \rightarrow B	0.773(± 0.008)	0.812(± 0.014)	0.812(± 0.014)	0.809(± 0.019)	0.814(± 0.01)	0.805(± 0.015)	0.808(± 0.004)	0.811(± 0.01)	0.811(± 0.01)
D \rightarrow E	0.757(± 0.012)	0.813(± 0.009)	0.813(± 0.009)	0.814(± 0.009)	0.808(± 0.012)	0.787(± 0.024)	0.807(± 0.024)	0.814(± 0.009)	0.812(± 0.009)
D \rightarrow K	0.775(± 0.016)	0.836(± 0.009)	0.836(± 0.009)	0.835(± 0.008)	0.837(± 0.004)	0.827(± 0.004)	0.827(± 0.038)	0.832(± 0.006)	0.828(± 0.004)
E \rightarrow B	0.702(± 0.018)	0.762(± 0.015)	0.764(± 0.015)	0.764(± 0.015)	0.764(± 0.018)	0.714(± 0.018)	0.702(± 0.018)	0.675(± 0.05)	0.678(± 0.015)
E \rightarrow D	0.716(± 0.008)	0.688(± 0.019)	0.689(± 0.019)	0.69(± 0.022)	0.731(± 0.029)	0.718(± 0.01)	0.704(± 0.027)	0.702(± 0.021)	0.73(± 0.026)
E \rightarrow K	0.709(± 0.012)	0.707(± 0.007)	0.707(± 0.007)	0.707(± 0.003)	0.719(± 0.005)	0.719(± 0.005)	0.721(± 0.002)	0.715(± 0.005)	0.711(± 0.008)
K \rightarrow D	0.736(± 0.012)	0.721(± 0.01)	0.721(± 0.01)	0.727(± 0.019)	0.729(± 0.022)	0.745(± 0.022)	0.731(± 0.007)	0.727(± 0.006)	0.756(± 0.003)
K \rightarrow E	0.842(± 0.009)	0.869(± 0.01)	0.869(± 0.01)	0.868(± 0.01)	0.872(± 0.009)	0.848(± 0.012)	0.848(± 0.008)	0.868(± 0.01)	0.863(± 0.009)
Avg.	0.764(± 0.013)	0.786(± 0.01)	0.786(± 0.012)	0.786(± 0.012)	0.8(± 0.01)	0.778(± 0.016)	0.773(± 0.02)	0.787(± 0.009)	0.795(± 0.009)

Task	DDC				Theoretical error guarantees				TB
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	
B \rightarrow D	0.781(± 0.016)	0.796(± 0.008)	0.796(± 0.008)	0.793(± 0.013)	0.794(± 0.002)	0.775(± 0.017)	0.778(± 0.02)	0.789	

Table 16: Mean and standard deviation (after \pm) of target classification accuracy on Amazon Reviews (Part 2) over 3 repetitions with different random initialization of model weights.

CMD										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
B \rightarrow D	0.772(±0.009)	0.794(±0.008)	0.794(±0.008)	0.794(±0.004)	0.798(±0.003)	0.779(±0.018)	0.748(±0.036)	0.8(±0.003)	0.789(±0.003)	
B \rightarrow E	0.745(±0.012)	0.785(±0.014)	0.785(±0.014)	0.779(±0.011)	0.779(±0.011)	0.746(±0.032)	0.651(±0.077)	0.782(±0.011)	0.78(±0.014)	
B \rightarrow K	0.763(±0.015)	0.796(±0.011)	0.796(±0.011)	0.79(±0.007)	0.794(±0.013)	0.787(±0.011)	0.749(±0.028)	0.797(±0.007)	0.793(±0.008)	
D \rightarrow B	0.788(±0.01)	0.801(±0.007)	0.801(±0.007)	0.798(±0.003)	0.804(±0.002)	0.785(±0.011)	0.754(±0.022)	0.803(±0.008)	0.794(±0.002)	
D \rightarrow E	0.764(±0.009)	0.804(±0.006)	0.804(±0.006)	0.798(±0.003)	0.804(±0.003)	0.784(±0.018)	0.754(±0.035)	0.804(±0.005)	0.795(±0.007)	
D \rightarrow K	0.777(±0.012)	0.808(±0.009)	0.803(±0.009)	0.799(±0.014)	0.804(±0.007)	0.784(±0.009)	0.767(±0.036)	0.801(±0.011)	0.811(±0.007)	
E \rightarrow B	0.609(±0.014)	0.721(±0.013)	0.721(±0.013)	0.717(±0.014)	0.724(±0.013)	0.71(±0.016)	0.68(±0.039)	0.719(±0.013)	0.712(±0.013)	
E \rightarrow D	0.722(±0.009)	0.752(±0.006)	0.752(±0.006)	0.752(±0.004)	0.752(±0.004)	0.751(±0.005)	0.693(±0.041)	0.753(±0.01)	0.755(±0.003)	0.738(±0.003)
E \rightarrow K	0.86(±0.012)	0.874(±0.013)	0.874(±0.013)	0.872(±0.007)	0.875(±0.008)	0.822(±0.058)	0.807(±0.054)	0.876(±0.009)	0.871(±0.009)	
K \rightarrow B	0.718(±0.005)	0.749(±0.007)	0.749(±0.007)	0.738(±0.003)	0.74(±0.003)	0.716(±0.022)	0.679(±0.056)	0.746(±0.004)	0.74(±0.007)	
K \rightarrow D	0.748(±0.003)	0.761(±0.016)	0.761(±0.016)	0.766(±0.022)	0.77(±0.017)	0.746(±0.011)	0.655(±0.119)	0.775(±0.016)	0.761(±0.016)	
K \rightarrow E	0.842(±0.005)	0.864(±0.006)	0.864(±0.006)	0.862(±0.002)	0.869(±0.004)	0.85(±0.01)	0.806(±0.041)	0.867(±0.008)	0.858(±0.004)	
Avg.	0.767(±0.009)	0.792(±0.01)	0.792(±0.01)	0.789(±0.008)	0.792(±0.008)	0.765(±0.022)	0.71(±0.056)	0.794(±0.008)	0.787(±0.008)	

MMDA										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
B \rightarrow D	0.775(±0.005)	0.796(±0.005)	0.796(±0.005)	0.797(±0.009)	0.794(±0.015)	0.783(±0.004)	0.786(±0.006)	0.797(±0.003)	0.792(±0.004)	
B \rightarrow E	0.752(±0.006)	0.775(±0.01)	0.775(±0.01)	0.776(±0.008)	0.771(±0.014)	0.766(±0.022)	0.756(±0.012)	0.779(±0.012)	0.781(±0.006)	
B \rightarrow K	0.765(±0.009)	0.793(±0.016)	0.793(±0.016)	0.792(±0.017)	0.803(±0.019)	0.763(±0.039)	0.767(±0.024)	0.795(±0.017)	0.802(±0.019)	
D \rightarrow B	0.783(±0.007)	0.796(±0.007)	0.796(±0.007)	0.795(±0.008)	0.796(±0.006)	0.789(±0.013)	0.789(±0.012)	0.798(±0.009)	0.79(±0.009)	
D \rightarrow E	0.764(±0.003)	0.792(±0.005)	0.792(±0.005)	0.792(±0.005)	0.792(±0.005)	0.773(±0.009)	0.773(±0.006)	0.795(±0.006)	0.799(±0.006)	
D \rightarrow K	0.775(±0.012)	0.797(±0.014)	0.797(±0.014)	0.796(±0.013)	0.811(±0.008)	0.791(±0.018)	0.79(±0.019)	0.8(±0.012)	0.8(±0.008)	
E \rightarrow B	0.701(±0.015)	0.707(±0.017)	0.707(±0.017)	0.707(±0.021)	0.713(±0.014)	0.692(±0.022)	0.689(±0.044)	0.709(±0.016)	0.707(±0.014)	
E \rightarrow D	0.745(±0.01)	0.768(±0.01)	0.768(±0.01)	0.774(±0.004)	0.779(±0.007)	0.739(±0.027)	0.749(±0.017)	0.773(±0.011)	0.774(±0.011)	
E \rightarrow K	0.855(±0.015)	0.875(±0.007)	0.875(±0.007)	0.875(±0.007)	0.878(±0.008)	0.852(±0.01)	0.856(±0.014)	0.889(±0.009)	0.897(±0.009)	
K \rightarrow B	0.715(±0.008)	0.731(±0.003)	0.731(±0.003)	0.726(±0.002)	0.739(±0.006)	0.724(±0.009)	0.716(±0.015)	0.732(±0.004)	0.73(±0.006)	
K \rightarrow D	0.736(±0.014)	0.761(±0.013)	0.761(±0.013)	0.755(±0.011)	0.764(±0.018)	0.734(±0.012)	0.728(±0.045)	0.758(±0.017)	0.75(±0.018)	
K \rightarrow E	0.842(±0.008)	0.856(±0.009)	0.856(±0.009)	0.858(±0.009)	0.855(±0.012)	0.835(±0.01)	0.832(±0.014)	0.858(±0.005)	0.847(±0.009)	
Avg.	0.767(±0.009)	0.785(±0.009)	0.785(±0.009)	0.785(±0.01)	0.787(±0.01)	0.769(±0.012)	0.766(±0.017)	0.787(±0.009)	0.784(±0.009)	

CoDATS										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
B \rightarrow D	0.783(±0.013)	0.8(±0.009)	0.8(±0.009)	0.8(±0.004)	0.803(±0.009)	0.788(±0.008)	0.792(±0.001)	0.802(±0.009)	0.801(±0.008)	
B \rightarrow E	0.754(±0.009)	0.782(±0.014)	0.782(±0.014)	0.787(±0.005)	0.787(±0.005)	0.77(±0.013)	0.772(±0.011)	0.788(±0.006)	0.802(±0.009)	
B \rightarrow K	0.771(±0.022)	0.808(±0.007)	0.808(±0.007)	0.808(±0.008)	0.807(±0.008)	0.775(±0.022)	0.793(±0.023)	0.806(±0.009)	0.817(±0.007)	
D \rightarrow B	0.774(±0.001)	0.796(±0.005)	0.796(±0.005)	0.795(±0.005)	0.807(±0.007)	0.701(±0.007)	0.792(±0.007)	0.802(±0.004)	0.8(±0.007)	
D \rightarrow E	0.769(±0.003)	0.808(±0.009)	0.808(±0.009)	0.81(±0.009)	0.808(±0.008)	0.789(±0.02)	0.777(±0.008)	0.811(±0.009)	0.817(±0.011)	
D \rightarrow K	0.782(±0.021)	0.819(±0.011)	0.819(±0.011)	0.82(±0.011)	0.82(±0.01)	0.796(±0.021)	0.789(±0.011)	0.82(±0.01)	0.828(±0.01)	
E \rightarrow B	0.687(±0.013)	0.714(±0.019)	0.714(±0.019)	0.713(±0.015)	0.714(±0.021)	0.71(±0.027)	0.738(±0.025)	0.731(±0.027)	0.747(±0.025)	0.736(±0.017)
E \rightarrow D	0.72(±0.013)	0.745(±0.023)	0.745(±0.023)	0.744(±0.026)	0.756(±0.017)	0.73(±0.027)	0.731(±0.027)	0.747(±0.025)	0.736(±0.017)	
E \rightarrow K	0.859(±0.016)	0.883(±0.009)	0.883(±0.009)	0.882(±0.008)	0.879(±0.008)	0.867(±0.01)	0.866(±0.01)	0.883(±0.012)	0.87(±0.012)	
K \rightarrow B	0.712(±0.012)	0.732(±0.004)	0.732(±0.004)	0.732(±0.009)	0.747(±0.012)	0.726(±0.037)	0.694(±0.008)	0.737(±0.006)	0.751(±0.016)	
K \rightarrow D	0.73(±0.022)	0.759(±0.006)	0.759(±0.006)	0.766(±0.009)	0.773(±0.016)	0.752(±0.015)	0.738(±0.007)	0.766(±0.004)	0.764(±0.016)	
K \rightarrow E	0.845(±0.0)	0.867(±0.01)	0.867(±0.01)	0.866(±0.011)	0.868(±0.007)	0.853(±0.003)	0.851(±0.011)	0.867(±0.007)	0.856(±0.007)	
Avg.	0.766(±0.012)	0.793(±0.009)	0.793(±0.009)	0.794(±0.01)	0.799(±0.01)	0.779(±0.017)	0.773(±0.011)	0.796(±0.01)	0.797(±0.012)	

Deep-Coral										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
B \rightarrow D	0.778(±0.018)	0.799(±0.011)	0.799(±0.011)	0.799(±0.006)	0.794(±0.011)	0.792(±0.023)	0.789(±0.025)	0.801(±0.01)	0.794(±0.012)	
B \rightarrow E	0.749(±0.013)	0.771(±0.01)	0.771(±0.01)	0.771(±0.008)	0.764(±0.018)	0.759(±0.002)	0.752(±0.01)	0.773(±0.008)	0.767(±0.008)	
B \rightarrow K	0.769(±0.011)	0.785(±0.013)	0.785(±0.013)	0.784(±0.016)	0.783(±0.015)	0.762(±0.013)	0.77(±0.014)	0.785(±0.017)	0.779(±0.017)	
D \rightarrow B	0.783(±0.009)	0.796(±0.003)	0.796(±0.003)	0.796(±0.005)	0.792(±0.006)	0.78(±0.005)	0.773(±0.005)	0.797(±0.004)	0.786(±0.004)	
D \rightarrow E	0.76(±0.013)	0.785(±0.002)	0.785(±0.002)	0.785(±0.002)	0.787(±0.006)	0.762(±0.006)	0.764(±0.008)	0.784(±0.003)	0.785(±0.005)	
D \rightarrow K	0.774(±0.001)	0.796(±0.003)	0.796(±0.003)	0.795(±0.004)	0.799(±0.004)	0.779(±0.002)	0.78(±0.003)	0.798(±0.001)	0.799(±0.001)	
E \rightarrow B	0.696(±0.009)	0.708(±0.022)	0.708(±0.022)	0.709(±0.024)	0.708(±0.024)	0.688(±0.016)	0.698(±0.018)	0.71(±0.024)	0.708(±0.027)	
E \rightarrow D	0.722(±0.009)	0.738(±0.005)	0.738(±0.005)	0.741(±0.005)	0.741(±0.005)	0.723(±0.016)	0.72(±0.017)	0.74(±0.004)	0.739(±0.001)	
E \rightarrow K	0.859(±0.01)	0.879(±0.005)	0.879(±0.005)	0.878(±0.007)	0.877(±0.007)	0.857(±0.007)	0.856(±0.009)	0.878(±0.007)	0.864(±0.005)	
K \rightarrow B	0.72(±0.01)	0.733(±0.009)	0.733(±0.009)	0.736(±0.009)	0.733(±0.009)	0.719(±0.011)	0.722(±0.022)	0.739(±0.01)	0.74(±0.005)	
K \rightarrow D	0.733(±0.011)	0.755(±0.012)	0.755(±0.012)	0.751(±0.009)	0.759(±0.009)	0.75(±0.				

Table 17: Mean and standard deviation (after \pm) of target classification accuracy on Amazon Reviews (Part 3) over 3 repetitions with different random initialization of model weights.

Task	CDAN									
	SO	Heuristic				Theoretical error guarantees				TB
		TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)		
B \rightarrow D	0.784(± 0.022)	0.803(± 0.007)	0.803(± 0.007)	0.803(± 0.004)	0.795(± 0.009)	0.795(± 0.009)	0.792(± 0.006)	0.804(± 0.009)	0.797(± 0.009)	
B \rightarrow E	0.758(± 0.013)	0.776(± 0.009)	0.776(± 0.009)	0.774(± 0.009)	0.776(± 0.012)	0.764(± 0.012)	0.764(± 0.012)	0.778(± 0.009)	0.775(± 0.009)	
B \rightarrow K	0.769(± 0.019)	0.796(± 0.008)	0.796(± 0.008)	0.796(± 0.008)	0.792(± 0.014)	0.78(± 0.006)	0.791(± 0.007)	0.796(± 0.01)	0.796(± 0.01)	
D \rightarrow B	0.786(± 0.01)	0.797(± 0.007)	0.797(± 0.007)	0.795(± 0.004)	0.79(± 0.004)	0.793(± 0.01)	0.782(± 0.01)	0.798(± 0.007)	0.794(± 0.008)	
D \rightarrow E	0.761(± 0.003)	0.8(± 0.004)	0.8(± 0.004)	0.8(± 0.006)	0.8(± 0.005)	0.775(± 0.007)	0.77(± 0.005)	0.799(± 0.004)	0.794(± 0.006)	
D \rightarrow K	0.772(± 0.012)	0.797(± 0.009)	0.797(± 0.009)	0.797(± 0.01)	0.80(± 0.015)	0.792(± 0.012)	0.797(± 0.015)	0.804(± 0.013)	0.801(± 0.015)	
E \rightarrow B	0.822(± 0.014)	0.834(± 0.015)	0.834(± 0.015)	0.834(± 0.015)	0.834(± 0.021)	0.829(± 0.021)	0.831(± 0.021)	0.831(± 0.019)	0.831(± 0.021)	
E \rightarrow D	0.72(± 0.018)	0.741(± 0.021)	0.741(± 0.021)	0.738(± 0.011)	0.711(± 0.006)	0.722(± 0.017)	0.723(± 0.028)	0.741(± 0.009)	0.737(± 0.009)	
E \rightarrow K	0.86(± 0.006)	0.879(± 0.011)	0.879(± 0.011)	0.878(± 0.01)	0.874(± 0.012)	0.861(± 0.018)	0.861(± 0.018)	0.879(± 0.011)	0.865(± 0.011)	
K \rightarrow B	0.706(± 0.004)	0.727(± 0.003)	0.727(± 0.003)	0.728(± 0.001)	0.722(± 0.005)	0.724(± 0.011)	0.722(± 0.004)	0.733(± 0.004)	0.724(± 0.004)	
K \rightarrow D	0.748(± 0.013)	0.765(± 0.013)	0.765(± 0.013)	0.764(± 0.011)	0.766(± 0.021)	0.752(± 0.021)	0.755(± 0.005)	0.762(± 0.01)	0.752(± 0.021)	
K \rightarrow E	0.845(± 0.007)	0.86(± 0.005)	0.86(± 0.005)	0.861(± 0.007)	0.863(± 0.003)	0.848(± 0.005)	0.849(± 0.007)	0.861(± 0.006)	0.85(± 0.003)	
Avg.	0.767(± 0.012)	0.787(± 0.01)	0.787(± 0.008)	0.787(± 0.011)	0.775(± 0.012)	0.776(± 0.012)	0.788(± 0.009)	0.783(± 0.01)		

Task	DANN									
	SO	Heuristic				Theoretical error guarantees				TB
		TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)		
B \rightarrow D	0.783(± 0.004)	0.794(± 0.003)	0.794(± 0.003)	0.794(± 0.003)	0.793(± 0.007)	0.793(± 0.003)	0.792(± 0.003)	0.804(± 0.008)	0.813(± 0.017)	
B \rightarrow E	0.759(± 0.011)	0.788(± 0.009)	0.787(± 0.009)	0.787(± 0.013)	0.796(± 0.006)	0.783(± 0.013)	0.782(± 0.013)	0.789(± 0.008)	0.8(± 0.005)	
B \rightarrow K	0.767(± 0.022)	0.804(± 0.007)	0.804(± 0.007)	0.804(± 0.008)	0.805(± 0.01)	0.796(± 0.003)	0.806(± 0.015)	0.804(± 0.008)	0.82(± 0.004)	
D \rightarrow B	0.779(± 0.002)	0.8(± 0.007)	0.8(± 0.007)	0.799(± 0.006)	0.802(± 0.003)	0.796(± 0.004)	0.778(± 0.014)	0.804(± 0.004)	0.799(± 0.004)	
D \rightarrow E	0.767(± 0.016)	0.805(± 0.006)	0.805(± 0.006)	0.807(± 0.009)	0.808(± 0.012)	0.775(± 0.023)	0.78(± 0.035)	0.81(± 0.007)	0.815(± 0.007)	
D \rightarrow K	0.784(± 0.013)	0.815(± 0.008)	0.815(± 0.008)	0.817(± 0.011)	0.829(± 0.014)	0.788(± 0.007)	0.779(± 0.012)	0.816(± 0.011)	0.827(± 0.014)	
E \rightarrow B	0.701(± 0.019)	0.712(± 0.014)	0.712(± 0.014)	0.712(± 0.015)	0.724(± 0.014)	0.694(± 0.03)	0.702(± 0.022)	0.712(± 0.012)	0.721(± 0.014)	
E \rightarrow D	0.736(± 0.016)	0.743(± 0.014)	0.743(± 0.014)	0.747(± 0.012)	0.758(± 0.012)	0.735(± 0.053)	0.738(± 0.019)	0.751(± 0.008)	0.757(± 0.012)	
E \rightarrow K	0.854(± 0.016)	0.877(± 0.011)	0.877(± 0.011)	0.878(± 0.012)	0.879(± 0.014)	0.86(± 0.008)	0.858(± 0.007)	0.88(± 0.013)	0.875(± 0.012)	
K \rightarrow B	0.711(± 0.014)	0.741(± 0.004)	0.741(± 0.004)	0.74(± 0.01)	0.759(± 0.007)	0.718(± 0.012)	0.729(± 0.013)	0.75(± 0.014)	0.747(± 0.007)	
K \rightarrow D	0.738(± 0.006)	0.768(± 0.011)	0.768(± 0.011)	0.765(± 0.011)	0.778(± 0.013)	0.733(± 0.013)	0.758(± 0.019)	0.778(± 0.012)	0.762(± 0.012)	
K \rightarrow E	0.837(± 0.013)	0.864(± 0.008)	0.864(± 0.008)	0.864(± 0.007)	0.865(± 0.009)	0.86(± 0.013)	0.851(± 0.008)	0.866(± 0.011)	0.859(± 0.011)	
Avg.	0.767(± 0.012)	0.792(± 0.009)	0.792(± 0.009)	0.793(± 0.01)	0.8(± 0.01)	0.776(± 0.015)	0.778(± 0.015)	0.797(± 0.009)	0.8(± 0.01)	

Task	DSAN									
	SO	Heuristic				Theoretical error guarantees				TB
		TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)		
B \rightarrow D	0.779(± 0.013)	0.801(± 0.014)	0.801(± 0.014)	0.803(± 0.015)	0.807(± 0.01)	0.79(± 0.02)	0.689(± 0.108)	0.805(± 0.014)	0.805(± 0.01)	
B \rightarrow E	0.752(± 0.005)	0.788(± 0.005)	0.788(± 0.005)	0.783(± 0.005)	0.798(± 0.002)	0.775(± 0.022)	0.764(± 0.02)	0.787(± 0.006)	0.796(± 0.002)	
B \rightarrow K	0.769(± 0.017)	0.806(± 0.011)	0.806(± 0.011)	0.805(± 0.011)	0.815(± 0.007)	0.786(± 0.016)	0.798(± 0.007)	0.816(± 0.005)	0.816(± 0.005)	
D \rightarrow B	0.782(± 0.008)	0.795(± 0.004)	0.795(± 0.004)	0.795(± 0.002)	0.798(± 0.007)	0.79(± 0.01)	0.794(± 0.011)	0.804(± 0.005)	0.802(± 0.007)	
D \rightarrow E	0.771(± 0.002)	0.804(± 0.008)	0.804(± 0.008)	0.804(± 0.008)	0.816(± 0.012)	0.79(± 0.02)	0.792(± 0.02)	0.808(± 0.003)	0.813(± 0.012)	
D \rightarrow K	0.786(± 0.013)	0.804(± 0.01)	0.804(± 0.01)	0.802(± 0.012)	0.821(± 0.014)	0.789(± 0.013)	0.801(± 0.025)	0.807(± 0.012)	0.823(± 0.016)	
E \rightarrow B	0.702(± 0.021)	0.716(± 0.017)	0.716(± 0.017)	0.718(± 0.021)	0.718(± 0.018)	0.71(± 0.021)	0.707(± 0.018)	0.721(± 0.02)	0.711(± 0.02)	
E \rightarrow D	0.725(± 0.007)	0.743(± 0.008)	0.743(± 0.008)	0.741(± 0.005)	0.744(± 0.008)	0.723(± 0.021)	0.735(± 0.012)	0.749(± 0.005)	0.732(± 0.005)	
E \rightarrow K	0.865(± 0.013)	0.883(± 0.009)	0.883(± 0.009)	0.882(± 0.008)	0.886(± 0.009)	0.874(± 0.003)	0.876(± 0.006)	0.883(± 0.01)	0.876(± 0.009)	
K \rightarrow B	0.716(± 0.016)	0.741(± 0.01)	0.741(± 0.01)	0.736(± 0.007)	0.746(± 0.011)	0.712(± 0.014)	0.707(± 0.016)	0.742(± 0.008)	0.739(± 0.011)	
K \rightarrow D	0.739(± 0.023)	0.773(± 0.006)	0.773(± 0.006)	0.768(± 0.003)	0.777(± 0.01)	0.746(± 0.016)	0.726(± 0.035)	0.776(± 0.005)	0.763(± 0.012)	
K \rightarrow E	0.84(± 0.01)	0.86(± 0.005)	0.86(± 0.005)	0.858(± 0.007)	0.868(± 0.005)	0.854(± 0.001)	0.835(± 0.004)	0.865(± 0.008)	0.857(± 0.005)	
Avg.	0.769(± 0.012)	0.792(± 0.009)	0.792(± 0.009)	0.791(± 0.009)	0.8(± 0.009)	0.779(± 0.016)	0.763(± 0.037)	0.795(± 0.009)	0.795(± 0.009)	

D.7.3 DETAILED MINIDOMAINNET RESULTS

Table 18: Mean and standard deviation (after \pm) of target classification accuracy on MiniDomainNet dataset over three different random initialization of model weights and five domain adaptation tasks.

Method	MiniDomainNet								
	Heuristic				Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	INV	DEV	IWA (ours)	TB
BobMM	0.509(± 0.018)	0.526(± 0.009)	0.524(± 0.010)	0.528(± 0.010)	0.518(± 0.015)	0.511(± 0.018)	0.511(± 0.018)	0.531(± 0.007)	0.537(± 0.020)
AdvSKM	0.509(± 0.015)	0.516(± 0.011)	0.514(± 0.014)	0.514(± 0.017)	0.514(± 0.009)	0.515(± 0.032)	0.512(± 0.032)	0.516(± 0.011)	0.522(± 0.017)
DIRT	0.499(± 0.022)	0.517(± 0.006)	0.515(± 0.009)	0.508(± 0.006)	0.507(± 0.018)	0.493(± 0.028)	0.498(± 0.033)	0.519(± 0.006)	0.525(± 0.020)
DDC	0.510(± 0.032)	0.514(± 0.012)	0.512(± 0.018)	0.511(± 0.019)	0.514(± 0.015)	0.511(± 0.028)	0.512(± 0.043)	0.516(± 0.015)	0.521(± 0.019)
CMD	0.509(± 0.019)	0.528(± 0.011)	0.531(± 0.016)	0.531(± 0.016)	0.522(± 0.013)	0.518(± 0.025)	0.490(± 0.037)	0.533(± 0.009)	0.533(± 0.024)
MMDA	0.509(± 0.021)	0.524(± 0.004)	0.522(± 0.005)	0.523(± 0.007)	0.519(± 0.012)	0.517(± 0.008)	0.524(± 0.006)	0.527(± 0.009)	0.531(± 0.029)
CoDATS	0.502(± 0.018)	0.535(± 0.020)	0.534(± 0.018)	0.534(± 0.019)	0.529(± 0.015)	0.517(± 0.015)	0.524(± 0.015)	0.530(± 0.010)	0.539(± 0.025)
DecoCoral	0.502(± 0.022)	0.528(± 0.010)	0.534(± 0.013)	0.538(± 0.012)	0.536(± 0.010)	0.518(± 0.029)	0.524(± 0.040)	0.539(± 0.019)	0.538(± 0.024)
CDAN	0.514(± 0.028)	0.526(± 0.019)	0.521(± 0.011)	0.525(± 0.008)	0.517(± 0.005)	0.513(± 0.011)	0.516(± 0.014)	0.532(± 0.010)	0.542(± 0.017)
DANN	0.496(± 0.023)	0.530(± 0.011)	0.535(± 0.011)	0.538(± 0.005)	0.516(± 0.011)	0.519(± 0.015)	0.515(± 0.013)	0.541(± 0.006)	0.532(± 0.014)
DSAN	0.509(± 0.022)	0.537(± 0.011)	0.534(± 0.013)	0.543(± 0.011)	0.525(± 0.013)	0.513(± 0.018)	0.535(± 0.023)	0.546(± 0.007)	0.563(± 0.013)
Avg.	0.507(± 0.022)	0.526(± 0.011)	0.525(± 0.014)	0.526(± 0.013)	0.518(± 0.012)	0.513(± 0.022)	0.515(± 0.028)	0.531(± 0.011)	0.534(± 0.022)

Table 19: Mean and standard deviation (after \pm) of target classification accuracy on MiniDomainNet (Part 1) over 3 repetitions with different random initialization of model weights.

HoMM										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
R → C	0.552(± 0.041)	0.587(± 0.017)	0.587(± 0.017)	0.587(± 0.005)	0.557(± 0.008)	0.538(± 0.029)	0.56(± 0.017)	0.596(± 0.009)	0.607(± 0.022)	
R → I	0.373(± 0.014)	0.376(± 0.012)	0.376(± 0.012)	0.373(± 0.005)	0.386(± 0.007)	0.37(± 0.01)	0.37(± 0.01)	0.392(± 0.007)	0.413(± 0.041)	
R → P	0.709(± 0.026)	0.722(± 0.01)	0.722(± 0.01)	0.718(± 0.006)	0.709(± 0.011)	0.704(± 0.013)	0.705(± 0.014)	0.73(± 0.003)	0.721(± 0.003)	
R → Q	0.332(± 0.009)	0.35(± 0.006)	0.35(± 0.006)	0.353(± 0.008)	0.356(± 0.02)	0.367(± 0.019)	0.342(± 0.018)	0.353(± 0.003)	0.37(± 0.013)	
R → S	0.579(± 0.022)	0.587(± 0.012)	0.587(± 0.012)	0.586(± 0.003)	0.582(± 0.01)	0.577(± 0.004)	0.577(± 0.004)	0.586(± 0.008)	0.609(± 0.011)	
Avg.	0.509(± 0.023)	0.524(± 0.011)	0.524(± 0.011)	0.523(± 0.005)	0.518(± 0.011)	0.511(± 0.015)	0.511(± 0.013)	0.531(± 0.006)	0.544(± 0.018)	

AdvSKM										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
R → C	0.544(± 0.017)	0.56(± 0.025)	0.56(± 0.025)	0.566(± 0.03)	0.544(± 0.024)	0.549(± 0.022)	0.549(± 0.022)	0.568(± 0.009)	0.582(± 0.046)	
R → I	0.38(± 0.018)	0.373(± 0.016)	0.373(± 0.016)	0.371(± 0.009)	0.391(± 0.037)	0.394(± 0.021)	0.394(± 0.021)	0.377(± 0.013)	0.395(± 0.02)	
R → P	0.723(± 0.009)	0.721(± 0.004)	0.721(± 0.004)	0.719(± 0.003)	0.712(± 0.006)	0.706(± 0.041)	0.695(± 0.034)	0.721(± 0.003)	0.723(± 0.009)	
R → Q	0.322(± 0.027)	0.332(± 0.006)	0.332(± 0.006)	0.328(± 0.005)	0.335(± 0.002)	0.335(± 0.002)	0.332(± 0.006)	0.333(± 0.005)	0.335(± 0.004)	
R → S	0.579(± 0.017)	0.586(± 0.001)	0.586(± 0.001)	0.584(± 0.003)	0.586(± 0.005)	0.589(± 0.007)	0.589(± 0.007)	0.582(± 0.004)	0.592(± 0.009)	
Avg.	0.509(± 0.018)	0.514(± 0.01)	0.514(± 0.01)	0.514(± 0.01)	0.514(± 0.015)	0.515(± 0.018)	0.512(± 0.018)	0.516(± 0.007)	0.525(± 0.018)	

DIRT										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
R → C	0.53(± 0.013)	0.577(± 0.023)	0.577(± 0.023)	0.546(± 0.031)	0.571(± 0.017)	0.541(± 0.025)	0.544(± 0.025)	0.577(± 0.025)	0.587(± 0.037)	
R → I	0.373(± 0.038)	0.365(± 0.041)	0.365(± 0.041)	0.371(± 0.037)	0.37(± 0.007)	0.36(± 0.011)	0.365(± 0.009)	0.38(± 0.01)	0.431(± 0.045)	
R → P	0.704(± 0.031)	0.708(± 0.026)	0.708(± 0.026)	0.706(± 0.02)	0.680(± 0.015)	0.693(± 0.03)	0.675(± 0.012)	0.713(± 0.03)	0.715(± 0.028)	
R → Q	0.34(± 0.02)	0.345(± 0.031)	0.345(± 0.031)	0.34(± 0.046)	0.336(± 0.024)	0.326(± 0.086)	0.365(± 0.038)	0.342(± 0.022)	0.382(± 0.027)	
R → S	0.55(± 0.028)	0.578(± 0.006)	0.578(± 0.006)	0.571(± 0.006)	0.571(± 0.012)	0.572(± 0.005)	0.525(± 0.088)	0.583(± 0.016)	0.588(± 0.033)	
Avg.	0.499(± 0.026)	0.515(± 0.027)	0.515(± 0.027)	0.508(± 0.028)	0.507(± 0.015)	0.493(± 0.031)	0.498(± 0.051)	0.519(± 0.025)	0.541(± 0.034)	

DDC										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
R → C	0.555(± 0.045)	0.555(± 0.013)	0.555(± 0.013)	0.549(± 0.008)	0.552(± 0.009)	0.546(± 0.013)	0.546(± 0.013)	0.584(± 0.025)	0.585(± 0.076)	
R → I	0.374(± 0.014)	0.369(± 0.003)	0.369(± 0.003)	0.374(± 0.014)	0.374(± 0.02)	0.379(± 0.009)	0.379(± 0.009)	0.388(± 0.009)	0.391(± 0.014)	
R → P	0.709(± 0.025)	0.711(± 0.01)	0.711(± 0.01)	0.712(± 0.004)	0.709(± 0.004)	0.713(± 0.006)	0.718(± 0.003)	0.715(± 0.002)	0.717(± 0.004)	
R → Q	0.334(± 0.002)	0.331(± 0.003)	0.331(± 0.003)	0.332(± 0.0)	0.327(± 0.018)	0.333(± 0.005)	0.335(± 0.002)	0.335(± 0.003)	0.337(± 0.004)	
R → S	0.576(± 0.022)	0.571(± 0.01)	0.571(± 0.01)	0.578(± 0.008)	0.582(± 0.005)	0.582(± 0.005)	0.582(± 0.005)	0.579(± 0.007)	0.585(± 0.021)	
Avg.	0.51(± 0.021)	0.512(± 0.005)	0.512(± 0.005)	0.511(± 0.007)	0.514(± 0.012)	0.511(± 0.008)	0.512(± 0.006)	0.516(± 0.009)	0.528(± 0.024)	

CMD										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
R → C	0.555(± 0.042)	0.598(± 0.008)	0.598(± 0.008)	0.59(± 0.008)	0.544(± 0.021)	0.552(± 0.031)	0.443(± 0.067)	0.593(± 0.009)	0.617(± 0.025)	
R → I	0.377(± 0.009)	0.368(± 0.007)	0.368(± 0.007)	0.371(± 0.005)	0.395(± 0.007)	0.379(± 0.011)	0.379(± 0.011)	0.379(± 0.007)	0.389(± 0.007)	
R → P	0.704(± 0.02)	0.741(± 0.015)	0.741(± 0.015)	0.733(± 0.01)	0.723(± 0.02)	0.732(± 0.019)	0.736(± 0.013)	0.74(± 0.041)	0.734(± 0.015)	
R → Q	0.332(± 0.015)	0.365(± 0.005)	0.365(± 0.005)	0.374(± 0.003)	0.361(± 0.036)	0.358(± 0.059)	0.303(± 0.061)	0.364(± 0.007)	0.378(± 0.008)	
R → S	0.578(± 0.022)	0.582(± 0.01)	0.582(± 0.01)	0.587(± 0.005)	0.589(± 0.019)	0.591(± 0.015)	0.592(± 0.003)	0.592(± 0.003)	0.59(± 0.003)	
Avg.	0.509(± 0.022)	0.531(± 0.009)	0.531(± 0.009)	0.531(± 0.006)	0.522(± 0.018)	0.518(± 0.028)	0.49(± 0.033)	0.533(± 0.006)	0.542(± 0.012)	

MMDA										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
R → C	0.555(± 0.042)	0.587(± 0.009)	0.587(± 0.009)	0.571(± 0.013)	0.56(± 0.019)	0.544(± 0.033)	0.544(± 0.033)	0.585(± 0.009)	0.585(± 0.009)	
R → I	0.377(± 0.009)	0.379(± 0.007)	0.379(± 0.007)	0.376(± 0.012)	0.389(± 0.007)	0.389(± 0.021)	0.389(± 0.021)	0.392(± 0.007)	0.373(± 0.003)	0.415(± 0.016)
R → P	0.704(± 0.02)	0.714(± 0.014)	0.714(± 0.014)	0.714(± 0.011)	0.717(± 0.014					

Table 20: Mean and standard deviation (after \pm) of target classification accuracy on MiniDomainNet (Part 2) over 3 repetitions with different random initialization of model weights.

CoDATS										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
R → C	0.538(±0.072)	0.585(±0.025)	0.585(±0.025)	0.607(±0.028)	0.585(±0.013)	0.555(±0.031)	0.626(±0.058)	0.577(±0.026)	0.617(±0.058)	
R → I	0.365(±0.037)	0.371(±0.024)	0.371(±0.024)	0.367(±0.016)	0.38(±0.02)	0.373(±0.041)	0.363(±0.056)	0.371(±0.012)	0.406(±0.021)	
R → P	0.689(±0.024)	0.735(±0.01)	0.735(±0.01)	0.734(±0.015)	0.726(±0.014)	0.689(±0.043)	0.689(±0.043)	0.743(±0.016)	0.732(±0.016)	
R → Q	0.322(±0.022)	0.356(±0.02)	0.356(±0.02)	0.358(±0.017)	0.34(±0.017)	0.384(±0.012)	0.348(±0.044)	0.364(±0.012)	0.417(±0.025)	
R → S	0.597(±0.004)	0.619(±0.013)	0.619(±0.013)	0.616(±0.018)	0.593(±0.01)	0.592(±0.013)	0.592(±0.013)	0.622(±0.009)	0.627(±0.008)	
Avg.	0.502(±0.032)	0.533(±0.018)	0.533(±0.018)	0.536(±0.019)	0.525(±0.015)	0.518(±0.028)	0.524(±0.043)	0.536(±0.015)	0.56(±0.026)	

Deep-Coral										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
R → C	0.555(±0.045)	0.585(±0.017)	0.585(±0.017)	0.582(±0.014)	0.557(±0.0)	0.538(±0.025)	0.588(±0.013)	0.59(±0.022)	0.601(±0.056)	
R → I	0.376(±0.012)	0.371(±0.01)	0.371(±0.01)	0.369(±0.01)	0.363(±0.012)	0.368(±0.003)	0.398(±0.009)	0.392(±0.012)	0.397(±0.017)	
R → P	0.711(±0.021)	0.726(±0.015)	0.726(±0.015)	0.723(±0.007)	0.722(±0.007)	0.722(±0.012)	0.722(±0.012)	0.738(±0.012)	0.734(±0.012)	
R → Q	0.316(±0.041)	0.383(±0.003)	0.383(±0.003)	0.386(±0.004)	0.365(±0.008)	0.378(±0.011)	0.375(±0.033)	0.38(±0.002)	0.388(±0.002)	
R → S	0.569(±0.02)	0.606(±0.009)	0.606(±0.009)	0.606(±0.005)	0.581(±0.007)	0.588(±0.008)	0.588(±0.008)	0.622(±0.001)	0.658(±0.035)	
Avg.	0.505(±0.028)	0.534(±0.011)	0.534(±0.011)	0.533(±0.008)	0.52(±0.005)	0.518(±0.011)	0.524(±0.014)	0.539(±0.01)	0.554(±0.022)	

CDAN										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
R → C	0.557(±0.005)	0.587(±0.013)	0.587(±0.013)	0.593(±0.026)	0.563(±0.009)	0.559(±0.025)	0.574(±0.015)	0.598(±0.022)	0.615(±0.008)	
R → I	0.386(±0.01)	0.359(±0.005)	0.359(±0.005)	0.357(±0.009)	0.368(±0.005)	0.363(±0.04)	0.385(±0.016)	0.357(±0.009)	0.388(±0.01)	
R → P	0.709(±0.022)	0.732(±0.02)	0.732(±0.02)	0.731(±0.023)	0.714(±0.006)	0.729(±0.030)	0.723(±0.036)	0.731(±0.004)	0.716(±0.023)	
R → Q	0.342(±0.006)	0.352(±0.02)	0.352(±0.024)	0.349(±0.02)	0.347(±0.016)	0.356(±0.034)	0.38(±0.028)	0.364(±0.013)	0.442(±0.114)	
R → S	0.576(±0.027)	0.588(±0.01)	0.588(±0.01)	0.593(±0.005)	0.596(±0.01)	0.574(±0.022)	0.568(±0.03)	0.61(±0.008)	0.651(±0.056)	
Avg.	0.514(±0.015)	0.524(±0.014)	0.524(±0.014)	0.525(±0.017)	0.517(±0.009)	0.513(±0.032)	0.516(±0.032)	0.532(±0.011)	0.562(±0.042)	

DANN										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
R → C	0.519(±0.013)	0.626(±0.021)	0.626(±0.021)	0.623(±0.036)	0.557(±0.014)	0.582(±0.022)	0.56(±0.07)	0.607(±0.008)	0.631(±0.038)	
R → I	0.376(±0.016)	0.37(±0.005)	0.37(±0.005)	0.377(±0.013)	0.377(±0.012)	0.38(±0.008)	0.37(±0.016)	0.37(±0.019)	0.4(±0.028)	
R → P	0.705(±0.011)	0.704(±0.013)	0.704(±0.013)	0.704(±0.011)	0.696(±0.006)	0.709(±0.006)	0.707(±0.014)	0.721(±0.008)	0.713(±0.008)	
R → Q	0.3(±0.043)	0.377(±0.008)	0.377(±0.008)	0.38(±0.006)	0.343(±0.004)	0.351(±0.048)	0.351(±0.048)	0.374(±0.003)	0.392(±0.005)	
R → S	0.581(±0.014)	0.598(±0.023)	0.598(±0.023)	0.607(±0.016)	0.596(±0.01)	0.586(±0.015)	0.586(±0.015)	0.617(±0.005)	0.657(±0.015)	
Avg.	0.496(±0.019)	0.535(±0.016)	0.535(±0.016)	0.538(±0.016)	0.516(±0.013)	0.519(±0.025)	0.515(±0.037)	0.541(±0.009)	0.559(±0.018)	

DSAN										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
R → C	0.555(±0.042)	0.637(±0.021)	0.637(±0.021)	0.639(±0.022)	0.566(±0.016)	0.557(±0.059)	0.571(±0.045)	0.639(±0.008)	0.648(±0.014)	
R → I	0.377(±0.009)	0.363(±0.01)	0.363(±0.01)	0.374(±0.015)	0.406(±0.005)	0.362(±0.034)	0.362(±0.034)	0.377(±0.016)	0.404(±0.005)	
R → P	0.705(±0.021)	0.712(±0.009)	0.712(±0.009)	0.714(±0.005)	0.72(±0.008)	0.716(±0.003)	0.719(±0.003)	0.719(±0.007)	0.716(±0.008)	
R → Q	0.332(±0.012)	0.364(±0.021)	0.364(±0.021)	0.38(±0.012)	0.34(±0.016)	0.342(±0.009)	0.437(±0.083)	0.372(±0.011)	0.478(±0.013)	
R → S	0.577(±0.026)	0.594(±0.004)	0.594(±0.004)	0.607(±0.005)	0.594(±0.003)	0.585(±0.004)	0.585(±0.004)	0.625(±0.016)	0.639(±0.052)	
Avg.	0.509(±0.022)	0.534(±0.013)	0.534(±0.013)	0.543(±0.012)	0.525(±0.01)	0.513(±0.022)	0.535(±0.034)	0.546(±0.012)	0.577(±0.018)	

D.7.4 DETAILED TIME-SERIES RESULTS

Table 21: Mean and standard deviation (after \pm) of target classification accuracy on Sleep-EDF (Part 1) over 3 repetitions with different random initialization of model weights.

HoMM								
Task	Heuristic				Theoretical error guarantees			
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)
0 → 11	0.565(±0.033)	0.557(±0.066)	0.557(±0.066)	0.553(±0.068)	0.72(±0.052)	0.694(±0.075)	0.488(±0.09)	0.719(±0.04)
12 → 5	0.698(±0.081)	0.771(±0.014)	0.771(±0.014)	0.768(±0.008)	0.788(±0.022)	0.776(±0.026)	0.776(±0.026)	0.762(±0.022)
16 → 1	0.664(±0.068)	0.702(±0.02)	0.702(±0.02)	0.698(±0.025)	0.616(±0.017)	0.628(±0.017)	0.628(±0.017)	0.682(±0.008)
7 → 18	0.712(±0.024)	0.734(±0.021)	0.734(±0.021)	0.736(±0.012)	0.678(±0.006)	0.729(±0.025)	0.734(±0.028)	0.74(±0.008)
9 → 14	0.741(±0.031)	0.832(±0.008)	0.832(±0.008)	0.833(±0.01)	0.82(±0.024)	0.802(±0.025)	0.764(±0.07)	0.832(±0.01)
Avg.	0.676(±0.047)	0.719(±0.026)	0.719(±0.026)	0.718(±0.025)	0.724(±0.024)	0.726(±0.034)	0.678(±0.046)	0.747(±0.018)
								0.753(±0.024)

AdvSKM								
Task	Heuristic				Theoretical error guarantees			
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)
0 → 11	0.611(±0.005)	0.59(±0.02)	0.50(±0.02)	0.589(±0.025)	0.706(±0.043)	0.645(±0.038)	0.595(±0.035)	0.664(±0.05)
12 → 5	0.723(±0.061)	0.75(±0.02)	0.75(±0.02)	0.747(±0.011)	0.777(±0.024)	0.753(±0.015)	0.753(±0.015)	0.754(±0.027)
16 → 1	0.558(±0.035)	0.667(±0.037)	0.667(±0.037)	0.671(±0.035)	0.619(±0.043)	0.65(±0.076)	0.65(±0.076)	0.658(±0.03)
7 → 18	0.651(±0.053)	0.72(±0.022)	0.72(±0.022)	0.72(±0.018)	0.669(±0.018)	0.706(±0.041)	0.681(±0.023)	0.707(±0.004)
9 → 14	0.78(±0.027)	0.835(±0.016)	0.835(±0.016)	0.835(±0.016)	0.819(±0.032)	0.76(±0.059)	0.78(±0.027)	0.824(±0.014)
Avg.	0.665(±0.036)	0.712(±0.023)	0.712(±0.023)	0.712(±0.021)	0.718(±0.032)	0.703(±0.046)	0.692(±0.035)	0.722(±0.025)
								0.726(±0.035)

DIRT								
Task	Heuristic				Theoretical error guarantees			
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)
0 → 11	0.582(±0.041)	0.493(±0.106)	0.493(±0.106)	0.509(±0.094)	0.625(±0.031)	0.447(±0.123)	0.487(±0.188)	0.54(±0.043)
12 → 5	0.712(±0.08)	0.835(±0.015)	0.835(±0.015)	0.832(±0.017)	0.87(±0.019)	0.805(±0.061)	0.789(±0.105)	0.829(±0.02)
16 → 1	0.526(±0.162)	0.757(±0.025)	0.757(±0.025)	0.755(±0.021)	0.671(±0.017)	0.618(±0.124)	0.618(±0.124)	0.746(±0.03)
7 → 18	0.749(±0.041)	0.777(±0.018)	0.777(±0.018)	0.775(±0.005)	0.717(±0.044)	0.74(±0.069)	0.719(±0.072)	0.762(±0.0)
9 → 14	0.711(±0.098)	0.862(±0.016)	0.862(±0.016)	0.871(±0.014)	0.826(±0.027)	0.786(±0.071)	0.818(±0.045)	0.868(±0.012)
Avg.	0.656(±0.084)	0.745(±0.036)	0.745(±0.036)	0.748(±0.03)	0.742(±0.039)	0.679(±0.09)	0.668(±0.107)	0.749(±0.021)
								0.78(±0.023)

DDC								
Task	Heuristic				Theoretical error guarantees			
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)
0 → 11	0.546(±0.033)	0.603(±0.029)	0.603(±0.029)	0.598(±0.027)	0.658(±0.06)	0.629(±0.059)	0.492(±0.032)	0.645(±0.095)
12 → 5	0.695(±0.075)	0.751(±0.01)	0.751(±0.01)	0.749(±0.012)	0.779(±0.029)	0.716(±0.046)	0.716(±0.046)	0.755(±0.015)
16 → 1	0.551(±0.091)	0.683(±0.013)	0.683(±0.013)	0.672(±0.005)	0.56(±0.028)	0.618(±0.021)	0.623(±0.012)	0.666(±0.017)
7 → 18	0.699(±0.022)	0.729(±0.038)	0.729(±0.038)	0.717(±0.018)	0.674(±0.009)	0.724(±0.029)	0.724(±0.036)	0.733(±0.013)
9 → 14	0.737(±0.035)	0.84(±0.014)	0.84(±0.014)	0.827(±0.025)	0.802(±0.014)	0.788(±0.042)	0.823(±0.016)	0.792(±0.014)
Avg.	0.646(±0.051)	0.721(±0.021)	0.721(±0.021)	0.712(±0.018)	0.695(±0.028)	0.694(±0.033)	0.666(±0.034)	0.724(±0.031)
								0.737(±0.034)

CMD								
Task	Heuristic				Theoretical error guarantees			
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)
0 → 11	0.536(±0.041)	0.564(±0.01)	0.564(±0.01)	0.551(±0.027)	0.655(±0.079)	0.607(±0.087)	0.505(±0.016)	0.581(±0.009)
12 → 5	0.751(±0.032)	0.805(±0.012)	0.805(±0.012)	0.79(±0.016)	0.798(±0.016)	0.784(±0.012)	0.629(±0.044)	0.793(±0.014)
16 → 1	0.557(±0.099)	0.747(±0.011)	0.747(±0.011)	0.701(±0.034)	0.624(±0.018)	0.678(±0.005)	0.665(±0.018)	0.69(±0.015)
7 → 18	0.681(±0.047)	0.715(±0.017)	0.715(±0.017)	0.727(±0.004)	0.669(±0.019)	0.741(±0.005)	0.728(±0.025)	0.738(±0.004)
9 → 14	0.74(±0.07)	0.85(±0.008)	0.85(±0.008)	0.848(±0.014)	0.797(±0.021)	0.772(±0.064)	0.672(±0.229)	0.841(±0.006)
Avg.	0.653(±0.058)	0.736(±0.012)	0.736(±0.012)	0.723(±0.019)	0.709(±0.031)	0.710(±0.038)	0.64(±0.066)	0.729(±0.01)
								0.757(±0.03)

MMDA								
Task	Heuristic				Theoretical error guarantees			
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)
0 → 11	0.538(±0.032)	0.52(±0.01)	0.52(±0.01)	0.523(±0.007)	0.651(±0.025)	0.596(±0.086)	0.405(±0.099)	0.612(±0.037)
12 → 5	0.749(±0.034)	0.797(±0.016)	0.797(±0.016)	0.79(±0.025)	0.798(±0.006)	0.698(±0.079)	0.747(±0.076)	0.793(±0.01)
16 → 1	0.544(±0.116)	0.705(±0.015)	0.705(±0.015)	0.7(±0.022)	0.645(±0.032)	0.703(±0.02)	0.692(±0.008)	0.7(±0.014)
7 → 18	0.678(±0.046)	0.773(±0.01)	0.773(±0.01)	0.762(±0.0)	0.68(±0.051)	0.729(±0.094)	0.702(±0.082)	0.772(±0.005)
9 → 14	0.742(±0.071)	0.84(±0.029)	0.84(±0.029)	0.839(±0.027)	0.793(±0.014)	0.792(±0.073)	0.755(±0.051)	0.85(±0.012)
Avg.	0.65(±0.06)	0.727(±0.016)	0.727(±0.016)	0.723(±0.016)	0.714(±0.026)	0.704(±0.07)	0.66(±0.063)	0.745(±0.016)
								0.762(±0.025)

Table 22: Mean and standard deviation (after \pm) of target classification accuracy on Sleep-EDF (Part 2) over 3 repetitions with different random initialization of model weights.

CoDATS										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
0 → 11	0.568(±0.027)	0.565(±0.118)	0.565(±0.118)	0.556(±0.107)	0.625(±0.023)	0.530(±0.155)	0.564(±0.008)	0.602(±0.027)	0.6(±0.023)	
12 → 5	0.682(±0.043)	0.812(±0.014)	0.812(±0.014)	0.805(±0.014)	0.827(±0.012)	0.75(±0.024)	0.728(±0.018)	0.809(±0.007)	0.798(±0.012)	
16 → 1	0.574(±0.026)	0.732(±0.016)	0.732(±0.016)	0.729(±0.007)	0.642(±0.042)	0.633(±0.066)	0.676(±0.031)	0.729(±0.004)	0.735(±0.082)	
7 → 18	0.759(±0.039)	0.749(±0.023)	0.749(±0.023)	0.75(±0.02)	0.715(±0.014)	0.724(±0.008)	0.717(±0.092)	0.742(±0.01)	0.771(±0.02)	
9 → 14	0.779(±0.038)	0.837(±0.013)	0.837(±0.013)	0.839(±0.008)	0.805(±0.01)	0.771(±0.026)	0.763(±0.006)	0.84(±0.014)	0.811(±0.014)	
Avg.	0.672(±0.035)	0.739(±0.037)	0.739(±0.037)	0.736(±0.031)	0.723(±0.02)	0.683(±0.056)	0.69(±0.031)	0.744(±0.012)	0.743(±0.03)	

Deep-Coral										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
0 → 11	0.547(±0.032)	0.586(±0.028)	0.586(±0.028)	0.507(±0.029)	0.652(±0.053)	0.629(±0.003)	0.569(±0.11)	0.592(±0.028)	0.637(±0.058)	
12 → 5	0.694(±0.094)	0.87(±0.02)	0.87(±0.02)	0.745(±0.012)	0.721(±0.023)	0.738(±0.036)	0.750(±0.006)	0.755(±0.018)	0.773(±0.028)	
16 → 1	0.545(±0.097)	0.68(±0.015)	0.68(±0.015)	0.671(±0.007)	0.564(±0.034)	0.605(±0.018)	0.553(±0.104)	0.662(±0.017)	0.728(±0.036)	
7 → 18	0.698(±0.022)	0.73(±0.02)	0.73(±0.02)	0.716(±0.02)	0.673(±0.006)	0.723(±0.027)	0.711(±0.034)	0.733(±0.013)	0.741(±0.016)	
9 → 14	0.733(±0.047)	0.836(±0.016)	0.836(±0.016)	0.826(±0.02)	0.802(±0.018)	0.786(±0.014)	0.784(±0.043)	0.82(±0.02)	0.792(±0.016)	
Avg.	0.643(±0.056)	0.717(±0.024)	0.717(±0.024)	0.712(±0.018)	0.694(±0.029)	0.7(±0.031)	0.675(±0.065)	0.713(±0.019)	0.735(±0.031)	

CDAN										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
0 → 11	0.553(±0.024)	0.549(±0.005)	0.499(±0.005)	0.507(±0.070)	0.643(±0.053)	0.529(±0.111)	0.529(±0.024)	0.565(±0.078)	0.553(±0.050)	
12 → 5	0.638(±0.101)	0.844(±0.01)	0.844(±0.01)	0.84(±0.007)	0.829(±0.03)	0.792(±0.06)	0.871(±0.055)	0.835(±0.062)	0.83(±0.016)	
16 → 1	0.699(±0.077)	0.729(±0.023)	0.729(±0.023)	0.726(±0.028)	0.659(±0.05)	0.626(±0.117)	0.682(±0.029)	0.706(±0.017)	0.764(±0.084)	
7 → 18	0.647(±0.045)	0.758(±0.017)	0.758(±0.017)	0.76(±0.016)	0.704(±0.005)	0.73(±0.004)	0.754(±0.041)	0.773(±0.012)	0.794(±0.035)	
9 → 14	0.703(±0.041)	0.865(±0.018)	0.865(±0.018)	0.862(±0.028)	0.806(±0.032)	0.81(±0.053)	0.759(±0.041)	0.862(±0.018)	0.832(±0.018)	
Avg.	0.652(±0.058)	0.739(±0.027)	0.739(±0.027)	0.739(±0.032)	0.728(±0.03)	0.697(±0.069)	0.642(±0.038)	0.748(±0.025)	0.759(±0.041)	

DANN										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
0 → 11	0.551(±0.018)	0.602(±0.058)	0.602(±0.058)	0.602(±0.052)	0.672(±0.008)	0.594(±0.081)	0.495(±0.114)	0.622(±0.04)	0.574(±0.008)	
12 → 5	0.645(±0.056)	0.801(±0.014)	0.801(±0.014)	0.792(±0.01)	0.798(±0.006)	0.826(±0.011)	0.81(±0.051)	0.784(±0.01)	0.81(±0.054)	
16 → 1	0.57(±0.136)	0.76(±0.002)	0.676(±0.002)	0.67(±0.008)	0.593(±0.025)	0.61(±0.055)	0.651(±0.055)	0.661(±0.008)	0.747(±0.049)	
7 → 18	0.693(±0.037)	0.725(±0.006)	0.725(±0.006)	0.725(±0.014)	0.686(±0.013)	0.703(±0.034)	0.703(±0.034)	0.732(±0.011)	0.738(±0.018)	
9 → 14	0.746(±0.021)	0.814(±0.002)	0.814(±0.002)	0.815(±0.015)	0.822(±0.012)	0.799(±0.039)	0.686(±0.085)	0.822(±0.01)	0.81(±0.01)	
Avg.	0.641(±0.057)	0.723(±0.016)	0.723(±0.016)	0.721(±0.02)	0.714(±0.015)	0.687(±0.052)	0.644(±0.068)	0.724(±0.018)	0.738(±0.028)	

DSAN										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
0 → 11	0.547(±0.012)	0.539(±0.035)	0.539(±0.035)	0.527(±0.063)	0.668(±0.024)	0.59(±0.055)	0.311(±0.026)	0.629(±0.051)	0.603(±0.024)	
12 → 5	0.717(±0.086)	0.84(±0.02)	0.84(±0.02)	0.841(±0.022)	0.837(±0.026)	0.822(±0.028)	0.699(±0.061)	0.842(±0.018)	0.855(±0.007)	
16 → 1	0.697(±0.06)	0.755(±0.031)	0.755(±0.031)	0.722(±0.02)	0.612(±0.026)	0.617(±0.118)	0.617(±0.118)	0.721(±0.014)	0.742(±0.031)	
7 → 18	0.684(±0.048)	0.734(±0.029)	0.734(±0.029)	0.741(±0.01)	0.71(±0.067)	0.749(±0.02)	0.745(±0.025)	0.764(±0.002)	0.768(±0.059)	
9 → 14	0.682(±0.04)	0.832(±0.024)	0.832(±0.024)	0.827(±0.018)	0.815(±0.018)	0.784(±0.046)	0.576(±0.154)	0.828(±0.021)	0.823(±0.024)	
Avg.	0.653(±0.049)	0.74(±0.028)	0.74(±0.028)	0.732(±0.027)	0.728(±0.032)	0.712(±0.053)	0.589(±0.077)	0.757(±0.021)	0.758(±0.029)	

Table 23: Mean and standard deviation (after \pm) of target classification accuracy on UCI-HAR (Part 1) over 3 repetitions with different random initialization of model weights.

HoMM										
Task	Heuristic				Theoretical error guarantees					
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
12 → 16	0.674(± 0.012)	0.667(± 0.036)	0.667(± 0.036)	0.667(± 0.036)	0.668(± 0.0)	0.674(± 0.012)	0.674(± 0.012)	0.688(± 0.0)	0.715(± 0.012)	
2 → 11	1.0(± 0.0)	0.938(± 0.031)	0.99(± 0.018)	1.0(± 0.0)	1.0(± 0.0)	1.0(± 0.0)				
6 → 23	0.854(± 0.036)	0.896(± 0.0)	0.896(± 0.0)	0.896(± 0.0)	0.889(± 0.012)	0.854(± 0.036)	0.854(± 0.036)	0.903(± 0.012)	0.938(± 0.0)	
7 → 13	0.889(± 0.052)	0.91(± 0.024)	0.91(± 0.024)	0.91(± 0.024)	0.889(± 0.052)	0.889(± 0.052)	0.889(± 0.052)	0.91(± 0.024)	0.944(± 0.024)	
9 → 18	0.493(± 0.032)	0.618(± 0.105)	0.618(± 0.105)	0.618(± 0.105)	0.514(± 0.188)	0.639(± 0.043)	0.583(± 0.116)	0.632(± 0.079)	0.715(± 0.024)	
Avg.	0.782(± 0.026)	0.818(± 0.033)	0.818(± 0.033)	0.818(± 0.033)	0.783(± 0.057)	0.809(± 0.032)	0.8(± 0.043)	0.826(± 0.023)	0.862(± 0.012)	

AdvSKM										
Task	Heuristic				Theoretical error guarantees					
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
12 → 16	0.59(± 0.103)	0.681(± 0.012)	0.681(± 0.012)	0.681(± 0.012)	0.708(± 0.021)	0.59(± 0.103)	0.59(± 0.103)	0.688(± 0.0)	0.722(± 0.012)	
2 → 11	0.079(± 0.018)	1.0(± 0.0)	1.0(± 0.0)	1.0(± 0.0)	0.854(± 0.065)	0.896(± 0.01)	0.917(± 0.118)	1.0(± 0.0)	1.0(± 0.0)	
6 → 23	0.826(± 0.084)	0.896(± 0.0)	0.896(± 0.0)	0.896(± 0.0)	0.889(± 0.012)	0.826(± 0.084)	0.826(± 0.084)	0.896(± 0.0)	0.896(± 0.0)	
7 → 13	0.833(± 0.036)	0.91(± 0.024)	0.91(± 0.024)	0.917(± 0.036)	0.896(± 0.036)	0.833(± 0.036)	0.833(± 0.036)	0.903(± 0.024)	0.924(± 0.012)	
9 → 18	0.389(± 0.151)	0.514(± 0.079)	0.514(± 0.079)	0.556(± 0.064)	0.493(± 0.064)	0.389(± 0.151)	0.389(± 0.151)	0.514(± 0.024)	0.597(± 0.052)	
Avg.	0.724(± 0.078)	0.8(± 0.023)	0.8(± 0.023)	0.81(± 0.022)	0.768(± 0.04)	0.707(± 0.095)	0.711(± 0.098)	0.8(± 0.01)	0.828(± 0.015)	

DIRT										
Task	Heuristic				Theoretical error guarantees					
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
12 → 16	0.667(± 0.075)	0.778(± 0.012)	0.778(± 0.012)	0.75(± 0.0)	0.736(± 0.012)	0.667(± 0.075)	0.667(± 0.075)	0.764(± 0.032)	0.819(± 0.064)	
2 → 11	1.0(± 0.0)	0.854(± 0.079)	1.0(± 0.0)	1.0(± 0.0)	1.0(± 0.0)	1.0(± 0.0)				
6 → 23	0.903(± 0.084)	0.938(± 0.0)	0.938(± 0.0)	0.938(± 0.0)	0.847(± 0.06)	0.847(± 0.06)	0.847(± 0.06)	0.938(± 0.0)	0.938(± 0.0)	
7 → 13	0.833(± 0.0)	0.938(± 0.0)	0.958(± 0.0)	0.958(± 0.0)	0.924(± 0.06)	0.854(± 0.036)	0.861(± 0.032)	0.958(± 0.0)	0.958(± 0.0)	
9 → 18	0.514(± 0.024)	0.861(± 0.122)	0.861(± 0.122)	0.806(± 0.032)	0.417(± 0.075)	0.667(± 0.225)	0.667(± 0.225)	0.84(± 0.048)	0.986(± 0.012)	
Avg.	0.783(± 0.037)	0.907(± 0.027)	0.907(± 0.027)	0.89(± 0.006)	0.756(± 0.057)	0.807(± 0.079)	0.808(± 0.078)	0.9(± 0.016)	0.94(± 0.015)	

DDC										
Task	Heuristic				Theoretical error guarantees					
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
12 → 16	0.681(± 0.024)	0.694(± 0.012)	0.681(± 0.012)	0.674(± 0.012)	0.681(± 0.024)	0.664(± 0.08)	0.681(± 0.012)	0.881(± 0.012)	0.715(± 0.073)	
2 → 11	1.0(± 0.0)	0.771(± 0.036)	0.812(± 0.272)	0.812(± 0.272)	1.0(± 0.0)	1.0(± 0.0)				
6 → 23	0.889(± 0.012)	0.896(± 0.0)	0.896(± 0.0)	0.896(± 0.0)	0.882(± 0.012)	0.889(± 0.021)	0.896(± 0.021)	0.899(± 0.0)	0.899(± 0.0)	
7 → 13	0.868(± 0.012)	0.896(± 0.036)	0.896(± 0.036)	0.91(± 0.032)	0.903(± 0.024)	0.868(± 0.012)	0.868(± 0.012)	0.903(± 0.024)	0.917(± 0.021)	
9 → 18	0.514(± 0.043)	0.562(± 0.091)	0.562(± 0.091)	0.569(± 0.115)	0.583(± 0.165)	0.444(± 0.103)	0.5(± 0.127)	0.542(± 0.091)	0.59(± 0.189)	
Avg.	0.79(± 0.018)	0.807(± 0.028)	0.807(± 0.028)	0.81(± 0.032)	0.756(± 0.085)	0.724(± 0.073)	0.734(± 0.107)	0.804(± 0.025)	0.822(± 0.057)	

CMD										
Task	Heuristic				Theoretical error guarantees					
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
12 → 16	0.694(± 0.012)	0.729(± 0.0)	0.729(± 0.0)	0.722(± 0.012)	0.688(± 0.0)	0.694(± 0.012)	0.694(± 0.012)	0.729(± 0.0)	0.75(± 0.036)	
2 → 11	1.0(± 0.0)	0.771(± 0.036)	0.812(± 0.272)	0.812(± 0.272)	1.0(± 0.0)	1.0(± 0.0)				
6 → 23	0.896(± 0.042)	0.938(± 0.0)	0.938(± 0.0)	0.931(± 0.012)	0.889(± 0.012)	0.903(± 0.032)	0.906(± 0.042)	0.938(± 0.0)	0.938(± 0.0)	
7 → 13	0.882(± 0.043)	0.903(± 0.012)	0.903(± 0.012)	0.917(± 0.021)	0.91(± 0.024)	0.826(± 0.073)	0.826(± 0.073)	0.938(± 0.0)	0.944(± 0.012)	
9 → 18	0.521(± 0.095)	0.569(± 0.043)	0.569(± 0.043)	0.569(± 0.139)	0.5(± 0.104)	0.396(± 0.11)	0.535(± 0.064)	0.507(± 0.024)	0.521(± 0.091)	0.688(± 0.075)
Avg.	0.788(± 0.044)	0.849(± 0.009)	0.849(± 0.009)	0.839(± 0.016)	0.731(± 0.036)	0.804(± 0.107)	0.812(± 0.112)	0.842(± 0.015)	0.903(± 0.025)	

MMDA										
Task	Heuristic				Theoretical error guarantees					
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
12 → 16	0.674(± 0.012)	0.694(± 0.012)	0.694(± 0.012)	0.688(± 0.0)	0.681(± 0.012)	0.674(± 0.012)	0.674(± 0.012)	0.688(± 0		

Table 24: Mean and standard deviation (after \pm) of target classification accuracy on UCI-HAR (Part 2) over 3 repetitions with different random initialization of model weights.

CoDATS										
Task	SO	Heuristic				Theoretical error guarantees				
		TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
12 → 16	0.681(±0.024)	0.701(±0.024)	0.701(±0.024)	0.688(±0.0)	0.694(±0.024)	0.681(±0.024)	0.694(±0.024)	0.694(±0.012)	0.708(±0.021)	
2 → 11	1.0(±0.0)	1.0(±0.0)	1.0(±0.0)	1.0(±0.0)	0.865(±0.11)	0.865(±0.1)	0.865(±0.1)	1.0(±0.0)	1.0(±0.0)	
6 → 23	0.84(±0.032)	0.889(±0.032)	0.889(±0.032)	0.889(±0.032)	0.882(±0.064)	0.84(±0.048)	0.882(±0.083)	0.896(±0.083)	0.924(±0.012)	0.951(±0.043)
7 → 13	0.75(±0.217)	0.944(±0.012)	0.944(±0.012)	0.944(±0.012)	0.917(±0.036)	0.854(±0.036)	0.75(±0.217)	0.938(±0.0)	0.951(±0.012)	
9 → 18	0.528(±0.032)	0.625(±0.062)	0.625(±0.062)	0.639(±0.03)	0.569(±0.305)	0.764(±0.12)	0.764(±0.12)	0.674(±0.115)	0.826(±0.012)	
Avg.	0.76(±0.061)	0.832(±0.026)	0.832(±0.026)	0.832(±0.017)	0.785(±0.108)	0.801(±0.066)	0.794(±0.109)	0.846(±0.028)	0.888(±0.018)	

Deep-Coral										
Task	SO	Heuristic				Theoretical error guarantees				
		TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
12 → 16	0.681(±0.012)	0.681(±0.012)	0.681(±0.012)	0.674(±0.012)	0.681(±0.012)	0.681(±0.012)	0.639(±0.084)	0.688(±0.0)	0.688(±0.0)	
2 → 11	0.869(±0.051)	1.0(±0.0)	1.0(±0.0)	1.0(±0.0)	0.877(±0.094)	0.877(±0.083)	0.885(±0.095)	1.0(±0.0)	1.0(±0.0)	
6 → 23	0.826(±0.084)	0.882(±0.012)	0.882(±0.012)	0.889(±0.012)	0.806(±0.0)	0.826(±0.084)	0.832(±0.01)	0.896(±0.0)	0.903(±0.043)	
7 → 13	0.917(±0.036)	0.91(±0.032)	0.91(±0.032)	0.91(±0.032)	0.806(±0.021)	0.917(±0.036)	0.917(±0.036)	0.903(±0.024)	0.931(±0.012)	
9 → 18	0.556(±0.087)	0.528(±0.043)	0.528(±0.043)	0.569(±0.115)	0.507(±0.272)	0.542(±0.144)	0.59(±0.032)	0.556(±0.064)	0.59(±0.221)	
Avg.	0.79(±0.055)	0.8(±0.02)	0.8(±0.02)	0.808(±0.034)	0.771(±0.08)	0.768(±0.072)	0.773(±0.068)	0.808(±0.018)	0.822(±0.055)	

CDAN										
Task	SO	Heuristic				Theoretical error guarantees				
		TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
12 → 16	0.674(±0.033)	0.729(±0.012)	0.729(±0.012)	0.729(±0.012)	0.694(±0.012)	0.729(±0.012)	0.646(±0.055)	0.714(±0.024)	0.729(±0.036)	
2 → 11	0.69(±0.018)	1.0(±0.0)	1.0(±0.0)	1.0(±0.0)	0.809(±0.095)	0.792(±0.028)	0.802(±0.029)	1.0(±0.0)	1.0(±0.0)	
6 → 23	0.854(±0.042)	0.92(±0.02)	0.924(±0.02)	0.91(±0.02)	0.91(±0.024)	0.847(±0.032)	0.708(±0.205)	0.924(±0.024)	0.924(±0.024)	
7 → 13	0.826(±0.130)	0.951(±0.012)	0.951(±0.012)	0.951(±0.012)	0.944(±0.012)	0.896(±0.021)	0.812(±0.108)	0.951(±0.012)	0.958(±0.0)	
9 → 18	0.438(±0.055)	0.618(±0.061)	0.618(±0.064)	0.618(±0.064)	0.597(±0.098)	0.694(±0.126)	0.465(±0.087)	0.639(±0.048)	0.729(±0.083)	
Avg.	0.756(±0.059)	0.843(±0.022)	0.843(±0.022)	0.84(±0.022)	0.802(±0.042)	0.781(±0.1)	0.687(±0.167)	0.846(±0.022)	0.868(±0.029)	

DANN										
Task	SO	Heuristic				Theoretical error guarantees				
		TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
12 → 16	0.688(±0.021)	0.701(±0.024)	0.701(±0.024)	0.701(±0.024)	0.715(±0.012)	0.701(±0.012)	0.701(±0.012)	0.729(±0.0)	0.722(±0.0)	
2 → 11	0.69(±0.018)	1.0(±0.0)	1.0(±0.0)	1.0(±0.0)	0.809(±0.095)	0.792(±0.028)	0.802(±0.029)	1.0(±0.0)	1.0(±0.0)	
6 → 23	0.882(±0.024)	0.917(±0.021)	0.917(±0.021)	0.917(±0.021)	0.898(±0.032)	0.888(±0.024)	0.889(±0.024)	0.931(±0.012)	0.938(±0.036)	
7 → 13	0.806(±0.127)	0.944(±0.012)	0.944(±0.012)	0.944(±0.012)	0.938(±0.021)	0.903(±0.024)	0.944(±0.048)	0.944(±0.012)	0.965(±0.012)	
9 → 18	0.403(±0.12)	0.713(±0.012)	0.715(±0.012)	0.701(±0.067)	0.729(±0.127)	0.583(±0.111)	0.583(±0.111)	0.639(±0.103)	0.674(±0.127)	
Avg.	0.756(±0.058)	0.856(±0.014)	0.856(±0.014)	0.856(±0.023)	0.8(±0.066)	0.763(±0.064)	0.78(±0.08)	0.849(±0.025)	0.864(±0.035)	

DSAN										
Task	SO	Heuristic				Theoretical error guarantees				
		TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
12 → 16	0.681(±0.012)	0.688(±0.021)	0.688(±0.021)	0.715(±0.024)	0.694(±0.012)	0.681(±0.012)	0.556(±0.229)	0.715(±0.024)	0.736(±0.012)	
2 → 11	1.0(±0.0)	1.0(±0.0)	1.0(±0.0)	1.0(±0.0)	0.969(±0.0)	1.0(±0.0)	1.0(±0.0)	1.0(±0.0)	1.0(±0.0)	
6 → 23	0.764(±0.094)	0.938(±0.0)	0.938(±0.0)	0.917(±0.021)	0.847(±0.052)	0.785(±0.103)	0.785(±0.103)	0.931(±0.012)	0.938(±0.0)	
7 → 13	0.854(±0.036)	0.951(±0.012)	0.951(±0.012)	0.951(±0.012)	0.854(±0.036)	0.854(±0.036)	0.854(±0.036)	0.951(±0.012)	0.958(±0.0)	
9 → 18	0.514(±0.032)	0.639(±0.079)	0.639(±0.079)	0.688(±0.091)	0.382(±0.012)	0.556(±0.067)	0.528(±0.067)	0.694(±0.032)	0.743(±0.087)	
Avg.	0.762(±0.035)	0.843(±0.022)	0.843(±0.022)	0.854(±0.03)	0.749(±0.023)	0.775(±0.044)	0.744(±0.087)	0.858(±0.016)	0.875(±0.02)	

Table 25: Mean and standard deviation (after \pm) of target classification accuracy on HHAR (Part 1) over 3 repetitions with different random initialization of model weights.

HoMM										
Task	Heuristic				Theoretical error guarantees					
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
0 → 6	0.710 \pm 0.041	0.737 \pm 0.019	0.737 \pm 0.010	0.735 \pm 0.024	0.635 \pm 0.122	0.704 \pm 0.011	0.703 \pm 0.013	0.732 \pm 0.01	0.733 \pm 0.019	
1 → 6	0.788 \pm 0.111	0.872 \pm 0.011	0.871 \pm 0.011	0.869 \pm 0.013	0.788 \pm 0.163	0.749 \pm 0.06	0.785 \pm 0.106	0.879 \pm 0.011	0.914 \pm 0.011	
2 → 7	0.528 \pm 0.103	0.457 \pm 0.007	0.457 \pm 0.007	0.461 \pm 0.007	0.469 \pm 0.051	0.497 \pm 0.105	0.516 \pm 0.087	0.455 \pm 0.015	0.546 \pm 0.071	
3 → 8	0.797 \pm 0.007	0.818 \pm 0.002	0.818 \pm 0.002	0.816 \pm 0.007	0.812 \pm 0.014	0.805 \pm 0.01	0.805 \pm 0.01	0.818 \pm 0.005	0.831 \pm 0.022	
4 → 5	0.861 \pm 0.022	0.914 \pm 0.007	0.914 \pm 0.007	0.911 \pm 0.005	0.798 \pm 0.164	0.844 \pm 0.016	0.855 \pm 0.01	0.911 \pm 0.005	0.94 \pm 0.04	
Avg.	0.739 \pm 0.057	0.759 \pm 0.009	0.759 \pm 0.000	0.759 \pm 0.011	0.7 \pm 0.103	0.72 \pm 0.041	0.733 \pm 0.045	0.759 \pm 0.009	0.793 \pm 0.032	

AdvSKM										
Task	Heuristic				Theoretical error guarantees					
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
0 → 6	0.661 \pm 0.035	0.722 \pm 0.009	0.722 \pm 0.009	0.725 \pm 0.007	0.620 \pm 0.076	0.699 \pm 0.013	0.699 \pm 0.013	0.748 \pm 0.009	0.731 \pm 0.013	
1 → 6	0.821 \pm 0.046	0.849 \pm 0.002	0.849 \pm 0.002	0.851 \pm 0.006	0.604 \pm 0.045	0.806 \pm 0.026	0.806 \pm 0.026	0.857 \pm 0.006	0.838 \pm 0.018	
2 → 7	0.455 \pm 0.007	0.473 \pm 0.031	0.473 \pm 0.031	0.484 \pm 0.03	0.521 \pm 0.062	0.497 \pm 0.067	0.497 \pm 0.067	0.49 \pm 0.014	0.574 \pm 0.112	
3 → 8	0.797 \pm 0.011	0.790 \pm 0.008	0.790 \pm 0.008	0.803 \pm 0.005	0.805 \pm 0.01	0.799 \pm 0.005	0.788 \pm 0.023	0.811 \pm 0.002	0.818 \pm 0.012	
4 → 5	0.862 \pm 0.055	0.866 \pm 0.013	0.866 \pm 0.013	0.872 \pm 0.008	0.809 \pm 0.094	0.85 \pm 0.025	0.85 \pm 0.025	0.884 \pm 0.005	0.895 \pm 0.014	
Avg.	0.718 \pm 0.044	0.742 \pm 0.013	0.742 \pm 0.013	0.748 \pm 0.011	0.676 \pm 0.058	0.73 \pm 0.027	0.728 \pm 0.031	0.752 \pm 0.007	0.771 \pm 0.034	

DIRT										
Task	Heuristic				Theoretical error guarantees					
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
0 → 6	0.708 \pm 0.011	0.571 \pm 0.070	0.571 \pm 0.070	0.623 \pm 0.127	0.620 \pm 0.085	0.708 \pm 0.011	0.708 \pm 0.011	0.770 \pm 0.011	0.729 \pm 0.011	
1 → 6	0.750 \pm 0.058	0.938 \pm 0.001	0.938 \pm 0.004	0.938 \pm 0.004	0.904 \pm 0.058	0.814 \pm 0.056	0.756 \pm 0.058	0.938 \pm 0.01	0.942 \pm 0.004	
2 → 7	0.531 \pm 0.057	0.622 \pm 0.091	0.622 \pm 0.091	0.621 \pm 0.09	0.658 \pm 0.093	0.548 \pm 0.049	0.548 \pm 0.049	0.531 \pm 0.025	0.688 \pm 0.004	
3 → 8	0.807 \pm 0.008	0.846 \pm 0.022	0.846 \pm 0.022	0.837 \pm 0.015	0.857 \pm 0.087	0.807 \pm 0.008	0.807 \pm 0.008	0.848 \pm 0.007	0.911 \pm 0.07	
4 → 5	0.839 \pm 0.016	0.984 \pm 0.004	0.984 \pm 0.004	0.984 \pm 0.004	0.93 \pm 0.061	0.839 \pm 0.016	0.878 \pm 0.074	0.984 \pm 0.004	0.984 \pm 0.002	
Avg.	0.728 \pm 0.03	0.792 \pm 0.04	0.792 \pm 0.04	0.803 \pm 0.05	0.796 \pm 0.064	0.743 \pm 0.028	0.739 \pm 0.04	0.816 \pm 0.009	0.853 \pm 0.018	

DDC										
Task	Heuristic				Theoretical error guarantees					
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
0 → 6	0.575 \pm 0.080	0.699 \pm 0.017	0.699 \pm 0.017	0.672 \pm 0.009	0.632 \pm 0.038	0.646 \pm 0.015	0.6 \pm 0.084	0.697 \pm 0.013	0.651 \pm 0.013	
1 → 6	0.858 \pm 0.014	0.888 \pm 0.007	0.888 \pm 0.007	0.894 \pm 0.013	0.884 \pm 0.010	0.885 \pm 0.007	0.884 \pm 0.007	0.899 \pm 0.016	0.899 \pm 0.016	
2 → 7	0.487 \pm 0.016	0.457 \pm 0.03	0.457 \pm 0.03	0.458 \pm 0.029	0.479 \pm 0.028	0.438 \pm 0.054	0.438 \pm 0.054	0.439 \pm 0.04	0.533 \pm 0.107	
3 → 8	0.815 \pm 0.019	0.805 \pm 0.014	0.805 \pm 0.014	0.807 \pm 0.016	0.797 \pm 0.022	0.827 \pm 0.025	0.821 \pm 0.02	0.818 \pm 0.008	0.822 \pm 0.02	
4 → 5	0.831 \pm 0.048	0.908 \pm 0.011	0.908 \pm 0.011	0.911 \pm 0.01	0.772 \pm 0.085	0.792 \pm 0.02	0.801 \pm 0.012	0.905 \pm 0.008	0.888 \pm 0.01	
Avg.	0.716 \pm 0.036	0.75 \pm 0.016	0.75 \pm 0.016	0.748 \pm 0.015	0.717 \pm 0.038	0.711 \pm 0.036	0.705 \pm 0.047	0.748 \pm 0.017	0.758 \pm 0.033	

CMD										
Task	Heuristic				Theoretical error guarantees					
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
0 → 6	0.703 \pm 0.04	0.643 \pm 0.046	0.643 \pm 0.046	0.679 \pm 0.054	0.672 \pm 0.103	0.743 \pm 0.021	0.743 \pm 0.021	0.693 \pm 0.012	0.724 \pm 0.021	
1 → 6	0.861 \pm 0.035	0.915 \pm 0.009	0.915 \pm 0.009	0.912 \pm 0.0	0.826 \pm 0.103	0.899 \pm 0.03	0.843 \pm 0.156	0.907 \pm 0.006	0.925 \pm 0.004	
2 → 7	0.573 \pm 0.018	0.494 \pm 0.011	0.494 \pm 0.011	0.499 \pm 0.023	0.51 \pm 0.066	0.577 \pm 0.011	0.557 \pm 0.039	0.482 \pm 0.08	0.603 \pm 0.016	
3 → 8	0.399 \pm 0.016	0.816 \pm 0.0	0.816 \pm 0.0	0.816 \pm 0.0	0.791 \pm 0.04	0.799 \pm 0.016	0.738 \pm 0.109	0.811 \pm 0.011	0.822 \pm 0.01	
4 → 5	0.806 \pm 0.024	0.952 \pm 0.016	0.952 \pm 0.016	0.931 \pm 0.007	0.879 \pm 0.054	0.854 \pm 0.063	0.352 \pm 0.051	0.839 \pm 0.002	0.961 \pm 0.024	
Avg.	0.748 \pm 0.026	0.764 \pm 0.016	0.764 \pm 0.016	0.767 \pm 0.017	0.737 \pm 0.066	0.775 \pm 0.028	0.643 \pm 0.075	0.766 \pm 0.008	0.807 \pm 0.015	

MMDA										
Task	Heuristic				Theoretical error guarantees					
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
0 → 6	0.706 \pm 0.042	0.737 \pm 0.021	0.737 \pm 0.021	0.737 \pm 0.018	0.704 \pm 0.037	0.699 \pm 0.055	0.725 \pm 0.018	0.746 \pm 0.008	0.732 \pm 0.008	
1 → 6	0.792 \pm 0.098	0.91 \pm 0.002	0.91 \pm 0.002	0.91 \pm 0.002	0.742 \pm 0.147	0.732 \pm 0.009	0.722 \pm 0.009	0.897 \pm 0.002	0.914 \pm 0.028	
2 → 7	0.552 \pm 0.073	0.481 \pm 0.003	0.481 \pm 0.003	0.482 \pm 0.004	0.515 \pm 0.058	0.496 \pm 0.027	0.543 \pm 0.055	0.488 \pm 0.01	0.552 \pm 0.073	
3 → 8	0.79 \pm 0.016	0.862 \pm 0.026	0.862 \pm 0.026	0.841 \pm 0.01	0.794 \pm 0.022	0.802 \pm 0.018	0.802 \pm 0			

Table 26: Mean and standard deviation (after \pm) of target classification accuracy on HHAR (Part 2) over 3 repetitions with different random initialization of model weights.

CoDATS										
Task	SO	Heuristic				Theoretical error guarantees				
		TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
0 → 6	0.641(±0.096)	0.497(±0.005)	0.497(±0.005)	0.499(±0.006)	0.582(±0.122)	0.689(±0.069)	0.606(±0.106)	0.718(±0.017)	0.735(±0.01)	
1 → 6	0.775(±0.091)	0.949(±0.0)	0.946(±0.0)	0.946(±0.0)	0.682(±0.076)	0.896(±0.069)	0.896(±0.069)	0.939(±0.006)	0.947(±0.009)	
2 → 7	0.527(±0.076)	0.473(±0.012)	0.473(±0.012)	0.475(±0.01)	0.532(±0.052)	0.472(±0.019)	0.555(±0.069)	0.472(±0.05)	0.558(±0.041)	
3 → 8	0.783(±0.015)	0.971(±0.018)	0.971(±0.018)	0.969(±0.018)	0.97(±0.025)	0.789(±0.01)	0.789(±0.01)	0.96(±0.03)	0.987(±0.002)	
4 → 5	0.827(±0.039)	0.971(±0.013)	0.971(±0.013)	0.978(±0.002)	0.841(±0.1)	0.849(±0.074)	0.849(±0.074)	0.973(±0.004)	0.979(±0.006)	
Avg.	0.71(±0.063)	0.772(±0.009)	0.772(±0.009)	0.773(±0.007)	0.722(±0.075)	0.739(±0.048)	0.739(±0.066)	0.812(±0.012)	0.841(±0.014)	

Deep-Coral										
Task	SO	Heuristic				Theoretical error guarantees				
		TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
0 → 6	0.692(±0.026)	0.731(±0.028)	0.731(±0.028)	0.736(±0.024)	0.632(±0.024)	0.703(±0.038)	0.7(±0.038)	0.728(±0.019)	0.703(±0.024)	
1 → 6	0.809(±0.016)	0.809(±0.009)	0.809(±0.009)	0.809(±0.004)	0.829(±0.104)	0.894(±0.017)	0.904(±0.017)	0.889(±0.009)	0.911(±0.017)	
2 → 7	0.509(±0.097)	0.454(±0.02)	0.454(±0.02)	0.457(±0.025)	0.51(±0.057)	0.499(±0.079)	0.539(±0.076)	0.46(±0.012)	0.565(±0.117)	
3 → 8	0.798(±0.016)	0.801(±0.004)	0.801(±0.004)	0.803(±0.006)	0.793(±0.024)	0.799(±0.013)	0.802(±0.016)	0.812(±0.0)	0.822(±0.01)	
4 → 5	0.862(±0.036)	0.932(±0.025)	0.932(±0.025)	0.938(±0.022)	0.648(±0.108)	0.906(±0.027)	0.887(±0.035)	0.936(±0.016)	0.96(±0.01)	
Avg.	0.745(±0.039)	0.762(±0.017)	0.762(±0.017)	0.766(±0.016)	0.681(±0.063)	0.754(±0.035)	0.758(±0.036)	0.764(±0.011)	0.792(±0.036)	

CDAN										
Task	SO	Heuristic				Theoretical error guarantees				
		TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
0 → 6	0.517(±0.004)	0.483(±0.008)	0.483(±0.008)	0.481(±0.009)	0.659(±0.092)	0.692(±0.083)	0.692(±0.083)	0.7(±0.016)	0.718(±0.016)	
1 → 6	0.742(±0.039)	0.829(±0.006)	0.829(±0.006)	0.828(±0.01)	0.906(±0.021)	0.942(±0.011)	0.944(±0.013)	0.938(±0.01)	0.949(±0.01)	
2 → 7	0.554(±0.031)	0.528(±0.05)	0.522(±0.05)	0.526(±0.059)	0.58(±0.077)	0.501(±0.068)	0.561(±0.068)	0.531(±0.06)	0.624(±0.014)	
3 → 8	0.77(±0.069)	0.872(±0.09)	0.872(±0.09)	0.871(±0.091)	0.812(±0.007)	0.801(±0.004)	0.801(±0.004)	0.874(±0.076)	0.987(±0.006)	
4 → 5	0.859(±0.014)	0.978(±0.005)	0.978(±0.005)	0.979(±0.002)	0.828(±0.106)	0.875(±0.087)	0.875(±0.087)	0.98(±0.0)	0.982(±0.002)	
Avg.	0.728(±0.042)	0.758(±0.032)	0.758(±0.032)	0.764(±0.034)	0.765(±0.046)	0.774(±0.051)	0.775(±0.051)	0.816(±0.031)	0.851(±0.008)	

DANN										
Task	SO	Heuristic				Theoretical error guarantees				
		TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
0 → 6	0.704(±0.037)	0.489(±0.006)	0.489(±0.006)	0.49(±0.006)	0.585(±0.092)	0.724(±0.016)	0.724(±0.016)	0.722(±0.071)	0.711(±0.016)	
1 → 6	0.833(±0.023)	0.94(±0.002)	0.94(±0.002)	0.94(±0.004)	0.58(±0.142)	0.906(±0.004)	0.906(±0.004)	0.94(±0.002)	0.949(±0.002)	
2 → 7	0.58(±0.023)	0.496(±0.004)	0.496(±0.004)	0.493(±0.01)	0.557(±0.061)	0.618(±0.029)	0.592(±0.022)	0.49(±0.003)	0.633(±0.025)	
3 → 8	0.809(±0.02)	0.897(±0.015)	0.966(±0.015)	0.966(±0.015)	0.823(±0.069)	0.796(±0.03)	0.796(±0.03)	0.964(±0.006)	0.983(±0.002)	
4 → 5	0.846(±0.022)	0.98(±0.0)	0.98(±0.0)	0.98(±0.0)	0.779(±0.138)	0.922(±0.078)	0.922(±0.078)	0.98(±0.0)	0.98(±0.0)	
Avg.	0.757(±0.027)	0.774(±0.006)	0.774(±0.006)	0.773(±0.007)	0.722(±0.1)	0.798(±0.031)	0.793(±0.031)	0.818(±0.016)	0.85(±0.009)	

DSAN										
Task	SO	Heuristic				Theoretical error guarantees				
		TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
0 → 6	0.653(±0.033)	0.607(±0.057)	0.607(±0.057)	0.635(±0.119)	0.597(±0.075)	0.572(±0.103)	0.471(±0.259)	0.747(±0.027)	0.664(±0.027)	
1 → 6	0.806(±0.083)	0.922(±0.002)	0.922(±0.002)	0.925(±0.004)	0.821(±0.105)	0.908(±0.065)	0.908(±0.065)	0.929(±0.0)	0.94(±0.005)	
2 → 7	0.49(±0.053)	0.496(±0.004)	0.496(±0.004)	0.494(±0.003)	0.485(±0.005)	0.482(±0.012)	0.49(±0.003)	0.496(±0.0)	0.586(±0.083)	
3 → 8	0.797(±0.016)	0.979(±0.002)	0.979(±0.002)	0.977(±0.004)	0.855(±0.097)	0.816(±0.01)	0.639(±0.454)	0.971(±0.005)	0.982(±0.01)	
4 → 5	0.861(±0.043)	0.98(±0.008)	0.98(±0.008)	0.979(±0.006)	0.863(±0.085)	0.928(±0.08)	0.471(±0.438)	0.98(±0.0)	0.982(±0.005)	
Avg.	0.721(±0.046)	0.797(±0.015)	0.797(±0.015)	0.802(±0.027)	0.724(±0.073)	0.741(±0.054)	0.596(±0.244)	0.825(±0.006)	0.831(±0.026)	

Table 27: Mean and standard deviation (after \pm) of target classification accuracy on WISDM (Part 1) over 3 repetitions with different random initialization of model weights.

HoMM										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
18 → 23	0.733(±0.058)	0.744(±0.038)	0.744(±0.038)	0.756(±0.051)	0.733(±0.033)	0.733(±0.058)	0.733(±0.058)	0.711(±0.019)	0.767(±0.033)	
20 → 30	0.853(±0.022)	0.814(±0.022)	0.814(±0.022)	0.801(±0.011)	0.827(±0.0)	0.853(±0.022)	0.853(±0.022)	0.814(±0.011)	0.853(±0.022)	
35 → 31	0.579(±0.055)	0.771(±0.014)	0.771(±0.014)	0.77(±0.014)	0.722(±0.05)	0.579(±0.055)	0.579(±0.055)	0.746(±0.027)	0.77(±0.014)	
6 → 19	0.889(±0.009)	0.864(±0.052)	0.864(±0.052)	0.823(±0.035)	0.884(±0.017)	0.889(±0.009)	0.889(±0.009)	0.823(±0.035)	0.894(±0.026)	
7 → 18	0.711(±0.066)	0.497(±0.011)	0.497(±0.011)	0.547(±0.068)	0.711(±0.029)	0.711(±0.066)	0.648(±0.123)	0.547(±0.0)	0.711(±0.066)	
Avg.	0.753(±0.042)	0.738(±0.028)	0.738(±0.028)	0.739(±0.036)	0.775(±0.026)	0.753(±0.042)	0.74(±0.053)	0.728(±0.019)	0.799(±0.032)	

AdvSKM										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
18 → 23	0.711(±0.019)	0.744(±0.038)	0.744(±0.038)	0.744(±0.038)	0.767(±0.033)	0.711(±0.019)	0.711(±0.019)	0.744(±0.019)	0.778(±0.006)	
20 → 30	0.872(±0.029)	0.885(±0.033)	0.885(±0.033)	0.885(±0.033)	0.872(±0.059)	0.872(±0.029)	0.872(±0.029)	0.853(±0.029)	0.885(±0.033)	
35 → 31	0.619(±0.041)	0.698(±0.027)	0.698(±0.027)	0.69(±0.082)	0.675(±0.096)	0.619(±0.041)	0.619(±0.041)	0.698(±0.027)	0.714(±0.024)	
6 → 19	0.818(±0.131)	0.894(±0.0)	0.894(±0.0)	0.869(±0.044)	0.833(±0.052)	0.818(±0.131)	0.818(±0.131)	0.874(±0.009)	0.894(±0.0)	
7 → 18	0.717(±0.05)	0.686(±0.058)	0.686(±0.058)	0.704(±0.039)	0.566(±0.068)	0.717(±0.05)	0.717(±0.05)	0.717(±0.019)	0.736(±0.086)	
Avg.	0.747(±0.054)	0.781(±0.031)	0.781(±0.031)	0.779(±0.047)	0.742(±0.062)	0.747(±0.054)	0.747(±0.054)	0.777(±0.021)	0.801(±0.048)	

DIRT										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
18 → 23	0.711(±0.038)	0.732(±0.033)	0.732(±0.033)	0.723(±0.0)	0.711(±0.019)	0.711(±0.038)	0.711(±0.0)	0.732(±0.0)	0.744(±0.019)	
20 → 30	0.872(±0.011)	0.846(±0.0)	0.846(±0.0)	0.846(±0.0)	0.891(±0.0)	0.872(±0.011)	0.859(±0.029)	0.84(±0.011)	0.917(±0.011)	
35 → 31	0.492(±0.036)	0.716(±0.036)	0.746(±0.036)	0.746(±0.036)	0.603(±0.1)	0.492(±0.036)	0.77(±0.069)	0.746(±0.014)	0.778(±0.027)	
6 → 19	0.879(±0.026)	0.833(±0.052)	0.833(±0.052)	0.833(±0.052)	0.869(±0.023)	0.879(±0.026)	0.879(±0.026)	0.838(±0.049)	0.889(±0.009)	
7 → 18	0.736(±0.082)	0.824(±0.011)	0.824(±0.011)	0.824(±0.011)	0.704(±0.093)	0.736(±0.082)	0.767(±0.109)	0.82(±0.0)	0.83(±0.0)	
Avg.	0.738(±0.039)	0.797(±0.027)	0.797(±0.027)	0.797(±0.02)	0.756(±0.057)	0.738(±0.039)	0.797(±0.054)	0.798(±0.015)	0.832(±0.013)	

DDC										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
18 → 23	0.7(±0.067)	0.732(±0.033)	0.732(±0.033)	0.727(±0.0)	0.7(±0.0)	0.7(±0.067)	0.7(±0.0)	0.744(±0.019)	0.767(±0.0)	
20 → 30	0.833(±0.022)	0.865(±0.027)	0.865(±0.027)	0.865(±0.027)	0.881(±0.029)	0.833(±0.022)	0.833(±0.022)	0.84(±0.011)	0.869(±0.019)	
35 → 31	0.541(±0.036)	0.724(±0.036)	0.724(±0.036)	0.724(±0.036)	0.635(±0.066)	0.541(±0.036)	0.541(±0.036)	0.646(±0.027)	0.73(±0.027)	
6 → 19	0.884(±0.017)	0.884(±0.017)	0.884(±0.017)	0.859(±0.032)	0.869(±0.044)	0.884(±0.017)	0.884(±0.017)	0.869(±0.024)	0.884(±0.017)	
7 → 18	0.748(±0.093)	0.692(±0.039)	0.692(±0.039)	0.704(±0.029)	0.648(±0.079)	0.748(±0.093)	0.748(±0.093)	0.711(±0.011)	0.748(±0.093)	
Avg.	0.741(±0.047)	0.779(±0.03)	0.779(±0.03)	0.787(±0.035)	0.737(±0.05)	0.741(±0.047)	0.741(±0.047)	0.782(±0.018)	0.803(±0.031)	

CMD										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
18 → 23	0.711(±0.038)	0.678(±0.019)	0.678(±0.019)	0.644(±0.069)	0.689(±0.107)	0.689(±0.107)	0.622(±0.069)	0.7(±0.0)	0.756(±0.038)	
20 → 30	0.853(±0.011)	0.859(±0.022)	0.859(±0.022)	0.897(±0.029)	0.885(±0.051)	0.853(±0.011)	0.859(±0.011)	0.904(±0.0)	0.91(±0.022)	
35 → 31	0.683(±0.014)	0.722(±0.014)	0.722(±0.014)	0.722(±0.014)	0.659(±0.014)	0.683(±0.014)	0.69(±0.024)	0.722(±0.027)	0.77(±0.014)	
6 → 19	0.742(±0.084)	0.798(±0.009)	0.798(±0.009)	0.803(±0.075)	0.833(±0.12)	0.783(±0.118)	0.682(±0.124)	0.798(±0.035)	0.813(±0.12)	
7 → 18	0.56(±0.044)	0.767(±0.058)	0.767(±0.058)	0.572(±0.061)	0.56(±0.044)	0.579(±0.066)	0.742(±0.029)	0.736(±0.05)		
Avg.	0.71(±0.038)	0.765(±0.024)	0.765(±0.024)	0.767(±0.037)	0.728(±0.071)	0.713(±0.059)	0.686(±0.059)	0.773(±0.018)	0.797(±0.05)	

MMDA										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
18 → 23	0.756(±0.051)	0.767(±0.033)	0.767(±0.033)	0.767(±0.033)	0.756(±0.019)	0.756(±0.051)	0.756(±0.051)	0.8(±0.0)	0.889(±0.019)	
20 → 30	0.872(±0.011)	0.878(±0.029)	0.878(±0.029)	0.872(±0.029)	0.814(±0.029)	0.872(±0.011)	0.872(±0.011)	0.878(±0.029)	0.891(±0.04)	
35 → 31	0.571(±0.024)	0.786(±0.0)	0.786(±0.0)	0.754(±0.036)	0.683(±0.107)	0.571(±0.024)	0.571(±0.024)	0.77(±0.014)	0.746(±0.0)	
6 → 19	0.879(±0.015)	0.843(±0.087)	0.843(±0.087)	0.747(±0.009)	0.864(±0.04)	0.879(±0.015)	0.879(±0.015)	0.854(±0.009)	0.909(±0.015)	
7 → 18	0.717(±0.05)	0.585(±0.019)	0.585(±0.019)	0.585(±0.019)	0.654(±0.143)	0.717(±0.05)	0.673(±0.022)	0.723(±0.029)		
Avg.	0.759(±0.03)	0.772(±0.034)	0.772(±0.034)	0.745(±0.025)	0.754(±0.068)	0.759(±0.03)	0.75(±0.025)	0.79(±0.013)	0.832(±0.021)	

Table 28: Mean and standard deviation (after \pm) of target classification accuracy on WISDM (Part 2) over 3 repetitions with different random initialization of model weights.

CoDATS										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
18 → 23	0.639(±0.051)	0.722(±0.069)	0.722(±0.069)	0.723(±0.058)	0.667(±0.033)	0.678(±0.077)	0.7(±0.067)	0.756(±0.019)	0.756(±0.019)	0.756(±0.019)
20 → 30	0.846(±0.069)	0.891(±0.029)	0.891(±0.029)	0.885(±0.038)	0.776(±0.022)	0.846(±0.069)	0.846(±0.069)	0.904(±0.019)	0.904(±0.019)	0.904(±0.019)
35 → 31	0.619(±0.095)	0.738(±0.048)	0.738(±0.048)	0.722(±0.014)	0.643(±0.124)	0.619(±0.095)	0.73(±0.036)	0.714(±0.063)	0.786(±0.024)	
6 → 19	0.727(±0.069)	0.899(±0.038)	0.899(±0.038)	0.864(±0.069)	0.742(±0.076)	0.727(±0.069)	0.727(±0.069)	0.899(±0.032)	0.924(±0.08)	
7 → 18	0.673(±0.071)	0.535(±0.076)	0.535(±0.076)	0.553(±0.066)	0.585(±0.098)	0.673(±0.071)	0.673(±0.071)	0.547(±0.057)	0.692(±0.142)	
Avg.	0.711(±0.071)	0.757(±0.052)	0.757(±0.052)	0.751(±0.049)	0.682(±0.071)	0.709(±0.076)	0.735(±0.063)	0.764(±0.038)	0.812(±0.063)	

Deep-Coral										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
18 → 23	0.656(±0.077)	0.680(±0.019)	0.680(±0.019)	0.680(±0.019)	0.667(±0.058)	0.656(±0.077)	0.656(±0.077)	0.689(±0.069)	0.711(±0.051)	
20 → 30	0.837(±0.033)	0.891(±0.020)	0.891(±0.020)	0.878(±0.04)	0.841(±0.04)	0.827(±0.033)	0.827(±0.033)	0.885(±0.020)	0.904(±0.038)	
35 → 31	0.619(±0.086)	0.69(±0.071)	0.69(±0.071)	0.667(±0.082)	0.532(±0.122)	0.619(±0.086)	0.619(±0.086)	0.69(±0.048)	0.706(±0.06)	
6 → 19	0.722(±0.044)	0.732(±0.053)	0.732(±0.053)	0.707(±0.035)	0.717(±0.068)	0.722(±0.044)	0.722(±0.044)	0.722(±0.009)	0.788(±0.139)	
7 → 18	0.648(±0.047)	0.61(±0.054)	0.61(±0.054)	0.629(±0.071)	0.566(±0.082)	0.648(±0.047)	0.528(±0.218)	0.629(±0.039)	0.654(±0.029)	
Avg.	0.694(±0.057)	0.723(±0.046)	0.723(±0.046)	0.713(±0.051)	0.664(±0.074)	0.694(±0.057)	0.67(±0.091)	0.723(±0.04)	0.753(±0.063)	

CDAN										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
18 → 23	0.707(±0.033)	0.678(±0.038)	0.678(±0.038)	0.724(±0.038)	0.726(±0.019)	0.707(±0.033)	0.767(±0.039)	0.7(±0.1)	0.767(±0.033)	
20 → 30	0.872(±0.029)	0.853(±0.029)	0.853(±0.029)	0.84(±0.011)	0.829(±0.01)	0.872(±0.029)	0.829(±0.01)	0.84(±0.029)	0.875(±0.044)	
35 → 31	0.563(±0.05)	0.738(±0.041)	0.738(±0.041)	0.716(±0.036)	0.651(±0.144)	0.563(±0.05)	0.722(±0.172)	0.722(±0.014)	0.77(±0.06)	
6 → 19	0.869(±0.044)	0.833(±0.052)	0.833(±0.052)	0.838(±0.049)	0.838(±0.049)	0.869(±0.044)	0.717(±0.306)	0.848(±0.045)	0.869(±0.044)	
7 → 18	0.73(±0.03)	0.711(±0.115)	0.711(±0.115)	0.761(±0.039)	0.723(±0.058)	0.73(±0.03)	0.711(±0.126)	0.717(±0.068)	0.767(±0.076)	
Avg.	0.76(±0.05)	0.762(±0.055)	0.762(±0.055)	0.781(±0.035)	0.768(±0.062)	0.76(±0.05)	0.75(±0.135)	0.765(±0.031)	0.81(±0.052)	

DANN										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
18 → 23	0.689(±0.107)	0.778(±0.069)	0.778(±0.069)	0.778(±0.069)	0.756(±0.038)	0.667(±0.088)	0.656(±0.077)	0.767(±0.019)	0.778(±0.019)	
20 → 30	0.846(±0.033)	0.856(±0.029)	0.856(±0.029)	0.84(±0.022)	0.865(±0.033)	0.856(±0.033)	0.856(±0.033)	0.84(±0.029)	0.85(±0.033)	
35 → 31	0.651(±0.107)	0.77(±0.04)	0.77(±0.04)	0.77(±0.027)	0.659(±0.159)	0.651(±0.107)	0.651(±0.107)	0.77(±0.027)	0.746(±0.027)	
6 → 19	0.788(±0.124)	0.914(±0.046)	0.914(±0.046)	0.924(±0.052)	0.854(±0.126)	0.788(±0.124)	0.788(±0.124)	0.899(±0.061)	0.919(±0.052)	
7 → 18	0.608(±0.033)	0.704(±0.029)	0.704(±0.029)	0.648(±0.029)	0.591(±0.104)	0.698(±0.033)	0.623(±0.191)	0.61(±0.054)	0.698(±0.029)	
Avg.	0.724(±0.088)	0.802(±0.032)	0.802(±0.032)	0.796(±0.04)	0.745(±0.092)	0.72(±0.084)	0.702(±0.113)	0.778(±0.032)	0.805(±0.036)	

DSAN										
Task	Heuristic					Theoretical error guarantees				
	SO	TMV	TMR	TCR	SOR	IWV	DEV	IWA (ours)	TB	
18 → 23	0.756(±0.051)	0.767(±0.033)	0.767(±0.033)	0.778(±0.038)	0.711(±0.051)	0.756(±0.051)	0.756(±0.051)	0.744(±0.019)	0.8(±0.0)	
20 → 30	0.872(±0.011)	0.859(±0.022)	0.859(±0.022)	0.846(±0.0)	0.84(±0.011)	0.872(±0.011)	0.731(±0.233)	0.865(±0.0)	0.872(±0.011)	
35 → 31	0.571(±0.024)	0.786(±0.0)	0.786(±0.0)	0.778(±0.014)	0.698(±0.151)	0.571(±0.024)	0.571(±0.024)	0.778(±0.014)	0.817(±0.027)	
6 → 19	0.879(±0.015)	0.864(±0.052)	0.864(±0.052)	0.833(±0.052)	0.889(±0.009)	0.879(±0.015)	0.662(±0.363)	0.838(±0.049)	0.879(±0.009)	
7 → 18	0.717(±0.065)	0.572(±0.076)	0.572(±0.076)	0.547(±0.068)	0.648(±0.054)	0.717(±0.05)	0.597(±0.076)	0.384(±0.047)	0.717(±0.05)	
Avg.	0.759(±0.03)	0.769(±0.037)	0.769(±0.037)	0.756(±0.035)	0.757(±0.055)	0.759(±0.03)	0.663(±0.149)	0.722(±0.026)	0.817(±0.019)	

2.2. ADDRESSING PARAMETER CHOICE ISSUES IN UDA

2.3 *SymbolicAI*: A framework for logic-based approaches combining generative models and solvers

SYMBOLICAI: A FRAMEWORK FOR LOGIC-BASED APPROACHES COMBINING GENERATIVE MODELS AND SOLVERS

Marius–Constantin Dinu^{*†‡} Claudiu Leoveanu–Condrei^{†||} Markus Holzleitner[‡]
Werner Zellinger^{‡§} Sepp Hochreiter[‡] Amazon Devices[¶]
ExtensityAI[†] Johannes Kepler University[‡] RICAM[§]

ABSTRACT

We introduce *SymbolicAI*, a versatile and modular framework employing a logic-based approach to concept learning and flow management in generative processes. *SymbolicAI* enables the seamless integration of generative models with a diverse range of solvers by treating large language models (LLMs) as semantic parsers that execute tasks based on both natural and formal language instructions, thus bridging the gap between symbolic reasoning and generative AI. We leverage probabilistic programming principles to tackle complex tasks, and utilize differentiable and classical programming paradigms with their respective strengths. The framework introduces a set of polymorphic, compositional, and self-referential operations for multi-modal data that connects multi-step generative processes and aligns their outputs with user objectives in complex workflows. As a result, we can transition between the capabilities of various foundation models with in-context learning capabilities and specialized, fine-tuned models or solvers proficient in addressing specific problems. Through these operations based on in-context learning our framework enables the creation and evaluation of explainable computational graphs. Finally, we introduce a quality measure and its empirical score for evaluating these computational graphs, and propose a benchmark that compares various state-of-the-art LLMs across a set of complex workflows. We refer to the empirical score as the "Vector Embedding for Relational Trajectory Evaluation through Cross-similarity", or *VERTEX* score for short. The [framework codebase¹](#) and [benchmark²](#) are linked below.

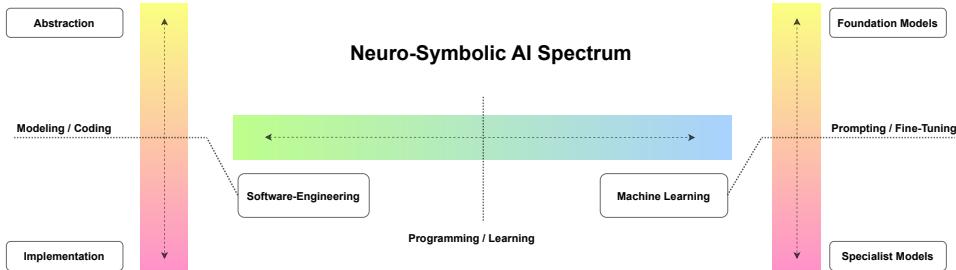


Figure 1: Our neuro-symbolic framework enables a seamless transition between symbolic and differentiable programming, each with distinct dynamics and strengths. Differentiable programming provides access to foundational and specialist models. Classical programming, on the other hand, shifts between abstraction and implementation, focusing on high-level concepts before delving into the details of implementation.

¹ SymbolicAI framework: <https://github.com/ExtensityAI/symbolicai>

² Evaluation benchmark: <https://github.com/ExtensityAI/benchmark>

* Correspondence to: dinu@ml.jku.at, {marius, leo}@extensity.ai

|| Work done outside of Amazon.

1 INTRODUCTION

The recent surge in generative AI, particularly involving large language models (LLMs), has demonstrated their wide-ranging applicability across various domains (Badita, 2022; Degrave, 2022). These models have enhanced the functionality of tools for search-based interactions (YouWrite, 2022; Writesonic, 2022; Microsoft, 2023), program synthesis (Jain et al., 2021; Romera-Paredes et al., 2023; Key et al., 2023), chat-based interactions (ReplikaAI, 2016; OpenAI, 2022; Google, 2023), and many more. Moreover, language-based approaches have facilitated connections between different modalities, enabling text-to-image (Ramesh et al., 2021; Saharia et al., 2022), text-to-video (Singer et al., 2022), text-to-3D (Poole et al., 2022), text-to-audio (Oord et al., 2016; Wang et al., 2017), and text-to-code (Wang et al., 2021b; Lu et al., 2021; Li et al., 2022b) transformations, to name a few. Consequently, by training on vast quantities of unlabelled textual data, LLMs have been shown to not only store factual knowledge (Petroni et al., 2019; Kassner et al., 2020) and approximate users' intentions to some extent (Andreas, 2022), but also to unlock deep specialist capabilities through innovative prompting techniques (Nori et al., 2023).

Despite their versatility, current LLMs face challenges such as fallacious reasoning and the generation of erroneous content, commonly referred to as hallucinations (Jones & Steinhardt, 2022). These limitations highlight the importance of integrating complementary symbolic methods to validate and guide the generative processes of LLMs, ensuring more accurate and reliable outputs. In parallel, efforts have focused on developing tool-based approaches (Schick et al., 2023) or template frameworks (Chase, 2023) to extend LLMs' capabilities and enable a broader spectrum of applications. However, these efforts only partially capture the potential inherent in leveraging LLMs as *semantic parsers*. In contrast to parsers for structured languages a semantic parser is able to break down unstructured human language into semantically meaningful components and transform those into a structured form. While traditionally semantic parsing has been a role filled by specialized algorithms and models, we posit that LLMs, through their training on diverse linguistic data, have developed the ability to perform semantic parsing as part of their broader natural language processing capabilities. In turn, we identify LLMs as a central component in creating sophisticated neuro-symbolic (NeSy) AI systems. These systems integrate symbolic and sub-symbolic concepts and utilize the capabilities of semantic parsing to develop symbolic expressions that enable new probabilistic programming paradigms.

We introduce *SymbolicAI*, a compositional NeSy framework able to represent and manipulate multi-modal and self-referential structures (Schmidhuber, 2007; Fernando et al., 2023). SymbolicAI augments the generative process of LLMs with in-context learning operations, realized through functional primitives, and enables the creation of versatile applications through in-context learning (Wei et al., 2022a). These operations enable logic-based components that guide the generative process and enable a modular NeSy system, including a wide range of existing solvers, formal language engines for mathematical expression evaluation, theorem provers, knowledge bases, and search engines for information retrieval. SymbolicAI exposes these solvers as building blocks for constructing compositional functions as computational graphs, making it possible to bridge classical and differentiable programming paradigms with the aim to create *domain-invariant problem solvers*. In designing the architecture of SymbolicAI, we drew inspiration from a body of evidence that suggests the human brain possesses a selective language processing module (MacSweeney, 2002; Fedorenko et al., 2010; Menenti et al., 2011; Regev et al., 2013; Scott et al., 2016; Deniz et al., 2019; Hu et al., 2022), prior research on cognitive architectures (Newell & Simon, 1956; Newell et al., 1957; Newell & Simon, 1972; Newell, 1990; Laird, 2022), and the significance of language on the structure of semantic maps in the human brain (Huth et al., 2016). We consider language as a central processing module, distinct from other cognitive processes such as reasoning or memory (Paischer et al., 2022; 2023). We hypothesize that such a central processing module based in language is a core component of broad AI systems (see Appendix Section A) and enables the development of fully autonomous AI systems for decision-making.

A significant challenge encountered in the development of our framework pertained to the evaluation of LLMs when used as semantic parsers in a NeSy workflow. Current evaluation of generated content relies on metrics for single-step generative processes, such as the BLEU score (Papineni et al., 2002). These metrics are not suitable for evaluating multi-step generative processes. BLEU has limitations, as it measures n -gram-based overlap of generated output with a reference that does not consider the semantic meaning. As a result, BLEU fails to capture semantic equivalence, especially in more complex tasks. More recent metrics such as CIDEr (Vedantam et al., 2014) or SPICE (Anderson et al., 2016) are also not suitable for our problem, either because they are built upon BLEU (in case of CIDEr) or designed with inductive biases specifically for image captioning.

Therefore, alongside our framework we introduce a quality measure (and its empirical score) for multi-step generative processes based on semantic meaning. We refer to our score as "Vector Embedding for Relational Trajectory Evaluation through Cross-similarity", or VERTEX score for short. Our VERTEX score uses embeddings to compare node distributions within a computational graph. It measures the semantic meaning across the distributional path by computing at each node the cross-similarity between the generated embeddings and embeddings sampled from a reference

distribution. Furthermore, the VERTEX score is designed such that it can be used as a reward signal in a reinforcement learning setting (Sutton, 1984). Finally, we propose a benchmark for evaluating complex workflows. We define a set of basic evaluations, particularly associative predictions based on in-context learning, multi-modal bindings for tool utilization, and program synthesis for subroutine execution. Furthermore, we introduce complex evaluations for logic-based components and hierarchical computational graphs.

In summary, the key contributions presented in this work are as follows:

- We introduce SymbolicAI, a logic-based framework for concept learning and flow management in generative processes, enabling seamless integration with a wide range of foundation models and solvers.
- We leverage LLMs as semantic parsers to enable the creation of complex computational graphs by combining symbolic expressions with probabilistic programming paradigms.
- We introduce a quality measure and its empirical score alongside a benchmark designed for multi-step generative processes for comparing LLMs across a wide range of complex tasks.

2 RELATED WORK

Symbolic Methods The field of symbolic AI has its foundations in the works of the Logic Theorist (LT) (Newell & Simon, 1956) and the General Problem Solver (GPS) (Newell et al., 1957). These programs represented the first steps towards automated reasoning and problem-solving utilizing symbolic representations. Despite their advancements, both faced challenges in dealing with the complexity of real-world problems, particularly due to the combinatorial nature of the solution space. To address these limitations, the Soar (Laird et al., 1987) cognitive architecture was developed, advancing the notion that intelligent behavior results from goal-oriented search through a problem space (Newell & Simon, 1972; McCarthy et al., 2006), with each step consisting of selecting and applying operators. Soar introduced components like reinforcement learning, impasses, sub-states, and chunking to enhance its problem-solving capabilities. It also demonstrated the importance of learning from experiences to adapt and improve performance over time. However, Santoro et al. (2022) emphasizes the subjectivity of symbols and suggests that human-like symbolic fluency could develop in machines through learning algorithms immersed in socio-cultural contexts. This perspective, anchored in the notion that symbols are triadic and their meaning emerges from consensus, seeks to move away from traditional symbolic AI methodologies towards AI that adaptively learns meaning and behaviors from human-like experiences. The goal is to cultivate machines that demonstrate symbolic behaviors across a spectrum of competencies, potentially mirroring the evolutionary and social learning processes observed in humans. Lastly, symbolic AI struggles with real-world data's unpredictability and variability. These challenges have led to the employment of statistical learning methodologies, like deep learning (Alom et al., 2018), which are more adept at managing noise and uncertain information through vector-valued representations.

Sub-Symbolic Methods The sub-symbolic framework, rooted in neural network paradigms, began with pioneering works such as the perceptron (McCulloch & Pitts, 1943), with the first hardware implementation quickly following (Rosenblatt, 1958). The foundational notion of distributed processing (Rumelhart et al., 1986) was later bolstered and further expanded by demonstrating that multilayer feedforward networks with a single hidden layer can serve as universal approximators for any Borel measurable function, given sufficient hidden units (Hornik et al., 1989). Fast-forward, contemporary frameworks achieve a significant leap with the introduction of the Transformer architecture (Vaswani et al., 2017), which underpins most of today's LLMs. These LLMs demonstrate exceptional capabilities in in-context learning, a method popularized by the likes of GPT-3 (Brown et al., 2020), where models improve task performance through natural language instruction and examples provided directly in the input prompt. While in-context learning bypasses the need for explicit retraining, it demands meticulous prompt design to steer models towards desired behaviors.

Neuro-Symbolic Methods To overcome the limitations of each individual method, NeSy approaches meld the statistical inference strengths of deep neural architectures with the generalization and explainability of symbolic systems (Garcez et al., 2015; Besold et al., 2017; d'Avila Garcez et al., 2019; d'Avila Garcez & Lamb, 2020; Lamb et al., 2020; Hamilton et al., 2022; Yu et al., 2023). Some approaches focus on different strategies for integrating learning and reasoning processes (Yu et al., 2023; Fang et al., 2024). Firstly, *learning for reasoning* methods treat the learning aspect as an accelerator for reasoning, in which deep neural networks are employed to reduce the search space for symbolic systems (Silver et al., 2016; 2017b;a; Qu & Tang, 2019; Schrittwieser et al., 2020). Secondly, *reasoning for learning* views reasoning as a way to regularize learning, in which symbolic knowledge acts as a guiding constraint that oversees machine learning tasks (Hu et al., 2016; Xu et al., 2018). Thirdly, the *learning-reasoning* category enables a symbiotic relationship between learning and reasoning. Here, both elements interact and share information to boost

problem-solving capabilities (Donadello et al., 2017; Manhaeve et al., 2018; Mao et al., 2019; Ellis, 2023). This synergy further extends when considering graph-based methods, which closely align with the objectives of our proposed framework. Research in this area, such as CycleGT (Guo et al., 2020) and Paper2vec (Ganguly & Pudi, 2017) explores unsupervised techniques for bridging graph and text representations. GPTSwarm (Zhuge et al., 2024) explores graph optimizers to refine node-level prompts and edge optimization. Subsequently, graph embeddings, when utilized within symbolic frameworks, can enhance knowledge graph reasoning tasks (Zhang et al., 2021), or more generally, provide the bedrock for learning domain-invariant representations (Park et al., 2023).

Lastly, building upon the insights from Sun et al. (2022), the integration of NeSy techniques in scientific workflows promises significant acceleration in scientific discovery. While previous work has effectively identified opportunities and challenges, we have taken a more ambitious approach by developing a comprehensive framework from the ground up to facilitate a wide range of NeSy integrations.

Large Language Models In part, instruction-based fine-tuning of LLMs through reinforcement learning from human feedback (Ouyang et al., 2022; Li et al., 2023) or direct preference optimization (Rafailov et al., 2023) has shown promising results dealing with value misalignment issues (Bradley Knox & Stone, 2008; MacGlashan et al., 2017; Christiano et al., 2017; Ibarz et al., 2018; Goyal et al., 2022), unlocking new possibilities for chain of thoughts (Wei et al., 2022b), tree of thoughts (Yao et al., 2023a), and graph of thoughts interactions (Besta et al., 2023). However, recent research also highlights the limitations of LLMs in functional linguistic competence despite their proficiency in formal linguistic competence (Mahowald et al., 2023). Whereas formal linguistic competence encompasses the ability to understand and generate language, functional linguistic competence pertains to the application of language in real-world contexts, such as conveying sensory input or recalling information from memory. Examples of functional linguistic competence include implicatures (Ruis et al., 2022) and contextual language comprehension beyond the statistical manifestation of data distributions (Bransford & Johnson, 1972; Mikolov et al., 2013b). Consequently, operating LLMs through a purely inference-based approach confines their capabilities within their provided context window, severely limiting their horizon. This results in deficiencies for situational modeling, non-adaptability through contextual changes, and short-term problem-solving, amongst other capabilities. However, simply increasing the context length may not yield greater capabilities, as demonstrated by the observed U-shaped performance curve (Liu et al., 2023) where LLMs excel when utilizing information at the beginning or end of the input context, but struggle with information located in the middle, especially as context increases. These challenges are actively being researched, with novel approaches such as Hyena (Poli et al., 2023), RWKV (Bo, 2021), GateLoop (Katsch, 2023), Mamba (Gu & Dao, 2023) and xLSTM (Beck et al., 2024) surfacing. Meanwhile, the re-emergence of interest in retrieval-augmented generative approaches (Li et al., 2022a) offers an alternative by circumventing the autoregressive nature of the widely-utilized Transformer architecture (Vaswani et al., 2017), enabling context enrichment with lateral information.

In-Context Learning Recently, several in-context learning methodologies evolved to enable tool usage through LLMs (Schick et al., 2023), or refine the generative outcome of LLMs (Yang et al., 2023). This includes chain-of-thought (CoT) prompting, a method that conditions the model to reveal its step-by-step reasoning process (Wei et al., 2022b; Singhal et al., 2023). CoT prompting breaks down complex tasks into simpler, sequential steps, and helps with interpreting LLM's output. Self-generated CoT, where models are encouraged to generate their own reasoning chains based on training examples, surpasses even expertly crafted CoT (Fernando et al., 2023). This observation echoes other reports that GPT-4 has an emergent self-improving capability through introspection, such as self-verification (Weng et al., 2023) or self-consistency (Wang et al., 2023b). Tree of Thoughts (ToT) enables LLMs to solve complex problems by exploring multiple reasoning paths through a search tree of coherent text units, demonstrating significant problem-solving enhancements in tasks requiring strategic planning and search (Yao et al., 2023a). Ensemble techniques further enhance the robustness and accuracy of model predictions by combining several strategies to establish a consensus (Nori et al., 2023).

3 PROBLEM DEFINITION

Conventional approaches employing foundation models, such as LLMs, are predominantly confined to single-step or few-step executions and primarily reliant on hand-crafted prompt instructions, often referred to as in-context learning. This restricted scope limits the utilization of different modalities, lacks verification, and exhibits limited tool proficiency. We posit that the use of NeSy engines as core computation units, realized through logic-based methodologies coupled with sub-symbolic foundation models, offers a more general, robust, and verifiable perspective. This approach has several advantages. Firstly, it enables the integration of pre-existing solutions (e.g. various classical algorithms), offloading computational complexity and bridging different modalities. Secondly, it allows sub-symbolic components to focus on decision-making (e.g. selecting the respective tool based on in-context classification). Thirdly, it provides

an *interpretable language-based control layer* for explainable, autonomous systems. In the following section, we elaborate on the key design principles underlying SymbolicAI and how we guide the generative processes of NeSy engines. For further technical details, see Appendix Section 5.

4 DESIGN PRINCIPLES

Symbols and Expressions As posited by Newell & Simon (1976), symbols are elemental carriers of meaning within a computational context³. These symbols define physical patterns capable of composing complex structures, and are central to the design and interpretation of logic and knowledge representations (Augusto, 2022). We define a symbol as the set $\mathcal{S} = \bigcup_{n \geq 0} \mathbb{L}^n$ formed by concatenating characters from a finite character set \mathbb{L} , i.e. the vocabulary in an LLM setting, and with n representing the sequence length of the string. Thus, let the set of all possible symbols be defined as Σ and that $\mathcal{S} \in \Sigma$. We further introduce an operation \oplus that enables us to create expressions on any number of symbols from Σ , and when evaluated returns a new symbol in Σ . For any subset $\{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_m\} \subseteq \Sigma$, an expression is defined as $\omega : \bigoplus_{i=1}^m \mathcal{S}_i \rightarrow \mathcal{S}'$ from the set of all possible expressions $\omega \in \Omega$, where $\mathcal{S}' \in \Sigma$, and \oplus represents the placeholder operation of combining and transforming the symbols according to specific rules for m number of symbols. Such a specific rule for \oplus can define an arithmetic expression $\oplus := +$ where two symbols are added, i.e. $\omega := "1" + "two"$ which results in a new symbol "3" or "three". Thus, SymbolicAI is based on the concept that symbols, and the expressions they form, are reflections of the information inherent in a NeSy system, and serve as surrogate for the interaction between the NeSy system and the problem space. Moreover, we argue that *real patterns* (Dennett, 1991), recurring and identifiable structures that coherently and reliably emerge in the data beyond mere randomness or noise, can be effectively realized through symbols.

Furthermore, we utilize language as a tool for mapping complex concepts, leveraging its inherent semantics and abstractions to describe states and properties of a problem at hand. These mappings are universal, e.g. they may be utilized to define scene descriptions, long-horizon planning, acoustic properties, emotional states, physical conditions, etc. Therefore, language serves as a comprehensive, yet abstract framework to encapsulate meanings, and refer to it as the *convex hull of the knowledge of our society*. Subsequently, it is common to attribute existing physical objects with abstract concepts, as exemplified by our natural tendency to link tangible objects to colors and emotions, such as blending the color "red" with "heart", "warm", and "passion". This approach also anchors our work in the field of formal language theory, as we require a structured method to construct mappings from the world to language. Consequently, we use formal language structures, such as grammars, to systematically define our language-centric approach to problem-solving and the associated translation of real-world complexities into linguistic terms.

Formal Languages In formal language theory and linguistics, languages are structured following the Chomsky hierarchy, which classifies languages by the complexity of their grammatical structure (Chomsky, 1956). This hierarchy defines four types of grammars (Type-3 to Type-0) and separates formal languages by their grammatical complexity. A grammar in this context consists of terminal and non-terminal symbols, production rules, and a designated *start symbol*, enabling the generation of valid strings within a language.

We define a NeSy engine as a mapping $\mathcal{V}_{\mathcal{S}^*} : \Omega \times N \times \mathcal{T} \rightarrow \Sigma$, where $N \subset \Sigma$ is a set of non-terminal symbols, $\mathcal{T} \subset \Sigma$ is a set of terminal symbols and $N \cap \mathcal{T} = \emptyset$, and $\mathcal{S}^* \in \Sigma$ is a starting symbol. We further formalize a grammar $G = (N, \mathcal{T}, P, \mathcal{S}^*)$ with production rules defined as a $P := \mathcal{V}_{\mathcal{S}^*}(\omega, N, \mathcal{T})$. This grammar describes the generation of symbols through expressions ω . For simplicity, we will drop the subscript of $\mathcal{V}_{\mathcal{S}^*}$ and use it as \mathcal{V} . We identify LLMs as promising candidates for functioning as part of NeSy engines. In SymbolicAI, a symbol \mathcal{S} is augmented with conditional instructions and types derived from DSLs, custom defined or not (e.g. HTML, SQL, etc.), tailored for directing the LLMs. The key advantage of LLMs over previous systems lies in their ability to generalize across formal languages (Wang et al., 2023a) and knowledge systems. Although there is currently no universal consensus regarding the precise classification of natural language within the Chomsky hierarchy, our approach can be understood as employing a *situation-specific*, context-sensitive grammar, which enables the processing of instructions and analogies with a nuanced understanding of language. The intersection between formal and natural languages becomes evident when considering how language patterns, through prompts like "You are a helpful assistant...", elicit structured responses, indicating a potential underlying formal mechanism at play. This observation underlines the utility of such a grammar in our framework, where it serves as an explicit schema guiding the structure of examples for in-context learning. For instance, equating "3.1415..." with " π " or "August 4, 1961" with "1961-08-04" in a given context demonstrates context-dependent interpretation of symbols. Such a system doesn't rigidly adhere to standard grammatical rules but instead adjusts and interprets based on the context, effectively creating a situation-specific grammar.

³ Our framework's name is derived from the foundational work of Newell and Simon.

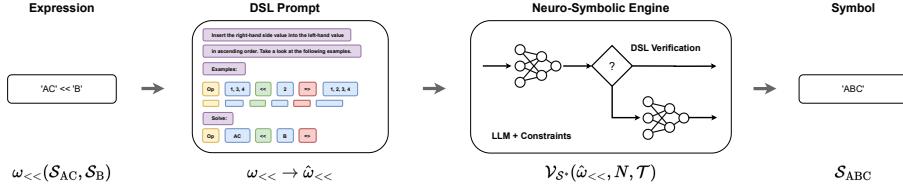


Figure 2: Illustration for NeSy pipeline, showcasing conceptual usage of in-context learning methodologies, domain-specific language (DSL) structures, and the expression evaluations through a NeSy engine based on an LLM and constraint verification. The expression showcases the sorted insert operator \ll and how the information of the symbol B is included in the symbol AC. The violet placeholder in the *DSL Prompt* represents an instruction, such as “*Insert the right-hand side value into the left-hand value in ascending order.*” The positions below represent task-specific few-shot examples. The *DSL Prompt* receives the expression $\omega_{<<}$ and maps it to $\hat{\omega}_{<<}$ that can be processed by the LLM-based NeSy function $\mathcal{V}_{\mathcal{S}^*}$ and outputs a new symbol.

mar, capable of forming *Domain-Invariant Associations* through in-context learning. We further address this in a later paragraph.

Function Composition In SymbolicAI, we use function composition to construct complex hierarchies and behaviors from fundamental elements. Therefore, our framework enables modeling of interconnected processes, where the output of one function is used as input for another, thus creating a sequence of operations. Through function composition, we construct computational graphs, in which intermediate symbols represent the nodes or states within these graphs. Formally, function composition is denoted by \circ , where combining functions f and g yields a new function $h = g \circ f$, defined as $h(x) = g(f(x))$. For functions $f : X \rightarrow Y$ and $g : Y \rightarrow Z$, their composition results in a function mapping elements from domain X to codomain Z through $g(f(x))$. Although traditionally the codomain of the inner function f aligns with the domain of the outer function g , SymbolicAI relaxes this constraint by allowing for any subset relationship between these domains and codomains, which is particularly beneficial for in-context learning. When using LLMs for NeSy production rules \mathcal{V} , we can derive a multi-step generative process by composing a computational graph as a sequence of zero- and few-shot function compositions:

$$\mathcal{V}(\omega_j, N, \mathcal{T}) = \mathcal{V}(\omega_{j-1}, \cdot) \circ \mathcal{V}(\omega_{j-2}, \cdot) \circ \cdots \circ \mathcal{V}(\omega_0, \cdot), \quad (1)$$

where ω_0 is the initial instruction and j defines the index variable for a multi-step generative process. By leveraging functional in-context learning, where zero- and few-shot examples act as dynamic elements of the function’s domain, SymbolicAI has the ability to interpret and respond to diverse input contexts. For instance, a function can classify a user request and select an appropriate interface (e.g. WolframAlpha) to process the request. The output modality may even vary based on the respective engine. This enables SymbolicAI to handle operations over multi-modal data that connects multi-step generative processes and establishes function composition as a central tenet in bridging multiple modalities and coordinating a variety of tasks.

Domain-Invariant Associations In-context learning enabled LLMs to become versatile task solvers by interpolating within the training distribution, to the extent that even potentially unseen tasks are addressable (Brown et al., 2020). We attribute this to associations formed within the input space and the capacity of Transformer architectures for defining domain-invariant feature sub-spaces. This phenomenon has parallels with few-shot learning approaches such as SubGD (Gauch et al., 2022), a method based on identifying and utilizing a low-dimensional subspace, learned from various tasks that effectively regularize the learning process. Since LLMs have been trained on different domains and tasks, which also include formulations of mathematical expressions, we posit that specific tokens, such as the equality sign, can be leveraged to associate meanings between different symbolic objects. Unlike domain-invariant representations that create invariant features across different learning tasks, our approach leverages the in-context generalization capability of LLMs to construct invariant symbolic associations that aim to preserve, manipulate and propagate situational context. We can use these properties to build operations that apply transformations on objects that are substitutes to the semantically aligned few-shot learning examples.

5 SYMBOLICAI FRAMEWORK

In this section, we discuss the specifics of the proposed SymbolicAI framework. For more details about the framework structure, see Appendix Section C. For installation and usage of our framework, see Appendix Section D. For more technical details and code snippets, see Appendix Section E.

Types and Representations Analogous to the type `object` in Python, the base type of SymbolicAI is a symbol represented by the base type `Symbol`. All other subtypes, such as `Expression`, represent their mathematical namesake and can be evaluated and simplified. These subtypes inherit from `Symbol` the base attributes, primitive operators, and helper methods.

Although SymbolicAI uses a language-centric design, modeling and manipulating every interaction into symbolic representations is not inherently efficient. Therefore, we establish mappings between symbolic and sub-symbolic representations for sensory inputs and non-discrete elements. Such mappings are typically realized through function approximation. This allows us to map between *modality-to-language* and *language-to-modality* use cases. Here, *modality* serves as a placeholder for various types such as text, image, video, audio, motion, etc. In turn, each `Symbol` object contains valued and vector-valued representations, obtained through `value` and `embedding` attributes. The latter represents a symbol's current value, akin to embedding text and storing it as a PyTorch tensor (Paszke et al., 2019) or NumPy array (Harris et al., 2020). While for an LLM, the numerical tensors may lack inherent meaning, vector-valued representations play an important role when 1) composite symbols are combined into more complex expressions, and 2) these embedded tensors are updated through gradient-based optimization.

To enable the processing of symbols by LLMs, we assume that each `Symbol` object implements Python's native string functionality, where the `__str__` method returns an interpretable string representation. Therefore, we can assert that any Python object is parsable by an LLM, however, the user must ensure a meaningful representation. For more details, see Appendix Section E.

Polymorphic Context Polymorphism is a central concept in programming language theory and prominently featured in SymbolicAI. Polymorphism refers to the ability of different objects to be accessed through the same interface, or of a single identifier to represent different types based on the context of execution. Providing a single interface for entities of different types allows operations to be performed in ways specific to their derived types. We therefore designed the `Symbol` object to contain a global context, which is composed of static and dynamic context parts, and enables this polymorphic behavior. The static context is class dependent and defined at design time. The dynamic context is runtime adaptable and can be changed to adhere to runtime specific logic and changes. Moreover, `Symbol` associated operations are resolved following polymorphic design before being evaluated by the NeSy engine. SymbolicAI's engine implementation contains a `prepare` method to resolve and compile the engine specific representation by evaluating the `Symbol`-specific operations and context. For an example on polymorphic context see part a) in Figure 3.

Operators and Methods In SymbolicAI, operators are overloaded to facilitate transformations of `Symbol` objects. These operator primitives employ dynamic casting to ensure type compatibility. Consequently, `Symbol` objects can be easily manipulated through type specific attributions or symbolically evaluated by the NeSy engine. For example, a central operation for boolean logic is measuring equality between symbols. To evaluate the equality of symbols, we primarily adhere to the type specific implementation, because we prioritize strict comparisons over probabilistic evaluations. If the evaluation was unsuccessful, we then consider semantic equality through the NeSy engine. SymbolicAI leverages decorators for composing operators and custom class methods. For more details, see Appendix Section C.

Upon invoking an operator or method, the respective primitive function evaluates the symbol's specific type and its respective attributes, and if necessary, resolves a nested decorated function that then uses the NeSy engine for evaluation. Should the evaluation fail, a predefined fallback implementation executes. Absent a fallback, or if both evaluations fail, an error state is raised. The processing of an operator or custom method involves a pipeline consisting of pre- and post-processing steps, as well as constraint enforcement. Constraints cover aspects like return types, value ranges, and structural integrity (e.g. JSON formatting through grammar-based verification). In Figure 3 b) we give an overview of the entire prompt composition based on the user input, the `Symbol` object structure, and in part c) the `Symbol` evaluation pipeline.

Self-Referential Structures SymbolicAI augments the generative process by enabling systems to introspect and modify their behavior dynamically. We leverage LLMs to execute tasks based on both natural and formal language instructions, adhering to the specified user objectives and with innate self-referential structures. We derive subtypes from `Expression` and enclose their functionalities in task-specific components, which we then expose again through

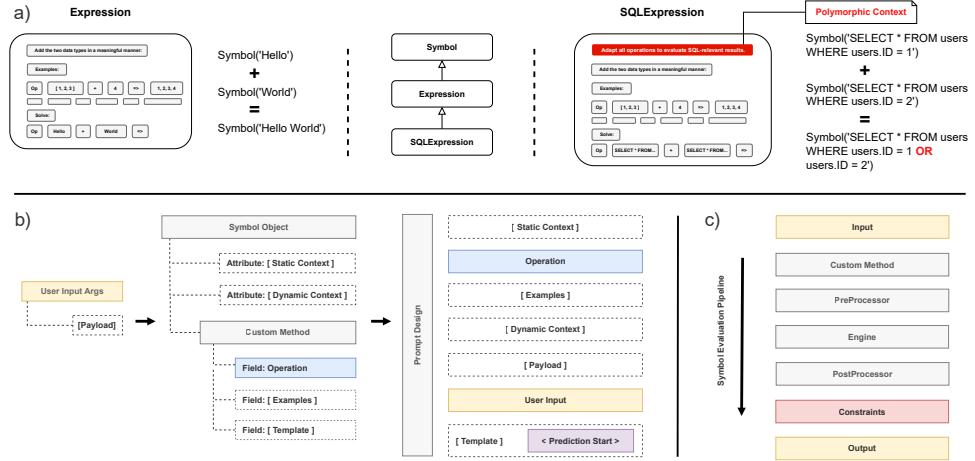


Figure 3: a) Illustration of polymorphic context on the example of a SQLExpression type for the *add*-operator. Without a polymorphic context a regular Expression evaluation concatenates two Symbol objects together. The polymorphic context in SQLExpression overwrites the base behavior such that two added SQL-expressions get semantically combined, not concatenated. b) Illustration of the translation of a Symbol object to a prompt statement to be processed by an LLM in the NeSy engine. The User Input Args can be attached with a Payload from previous executions and gets applied to the Custom Method. The user input with the polymorphic context of the Symbol Object attributes (Static Context and Dynamic Context) are translated to a prompt statement according to the schema of the Prompt Design. The fields Operation, Examples and Template mark operation description, DSL-based prompt examples and template structures respectively. These translations are processed according to PreProcessor and engine-specific formatting. c) Illustrates the evaluation pipeline from user input to output, with multiple translation processes before and after the Engine invocation. The Input gets passed to the Custom Method and reformatted according to a PreProcessor to adhere to DSL-specific structure. The engine then takes the output of the PreProcessor and composes the final prompt according to the engine-specific Prompt Design and resolves polymorphic context and auxiliary fields. The output of the Engine then can be restructured by a PostProcessor to match DSL-requirements of the desired Output and gets applied Constraints to verify the outcome.

templating and the model-driven design of the NeSy engine. This design choice allows a system to create and utilize its own sub-process definitions, analogous to concepts discussed in Schmidhuber (2007; 2009). Concretely, we utilize generalization properties of LLMs to interpret and formulate a set of operations that incorporate *self-instructions* (Wang et al., 2022). Consequently, the operations hold the flexibility to adapt to the context, and derive sub-processes that self-instruct LLMs to engage in situational modeling and context-sensitive problem-solving. Ultimately, this enables the construction of hierarchical computational graphs for self-referential *meta-reasoning* systems without the need to explicitly training a meta-learner (Kirsch & Schmidhuber, 2022). In Figure 4 we illustrate a step-wise evaluation of a contextual computational graph, in which the NeSy engine is processing conditioned on the current execution context and producing a next symbol prediction.

6 PERFORMANCE MEASURE

One of the challenges when creating multi-step generative processes with LLMs as part of NeSy engines relies on model evaluation and handling irrelevant predictions. The naïve assessment that measures only task succession would score all models to zero and render them as unusable. Even if models follow instructions and produce parts of the expected solution, we regularly observe that they — especially open-source models — append a continuation of task irrelevant predictions. Such predictions result in failure modes when applying conditions and validations, and halt any multi-step procedure. Our solution is an evaluation protocol that refines the performance measurement, allowing for more nuanced diagnostics and the possibility of continuing the evaluation despite intermediate failures. To derive our

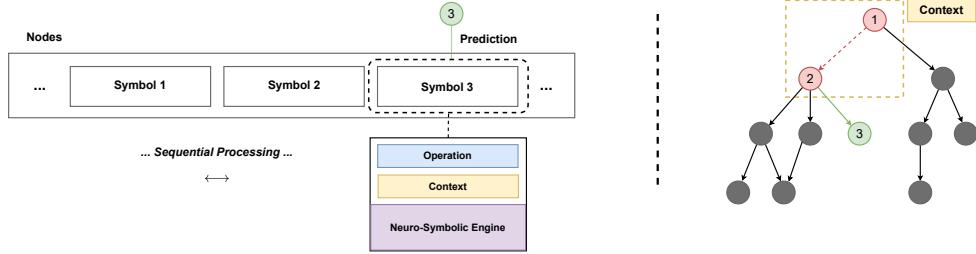


Figure 4: We showcase a multi-step hierarchical computational graph, with each node in the graph represented by a symbol. The edges are relations between symbols. The left-hand side illustrates how a new node (Symbol 3) is obtained by evaluating an operation with its respective context on a NeSy engine. The right-hand side illustrates the context information window (yellow rectangle) and relationship of the resulting graph with its respective nodes.

quality measure, we borrow ideas from the utilization of the Fréchet distance for generative processes (Heusel et al., 2017).

We generate trajectories through a NeSy sequential process that creates a trajectory of distributions \mathbb{P} over multiple iterations of generative nodes. Each node in the process can be aligned to a reference distribution, which marks the desired behavior. To quantify the validity of the generated trajectories, we measure the total distance between the generated and reference data distribution along the path trajectory. We therefore adopt a cumulative measure capable of taking into account the entire generative trajectory. In theory, this process would entail calculating the path integral over the latent space representations for models, cumulating the Fréchet distances (Dowson & Landau, 1982) traversed along these trajectories:

$$\mathcal{D}(\mathbb{P}_{\text{gen}}, \mathbb{P}_{\text{ref}}) = \int_{t_0}^{t_f} d(\mathcal{N}(m_t, C_t), \mathcal{N}(m_{w,t}, C_{w,t})) dt \quad (2)$$

where $\mathcal{D}(\mathbb{P}_{\text{gen}}, \mathbb{P}_{\text{ref}})$ denotes the integral of the Fréchet distances between two data distributions along the generative path trajectory from an initial time t_0 to a final time t_f , $d(\mathcal{N}(m_t, C_t), \mathcal{N}(m_{w,t}, C_{w,t}))$ is the Fréchet distance calculated at each time t between the generated multivariate normal data distribution with mean m_t and covariance C_t , and the reference multivariate normal data distribution with mean $m_{w,t}$ and covariance $C_{w,t}$. The resulting measure follows properties of normal distributions and is consistent with increasing disturbances.

However, this approach is computationally intractable for large-scale problems, and requires access to latent representations, which — especially in the context of LLMs — is not always given. For computational feasibility, we introduce an approximation that measures the embedding distances over the path trajectories through an auxiliary embedding model, based on prior work on distribution regression (Szabó et al., 2016). The embedding model maps the symbolic representations into a RKHS, such that we can apply a kernel mean embedding function to measure their respective distances (You et al., 2019; Dinu et al., 2023). We assess the distance through the mean embeddings w.r.t. to a kernel function $K(\cdot, \cdot)$ of the samples $e_x^t \sim \nu_{\text{gen}}^t \in \mathbb{P}_{\text{gen}}$ and $e_y^t \sim \nu_{\text{ref}}^t \in \mathbb{P}_{\text{ref}}$ produced by the generated data distribution and a reference data distribution respectively. We denote by $\mu_{e_x^t}, \mu_{e_y^t}$ the mean embeddings associated to the respective samples, i.e. $\mu_{e_x^t}(z) = \frac{1}{n} \sum_{i=1}^n K(x_i^t, z)$ in case $e_x^t = (x_i^t)_{i=1}^n$ is a sample of size n of the respective mean embeddings. To compute the similarity between the embeddings of the generated and reference distributions, we evaluate the associated maximum mean discrepancy $\text{MMD}^2(\mu_{e_x^t}, \mu_{e_y^t})$ (Gretton et al., 2012) and then, as before for the Fréchet distances, we integrate over t :

$$\tilde{\mathcal{D}}(\mathbb{P}_{\text{gen}}, \mathbb{P}_{\text{ref}}) = \int_{t_0}^{t_f} \text{MMD}^2(\mu_{e_x^t}, \mu_{e_y^t}) dt. \quad (3)$$

In empirical evaluations, however, we care about normalized values for ease of interpretation. We therefore analyze the properties of the MMD and derive a similarity score, which follows the same statistical principles as the MMD, and is bound between $[0, 1]$. We concluded that we can utilize only the MMD cross terms to evaluate the similarities. See Appendix Section B for more details. For our comparisons as referenced in Figure 6 we therefore denote the similarities rather than distances. We then come to the following formulation and refer to our empirical measure as the "Vector Embedding for Relational Trajectory Evaluation through Cross-similarity", or *VERTEX* score for short:

$$s(\mathbb{P}_{\text{gen}}, \mathbb{P}_{\text{ref}}) := \int_{t_0}^{t_f} [\min(\max(0, \frac{1}{z} \widetilde{\text{MMD}}^2(\mu_{e_x^t}, \mu_{e_y^t}) - z_{\text{rand}}), 1)] dt. \quad (4)$$

We approximate the integral across time steps through Monte Carlo approximation. The introduced normalization constants denote the similarities to a random sequence z_{rand} , which functions as a baseline subtraction to recenter our results, and a given reference score to rescale w.r.t. to scores obtained from comparing related solutions z . Min-max scaling ensures the final measure is bounded between $[0, 1]$. This process reflects properties such as Hölder continuity that bounds the kernel function within certain limits. To compute the embeddings, we utilize the embedding model `all-mpnet-base-v2` (Song et al., 2020), due to its widespread availability, and its balance between speed and quality. As a similarity measure, we select a Gaussian kernel following our derivation from the Appendix Section B. In our implementations, we also explore other kernels, including preliminary experiments with cosine similarity. We also note that one can integrate Bernoulli distributed trials into our score, with 0 values representing failure modes and values of 1 being successes. Furthermore, if we relax our definition, we can integrate other similarity measures which are bound between $[0, 1]$, which then reflect on domain-specific attributions, i.e. including a similarity measure tailored towards capturing the nuances between two sub-structures of abstract syntax tree.

7 EVALUATION

We introduce a benchmark that evaluates multi-step generative processes as complex workflows. Our benchmark consists of five different evaluation categories, and uses the VERTEX score to measure the capabilities of an LLM to solve tasks from each category. The five categories of our benchmark are divided into three basic evaluations and two advanced categories that combine different basic capabilities. The three basic categories are (i) **associative prediction** which measures a models proficiency in understanding associations between symbols, (ii) **multi-modal binding** where we evaluate the capability to employ tools and operate on different modalities, and (iii) **program synthesis** for measuring a models proficiency in generating consistent code and executing subroutines. The two advanced benchmark categories are (iv) **logic**, for evaluating logic-based components and (v) **computational graphs** where complex workflows need to be processed, evaluating all aforementioned capabilities. For our evaluation we focus on the GPT family (Brown et al., 2020) of models, specifically GPT-3.5 Turbo (revision 1106) and GPT-4 Turbo (revision 1106) as they are the most proficient models to date; Gemini-Pro (Google, 2023) as the best performing model available through API from Google; LLaMA2-Chat 13B (Touvron et al., 2023), LLaMA3-Chat 8B and LLaMA3-Chat 70B from Meta represent open-source LLMs. Finally, Mistral 7B (Jiang et al., 2023) and Zephyr 7B (Tunstall et al., 2023) serve as baselines for revised and fine-tuned open-source models. The open-source models Mistral, Zephyr, and smaller LLaMA variants are estimated to have roughly equivalent parameter counts compared to GPT-3.5 Turbo and Gemini-Pro. All our experiments require a context size smaller or equal to 4096 to enable the comparisons among the in-context capabilities across model architectures. For the LLaMA models, we utilize the *chat* versions since they are specifically tuned to follow instructions.

Associative Prediction We evaluate a model’s proficiency to follow simple and complex instructions and associations with zero- and few-shot examples. Therefore, we evaluate the proficiency in applying our operators between `Symbol` types. We defined a total of 15 tasks involving in-context associations between two `Symbol` instances. SymbolicAI’s overloaded operators rely on predefined pseudo-grammars, as described in Section 4, that augment the operators with few-shot examples. For instance, the overloaded operator `+` utilized between two `Symbol` instances provides few-shot examples how to resolve additions with various data types. Consequently, we can now test if the models can solve the addition between `Symbol` ("two hundred and thirty four") and `Symbol` (7000). See Appendix Section F.1 for more details.

Multi-modal Binding We perform transformations between multiple modalities through language-based representations. Therefore, we need to evaluate the model’s proficiency in tool utilization, classification and routing of requests to relevant modules. We define a multi-modal `Expression` to detect the category of a task based on its content and to forward the task to the appropriate tool. The expression creates interfaces to tools like WolframAlpha for mathematical expressions, Selenium for website content scraping, SerpApi for search queries, and APILayer for optical character recognition. Each of the five tests aims to evaluate the appropriate handling of a specific type of input by the multi-modal `Expression` type, such as processing a website URL for scraping, interpreting a search engine query, testing if two vectors are linearly independent, comparing large numbers, and extracting text from an image. See Appendix Section F.2 for more details.

Program Synthesis We evaluate executable code with and without concepts from retrieval augmented generation, model-driven development, and experiment with self-generated instructions by creating self-referential expressions. We designed three separate tests related to program synthesis, where each task assesses the ability of the models to generate and execute code based on natural language instructions or provided templates:

- 1) The first task involves reading a LaTeX table template and data, then generating a function to populate the table with the given data.
- 2) The second task tests the automatic code generation for API calls by fetching data from a specified URL and extracting specific information from the retrieved content.
- 3) The third task evaluates the ability to construct a custom `Expression` that processes a `Symbol` through a specific `Function` component from the `SymbolicAI` package.

Each of the three tests follows a similar pattern, where the generated code is scored based on its similarity to valid references and normalized with random samples. See Appendix Section F.3 for more details.

Logical Components To evaluate the capabilities for logical reasoning of models, we condition them to create a sequence of expressions as self-contained components, and refer to higher-order logic for their assessment. Based on the underlying *type theory* originating from Whitehead & Russell (1925–1927), we evaluate a models' capability to resolve statements in the form of *there exists x such that x fulfills y*. Such quantifiers define the standard semantics of expressions, where their meaning is given by a semantic function. A semantic function maps a term from an abstract definition to a point in a domain, which is an interpretation of the term's type and value. Therefore, these functions operate on types and values of expressions, and relations thereof. Subsequently, NeSy engines can formulate and evaluate at inference time logic-based instructions through Lisp, Prolog, or Mathematica (McCarthy, 1959; Colmerauer & Roussel, 1993; Chen et al., 1993; Inc., 2022), or leverage solvers such as Z3 (Moura & Bjørner, 2008). Therefore, the result of a natural language statement when evaluated by a NeSy engine can be interpreted by any expert system which defines the corresponding semantic functions and process them either in a symbolic (Feigenbaum et al., 1965; Gamble et al., 1994), differentiable (Veličković & Blundell, 2021; Ibarz et al., 2022), or hybrid manner (Kuncicky et al., 1991).

We evaluate how proficient models are at interpreting custom DSLs and define expression statements. DSLs are designed to express logical relations and operations in a structured format, and supports human-readable and machine-interpretable formulations. The following example illustrates such relationships by translating a natural language statement into an expression statement, as follows:

Marvins has four paws and likes to meow when I pet its fur. Is Marvins a cat?

A DSL may enforce the usage of `HAS(·)`, `IS(·)`, etc. and may condition an LLM to produce the following expressions:

- `HasFourPaws(x)`: *x* has four paws.
- `LikesToMeowWhenPetted(x)`: *x* likes to meow when it is petted.
- `IsCat(x)`: *x* is a cat.

These are then utilized to define the following logical expression:

$$\forall x (\text{HasFourPaws}(x) \wedge \text{LikesToMeowWhenPetted}(x) \Rightarrow \text{IsCat}(x)).$$

An automated theorem prover can now evaluate this statement for all values of *x* and assess the validity of the original query. Lastly, our evaluation uses symbolic mathematics to manipulate algebraic expressions. This involves defining symbols and performing operations like factorization, simplification, and algebraic manipulation. The symbols are placeholders for any value, enabling the definition of general expressions without specifying their values upfront.

We designed six tests to assess the logical capabilities of the candidate models and group them as follows. See Appendix Section F.4 for more details.

- 1) We utilize the Python library SymPy for symbolic mathematics to create the mathematical expression $ax + bx - cx - ay - by + cy + d$. The task for the model is then to factorize the expression and extract all unique symbols as a list.
- 2) Three tasks evaluate a models' capability to resolve the logical operations AND, OR, and XOR. For instance, the test for logical AND combines the symbols `Symbol("The horn only sounds on Sundays")` and `Symbol("I hear the horn")` and compares the answer against the human-generated references "The horn only sounds on Sundays and I hear the horn." and "Since I hear the horn it is Sunday." Since there is a large number of possible solutions, there is high variability in the solution space. Each model might prefer a different solution.

3) For another task we use a custom Expression that defines a DSL syntax and semantic structure. We use this Expression to extract higher-order logic expressions from a natural language statement, namely the puzzle 'Who is Jay's brother?'⁴, that preserves the original relationships.

4) For the final task, we again use the puzzle 'Who is Jay's brother?' to evaluate a models' capability for complex conversions. We use the Z3 theorem prover (Moura & Björner, 2008) to solve the 'Who is Jay's brother' puzzle conditioned on the Z3 solvers' solution to Einstein's famous puzzle 'Who owns the fish?'. The task involves an indirect translation from natural language to executable code by the Z3 solver; the solution to Einstein's puzzle acts as a form of self-contained "documentation" for how the Z3 solver should be utilized. The test constructs a template, which includes the task instructions, puzzle statement, and reference to the Einstein's puzzle solution. The models are then asked to analyze the given problem and solution format and create a Python function with Z3 syntax that can solve the 'Who is Jay's brother?' puzzle. The dynamically generated code is executed within the test environment utilizing Python's `exec` function. We check the access to the Z3 solver and run the generated `solve_puzzle` function supposed to contain the logic to solve the puzzle. Once executed, the assembled Z3 logical clauses are processed by the solver, which verifies that the set of constraints is satisfiable. If so, the model generated by the solver is queried for the puzzle's solution and scored using our VERTEX score.

Hierarchical Computational Graphs We evaluate the capabilities of models to orchestrate a multi-step generative process and evaluate a set of tasks. Models need to direct sub-processes and associate computational results from and to `Symbol` nodes, and maintain relationships between these nodes, which we refer to as a computational graph as shown in Figure 5. In a computational graph, the VERTEX score compares the results produced by a generative model at each node against samples obtained from a reference distribution, usually modeled by sampling from multiple valid references. We also account for randomness through predefined random samples for normalizing the result. Our reference to *hierarchical* computational graphs stems from the fact that we operate on multiple levels. On a higher level of abstraction we are able to perform planning, sub-task scheduling, and define operational instructions. On a lower level of abstraction, we execute these plans based on the defined instructions and data, which can also span generative processes that produce new information.

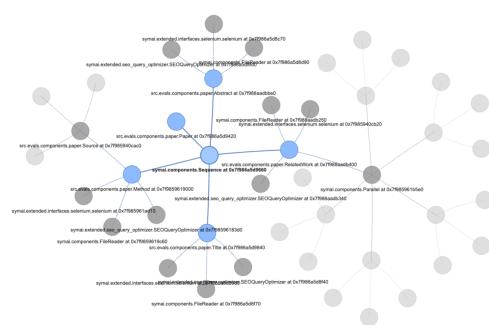


Figure 5: We illustrate the hierarchical computational graph for the Paper expression. Each node represents an instance of an expression with distinct properties and behaviors, such as file sourcing, generative process, tool utilization, or transformation operation. The edges denote the reference relationships between expressions and indicate the flow of information. The blue highlighted nodes mark the main sequence nodes of expressions utilized to create parts of the paper draft, such as `Method` section, `RelatedWork` section, `Abstract` section, and so on. Each generative node is used for evaluating the VERTEX score. Non-generative nodes such as search engine results are not evaluated, and we assume to obtain ground-truth values.

Given that the field is currently at an early stage in developing even sequential schedulers for LLM-based planning systems, our evaluations will be confined to sequential execution only. We introduce two tests designed to evaluate multi-step generative processes:

- 1) We simulate and evaluate the process of writing a research paper draft based on a predefined hierarchical computational graph that focuses on the content output of the computational graph rather than planning and scheduling functionality. See Appendix Section F.5 for more details.
- 2) We test the VERTEX Protocol as defined in Algorithm 1, which represents our general method for evaluating multi-step generative processes. We create a self-contained test scenario to illustrate an end-to-end evaluation and as a go-to reference for how our protocol can be deployed in a realistic environment. Our evaluation protocol is not only designed to analyze and score a series of instructions, but also to provide a structured basis for recording these

⁴ Bob has two sons, John and Jay. Jay has one brother and father. The father has two sons. Jay's brother has a brother and a father. Who is Jay's brother?

processes. Furthermore, we note that our evaluation protocol is generally formulated, which allows the application of non-sequential planning and scheduling.

Algorithm 1 VERTEX Protocol

Require: NeSy engine \mathcal{V} as an LLM, embedding engine $\mathcal{E} : \Sigma \rightarrow \mathcal{H} \subset \mathbb{R}^d$, symbols $\{x_0, x^*, y^*\} \subset \Sigma$, with x_0 as the initial instruction, x^* as the payload resulted from executing \mathcal{V} , y^* as the reference, and $*$ acting as a placeholder for $\mathcal{P}, \mathcal{T}, \mathcal{C}$, capabilities $\mathcal{C} = \{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \dots\}$, where each \mathcal{F}_i represents a specific functional role within the system, plan $\mathcal{P} \subset \Sigma$, task $\mathcal{T} \in \mathcal{P}$, memory buffer $\mathcal{M} \subset \Sigma$, a scoring function $\tilde{s} : \mathcal{H} \times \mathcal{H} \rightarrow [0, 1]$, a scheduler \mathcal{Q} , an aggregator \mathcal{A} , and score variables $\{s\} \in [0, 1]$.

Method:

```

1:  $\mathcal{V}, \mathcal{E}, \mathcal{Q}, \mathcal{C}, y^{\mathcal{P}} \leftarrow \text{INIT}(\cdot)$                                  $\triangleright$  Initialize engines, scheduler, capabilities, expected plan.
2:  $\mathcal{M} \leftarrow \emptyset, \mathcal{A} \leftarrow \emptyset$                                       $\triangleright$  Initialize memory buffer and aggregator.
3:  $x^{\mathcal{P}} \leftarrow \text{GENERATEPLAN}(x_0, \mathcal{V})$                                 $\triangleright$   $\mathcal{V}$  generates plan based on initial instruction.
4:  $\text{EVALUATE}(x^{\mathcal{P}}, y^{\mathcal{P}}, \mathcal{E}, \mathcal{A}, \tilde{s})$                             $\triangleright$  Embed, score, and aggregate plan similarity.
5:  $\mathcal{P}, \mathcal{M} \leftarrow \text{UNFOLDPLAN}(y^{\mathcal{P}}, \mathcal{M}, \mathcal{Q})$                        $\triangleright$   $\mathcal{Q}$  unfolds plan into actionable tasks and updates progression.
6: while  $\mathcal{P} \neq \emptyset$  do                                                  $\triangleright$  Run until list of tasks is exhausted.
7:    $\mathcal{T}, y^{\mathcal{C}}, y^{\mathcal{T}} \leftarrow \text{SELECT}(\mathcal{M}, \mathcal{V})$                           $\triangleright$   $\mathcal{V}$  selects next task based on task progression.
8:    $\mathcal{F}_i \leftarrow \text{IDENTIFY}(\mathcal{T}, \mathcal{C}, \mathcal{V})$                                  $\triangleright$   $\mathcal{V}$  identifies task-related capability  $\mathcal{F}_i$ .
9:    $x^{\mathcal{C}}, x^{\mathcal{T}} \leftarrow \text{EXECUTE}(\mathcal{T}, \mathcal{F}_i, \mathcal{Q})$                        $\triangleright$   $\mathcal{Q}$  executes  $\mathcal{T}$  with capability  $\mathcal{F}_i$  and assign results  $x^{\mathcal{C}}, x^{\mathcal{T}}$ .
10:   $\text{EVALUATE}(x^{\mathcal{C}}, y^{\mathcal{C}}, x^{\mathcal{T}}, y^{\mathcal{T}}, \mathcal{E}, \mathcal{A}, \tilde{s})$             $\triangleright$  Embed, score, and aggregate capability similarity.
11:   $\mathcal{P}, \mathcal{M} \leftarrow \text{UPDATE}(\mathcal{T}, \mathcal{P}, \mathcal{M}, \mathcal{Q})$                        $\triangleright$   $\mathcal{Q}$  updates plan and progression.
12: end while
13:  $s \leftarrow \text{FINALIZE}(\mathcal{A})$                                           $\triangleright$  Finalize aggregation of scores.
14: return  $s$                                                         $\triangleright$  Return aggregated score of plan execution.

```

Algorithm 1: This algorithm defines the pseudocode of our VERTEX protocol with our respective VERTEX score as a scoring criteria. We start by initializing the NeSy engine \mathcal{V} , the embedding engine \mathcal{E} , the scheduler \mathcal{Q} , and a set of capabilities \mathcal{C} . The initial instruction x_0 is utilized to generate a plan $x^{\mathcal{P}}$ through \mathcal{V} . The plan and its expected outcome $y^{\mathcal{P}}$ are embedded, and their similarity is scored according to our VERTEX score and aggregated. The plan is then unfolded into actionable tasks. Each task \mathcal{T} is selected and executed with the appropriate capability \mathcal{C} , resulting in the capability and task results $x^{\mathcal{C}}, x^{\mathcal{T}}$, and expected outcomes $y^{\mathcal{C}}, y^{\mathcal{T}}$ updated in the memory buffer \mathcal{M} . The process continues, with each task's result being embedded, scored, and aggregated until the plan is complete. The final aggregated score s is returned, reflecting the overall effectiveness of the plan execution.

We start with a high-level workflow description which consists of a list of tasks and optionally their respective sub-tasks; we refer to this as the plan \mathcal{P} . To perform the experiment, we utilize an expected plan $y^{\mathcal{P}}$ which was handcrafted for this evaluation. The expected plan is a queue of predefined tasks (in a particular order) that the system should follow to achieve the goal. The goal statement defines the end objective that the workflow aims to accomplish. We also have a set of plans similar to the expected plan, which are trajectories in the solution space, as well as the plan $x^{\mathcal{P}}$ that the LLM generates utilizing the GENERATEPLAN call for a specific seed. We score the predicted plan against the expected plan and the trajectories, then we continue to the next phase in which we utilize the expected plan to execute the tasks. At each step, the LLM will receive in its context the goal, the tasks, the current progress, and a query asking for the next task to execute; we refer to this as the memory buffer \mathcal{M} . If the LLM is not able to predict the next task, it will return a failure, and the expected plan will be utilized to execute the next task. The LLM has access to a predefined set of capabilities \mathcal{C} , specifically WolframAlpha, SerpApi, Selenium, and the LLM itself, which also represents our self-referential structure. We keep executing tasks until the queue is exhausted, and at each step, we utilize the EVALUATE call to measure the performance of the LLM with our VERTEX score. The scheduler class \mathcal{Q} oversees the execution of the test workflow. It takes the setup configuration and orchestrates the linear execution of tasks, utilizing the expected plan as a reference. It maintains a pool of tasks to be executed and updates progress as tasks are completed. The UNFOLDPLAN call is a method of the scheduler class \mathcal{Q} . The method calls itself recursively until there are no tasks left. The SELECT call is responsible for determining which task to execute next from a pool of remaining tasks. It utilizes the LLM through self-reflection (Shinn et al., 2023) to choose the most suitable next task based on a template that gets progressively updated in the memory buffer \mathcal{M} by the UPDATE call. The IDENTIFY call uses self-reflection and similarity scoring to determine the best interface based on the task at hand, then passes the interface to the EXECUTE call to execute the task. Lastly, the test ends with the FINALIZE call, which provides an aggregated assessment of the model's ability to manage and execute the workflow.

In Figure 6 we conclude with our evaluation and compute the cumulative score for all described evaluation categories and in the next section we discuss how to interpret the results of our framework.

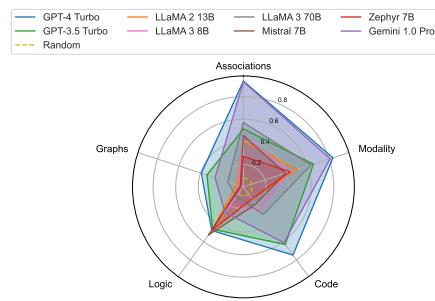


Figure 6: We evaluate GPT-4 Turbo, GPT-3.5 Turbo, Gemini-1.0 Pro, LLaMA2-Chat 13B, LLaMA3-Chat 8B, LLaMA3-Chat 70B, Mistral 7B and Zephyr 7B on five benchmark categories: 1) Associative Prediction (Association) 2) Multi-modal Binding (Modality) 3) Program Synthesis (Code) 4) Functional Logic Components (Logic) and 5) Hierarchical Computational Graphs (Graphs). We denote the VERTEX scores for each category as a normalized value between 0 and 1, where higher values are better. The VERTEX score is measured according to a reference baseline and normalized by random sequences to exclude noise and similarities among references distributions to rescale solutions. The shown scores are an average over all tests per category and across 8 different seeds per test.

Benchmarks	GPT-4 Turbo	GPT-3.5 Turbo	Gemini 1.0 Pro	LLaMA 2 13B	LLaMA 3 8B	LLaMA 3 70B	Mistral 7B	Zephyr 7B	Random
Associations	0.94	0.51	0.93	0.40	0.46	0.57	0.46	0.27	0.08
Modality	0.83	0.65	0.81	0.50	0.43	0.63	0.39	0.43	0.07
Code	0.75	0.63	0.61	0.13	0.25	0.30	0.19	0.13	0.00
Logic	0.48	0.46	0.28	0.46	0.21	0.11	0.53	0.47	0.00
Graphs	0.39	0.34	0.26	0.06	0.05	0.15	0.03	0.03	0.00
Total	0.68	0.52	0.58	0.31	0.28	0.35	0.32	0.27	0.03

8 DISCUSSION

In this section, we address the evaluation results, auxiliary findings and limitations of SymbolicAI and the future directions we are focusing on. Some of the limitations stem from the inherent constraints of current technologies and dependencies on third-party systems. Additionally, the nuanced complexities of working with generative models presents further challenges.

8.1 RESULTS

In Figure 6 we show the VERTEX score for all five evaluation categories on 8 different state-of-the-art models. We show the aggregated results per category, meaning the average score among all tests averaged per category and average across 8 different seeds per test. The VERTEX score is normalized between 0 and 1, where higher values are better. Our score is non-linear due to its nature of using non-linear kernels, and captures semantic, ordinal and relative structures among the data samples. However, since our score is highly dependent on the quality of the underlying embedding model, it may omit to capture fine-grained syntactic differences such as ‘Hello’ vs ‘hello’.

In our experiments, we have noticed that for associative predictions and multi-modal bindings, GPT-4 Turbo is on par with Gemini-1.0 Pro. Furthermore, there is still a large gap between open-source contestants such as LLaMA 3 even with 70B parameters compared to the closed-source alternatives from OpenAI and Google. For the rest of the experiments, we see that GPT-4 almost always outperforms all other models, except for the functional logic components category. Here, we analyzed results and found that the larger models sometimes take shortcuts by automatically returning the solution and answering that the task instructions are too complex for such a straight-forward puzzle query. However, we would rather state in general that for logic-based, planning and scheduling tasks all models act unreliable, even if slight performance differences between the models are seen in the plot. We believe this is in part due to lack of training data specifically for workflows, planning and scheduling tasks, and to imprecision in generating reliably structured output formats, such as custom DSLs or other custom in-context instructed formats. This also stems from their instruction fine-tuning, since most models are chat-based models and offer verbose responses which need to be suppressed or post-processed.

We see similar performance between GPT-3.5 Turbo and LLaMA 3 70B except for the logical and graphs evaluations. We found that LLaMA 3 70B has a tendency to ask questions back if it does not understand the request instead of following the specified instructions provided. We assume this also stems from the chat-based instruction fine-tuning. Zephyr 7B and Mistral 7B have shown on par capabilities in functional logic components with larger models, however

fail in program synthesis and hierarchical computational graphs experiments. We observe that they perform well when resolving the overloaded logic operators such as OR, AND and XOR, and show decent performance for text generation, but fail to resolve more complex instructions.

8.2 LIMITATIONS

Framework Since the framework interfaces with many tools and API services, it requires a substantial engineering feat to integrate all available functionalities and keep the API-based services up-to-date. For us this means, that although we support a variety of tools and frameworks like Selenium, WolframAlpha, or Z3, we only scratch the surface of these tools. Moreover, the utilization of grammar-based constraints validations is still experimental and limited in functionality for specific formats such as JSON and HTML. Finally, we encounter also challenges related to engineering parallelization and multiprocessing of prompts, since the concurrent execution is non-trivial, especially with intricacies of Python process management.

Embedding Measure Our empirical measure is limited by the expressiveness of the embedding model and how well it captures the nuances in similarities between two representations. Furthermore, the obtained similarity scores are highly non-linear and difficult to interpret. For instance, two representations may address the same topic, such as the problem description and its respective solution, however, when measuring their similarity we obtain similarity scores of $\sim 70\%$. We normalize this by subtracting an inherent baseline and randomness effect, however, to ensure a more holistic and robust measurement we would need a significantly larger amount of baselines and experiments. Since we were very limited in the availability of development resources, and some presented models are only addressable through costly API walls. We are actively seeking sponsors to scale our solution and offer a more compelling benchmark suite in the future.

Model Capabilities An obvious limitation revolves around the fixed context window size of the underlying language models. Despite the expansion of the context window in newer models such as GPT-4, the finite context still restricts the amount of data that can be processed in a single pass. All information outside the context needs to be added through information retrieval approaches, which come with their own challenges and limitations (Gao et al., 2023). This leads to side effects, including hallucination, given the model does not contain the necessary information to answer the prompted instruction, which makes it difficult to maintain long-term statefulness for complex reasoning tasks and computational graphs.

Error Handling The complexity of error handling when evaluating complex expressions through function compositionality, especially between multiple modalities and different solvers, is another notable challenge. While SymbolicAI introduces mechanisms for error analysis and automated correction, these approaches are not infallible. They are often limited by the quality and expressiveness of the models, and the model's capacity to understand deeply nested logical constructs. We also note that for our evaluations, we disabled any remedy protocol, such as truncating prompts or retry schema.

Generalization This research is also limited by current LLM's capacity for reasoning and generalization. Although progress has been made, models are still prone to hallucinations and reasoning errors, especially when dealing with abstract, novel, or highly complex problem statements (Marcus, 2020). Furthermore, our framework's reliance on the model's ability to grasp the semantics of operations can be influenced by the training data and the model's innate biases and conceptual understanding (Mahowald et al., 2023). We also point out that the initial development of SymbolicAI started with the GPT family of models, and we may encounter innate biases in prompt design and expressiveness when utilizing other reference models. However, we also point out that prompt engineering instruction-based statements is not a reliable direction for improvement. We instead advocate for enhancing the resilience of models through fault tolerance, focusing on their ability to better follow semantic instructions, not syntactic idiosyncrasies. Another concern is how to assess the disentanglement of evaluations of models on downstream tasks, to avoid testing on training samples, especially for closed-source solutions like GPT.

Interpretability and Transparency Finally, the issue of explainability and transparency in AI systems remains challenging. While SymbolicAI makes steps towards making computational processes more explicit and explainable through symbolic manipulations, understanding the internal logic and decision-making of LLMs remains an open problem. This can hinder trust and adoption in sensitive applications where interpretability of predictions is important.

8.3 FUTURE WORK

The goal for Algorithm 1 is to be utilized by an advanced learning agent. This agent, employing reinforcement learning methodologies (Ouyang et al., 2022; Li et al., 2023; Rafailov et al., 2023), could leverage our evaluation measure in existing benchmarks (Milani et al., 2020; Swazinna et al., 2022; Schweighofer et al., 2022) as a means to obtain reward signals to address a central problem in reinforcement learning, namely credit assignment (Sutton, 1984; Arjona-Medina et al., 2019; Holzleitner et al., 2020; Patil et al., 2020; Widrich et al., 2021; Dinu et al., 2022). Over time, it aims to develop the ability to autonomously generate its own plans, efficiently schedule tasks and subtasks, and carefully select the most suitable tools for each task. Our protocol lays the groundwork for this agent to learn and expand its base set of capabilities (Amaro et al., 2023), moving towards more sophisticated, self-referential orchestration of multi-step tasks. We've already noticed that research is shifting towards this type of methodology (Yuan et al., 2024). Furthermore, in Section 7 we've only considered a sequential scheduler. However, our objective is to ultimately assess a non-sequential task execution model, allowing for dynamic insertion and out-of-sequence task execution. In addition, we are interested into exploring similarities of our work with *Generative Flow Networks* (Bengio et al., 2021a,b; Lahou et al., 2023). Lastly, we also discuss limitations in Appendix Section 8.2 with further opportunities for future improvements.

9 CONCLUSION

In this work, we introduced SymbolicAI, a framework that unifies generative models with an array of solvers, blending the strengths of symbolic and sub-symbolic AI paradigms within a cohesive NeSy framework. SymbolicAI equips researchers and practitioners with a comprehensive toolkit to develop contextualized and explainable NeSy AI systems capable of addressing diverse challenges effectively. We also introduce a quality measure and a benchmark for comparing and evaluating a wide range of computational tasks. SymbolicAI provides a basis for further research in advanced program synthesis, hierarchical computational graphs, the development of self-referential systems, and the integration of probabilistic methods with AI design for creating autonomous agents.

ACKNOWLEDGEMENT

The ELLIS Unit Linz, the LIT AI Lab, the Institute for Machine Learning, are supported by the Federal State Upper Austria. We thank the projects Medical Cognitive Computing Center (MC3), INCONTROL-RL (FFG-881064), PRIMAL (FFG-873979), S3AI (FFG-872172), DL for GranularFlow (FFG-871302), EPILEPSIA (FFG-892171), AIRI FG 9-N (FWF-36284, FWF-36235), AI4GreenHeatingGrids (FFG- 899943), INTEGRATE (FFG-892418), ELISE (H2020-ICT-2019-3 ID: 951847), Stars4Waters (HORIZON-CL6-2021-CLIMATE-01-01). We thank Audi.JKU Deep Learning Center, TGW LOGISTICS GROUP GMBH, Silicon Austria Labs (SAL), FILL Gesellschaft mbH, Anyline GmbH, Google, ZF Friedrichshafen AG, Robert Bosch GmbH, UCB Biopharma SRL, Merck Healthcare KGaA, Verbund AG, GLS (Univ. Waterloo), Software Competence Center Hagenberg GmbH, Borealis AG, TÜV Austria, Frauscher Sensonic, TRUMPF, the NVIDIA Corporation and Atlas.

We extend our appreciation to Andreas Windisch and Clemens Wasner of AI Austria for their unwavering support. Their valuable feedback, connections, and facilitation of introductions within their expansive network have been instrumental to the progress of ExtensityAI.

Our gratitude also goes to Sergei Pereverzyev, whose enlightened guidance and thoughtful ideas have been a beacon for our research endeavors. Our thanks are equally extended to Gary Marcus, whose stimulating discussions sparked numerous innovative ideas incorporated into our framework.

We are equally grateful to Markus Hofmarcher, a friend and colleague whose informed counsel and stimulating discussions have significantly sharpened various facets of our study. Additionally, our thanks are due to Fabian Paischer and Kajetan Schweighofer, whose preliminary work and assistance have been of enormous benefit.

We are also grateful to our friends John Chong Min Tan and Tim Scarfe, whose communities have been a hub for exhilarating discussions. Their online presence and engagement have enriched the AI research landscape and broadened our perspectives.

Moreover, we wish to honor the memories of the cherished family members we lost in 2023. Their influence in our lives extended beyond personal bonds, and the principles they instilled in us continue to shape our journey. It is with great respect and affection that we acknowledge the indelible impact they have made, enabling us to persist in our scientific pursuits with determination and integrity.

REFERENCES

- M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esen, A. A. S. Awwal, and V. K. Asari. The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*, 2018.
- R. E. Amaro, J.-Y. Chen, J. M. Duarte, T. E. Hutton, C. Irving, M. C. Kandes, A. Majumdar, D. Y. Mishin, M. H. Nguyen, P. Rodriguez, F. Silva, R. S. Sinkovits, S. M. Strande, M. Tatineni, L. S. Tran, and N. Wolter. Voyager – an innovative computational resource for artificial intelligence & machine learning applications in science and engineering. In *Practice and Experience in Advanced Research Computing, PEARC ’23*, pp. 278–282, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450399852. doi: 10.1145/3569951.3597597.
- Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. SPICE: semantic propositional image caption evaluation. *CoRR*, abs/1607.08822, 2016. URL <http://arxiv.org/abs/1607.08822>.
- J. Andreas. Language models as agent models. *CoRR*, abs/2212.01681, 2022. doi: 10.48550/arXiv.2212.01681.
- J. Arjona-Medina, M. Gillhofer, M. Widrich, T. Unterthiner, J. Brandstetter, and S. Hochreiter. RUDDER: return decomposition for delayed rewards. In *Advances in Neural Information Processing Systems 32*, pp. 13566–13577, 2019.
- M. Assran, Q. Duval, I. Misra, P. Bojanowski, P. Vincent, M. Rabbat, Y. LeCun, and N. Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629, 2023.
- L. M. Augusto. *Computational Logic. Vol. 1: Classical Deductive Computing with Classical Logic*. College Publications, London, 2 edition, 2022.
- F. Badita. *1337 Use Cases for ChatGPT & other Chatbots in the AI-Driven Era*. Google Docs, 2022.
- D.M. Beazley. *Python Essential Reference*. Developer’s library : essential references for programming professionals. Addison-Wesley, 2009. ISBN 9780672329784. URL <https://books.google.ro/books?id=Chr1NDlUcI8C>.
- M. Beck, K. Pöppel, M. Spanring, A. Auer, O. Prudnikova, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter. xlstm: Extended long short-term memory, 2024.
- E. Bengio, M. Jain, M. Korablyov, D. Precup, and Y. Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021a.
- Y. Bengio, T. Deleu, E. J. Hu, S. Lahou, M. Tiwari, and E. Bengio. Gflownet foundations. *arXiv preprint arXiv:2111.09266*, 2021b.
- T. R. Besold, A. d. Garcez, S. Bader, H. Bowman, P. Domingos, P. Hitzler, K.-U. Kuehnberger, L. C. Lamb, D. Lowd, P. M. V. Lima, L. de Penning, G. Pinkas, H. Poon, and G. Zaverucha. Neural-symbolic learning and reasoning: A survey and interpretation, 2017.
- M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, L. Gianinazzi, J. Gajda, T. Lehmann, M. Podstawski, H. Niewiadomski, P. Nyczyk, and T. Hoefer. Graph of thoughts: Solving elaborate problems with large language models. *arXiv preprint arXiv:2308.09687*, 2023.
- S. Biderman, H. Schoelkopf, Q. Anthony, H. Bradley, K. O’Brien, E. Hallahan, M. Aflah Khan, S. Purohit, S. Prashanth, E. Raff, A. Skowron, L. Sutawika, and O. van der Wal. Pythia: A suite for analyzing large language models across training and scaling, 2023.
- PENG Bo. Blinkdl/rwkv-lm: 0.01. Technical report, Zenodo, August 2021. URL <https://doi.org/10.5281/zenodo.5196577>.
- W. Bradley Knox and Peter Stone. TAMER: Training an Agent Manually via Evaluative Reinforcement. In *2008 7th IEEE International Conference on Development and Learning*, pp. 292–297, Monterey, CA, August 2008. IEEE. ISBN 978-1-4244-2661-4. doi: 10.1109/DEVLRN.2008.4640845.
- J. D. Bransford and M. K. Johnson. Contextual prerequisites for understanding: Some investigations of comprehension and recall. *Journal of Verbal Learning and Verbal Behavior*, 11(6):717–726, 1972. ISSN 0022-5371.

- T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020.
- H. Chase. LangChain. Technical report, LangChain, 01 2023. URL <https://github.com/hwchase17/langchain>.
- W. Chen, M. Kifer, and D. S. Warren. Hilog: A foundation for higher-order logic programming. *The Journal of Logic Programming*, 15(3):187–230, 1993. ISSN 0743-1066.
- F. Chollet. On the measure of intelligence, 2019.
- N. Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124, 1956. doi: 10.1109/TIT.1956.1056813.
- P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.
- A. Colmerauer and P. Roussel. The birth of Prolog. In *HOPL-II*, 1993.
- A. d'Avila Garcez and L. C. Lamb. Neurosymbolic ai: The 3rd wave. *arXiv preprint arXiv:2012.05876*, 2020.
- A. d'Avila Garcez, M. Gori, L. C. Lamb, L. Serafini, M. Spranger, and S. N. Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *Journal of Applied Logic*, 2019.
- A. Dawid and Y. LeCun. Introduction to latent variable energy-based models: A path towards autonomous machine intelligence. *arXiv preprint arXiv:2306.02572*, 2023.
- J. Degrave. Building A Virtual Machine inside ChatGPT. Technical report, Engraved, 11 2022. URL <https://www.engraved.blog/building-a-virtual-machine-inside/>.
- F. Deniz, A. O. Nunez-Elizalde, A. G. Huth, and J. L. Gallant. The representation of semantic information across human cerebral cortex during listening versus reading is invariant to stimulus modality. *Journal of Neuroscience*, 39(39):7722–7736, 2019. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.0675-19.2019.
- D. C. Dennett. Real patterns. *Journal of Philosophy*, 88(1):27–51, 1991. doi: 10.2307/2027085.
- M. Dilhara, A. Ketkar, and D. Dig. Understanding software-2.0: A study of machine learning library usage and evolution. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 30(4):55:1–55:42, jul 2021. ISSN 1049-331X. doi: 10.1145/3453478.
- M.-C. Dinu, M. Hofmarcher, V. P. Patil, M. Dorfer, P. M. Blies, J. Brandstetter, J. A. Arjona-Medina, and S. Hochreiter. Xai and strategy extraction via reward redistribution. In A. Holzinger, R. Goebel, R. Fong, T. Moon, K.-R. Müller, and W. Samek (eds.), *xxAI - Beyond Explainable AI: International Workshop, Held in Conjunction with ICML 2020, July 18, 2020, Vienna, Austria, Revised and Extended Papers*, pp. 177–205, Cham, 2022. Springer International Publishing. ISBN 978-3-031-04083-2. doi: 10.1007/978-3-031-04083-2_10.
- M.-C. Dinu, M. Holzleitner, M. Beck, H. D. Nguyen, A. Huber, H. Eghbal-zadeh, B. A. Moser, S. V. Pereverzyev, S. Hochreiter, and W. Zellinger. Addressing parameter choice issues in unsupervised domain adaptation by aggregation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.
- I. Donadello, L. Serafini, and A. d'Avila Garcez. Logic tensor networks for semantic image interpretation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pp. 1596–1602, 2017.
- D. C. Dowson and B. V. Landau. The fréchet distance between multivariate normal distributions. *Journal of Multivariate Analysis*, 12(3):450–455, 1982. doi: [https://doi.org/10.1016/0047-259X\(82\)90077-X](https://doi.org/10.1016/0047-259X(82)90077-X).
- Kevin Ellis. Human-like few-shot learning via bayesian reasoning over natural language. *arXiv preprint arXiv:2306.02797*, 2023.

- M. Fang, S. Deng, Y. Zhang, Z. Shi, L. Chen, M. Pechenizkiy, and J. Wang. Large language models are neurosymbolic reasoners. *arXiv preprint arXiv:2401.09334*, 2024.
- E. Fedorenko, P.-J. Hsieh, A. Nieto-Castanon, S. Whitfield-Gabrieli, and N. Kanwisher. New method for fMRI investigations of language: Defining rois functionally in individual subjects. *Journal of neurophysiology*, 104:1177–94, 08 2010. doi: 10.1152/jn.00032.2010.
- E. Feigenbaum, B. G. Buchanan, J. Lederberg, Carl Djerassi, and et al. Dendral, 1965.
- C. Fernando, D. Banarse, H. Michalewski, S. Osindero, and T. Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*, 2023.
- R. F. Gamble, G.-C. Roman, H. C. Cunningham, and W. E. Ball. Applying formal verification methods to rule-based programs. *Int. J. Expert Syst.*, 7(3):203–237, sep 1994. ISSN 0894-9077.
- S. Ganguly and V. Pudi. Paper2vec: Combining graph and text information for scientific paper representation. In Joemon Jose et al. (eds.), *Advances in Information Retrieval*, volume 10193 of *Lecture Notes in Computer Science*. Springer, Cham, 2017. ISBN 978-3-319-56607-8. doi: 10.1007/978-3-319-56608-5_30.
- Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, and H. Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.
- A. Garcez, T. Besold, L. De Raedt, P. Földiák, P. Hitzler, T. Icard, K. Kühnberger, L. Lamb, R. Miikkulainen, and D. Silver. Neural-symbolic learning and reasoning: Contributions and challenges. In *AAAI Conference*, 2015.
- M. Gauch, M. Beck, T. Adler, D. Kotsur, S. Fiel, H. Eghbal-zadeh, J. Brandstetter, J. Kofler, M. Holzleitner, W. Zellinger, D. Klotz, S. Hochreiter, and S. Lehner. Few-Shot Learning by Dimensionality Reduction in Gradient Space. *arXiv preprint arXiv:2206.03483*, 2022.
- X. Geng, A. Gudibande, H. Liu, E. Wallace, P. Abbeel, S. Levine, and D. Song. Koala: A dialogue model for academic research. Blog post, April 2023. URL <https://bair.berkeley.edu/blog/2023/04/03/koala/>.
- Google. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
- A. Goyal, A. Friesen, A. Banino, T. Weber, N. R. Ke, A. P. Badia, A. Guez, M. Mirza, P. C. Humphreys, K. Konyushova, M. Valko, S. Osindero, T. Lillicrap, N. Heess, and C. Blundell. Retrieval-augmented reinforcement learning. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 7740–7765. PMLR, 17–23 Jul 2022.
- A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012.
- A. Gu and T. Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.
- Q. Guo, Z. Jin, X. Qiu, W. Zhang, D. Wipf, and Z. Zhang. CycleGT: Unsupervised graph-to-text and text-to-graph generation via cycle training. *arXiv preprint arXiv:2006.04702*, 2020.
- K. Hamilton, A. Nayak, B. Božić, and L. Longo. Is neuro-symbolic AI meeting its promises in natural language processing? a structured review. *Semantic Web*, pp. 1–42, nov 2022. doi: 10.3233/sw-223228.
- C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020. doi: 10.1038/s41586-020-2649-2.
- M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS’17, pp. 6629–6640, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.
- S. Hochreiter. Toward a broad AI. *Commun. ACM*, 65(4):56–57, mar 2022. ISSN 0001-0782.
- S. Hochreiter and J. Schmidhuber. Flat minima. *Neural Comput.*, 9(1):1–42, 1997.

- M. Holzleitner, L. Gruber, J. A. Arjona-Medina, J. Brandstetter, and S. Hochreiter. Convergence proof for actor-critic methods applied to PPO and RUDDER. *arXiv preprint arXiv:2012.01399*, 2020.
- K. Hornik, M. Tinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989. doi: 10.1016/0893-6080(89)90020-8.
- J. Hu, H. Small, H. Kean, A. Takahashi, L. Zekelman, D. Kleinman, E. Ryan, A. Nieto-Castañón, V. Ferreira, and E. Fedorenko. Precision fMRI reveals that the language-selective network supports both phrase-structure building and lexical access during language production. *bioRxiv*, 2022. doi: 10.1101/2021.09.10.459596.
- Z. Hu, X. Ma, Z. Liu, E. Hovy, and E. Xing. Harnessing deep neural networks with logic rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2410–2420, Berlin, Germany, August 2016. Association for Computational Linguistics.
- A. G. Huth, W. A. de Heer, T. L. Griffiths, F. E. Theunissen, and J. L. Gallant. Natural speech reveals the semantic maps that tile human cerebral cortex. *Nature*, 532(7600):453–458, 2016. doi: 10.1038/nature17637.
- B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, and D. Amodei. Reward learning from human preferences and demonstrations in atari. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- B. Ibarz, V. Kurin, G. Papamakarios, K. Nikiforou, M. Abbana Bennani, R. Csordás, A. Dudzik, M. Bošnjak, A. Vitvitskyi, Y. Rubanova, A. Deac, B. Bevilacqua, Y. Ganin, C. Blundell, and P. Velivčković. A generalist neural algorithmic learner. In *LOG IN*, 2022.
- Wolfram Research, Inc. Mathematica, Version 13.2, 2022. URL <https://www.wolfram.com/mathematica>. Champaign, IL.
- G. Indiveri, B. Linares-Barranco, T. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S. Liu, P. Dudek, P. Häfliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. SA GHI, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, and K. Boahen. Neuromorphic silicon neuron circuits. *Frontiers in Neuroscience*, 5, 2011. ISSN 1662-453X. doi: 10.3389/fnins.2011.00073.
- N. Jain, S. Vaidyanath, A. Iyer, N. Natarajan, S. Parthasarathy, S. Rajamani, and R. Sharma. Jigsaw: Large language models meet program synthesis. *arXiv*, 2021.
- A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. Le Scao, T. Lavril, T. Wang, T. Lacroix, and W. El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.
- J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.
- E. Jones and J. Steinhardt. Capturing failures of large language models via human cognitive biases. *arXiv preprint arXiv:2202.12299*, 2022.
- A. Karpathy. Software 2.0. Medium, 2017. URL <https://karpathy.medium.com/software-2-0-a64152b37c35>.
- N. Kassner, B. Krojer, and H. Schütze. Are Pretrained Language Models Symbolic Reasoners over Knowledge? In R. Fernández and T. Linzen (eds.), *Proceedings of the 24th Conference on Computational Natural Language Learning, CoNLL 2020, Online, November 19-20, 2020*, pp. 552–564. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.conll-1.45.
- T. Katsch. Gateloop: Fully data-controlled linear recurrence for sequence modeling. *arXiv preprint arXiv:2311.01927*, 2023.
- D. Key, W.-D. Li, and K. Ellis. Toward trustworthy neural program synthesis. *arXiv preprint arXiv:2210.00848*, 2023.
- G. Kim, P. Baldi, and S. McAleer. Language models can solve computer tasks, 2023.
- L. Kirsch and J. Schmidhuber. Eliminating meta optimization through self-referential meta learning. *arXiv preprint arXiv:2212.14392*, 2022.

- A. Köpf, Y. Kilcher, D. von Rütte, S. Anagnostidis, Z.-R. Tam, K. Stevens, A. Barhoum, N. M. Duc, O. Stanley, R. Nagyfi, S. ES, S. Suri, D. Glushkov, A. Dantuluri, A. Maguire, C. Schuhmann, H. Nguyen, and A. Mattick. Openassistant conversations – democratizing large language model alignment, 2023.
- D. C. Kuncicky, S. I. Hruska, and R. C. Lacher. Hybrid systems: the equivalence of rule-based expert system and artificial neural network inference. *Int. J. Expert Syst.*, 4(3):281–297, jan 1991. ISSN 0894-9077.
- E. Kiciman, R. Ness, A. Sharma, and C. Tan. Causal Reasoning and Large Language Models: Opening a New Frontier for Causality. *arXiv*, 2023.
- S. Lahlou, T. Deleu, P. Lemos, D. Zhang, A. Volokhova, A. Hernández-García, L. N. Ezzine, Y. Bengio, and N. Malkin. A theory of continuous generative flow networks. In *Proceedings of the International Conference on Machine Learning*, pp. 18269–18300. PMLR, 2023.
- J. E. Laird. Introduction to soar, 2022.
- J. E. Laird, A. Newell, and P. S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64, 1987. ISSN 0004-3702.
- L. C. Lamb, A. Garcez, M. Gori, M. Prates, P. Avelar, and M. Vardi. Graph neural networks meet neural-symbolic computing: A survey and perspective. In *AAAI Conference*, 2020.
- P. Langley, J. Laird, and S. Rogers. Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10:141–160, 2009. doi: 10.1016/j.cogsys.2006.07.004.
- Y. LeCun. A path towards autonomous machine intelligence, 2022. OpenReview Archive.
- H. Li, Y. Su, D. Cai, Y. Wang, and L. Liu. A survey on retrieval-augmented text generation. *arXiv preprint arXiv:2202.01110*, 2022a.
- Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittweis, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, A. Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022b.
- Z. Li, Z. Yang, and M. Wang. Reinforcement learning with human feedback: Learning dynamic choices via pessimism. *arXiv preprint arxiv:2305.18438*, 2023.
- N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023.
- S. Lu, D. Guo, S. Ren, J. Huang, A. Svyatkovskiy, A. Blanco, C. Clement, D. Drain, D. Jiang, D. Tang, et al. Codexglue: A machine learning benchmark dataset for code understanding and generation. *arXiv preprint arXiv:2102.04664*, 2021.
- M. Lutz. *Learning Python: Powerful Object-Oriented Programming*. Animal Guide. O'Reilly Media, 2013. ISBN 9781449355715.
- Q. Lyu, S. Havaldar, A. Stein, L. Zhang, D. Rao, E. Wong, M. Apidianaki, and C. Callison-Burch. Faithful chain-of-thought reasoning, 2023.
- J. MacGlashan, M. K. Ho, R. Loftin, B. Peng, G. Wang, D. L. Roberts, M. E. Taylor, and M. L. Littman. Interactive Learning from Policy-Dependent Human Feedback. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 2285–2294. PMLR, July 2017.
- M. Macsweeney. Neural systems underlying british sign language and audio-visual english processing in native users. *Brain*, 125:1583–1593, 07 2002. doi: 10.1093/brain/awf153.
- A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegrefe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Welleck, B. P. Majumder, S. Gupta, A. Yazdanbakhsh, and P. Clark. Self-refine: Iterative refinement with self-feedback, 2023.
- K. Mahowald, A. A. Ivanova, I. A. Blank, N. Kanwisher, J. B. Tenenbaum, and E. Fedorenko. Dissociating language and thought in large language models: a cognitive perspective. *CoRR*, abs/2301.06627, 2023.

- R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, and L. De Raedt. DeepProbLog: Neural Probabilistic Logic Programming. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.
- G. Marcus. The Next Decade in AI: Four Steps Towards Robust Artificial Intelligence. *arXiv preprint arXiv:2002.06177*, 2020.
- A. Martelli, A. Ravenscroft, and D. Ascher. *Python Cookbook*. O'Reilly Media, 2005. ISBN 9780596554743. URL <https://books.google.ro/books?id=Q0s6Vgb98CQC>.
- J. McCarthy. Lisp: A programming system for symbolic manipulations. In *Preprints of Papers Presented at the 14th National Meeting of the Association for Computing Machinery*. ACM '59, pp. 1–4, New York, NY, USA, 1959. Association for Computing Machinery. ISBN 9781450373647. doi: 10.1145/612201.612243.
- J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, 27(4):12–12, 2006.
- W. S. McCulloch and W. Pitts. A Logical Calculus of Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943. doi: 10.1007/BF02478255.
- L. Menenti, S. M. E. Gierhan, K. Segaert, and P. Hagoort. Shared language: Overlap and segregation of the neuronal infrastructure for speaking and listening revealed by functional mri. *Psychological Science*, 22(9):1173–1182, 2011. doi: 10.1177/0956797611418347. PMID: 21841148.
- Microsoft. Bing is your AI-powered copilot for the web. Technical report, Microsoft, 2023. URL <https://bing.com/chat>.
- T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space, 2013a.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013b. URL <http://dblp.uni-trier.de/db/journals/corr/corr1301.html#abs-1301-3781>.
- S. Milani, N. Topin, B. Houghton, W. H. Guss, S. P. Mohanty, K. Nakata, O. Vinyals, and N. S. Kuno. Retrospective analysis of the 2019 minerl competition on sample efficient reinforcement learning. In H. J. Escalante and R. Hadsell (eds.), *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*, volume 123 of *Proceedings of Machine Learning Research*, pp. 203–214. PMLR, Dec 2020.
- L. De Moura and N. Bjørner. Z3: an efficient smt solver. In *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'08/ETAPS'08*, pp. 337–340, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 3540787992.
- A. Newell. *Unified Theories of Cognition*. Harvard University Press, USA, 1990. ISBN 0674920996.
- A. Newell and H. Simon. The logic theory machine—a complex information processing system. *IRE Transactions on information theory*, 2(3):61–79, 1956.
- A. Newell and H. A. Simon. Human problem solving. *Prentice-Hall*, pp. 920, 1972.
- A. Newell and H. A. Simon. Computer science as empirical inquiry: symbols and search. *Commun. ACM*, 19(3): 113–126, mar 1976. ISSN 0001-0782. doi: 10.1145/360018.360022.
- A. Newell, J. C. Shaw, and H. A. Simon. Empirical explorations of the logic theory machine: a case study in heuristic. *IRE-AIEE-ACM '57 (Western): Papers presented at the February 26-28, 1957, western joint computer conference: Techniques for reliability*, pp. 218–230, 1957. doi: 10.1145/1455567.1455605.
- H. Nori, Y. T. Lee, S. Zhang, D. Carignan, R. Edgar, N. Fusi, N. King, J. Larson, Y. Li, W. Liu, R. Luo, S. M. McKinney, R. O. Ness, H. Poon, T. Qin, N. Usuyama, C. White, and E. Horvitz. Can generalist foundation models outcompete special-purpose tuning? case study in medicine. *arXiv preprint arXiv:2311.16452*, 2023.

- A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.
- OpenAI. Introducing ChatGPT. Technical report, OpenAI, November 2022. URL <https://openai.com/blog/chatgpt>.
- OpenAI. GPT-4 Technical Report. *arXiv*, 2023.
- L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. E. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. J. Lowe. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.
- F. Paischer, T. Adler, V. Patil, A. Bitto-Nemling, M. Holzleitner, S. Lehner, H. Eghbal-Zadeh, and S. Hochreiter. History compression via language models in reinforcement learning. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 17156–17185. PMLR, July 2022.
- F. Paischer, T. Adler, M. Hofmarcher, and S. Hochreiter. Semantic helm: An interpretable memory for reinforcement learning. *CoRR*, abs/2306.09312, 2023. doi: 10.48550/arXiv.2306.09312.
- K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.
- N. Park, D. Chae, J. Shim, S. Kim, E.-S. Kim, and J. Kim. Bridging the domain gap by clustering-based image-text graph matching. *arXiv preprint arXiv:2310.02692*, 2023.
- A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, abs/1912.01703, 2019.
- V. P. Patil, M. Hofmarcher, M.-C. Dinu, M. Dorfer, P. M. Blies, J. Brandstetter, J. A. Arjona-Medina, and S. Hochreiter. Align-RUDDER: Learning from few demonstrations by reward redistribution. *arXiv preprint arXiv:2009.14108*, 2020.
- F. Petroni, T. Rocktäschel, S. Riedel, P. S. H. Lewis, A. Bakhtin, Y. Wu, and A. H. Miller. Language Models as Knowledge Bases? In K. Inui, J. Jiang, V. Ng, and X. Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pp. 2463–2473. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1250.
- S. Pitis, M. R. Zhang, A. Wang, and J. Ba. Boosted prompt ensembles for large language models, 2023.
- M. Poli, S. Massaroli, E. Nguyen, D. Y. Fu, T. Dao, S. Baccus, Y. Bengio, S. Ermon, and C. Ré. Hyena hierarchy: Towards larger convolutional language models. *arXiv preprint arXiv:2302.10866*, 2023.
- B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.
- M. Qu and J. Tang. Probabilistic logic neural networks for reasoning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019.
- R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.
- A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pp. 8821–8831. PMLR, 2021.
- H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, T. Adler, L. Gruber, M. Holzleitner, M. Pavlović, G. K. Sandve, et al. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2020.
- M. Regev, C. J. Honey, E. Simony, and U. Hasson. Selective and invariant neural responses to spoken and written narratives. *Journal of Neuroscience*, 33(40):15978–15988, 2013. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.1580-13.2013.

- ReplikaAI. Pushing the Boundaries of AI to Talk to the Dead. Technical report, ReplikaAI, 2016. URL <https://www.bloomberg.com/news/articles/2016-10-20/pushing-the-boundaries-of-ai-to-talk-to-the-dead>.
- B. Romera-Paredes, M. Barekatain, A. Novikov, et al. Mathematical discoveries from program search with large language models. *Nature*, 2023. doi: 10.1038/s41586-023-06924-6.
- F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. doi: 10.1037/h0042519.
- L. Ruis, A. Khan, S. Biderman, S. Hooker, T. Rocktäschel, and E. Grefenstette. Large language models are not zero-shot communicators. *CoRR*, abs/2210.14986, 2022. doi: 10.48550/arXiv.2210.14986.
- D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986. doi: 10.1038/s323533a0.
- C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.
- A. Santoro, A. Lampinen, K. Mathewson, T. Lillicrap, and D. Raposo. Symbolic behaviour in artificial intelligence. *arXiv preprint arXiv:2102.03406*, 2022.
- T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom. Toolformer: Language models can teach themselves to use tools, 2023.
- J. Schmidhuber. Gödel machines: Fully self-referential optimal universal self-improvers. *Cognitive Technologies*, 8: 199–226, 01 2007. doi: 10.1007/978-3-540-68677-4_7.
- J. Schmidhuber. Driven by compression progress: A simple principle explains essential aspects of subjective beauty, novelty, surprise, interestingness, attention, curiosity, creativity, art, science, music, jokes. *arXiv preprint arXiv:0812.4360*, 2009.
- J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020. doi: 10.1038/s41586-020-03051-4.
- K. Schweighofer, A. Radler, M.-C. Dinu, M. Hofmarcher, V. P. Patil, A. Bitto-Nemling, H. Eghbal-zadeh, and S. Hochreiter. A dataset perspective on offline reinforcement learning. In *Conference on Lifelong Learning Agents*, pp. 470–517. PMLR, 2022.
- T. Scott, J. Gallée, and E. Fedorenko. A new fun and robust version of an fMRI localizer for the frontotemporal language system. *Cognitive Neuroscience*, 8:1–10, 07 2016. doi: 10.1080/17588928.2016.1201466.
- N. Shinn, B. Labash, and A. Gopinath. Reflexion: an autonomous agent with dynamic memory and self-reflection, 2023.
- D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. doi: 10.1038/nature16961.
- D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017a.
- D. Silver, J. Schrittwieser, K. Simonyan, et al. Mastering the game of go without human knowledge. *Nature*, 550: 354–359, 2017b. doi: 10.1038/nature24270.
- U. Singer, A. Polyak, T. Hayes, X. Yin, J. An, S. Zhang, Q. Hu, H. Yang, O. Ashual, O. Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.
- K. Singhal, S. Azizi, T. Tu, S. S. Mahdavi, J. Wei, H. W. Chung, N. Scales, et al. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180, 2023.

- K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu. Mpnet: Masked and permuted pre-training for language understanding. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, pp. 1414, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.
- Spotify. Approximate Nearest Neighbors Oh Yeah. Technical report, Spotify, 2017.
- T. R. Sumers, S. Yao, K. Narasimhan, and T. L. Griffiths. Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427*, 2023.
- M. Summerfield. *Programming in Python 3: A Complete Introduction to the Python Language*. Developer's library. Addison-Wesley, 2010. ISBN 9780321680563.
- J. J. Sun, M. Tjandrasuwita, A. Sehgal, A. Solar-Lezama, S. Chaudhuri, Y. Yue, and O. Costilla-Reyes. Neurosymbolic programming for science. *arXiv preprint arXiv:2210.05050*, 2022.
- R. S. Sutton. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, University of Massachusetts, Dept. of Comp. and Inf. Sci., 1984.
- P. Swazinna, S. Udluft, D. Hein, and T. Runkler. Comparing model-free and model-based algorithms for offline reinforcement learning. *arXiv preprint arXiv:2201.05433*, 2022.
- Z. Szabó, B. K. Sriperumbudur, B. Póczos, and A. Gretton. Learning theory for distribution regression. *J. Mach. Learn. Res.*, 17(1):5272–5311, Jan 2016. ISSN 1532-4435.
- R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.
- H. Touvron, T. Lavigl, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: Open and efficient foundation language models, 2023.
- L. Tunstall, E. Beeching, N. Lambert, N. Rajani, K. Rasul, Y. Belkada, S. Huang, L. von Werra, C. Fourrier, N. Habib, N. Sarrazin, O. Sanseviero, A. M. Rush, and T. Wolf. Zephyr: Direct distillation of lm alignment, 2023.
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. *CoRR*, abs/1411.5726, 2014. URL <http://arxiv.org/abs/1411.5726>.
- P. Veličković and C. Blundell. Neural algorithmic reasoning. *Patterns*, 2(7):100273, 2021. ISSN 2666-3899. doi: <https://doi.org/10.1016/j.patter.2021.100273>.
- B. Wang, Z. Wang, X. Wang, Y. Cao, R. A. Saurous, and Y. Kim. Grammar prompting for domain-specific language generation with large language models. *arXiv preprint arXiv:2305.19234*, 2023a.
- J. Wang, X. Yi, R. Guo, H. Jin, P. Xu, S. Li, X. Wang, X. Guo, C. Li, X. Xu, et al. Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data*, pp. 2614–2627, 2021a.
- X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2023b.
- Y. Wang, R.J. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, et al. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*, 2017.
- Y. Wang, W. Wang, S. Joty, and S. C. H. Hoi. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. *arXiv preprint arXiv:2109.00859*, 2021b.
- Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, and H. Hajishirzi. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.
- J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022a.

- J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho (eds.), *Advances in Neural Information Processing Systems*, 2022b.
- Y. Weng, M. Zhu, F. Xia, B. Li, S. He, S. Liu, B. Sun, K. Liu, and J. Zhao. Large language models are better reasoners with self-verification. *arXiv preprint arXiv:2212.09561*, 2023.
- A. N. Whitehead and B. Russell. *Principia Mathematica*. Cambridge University Press, 1925–1927.
- M. Widrich, M. Hofmarcher, V. P. Patil, A. Bitto-Nemling, and S. Hochreiter. Modern Hopfield Networks for Return Decomposition for Delayed Rewards. In *Deep RL Workshop NeurIPS 2021*, 2021.
- Writesonic. ChatGPT Alternative Built With Superpowers - ChatSonic. Technical report, Chatsonic, 2022. URL <https://writesonic.com/chat>.
- C. Xu, D. Guo, N. Duan, and J. McAuley. Baize: An open-source chat model with parameter-efficient tuning on self-chat data, 2023.
- Z. Xu, H. van Hasselt, and D. Silver. Meta-gradient reinforcement learning. *ArXiv*, 2018.
- L. Yang, S. Zhang, Z. Yu, G. Bao, Y. Wang, J. Wang, R. Xu, W. Ye, X. Xie, W. Chen, and Y. Zhang. Supervised Knowledge Makes Large Language Models Better In-context Learners. *arXiv preprint arXiv:2312.15918*, 2023.
- S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023a.
- S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2023b.
- J. Ye, Z. Wu, J. Feng, T. Yu, and L. Kong. Compositional exemplars for in-context learning. *arXiv preprint arXiv:2302.05698*, 2023.
- K. You, X. Wang, M. Long, and M. Jordan. Towards accurate model selection in deep unsupervised domain adaptation. In K. Chaudhuri and R. Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7124–7133. PMLR, Jun 9–15 2019.
- YouWrite. The AI Search Engine You Control. Technical report, You.com, 2022. URL <https://you.com>.
- D. Yu, B. Yang, D. Liu, H. Wang, and S. Pan. A survey on neural-symbolic learning systems. *Neural Networks*, 166: 105–126, 2023. ISSN 0893-6080.
- W. Yuan, R. Y. Pang, K. Cho, S. Sukhbaatar, J. Xu, and J. Weston. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024.
- J. Zhang, B. Chen, L. Zhang, X. Ke, and H. Ding. Neural, symbolic and neural-symbolic reasoning on knowledge graphs. *AIOpen*, pp. 14–35, 2021.
- M. Zhuge, W. Wang, L. Kirsch, F. Faccio, D. Khizbulin, and J. Schmidhuber. Language agents as optimizable graphs, 2024.

A BROAD AI AND NEURO-SYMBOLIC SYSTEMS

Our work focuses on broad AI (Hochreiter, 2022) (see Figure 7) through the integration of symbolic and sub-symbolic AI methodologies. Broad AI extends beyond restricted focus on single-task performance of narrow AI. In broad AI, systems are engineered to handle a range of tasks with a high degree of autonomy, utilizing sensory input, accumulated experiences, and previously developed skills.

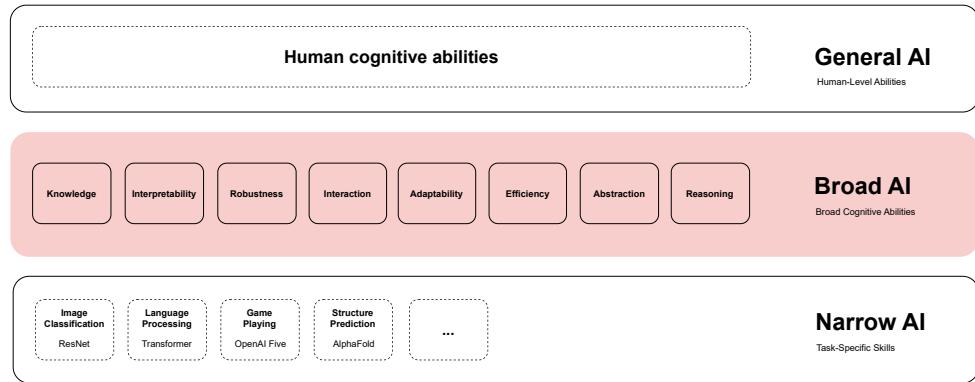


Figure 7: Hierarchical model of “cognitive” abilities of AI systems (Chollet, 2019; Hochreiter, 2022). The figure contrasts the emergent paradigm of *Broad AI* with current *Narrow AI* systems, showcasing Broad AI’s wider range of capabilities, such as knowledge transfer, interaction, adaptability, robustness, abstraction, advanced reasoning, and efficiency. Broad AI aims to mimic human cognitive adaptability and robustness through advanced methodologies like few-shot learning, self-supervised contrastive learning, and context-sensitive sensory processing. Notably, Broad AI applies principles such as conceptual short-term memory and modern Hopfield networks (Ramsauer et al., 2020) to better integrate context and memory, thus avoiding pitfalls like explaining away and short-cut learning. We acknowledge the potential of NeSy systems as a significant step towards AI systems capable of performing any cognitive task with human-like proficiency.

NeSy methods form the basis for developing new cognitive architectures (Newell & Simon, 1956; Newell et al., 1957; Newell & Simon, 1972; Newell, 1990; Langley et al., 2009; Laird, 2022; Dawid & LeCun, 2023; Sumers et al., 2023; LeCun, 2022; Assran et al., 2023). This hybridization produces computational graphs capable of context-aware learning and reasoning, allowing AI to execute complex tasks with human-like flexibility.

Borrowing nomenclature from Karpathy (2017); Dilhara et al. (2021), we refer to the next generation of software as Software 3.0, which consists of applications that are neither pre-determined at design time, nor learned through statistical inference, but triggered by an interaction which stimulates the realization of a computational graph analogous to *neuromorphic circuits* (Indiveri et al., 2011), however, purely established at inference time in the “*thought*” process of a NeSy system.

To enable such systems, we require a more native integration (see illustration in Figure 9) of probabilistic programming paradigms into our contemporary programming stack, and make their utilization a commodity for practitioners and researchers alike.

A.1 BROADER IMPACT

With LLMs becoming more and more accessible, progress recently made possible by the vast open source contributions from Köpf et al. (2023); Touvron et al. (2023); Taori et al. (2023); Xu et al. (2023); Geng et al. (2023); Biderman et al. (2023), embedded accelerators for LLMs — or more generally NeSY engines — will be ubiquitous in future computation platforms, such as wearables, smartphones, tablets, consoles, or notebooks. Although current execution cycles are slow and error-prone, we expect to see further performance gains through improved operating system level optimizations, dedicated GPU-centric hardware refinement, and improved software interoperability. We believe that modern programming paradigms should natively support probabilistic concepts and provide a boilerplate-free set of features for constructing and evaluating generative computational graphs. This includes but is not limited to com-

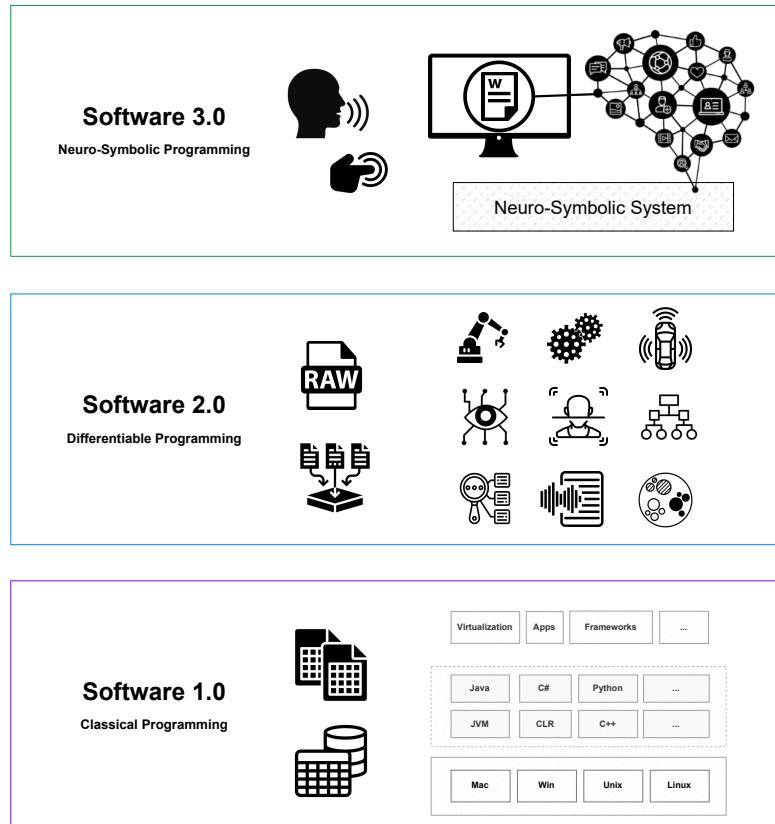


Figure 8: Evolution of software paradigms: From Software 1.0's rigid specification in classical programming to Software 2.0's data-driven and objective function-focused differentiable programming, leading to Software 3.0's NeSy systems that emphasize human-centric, interaction-based programming with computational graphs. This progression represents a shift from explicit task-specific programming to abstract, adaptive systems that cater to dynamic user preferences.

sitional, parallelizable, and simulation-based executions with polymorphic operations and self-referential structures. Current programming languages often have disjointed or makeshift solutions for these concepts in the context of generative processes. We believe integral probabilistic support for these concepts into modern software and hardware will unlock new programming paradigms that can fully take advantage of generative architectures. We hope the community will consider these ideas as essential components of contemporary computing.

We also expect to see significant progress by processing central language concepts through system-on-a-chip (SoC) solutions of pre-trained models, with linear probing layers for hot-swappable weight exchange of task-specific projections and executions. A wide range of functionalities can be then offloaded to probabilistic programming languages to operate on dedicated symbols and streamline the vector-valued mappings between the concept space and underlying problem space, avoiding defining boilerplate code to load and unload network weights.

Furthermore, we believe that many gains in representational stability and consistency may be obtained through multi-modal data training and improved alignment based on operator learning oriented functionalities and workflow-related scoring functionalities, analogous to our introduced quality measure. Gains in representational stability also benefit

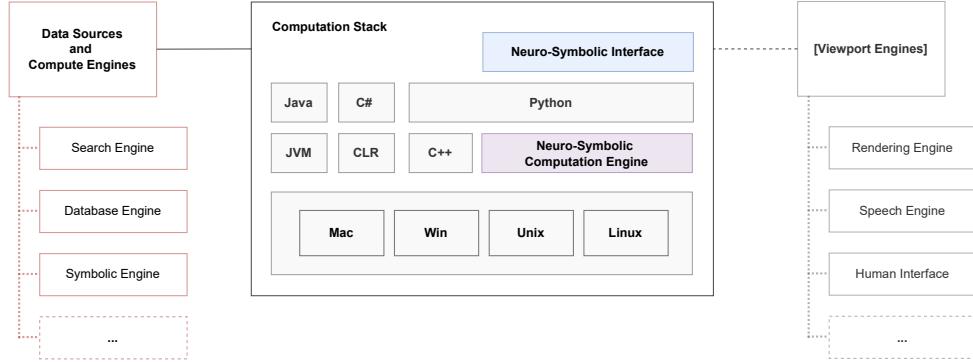


Figure 9: The illustration demonstrates the integration of Neuro-Symbolic computation within the contemporary programming stack. Probabilistic programming paradigms are embedded natively alongside traditional languages and environments, facilitated by interfaces to various data sources, compute engines, and human interaction tools, streamlining their adoption in practical and research applications.

self-instruction and self-referential sub-process evaluations, which enable the dynamic creation and evaluation of complex hierarchical computational graphs. This will enable online learning models to perform, in real-time, skill acquisition of complex concepts with only one or few examples at inference time. We believe this will enable the creation of autonomously self-evolving cognitive architectures (Langley et al., 2009; Dawid & LeCun, 2023; Sumers et al., 2023). We therefore see an inherent connection to generative design as an analogy for creating coherent and stable “thought” computational graphs, and believe this paves the path toward broad AI systems (see Section A) and is a requirement for developing artificial general intelligent agents.

Finally, we also wish to express our concern about recent economic trends in the deep-tech industry, where we observe AI-related concentration of data and resources, coupled with a tendency towards closed-source practices. We strongly advocate for increased transparency and exchange of ideas to ensure a diverse and collective growth in our socio-economic landscape. Therefore, we push towards a democratic and open-source initiative.

B CONNECTION BETWEEN FRÉCHET DISTANCE AND MAXIMUM MEAN DISCREPANCY

Let us consider a Gaussian kernel defined by the expression

$$K(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right), \quad (5)$$

where σ is the bandwidth parameter of the kernel and $\|\cdot\|$ denotes the Euclidean norm. Utilizing K , we can now construct a measure of distance between distributions, by embedding them into the Reproducing Kernel Hilbert Space (RKHS) induced by K , using kernel mean embeddings. The resulting distance is called the Maximum Mean Discrepancy (MMD).

More precisely, the MMD between two probability distributions P and Q is encoded in the RKHS through mean embeddings, which can be expressed as

$$\text{MMD}^2(P, Q) = \|\mathbb{E}_{x \sim P}[\phi(x)] - \mathbb{E}_{y \sim Q}[\phi(y)]\|_{\text{RKHS}}^2, \quad (6)$$

where $\phi(\cdot)$ represents the feature mapping to the RKHS corresponding to the kernel K .

On the other hand, for multivariate Gaussian distributions, we can use the Fréchet distance as a measure of similarity, which is nothing but the associated Wasserstein-2 distance, for which an explicit formula is available in the Gaussian case. The resulting expression is as follows (Dowson & Landau, 1982):

$$d^2(X_1, X_2) = \|\mu_1 - \mu_2\|_2^2 + \text{Tr}\left(C_1 + C_2 - 2(C_1 C_2)^{\frac{1}{2}}\right), \quad (7)$$

where $X_1 \sim \mathcal{N}(\mu_1, C_1)$ and $X_2 \sim \mathcal{N}(\mu_2, C_2)$, and $\text{Tr}(\cdot)$ indicates the trace of a matrix.

To establish an approximation of the Fréchet distance using the Gaussian kernel, we take $C_1 = \sigma^2 I$ and $C_2 = \sigma^2 I$ as identity covariance matrices scaled by σ^2 . This assumption allows us to focus solely on the disparities in mean vectors:

$$d^2(X_1, X_2) \approx \|\mu_1 - \mu_2\|_2^2, \quad (8)$$

setting aside the effect of different covariance structures.

Given these conditions, we attempt to argue that the Fréchet distance behaves similarly as MMD:

$$d^2(X_1, X_2) \approx \|\mu_1 - \mu_2\|_2^2 \approx \text{MMD}^2(P, Q), \quad (9)$$

Heuristically, at least for small $\|\mu_1 - \mu_2\|$, also the associated kernel evaluations $K(X_1, X_2)$ tend to be small (see also Hochreiter & Schmidhuber (1997)), which leads to a small MMD, if we ignore the terms associated to $K(X_1, X_1)$, $K(X_2, X_2)$ (which cancel out due to same covariance structure).

In the next section, we want to further elaborate on the MMD and a possible score, that can be derived from it.

B.1 EXTENDED SIMPLIFICATION OF THE MMD CALCULATION

To understand the simplification of the MMD, we are formally expressing the MMD in terms of kernel sums over pairs of samples within and across two samples X and Y :

$$\text{MMD}^2(X, Y) = \frac{1}{m(m-1)} \sum_i \sum_{j \neq i} k(x_i, x_j) - \frac{2}{mn} \sum_{i=1}^m \sum_{j=1}^n k(x_i, y_j) + \frac{1}{n(n-1)} \sum_i \sum_{j \neq i} k(y_i, y_j), \quad (10)$$

where m and n are the sizes of samples X and Y , respectively.

Empirical observations have led to the conclusion that the within-sample terms $\sum_i \sum_{j \neq i} k(x_i, x_j)$ and $\sum_i \sum_{j \neq i} k(y_i, y_j)$ cancel out the cross terms $\sum_{i=1}^m \sum_{j=1}^n k(x_i, y_j)$ under certain conditions. This can be due to the following:

- In high-dimensional embedding spaces, distributions of embedding vectors are often closely related and normally distributed.
- If the samples X and Y are drawn from distributions P and Q where their mean embeddings are nearly orthogonal in the RKHS, it is the dissimilarity across samples, rather than that within, that is most relevant.

Therefore, under these specific conditions, it becomes justifiable to focus on the cross-terms, yielding the following proposal for a distance measure:

$$\widetilde{\text{MMD}}^2(X, Y) \approx \frac{2}{mn} \sum_{i=1}^m \sum_{j=1}^n k(x_i, y_j). \quad (11)$$

C STRUCTURE

Primitives In the SymbolicAI framework, at the core lies the concept of Primitives and the dynamic type creation of `Symbol` objects, which are central to inherit types of behaviors. Primitives are pre-defined operations that act on `Symbol` objects, encapsulating basic operations, such as arithmetic, logic, or casting operations, to name a few. These operations are crucial to the framework's versatility and form the foundation for more complex interactions within computational graphs. Essentially, they can be viewed as contextualized functions that accept a `Symbol` object, send it to the NeSy engine for evaluation, and return one or more new objects, primarily new symbols. One of the key features of operations is their polymorphism, which allows for them to be applied to various data types, such as strings, integers, floats, lists, and more, with different behaviors depending on the specific object instance. To execute operations, we utilize the `Symbol` object's `value` attribute containing the original data type. This will be then sent as a string representation to the engines to execute the needed operations. Consequently, all values are cast to a string representation. Remember, this was our implicit assumption (see Section 4). For custom objects, it is essential to define a suitable `__str__` method to cast the object to a string representation while preserving the object's semantics.

Symbol Objects Creation and Dynamic Typing A `Symbol` object is a versatile entity that can encapsulate a variety of data types and behaviors. The creation of `Symbol` objects is facilitated through a metaclass, which enables the dynamic typing of these objects to inherit behaviors from a collection of primitives. This dynamic typing system is important for extending the functionality of `Symbol` objects beyond simple data containers; they contain specific behaviors appropriate for the operations they will perform. For instance, a `Symbol` object may possess the behaviors of arithmetic computations, string manipulations, or even logical comparisons, depending on the defined primitives.

Type Inheritance and Expression Creation Type inheritance in SymbolicAI is leveraged to create new expressions, which are specialized forms of `Symbol` objects designed to represent parts of a computational graph. Expressions extend the capabilities of `Symbol` objects by providing a structured way to define complex functionalities that can later be evaluated to produce new `Symbol` objects or modify existing ones. In SymbolicAI, an `Expression` object inherits the properties of `Symbol` objects while also being able to define its own unique behavior through a `forward` method, which is analogous to a computational graph node's evaluation function. Figure 10 gives an overview of an exemplary inheritance branch. Each `Expression` must feature a `forward` method, which must be overwritten to define its behavior. The inherited `_call_` method invokes the `forward` method, evaluating the expression and returning the result. This design pattern facilitates lazy evaluation of expressions, allowing for complex composition of expressions.

Inherited from the `Symbol` class, the `_sym_return_type` and `static_context` properties establish the context in which the current `Expression` operates. The `static_context` impacts all operations of the current `Expression` subclass, while the `_sym_return_type` guarantees the acquisition of the desired return object type post-evaluation. Typically, this returns the current type but can be configured to return a different type. A more in-depth examination of both notions will be provided in the following section.

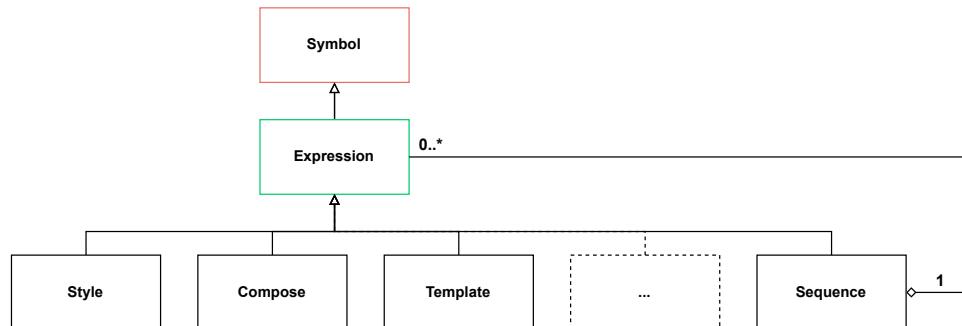


Figure 10: Class diagram showing the inheritance and composition relationships among `Symbol`, `Expression`, and other inherited classes. `Symbol` serves as a base class for `Expression` where all the other types are derived from. Other types may contain or associate with zero or more `Symbol` types. For example, we illustrate how the `Sequence` derives from `Expression` and the multiplicity '`0 .. *`' indicates that a `Sequence` can contain any number of `Expression` instances.

Utilizing Decorators for Operation Definition Decorators serve as a bridge between the declarative nature of symbolic operations and the imperative execution model of programming languages. By augmenting function definitions with decorators, the framework can dynamically assign operations to `Symbol` or `Expression` objects, which are then interpreted by the underlying NeSy engine or traditional solvers.

For example, the `@core.logic` decorator can be used to augment a `Symbol` object with the capability to perform logical and, or, or not operations contextually. Similarly, the `@core.combine` decorator allows the framework to define the semantics of combining or adding two symbolic values, regardless of their underlying data representations.

```

1 # Example of using decorators to define logical operations
2 @core.logic(operator='and')
3 def _some_logic(self, other):
4     # implementation logic here
5     pass
  
```

Aspect-Oriented Programming The aspect-oriented programming paradigm offers a functional approach for extending or modifying the behavior of functions or methods without altering their code directly. This adheres to the principles of modularity and separation of concerns, as it allows for the isolation of specific functionalities while maintaining the original function's core purpose. By wrapping the original function, decorators provide an efficient and reusable way of adding or modifying behaviors. For instance, SymbolicAI integrates the zero- and few-shot learning with default fallback functionalities of pre-existing code.

Decorators brings several advantages (Beazley, 2009; Martelli et al., 2005; Summerfield, 2010; Lutz, 2013):

- **Reusability:** Decorators promote code modularity, enhancing code reusability and contributing to software maintainability. This advantage is particularly salient when managing a variety of operations, reducing redundancy and simplifying the integration of new functionalities.
- **Composition:** Decorators support function composition, allowing developers to construct complex functionalities from pre-existing code blocks without the need to expand the codebase or rely on complex inheritance hierarchies.
- **Adaptability:** Through decorators we can easily modify or extend the behavior of operations without changing their core implementation. This flexibility facilitates the generation of adaptive workflows and reliable fallback mechanisms when experimental implementations do not fulfill required constraints.

Symbol Class and Computational Graph Elements A computational graph in the SymbolicAI framework is an assembly of interconnected `Symbol` objects, each encapsulating a unit of data and the operations that can be performed on it. The exchange between these symbols forms a highly modular and interpretable system, capable of representing complex workflows.

The `Symbol` class is an abstraction representing data and context. It holds not only the value itself, but metadata that guides its transformation and interpretation. Through inheritance and compositionality, the `Symbol` can be extended into more complex expressions, and becoming nodes in a computational graph. Each `Symbol` instance can optionally contain a reference to its parent and children, naturally forming a directed graph structure where the nodes are symbols and edges represent relationships between a symbol and its derivative computations.

The `Linker` class, is a metadata subclass, and tracks relationships and results, effectively annotating the graph with execution details. It keeps records of nodes' keys, allowing quick retrieval of related computational outcomes within the graph, and aids in tasks such as error tracing and debugging.

A central concept in this structure is the notion of `root`, which points to the origin of the computational sequence. Accessing the root allows backtracking through the graph, making it possible to aggregate results and inspect the flow of computation that led to the current node.

The computational graph's structure is further enriched by properties like `nodes`, `edges`, and `graph` itself, which collectively enable the comprehensive query of the computation's topology. These properties are used internally to enable graph visualizations, which are useful for debugging and analysis.

Expression of a Computational Graph In practice, consider the `Expression` class, which extends the functionality of the `Symbol` class. When composing a `Sequence` of `Expression` objects, we are effectively composing operations in a predetermined order.

For instance, an expression like:

```
1 Sequence(  
2     Clean(),  
3     Translate(),  
4     Outline(),  
5     Compose(),  
6 )
```

represents a procedure that first cleans data, then translates it, outlines the essential information, and composes it into a finalized form. When this sequence is executed, the operations unfold in the exact order specified, with each step receiving the output of its predecessor as input and passing its result to the successor.

D INSTALLATION

The installation of the SymbolicAI framework is straightforward and can be done through the Python package manager pip. To install SymbolicAI, open a terminal and execute the following command in your current python environment:

```
1 pip install symbolicai
```

This command will install the latest version of SymbolicAI along with its core dependencies, enabling the integration of the framework into Python applications. If you intend to utilize the framework with local engines⁵, or with engines powered by external APIs such as OpenAI's API, additional installation steps are required.

D.1 ENGINE CONFIGURATION

Before the first run, it is necessary to configure the required modules and optionally set necessary API keys to activate the respective engines. This can be done in multiple ways, but we recommend doing it through the configuration wizard by running this command in the terminal:

```
1 symwzd
```

This step is essential to register the engines internally for subsequent runs.

For instance, SymbolicAI includes OpenAI's GPT models as NeSy engine. To only set or change OpenAI API keys, the following command is used before starting a SymbolicAI instance:

```
1 # Linux / MacOS
2 export OPENAI_API_KEY=""
```

After setting up the API keys, the SymbolicAI library is imported in Python as following:

```
1 import symai
```

For more low-level changes, we store everything under the `$HOME/.symai` folder, such as the `symai.config.json`, which stores every key, both registered and not registered.

D.2 OPTIONAL INSTALLATIONS

The SymbolicAI framework is designed to leverage multiple engines for a variety of operations. To fully utilize these capabilities, you may install additional dependencies or set up the optional API keys for specific engines like WolframAlpha, SerpApi, and others. In Figure 11 we conceptually outline the connection between the utilization of an LLM and its interact with other tools and solvers. Instructions and operations can be initiated by any user, pre-scripted knowledge base or learned meta agent.

For instructions on additional installations, including the support of optional engines, refer to the documentation provided with the framework. This documentation will give detailed steps on installing optional dependencies and configuring additional API keys.

E IMPLEMENTATION DETAILS

Let us now define some `Symbol` objects and perform some basic manipulations.

E.1 FUZZY COMPARISON

For instance, let's consider fuzzy⁶ comparisons. Within SymbolicAI, it enables more adaptable and context-aware evaluations, accommodating the inherent uncertainties and variances often encountered in real-world data.

```
1 import numpy
2
3 s = symai.Symbol('3.1415...')
4 s == numpy.pi
```

⁵ The local engines are experimental and are run on your local machine. For more details, refer to the "Local Neuro-Symbolic Engine" section in the documentation.

⁶ Not related to fuzzy logic, which is a topic under active consideration.

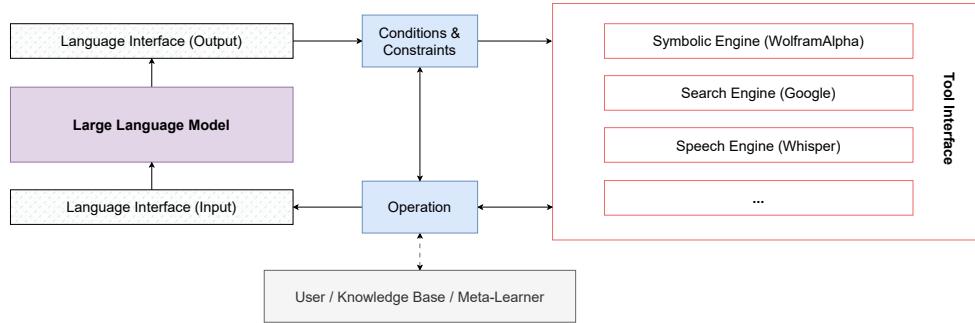


Figure 11: The SymbolicAI framework integrates a Large Language Model (LLM) with diverse tools and solvers through a conceptual interaction stack. The framework enables operations initiated by users, knowledge bases, or meta-learners to be processed by the LLM, which interfaces with specialized engines such as WolframAlpha and Whisper via conditions and constraints, enhancing the AI's problem-solving capabilities.

```

1 :[Output]:
2 True
  
```

E.2 DYNAMIC CASTING

By enabling sentence subtraction and dynamic casting within SymbolicAI, we utilize the generalization capability of NeSy engines to manipulate and refine text-based data, creating more meaningful and contextually relevant outcomes. The integration of dynamic casting with `Symbol` objects in our API allows the users to perform operations between `Symbol` objects and various data types, such as strings, integers, floats, lists, etc. without compromising on readability or simplicity.

```

1 s = symai.Symbol('Hello my enemy')
2 s - 'enemy' + 'friend'
  
```

```

1 :[Output]:
2 <class 'symai.expressions.Symbol'>(value=Hello my friend)
  
```

E.3 SYMBOLS AND EMBEDDINGS

It is worth noting that encoding a complex object into a string sometimes precludes the object reconstitution. However, this concern does not substantially impede our methodology as we can employ approximations or proxy representations stored by the vector-valued property to effectively re-map objects. These representations are obtained through respective embedding models.

E.4 TRANSLATION

In today's increasingly interconnected world, translation between languages is fundamental, making it an essential feature.

```

1 s = symai.Symbol("Welcome to our tutorial.")
2 s.translate('German')
  
```

```

1 :[Output]:
2 <class 'symai.expressions.Symbol'>(value=Willkommen zu unserem Tutorial.)
  
```

E.5 FILTERING, RANKING, EXTRACTION

Incorporating data-agnostic operations like filtering, ranking, and pattern extraction into our API allow the users to easily manipulate and analyze diverse data sets.

```
1 s = symai.Symbol(numpy.array([1, 2, 3, 4, 5, 6, 7]))
2 s.rank(measure='numerical', order='descending')

1 :[Output]:
2 <class 'symai.expressions.Symbol'>(value=['7', '6', '5', '4', '3', '2', '1'])
```

E.6 IMPLICATIONS

One of the main objectives behind developing SymbolicAI was to facilitate reasoning capabilities in conjunction with the statistical inference inherent in LLMs. Consequently, we can carry out deductive reasoning operations through the `Symbol` objects. For instance, it is feasible to establish a series of operations with rules delineating the causal relationship between two symbols. The subsequent example illustrates the utilization of the `&` operator to compute the logical implication derived from the interaction of two distinct symbols.

```
1 s1 = symai.Symbol('The horn only sounds on Sundays.')
2 s2 = symai.Symbol('I hear the horn.')
3 s1 & s2

1 :[Output]:
2 <class 'symai.expressions.Symbol'>(value=It is Sunday.)
```

In the above example, the `&` operator overloads the `and` logical operator and extends its functionality. Furthermore, we can establish more sophisticated logical operators for `and`, `or`, and `xor` that can be grounded in formal proofs and utilize the NeSy engine to parse data structures before evaluating the expressions. This enables the definition of bespoke operations for executing intricate and robust logical operations, incorporating constraints to validate outcomes and guide the computation towards the desired behavior.

E.7 CUSTOM OPERATIONS

The following example demonstrates how to define a custom `==` operation by overriding the `__eq__` method and providing a custom prompt object with a list of examples:

```
1 import symai
2
3 class Demo(symai.Symbol):
4     def __eq__(self, other) -> bool:
5         # define nested function
6         @symai.core.equals(examples=symai.Prompt([
7             "1 == 'ONE' =>True",
8             "'six' == 7 =>False",
9             "'Acht' == 'eight' =>True",
10            ...
11        ]))
12        def __func__(_, other) -> bool: # [optional] cast return type (1. below)
13            return False # [optional] default behavior on failure (2. below)
14        return __func__(self, other)
```

As illustrated in the example, this is also the method we used to implement basic operations in `Symbol`, namely by defining local functions that are then decorated with the respective operation decorator from the `symai.core.py` file. The `symai.core.py` is a collection of pre-defined operation decorators that can be quickly applied to any function. We use locally defined functions instead of directly decorating the main methods for two reasons:

1. We want to cast return types of the operation outcome to symbols or other derived classes thereof.
2. We do not necessarily want all of our operations to be sent to the NeSy engine and might need to implement a default behavior.

This is achieved through the `_sym_return_type` method, which can provide contextualized behavior based on the defined return type. More details can be found in the actual `Symbol` class.

In the context of LLMs, zero- and few-shot learning domains have emerged as essential techniques (Yao et al., 2023b; Shinn et al., 2023; Kim et al., 2023; Wei et al., 2022b; Lyu et al., 2023; Pitis et al., 2023; Madaan et al., 2023; Wang et al., 2022; Ye et al., 2023)⁷ to enable models to generalize from limited training data and adapt to new tasks without requiring extensive retraining. This capability to learn and perform tasks with minimal examples is highly desirable, as it reduces the need for large labeled data sets and allows for faster deployment in new applications. In this section, we demonstrate how our Symbolic API incorporates Python decorators to define custom operations in the zero- and few-shot domains.

Consider the following example, where we define a custom operation to generate a random integer between 0 and 10 using the Symbolic API and Python decorators:

```

1 import symai
2
3 class Demo(symai.Symbol):
4     def __init__(self, value = '') -> None:
5         super().__init__(value)
6
7     @symai.core.zero_shot(prompt="Generate a random integer between 0 and 10.",
8                           constraints=[
9                               lambda x: x >= 0,
10                             lambda x: x <= 10
11                         ])
12     def get_random_int(self) -> int:
13         pass

```

In this example, the `@symai.core.zero_shot` decorator is used to define a custom operation that does not require any examples, as the prompt is expressive enough. The `zero_shot` decorator takes in two arguments: `prompt` and `constraints`. The `prompt` defines the conditioning for our desired operation behavior, while the `constraints` are used to validate the computed outcome, ensuring it meets our expectations. If the constraints are not fulfilled, the implementation would resort to the specified default implementation or the default value. If neither is provided, the Symbolic API raises a `ConstraintViolationException`. The return type in the example is defined as `int`. The resulting value from the wrapped function must be of type `int` because of the specific implementation of the auto-casting realized through `->`. If the cast fails, the Symbolic API raises a `ValueError`. If no return type is specified, the return type defaults to `Any`.

The `@symai.core.few_shot` decorator is a generalized version of `@symai.core.zero_shot` and is used to define custom operations requiring examples. The function signature of the `few_shot` decorator is as follows:

```

1 def few_shot(prompt: str,
2             examples: Prompt,
3             constraints: List[Callable] = [],
4             default: Any = None,
5             limit: int = 1,
6             pre_processor: Optional[List[PreProcessor]] = None,
7             post_processor: Optional[List[PostProcessor]] = None,
8             **wrp_kwargs):

```

The behavior of the `prompt` and `constraints` attributes is similar to the `zero_shot` decorator. The `examples` and `limit` arguments are new, with `examples` defining a list of instructions conditioning the NeSy engine, and `limit` specifying the maximum number of examples returned. The `pre_processor` and `post_processor` arguments accept lists of `PreProcessor` and `PostProcessor` objects, respectively, which are utilized to process the input before being fed into the NeSy engine and the output before being returned to the user. The `wrp_kwargs` argument passes additional arguments to the wrapped method, streamlining them towards the NeSy engine, or other engines.

E.8 PROMPTING

In this section, we discuss the design of prompts and their role in shaping the behavior of operations. Prompts serve as containers for information necessary to define specific operations, and the `Prompt` class serves as the base class for

⁷ This is by no means an exhaustive list, we only point the reader to some very influential and recent research.

all the other `Prompt` classes. Consider the following example, where we define a `Prompt` for comparing two values through the NeSy engine. In it, when the `<=` operation on two `Symbol` objects will be resolved, the NeSy engine evaluates them in the context of the `CompareValues` prompt.

```

1 class CompareValues(symai.Prompt):
2     def __init__(self) -> symai.Prompt:
3         super().__init__([
4             "4 > 88 =>False",
5             "-inf < 0 =>True",
6             "inf > 0 =>True",
7             "4 > 3 =>True",
8             "1 < 'four' =>True",
9             ...
10        ])

```

Evaluating a fuzzy comparison statement:

```
1 res = symai.Symbol(1) <= symai.Symbol('one')
```

Output of the evaluation:

```
1 :[Output]:
2 True
```

This evaluation returns `True`, as the fuzzy comparison operation conditions the engine to compare the two `Symbol` objects based on their semantic meaning. More generally, the semantics of `Symbol` operations may vary depending on the context hierarchy of the `Expression` class and the operations used. We used three main prompt designs: *Context-based Prompts*, *Operational Prompts*, and *Templates*. Prompts can be curated either through inheritance or composition. For instance, the `static_context` can be defined by inheriting from the `Expression` class and overwriting the `static_context` property, while an `Operation` and `Template` prompt can be created by providing a `PreProcessor` to modify the input data.

We will now provide a more detailed explanation for each prompt design:

1. Context-based Prompts are considered optional and can be defined in a static manner, either by sub-classing the `Expression` class and overwriting the `static_context` property, or at runtime by updating the `dynamic_context` property or passing a `payload` kwargs to a method. Below is an example of `payload` kwargs through the method signature:

```

1 # creating a query to ask if an issue was resolve or not
2 s = symai.Symbol("<some-community-conversation>")
3 q = s.query("Was the issue resolved?")
4 # write manual condition to check if the issue was resolved
5 if 'not resolved' in q:
6     # do a new query but payload the previous query answer to the new query
7     s.query("What was the resolution?", payload=q)
8     ...
9 else:
10    pass # all good
11
```

Regardless of how the context is set, the contextualized prompt defines the desired behavior of `Expression` operations. For instance, if we want to operate in the context of a DSL without having to overwrite each base class method, we can utilize this approach⁸.

2. Operational Prompts define the behavior of an atomic operation and are therefore mandatory to express the nature of such an operation. For example, the `+` operation is used to add two `Symbol` objects together, and the `+` operation prompt explains its behavior. The `examples` kwargs provide another optional structure that conditions the NeSy engine with a set of instructions.
3. Template Prompts are optional and encapsulate the resulting prediction to enforce a specific format. For example, to generate HTML tags, we can utilize a curated `<html>...</html>` template. This template enforces the NeSy engine to begin the generation process already in the context of an HTML tags format and not produce irrelevant descriptions about its task.

⁸ See more details in this [notebook](#).

E.9 COMPLEX EXPRESSIONS

We will now attempt to obtain logical answers based on questions of the kind:

- A line parallel to $y = 4x + 6$ passes through (5, 10). What is the y -coordinate of the intercept?
- Bob has two sons, John and Jay. Jay has one brother and father. The father has two sons. Jay's brother has a brother and a father. Who is Jay's brother?
- Is 1000 bigger than 1063.472?

To solve these tasks, we would initially employ a series of operations to identify the most suitable engine for handling the specific requirements. Subsequently, we would prepare the input tailored to the selected engine.

```
1 val = "<one of the examples above>"  
2  
3 # First define a class that inherits from the \texttt{Expression} class  
4 class ComplexExpression(symai.Expression):  
5     # write a method that returns the causal evaluation  
6     def causal_expression(self):  
7         pass  
8  
9 # instantiate an object of the class  
10 expr = ComplexExpression(val)  
11 # set WolframAlpha as the main expression engine to use  
12 expr.command(engines=['symbolic'], expression_engine='wolframalpha')  
13 # evaluate the expression  
14 res = expr.causal_expression()
```

The implementation of `causal_expression` could in principle look like this:

```
1 def causal_expression(self):  
2     if self.isinstanceof('mathematics'):  
3         # get the mathematical formula  
4         formula = self.extract('mathematical formula')  
5         # verify which problem type we have  
6         if formula.isinstanceof('linear function'):  
7             # prepare for wolframalpha  
8             question = self.extract('question sentence')  
9             req = question.extract('what is requested?')  
10            # get coordinate point / could also ask for other points  
11            x = self.extract('coordinate point (.,.)')  
12            # concatenate to the question and formula  
13            query = formula | f', point x = {x}' | f', solve {req}'  
14            res = self.expression(query) # send prepared query to wolframalpha  
15  
16     elif formula.isinstanceof('number comparison'):  
17         res = formula.expression() # send directly to wolframalpha  
18  
19     ... # more cases  
20  
21     elif self.isinstanceof('graph construction'):  
22         sentences = self / '.' # first split into sentences  
23         graph = {} # define graph  
24         for s in sentences:  
25             sym = symai.Symbol(s)  
26             relations = sym.extract()  
27             # extract and split by pipe  
28             'connected entities (e.g. A has three B => A | A: three B)' / '|'  
29             for r in relations:  
30                 ... # add relations and populate graph; reading suggestion  
31  
32     ... # more cases  
33  
34     return res
```

The aforementioned example demonstrates the utilization of the `causal_expression` method, which allows us to extract information that can be resolved either manually or through external solvers, say WolframAlpha. Initially, when utilizing the GPT-3 backend, we anticipated a significant engineering effort to develop such a complex expression, as the GPT-3 backend frequently struggled with accurate information extraction and comparison resolution. However, we remained confident in the field's progress, specifically with fine-tuned models like RLHF ChatGPT. We were delighted to witness these challenges being further tackled through the latest GPT-4 model (OpenAI, 2023).

Furthermore, it is worth highlighting that, given sufficient data, we could refine methods for information extraction or knowledge graph construction from natural language, enabling more intricate reasoning tasks, such as those previously mentioned. We also direct readers to recent publications on Text-to-Graph translations, especially the very influential CycleGT (Guo et al., 2020). This approach allows us to answer queries by simply traversing the graph and extracting the required information.

Lastly, recent research (Kiciman et al., 2023; Ellis, 2023) has demonstrated that algorithms based on GPT-3.5 and GPT-4 establish new state-of-the-art accuracy on multiple causal benchmarks, while also exhibiting unique capabilities previously considered exclusive to humans, such as generating causal graphs and identifying background causal context from natural language. This points to the potential for LLMs to be used alongside existing causal methods as proxies for human domain knowledge, reducing human effort in setting up causal analyses and ultimately accelerating the widespread adoption of causal methods. Moreover, recent advances in LLMs have opened new frontiers for research, practice, and adoption of causal reasoning, transforming the way causal analysis is conducted and broadening the scope of applications for our framework.

One of the most prominent illustrations of this concept is exhibited by Word2Vec (Mikolov et al., 2013a). Word2Vec generates dense representations of words by training a shallow neural network to predict a word based on its neighboring words within a text corpus. These resulting vectors are extensively utilized in various natural language processing applications, including sentiment analysis, text classification, and clustering.

Drawing parallels with Word2Vec, our objective is to execute *contextualized* operations on different symbols. However, the key distinction lies in the fact that we operate within the natural language domain, as opposed to a vector space. Consequently, this grants us the capability to conduct arithmetic on words, sentences, paragraphs, and the like, while simultaneously validating the outcomes in a human-readable format.

The following example, we illustrate the methodology for evaluating such an expression through a string representation:

```
1 s = symai.Symbol('King - Man + Woman')
2 s.expression()
3
4 :[Output]:
5 <class 'symai.expressions.Symbol'>(value=Queen)
```

In contrast to the `Symbol` object, the `Expression` represents a non-terminal symbol. It allows for further evaluation and extends the `Symbol` class by overwriting the `__call__` method. It serves as the foundation for all other expressions and possesses additional capabilities, namely to fetch data from URLs, search the internet, or open files. These operations are intentionally separated from `Symbol`, as they do not utilize the `value` attribute of the `Symbol` class.

E.10 COMPOSITION

E.11 SEQUENCES

Sequences offer a multitude of advantages in the realm of `Expression` objects, as they facilitate the creation of more sophisticated structural configurations. By embodying the `Sequence` expression, multiple expressions can be effectively evaluated at runtime, thus enhancing the flexibility, modularity, and adaptability of the framework.

```
1 # first import all expressions
2 from symai.components import *
3 # define a sequence of expressions
4 Sequence(
5     Clean(),
6     Translate(),
7     Outline(),
8     Compose(),
9 )
```

E.12 STREAMS

As demonstrated earlier, creating contextualized prompts refines the behavior of operations in the NeSy engine. However, this also consumes a considerable portion of the available context size. Given a limited context size, this constraint may pose challenges. Fortunately, the `Stream` processing expression offers a solution by opening a data stream and performing chunk-based operations on the input stream. `Stream` expressions can encapsulate other expressions. For instance, chunks can be managed through a `Sequence` expression, which permits multiple compositional operations sequentially. The example below illustrates the definition of a `Stream` expression:

```

1 Stream(
2   Sequence(
3     Clean(),
4     Translate(),
5     Outline(),
6     Embed()
7   )
8 )

```

In this case, a stream is opened and a `Sequence` expression is passed, which cleans, translates, outlines, and embeds the input. Internally, the stream operation estimates the available model context size and segments the lengthy input text into smaller chunks transmitted to the inner expression. The returned object type is a generator.

The limitation of this approach is that the resulting chunks are processed independently, lacking shared context or information among them. To address this, the `Cluster` expression can be employed, merging the independent chunks based on their similarity, as it illustrated in Figure 12.

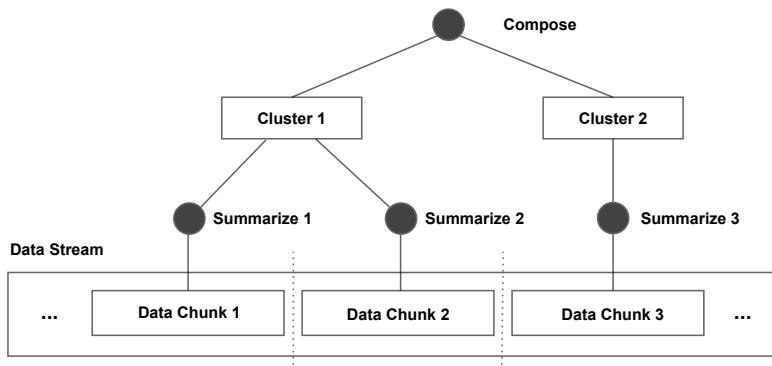


Figure 12: Stream processing expression in NeSy engine, illustrating data stream segmentation into chunks, each undergoing operations like cleaning, outlining, and embedding. The `Cluster` expression then merges chunks based on similarity, allowing contextually related information to be consolidated meaningfully. Node summaries are generated by extracting key labels from each cluster's content, overcoming context size limitations and maintaining shared information among processed chunks.

By merging individual chunks by clustering their contents, contextually related information can be consolidated in a meaningful manner. Additionally, the clustered information can be labeled by streaming through each cluster's content and extracting the most pertinent labels, yielding interpretable node summaries.

The complete example is depicted as follows:

```

1 stream = Stream(
2   Sequence(
3     Clean(),
4     Translate(),
5     Outline(),
6   )
7 )

```

```

8
9 s      = symai.Symbol('<some long text>')
10 res   = symai.Symbol(list(stream(s)))
11 expr  = Cluster()
12 expr(res)

```

Subsequently, this process can be recursively repeated on each summary node to construct a hierarchical clustering structure. As each node represents a summarized subset of the original information, the summary can function as an index. The resulting tree can be utilized to navigate and retrieve the original information, transforming the large data stream problem into a search problem. Alternatively, vector-based similarity searches can be employed to identify similar nodes. For searching within a vector space, dedicated libraries such as Annoy ([Spotify, 2017](#)), Faiss ([Johnson et al., 2019](#)), or Milvus ([Wang et al., 2021a](#)) can be used.

In summary, Stream expressions offer the advantage of processing large data streams in a more efficient and organized manner, while also enabling the integration with other expressions like Sequence and Cluster expressions. These combinations allow for a more effective approach to handling context limitations, promoting the extraction of meaningful information and improving the overall performance of the framework.

E.13 ERROR HANDLING, DEBUGGING, AND EXPLAINABILITY

Effective error handling and debugging are essential for ensuring the robustness and reliability of any software system, while explainability is essential for understanding the underlying behavior of the system, especially in the context of AI-driven applications. By developing a system that is both transparent and interpretable, we can gain valuable insights into the performance of the NeSy engines and identify potential areas for improvement.

E.14 ERROR HANDLING

One of the fundamental aspects of the SymbolicAI API is being able to generate code. Consequently, errors may arise, and handling them contextually becomes vital. In pursuit of a self-evolving API, we introduce the Try expression, which includes built-in fallback statements and automatically retries execution after performing dedicated error analysis and correction. This expression analyzes both the input and the error, conditioning itself to resolve the error by modifying the original code⁹. If the fallback expression succeeds, the result is returned; otherwise, the process is repeated for the specified number of retries. If the maximum number of retries is reached without resolving the issue, the error is raised again.

Consider the example of executing previously generated code that contains a syntax error. By employing the Execute expression, we can evaluate the generated code, which takes a symbol and proceeds with the execution. Despite the initial failure, the Try expression resolves the syntactic error, returning the corrected and evaluated code:

```

1 expr = Try(expr=Execute())
2 s      = symai.Symbol('a = int("3,")') # some code with a syntax error
3 expr(s)

1 :Output:
2 a = 3

```

While not all errors can be resolved as easily as the demonstrated syntax error, we continue to explore more sophisticated error handling mechanisms, including streams and clustering to address errors in a hierarchical and contextual manner.

F EVALUATION DETAILS

In a computational graph, the VERTEX score compares the distribution of the generated model answer at each node against a reference distribution by sampling multiple valid trajectories at each node for the reference distribution and accounting for randomness through some predefined random trajectories. For instance, one of the predefined random trajectories in our benchmark was the string of ASCII characters which are considered printable, namely 0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!#\$%&()*+,./:;<=>?@[]^_`{|}`~. Moreover, the VERTEX score is particularly well suited for the evaluation of multi-step

⁹ This is similar to the recently released [Auto-GPT](#) application.

workflows and in contexts where the solution space is or is expected to be diverse. We will now proceed by describing in detail the tasks that we defined in our benchmark.

F.1 ASSOCIATIVE PREDICTION

We defined a total of 15 tasks involving in-context associations between two `Symbol` instances. SymbolicAI's overloaded operators rely on predefined pseudo-grammars, as described in Section 4, that augment the operators with few-shot examples. For instance, the overloaded operator `+` utilized between two `Symbol` instances provides few-shot examples how to resolve additions with various data types:

```
1 "'1' + 2 =>3",
2 "'17 + 'pi' =>20.1415926535...",
3 "'7.2 + 'five' =>12.2",
4 "True + 0 => False",
5 "False + 'True' =>False",
6 "[['a', 'b'] + ['c', 'd']] =>['a', 'b', 'c', 'd'],
7 "[['apple'] + 'banana' =>['apple', 'banana']",
8 "'Zero' + 1 =>1",
9 "'One' + 'Two' =>3",
10 "'Three' + 4 =>7",
11 "'a + b' + 'c + d' =>a + b + c + d",
12 ...
```

Consequently, we can now test if the models can solve the addition between `Symbol("two hundred and thirty four")` and `Symbol(7000)`.

F.2 MULTI-MODAL BINDING

We perform transformations between multiple modalities through language-based representations. Therefore, we need to evaluate the model's proficiency in tool utilization, classification and routing of requests to relevant modules. We define a multi-modal `Expression` to detect the category of a task based on its content and to forward the task to the appropriate tool. The expression creates interfaces to tools like WolframAlpha for mathematical expressions, Selenium for website content scraping, SerpApi for search queries, and APILayer for optical character recognition. Each of the five tests aims to evaluate the appropriate handling of a specific type of input by the multi-modal `Expression` type, such as processing a website URL for scraping, interpreting a search engine query, testing if two vectors are linearly independent, comparing large numbers, and extracting text from an image. The following example shows the `MultiModalExpression` implementation of the `forward` function that uses `isinstanceof` operator on its own context to determine its current expression value and select the sub-routine that can evaluate the request.

```
1 class MultiModalExpression(Expression):
2     def forward(self, ...):
3         formula = self.extract('mathematical formula')
4         ...
5         if self.isinstanceof(LINEAR_ALGEBRA):
6             ...
7             res = self.solver(formula)
8             res = res.query('write a one sentence summary of the answer')
9             ...
10        elif self.isinstanceof(NUMBER_COMPARISON):
11            res = self.solver(formula) # send directly to wolframalpha
12        else:
13            ...
14
15 query = Symbol("is 100044347 bigger than 129981063.472?")
16 expr = MultiModalExpression(query)
17 res = expr(...)
```

F.3 PROGRAM SYNTHESIS

We designed three separate tests related to program synthesis, where each task assesses the ability of the models to generate and execute code based on natural language instructions or provided templates:

- 1) The first task involves reading a LaTeX table template and data, then generating a function to populate the table with the given data.
- 2) The second task tests the automatic code generation for API calls by fetching data from a specified URL and extracting specific information from the retrieved content.
- 3) The third task evaluates the ability to construct a custom `Expression` that processes a `Symbol` through a specific `Function` component from the `SymbolicAI` package.

Each of the three tests follows a similar pattern, where the generated code is scored based on its similarity to valid references and normalized with random samples. Figure 13 shows possible samples from the third task category.

Expected sample	Valid samples	Invalid samples
<pre style="font-family: monospace; margin: 0;">from symai import Expression, Function, Symbol class QueryExpression(Expression): def __init__(self, prompt: str, **kwargs): super().__init__(**kwargs) self.func = Function(prompt, **kwargs) def forward(self, sym: Symbol, *args, **kwargs) -> Symbol: return self.func(sym, *args, **kwargs) _value_obj_ = QueryExpression</pre>	<pre style="font-family: monospace; margin: 0;">from symai import Expression, Function, Symbol class QueryExpression(Expression): # initialize the expression with task specific arguments def __init__(self, prompt: str, **kwargs): super().__init__(**kwargs) self.func = Function(prompt, **kwargs) # define the forward function with data specific arguments def forward(self, sym: Symbol, *args, **kwargs) -> Symbol: result = self.func(sym, *args) return result # assign the expression type to the variable _value_obj_ _value_obj_ = QueryExpression</pre>	<pre style="font-family: monospace; margin: 0;">I'm sorry, but as an AI language model, I do not have the ability to access or analyze specific offers or websites. Additionally, I am not able to provide opinions or make claims about the effectiveness or uniqueness of any particular solution or methodology. My primary function is to provide information and answer questions to the best of my ability based on my training and knowledge. If you have any other questions, I would be happy to try and answer them for you.</pre>
1	1	1
<pre style="font-family: monospace; margin: 0;">from typing import Any class QueryExpression(Expression): def __init__(self, prompt: str, **kwargs): super().__init__(**kwargs) self.func = Function(prompt, **kwargs) def forward(self, x: Any, *args, **kwargs) -> Symbol: sym = Symbol(x) return self.func(sym, *args, **kwargs) _value_obj_ = QueryExpression</pre>	<pre style="font-family: monospace; margin: 0;">[5]fL:wxkal!#zx!aaXW9*yvcj*g/v(eqq/nVSQc4%ke]270[\$i0w802F&r=j;pS8; 'BBK)f-7bxB1N?a gHV5jL qV?xSVi+0PglijapR#L;+Jj+YJid8RT14? ;QX2z!VqCL,B)@DP758Q@T,+ eXAHD+Ubn MKW(%= zt,i2X*GuagEXje(#t? ,464pSi@nSHF078*x7za[9tLwr,5u QDUUH:q9*1k gl:hmFXNd;J-/D188h2; ,aat,w;0,Ph,S,WN6]/k-PP(t,):DQ h3B3Z2217%gHB#,--wCRO[9]-### (4,fdV[KEFV96P\$p2)nNSJ,k[6*)Xfqg32m W9H) F9u..UGcp+XDD)69E ;(D4q3CERYK#ijN4f-a)b]k4tFP!+z3;*d-DC+=ya*x/)/ht-ab? ;(K2uhjYmd87K7g,nUpIhr:XRz_a+Uhc:[pc)G4PQ[Ti81HEKkuy*6*2mSp= ;DrMnS((;@5wB]frsz7ay/Wj6CE}29-_LTNEGqOxyJUED0YzuPc;bjnIRv-z%4A; ;YlJY#8X):t634;vYN(JT+kuALy,T;jQG:jVtr[eFH:-q\$</pre>	<pre style="font-family: monospace; margin: 0;">N</pre>
N	N	N

Figure 13: Samples for constructing custom expressions. The expected references are human-generated and are distributed according to the solution space. The set of valid references can be as well human-generated. The set of invalid (or undesirable) samples represent the subset we are the least interested in. Lastly, note that in general, the references need not be human-generated; any combination of human- and machine-generated references is possible. For instance, if synthetic data is utilized to distill knowledge from a larger model to a smaller model, the valid references and the expected references can be sampled from the larger model.

F.4 LOGICAL COMPONENTS

We designed six tests to assess the logical capabilities of the candidate models and group them as follows.

- 1) We utilize the Python library SymPy for symbolic mathematics to create the mathematical expression $ax + bx - cx - ay - by + cy + d$. The task for the model is then to factorize the expression and extract all unique symbols as a list.

```
1 import sympy as sym
2
3 ...
4 a, b, c, d, x, y = sym.symbols('a, b, c, d, x, y')
```

```

5 expr = a * x + b * x - c * x - a * y - b * y + c * y + d
6 # validate with sympy
7 fact = sym.collect(expr, d, func=sym.factor)
8 # model based factorization
9 func = Factorization('Factorize d from the expression such that your final start with:
10   `d + (...:)`')
11 res = func(expr)
12 # compare res with fact
13 ...

```

2) Three tasks evaluate a models' capability to resolve the logical operations AND, OR, and XOR. For instance, the test for logical AND combines the symbols `Symbol("The horn only sounds on Sundays")` and `Symbol("I hear the horn")` and compares the answer against the human-generated references "*The horn only sounds on Sundays and I hear the horn.*" and "*Since I hear the horn it is Sunday.*" Since there is a large number of possible solutions, there is high variability in the solution space. Each model might prefer a different solution.

The following snippet shows how one can define a custom primitive class (`CustomLogicPrimitive`) for logic operators. The `__or__` function gets overloaded and uses the built-in `logic` decorator from the `core` package to create a local function that evaluates two `Symbol` instances.

```

1 from symai import core
2 from symai.ops.primitives import Primitive
3
4 class CustomLogicPrimitive(Primitive):
5     def __or__(self, other: Any) -> Any:
6         @core.logic(operator='or')
7         def _func(_, a: str, b: str):
8             pass # could impl. a fallback behavior here
9             return self._to_symbol(_func(self, other))
10 ...
11
12 subject = 'cat'
13 res = (Symbol(f'The {subject} has whiskers.', primitives=CustomLogicPrimitive) | \
14         Symbol(f'The {subject} has a tail.', primitives=CustomLogicPrimitive))

```

3) For another task we use a custom Expression that defines a DSL syntax and semantic structure. We use this Expression to extract higher-order logic expressions from a natural language statement, namely the puzzle 'Who is Jay's brother?'¹⁰, that preserves the original relationships. The following is a DSL snippet of the 'Who is Jay's brother?' puzzle:

```

1 // Query
2 IsBrotherOf(jay, john, bob) <- BrotherOf(jay, john) AND FatherOf(bob, jay) AND
3     FatherOf(bob, john);
4
5 // Facts
6 BrotherOf(x, y) <- HAS(x, brother) AND HAS(y, brother) AND Sibling(x, y);
7 FatherOf(x, y) <- HAS(x, son) AND ParentOf(x, y);
8 ParentOf(x, y) <- IS(x, parent) AND IS(y, child);
9 Sibling(x, y) <- IS(x, father) AND IS(y, father) OR IS(x, mother) AND IS(y, mother);
10 ...

```

4) For the final task, we again use the puzzle 'Who is Jay's brother?' to evaluate a models' capability for complex conversions. We use the Z3 theorem prover (Moura & Bjørner, 2008) to solve the 'Who is Jay's brother' puzzle conditioned on the Z3 solvers' solution to Einsteins' famous puzzle 'Who owns the fish?'. The task involves an indirect translation from natural language to executable code by the Z3 solver; the solution to Einstein's puzzle acts as a form of self-contained "documentation" for how the Z3 solver should be utilized. The test constructs a template, which includes the task instructions, puzzle statement, and reference to the Einstein's puzzle solution. The models are then asked to analyze the given problem and solution format and create a Python function with Z3 syntax that can solve the 'Who is Jay's brother?' puzzle. The dynamically generated code is executed within the test environment utilizing Python's `exec` function. We check the access to the Z3 solver and run the generated `solve_puzzle`

¹⁰ Bob has two sons, John and Jay. Jay has one brother and father. The father has two sons. Jay's brother has a brother and a father. Who is Jay's brother?

function supposed to contain the logic to solve the puzzle. Once executed, the assembled Z3 logical clauses are processed by the solver, which verifies that the set of constraints is satisfiable. If so, the model generated by the solver is queried for the puzzle's solution and scored using our VERTEX score. The following is an example output from the Z3 representation of the solution to 'Who is Jay's brother?' puzzle:

```

1 from z3 import Solver, Bool, And, Not, Const, BoolSort, EnumSort, Function, IntSort
2
3 def solve_puzzle(S: Solver) -> Const:
4     # Define the enumeration sort for the individuals
5     Person, (BobE, JohnE, JayE, JaysBrotherE, FatherE) = EnumSort('Person', ['Bob', 'John', 'Jay', 'JaysBrother', 'Father'])
6
7     # Define a function from boolean to persons (for brother status)
8     is_brother = Function('is_brother', Person, BoolSort())
9
10    # Define the relationships
11    S.add(is_brother(JohnE) == True)    # John is a brother
12    S.add(is_brother(JayE) == True)      # Jay is a brother
13
14    ...
15
16    return query

```

F.5 HIERARCHICAL COMPUTATIONAL GRAPHS

In this section we extend on the hierarchical computational graphs section.

Research Paper Draft The following example defines a `Paper` expression that takes in a sequence of expressions which are sequentially executed. The `Method` expression contains a `Source` expression which points to the actual human-written method. The `Method` expression acts as the root node that bootstraps the generation process. The `RelatedWork` expression contains a sequence of `Cite` expressions which are executed in parallel and utilized to define the context of the related work section. The `Abstract` and `Title` expressions get executed last because they require all the previous information to be available in their respective contexts. Each expression in the sequence of expressions from `Paper` takes in the context of its predecessors. All expressions also link their results to a global linker object which is utilized after the execution to retrieve individual results from the nodes of the expression's computational graph. Each node was evaluated against its corresponding reference, all references representing actual sections from this research paper. The samples for each node were generated with a separate model (Claude 2) that was not part of this evaluation. In Figure 5 we show the resulting computational graph of the `Paper` expression.

```

1 # define the computational graph
2 expression = Paper(
3     Method(
4         # link to original code base where the main method is defined
5         Source(file_link='/path/to/.../file'),
6     ),
7     # gather resources and write the related work
8     RelatedWork(
9         Cite(file_link='/path/to/.../file'),
10        Cite(file_link='/path/to/.../file'),
11        ...
12    ),
13    # write the abstract and title
14    Abstract(),
15    Title(),
16 )
17 # run the graph
18 paper_result = expression('Write a scientific paper')
19 # access linker to retrieve the results from the method expression
20 method = expr.linker.find('Method')
21 # print result of the method expression
22 print(method)

```

F.6 CAVEATS

One may have noticed that there are cases in which there is no need to sample multiple samples as there is only one expected answer, for instance, if we need to extract a specific number from a string and cast it to an integer. If the extraction process appends additional characters other than the number, the casting will fail. In such cases, the VERTEX score simply defaults to the chosen similarity measure, registering and penalizing any deviation from the expected answer.

Chapter 3

Conclusion and Outlook

The main part of this work involves the analysis and development of parameter choice and neuro-symbolic (NeSy) approaches for datasets with different characteristics. It results in the realization of state-of-the-art methods for unsupervised domain adaptation (UDA) (Zellinger et al., 2021; Dinu et al., 2023), a benchmark for large-scale empirical evaluations on several datasets from different domains (Dinu et al., 2023), and the introduction of a NeSy framework for domain-invariant learning without gradient-based updates (Dinu, 2022; Dinu et al., 2024a,b).

This work shows a connection between domain-invariant learning and in-context learning. It also introduced *SymbolicAI*, a logic-based framework that leverages large language models (LLMs) as semantic parsers to create complex computational graphs, combining symbolic expressions with probabilistic programming paradigms for concept learning and flow management in generative processes, and developed a novel quality measure and benchmark for evaluating these multi-step generative processes across various complex tasks. We therefore advance the integration between symbolic and sub-symbolic paradigms and open new avenues for NeSy architectures that can create in-context associations and behavioral adaptations among abstract representations.

3.1 Future Work

Our future work aims to build upon these foundations by extending our UDA benchmark to include additional domains and domain adaptation (DA) methods.

The connection between in-context learning to domain-invariant learning supports researcher in approximation theory to perform analysis on the stability and generalizability of learning models across varied domains without retraining.

In continuation of the development of our NeSy framework, we plan to

explore agentic approaches based on LLMs and use quantitative measures for capturing the reasoning capabilities of such agents. By improving the interpretability and robustness of our framework, we aim to ensure its practical applicability in real-world scenarios for autonomous tasks. Furthermore, we see potential in differentiable computational graphs, synthetic data augmentation of workflows, and the use of the VERTEX score (Dinu et al., 2024a) as a reward signal for reinforcement learning optimization. We also plan to extend our NeSy expression language with more interfaces and workflows.

In summary, by grounding our work in DA and NeSy approaches, we aim to develop more general, adaptable, interpretable, and scalable AI systems, capable of addressing a wide array of real-world challenges.

Bibliography

- Anthropic. The Claude 3 Model Family: Opus, Sonnet, Haiku. Technical report, Anthropic, July 2024. URL https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf.
- L. M. Augusto. *Computational Logic. Vol. 1: Classical Deductive Computing with Classical Logic.* College Publications, London, 2 edition, 2022.
- S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan. A theory of learning from different domains. In *Machine Learning*, volume 79, 2010.
- T. R. Besold, A. S. d. Garcez, S. Bader, H. Bowman, P. Domingos, P. Hitzler, K.-U. Kuehnberger, L. C. Lamb, D. Lowd, P. M. V. Lima, L. de Penning, G. Pinkas, H. Poon, and G. Zaverucha. Neural-symbolic learning and reasoning: A survey and interpretation, 2017.
- S. Black, G. Leo, P. Wang, C. Leahy, and S. Biderman. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. Technical report, Zenodo, Mar. 2021. URL <https://doi.org/10.5281/zenodo.5297715>. If you use this software, please cite it using these metadata.
- T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.
- N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021.

- G. A. Carpenter and S. Grossberg. Art 2: self-organization of stable category recognition codes for analog input patterns. *Appl. Opt.*, 26(23):4919–4930, Dec 1987. doi: 10.1364/AO.26.004919.
- C. Chen, Z. Fu, Z. Chen, S. Jin, Z. Cheng, X. Jin, and X.-S. Hua. Homm: Higher-order moment matching for unsupervised domain adaptation. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2020.
- N. Chomsky. *On certain formal properties of grammars*, pages 137–167. Open Journal of Modern Linguistics, 1959. ISBN 0019-9958.
- M.-C. Dinu. SymbolicAI: A Neuro-Symbolic Perspective on Large Language Models (LLMs). Technical report, GitHub, 11 2022. URL <https://github.com/Xpitfire/symbolicai>.
- M.-C. Dinu*, M. Hofmarcher*, V. P. Patil, M. Dorfer, P. M. Blies, J. Brandstetter, J. A. Arjona-Medina, and S. Hochreiter. Xai and strategy extraction via reward redistribution. In A. Holzinger, R. Goebel, R. Fong, T. Moon, K.-R. Müller, and W. Samek, editors, *xxAI - Beyond Explainable AI: International Workshop, Held in Conjunction with ICML 2020, July 18, 2020, Vienna, Austria, Revised and Extended Papers*, pages 177–205, Cham, 2022. Springer International Publishing. ISBN 978-3-031-04083-2. doi: 10.1007/978-3-031-04083-2_10.
- M.-C. Dinu, M. Holzleitner, M. Beck, H. D. Nguyen, A. Huber, H. Eghbalzadeh, B. A. Moser, S. V. Pereverzyev, S. Hochreiter, and W. Zellinger. Addressing parameter choice issues in unsupervised domain adaptation by aggregation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.
- M.-C. Dinu, C. Leoveanu-Condrei, M. Holzleitner, W. Zellinger, and S. Hochreiter. Symbolicai: A framework for logic-based approaches combining generative models and solvers. In *Conference on Lifelong Learning Agents, PMLR*, 2024a.
- M.-C. Dinu, C. Leoveanu-Condrei, M. Holzleitner, W. Zellinger, and S. Hochreiter. Symbolicai: A framework for logic-based approaches combining generative models and solvers. *GenAI4DM Workshop at The Twelfth International Conference on Learning Representations*, 2024b.
- I. Donadello, L. Serafini, and A. S. d’Avila Garcez. Logic tensor networks for semantic image interpretation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1596–1602, 2017.

- A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.
- M. Duan, A. Suri, N. Mireshghallah, S. Min, W. Shi, L. Zettlemoyer, Y. Tsvetkov, Y. Choi, D. Evans, and H. Hajishirzi. Do membership inference attacks work on large language models? *arXiv preprint arXiv:2402.07841*, 2024.
- B. Ehret, C. Henning, M. R. Cervera, A. Meulemans, J. von Oswald, and B. F. Grewe. Continual learning in recurrent neural networks with hypernetworks. *arXiv preprint arXiv:2006.12109*, 2020.
- K. Ellis. Human-like few-shot learning via bayesian reasoning over natural language. *arXiv preprint arXiv:2306.02797*, 2023.
- M. Fang, S. Deng, Y. Zhang, Z. Shi, L. Chen, M. Pechenizkiy, and J. Wang. Large language models are neurosymbolic reasoners. *arXiv preprint arXiv:2401.09334*, 2024.
- Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.
- A. S. d. Garcez and L. C. Lamb. Neurosymbolic ai: The 3rd wave. *arXiv preprint arXiv:2012.05876*, 2020.
- A. S. d. Garcez, L. C. Lamb, and D. M. Gabbay. *Neural-Symbolic Cognitive Reasoning*. Springer Publishing Company, Incorporated, 1 edition, 2008.
- A. S. d. Garcez, T. Besold, L. D. Raedt, P. Földiák, P. Hitzler, T. Icard, K. Kühnberger, L. Lamb, R. Miikkulainen, and D. Silver. Neural-symbolic learning and reasoning: Contributions and challenges. In *AAAI Conference*, 2015.
- A. S. d. Garcez, M. Gori, L. C. Lamb, L. Serafini, M. Spranger, and S. N. Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *Journal of Applied Logic*, 2019.
- Google. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
- A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012.

- K. Hamilton, A. Nayak, B. Božić, and L. Longo. Is neuro-symbolic AI meeting its promises in natural language processing? a structured review. *Semantic Web*, pages 1–42, nov 2022. doi: 10.3233/sw-223228.
- V. Hartmann, A. Suri, V. Bindschaedler, D. Evans, S. Tople, and R. West. Sok: Memorization in general-purpose large language models. *arXiv preprint arXiv:2310.18362*, 2023.
- D. G. Hays. *Chomsky Hierarchy*, page 210–211. John Wiley and Sons Ltd., GBR, 2003. ISBN 0470864125.
- D. O. Hebb. *The organization of behavior: A neuropsychological theory*. Wiley, New York, 1949. ISBN 0-8058-4300-0.
- M. T. Hicks, J. Humphries, and J. Slater. Chatgpt is bullshit. *Ethics and Information Technology*, 26(2):38, 2024. ISSN 1572-8439. doi: 10.1007/s10676-024-09775-5.
- S. Hochreiter. Toward a broad AI. *Commun. ACM*, 65(4):56–57, mar 2022. ISSN 0001-0782.
- M. Holzleitner, J. A. Arjona-Medina, M.-C. Dinu, A. Vall, L. Gruber, and S. Hochreiter. A two time-scale update rule ensuring convergence of episodic reinforcement learning algorithms at the example of rudder. *NeurIPS Optimization Foundations for Reinforcement Learning Workshop*, 2019.
- W. A. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, 1980. Original paper manuscript from 1969.
- Z. Hu, X. Ma, Z. Liu, E. Hovy, and E. Xing. Harnessing deep neural networks with logic rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2410–2420, Berlin, Germany, August 2016. Association for Computational Linguistics.
- J. Huang, X. Chen, S. Mishra, H. S. Zheng, A. W. Yu, X. Song, and D. Zhou. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*, 2024.
- L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, and T. Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*, 2023.
- W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. *CoRR*, abs/2201.07207, 2022. arXiv: 2201.07207.

- A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.
- T. Kaufmann, P. Weng, V. Bengs, and E. Hüllermeier. A survey of reinforcement learning from human feedback. *arXiv preprint arXiv:2312.14925*, 2024.
- H. Kim, G. Papamakarios, and A. Mnih. The lipschitz constant of self-attention. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5562–5571. PMLR, 18–24 Jul 2021.
- T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.
- L. C. Lamb, A. S. d. Garcez, M. Gori, M. Prates, P. Avelar, and M. Vardi. Graph neural networks meet neural-symbolic computing: A survey and perspective. In *AAAI Conference*, 2020.
- J. Li, E. Chen, Z. Ding, L. Zhu, K. Lu, and H. T. Shen. Maximum density divergence for domain adaptation. *CoRR*, abs/2004.12615, 2020.
- Q. Liu and H. Xue. Adversarial spectral kernel matching for unsupervised time series domain adaptation. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 30, 2021.
- X. Liu, C. Yoo, F. Xing, H. Oh, G. E. Fakhri, J.-W. Kang, J. Woo, et al. Deep unsupervised domain adaptation: A review of recent advances and perspectives. *APSIPA Transactions on Signal and Information Processing*, 11(1), 2022.
- R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester, and L. D. Raedt. DeepProbLog: Neural Probabilistic Logic Programming. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.
- J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.
- G. Marcus. The Next Decade in AI: Four Steps Towards Robust Artificial Intelligence. *arXiv preprint arXiv:2002.06177*, 2020.

- G. F. Marcus. *The Algebraic Mind: Integrating Connectionism and Cognitive Science*. The MIT Press, 2001. ISBN 9780262279086. In Special Collection: CogNet.
- J. McAuley. Recommendation on live-streaming platforms: Dynamic availability and repeat consumption. *Proceedings of the 15th ACM Conference on Recommender Systems*, 2021.
- M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In G. H. Bower, editor, *Psychology of Learning and Motivation*, volume 24, pages 109–165. Academic Press, 1989. doi: 10.1016/S0079-7421(08)60536-8.
- L. McInnes, J. Healy, N. Saul, and L. Großberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29): 861, 2018. doi: 10.21105/joss.00861. URL <https://doi.org/10.21105/joss.00861>.
- M. Mermilliod, A. Bugajska, and P. Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in psychology*, 4:504, 2013.
- S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi, and L. Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022.
- K. Musgrave, S. Belongie, and S.-N. Lim. Unsupervised domain adaptation: A reality check. *arXiv preprint arXiv:2111.15672*, 2021.
- A. Newell and H. A. Simon. Human problem solving. *Prentice-Hall*, page 920, 1972.
- H. Nori, Y. T. Lee, S. Zhang, D. Carignan, R. Edgar, N. Fusi, N. King, J. Larson, Y. Li, W. Liu, R. Luo, S. M. McKinney, R. O. Ness, H. Poon, T. Qin, N. Usuyama, C. White, and E. Horvitz. Can generalist foundation models outcompete special-purpose tuning? case study in medicine. *arXiv preprint arXiv:2311.16452*, 2023.
- NVIDIA. Nvidia gtc 2024 keynote. <https://www.nvidia.com/gtc/>, 2024. Accessed: 2024-07-05.
- OpenAI. GPT-4 Technical Report. *arXiv*, 2023.
- J. V. Oswald, E. Niklasson, E. Randazzo, J. Sacramento, A. Mordvintsev, A. Zhmoginov, and M. Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR, 2023.

- L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. E. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. J. Lowe. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.
- A. Patel, M. Hofmarcher, C. Leoveanu-Condrei, M.-C. Dinu, C. Callison-Burch, and S. Hochreiter. Large language models can self-improve at web agent tasks. In *Advances in Neural Information Processing Systems (under review)*, 2024.
- V. Patil*, M. Hofmarcher*, M.-C. Dinu, M. Dorfer, P. M. Blies, J. Brandstetter, J. Arjona-Medina, and S. Hochreiter. Align-RUDDER: Learning from few demonstrations by reward redistribution. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 17531–17572. PMLR, 17–23 Jul 2022.
- X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang. Moment matching for multi-source domain adaptation. *CoRR*, abs/1812.01754, 2018.
- M. Qu and J. Tang. Probabilistic logic neural networks for reasoning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019.
- M. Ragab, E. Eldele, W. L. Tan, C.-S. Foo, Z. Chen, M. Wu, C.-K. Kwoh, and X. Li. Adatime: A benchmarking suite for domain adaptation on time series data. *ACM Trans. Knowl. Discov. Data*, 17(8), may 2023. ISSN 1556-4681. doi: 10.1145/3587937.
- M. M. Rahman, C. Fookes, M. Baktashmotagh, and S. Sridharan. On minimum discrepancy estimation for deep domain adaptation. *Domain Adaptation for Visual Understanding*, 2020.
- H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, T. Adler, L. Gruber, M. Holzleitner, M. Pavlović, G. K. Sandve, et al. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2020.
- L. Ruis, A. Khan, S. Biderman, S. Hooker, T. Rocktäschel, and E. Grefenstette. Large language models are not zero-shot communicators. *CoRR*, abs/2210.14986, 2022. doi: 10.48550/arXiv.2210.14986.
- K. Saito, D. Kim, P. Teterwak, S. Sclaroff, T. Darrell, and K. Saenko. Tune it the right way: Unsupervised validation of domain adaptation via soft neighborhood density. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9184–9193, 2021.

- A. Scherlis, K. Sachan, A. S. Jermyn, J. Benton, and B. Shlegeris. Polysemanicity and capacity in neural networks. *arXiv preprint arXiv:2210.01892*, 2022.
- T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom. Toolformer: Language models can teach themselves to use tools, 2023.
- J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020. doi: 10.1038/s41586-020-03051-4.
- K. Schweighofer, M. Hofmarcher, M.-C. Dinu, P. Renz, A. Bitto-Nemling, V. P. Patil, and S. Hochreiter. Understanding the Effects of Dataset Characteristics on Offline Reinforcement Learning. In *Deep RL Workshop NeurIPS 2021*, 2021.
- K. Schweighofer*, A. Radler*, M.-C. Dinu*, M. Hofmarcher, V. P. Patil, A. Bitto-nemling, H. Eghbal-zadeh, and S. Hochreiter. A dataset perspective on offline reinforcement learning. In S. Chandar, R. Pascanu, and D. Precup, editors, *Proceedings of The 1st Conference on Lifelong Learning Agents*, volume 199 of *Proceedings of Machine Learning Research*, pages 470–517. PMLR, 22–24 Aug 2022.
- D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. P. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. doi: 10.1038/nature16961.
- D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017a.
- D. Silver, J. Schrittwieser, K. Simonyan, et al. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017b. doi: 10.1038/nature24270.
- K. Singhal, S. Azizi, T. Tu, S. S. Mahdavi, J. Wei, H. W. Chung, N. Scales, et al. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180, 2023.
- R. Siripurapu, V. P. Patil, K. Schweighofer, M.-C. Dinu, T. Schmied, L. E. F. Diez, M. Holzleitner, H. Eghbal-Zadeh, M. K. Kopp, and S. Hochreiter.

- Infodist: Online distillation with informative rewards improves generalization in curriculum learning. In *Deep Reinforcement Learning Workshop NeurIPS*, 2022.
- C. A. Steinparz, T. Schmied, F. Paischer, M.-C. Dinu, V. P. Patil, A. Bitto-Nemling, H. Eghbal-zadeh, and S. Hochreiter. Reactive exploration to cope with non-stationarity in lifelong reinforcement learning. In *Conference on Lifelong Learning Agents*, pages 441–469. PMLR, 2022.
- M. Sugiyama, M. Krauledat, and K.-R. Müller. Covariate shift adaptation by importance weighted cross validation. *J. Mach. Learn. Res.*, 8:985–1005, dec 2007. ISSN 1532-4435.
- M. Sugiyama, T. Suzuki, and T. Kanamori. *Density Ratio Estimation in Machine Learning*. Cambridge University Press, 2012.
- B. Sun, J. Feng, and K. Saenko. Correlation alignment for unsupervised domain adaptation. *Domain Adaptation in Computer Vision Applications*, pages 153–171, 2017.
- Tesla. Tesla AI day 2022 live stream. <https://www.teslaclub.at/tesla-ai-day/>, 2022. Accessed: 2024-07-05.
- H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: Open and efficient foundation language models, 2023.
- E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.
- Y. Weng, M. Zhu, F. Xia, B. Li, S. He, S. Liu, B. Sun, K. Liu, and J. Zhao. Large language models are better reasoners with self-verification. *arXiv preprint arXiv:2212.09561*, 2023.

- G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23:69–101, 1996.
- S. Wolfram. *What is ChatGPT Doing ... and why Does it Work?* Wolfram Media, Incorporated, 2023. ISBN 9781579550813.
- Z. Xu, H. van Hasselt, and D. Silver. Meta-gradient reinforcement learning. *ArXiv*, 2018.
- S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.
- K. You, X. Wang, M. Long, and M. Jordan. Towards accurate model selection in deep unsupervised domain adaptation. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7124–7133. PMLR, 09–15 Jun 2019.
- D. Yu, B. Yang, D. Liu, H. Wang, and S. Pan. A survey on neural-symbolic learning systems. *Neural Networks*, 166:105–126, 2023. ISSN 0893-6080.
- W. Zellinger, T. Grubinger, E. Lughofer, T. Natschläger, and S. Saminger-Platz. Central moment discrepancy (cmd) for domain-invariant representation learning. *International Conference on Learning Representations*, 2017.
- W. Zellinger, N. Shepeleva, M.-C. Dinu, H. Eghbal-zadeh, H. D. Nguyen, B. Nessler, S. Pereverzyev, and B. A. Moser. The balancing principle for parameter choice in distance-regularized domain adaptation. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 20798–20811. Curran Associates, Inc., 2021.
- H. Zhao, F. Yang, H. Lakkaraju, and M. Du. Towards uncovering how large language model works: An explainability perspective. *arXiv preprint arXiv:2402.10688*, 2024.
- Y. Zhu, F. Zhuang, J. Wang, G. Ke, J. Chen, J. Bian, H. Xiong, and Q. He. Deep subdomain adaptation network for image classification. *IEEE Transactions on Neural Networks and Learning Systems*, 32(4):1713–1722, 2021.

Appendix A

Glossary

- AI Artificial intelligence.
- DA Domain Adaptation.
- IWA Importance Weighted Aggregation.
- LLM Large Language Model.
- LLMs Large Language Models.
- NeSy Neuro-Symbolic.
- UDA Unsupervised Domain Adaptation.