# Distribution-Based Invariant Deep Networks for Learning Meta-Features

**Gwendoline De Bie**[1] **Herilalaina Rakotoarison**[2] **Gabriel Peyré**[1] **Michele Sebag**[2]

[1]ENS, PSL University, Paris, France
[2]TAU, LISN-CNRS–INRIA, Université Paris-Saclay, Orsay, France

**Abstract** Recent advances in deep learning from probability distributions successfully achieve classification or regression from distribution samples, thus invariant under permutation of the samples. The first contribution of the paper is the Dida distributional architecture, extending the state of the art to achieve invariance under permutation of the features, too. The Dida properties of universal approximation, and robustness with respect to bounded transformations of the input distribution, are established. The second contribution is to empirically demonstrate the merits of the Dida architecture on two tasks defined at the dataset level. The first task consists of predicting whether any two dataset patches are extracted from the same initial dataset. The second task consists of predicting whether a hyper-parameter configuration dominates another configuration, in terms of the learning performance of a fixed learning algorithm on a dataset extracted from the OpenML benchmarking suite. On both tasks, Dida outperforms the state of the art as well as models based on hand-crafted meta-features. The penultimate layer neurons can thus be viewed as *learned* meta-features, defining an accurate and computationally affordable description of datasets.

## 1 Introduction

Deep networks architectures, initially devised for structured data such as images and speech, have been extended to enforce some invariance or equivariance properties (Shawe-Taylor, 1993) for more complex data representations.[1] The merit of invariant or equivariant neural architectures is twofold. On the one hand, they inherit the universal approximation properties of neural nets (Cybenko, 1989; Leshno et al., 1993). On the other hand, the fact that these architectures comply with the invariances attached to the considered data representation yields more robust and more general models (through constraining the neural weights and/or reducing the number of weights, as examplified by convolutional networks). For instance, when considering point clouds (Qi et al., 2017) or probability distributions (Bie et al., 2019), the network output is required to be invariant with respect to permutations of the input points.

**Related works**. Invariance or equivariance properties are relevant to a wide range of applications. In the sequence-to-sequence framework, one might want to relax the sequence order (Vinyals et al., 2016). When modelling dynamic cell processes, one might want to follow the cell evolution at a macroscopic level, in terms of distributions as opposed to, a set of individual cell trajectories (Hashimoto et al., 2016). In computer vision, one might want to handle a set of pixels, as opposed to a voxellized representation, for the sake of a better scalability in terms of data dimensionality and computational resources (Bie et al., 2019).

On the theoretical side, neural architectures enforcing invariance or equivariance properties have been pioneered by (Hartford et al., 2018). Characterizations of invariance or equivariance under group actions have been proposed in the finite (Ravanbakhsh et al., 2017) or infinite case

[1]Function $f : X \mapsto Y$ is said to be invariant under operator $\sigma$ defined on domain $X$ iff $f(\sigma(x)) = f(x)$ for all $x$ in $X$. Function $f : X \mapsto X$ is said to be equivariant iff $f(\sigma(x)) = \sigma(f(x))$ for all $x$ in $X$.

 

(Kondor and Trivedi, 2018). (Maron et al., 2018; Keriven and Peyré, 2019) have proposed a general characterization of linear layers enforcing invariance or equivariance properties with respect to the whole permutation group on the feature set. The universal approximation properties of such architectures have been established in the case of sets (Zaheer et al., 2017), point clouds (Qi et al., 2017), discrete measures (Bie et al., 2019), invariant (Maron et al., 2019) and equivariant (Keriven and Peyré, 2019) graph neural networks. (Maron et al., 2020) presents a neural architecture invariant w.r.t. the ordering of points and their features, handling point clouds.

**Motivations**. This paper aims to build representations of datasets through *learned meta-features*. Meta-features, meant to represent a dataset as a vector of characteristics, have been mentioned in the ML literature for over 40 years, in relation with several key ML challenges: a) learning a performance model, predicting *a priori* the performance of an algorithm (and the hyper-parameters thereof) on a dataset (Rice, 1976; Hutter et al., 2019); b) learning a generic model able of quick adaptation to new tasks, e.g. one-shot or few-shot learning, through the so-called meta-learning approach (Finn et al., 2017; Baz et al., 2021); c) hyper-parameter transfer learning (Perrone et al., 2018).

A large number of meta-features have been manually designed along the years (Muñoz et al., 2018; Rivolli et al., 2022), ranging from sufficient statistics to the so-called *landmarks* (Pfahringer et al., 2000), computing the performance of (fast) ML algorithms on the considered dataset. The challenge is the following: on the one hand meta-features should capture the joint distribution underlying the dataset in order to help tackling tasks a), b) and c); on the other hand, the meta-features should be sufficiently fast to compute to make sense using them (compared to tackling the above tasks using brute force). How to *learn meta-features* has been first investigated by (Jomaa et al., 2021) to our best knowledge. The authors build the Dataset2Vec representation by tackling a supervised learning problem at the dataset level: specifically, given two dataset patches, that is, two subsets of examples, described by two (different) subsets of features, Dataset2Vec is trained to predict whether those patches are extracted from the same initial dataset. In the same line of approach, Meskhi et al. (2021) have proposed to predict algorithm performances, then consider as meta-features the representations extracted at the last hidden layer.

**Contributions**. In order to learn meta-features, this paper proposes a new *distribution-based invariant deep architecture* (Dida), which is independent of the dimension $d$ of the distribution support. The merits of the Dida architecture are experimentally demonstrated on two tasks defined at the dataset level, significantly outperforming state of art architectures (Maron et al., 2020; Jomaa et al., 2021; Muñoz et al., 2018) on these tasks (Section 3).

The novelty of the approach is to handle continuous and discrete probability distributions on $\mathbb{R}^d$, extending state of art approaches dealing with point clouds (Maron et al., 2020; Jomaa et al., 2021). This extension yields more general approximation results (Appendix E).

**Notations**.. $[\![1\,;m]\!]$ denotes the set of integers $\{1,\dots m\}$. Distributions, including discrete distributions (datasets) are noted in bold font. Vectors are noted in italic, with $x[k]$ denoting the $k$-th coordinate of vector $x$.

## 2 Distribution-Based Invariant Networks for Meta-Feature Learning

This section describes the core of the proposed Dida architecture, specifically the mechanism of mapping a point distribution onto another one subject to sample and feature permutation invariance, referred to as *invariant layer*. The Lipchitzness and universal approximation properties of Dida , which guarantee both its robustness and expressiveness, are discussed in Appendix E.

### 2.1 Distribution-Based Invariant Layers

The building block of the proposed architecture, the invariant layer meant to satisfy the feature and label invariance requirements, is defined as follows, taking inspiration from Bie et al. (2019).

**Definition 1**. *(Distribution-based invariant layers) Let an interaction functional $\varphi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^r$ be $G$-invariant:*

$$\forall \sigma, z_1, z_2 \in G \times \mathbb{R}^d \times \mathbb{R}^d, \quad \varphi(z_1, z_2) = \varphi(\sigma(z_1), \sigma(z_2)).$$

*The distribution-based invariant layer $f_\varphi$ is defined as $f_\varphi : \mathbf{z} = (z_i)_{i \in [\![1;n]\!]} \in Z(\mathbb{R}^d) \mapsto f_\varphi(\mathbf{z}) \in Z(\mathbb{R}^r)$ with*

$$f_\varphi(\mathbf{z}) \stackrel{\text{def.}}{=} \left( \frac{1}{n} \sum_{j=1}^{n} \varphi(z_1, z_j), \ldots, \frac{1}{n} \sum_{j=1}^{n} \varphi(z_n, z_j) \right) \tag{1}$$

By construction, $f_\varphi$ is $G$-invariant if $\varphi$ is $G$-invariant. The construction of $f_\varphi$ is extended to the general case of possibly continuous probability distributions by replacing sums with integrals (Appendix F).

It is important that $f_\varphi$ invariant layers (in particular the first layer of the neural architecture) can handle datasets of arbitrary number of features $d_X$ and number of multi-labels $d_Y$. An original approach is to define $\varphi$ as follows. Let $z = (x, y)$ and $z' = (x', y')$ be two samples in $\mathbb{R}^{d_X} \times \mathbb{R}^{d_Y}$. Considering two functions (to be learned) $u : \mathbb{R}^4 \mapsto \mathbb{R}^t$ and $v : \mathbb{R}^t \mapsto \mathbb{R}^r$, then $\varphi$ is obtained by applying $v$ on the sum of $u(x[k], x'[k], y[\ell], y'[\ell])$ for $k$ ranging in $[\![1; d_X]\!]$ and $\ell$ in $[\![1; d_Y]\!]$:

$$\varphi(z, z') = v \left( \sum_{k=1}^{d_X} \sum_{\ell=1}^{d_Y} u(x[k], x'[k], y[\ell], y'[\ell]) \right) \tag{2}$$

By construction $\varphi$ is invariant to both feature and label permutations; this invariance property is instrumental to a good empirical performance (Section 3). Note that (after learning $u$ and $v$, implementation details in Appendix C) $f_\varphi$ can map a $n$-size dataset $\mathbf{z}$ onto an $n$-size $f_\varphi(\mathbf{z})$ dataset for any arbitrary $n$. The overall complexity of $f_\varphi$ is thus $\mathcal{O}(n^2.d_X.d_Y)$.

As said, $f_\varphi$ is based on interaction functionals $\varphi(z_i, z_j)$. This original architecture is rooted in theoretical and algorithmic motivations. On the one hand, interaction functionals are crucial components to reach universal approximation results (see Appendix H, Theorem 2). On the other hand, the use of local interactions allows to create more expressive architectures; the benefit of these architectures is illustrated in the experiments (Section 3).

### 2.2 Learning from distributions

Dida distributional neural architecture, defined on point distributions, maps a multi-labelled dataset $\mathbf{z} \in \mathrm{Z}(\mathbb{R}^d)$ onto a real-valued vector noted $\mathcal{F}_\zeta(\mathbf{z})$, with

$$\mathcal{F}_\zeta(\mathbf{z}) \stackrel{\text{def.}}{=} f_{\varphi_m} \circ \ldots \circ f_{\varphi_{o+1}} \circ f_{\varphi_o} \circ \ldots \circ f_{\varphi_1}(\mathbf{z}) \in \mathbb{R}^{d_{m+1}} \tag{3}$$

where $\zeta$ are the trainable parameters of the architecture (below). This architecture inherits from Lipschitzness of the interaction functional $\varphi$, which guarantees its robustness to input perturbation, as well as universal approximation abilities denoting its expressiveness. Both results are detailed in the general multi-labelled case, in Appendix E. For simplicity, only the single label case ($d_Y = 1$) is considered in the following.

The first invariant layer is defined from $\varphi_1$, mapping pairs of vectors in $\mathbb{R}^d$ ($d_1 = d$) onto $\mathbb{R}^{d_2}$; it is possibly followed by other invariant layers (the impact of using 1 *vs* 2 invariant layers is experimentally studied in Section 3). The last $o$-th invariant layer is followed by a first non-invariant one, defined from some $\varphi_{o+1}$ only depending on its second argument; it is possibly followed by other standard layers. The functions defined from the neural nodes on the penultimate layer are referred to as meta-features.

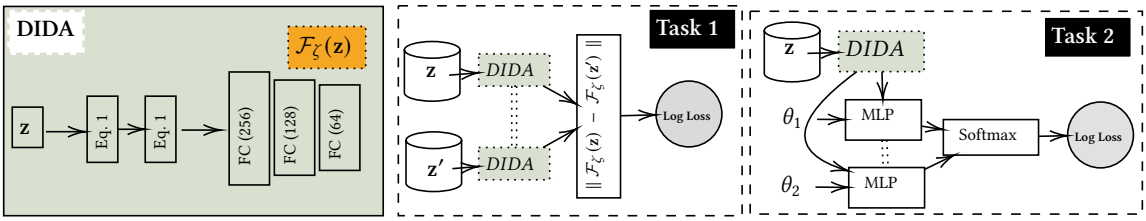


Figure 1: (Left) The DIDA architecture (FC for fully connected layer). (Middle) Task 1: Learning meta-features for patch identification using a Siamese architecture (Appendix B.1). (Right) Task 2: learning meta-features for ranking hyper-parameter configurations $\theta_1$ and $\theta_2$ (Appendix B.2).

## 3 Experimental Validation

All source codes (DIDA and baselines) are publicly available at https://github.com/herilalaina/dida-metafeatures for the sake of reproducibility.

### 3.1 Experimental setting

Two evaluation tasks are considered:
→ **Task 1** is a patch identification problem inspired from (Jomaa et al., 2021) aiming to identify if two dataset patches are extracted from a same dataset.
→ **Task 2** aims to rank hyper-parameter configurations for a fixed supervised learning algorithm, according to their performance on the considered dataset.

DIDA is compared to three baselines (detailed in Appendix C): three DSS (Maron et al., 2020) variants (linear invariant layers, non-linear invariant layers, and equivariant + invariant layers); DATASET2VEC (Jomaa et al., 2021); and a function of 43 hand-crafted meta-features.

Three benchmarks are used: TOY and UCI, taken from (Jomaa et al., 2021), and OpenML CC-18 (Bischl et al., 2019). The selection and selection of patches are detailed in Appendix A.

**Training setups.** The same DIDA architectures are used for both tasks, involving 1 or 2 invariant layers followed by 3 fully connected (FC) layers (Figure 1, left). All experiments run on 1 NVIDIA-Tesla-V100-SXM2 GPU with 32GB memory, using Adam optimizer with base learning rate $10^{-3}$ and batch size 32. For all considered architectures, meta-features $\mathcal{F}_\zeta(\mathbf{z})$ consist of the output of the penultimate layer, with $\zeta$ denoting the trained parameters.

### 3.2 Results

**Task 1.** Table 1 reports the empirical results on TOY and UCI datasets. On TOY, DIDA with 2 invariant layers, referred to as 2L-DIDA behaves on a par with DATASET2VEC and DSS. On UCI, the task appears to be more difficult, which is explained from the higher and more diverse number of features in the datasets. The fact that 2L-DIDA significantly outperforms all other approaches is explained from the interaction functional structure (Eqs. 1, 2), expected to better grasp contrasts among examples. DIDA with 1 invariant layer (1L-DIDA) is much behind 2L-DIDA; with a significantly lesser number of parameters than 2L-DIDA, the 1L-DIDA architecture might lack representational power. A fourth baseline, NO-FINV-DSS (Zaheer et al., 2017) only differs from DSS as it is *not* feature permutation invariant; this additional baseline is used to assess the impact of this invariance property. The fact that NO-FINV-DSS lags behind all DSS variants, all with similar number of parameters, confirms the importance of this invariance property. Note also that NO-FINV-DSS is outperformed by 1L-DIDA, while the latter involves significantly less parameters.

**Task 2.** The comparative performances are displayed in Table 2, reporting their ranking accuracy. 2L-DIDA (respectively 1L-DIDA) significantly outperforms all baseline approaches except in the *Alg* = LR case (resp., in the *Alg* = $k$-NN case). A higher performance gap is observed for the k-NN case, which is explained as this algorithm mostly exploits the local geometry of the examples.

| Method | # params | TOY | UCI |
|---|---|---|---|
| Hand-crafted | 53,312 | 77.05 %± 1.63 | 58.36 %± 2.64 |
| No-FInv-DSS (no inv. in features) | 1,297,692 | 90.49 %± 1.73 | 64.69 %± 4.89 |
| Dataset2Vec (reported from Jomaa et al. (2021)) | - | 96.19 %± 0.28 | 88.20 %± 1.67 |
| Dataset2Vec (our implementation) | 257,088 | 97.90 %± 1.87 | 77.05 %± 3.49 |
| DSS layers (Linear aggregation) | 1,338,684 | 89.32 %± 1.85 | 76.23 %± 1.84 |
| DSS layers (Non-linear aggregation) | 1,338,684 | 96.24 %± 2.04 | 83.97 %± 2.89 |
| DSS layers (Equivariant+invariant) | 1,338,692 | 96.26 %± 1.40 | 82.94 %± 3.36 |
| Dida (1 invariant layer) | 323,028 | 91.37 %± 1.39 | 81.03 %± 3.23 |
| Dida (2 invariant layers) | 1,389,089 | 97.20 % ± 0.10 | **89.70** % ± **1.89** |

Table 1: Comparative performances (average and std of accuracy over 10 runs) on Task 1 of Dida, No-FInv-DSS, Dataset2Vec, DSS and functions of hand-crafted meta-features.

| Method | SGD | SVM | LR | k-NN |
|---|---|---|---|---|
| Hand-crafted | 71.18 %± 0.41 | 75.39 %± 0.29 | 86.41 %± 0.419 | 65.44 %± 0.73 |
| Dataset2Vec (our implementation) | 74.43 %± 0.90 | 81.75 %± 1.85 | 89.18 %± 0.45 | 72.90 %± 1.13 |
| DSS (Linear aggregation) | 73.46 %± 1.44 | 82.91 %± 0.22 | 87.93 %± 0.58 | 70.07 %± 2.82 |
| DSS (Equivariant+Invariant) | 73.54 %± 0.26 | 81.29 %± 1.65 | 87.65 %± 0.03 | 68.55 %± 2.84 |
| DSS (Non-linear aggregation) | 74.13 %± 1.01 | 83.38 %± 0.37 | 87.92 %± 0.27 | 73.07 %± 0.77 |
| DIDA (1 invariant layer) | 77.31 %± 0.16 | 84.05 %± 0.71 | **90.16** %± **0.17** | 74.41 %± 0.93 |
| DIDA (2 invariant layers) | **78.41** %± **0.41** | **84.14** %± **0.02** | 89.77 %± 0.50 | **78.91** %± **0.54** |

Table 2: Comparative ranking performances (average and std over 3 runs) of Dida, Dataset2Vec, DSS and functions of hand-crafted meta-features.

## 4 Conclusion

The contribution of the paper is the Dida architecture, able to learn from discrete and continuous distributions on $\mathbb{R}^d$, invariant w.r.t. feature ordering, agnostic w.r.t. the size and dimension $d$ of the considered distribution sample (with $d$ less than some upper bound $D$). The merits of Dida are empirically and comparatively demonstrated on two tasks defined at the dataset level. Task 2 in particular constitutes a first step toward performance modelling Rice (1976), as the learned (algorithm-dependent) meta-features support an efficient ranking of the configurations for the current dataset. On the considered tasks, they improve on the considered baselines namely, Dataset2Vec, DSS and meta-features manually defined in the last two decades (Muñoz et al., 2018). Besides, this Dida architecture also enjoys universal approximation and robustness properties.

For further work, an initial perspective is to investigate the relationships between two datasets, and estimate *a priori* the chances of a successful domain adaptation (Alvarez-Melis and Fusi, 2021).

**Limitations and Broader Impact Statement**. A major limitation of Dida is on handling real datasets which may include missing values, categorical variables and outliers. An another challenge is that learning meta-features for AutoML tasks (e.g. recommending hyper-parameters or initializing optimization algorithms) requires sufficiently many datasets: quite a few of our early attempts failed due to current ML benchmarks being not sufficiently representative.

Dida requires an extensive compute resources (e.g. circa 4 hours on Task 1.TOY) to be effective. Nevertheless, the approach opens key perspective for AutoML in overcoming the need for domain experts, especially, when it comes to describing and comparing datasets.

## 5 Reproducibility Checklist

1. For all authors...

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]

   (b) Did you describe the limitations of your work? [Yes]

   (c) Did you discuss any potential negative societal impacts of your work? [Yes]

   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...

   (a) Did you state the full set of assumptions of all theoretical results? [Yes]

   (b) Did you include complete proofs of all theoretical results? [Yes]

3. If you ran experiments...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all requirements (e.g., `requirements.txt` with explicit version), an instructive `README` with installation, and execution commands (either in the supplemental material or as a URL)? [Yes] They are available at https://anonymous.4open.science/r/dida-metafeatures-5FD5/.

   (b) Did you include the raw results of running the given instructions on the given code and data? [Yes] They are summarized in Tables 1 and 2.

   (c) Did you include scripts and commands that can be used to generate the figures and tables in your paper based on the raw results of the code, data, and instructions given? [N/A]

   (d) Did you ensure sufficient code quality such that your code can be safely executed and the code is properly documented? [No]

   (e) Did you specify all the training details (e.g., data splits, pre-processing, search spaces, fixed hyperparameter settings, and how they were chosen)? [Yes] They are described in Appendix A.

   (f) Did you ensure that you compared different methods (including your own) exactly on the same benchmarks, including the same datasets, search space, code for training and hyperparameters for that code? [Yes]

   (g) Did you run ablation studies to assess the impact of different components of your approach? [Yes]

   (h) Did you use the same evaluation protocol for the methods being compared? [Yes]

   (i) Did you compare performance over time? [No]

   (j) Did you perform multiple runs of your experiments and report random seeds? [Yes]

   (k) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]

   (l) Did you use tabular or surrogate benchmarks for in-depth evaluations? [N/A]

   (m) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

(n) Did you report how you tuned hyperparameters, and what time and resources this required (if they were not automatically tuned by your AutoML method, e.g. in a NAS approach; and also hyperparameters of your own method)? [No]DIDA and DSS are manually tuned.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

   (a) If your work uses existing assets, did you cite the creators? [Yes]

   (b) Did you mention the license of the assets? [No]

   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

**Acknowledgements**. The authors have many people to thank!

## References


Alvarez-Melis, D. and Fusi, N. (2021). Dataset Dynamics via Gradient Flows in Probability Space. In *Proceedings of the 38th International Conference on Machine Learning*, pages 219–230. PMLR.

Baz, A. E., Guyon, I., Liu, Z., Rijn, J. N. v., Treguer, S., and Vanschoren, J. (2021). Advances in MetaDL: AAAI 2021 Challenge and Workshop. In *AAAI Workshop on Meta-Learning and MetaDL Challenge*, pages 1–16. PMLR.

Bie, G. D., Peyré, G., and Cuturi, M. (2019). Stochastic Deep Networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 1556–1565. PMLR.

Bischl, B., Casalicchio, G., Feurer, M., Hutter, F., Lang, M., Mantovani, R. G., van Rijn, J. N., and Vanschoren, J. (2019). OpenML Benchmarking Suites. *arXiv:1708.03731 [cs, stat]*. arXiv: 1708.03731.

Cox, D. A., Little, J. N., and O'Shea, D. (2018). *Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra*. OCLC: 1241676194.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314.

Dua, D. and Graff, C. (2017). {UCI} Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *arXiv:1703.03400 [cs]*. arXiv: 1703.03400.

Hartford, J., Graham, D., Leyton-Brown, K., and Ravanbakhsh, S. (2018). Deep Models of Interactions Across Sets. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1909–1918. PMLR.

Hashimoto, T., Gifford, D., and Jaakkola, T. (2016). Learning Population-Level Diffusions with Generative RNNs. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 2417–2426. PMLR.

Hutter, F., Kotthoff, L., and Vanschoren, J., editors (2019). *Automated Machine Learning: Methods, Systems, Challenges*. The Springer Series on Challenges in Machine Learning. Springer International Publishing, Cham.

Jomaa, H. S., Schmidt-Thieme, L., and Grabocka, J. (2021). Dataset2Vec: learning dataset meta-features. *Data Mining and Knowledge Discovery*, 35(3):964–985.

Keriven, N. and Peyré, G. (2019). Universal Invariant and Equivariant Graph Neural Networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Kondor, R. and Trivedi, S. (2018). On the Generalization of Equivariance and Convolution in Neural Networks to the Action of Compact Groups. In Dy, J. G. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2752–2760. PMLR.

Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. (2018). Invariant and Equivariant Graph Networks.

Maron, H., Fetaya, E., Segol, N., and Lipman, Y. (2019). On the Universality of Invariant Networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 4363–4371. PMLR.

Maron, H., Litany, O., Chechik, G., and Fetaya, E. (2020). On Learning Sets of Symmetric Elements. In *Proceedings of the 37th International Conference on Machine Learning*, pages 6734–6744. PMLR.

Meskhi, M. M., Rivolli, A., Mantovani, R. G., and Vilalta, R. (2021). Learning Abstract Task Representations. In *AAAI Workshop on Meta-Learning and MetaDL Challenge*, pages 127–137. PMLR.

Muñoz, M. A., Villanova, L., Baatar, D., and Smith-Miles, K. (2018). Instance spaces for machine learning classification. *Machine Learning*, 107(1):109–147.

Perrone, V., Jenatton, R., Seeger, M. W., and Archambeau, C. (2018). Scalable Hyperparameter Transfer Learning. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Peyré, G. and Cuturi, M. (2019). Computational Optimal Transport: With Applications to Data Science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607.

Pfahringer, B., Bensusan, H., and Giraud-Carrier, C. G. (2000). Meta-Learning by Landmarking Various Learning Algorithms. In Langley, P., editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pages 743–750. Morgan Kaufmann.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017). PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. pages 652–660.

Rahman, Q. I. and Schmeisser, G. (2002). *Analytic theory of polynomials*. Number new ser., 26 in London Mathematical Society monographs. Clarendon Press ; Oxford University Press, Oxford : New York.

Ravanbakhsh, S., Schneider, J., and Póczos, B. (2017). Equivariance Through Parameter-Sharing. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2892–2901. PMLR.

Rice, J. R. (1976). The Algorithm Selection Problem. In Rubinoff, M. and Yovits, M. C., editors, *Advances in Computers*, volume 15, pages 65–118. Elsevier.

Rivolli, A., Garcia, L. P. F., Soares, C., Vanschoren, J., and de Carvalho, A. C. P. L. F. (2022). Meta-features for meta-learning. *Knowledge-Based Systems*, 240:108101.

Santambrogio, F. (2015). *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*. Number 87 in Progress in Nonlinear Differential Equations and Their Applications. Springer International Publishing, Cham, 1st ed. 2015 edition.

Shawe-Taylor, J. (1993). Symmetries and discriminability in feedforward network architectures. *IEEE Transactions on Neural Networks*, 4(5):816–826.

Vinyals, O., Bengio, S., and Kudlur, M. (2016). Order Matters: Sequence to sequence for sets. In Bengio, Y. and LeCun, Y., editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. (2017). Deep Sets. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

# Details of Empirical Validation

## A Benchmark Details

Three benchmarks are used (Table 3): TOY and UCI, taken from Jomaa et al. (2021), and OpenML CC-18. TOY includes 10,000 datasets, where instances are distributed along mixtures of Gaussian, intertwinning moons and rings in $\mathbb{R}^2$, with 2 to 7 classes. UCI includes 121 datasets from the UCI Irvine repository Dua and Graff (2017). Datasets UCI and OpenML are normalized as follows: categorical features are one-hot encoded; numerical features are normalized; missing values are imputed with the feature mean (continuous features) or median (for categorical features). Patches are defined as follows. Given an initial dataset, a number $d_X$ of features and a number $n$ of examples are uniformly selected in the considered ranges (depending on the benchmark) described in Table 3. A patch is defined by (i) retaining $n$ examples uniformly selected with replacement in this initial dataset; (ii) retaining $d_X$ features uniformly selected with replacement among the initial features.

| | Datasets | | | Patches | |
|---|---|---|---|---|---|
| | # datasets | # samples | # features | # samples | # features |
| Toy Dataset | 10000 | [2048, 8192] | 2 | 200 | 2 |
| UCI | 121 | [10, 130064] | [3, 262] | [200, 500] | [2, 15] |
| OpenML CC-18 | 71 | [500, 100000] | [5, 3073] | [700, 900] | [3, 11] |

Table 3: Benchmarks and patches characteristics.

## B Evaluation Tasks

### B.1 Task 1: Patch identification

In Task 1, patches are extracted from datasets and the task consists in predicting whether two patches are extracted from the same dataset. Letting $\mathbf{u}$ denote a dataset with $n$ $d$-dimensional examples, patch $\mathbf{z}$ is constructed from $\mathbf{u}$, by selecting (uniformly with replacement) $n_z$ examples in $\mathbf{u}$ and considering their description based on $d_z$ features selected uniformly with replacement among $\mathbf{u}$ features. Size $n_z$ and number $d_z$ of features of the patch are uniformly selected (Table 3). In Task 1, an example is made of a pair of patches $(\mathbf{z}, \mathbf{z}')$, together with its associated label $\ell(\mathbf{z}, \mathbf{z}')$, set to 1 iff $\mathbf{z}$ and $\mathbf{z}$' are extracted from the same initial dataset $\mathbf{u}$ and $n_z = n_{z'}$.

For all considered architectures, the parameters are trained using a Siamese architecture (Figure 1, middle; Algorithm 1, Appendix B). The learned classifier $\hat{\ell}_\zeta(\mathbf{z}, \mathbf{z}')$ is the softmax $\exp\left(-\|\mathcal{F}_\zeta(\mathbf{z}) - \mathcal{F}_\zeta(\mathbf{z}')\|_2\right)$, with $\mathcal{F}_\zeta(\mathbf{z})$ and $\mathcal{F}_\zeta(\mathbf{z}')$ the meta-features computed for $\mathbf{z}$ and $\mathbf{z}'$, where $\zeta$ is trained to minimize the cross-entropy loss:

$$\sum_{\mathbf{z},\mathbf{z}'} \ell(\mathbf{z}, \mathbf{z}') \log(\hat{\ell}_\zeta(\mathbf{z}, \mathbf{z}')) + (1 - \ell(\mathbf{z}, \mathbf{z}')) \log(1 - \hat{\ell}_\zeta(\mathbf{z}, \mathbf{z}')) \tag{4}$$

### B.2 Task 2: Ranking ML configurations

Task 2 aims to comparatively assess two vectors of hyper-parameters $\theta$ and $\theta'$ of a fixed supervised learning algorithm $Alg$, referred to as configurations of $Alg$, depending on their performance on a dataset patch $\mathbf{z}$. For brevity and by abuse of language, the performance of a configuration $\theta$ on $\mathbf{z}$ is meant for the accuracy of the model learned from $\mathbf{z}$ using $Alg$ with configuration $\theta$, computed using a 3 fold cross validation.

**Algorithm 1**: Patch Identification

**Procedure** `Task_1(`$\mathcal{F}_\zeta$`, bench, N)`
  **input** : A meta-feature extractor $\mathcal{F}_\zeta$ in {Dida, Dataset2Vec, Deep Sets, DSS, Hand-crafted}, a benchmark *bench* in {Toy, UCI, OpenML}, and a number of iterations $N$
  **for** $i = 1, \ldots$ **do**
    $\mathbf{z}_1, \mathbf{z}_2, y \leftarrow$ `generate_patches(`*bench*`)`
    $m_1 \leftarrow \mathcal{F}_\zeta(\mathbf{z}_1)$
    $m_2 \leftarrow \mathcal{F}_\zeta(\mathbf{z}_2)$
    Compute loss (Equation 4), and update $\zeta$
  **end**

The considered ML algorithms are: Logistic regression (LR), SVM, k-Nearest Neighbours (k-NN), linear classifier learned with stochastic gradient descent (SGD). For each algorithm, a Task 2 problem is defined as follows (Algorithm 2). An example is made of a triplet $(\mathbf{z}, \theta, \theta')$, associated with a binary label $\ell(\mathbf{z}, \theta, \theta')$, set to 1 iff $\theta'$ yields better performance than $\theta$ on $\mathbf{z}$. Thus, the overall architecture consists of:

- a meta-feature extractor $\mathcal{F}_\zeta(\mathbf{z})$;
- a 2-layer FC network (depending on the considered *Alg* as they have different configuration spaces) with input vector $[\mathcal{F}_\zeta(\mathbf{z}); \theta; \theta']$

The overall is trained to minimize a cross-entropy loss (Equation 4).

**Algorithm 2**: Hyper-parameter Ranking

**Procedure** `Task_2(`$\mathcal{F}_\zeta$`, Alg)`
  **input** : A meta-feature extractor $\mathcal{F}_\zeta$ in {Dida, Dataset2Vec, Deep Sets, DSS, Hand-crafted}, an algorithm *Alg* in {SGD, SVM, LR, $k$-NN}.
  `NN` $\leftarrow$ 2-layer fully connected neural network
  **for** $i = 1, 2, \ldots$ **do**
    $\mathbf{z} \leftarrow$ `generate_patch(`*OpenML*`)`
    Sample $(\theta, \theta')$, two configurations of *Alg* (Table 5)
    Set binary target $y$ as 1 if `accuracy(`$\mathbf{z}, \theta$`) > accuracy(`$\mathbf{z}, \theta'$`)` else 0
    Compute loss (Equation 4) between y and `NN(`$[\mathcal{F}_\zeta(\mathbf{z}); \theta; \theta']$`)`
    Update $\zeta$ and `NN`
  **end**

In each epoch, a batch made of triplets $(\mathbf{z}, \theta, \theta')$ is built, with $\theta, \theta'$ uniformly drawn in the algorithm configuration space (Table 5) and $\mathbf{z}$ a patch of a dataset in the OpenML CC-2018 (Bischl et al., 2019), of size $n$ uniformly drawn in $[700; 900]$ and number of features $d$ in $[3; 10]$. Dida and all baselines are trained using Algorithm 2.

## C Baselines

Dida **details.** The functions $u$ and $v$ are represented by fully connected linear layers. They are configured as follows: $u$ =FC(8) and $v$=FC(1024), in both the first and second layers of Dida.

**Dataset2Vec details.** The publicly available implementation of Dataset2Vec [2] is implemented in TensorFlow, which is incompatible with our evaluation pipeline written in PyTorch. For this reason, we have included as baselines: (i) the reported accuracy from Jomaa et al. (2021), only available for Task 1; (ii) the computed accuracy from our own implementation of Dataset2Vec. Our Dataset2Vec implementation uses the same architecture as reported in Jomaa et al. (2021), Equation 4, namely

$$D : \mathbf{z} \in \mathrm{Z}_n(\mathbb{R}^d) \mapsto h\left(\frac{1}{d_X d_Y}\sum_{m=1}^{d_X}\sum_{t=1}^{d_Y} g\left(\frac{1}{n}\sum_{i=1}^{n} f(x_i[m], y_i[t])\right)\right) \tag{5}$$

where functions $f, g, h$ characterizing the architecture are chosen as depicted in the publicly available file $config.py$[3]. More precisely, $f, g$ are FC(128)-ReLU-ResFC(128, 128, 128)-FC(128) and $h$ is FC(128)-ReLU-FC(128)-ReLU where ResFC is a sequence of fully connected layer with skip connection. We provide our implementation of Dataset2Vec in the supplementary material.

**DSS layer details.**. We built our own implementation of invariant DSS layers, as follows. Linear invariant DSS layers (see Maron et al. (2020), Theorem 5, 3.) are of the form

$$L_{inv} : X \in \mathbb{R}^{n\times d} \mapsto L^H(\sum_{j=1}^{n} x_j) \in \mathbb{R}^K \tag{6}$$

where $L^H : \mathbb{R}^d \to \mathbb{R}^K$ is a linear $H$-invariant function. Our applicative setting requires that the implementation accommodates to varying input dimensions $d$ as well as permutation invariance, hence we consider the Deep Sets representation (see Zaheer et al. (2017), Theorem 7)

$$L^H : x = (x_1, \ldots, x_d) \in \mathbb{R}^d \mapsto \rho\left(\sum_{i=1}^{d} \varphi(x_i)\right) \in \mathbb{R}^K \tag{7}$$

where $\varphi : \mathbb{R} \to \mathbb{R}^{d+1}$ and $\rho : \mathbb{R}^{d+1} \to \mathbb{R}^K$ are modelled as (i) purely linear functions; (ii) FC networks, which extends the initial linear setting (6). In our case, $H = S_{d_X} \times S_{d_Y}$, hence, two invariant layers of the form (6-7) are combined to suit both feature- and label-invariance requirements. Both outputs are concatenated and followed by an FC network to form the DSS meta-features. The last experiments use DSS equivariant layers (see Maron et al. (2020), Theorem 1), which take the form

$$L_{eq} : X \in \mathbb{R}^{n\times d} \mapsto \left(L^1_{eq}(x_i) + L^2_{eq}(\sum_{j\neq i} x_j)\right)_{i\in[n]} \in \mathbb{R}^{n\times d} \tag{8}$$

where $L^1_{eq}$ and $L^2_{eq}$ are linear $H$-equivariant layers. Similarly, both feature- and label-equivariance requirements are handled via the Deep Sets representation of equivariant functions (see Zaheer et al. (2017), Lemma 3) and concatenated to be followed by an invariant layer, forming the DSS meta-features. All methods are allocated the same number of parameters to ensure fair comparison. We provide our implementation of the DSS layers in the supplementary material.

No-FInv-DSS **baseline (no invariance in feature permutation).**. This baseline aims at showcasing the empirical relevance of the invariance requirement in feature and label permutations, while retaining invariance in permutation with respect to the datasets. To this end, aggregation with respect to the examples is performed as exemplified in Zaheer et al. (2017), Theorem 2, namely

$$L : \mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_n) \in Z(\mathbb{R}^d) \mapsto \frac{1}{n}\sum_{i=1}^{n} g(\mathbf{z}_i) \in \mathbb{R}^K \tag{9}$$

[2] See https://github.com/hadijomaa/dataset2vec

[3] See https://github.com/hadijomaa/dataset2vec/blob/master/config.py

where $g : \mathbb{R}^d \to \mathbb{R}^K$ is an MLP with FC(128)-ReLU-FC(64)-ReLU-FC(32)-ReLU layers. To ensure label information is captured, the output is concatenated to the mean of labels $\bar{y} \stackrel{\text{def.}}{=} \frac{1}{n}\sum_{i=1}^{n} y_i$ and followed by and MLP with FC(1024)-ReLU-FC(700)-ReLU-FC(512) layers. The so-called No-FInv-DSS baseline defined as such, can be summed up as follows

$$\mathbf{z} \in \mathrm{Z}(\mathbb{R}^d) \mapsto \mathrm{MLP}([L(\mathbf{z}); \bar{y}]) \tag{10}$$

**Hand-crafted meta-features.** For the sake of reproducibility, the list of meta-features used in Section 3 is given in Table 4. Note that meta-features related to missing values and categorical features are omitted, as being irrelevant for the considered benchmarks. Hand-crafted meta-features are extracted using BYU `metalearn` library. In total, we extracted 43 meta-features.

## D Hyper-parameter spaces

In Task 2, the hyper-parameter configuration spaces of each algorithm are summarized in Table 5.

| | Parameter | Parameter values | Scale |
|---|---|---|---|
| **LR** | warm start | True, Fase | |
| | fit intercept | True, Fase | |
| | tol | [0.00001, 0.0001] | |
| | C | [1e-4, 1e4] | log |
| | solver | newton-cg, lbfgs, liblinear, sag, saga | |
| | max_iter | [5, 1000] | |
| **SVM** | kernel | linear, rbf, poly, sigmoid | |
| | C | [0.0001, 10000] | log |
| | shrinking | True, False | |
| | degree | [1, 5] | |
| | coef0 | [0, 10] | |
| | gamma | [0.0001, 8] | |
| | max_iter | [5, 1000] | |
| **KNN** | n_neighbors | [1, 100] | log |
| | p | [1, 2] | |
| | weights | uniform, distance | |
| **SGD** | alpha | [0.1, 0.0001] | log |
| | average | True, False | |
| | fit_intercept | True, False | |
| | learning rate | optimal, invscaling, constant | |
| | loss | hinge, log, modified_huber, squared_hinge, perceptron | |
| | penalty | l1, l2, elasticnet | |
| | tol | [1e-05, 0.1] | log |
| | eta0 | [1e-7, 0.1] | log |
| | power_t | [1e-05, 0.1] | log |
| | epsilon | [1e-05, 0.1] | log |
| | l1_ratio | [1e-05, 0.1] | log |

Table 5: Hyper-parameter configurations

| Meta-features | Mean | Min | Max |
|---|---|---|---|
| Quartile2ClassProbability | 0.500 | 0.75 | 0.25 |
| MinorityClassSize | 487.423 | 426.000 | 500.000 |
| Quartile3CardinalityOfNumericFeatures | 224.354 | 0.000 | 976.000 |
| RatioOfCategoricalFeatures | 0.347 | 0.000 | 1.000 |
| MeanCardinalityOfCategoricalFeatures | 0.907 | 0.000 | 2.000 |
| SkewCardinalityOfNumericFeatures | 0.148 | -2.475 | 3.684 |
| RatioOfMissingValues | 0.001 | 0.000 | 0.250 |
| MaxCardinalityOfNumericFeatures | 282.461 | 0.000 | 977.000 |
| Quartile2CardinalityOfNumericFeatures | 185.555 | 0.000 | 976.000 |
| KurtosisClassProbability | -2.025 | -3.000 | -2.000 |
| NumberOfNumericFeatures | 3.330 | 0.000 | 30.000 |
| NumberOfInstancesWithMissingValues | 2.800 | 0.000 | 1000.000 |
| MaxCardinalityOfCategoricalFeatures | 0.917 | 0.000 | 2.000 |
| Quartile1CardinalityOfCategoricalFeatures | 0.907 | 0.000 | 2.000 |
| MajorityClassSize | 512.577 | 500.000 | 574.000 |
| MinCardinalityOfCategoricalFeatures | 0.879 | 0.000 | 2.000 |
| Quartile2CardinalityOfCategoricalFeatures | 0.915 | 0.000 | 2.000 |
| NumberOfCategoricalFeatures | 1.854 | 0.000 | 27.000 |
| NumberOfFeatures | 5.184 | 4.000 | 30.000 |
| Dimensionality | 0.005 | 0.004 | 0.030 |
| SkewCardinalityOfCategoricalFeatures | -0.050 | -4.800 | 0.707 |
| KurtosisCardinalityOfCategoricalFeatures | -1.244 | -3.000 | 21.040 |
| StdevCardinalityOfNumericFeatures | 68.127 | 0.000 | 678.823 |
| StdevClassProbability | 0.018 | 0.000 | 0.105 |
| KurtosisCardinalityOfNumericFeatures | -1.060 | -3.000 | 12.988 |
| NumberOfInstances | 1000.000 | 1000.000 | 1000.000 |
| Quartile3CardinalityOfCategoricalFeatures | 0.916 | 0.000 | 2.000 |
| NumberOfMissingValues | 2.800 | 0.000 | 1000.000 |
| Quartile1ClassProbability | 0.494 | 0.463 | 0.500 |
| StdevCardinalityOfCategoricalFeatures | 0.018 | 0.000 | 0.707 |
| MeanClassProbability | 0.500 | 0.500 | 0.500 |
| NumberOfFeaturesWithMissingValues | 0.003 | 0.000 | 1.000 |
| MaxClassProbability | 0.513 | 0.500 | 0.574 |
| NumberOfClasses | 2.000 | 2.000 | 2.000 |
| MeanCardinalityOfNumericFeatures | 197.845 | 0.000 | 976.000 |
| SkewClassProbability | 0.000 | -0.000 | 0.000 |
| Quartile3ClassProbability | 0.506 | 0.500 | 0.537 |
| MinCardinalityOfNumericFeatures | 138.520 | 0.000 | 976.000 |
| MinClassProbability | 0.487 | 0.426 | 0.500 |
| RatioOfInstancesWithMissingValues | 0.003 | 0.000 | 1.000 |
| Quartile1CardinalityOfNumericFeatures | 160.748 | 0.000 | 976.000 |
| RatioOfNumericFeatures | 0.653 | 0.000 | 1.000 |
| RatioOfFeaturesWithMissingValues | 0.001 | 0.000 | 0.250 |

Table 4: Hand-crafted meta-features

# Theoritical results

## E Theoretical Analysis

This section investigates the properties of invariant-layer based neural architectures, and establishes their robustness w.r.t. bounded transformations of the involved distributions, and their approximation abilities w.r.t. the convergence in law. As already said, the discrete distribution case is considered for the sake of readability; the case of continuous distributions is detailed in Appendix F.

### E.1 Topology on Datasets

**Point clouds vs. distributions**.. The fact that datasets are preferably seen as probability distributions (as opposed to point clouds) is motivated as including many copies of a point in a dataset amounts to increasing its importance, which usually makes a difference in standard machine learning settings. Accordingly the topological framework used in the following is that of the convergence in law on distributions, with the distance among two datasets measured using the Wasserstein distance[4].

**Wasserstein distance**.. The standard 1-Wasserstein distance between two discrete probability distributions $\mathbf{z}, \mathbf{z}' \in \mathrm{Z}_n(\mathbb{R}^d) \times \mathrm{Z}_m(\mathbb{R}^d)$ is defined after Santambrogio (2015); Peyré and Cuturi (2019):

$$\mathrm{W}_1(\mathbf{z}, \mathbf{z}') \stackrel{\text{def.}}{=} \max_{f \in \mathrm{Lip}_1(\mathbb{R}^d)} \frac{1}{n} \sum_{i=1}^{n} f(z_i) - \frac{1}{m} \sum_{j=1}^{m} f(z'_j)$$

with $\mathrm{Lip}_1(\mathbb{R}^d)$ the space of scalar 1-Lipschitz functions on $\mathbb{R}^d$. The $G$-invariant 1-Wasserstein distance is defined to extend the above and account for the invariance under operators in $G$:

$$\overline{\mathrm{W}}_1(\mathbf{z}, \mathbf{z}') = \min_{\sigma \in G} \mathrm{W}_1(\sigma_\sharp \mathbf{z}, \mathbf{z}')$$

Accordingly, $\overline{\mathrm{W}}_1(\mathbf{z}, \mathbf{z}') = 0$ iff $\mathbf{z}$ and $\mathbf{z}'$ are equal in the sense of probability distributions up to sample and feature permutations (Appendix F).

**Lipschitz property**.. Let $\mathbf{z}^{(k)}$ be a sequence of distributions weakly converging toward $\mathbf{z}$ (noted $\mathbf{z}^{(k)} \rightharpoonup \mathbf{z}$). By construction, $\mathbf{z}^{(k)} \rightharpoonup \mathbf{z}$ iff $\mathrm{W}_1(\mathbf{z}^{(k)}, \mathbf{z}) \to 0$. Map $f$ from $\mathrm{Z}(\mathbb{R}^d)$ onto $\mathrm{Z}(\mathbb{R}^r)$ is said to be continuous iff for any sequence $\mathbf{z}^{(k)} \rightharpoonup \mathbf{z}$, then $f(\mathbf{z}^{(k)}) \rightharpoonup f(\mathbf{z})$. Map $f$ is said to be $C$-Lipschitz for $\overline{\mathrm{W}}_1$ iff

$$\forall \mathbf{z}, \mathbf{z}' \in \mathrm{Z}(\mathbb{R}^d), \quad \overline{\mathrm{W}}_1(f(\mathbf{z}), f(\mathbf{z}')) \leqslant C \overline{\mathrm{W}}_1(\mathbf{z}, \mathbf{z}'). \tag{11}$$

The $C$-Lipschitz property entails the continuity of $f$: if two input distributions are close in the permutation invariant 1-Wasserstein sense, their images by $f$ are close too.

### E.2 Lipschitzness results

Let us assume the interaction functional $\varphi$ to satisfy the Lipschitz property w.r.t. their first and second arguments ($\forall z \in \mathbb{R}^d, \varphi(z, \cdot)$ and $\varphi(\cdot, z)$ are $C_\varphi -$ Lipschitz.). Then invariant layer $f_\varphi$ also satisfy the Lipschitz property.

**Proposition 1**. *Invariant layer $f_\varphi$ of type (Eq. 1) is $(2rC_\varphi)$-Lipschitz in the sense of (Eq. 11).*

[4]In contrast, the distance among point clouds commonly relies on the Hausdorff distance among sets (see e.g., Qi et al. (2017)). This distance, that is standard for 2D and 3D data involved in graphics and vision domains, however faces some limitations in higher dimensional domains, e.g. due to max-pooling being a non-continuous operator w.r.t. the convergence in law topology.

A second result regards the case where two datasets $\mathbf{z}$ and $\mathbf{z}'$ are such that $\mathbf{z}'$ is the image of $\mathbf{z}$ through some diffeomorphism $\tau$ ($\mathbf{z} = (z_1, \ldots, z_n)$ and $\mathbf{z}' = \tau_\sharp \mathbf{z} = (\tau(z_1), \ldots, \tau(z_n))$. If $\tau$ is close to identity, then $f_\varphi(\tau_\sharp \mathbf{z})$ and $f_\varphi(\mathbf{z})$ are close too. More generally, if continuous transformations $\tau$ and $\xi$ respectively apply on the input and output space of $f_\varphi$, and are close to identity, then $\xi_\sharp f_\varphi(\tau_\sharp \mathbf{z})$ and $f_\varphi(\mathbf{z})$ are also close.

**Proposition 2.** *Let $\tau : \mathbb{R}^d \to \mathbb{R}^d$ and $\xi : \mathbb{R}^r \to \mathbb{R}^r$ be two Lipschitz maps with respectively Lipschitz constants $C_\tau$ and $C_\xi$. Then, $\forall \mathbf{z}, \mathbf{z}' \in Z(\Omega)$,*

$$\begin{aligned}
&\overline{\mathrm{W}}_1(\xi_\sharp f_\varphi(\tau_\sharp \mathbf{z}), f_\varphi(\mathbf{z})) \\
&\quad \leqslant \sup_{x \in f_\varphi(\tau(\Omega))} \|\xi(x) - x\|_2 + 2r \,\mathrm{Lip}(\varphi) \sup_{x \in \Omega} \|\tau(x) - x\|_2
\end{aligned}$$

*In addition, if $\tau$ is equivariant,*

$$\overline{\mathrm{W}}_1(\xi_\sharp f_\varphi(\tau_\sharp \mathbf{z}), \xi_\sharp f_\varphi(\tau_\sharp \mathbf{z}')) \leqslant 2r \, C_\varphi \, C_\tau \, C_\xi \overline{\mathrm{W}}_1(\mathbf{z}, \mathbf{z}')$$

Proofs: in Appendix G.

### E.3 Universal Approximation Results

Lastly, the universality of the proposed architecture is established, showing that the composition of an invariant layer (Eq. (1)) and a fully-connected layer is enough to yield the universal approximation property, over all functions defined on $Z(\mathbb{R}^d)$ with dimension $d$ less than some upper bound $D$.

**Theorem 1.** *Let $\mathcal{F} : Z(\Omega) \to \mathbb{R}$ be a $G$-invariant map on a compact $\Omega \subset \mathbb{R}^d$, continuous for the convergence in law. Then $\forall \varepsilon > 0$, there exists two continuous maps $\psi, \varphi$ such that*

$$\forall \mathbf{z} \in Z(\Omega), \quad |\mathcal{F}(\mathbf{z}) - \psi \circ f_\varphi(\mathbf{z})| < \varepsilon$$

*where $\varphi$ is $G$-invariant and independent of $\mathcal{F}$.*

*Sketch of proof* (details in Appendix H). 1. Let us define $\varphi = g \circ h$ where $h$ is the collection of $d_X$ elementary symmetric polynomials in the features and $d_Y$ elementary symmetric polynomials in the labels, which is invariant under $G$.
2. A discretization of $h(\Omega)$ on a grid is then considered, through $g$ that aims at collecting integrals over each cell of the discretization.
3. $\psi$ applies function $\mathcal{F}$ on this discretized measure; this requires $h$ to be bijective, and is achieved by $\tilde{h}$, through a projection on the quotient space $S_d/G$ and a restriction to its image compact $\Omega'$. To sum up, $f_\varphi$ defined as such computes an expectation which collects integrals over each cell of the grid to approximate measure $h_\sharp \mathbf{z}$ by a discrete counterpart $\widehat{h_\sharp \mathbf{z}}$. Hence $\psi$ applies $\mathcal{F}$ to $\tilde{h}_\sharp^{-1}(\widehat{h_\sharp \mathbf{z}})$. Continuity is obtained as follows: (i) proximity of $h_\sharp \mathbf{z}$ and $\widehat{h_\sharp \mathbf{z}}$ follows from Lemma 1 in Bie et al. (2019)) and gets tighter as the grid discretization step tends to 0; (ii) Map $\tilde{h}^{-1}$ is $1/d$-Hölder, after Theorem 1.3.1 from Rahman and Schmeisser (2002)); therefore Lemma 2 entails that $\overline{\mathrm{W}}_1(\mathbf{z}, \tilde{h}_\sharp^{-1} \widehat{h_\sharp \mathbf{z}})$ can be upper-bounded; (iii) since $\Omega$ is compact, by Banach-Alaoglu theorem, $\mathrm{Z}(\Omega)$ also is. Since $\mathcal{F}$ is continuous, it is thus uniformly weakly continuous: choosing a discretization step small enough ensures the result. □

After Theorem 1, any invariant continuous function defined on distributions with compact support can be approximated with arbitrary precision by an invariant neural network. This result holds for distributions with compact support in $\mathbb{R}^d$ for all $d \leq D$, with $D$ an upper bound on the dimension of the considered distribution supports. The proof (Appendix H) involves mainly three

steps: (i) an invariant layer $f_\varphi$ can be approximated by an invariant network; (ii) the universal approximation theorem Cybenko (1989); Leshno et al. (1993); (iii) uniform continuity is used to obtain uniform bounds. This result generalizes to the cases of finite distributions of any size $n$, and continuous distributions, the universality result established for fixed numbers of dimensions and points (Maron et al., 2020).

## F Extension to arbitrary distributions

**General notations**.. Let $X \in \mathcal{R}(\mathbb{R}^d)$ denote a random vector on $\mathbb{R}^d$ with $\alpha_X \in \mathcal{P}(\mathbb{R}^d)$ its law (a positive Radon measure with unit mass). By definition, its expectation denoted $\mathbb{E}(X)$ reads $\mathbb{E}(X) = \int_{\mathbb{R}^d} x \mathrm{d}\alpha_X(x) \in \mathbb{R}^d$, and for any continuous function $f : \mathbb{R}^d \to \mathbb{R}^r$, $\mathbb{E}(f(X)) = \int_{\mathbb{R}^d} f(x)\mathrm{d}\alpha_X(x)$. In the following, two random vectors $X$ and $X'$ with same law $\alpha_X$ are considered indistinguishable, noted $X' \sim X$. Letting $f : \mathbb{R}^d \mapsto \mathbb{R}^r$ denote a function on $\mathbb{R}^d$, the push-forward operator by $f$, noted $f_\sharp : \mathcal{P}(\mathbb{R}^d) \mapsto \mathcal{P}(\mathbb{R}^r)$ is defined as follows, for any $g$ continuous function from $\mathbb{R}^d$ to $\mathbb{R}^r$ ($g$ in $\mathcal{C}(\mathbb{R}^d; \mathbb{R}^r)$):

$$\forall g \in \mathcal{C}(\mathbb{R}^d; \mathbb{R}^r) \quad \int_{\mathbb{R}^r} g \mathrm{d}(f_\sharp \alpha) \stackrel{\text{def.}}{=} \int_{\mathbb{R}^d} g(f(x))\mathrm{d}\alpha(x)$$

Letting $\{x_i\}$ be a set of points in $\mathbb{R}^d$ with $w_i \geqslant 0$ such that $\sum_i w_i = 1$, the discrete measure $\alpha_X = \sum_i w_i \delta_{x_i}$ is the sum of the Dirac measures $\delta_{x_i}$ weighted by $w_i$.

**Invariances**.. In this paper, we consider functions on probability measures that are *invariant with respect to permutations of coordinates*. Therefore, denoting $S_d$ the $d$-sized permutation group, we consider measures over a symmetrized compact $\Omega \subset \mathbb{R}^d$ equipped with the following equivalence relation: for $\alpha, \beta \in \mathcal{P}(\Omega), \alpha \sim \beta \iff \exists \sigma \in S_d, \beta = \sigma_\sharp \alpha$, such that a measure and its permuted counterpart are indistinguishable in the corresponding quotient space, denoted alternatively $\mathcal{P}(\Omega)_{/\sim}$ or $\mathcal{R}(\Omega)_{/\sim}$. A function $\varphi : \Omega^n \to \mathbb{R}$ is said to be invariant (by permutations of coordinates) iff $\forall \sigma \in S_d, \varphi(x_1, \ldots, x_n) = \varphi(\sigma(x_1), \ldots, \sigma(x_n))$ (Definition 1).

**Tensorization**.. Letting $X$ and $Y$ respectively denote two random vectors on $\mathcal{R}(\mathbb{R}^d)$ and $\mathcal{R}(\mathbb{R}^p)$, the tensor product vector $X \otimes Y$ is defined as: $X \otimes Y \stackrel{\text{def.}}{=} (X', Y') \in \mathcal{R}(\mathbb{R}^d \times \mathbb{R}^p)$, where $X'$ and $Y'$ are independent and have the same law as $X$ and $Y$, i.e. $d(\alpha_{X\otimes Y})(x, y) = d\alpha_X(x) d\alpha_Y(y)$. In the finite case, for $\alpha_X = \frac{1}{n}\sum_i \delta_{x_i}$ and $\alpha_Y = \frac{1}{m}\sum_j \delta_{y_j}$, then $\alpha_{X\otimes Y} = \frac{1}{nm}\sum_{i,j} \delta_{x_i, y_j}$, weighted sum of Dirac measures on all pairs $(x_i, y_j)$. The $k-$fold tensorization of a random vector $X \sim \alpha_X$, with law $\alpha_X^{\otimes k}$, generalizes the above construction to the case of $k$ independent random variables with law $\alpha_X$. Tensorization will be used to define the law of datasets, and design universal architectures (Appendix H).

**Invariant layers**.. In the general case, a $G$-invariant layer $f_\varphi$ with invariant map $\varphi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^r$ such that $\varphi$ satisfies

$$\forall (x_1, x_2) \in (\mathbb{R}^d)^2, \forall \sigma \in G, \varphi(\sigma(x_1), \sigma(x_2)) = \varphi(x_1, x_2)$$

is defined as

$$f_\varphi : X \in \mathcal{R}(\mathbb{R}^d)_{/\sim} \mapsto \mathbb{E}_{X' \sim X}[\varphi(X, X')] \in \mathcal{R}(\mathbb{R}^r)_{/\sim}$$

where the expectation is taken over $X' \sim X$. Note that considering the couple $(X, X')$ of independent random vectors $X' \sim X$ amounts to consider the tensorized law $\alpha_X \otimes \alpha_X$.

*Remark* 1. Taking as input a discrete distribution $\alpha_X = \sum_{i=1}^n w_i \delta_{x_i}$, the invariant layer outputs another discrete distribution $\alpha_Y = \sum_{i=1}^n w_i \delta_{y_i}$ with $y_i = \sum_{j=1}^n w_j \varphi(x_i, x_j)$; each input point $x_i$ is mapped onto $y_i$ summarizing the pairwise interactions with $x_i$ after $\varphi$.

*Remark* 2. (Generalization to arbitrary invariance groups) The definition of invariant $\varphi$ can be generalized to arbitrary invariance groups operating on $\mathbb{R}^d$, in particular sub-groups of the permutation group $S_d$. After Maron et al. (2020) (Theorem 5), a simple and only way to design an invariant linear function is to consider $\varphi(z, z') = \psi(z + z')$ with $\psi$ being $G$-invariant. How to design invariant functions in the general non-linear case is left for further work.

*Remark* 3. Invariant layers can also be generalized to handle higher order interactions functionals, namely $f_\varphi(X) \overset{\text{def.}}{=} \mathbb{E}_{X_2,\ldots,X_N \sim X}[\varphi(X, X_2, \ldots, X_N)]$, which amounts to consider, in the discrete case, $N$-uple of inputs points $(x_{j_1}, \ldots, x_{j_N})$.

## G Proofs on Regularity

**Wasserstein distance**.. The regularity of the involved functionals is measured w.r.t. the 1-Wasserstein distance between two probability distributions $(\alpha, \beta) \in \mathcal{P}(\mathbb{R}^d)$

$$\begin{aligned} \mathrm{W}_1(\alpha, \beta) &\overset{\text{def.}}{=} \min_{\pi_1=\alpha, \pi_2=\beta} \int_{\mathbb{R}^d \times \mathbb{R}^d} \|x - y\| d\pi(x, y) \\ &\overset{\text{def.}}{=} \min_{X \sim \alpha, Y \sim \beta} \mathbb{E}(\|X - Y\|) \end{aligned}$$

where the minimum is taken over measures on $\mathbb{R}^d \times \mathbb{R}^d$ with marginals $\alpha, \beta \in \mathcal{P}(\mathbb{R}^d)$. $\mathrm{W}_1$ is known to be a norm Santambrogio (2015), that can be conveniently computed using

$$\mathrm{W}_1(\alpha, \beta) = \mathrm{W}_1(\alpha - \beta) = \max_{\mathrm{Lip}(g) \leqslant 1} \int_{\mathbb{R}^d} g(\alpha - \beta),$$

where $\mathrm{Lip}(g)$ is the Lipschitz constant of $g : \mathbb{R}^d \to \mathbb{R}$ with respect to the Euclidean norm (unless otherwise stated). For simplicity and by abuse of notations, $\mathrm{W}_1(X, Y)$ is used instead of $\mathrm{W}_1(\alpha, \beta)$ when $X \sim \alpha$ and $Y \sim \beta$. The convergence in law denoted $\rightharpoonup$ is equivalent to the convergence in Wasserstein distance in the sense that $X_k \rightharpoonup X$ is equivalent to $\mathrm{W}_1(X_k, X) \to 0$.

**Permutation-invariant Wasserstein distance**.. The Wasserstein distance is quotiented according to the permutation invariance equivalence classes: for $\alpha, \beta \in \mathcal{P}(\mathbb{R}^d)$

$$\begin{aligned} \overline{\mathrm{W}}_1(\alpha, \beta) &\overset{\text{def.}}{=} \min_{\sigma \in S_d} \mathrm{W}_1(\sigma_\sharp \alpha, \beta) \\ &= \min_{\sigma \in S_d} \max_{\mathrm{Lip}(g) \leqslant 1} \int_{\mathbb{R}^d} g \circ \sigma d\alpha - \int_{\mathbb{R}^d} g d\beta \end{aligned}$$

such that $\overline{\mathrm{W}}_1(\alpha, \beta) = 0 \iff \alpha \sim \beta$. $\overline{\mathrm{W}}_1$ defines a norm on $\mathcal{P}(\mathbb{R}^d)_{/\sim}$.

**Lipschitz property**.. A map $f : \mathcal{R}(\mathbb{R}^d) \to \mathcal{R}(\mathbb{R}^r)$ is continuous for the convergence in law (aka the weak* of measures) if for any sequence $X_k \rightharpoonup X$, then $f(X_k) \rightharpoonup f(X)$. Such a map is furthermore said to be $C$-Lipschitz for the permutation invariant 1-Wasserstein distance if

$$\forall (X, Y) \in (\mathcal{R}(\mathbb{R}^d)_{/\sim})^2, \ \overline{\mathrm{W}}_1(f(X), f(Y)) \leqslant C \overline{\mathrm{W}}_1(X, Y). \tag{12}$$

Lipschitz properties enable us to analyze robustness to input perturbations, since it ensures that if the input distributions of random vectors are close in the permutation invariant Wasserstein sense, the corresponding output laws are close, too.

**Proofs of Section E.2..**

*Proof.* (Proposition 1). For $\alpha, \beta \in \mathcal{P}(\mathbb{R}^d)$, Proposition 1 from Bie et al. (2019) yields $\mathrm{W}_1(f_\varphi(\alpha), f_\varphi(\beta)) \leqslant 2r\,\mathrm{Lip}(\varphi)\,\mathrm{W}_1(\alpha, \beta)$, hence, for $\sigma \in G$,

$$\begin{aligned}\mathrm{W}_1(\sigma_\sharp f_\varphi(\alpha), f_\varphi(\beta)) &\leqslant \mathrm{W}_1(\sigma_\sharp f_\varphi(\alpha), f_\varphi(\alpha)) \\ &\quad + \mathrm{W}_1(f_\varphi(\alpha), f_\varphi(\beta)) \\ &\leqslant \mathrm{W}_1(\sigma_\sharp f_\varphi(\alpha), f_\varphi(\alpha)) \\ &\quad + 2r\,\mathrm{Lip}(\varphi)\,\mathrm{W}_1(\alpha, \beta)\end{aligned}$$

hence, taking the infimum over $\sigma$ yields

$$\begin{aligned}\overline{\mathrm{W}}_1(f_\varphi(\alpha), f_\varphi(\beta)) &\leqslant \overline{\mathrm{W}}_1(\ \ f_\varphi(\alpha), f_\varphi(\alpha)) \\ &\quad + 2r\,\mathrm{Lip}(\varphi)\,\mathrm{W}_1(\alpha, \beta) \\ &\leqslant 2r\,\mathrm{Lip}(\varphi)\,\mathrm{W}_1(\alpha, \beta)\end{aligned}$$

Since $f_\varphi$ is invariant, for $\sigma \in G$, $f_\varphi(\mathbf{z}) = f_\varphi(\sigma_\sharp \mathbf{z})$,

$$\overline{\mathrm{W}}_1(f_\varphi(\alpha), f_\varphi(\beta)) \leqslant 2r\,\mathrm{Lip}(\varphi)\,\mathrm{W}_1(\sigma_\sharp\alpha, \beta)$$

Taking the infimum over $\sigma$ yields the result. □

*Proof.* (Proposition 2). To upper bound $\overline{\mathrm{W}}_1(\xi_\sharp f_\varphi(\tau_\sharp\alpha), f_\varphi(\alpha))$ for $\alpha \in \mathcal{P}(\mathbb{R}^d)$, we proceed as follows, using proposition 3 from Bie et al. (2019) and proposition 1:

$$\begin{aligned}\mathrm{W}_1(\xi_\sharp f_\varphi(\tau_\sharp\alpha_\varphi(\alpha)), f_\varphi(\alpha)) &\leqslant \mathrm{W}_1(\xi_\sharp f_\varphi(\tau_\sharp\alpha), f_\varphi(\tau_\sharp\alpha)) \\ &\quad + \mathrm{W}_1(f_\varphi(\tau_\sharp\alpha), f_\varphi(\alpha)) \\ &\leqslant \|\xi - id\|_{L^1(f_\varphi(\tau_\sharp\alpha))} \\ &\quad + \mathrm{Lip}(f_\varphi)\,\mathrm{W}_1(\tau_\sharp\alpha, \alpha) \\ &\leqslant \sup_{y \in f_\varphi(\tau(\Omega))} \|\xi(y) - y\|_2 \\ &\quad + 2r\,\mathrm{Lip}(\varphi) \sup_{x\in\Omega} \|\tau(x) - x\|_2\end{aligned}$$

For $\sigma \in G$, we get

$$\begin{aligned}\mathrm{W}_1(\sigma_\sharp\xi_\sharp f_\varphi(\tau_\sharp\alpha), f_\varphi(\alpha)) &\leqslant \mathrm{W}_1(\sigma_\sharp\xi_\sharp f_\varphi(\tau_\sharp\alpha), \xi_\sharp f_\varphi(\tau_\sharp\alpha)) \\ &\quad + \mathrm{W}_1(\xi_\sharp f_\varphi(\tau_\sharp\alpha), f_\varphi(\alpha))\end{aligned}$$

Taking the infimum over $\sigma$ yields

$$\begin{aligned}\overline{\mathrm{W}}_1(\xi_\sharp f_\varphi(\tau_\sharp\alpha), f_\varphi(\alpha)) &\leqslant \mathrm{W}_1(\xi_\sharp f_\varphi(\tau_\sharp\alpha), f_\varphi(\alpha)) \\ &\leqslant \sup_{y \in f_\varphi(\tau(\Omega))} \|\xi(y) - y\|_2 \\ &\quad + 2rC(\varphi) \sup_{x\in\Omega} \|\tau(x) - x\|_2\end{aligned}$$

Similarly, for $\alpha, \beta \in (\mathcal{P}(\mathbb{R}^d))^2$,

$$\begin{aligned}\mathrm{W}_1(\xi_\sharp f_\varphi(\tau_\sharp\alpha), \xi_\sharp f_\varphi(\tau_\sharp\beta)) &\leqslant \mathrm{Lip}(\xi)\,\mathrm{W}_1(f_\varphi(\tau_\sharp\alpha), f_\varphi(\tau_\sharp\beta)) \\ &\leqslant \mathrm{Lip}(\xi) \quad \mathrm{Lip}(f_\varphi)\,\mathrm{W}_1(\tau_\sharp\alpha, \tau_\sharp\beta) \\ &\leqslant 2r\,\mathrm{Lip}(\varphi)\ \mathrm{Lip}(\xi)\,\mathrm{Lip}(\tau)\,\mathrm{W}_1(\alpha, \beta)\end{aligned}$$

hence, for $\sigma \in G$,

$$\begin{aligned}\mathrm{W}_1(\sigma_\sharp \xi_\sharp f_\varphi(\tau_\sharp \alpha), \xi_\sharp f_\varphi(\tau_\sharp \beta)) \leqslant\ & \mathrm{W}_1(\sigma_\sharp \xi_\sharp f_\varphi(\tau_\sharp \alpha), \xi_\sharp f_\varphi(\tau_\sharp \alpha)) \\ & + \mathrm{W}_1(\xi_\sharp f_\varphi(\tau_\sharp \alpha), \xi_\sharp f_\varphi(\tau_\sharp \beta))\end{aligned}$$

and taking the infimum over $\sigma$ yields

$$\begin{aligned}\overline{\mathrm{W}}_1(\xi_\sharp f_\varphi(\tau_\sharp \alpha), \xi_\sharp f_\varphi(\tau_\sharp \beta)) &\leqslant \mathrm{W}_1(\xi_\sharp f_\varphi(\tau_\sharp \alpha), \xi_\sharp f_\varphi(\tau_\sharp \beta)) \\ &\leqslant 2r \operatorname{Lip}(\varphi) \operatorname{Lip}(\xi) \operatorname{Lip}(\tau) \mathrm{W}_1(\alpha, \beta)\end{aligned}$$

Since $\tau$ is equivariant: namely, for $\alpha \in \mathcal{P}(\mathbb{R}^d)$, $\sigma \in G$, $\tau_\sharp(\sigma_\sharp \alpha) = \sigma_\sharp(\tau_\sharp \alpha)$, hence, since $f_\varphi$ is invariant, $f_\varphi(\tau_\sharp(\sigma_\sharp \alpha)) = f_\varphi(\sigma_\sharp(\tau_\sharp \alpha)) = f_\varphi(\tau_\sharp \alpha)$, hence for $\sigma \in G$,

$$\begin{aligned}&\overline{\mathrm{W}}_1(\xi_\sharp f_\varphi(\tau_\sharp \alpha), \xi_\sharp f_\varphi(\tau_\sharp \beta)) \\ &\qquad \leqslant 2r \operatorname{Lip}(\varphi) \operatorname{Lip}(\xi) \operatorname{Lip}(\tau) \mathrm{W}_1(\sigma_\sharp \alpha, \beta)\end{aligned}$$

Taking the infimum over $\sigma$ yields the result. □

## H Proofs on Universality

**Detailed proof of Theorem 1.**. This paragraph details the result in the case of $S_d-$invariance, while the next one focuses on invariances w.r.t. products of permutations. Before providing a proof of Theorem 1 we first state two useful lemmas. Lemma 1 is mentioned for completeness, referring the reader to Bie et al. (2019), Lemma 1 for a proof.

**Lemma 1.** *Let $\left(S_j\right)_{j=1}^N$ be a partition of a domain including $\Omega$ ($S_j \subset \mathbb{R}^d$) and let $x_j \in S_j$. Let $(\varphi_j)_{j=1}^N$ a set of bounded functions $\varphi_j : \Omega \to \mathbb{R}$ supported on $S_j$, such that $\sum_j \varphi_j = 1$ on $\Omega$. For $\alpha \in \mathcal{P}(\Omega)$, we denote $\hat{\alpha}_N \overset{\text{def.}}{=} \sum_{j=1}^N \alpha_j \delta_{x_j}$ with $\alpha_j \overset{\text{def.}}{=} \int_{S_j} \varphi_j d\alpha$. One has, denoting $\Delta_j \overset{\text{def.}}{=} \max_{x \in S_j} \|x_j - x\|$,*

$$\mathrm{W}_1(\hat{\alpha}_N, \alpha) \leqslant \max_{1 \leqslant j \leqslant N} \Delta_j.$$

**Lemma 2.** *Let $f : \mathbb{R}^d \to \mathbb{R}^q$ a $1/p$-Hölder continuous function ($p \geqslant 1$), then there exists a constant $C > 0$ such that for all $\alpha, \beta \in \mathcal{P}(\mathbb{R}^d)$, $\mathrm{W}_1(f_\sharp \alpha, f_\sharp \beta) \leqslant C \mathrm{W}_1(\alpha, \beta)^{1/p}$.*

*Proof.* For any transport map $\pi$ with marginals $\alpha$ and $\beta$, $1/p$-Hölderness of $f$ with constant $C$ yields $\int ||f(x) - f(y)||_2 \mathrm{d}\pi(x, y) \leqslant C \int ||x - y||_2^{1/p} \mathrm{d}\pi(x, y) \leqslant C \left( \int ||x - y||_2 \mathrm{d}\pi(x, y) \right)^{1/p}$ using Jensen's inequality ($p \leqslant 1$). Taking the infimum over $\pi$ yields $\mathrm{W}_1(f_\sharp \alpha, f_\sharp \beta) \leqslant C \mathrm{W}_1(\alpha, \beta)^{1/p}$. □

Now we are ready to dive into the proof. Let $\alpha \in \mathcal{P}(\mathbb{R}^d)$. We consider:

- $h : x = (x_1, \ldots, x_d) \in \mathbb{R}^d \mapsto \left( \sum_{1 \leqslant j_1 < \ldots < j_i \leqslant d} x_{j_1} \cdot \ldots \cdot x_{j_i} \right)_{i=1 \ldots d} \in \mathbb{R}^d$ the collection of $d$ elementary symmetric polynomials; $h$ does not lead to a loss in information, in the sense that it generates the ring of $S_d$-invariant polynomials (see for instance Cox et al. (2018), chapter 7, theorem 3) while preserving the classes (see the proof of Lemma 2, appendix D from Maron et al. (2020));
- $h$ is obviously not injective, so we consider $\pi : \mathbb{R}^d \to \mathbb{R}^d / S_d$ the projection onto $\mathbb{R}^d / S_d$: $h = \tilde{h} \circ \pi$ such that $\tilde{h}$ is bijective from $\pi(\Omega)$ to its image $\Omega'$, compact of $\mathbb{R}^d$; $\tilde{h}$ and $\tilde{h}^{-1}$ are continuous;
- Let $(\varphi_i)_{i=1 \ldots N}$ the piecewise affine P1 finite element basis, which are hat functions on a discretization $(S_i)_{i=1 \ldots N}$ of $\Omega' \subset \mathbb{R}^d$, with centers of cells $(y_i)_{i=1 \ldots N}$. We then define $g : x \in \mathbb{R}^d \mapsto (\varphi_1(x), \ldots, \varphi_N(x)) \in \mathbb{R}^N$;

- $f : (\alpha_1, \dots, \alpha_N) \in \mathbb{R}^N \mapsto \mathcal{F}\left(\sum_{i=1}^N \alpha_i \delta_{\tilde{h}^{-1}(y_i)}\right) \in \mathbb{R}$.

We approximate $\mathcal{F}$ using the following steps:

- Lemma 1 (see Lemma 1 from Bie et al. (2019)) yields that $h_\sharp \alpha$ and $\widehat{h_\sharp \alpha} = \sum_{i=1}^N \alpha_i \delta_{y_i}$ are close: $\mathrm{W}_1(h_\sharp \alpha, \widehat{h_\sharp \alpha}) \leqslant \sqrt{d}/N^{1/d}$;
- The map $\tilde{h}^{-1}$ is regular enough ($1/d$-Hölder) such that according to Lemma 2, there exists a constant $C > 0$ such that

$$\begin{aligned}\mathrm{W}_1(\tilde{h}_\sharp^{-1}(h_\sharp \alpha), \tilde{h}_\sharp^{-1}\widehat{h_\sharp \alpha}) &\leqslant C\, \mathrm{W}_1(h_\sharp \alpha, \widehat{h_\sharp \alpha})^{1/d} \\ &\leqslant C d^{1/2d}/N^{1/d^2}\end{aligned}$$

Hence

$$\begin{aligned}\overline{\mathrm{W}}_1(\alpha, \tilde{h}_\sharp^{-1}\widehat{h_\sharp \alpha}) &\overset{\text{def.}}{=} \inf_{\sigma \in S_d} \mathrm{W}_1(\sigma_\sharp \alpha, \tilde{h}_\sharp^{-1}\widehat{h_\sharp \alpha}) \\ &\leqslant C d^{1/2d}/N^{1/d^2}.\end{aligned}$$

Note that $h$ maps the roots of polynomial $\prod_{i=1}^d (X - x^{(i)})$ to its coefficients (up to signs). Theorem 1.3.1 from Rahman and Schmeisser (2002) yields continuity and $1/d$-Hölderness of the reverse map. Hence $\tilde{h}^{-1}$ is $1/d$-Hölder.

- Since $\Omega$ is compact, by Banach-Alaoglu theorem, we obtain that $\mathcal{P}(\Omega)$ is weakly-* compact, hence $\mathcal{P}(\Omega)_{/\sim}$ also is. Since $\mathcal{F}$ is continuous, it is thus uniformly weak-* continuous: for any $\varepsilon > 0$, there exists $\delta > 0$ such that $\overline{\mathrm{W}}_1(\alpha, \tilde{h}_\sharp^{-1}\widehat{h_\sharp \alpha}) \leqslant \delta$ implies $|\mathcal{F}(\alpha) - \mathcal{F}(\tilde{h}_\sharp^{-1}\widehat{h_\sharp \alpha})| < \varepsilon$. Choosing $N$ large enough such that $C d^{1/2d}/N^{1/d^2} \leqslant \delta$ therefore ensures that $|\mathcal{F}(\alpha) - \mathcal{F}(\tilde{h}_\sharp^{-1}\widehat{h_\sharp \alpha})| < \varepsilon$.

**Extension of Theorem 1 to products of permutation groups**..

**Corollary 1**. *Let $\mathcal{F} : \mathcal{P}(\Omega)_{/\sim} \to \mathbb{R}$ a continuous $S_{d_1} \times \ldots \times S_{d_n}$-invariant map ($\sum_i d_i = d$), where $\Omega$ is a symmetrized compact over $\mathbb{R}^d$. Then $\forall \varepsilon > 0$, there exists three continuous maps $f, g, h$ such that*

$$\forall \alpha \in \mathcal{M}_+^1(\Omega)_{/\sim}, |\mathcal{F}(\alpha) - f \circ \mathbb{E} \circ g(h_\sharp \alpha)| < \varepsilon$$

*where $h$ is invariant; $g, h$ are independent of $\mathcal{F}$.*

*Proof.* We provide a proof in the case $G = S_d \times S_p$, which naturally extends to any product group $G = S_{d_1} \times \ldots \times S_{d_n}$. We trade $h$ for the collection of elementary symmetric polynomials in the first $d$ variables; and in the last $p$ variables:

$$\begin{aligned}h : (x_1, \dots, x_d, y_1, \dots, y_p) &\in \mathbb{R}^{d+p} \\ \mapsto ([\sum_{j_1 < \ldots < j_i} x_{j_1} \dots x_{j_i}]_{i=1}^d; [\sum_{j_1 < \ldots < j_i} y_{j_1} \dots y_{j_i}]_{i=1}^p) &\\ \in \mathbb{R}^{d+p}&\end{aligned}$$

up to normalizing constants (see Lemma 4). Step 1 (in Lemma 3) consists in showing that $h$ does not lead to a loss of information, in the sense that it generates the ring of $S_d \times S_p-$invariant polynomials. In step 2 (in Lemma 4), we show that $\tilde{h}^{-1}$ is $1/\max(d, p)-$Hölder. Combined with the proof of Theorem 1, this amounts to showing that the concatenation of Hölder functions (up to normalizing constants) is Hölder. With these ingredients, the sketch of the previous proof yields the result. □

**Lemma 3**. *Let the collection of symmetric invariant polynomials* $[P_i(X_1, \dots, X_d)]_{i=1}^d \overset{\text{def.}}{=} [\sum_{j_1<\dots<j_i} X_{j_1} \dots X_{j_i}]_{i=1}^d$ *and* $[Q_i(Y_1, \dots, Y_p)]_{i=1}^p = [\sum_{j_1<\dots<j_i} Y_{j_1} \dots Y_{j_i}]_{i=1}^p$. *The* $d+p-$*sized family* $(P_1, \dots, P_d, Q_1, \dots, Q_p)$ *generates the ring of* $S_d \times S_p-$*invariant polynomials.*

*Proof.* The result comes from the fact the fundamental theorem of symmetric polynomials (see Cox et al. (2018) chapter 7, theorem 3) does not depend on the base field. Every $S_d \times S_p-$invariant polynomial $P(X_1, \dots, X_d, Y_1, \dots, Y_p)$ is also $S_d \times I_p-$invariant with coefficients in $\mathbb{R}[Y_1, \dots, Y_p]$, hence it can be written $P = R(Y_1, \dots, Y_p)(P_1, \dots, P_d)$. It is then also $S_p-$invariant with coefficients in $\mathbb{R}[P_1, \dots, P_d]$, hence it can be written $P = S(Q_1, \dots, Q_p)(P_1, \dots, P_d) \in \mathbb{R}[P_1, \dots, P_d, Q_1, \dots, Q_p]$. □

**Lemma 4**. *Let* $h : (x, y) \in \Omega \subset \mathbb{R}^{d+p} \mapsto (f(x)/C_1, g(y)/C_2) \in \mathbb{R}^{d+p}$ *where* $\Omega$ *is compact,* $f : \mathbb{R}^d \to \mathbb{R}^d$ *is* $1/d-$*Hölder with constant* $C_1$ *and* $g : \mathbb{R}^p \to \mathbb{R}^p$ *is* $1/p-$*Hölder with constant* $C_2$. *Then* $h$ *is* $1/\max(d,p)-$*Hölder.*

*Proof.* Without loss of generality, we consider $d > p$ so that $\max(d,p) = d$, and $f, g$ normalized (f.i. $\forall x, x_0 \in (\mathbb{R}^d)^2, \|f(x) - f(x_0)\|_1 \leqslant \|x - x_0\|_1^{1/d}$). For $(x, y), (x_0, y_0) \in \Omega^2$, $\|h(x,y) - h(x_0,y_0)\|_1 \leqslant \|f(x) - f(x_0)\|_1 + \|g(y) - g(y_0)\|_1 \leqslant \|x - x_0\|_1^{1/d} + \|y - y_0\|_1^{1/p}$ since both $f, g$ are Hölder. We denote $D$ the diameter of $\Omega$, such that both $\|x - x_0\|_1/D \leqslant 1$ and $\|y - y_0\|_1/D \leqslant 1$ hold. Therefore $\|h(x,y) - h(x_0,y_0)\|_1 \leqslant D^{1/d} \left(\frac{\|x-x_0\|_1}{D}\right)^{1/d} + D^{1/p} \left(\frac{\|y-y_0\|_1}{D}\right)^{1/p} \leqslant 2^{1-1/d} D^{1/p-1/d} \|(x,y) - (x_0,y_0)\|_1^{1/d}$ using Jensen's inequality, hence the result. □

In the next two paragraphs, we focus the case of $S_d-$invariant functions for the sake of clarity, without loss of generality. Indeed, the same technique applies to $G-$invariant functions as $h$ in that case has the same structure: its first $d_X$ components are $S_{d_X}-$invariant functions of the first $d_X$ variables and its last $d_Y$ components are $S_{d_Y}-$invariant functions of the last variables.

**Extension of Theorem 1 to distributions on spaces of varying dimension.**.

**Corollary 2**. *Let* $I = [0;1]$ *and, for* $k \in [1; d_m], \mathcal{F}_k : \mathcal{P}(I^k) \to \mathbb{R}$ *continuous and* $S_k-$*invariant. Suppose* $(\mathcal{F}_k)_{k=1\dots d_m-1}$ *are restrictions of* $\mathcal{F}_{d_m}$, *namely,* $\forall \alpha_k \in \mathcal{P}(I^k), \mathcal{F}_k(\alpha_k) = \mathcal{F}_{d_m}(\alpha_k \otimes \delta_0^{\otimes d_m - k})$. *Then functions* $f$ *and* $g$ *from Theorem 1 are uniform: there exists* $f, g$ *continuous,* $h_1, \dots, h_{d_m}$ *continuous invariant such that*

$$\forall k = 1 \dots d_m, \forall \alpha_k \in \mathcal{P}(I^k), |\mathcal{F}_k(\alpha_k) - f \circ \mathbb{E} \circ g(h_{k\sharp}\alpha_k)| < \varepsilon.$$

*Proof.* Theorem 1 yields continuous $f, g$ and a continuous invariant $h_{d_m}$ such that $\forall \alpha \in \mathcal{P}(I^{d_m}), |\mathcal{F}_{d_m} - f \circ \mathbb{E} \circ g(h_{d_m\sharp}\alpha)| < \varepsilon$. For $k = 1 \dots d_m - 1$, we denote $h_k : (x_1, \dots, x_k) \in \mathbb{R}^k \mapsto ((\sum_{1 \leqslant j_1 < \dots < j_i \leqslant k} x^{(j_1)} \cdot \dots \cdot x^{(j_i)})_{i=1\dots k}, 0 \dots, 0) \in \mathbb{R}^{d_m}$. With the hypothesis, for $k = 1 \dots d_m - 1$, $\alpha_k \in \mathcal{P}(I^k)$, the fact that $h_{k\sharp}(\alpha_k) = h_{d_m\sharp}(\alpha_k \otimes \delta_0^{\otimes d_m - k})$ yields the result. □

**Approximation by invariant neural networks.**. Based on theorem 1, $\mathcal{F}$ is uniformly close to $f \circ \mathbb{E} \circ g \circ h$:

- We approximate $f$ by a neural network $f_\theta : x \in \mathbb{R}^N \mapsto C_1 \lambda(A_1 x + b_1) \in \mathbb{R}$, where $p_1$ is an integer, $A_1 \in \mathbb{R}^{p_1 \times N}, C_1 \in \mathbb{R}^{1 \times p_1}$ are weights, $b_1 \in \mathbb{R}^{p_1}$ is a bias and $\lambda$ is a non-linearity.
- Since each component $\varphi_j$ of $\varphi = g \circ h$ is permutation-invariant, it has the representation $\varphi_j : x = (x_1, \dots, x_d) \in \mathbb{R}^d \mapsto \rho_j \left(\sum_{i=1}^d u(x_i)\right)$ Zaheer et al. (2017) (which is a special case of our layers with a base function only depending on its first argument, see Section 2.1), $\rho_j : \mathbb{R}^{d+1} \to \mathbb{R}$, and $u : \mathbb{R} \to \mathbb{R}^{d+1}$ independent of $j$ (see Zaheer et al. (2017), theorem 7).

- We can approximate $\rho_j$ and $u$ by neural networks $\rho_{j,\theta} : x \in \mathbb{R}^{d+1} \mapsto C_{2,j}\lambda(A_{2,j}x + b_{2,j}) \in \mathbb{R}$ and $u_\theta : x \in \mathbb{R}^d \mapsto C_3\lambda(A_3 x + b_3) \in \mathbb{R}^{d+1}$, where $p_{2,j}, p_3$ are integers, $A_{2,j} \in \mathbb{R}^{p_{2,j}\times(d+1)}, C_{2,j} \in \mathbb{R}^{1\times p_{2,j}}, A_3 \in \mathbb{R}^{p_3\times 1}, C_3 \in \mathbb{R}^{(d+1)\times p_3}$ are weights and $b_{2,j} \in \mathbb{R}^{p_{2,j}}, b_3 \in \mathbb{R}^{p_3}$ are biases, and denote $\varphi_\theta(x) = (\varphi_{j,\theta}(x))_j \overset{\text{def.}}{=} (\rho_{j,\theta}(\sum_{i=1}^d u_\theta(x_i)))_j$.

Indeed, we upper-bound the difference of interest $|\mathcal{F}(\alpha) - f_\theta\left(\mathbb{E}_{X\sim\alpha}\left(\varphi_\theta(X)\right)\right)|$ by triangular inequality by the sum of three terms:

- $|\mathcal{F}(\alpha) - f\left(\mathbb{E}_{X\sim\alpha}\left(\varphi(X)\right)\right)|$
- $|f\left(\mathbb{E}_{X\sim\alpha}\left(\varphi(X)\right)\right) - f_\theta\left(\mathbb{E}_{X\sim\alpha}\left(\varphi(X)\right)\right)|$
- $|f_\theta\left(\mathbb{E}_{X\sim\alpha}\left(\varphi(X)\right)\right) - f_\theta\left(\mathbb{E}_{X\sim\alpha}\left(\varphi_\theta(X)\right)\right)|$

and bound each term by $\frac{\varepsilon}{3}$, which yields the result. The bound on the first term directly comes from theorem 1 and yields a constant $N$ which depends on $\varepsilon$. The bound on the second term is a direct application of the universal approximation theorem (UAT) (Cybenko, 1989; Leshno et al., 1993). Indeed, since $\alpha$ is a probability measure, input values of $f$ lie in a compact subset of $\mathbb{R}^N$: $\| \int_\Omega g \circ h(x) \mathrm{d}\alpha \|_\infty \leqslant \max_{x\in\Omega} \max_i |g_i \circ h(x)|$, hence the theorem is applicable as long as $\lambda$ is a nonconstant, bounded and continuous activation function. Let us focus on the third term. Uniform continuity of $f_\theta$ yields the existence of $\delta > 0$ s.t. $||u - v||_1 < \delta$ implies $|f_\theta(u) - f_\theta(v)| < \frac{\varepsilon}{3}$. Let us apply the UAT: each component $\varphi_j$ of $h$ can be approximated by a neural network $\varphi_{j,\theta}$. Therefore:

$$
\begin{aligned}
\|\mathbb{E}_{X\sim\alpha}\left(\varphi(X) - \varphi_\theta(X)\right)\|_1 &\leqslant \mathbb{E}_{X\sim\alpha}\|\varphi(X) - \varphi_\theta(X)\|_1 \\
&\leqslant \sum_{j=1}^N \int_\Omega |\varphi_j(x) - \varphi_{j,\theta}(x)| \mathrm{d}\alpha(x) \\
&\leqslant \sum_{j=1}^N \int_\Omega |\varphi_j(x) - \rho_{j,\theta}(\sum_{i=1}^d u(x_i))| \mathrm{d}\alpha(x) \\
&+ \sum_{j=1}^N \int_\Omega |\rho_{j,\theta}(\sum_{i=1}^d u(x_i)) - \rho_{j,\theta}(\sum_{i=1}^d u_\theta(x_i))| \mathrm{d}\alpha(x) \\
&\leqslant N\frac{\delta}{2N} + N\frac{\delta}{2N} = \delta
\end{aligned}
$$

using the triangular inequality and the fact that $\alpha$ is a probability measure. The first term is small by UAT on $\rho_j$ while the second also is, by UAT on $u$ and uniform continuity of $\rho_{j,\theta}$. Therefore, by uniform continuity of $f_\theta$, we can conclude.

**Universality of tensorization**.. This complementary theorem provides insight into the benefits of tensorization for approximating invariant regression functionals, as long as the test function is invariant.

**Theorem 2**. *The algebra*

$$
\mathcal{A}_\Omega \overset{\text{def.}}{=} \left\{ \mathcal{F} : \mathcal{P}(\Omega)_{/\sim} \to \mathbb{R}, \exists n \in \mathsf{NN}, \exists \varphi : \Omega^n \to \mathbb{R} \textit{invariant}, \forall \alpha, \mathcal{F}(\alpha) = \int_{\Omega^n} \varphi \mathrm{d}\alpha^{\otimes n} \right\}
$$

*where $\otimes n$ denotes the n-fold tensor product, is dense in $\mathcal{C}(\mathcal{M}_+^1(\Omega)_{/\sim})$.*

*Proof.* This result follows from the Stone-Weierstrass theorem. Since $\Omega$ is compact, by Banach-Alaoglu theorem, we obtain that $\mathcal{P}(\Omega)$ is weakly-* compact, hence $\mathcal{P}(\Omega)_{/\sim}$ also is. In order to apply Stone-Weierstrass, we show that $\mathcal{A}_\Omega$ contains a non-zero constant function and is an

algebra that separates points. A (non-zero, constant) 1-valued function is obtained with $n = 1$ and $\varphi = 1$. Stability by scalar is straightforward. For stability by sum: given $(\mathcal{F}_1, \mathcal{F}_2) \in \mathcal{A}_\Omega^2$ (with associated functions $(\varphi_1, \varphi_2)$ of tensorization degrees $(n_2, n_2)$), we denote $n \overset{\text{def.}}{=} \max(n_1, n_2)$ and $\varphi(x_1, \ldots, x_n) \overset{\text{def.}}{=} \varphi_1(x_1, \ldots, x_{n_1}) + \varphi_2(x_1, \ldots, x_{n_2})$ which is indeed invariant, hence $\mathcal{F}_1 + \mathcal{F}_2 = \int_{\Omega^n} \varphi \mathrm{d}\alpha^{\otimes n} \in \mathcal{A}_\Omega$. Similarly, for stability by product: denoting this time $n = n_1 + n_2$, we introduce the invariant $\varphi(x_1, \ldots, x_n) = \varphi_1(x_1, \ldots, x_{n_1}) \times \varphi_2(x_{n_1+1}, \ldots, x_n)$, which shows that $\mathcal{F} = \mathcal{F}_1 \times \mathcal{F}_2 \in \mathcal{A}_\Omega$ using Fubini's theorem. Finally, $\mathcal{A}_\Omega$ separates points: if $\alpha \neq \nu$, then there exists a symmetrized domain $S$ such that $\alpha(S) \neq \nu(S)$: indeed, if for all symmetrized domains $S$, $\alpha(S) = \nu(S)$, then $\alpha(\Omega) = \nu(\Omega)$ which is absurd. Taking $n = 1$ and $\varphi = 1_S$ (invariant since $S$ is symmetrized) yields an $\mathcal{F}$ such that $\mathcal{F}(\alpha) \neq \mathcal{F}(\nu)$. □