Stochastic Deep Networks

Gwendoline de Bie [1] Gabriel Peyré [2 1] Marco Cuturi [3 4]

Abstract

Machine learning is increasingly targeting areas where input data cannot be accurately described by a single vector, but can be modeled instead using the more flexible concept of *random* vectors, namely probability measures or more simply point clouds of varying cardinality. Using deep architectures on measures poses, however, many challenging issues. Indeed, deep architectures are originally designed to handle fixed-length vectors, or, using recursive mechanisms, ordered sequences thereof. In sharp contrast, measures describe a varying number of weighted observations with no particular order. We propose in this work a deep framework designed to handle crucial aspects of measures, namely permutation invariances, variations in weights and cardinality. Architectures derived from this pipeline can (i) map measures to measures — using the concept of push-forward operators; (ii) bridge the gap between measures and Euclidean spaces — through integration steps. This allows to design discriminative networks (to classify or reduce the dimensionality of input measures), generative architectures (to synthesize measures) and recurrent pipelines (to predict measure dynamics). We provide a theoretical analysis of these building blocks, review our architectures' approximation abilities and robustness w.r.t. perturbation, and try them on various discriminative and generative tasks.

1. Introduction

Deep networks can now handle increasingly complex structured data types, starting historically from images (Krizhevsky et al., 2012) and speech (Hinton et al., 2012) to deal now with shapes (Wu et al., 2015a), sounds (Lee et al., 2009), texts (LeCun et al., 1998) or graphs (Henaff et al., 2015). In each of these applications, deep networks rely on the composition of several elementary functions, whose tensorized operations stream well on GPUs, and whose computational graphs can be easily automatically differentiated through back-propagation. Initially designed for vectorial features, their extension to *sequences* of vectors using recurrent mechanisms (Hochreiter and Schmidhuber, 1997) had an enormous impact.

Our goal is to devise neural architectures that can handle *probability distributions* under any of their usual form: as discrete measures supported on (possibly weighted) point clouds, or densities one can sample from. Such probability distributions are challenging to handle using recurrent networks because no order between observations can be used to treat them recursively (although some adjustments can be made, as discussed in Vinyals et al. 2016) and because, in the discrete case, their size may vary across observations. There is, however, a strong incentive to define neural architectures that can handle distributions as inputs or outputs. This is particularly evident in computer vision, where the naive representation of complex 3D objects as vectors in spatial grids is often too costly memorywise, leads to a loss in detail, destroys topology and is blind to relevant invariances such as shape deformations. These issues were successfully tackled in a string of papers well adapted to such 3D settings (Qi et al., 2016; 2017; Fan et al., 2016). In other cases, ranging from physics (Godin et al., 2007), biology (Grover et al., 2011), ecology (Tereshko, 2000) to census data (Guckenheimer et al., 1977), populations cannot be followed at an individual level due to experimental costs or privacy concerns. In such settings where only macroscopic states are available, *densities* appear as the right object to perform inference tasks.

1.1. Previous works

Specificities of Probability Distributions. Data described in point clouds or sampled i.i.d. from a density are given *unordered*. Therefore architectures dealing with them are expected to be *permutation invariant*; they are also often expected to be equivariant to geometric transformations of input points (translations, rotations) and to capture *local structures* of points. Permutation invariance

[1]École Normale Supérieure, DMA, Paris, France [2]CNRS [3]CREST/ENSAE Paristech [4]Google Brain, Paris, France. Correspondence to: GDB <Gwendoline.De.Bie@ens.fr>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

or equivariance (Ravanbakhsh et al., 2016; Zaheer et al., 2017), or with respect to general groups of transformations (Gens and Domingos, 2014; Cohen and Welling, 2016; Ravanbakhsh et al., 2017) have been characterized, but without tackling the issue of locality. Pairwise interactions (Chen et al., 2014; Cheng et al., 2016; Guttenberg et al., 2016) are appealing and helpful in building permutation equivariant layers handling local information. Other strategies consist in augmenting the training data by all permutations or finding its *best ordering* (Vinyals et al., 2015). (Qi et al., 2016; 2017) are closer to our work in the sense that they combine the search for local features to permutation invariance, achieved by max pooling.

(Point) Sets *vs.* Probability (Distributions). An important distinction should be made between point *sets*, and point *clouds* which stand usually for discrete probability measures with uniform masses. The natural topology of (point) sets is the Hausdorff distance. That distance is very different from the natural topology for probability distributions, that of the convergence in law, *a.k.a* the weak* topology of measures. The latter is metrized (among other metrics) by the Wasserstein (optimal transport) distance, which plays a key role in our work. This distinction between sets and probability is crucial, because the architectures we propose here are designed to capture stably and efficiently regularity of maps to be learned with respect to the convergence in law. Note that this is a crucial distinction between our work and that proposed in PointNet (Qi et al., 2016) and PointNet++ (Qi et al., 2017), which are designed to be smooth and efficients architectures for the Hausdorff topology of point sets. Indeed, they are *not* continuous for the topology of measures (because of the max-pooling step) and cannot approximate efficiently maps which are smooth (e.g. Lipschitz) for the Wasserstein distance.

Centrality of optimal transport. The Wasserstein distance plays a central role in our architectures that are able to handle measures. Optimal transport has recently gained popularity in machine learning due to fast approximations, which are typically obtained using strongly-convex regularizers such as the entropy (Cuturi, 2013). The benefits of this regularization paved the way to the use of OT in various settings (Courty et al., 2017; Rolet et al., 2016; Huang et al., 2016). Although Wasserstein metrics have long been considered for inference purposes (Bassetti et al., 2006), their introduction in deep learning architectures is fairly recent, whether it be for generative tasks (Bernton et al., 2017; Arjovsky et al., 2017; Genevay et al., 2018) or regression purposes (Frogner et al., 2015; Hashimoto et al., 2016). The purpose of our work is to provide an extension of these works, to ensure that deep architectures can be used at a granulary level on measures directly. In particular, our work shares some of the goals laid out in (Hashimoto et al., 2016), which considers recurrent architectures for measures (a special case of our framework). The most salient distinction with respect to our work is that our building blocks take into account multiple interactions between samples from the distributions, while their architecture has no interaction but takes into account diffusion through the injection of random noise.

1.2. Contributions

In this paper, we design deep architectures that can (i) map measures to measures (ii) bridge the gap between measures and Euclidean spaces. They can thus accept as input for instance discrete distributions supported on (weighted) point clouds with an arbitrary number of points, can generate point clouds with an arbitrary number of points (arbitrary refined resolution) and are naturally invariant to permutations in the ordering of the support of the measure. The mathematical idealization of these architectures are infinite dimensional by nature, and they can be computed numerically either by sampling (Lagrangian mode) or by density discretization (Eulerian mode). The Eulerian mode resembles classical convolutional deep network, while the Lagrangian mode, which we focus on, defines a new class of deep neural models. Our first contribution is to detail this new framework for supervised and unsupervised learning problems over probability measures, making a clear connexion with the idea of iterative transformation of random vectors. These architectures are based on two simple building blocks: interaction functionals and self-tensorization. This machine learning pipeline works hand-in-hand with the use of optimal transport, both as a mathematical performance criterion (to evaluate smoothness and approximation power of these models) and as a loss functional for both supervised and unsupervised learning. Our second contribution is theoretical: we prove both quantitative Lipschitz robustness of these architectures for the topology of the convergence in law and universal approximation power. Our last contribution is a showcase of several instantiations of such deep stochastic networks for classification (mapping measures to vectorial features), generation (mapping back and forth measures to code vectors) and prediction (mapping measures to measures, which can be integrated in a recurrent network).

1.3. Notations

We denote $X \in \mathcal{R}(\mathbb{R}^q)$ a random vector on $\mathbb{R}^q$ and $\alpha_X \in \mathcal{M}_+^1(\mathbb{R}^q)$ its law, which is a positive Radon measure with unit mass. It satisfies for any continuous map $f \in \mathcal{C}(\mathbb{R}^q)$, $\mathbb{E}(f(X)) = \int_{\mathbb{R}^q} f(x)\mathrm{d}\alpha_X(x)$. Its expectation is denoted $\mathbb{E}(X) = \int_{\mathbb{R}^q} x\mathrm{d}\alpha_X(x) \in \mathbb{R}^q$. We denote $\mathcal{C}(\mathbb{R}^q;\mathbb{R}^r)$ the space of continuous functions from $\mathbb{R}^q$ to $\mathbb{R}^r$ and $\mathcal{C}(\mathbb{R}^q) \overset{\text{def.}}{=} \mathcal{C}(\mathbb{R}^q;\mathbb{R})$. In this paper, we focus on the *law* of a random vector, so that two vectors X and X' having the same law (denoted $X \sim X'$) are considered to be indistinguishable.

2. Stochastic Deep Architectures

In this section, we define *elementary blocks*, mapping random vectors to random vectors, which constitute a *layer* of our proposed architectures, and depict how they can be used to build deeper networks.

2.1. Elementary Blocks

Our deep architectures are defined by stacking a succession of simple elementary blocks that we now define.

Definition 1 (Elementary Block)**.** *Given a function $f : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}^r$, its associated elementary block $T_f : \mathcal{R}(\mathbb{R}^q) \to \mathcal{R}(\mathbb{R}^r)$ is defined as*

$$\forall\, X \in \mathcal{R}(\mathbb{R}^q), \quad T_f(X) \overset{\text{def.}}{=} \mathbb{E}_{X' \sim X}(f(X, X')) \qquad (1)$$

where X' is a random vector independent from X having the same distribution.

Remark 1 (Discrete random vectors). A particular instance, which is the setting we use in our numerical experiments, is when X is distributed uniformly on a set $(x_i)_{i=1}^n$ of n points i.e. when $\alpha_X = \frac{1}{n}\sum_{i=1}^n \delta_{x_i}$. In this case, $Y = T_f(X)$ is also distributed on n points

$$\alpha_Y = \frac{1}{n}\sum_{i=1}^n \delta_{y_i} \quad \text{where} \quad y_i = \frac{1}{n}\sum_{j=1}^n f(x_i, x_j).$$

This elementary operation (1) displaces the distribution of X according to pairwise interactions measured through the map f. As done usually in deep architectures, it is possible to localize the computation at some scale τ by imposing that $f(x, x')$ is zero for $\|x - x'\| \geqslant \tau$, which is also useful to reduce the computation time.

Remark 2 (Fully-connected case). As it is customary for neural networks, the map $f : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}^r$ we consider for our numerical applications are affine maps composed by a pointwise non-linearity, i.e.

$$f(x, x') = (\lambda(y_k))_{k=1}^r \quad \text{where} \quad y = A \cdot [x; x'] + b \in \mathbb{R}^r$$

where $\lambda : \mathbb{R} \to \mathbb{R}$ is a pointwise non-linearity (in our experiments, $\lambda(s) = \max(s, 0)$ is the ReLu map). The parameter is then $\theta = (A, b)$ where $A \in \mathbb{R}^{r \times 2q}$ is a matrix and $b \in \mathbb{R}^r$ is a bias.

Remark 3 (Deterministic layers). Classical "deterministic" deep architectures are recovered as special cases when X is a constant vector, assuming some value $x \in \mathbb{R}^q$ with probability 1, i.e. $\alpha_X = \delta_x$. A stochastic layer can output such a deterministic vector, which is important for instance for classification scores in supervised learning (see Section 4 for an example) or latent code vectors in auto-encoders (see Section 4 for an illustration). In this case, the map $f(x, x') = g(x')$ does not depend on its first argument, so that $Y = T_f(X)$ is constant equal to $y = \mathbb{E}_X(g(X)) = \int_{\mathbb{R}^q} g(x)\mathrm{d}\alpha_X(x)$. Such a layer thus computes a summary statistic vector of X according to g.

Remark 4 (Push-Forward). In sharp contrast to the previous remark, one can consider the case $f(x, x') = h(x)$ so that f only depends on its first argument. One then has $T_f(X) = h(X)$, which corresponds to the notion of push-forward of measure, denoted $\alpha_{T_f(X)} = h_\sharp \alpha_X$. For instance, for a discrete law $\alpha_X = \frac{1}{n}\sum_i \delta_{x_i}$ then $\alpha_{T_f(X)} = \frac{1}{n}\sum_i \delta_{h(x_i)}$. The support of the law of X is thus deformed by h.

Remark 5 (Higher Order Interactions and Tensorization). Elementary Blocks are generalized to handle higher-order interactions by considering $f : (\mathbb{R}^q)^N \to \mathbb{R}^r$, one then defines $T_f(X) \overset{\text{def.}}{=} \mathbb{E}_{X_2,\ldots,X_N}(f(X, X_2, \ldots, X_N))$ where $(X_2, \ldots, X_N)$ are independent and identically distributed copies of X. An equivalent and elegant way to introduce these interactions in a deep architecture is by adding a tensorization layer, which maps $X \mapsto X_2 \otimes \ldots \otimes X_N \in \mathcal{R}((\mathbb{R}^q)^{N-1})$. Section 3 details the regularity and approximation power of these tensorization steps.

2.2. Building Stochastic Deep Architectures

These elementary blocks are stacked to construct deep architectures. A *stochastic deep architecture* is thus a map

$$X \in \mathcal{R}(\mathbb{R}^{q_0}) \mapsto Y = T_{f_T} \circ \cdots \circ T_{f_1}(X) \in \mathcal{R}(\mathbb{R}^{q_T}), \quad (2)$$

where $f_t : \mathbb{R}^{q_{t-1}} \times \mathbb{R}^{q_{t-1}} \to \mathbb{R}^{q_t}$. Typical instances of these architectures includes:

- *Predictive:* this is the general case where the architecture inputs a random vector and outputs another random vector. This is useful to model for instance time evolution using recurrent networks, and is used in Section 4 to tackle a dynamic prediction problem.
- *Discriminative:* in which case Y is constant equal to a vector $y \in \mathbb{R}^{q_T}$ (i.e. $\alpha_Y = \delta_y$) which can represent either a classification score or a latent code vector. Following Remark 3, this is achieved by imposing that f_T only depends on its second argument. Section 4 shows applications of this setting to classification and variational auto-encoders (VAE).
- *Generative:* in which case the network should input a deterministic code vector $\tilde{x}_0 \in \mathbb{R}^{\tilde{q}_0}$ and should output a random vector Y. This is achieved by adding extra randomization through a fixed random vector $\bar{X}_0 \in \mathcal{R}(\mathbb{R}^{q_0 - \tilde{q}_0})$ (for instance a Gaussian noise) and stacking $X_0 = (\tilde{x}_0, \bar{X}_0) \in \mathcal{R}(\mathbb{R}^{q_0})$. Section 4

shows an application of this setting to VAE generative models. Note that while we focus for simplicity on VAE models, it is possible to use our architectures for GANs (Goodfellow et al., 2014) as well.

2.3. Recurrent Nets as Gradient Flows

Following the work of (Hashimoto et al., 2016), in the special case $\mathbb{R}^q = \mathbb{R}^r$, one can also interpret iterative applications of such a T_f (i.e. considering a recurrent deep network) as discrete optimal transport gradient flows (Santambrogio, 2015) (for the W_2 distance, see also Definition (3)) in order to minimize a quadratic interaction energy $\mathcal{E}(\alpha) \overset{\text{def.}}{=} \int_{\mathbb{R}^q \times \mathbb{R}^q} F(x, x')\mathrm{d}\alpha(x)\mathrm{d}\alpha(x')$ (we assume for ease of notation that F is symmetric). Indeed, introducing a step size $\tau > 0$, setting $f(x, x') = x - 2\tau \nabla_x F(x, x')$, one sees that the measure α_{X_ℓ} defined by the iterates $X_{\ell+1} = T_f(X_\ell)$ of a recurrent nets is approximating at time $t = \ell\tau$ the Wasserstein gradient flow $\alpha(t)$ of the energy $\mathcal{E}$. As detailed for instance in (Santambrogio, 2015), such a gradient flow is the solution of the PDE $\frac{\partial \alpha}{\partial t} = \mathrm{div}(\alpha \nabla(\mathcal{E}'(\alpha)))$ where $\mathcal{E}'(\alpha) = \int_{\mathbb{R}^q} F(x, \cdot)\mathrm{d}\alpha(x)$ is the "Euclidean" derivative of $\mathcal{E}$. The pioneer work of (Hashimoto et al., 2016) only considers linear and entropy functionals of the form $\mathcal{E}(\alpha) = \int (F(x) + \log(\frac{\mathrm{d}\alpha}{\mathrm{d}x}))\mathrm{d}\alpha(x)$ which leads to evolutions $\alpha(t)$ being Fokker-Plank PDEs. Our work can thus be interpreted as extending this idea to the more general setting of interaction functionals (see Section 3 for the extension beyond pairwise interactions).

3. Theoretical Analysis

In order to get some insight on these deep architectures, we now highlight some theoretical results detailing the regularity and approximation power of these functionals. This theoretical analysis relies on the Wasserstein distance, which allows us to make quantitative statements associated to the convergence in law.

3.1. Convergence in Law Topology

Wasserstein distance. In order to measure regularity of the involved functionals, and also to define loss functions to fit these architectures (see Section 4), we consider the p-Wasserstein distance (for $1 \leqslant p < +\infty$) between two probability distributions $(\alpha, \beta) \in \mathcal{M}_+^1(\mathbb{R}^q)$

$$\mathrm{W}_p^p(\alpha, \beta) \overset{\text{def.}}{=} \min_{\pi_1 = \alpha, \pi_2 = \beta} \int_{(\mathbb{R}^q)^2} \|x - y\|^p \mathrm{d}\pi(x, y) \quad (3)$$

where $\pi_1, \pi_2 \in \mathcal{M}_+^1(\mathbb{R}^q)$ are the two marginals of a coupling measure π, and the minimum is taken among coupling measures $\pi \in \mathcal{M}_+^1(\mathbb{R}^q \times \mathbb{R}^q)$.

A classical result (see (Santambrogio, 2015)) asserts that W_1 is a norm, and can be conveniently computed using

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

where $\mathrm{Lip}(g)$ is the Lipschitz constant of a map $g : \mathcal{X} \to \mathbb{R}$ (with respect to the Euclidean norm unless otherwise stated).

With an abuse of notation, we write $\mathrm{W}_p(X, Y)$ to denote $\mathrm{W}_p(\alpha_X, \alpha_Y)$, but one should be careful that we are considering distances between laws of random vectors. An alternative formulation is $\mathrm{W}_p(X, Y) = \min_{X', Y'} \mathbb{E}(\|X' - X'\|^p)^{1/p}$ where (X', Y') is a couple of vectors such that X' (resp. Y') has the same law as X (resp. Y), but of course X' and Y' are not necessarily independent. The Wasserstein distance metrizes the convergence in law (denoted $\rightharpoonup$) in the sense that $X_k \rightharpoonup X$ is equivalent to $\mathrm{W}_1(X_k, X) \to 0$.

In the numerical experiments, we estimate W_p using Sinkhorn's algorithm (Cuturi, 2013), which provides a smooth approximation amenable to (possibly stochastic) gradient descent optimization schemes, whether it be for generative or predictive tasks (see Section 4).

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

$$\forall\, (X, Y) \in \mathcal{R}(\mathbb{R}^q)^2,\ \mathrm{W}_1(T(X), T(Y)) \leqslant C\, \mathrm{W}_1(X, Y). \quad (4)$$

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

3.2. Regularity of Building blocks

Elementary blocks. The following proposition, whose proof can be found in Appendix B, shows that elementary blocks are robust to input perturbations.

Proposition 1 (Lipschitzianity of elementary blocks). *If for all x, $f(x, \cdot)$ and $f(\cdot, x)$ are $C(f)$-Lipschitz, then T_f is $2rC(f)$-Lipschitz in the sense of* (4).

As a composition of Lipschitz functions defines Lipschitz maps, the architectures of the form (2) are thus Lipschitz, with a Lipschitz constant upper-bounded by $2\sum_t q_t C(f_t)$, where we used the notations of Proposition 1.

Tensorization. As highlighted in Remark 5, tensorization plays an important role to define higher-order interaction blocks.

Definition 2 (Tensor product)**.** *Given $(X,Y) \in \mathcal{R}(\mathcal{X}) \times \mathcal{R}(\mathcal{Y})$, a tensor product random vector is $X \otimes Y \overset{\text{def.}}{=} (X',Y') \in \mathcal{R}(\mathcal{X} \times \mathcal{Y})$ where X' and Y' are independent and have the same laws as X and Y. This means that $\mathrm{d}\alpha_{X\otimes Y}(x,y) = \mathrm{d}\alpha_X(x)\mathrm{d}\alpha_Y(y)$ is the tensor product of the measures.*

Remark 6 (Tensor Product between Discrete Measures). If we consider random vectors supported on point clouds, with laws $\alpha_X = \frac{1}{n}\sum_{i=1}^n \delta_{x_i}$ and $\alpha_Y = \frac{1}{m}\sum_{j=1}^m \delta_{y_j}$, then $X \otimes Y$ is a discrete random vector supported on nm points, since $\alpha_{X\otimes Y} = \frac{1}{nm}\sum_{i,j}\delta_{(x_i,y_j)}$.

The following proposition shows that tensorization blocks maintain the stability property of a deep architecture.

Proposition 2 (Lipschitzness of tensorization)**.** *One has, for $(X,X',Y,Y') \in \mathcal{R}(\mathcal{X})^2 \times \mathcal{R}(\mathcal{Y})^2$,*

$$\mathrm{W}_1(X\otimes Y, X'\otimes Y') \leqslant \mathrm{W}_1(X,X') + \mathrm{W}_1(Y,Y').$$

Proof. One has

$$\begin{aligned}
&\mathrm{W}_1(\alpha\otimes\beta,\alpha'\otimes\beta')\\
&= \max_{\mathrm{Lip}(g)\leqslant 1} \int_1 g(x,y)[\mathrm{d}\alpha(x)\mathrm{d}\beta(y) - \mathrm{d}\alpha'(x)\mathrm{d}\beta'(y)]\\
&= \max_{\mathrm{Lip}(g)\leqslant 1} \int_{\mathcal{X}}\int_{\mathcal{Y}} g(x,y)[\mathrm{d}\beta(y)-\mathrm{d}\beta'(y)]\mathrm{d}\alpha(x)\\
&+ \int_{\mathcal{Y}}\int_{\mathcal{X}} g(x,y)[\mathrm{d}\alpha(x)-\mathrm{d}\alpha'(x)]\mathrm{d}\beta(y),
\end{aligned}$$

hence the result. □

3.3. Approximation Theorems

Universality of elementary block. The following theorem shows that any continuous map between random vectors can be approximated to arbitrary precision using three elementary blocks. Note that it includes through Λ a fixed random input which operates as an "excitation block" similar to the generative VAE models studied in Section 4.2.

Theorem 1. *Let $\mathcal{F} : \mathcal{R}(\mathcal{X}) \to \mathcal{R}(\mathcal{Y})$ be a continuous map for the convergence in law, where $\mathcal{X} \subset \mathbb{R}^q$ and $\mathcal{Y} \subset \mathbb{R}^r$ are compact. Then $\forall \varepsilon > 0$ there exists three continuous maps f, g, h such that*

$$\forall X \in \mathcal{R}(\mathcal{X}), \quad \mathrm{W}_1(\mathcal{F}(X), T_h\circ\Lambda\circ T_g\circ T_f(X)) \leqslant \varepsilon. \tag{5}$$

where $\Lambda : X \mapsto (X,U)$ concatenates a uniformly distributed random vector U.

The architecture that we use to prove this theorem is displayed on Figure 1, bottom (left). Since f, g and h are smooth maps, according to the universality theorem of neural networks (Cybenko, 1989; Leshno et al., 1993) (assuming some restriction on the non-linearity λ, namely its being a nonconstant, bounded and continuous function), it is possible to replace each of them (at the expense of increasing ε) by a sequence of fully connected layers (as detailed in Remark 2). This is detailed in Section D of the appendix.

Since deterministic vectors are a special case of random vectors (see Remark 3), this results encompasses as a special case universality for vector-valued maps $\mathcal{F} : \mathcal{R}(\Omega) \to \mathbb{R}^r$ (used for instance in classification in Section 4.1) and in this case only 2 elementary blocks are needed. Of course the classical universality of multi-layer perceptron (Cybenko, 1989; Leshno et al., 1993) for vectors-to-vectors maps $\mathcal{F} : \mathbb{R}^q \to \mathbb{R}^r$ is also a special case (using a single elementary block).

Proof. (of Theorem 1) In the following, we denote the probability simplex as $\Sigma_n = \left\{a \in \mathbb{R}_+^n \; ; \; \sum_i a_i = 1\right\}$. Without loss of generality, we assume $\mathcal{X} \subset [0,1]^q$ and $\mathcal{Y} \subset [0,1]^r$. We consider two uniform grids of n and m points $(x_i)_{i=1}^n$ of $[0,1]^q$ and $(y_j)_{j=1}^m$ of $[0,1]^r$. On these grids, we consider the usual piecewise affine P1 finite element bases $(\varphi_i)_{i=1}^n$ and $(\psi_j)_{j=1}^m$, which are continuous hat functions supported on cells $(R_i)_i$ and $(S_j)_j$ which are cubes of width $2/n^{1/q}$ and $2/m^{1/r}$. We define discretization operators as $D_{\mathcal{X}} : \alpha \in \mathcal{M}_1^+(\mathcal{X}) \mapsto (\int_{R_i}\varphi_i \mathrm{d}\alpha)_{i=1}^n \in \Sigma_n$ and $D_{\mathcal{Y}} : \beta \in \mathcal{M}_1^+(\mathcal{Y}) \mapsto (\int_{S_j}\psi_j\mathrm{d}\beta)_{j=1}^m \in \Sigma_n$. We also define $D_{\mathcal{X}}^* : a \in \Sigma_n \mapsto \sum_i a_i\delta_{x_i} \in \mathcal{M}_1^+(\mathcal{X})$ and $D_{\mathcal{Y}}^* : b \in \Sigma_m \mapsto \sum_j b_j\delta_{y_i} \in \mathcal{M}_1^+(\mathcal{Y})$.

The map $\mathcal{F}$ induces a discrete map $G : \Sigma_n \to \Sigma_m$ defined by $G \overset{\text{def.}}{=} D_{\mathcal{Y}} \circ \mathcal{F} \circ D_{\mathcal{X}}^*$. Remark that $D_{\mathcal{X}}^*$ is continuous from Σ_n (with the usual topology on $\mathbb{R}^n$) to $\mathcal{M}_+^1(\mathcal{X})$ (with the convergence in law topology), $\mathcal{F}$ is continuous (for the convergence in law), $D_{\mathcal{Y}}$ is continuous from $\mathcal{M}_+^1(\mathcal{Y})$ (with the convergence in law topology) to Σ_m (with the usual topology on $\mathbb{R}^m$). This shows that G is continuous.

For any $b \in \Sigma_m$, Lemma 2 proved in the appendices defines a continuous map H so that, defining U to be a random vector uniformly distributed on $[0,1]^r$ (with law $\mathcal{U}$), $H(b,U)$ has law $(1-\varepsilon)D_{\mathcal{Y}}^*(b) + \varepsilon\mathcal{U}$.

We now have all the ingredients, and define the three continuous maps for the elementary blocks as

$$f(x,x') = (\varphi_i(x'))_{i=1}^n \in \mathbb{R}^n, \quad g(a,a') = G(a') \in \mathbb{R}^m,$$
$$\text{and} \quad h((b,u),(b',u')) = H(b,u) \in \mathcal{Y}.$$

The corresponding architecture is displayed on Figure 1, bottom. Using these maps, one needs to control the error between $\mathcal{F}$ and $\hat{\mathcal{F}} \overset{\text{def.}}{=} T_h\circ\Lambda\circ T_g\circ T_f = H_\sharp\circ\Lambda\circ D_{\mathcal{Y}}\circ\mathcal{F}\circ D_{\mathcal{X}}^*\circ\mathcal{D}_{\mathcal{X}}$ where we denoted $H_\sharp(b) \overset{\text{def.}}{=} H(b,\cdot)_\sharp\mathcal{U}$ the law of $H(b,U)$ (i.e. the pushforward of the uniform distribution $\mathcal{U}$ of U by $H(b,\cdot)$).

(i) We define $\hat{\alpha} \overset{\text{def.}}{=} D_{\mathcal{X}}^*\mathcal{D}_{\mathcal{X}}(\alpha)$. The diameters of the cells R_i is $\Delta_j = \sqrt{q}/n^{1/q}$, so that Lemma 1 in the appendices shows that $\mathrm{W}_1(\alpha, \hat{\alpha}) \leqslant \sqrt{q}/n^{1/q}$. Since $\mathcal{F}$ is continuous for the convergence in law, choosing n large enough ensures that $\mathrm{W}_1(\mathcal{F}(\alpha), \mathcal{F}(\hat{\alpha})) \leqslant \varepsilon$.

(ii) We define $\hat{\beta} \overset{\text{def.}}{=} D_{\mathcal{Y}}^* D_{\mathcal{Y}} \mathcal{F}(\hat{\alpha})$. Similarly, using m large enough ensures that $\mathrm{W}_1(\mathcal{F}(\hat{\alpha}), \hat{\beta}) \leqslant \varepsilon$.

(iii) Lastly, let us define $\tilde{\beta} \overset{\text{def.}}{=} H_\sharp \circ D_{\mathcal{Y}}(\hat{\beta}) = \hat{\mathcal{F}}(\alpha)$. By construction of the map H in Lemma 2, one has hat $\tilde{\beta} = (1-\varepsilon)\hat{\beta} + \varepsilon \mathcal{U}$ so that $\mathrm{W}_1(\tilde{\beta}, \hat{\beta}) = \varepsilon\, \mathrm{W}_1(\hat{\beta}, \mathcal{U}) \leqslant C\varepsilon$ for the constant $C = 2\sqrt{r}$ since the measures are supported in a set of diameter $\sqrt{r}$.

Putting these three bounds (i), (ii) and (iii) together using the triangular inequality shows that $\mathrm{W}_1(\mathcal{F}(\alpha), \hat{\mathcal{F}}(\alpha)) \leqslant (2 + C)\varepsilon$. □

Universality of tensorization. The following Theorem, whose proof can be found in Appendix E, shows that one can approximate any continuous map using a high enough order of tensorization followed by an elementary block.

Theorem 2. *Let $\mathcal{F} : \mathcal{R}(\Omega) \to \mathbb{R}$ a continuous map for the convergence in law, where $\Omega \subset \mathbb{R}^q$ is compact. Then $\forall \varepsilon > 0$, there exists $n > 0$ and a continuous function f such that*

$$\forall\, X \in \mathcal{R}(\Omega), \quad |\mathcal{F}(X) - T_f \circ \theta_n(X)| \leqslant \varepsilon \qquad (6)$$

where $\theta_n(X) = X \otimes \ldots \otimes X$ is the n-fold self tensorization.

The architecture used for this theorem is displayed on the bottom (right) of Figure 1. The function f appearing in (6) plays a similar role as in (5), but note that the two-layers factorizations provided by these two theorems are very different. It is an interesting avenue for future work to compare them theoretically and numerically.

4. Applications

To exemplify the use of our stochastic deep architectures, we consider classification, generation and dynamic prediction tasks. The goal is to highlight the versatility of these architectures and their ability to handle as input and/or output both probability distributions and vectors. In all cases, the procedures displayed similar results when rerun, hence results can be considered as quite stable and representative.

4.1. Classification tasks

MNIST Dataset. We perform classification on the 2-D MNIST dataset of handwritten digits. To convert a MNIST image into a 2D point cloud, we threshold pixel values (threshold $\rho = 0.5$) and use as a support of the input empirical measure the $n = 256$ pixels of highest intensity, represented as points $(x_i)_{i=1}^n \subset \mathbb{R}^2$ (if there are less than $n = 256$ pixels of intensity over ρ, we repeat input coordinates), which are remapped along each axis by mean and variance normalization. Each image is therefore turned into a sum of $n = 256$ Diracs $\frac{1}{n}\sum_i \delta_{x_i}$. Our stochastic network architecture is displayed on the top of Figure 1 and is composed of 5 elementary blocks $(T_{f_k})_{k=1}^5$ with an interleaved self-tensorisation layer $X \mapsto X \otimes X$. The first elementary block T_{f_1} maps measures to measures, the second one T_{f_2} maps a measure to a deterministic vector (i.e. does not depend on its first coordinate, see Remark 3), and the last layers are classical vectorial fully-connected ones. We use a ReLu non-linearity λ (see Remark 2). The weights are learnt with a weighted cross-entropy loss function over a training set of 55,000 examples and tested on a set of 10,000 examples. Initialization is performed through the Xavier method (Glorot and Bengio, 2010) and learning with the Adam optimizer (Kingma and Ba, 2014). Table 1 displays our results, compared with the PointNet (Qi et al., 2016) baseline. We observe that maintaining stochasticity among several layers is beneficial (as opposed to replacing one Elementary Block with a fully connected layer allocating the same amount of memory).

Table 1. MNIST classification results

	input type	error (%)
PointNet	point set	0.78
Ours	measure (1 stochastic layer)	1.07
Ours	measure (2 stochastic layers)	**0.76**

ModelNet40 Dataset. We evaluate our model on the ModelNet40 (Wu et al., 2015b) shape classification benchmark. The dataset contains 3-D CAD models from 40 man-made categories, split into 9,843 examples for training and 2,468 for testing. We consider $n = 1,024$ samples on each surface, obtained by a farthest point sampling procedure. Our classification network is similar to the one displayed on top of Figure 1, excepted that the layer dimensions are $[3, 10, 500, 800, 400, 40]$. Our results are displayed in figure 2. As previously observed in 2D, peformance is improved by maintaining stochasticity among several layers, for the same amount of allocated memory.

Table 2. ModelNet40 classification results

	input type	accuracy (%)
3DShapeNets	volume	77
Pointnet	point set	89.2
Ours	measure (1 stochastic layer)	82.0
Ours	measure (2 stochastic layers)	**83.5**

4.2. Generative networks

We further evaluate our framework for generative tasks, on a VAE-type model (Kingma and Welling, 2013) (note that it would be possible to use our architectures for

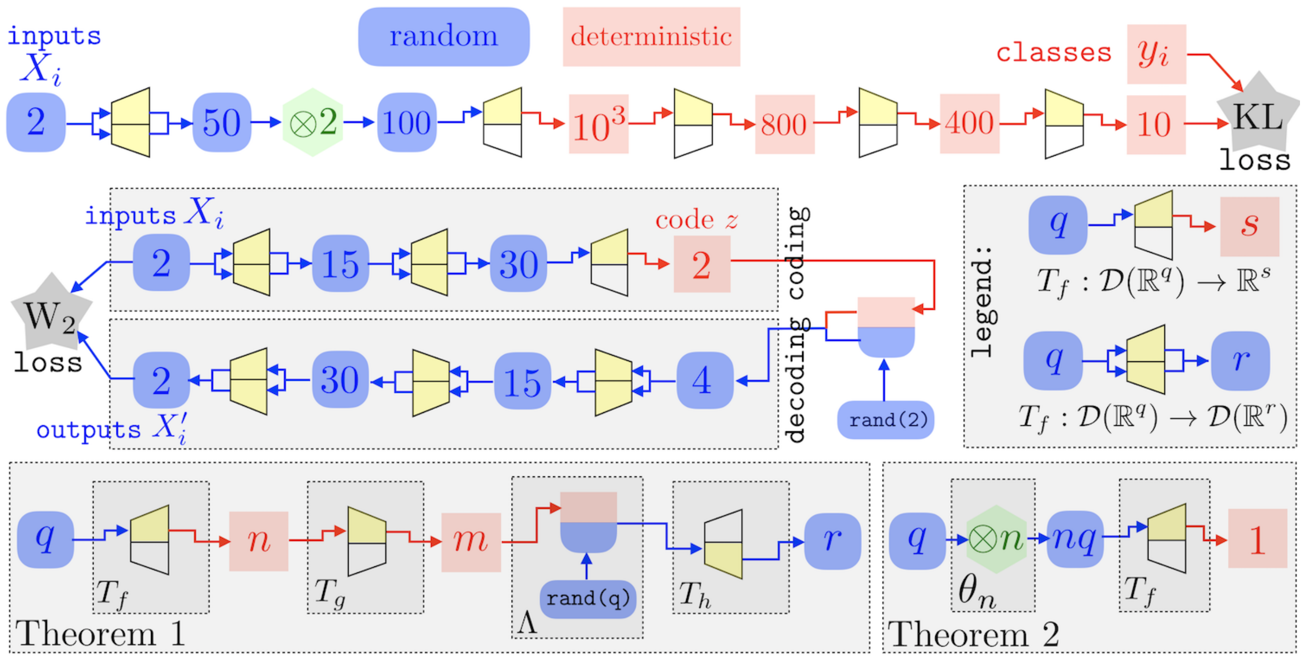

Figure 1. Top and center: two examples of deep stochastic architectures applied to the MNIST dataset: top for classification purpose (Section 4.1), center for generative model purpose (Section 4.2). Bottom: architecture for the proof of Theorems 1 and 2.

GANs (Goodfellow et al., 2014) as well). The task consists in generating outputs resembling the data distribution by decoding a random variable z sampled in a latent space $\mathcal{Z}$. The model, an encoder-decoder architecture, is learnt by comparing input and output measures using the W_2 Wasserstein distance loss, approximated using Sinkhorn's algorithm (Cuturi, 2013; Genevay et al., 2018). Following (Kingma and Welling, 2013), a Gaussian prior is imposed on the latent variable z. The encoder and the decoder are two mirrored architectures composed of two elementary blocks and three fully-connected layers each. The corresponding stochastic network architecture is displayed on the bottom of 1. Figure 2 displays an application on the MNIST database where the latent variable $z \in \mathbb{R}^2$ parameterizes a 2-D of manifold of generated digits. We use as input and output discrete probability measures of $n = 100$ Diracs, displayed as point clouds on the right of Figure 2.

4.3. Dynamic Prediction

The Cucker-Smale flocking model (Cucker and Smale, 2007) is non-linear dynamical system modelling the emergence of coherent behaviors, as for instance in the evolution of a flock of birds, by solving for positions and speed $x_i(t) \overset{\text{def.}}{=} (p_i(t) \in \mathbb{R}^d, v_i(t) \in \mathbb{R}^d)$ for $i = 1, \dots, n$

$$\dot{p}(t) = v(t), \quad \text{and} \quad \dot{v}(t) = \mathcal{L}(p(t))v(t) \tag{7}$$

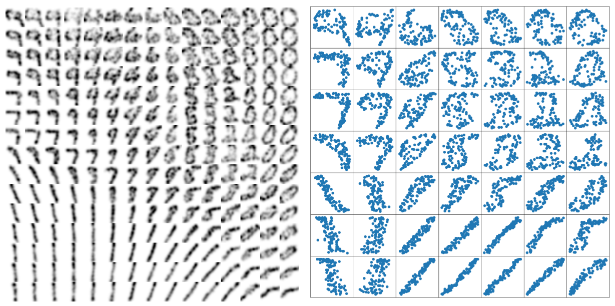

Figure 2. Left: Manifold of digits generated by the VAE network displayed on the bottom of 1. Right: Corresponding point cloud (displaying only a subset of the left images).

where $\mathcal{L}(p) \in \mathbb{R}^{n\times n}$ is the Laplacian matrix associated to a group of points $p \in (\mathbb{R}^d)^n$

$$\mathcal{L}(p)_{i,j} \overset{\text{def.}}{=} \frac{1}{1 + \|p_i - p_j\|^m}, \; \mathcal{L}(p)_{i,i} = -\sum_{j\neq i} \mathcal{L}(p)_{i,j}.$$

In the numerics, we set $m = 0.6$. This setting can be adapted to *weighted* particles $(x_i(t), \mu_i)_{i=1\cdots n}$, where each weight μ_i stands for a set of physical attributes impacting dynamics – for instance, mass – which is what we consider here. This model equivalently describes the evolution of the measure $\alpha(t) = \sum_{i=1}^n \mu_i \delta_{x_i(t)}$ in phase space $(\mathbb{R}^d)^2$, and following Remark 2.3 on the ability of our architectures to model dynamical system involving interactions, (7) can be discretized in time which leads to a recurrent network making use of a single elementary block T_f between each time step. Indeed, our block allows to maintain stochasticity among all layers – which is the natural way of proceeding to follow densities of particles over time.

It is however not the purpose of this article to study such a recurrent network and we aim at showcasing here whether deep (non-recurrent) architectures of the form (2) can accurately capture the Cucker-Smale model. More precisely, since in the evolution (7) the mean of $v(t)$ stays constant, we can assume $\sum_i v_i(t) = 0$, in which case it can be shown (Cucker and Smale, 2007) that particles ultimately reach stable positions $(p(t), v(t)) \mapsto (p(\infty), 0)$. We denote $\mathcal{F}(\alpha(0)) \overset{\text{def.}}{=} \sum_{i=1}^n \mu_i \delta_{p_i(\infty)}$ the map from some initial configuration in the phase space (which is described by a probability distribution $\alpha(0)$) to the limit probability distribution (described by a discrete measure supported on the positions $p_i(\infty)$). The goal is to approximate this map using our deep stochastic architectures. To showcase the flexibility of our approach, we consider a *non-uniform* initial measure $\alpha(0)$ and approximate its limit behavior $\mathcal{F}(\alpha(0))$ by a uniform one ($\mu_i = \frac{1}{n}$).

In our experiments, the measure $\alpha(t)$ models the dynamics of several (2 to 4) flocks of birds moving towards each other, exhibiting a limit behavior of a single stable flock. As shown in Figures 3 and 4, positions of the initial flocks are normally distributed, centered respectively at edges of a rectangle $(-4; 2), (-4; -2), (4; 2), (4; -2)$ with variance 1. Their velocities (displayed as arrows with lengths proportional to magnitudes in Figures 3 and 4) are uniformly chosen within the quarter disk $[0; -0.1] \times [0.1; 0]$. Their initial weights μ_i are normally distributed with mean 0.5 and sd 0.1, clipped by a ReLu and normalized. Figures 3 (representing densities) and 4 (depicting corresponding points' positions) show that for a set of $n = 720$ particles, quite different limit behaviors are successfully retrieved by a simple network composed of five elementary blocks with layers of dimensions $[2, 10, 20, 40, 60]$, learnt with a Wasserstein (Genevay et al., 2018) fitting criterion (computed with Sinkhorn's algorithm (Cuturi, 2013)).

Conclusion

In this paper, we have proposed a new formalism for stochastic deep architectures, which can cope in a seamless way with both probability distributions and deterministic feature vectors. The salient features of these architectures are their robustness and approximation power with respect to the convergence in law, which is crucial to tackle high dimensional classification, regression and generation problems over the space of probability distributions.

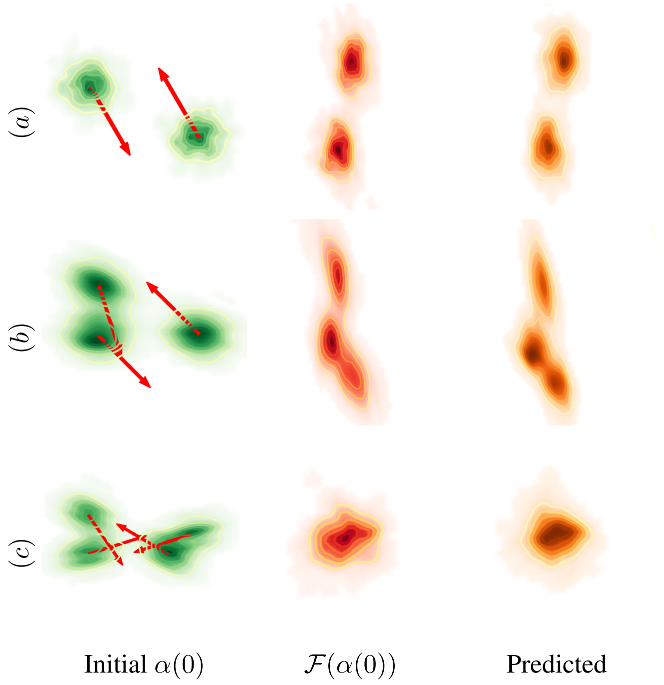

Figure 3. Prediction of the asymptotic density of the flocking model, for various initial speed values $v(0)$ and $n = 720$ particles. Eg. for top left cloud: (a) $v(0) = (0.050; -0.085)$; (b) $v(0) = (0.030; -0.094)$; (c) $v(0) = (0.056; -0.081)$.

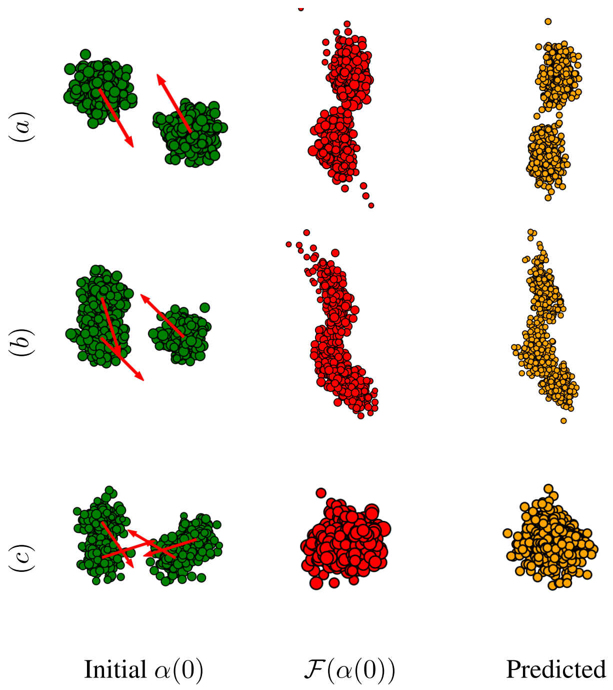

Figure 4. Prediction of particles' positions corresponding to Figure 3. Dots' diameters are proportial to weights μ_i (predicted ones all have the same size since $\mu_i = \frac{1}{n}$).

Acknowledgements

The work of Gwendoline de Bie was supported by a grant from Région Ile-de-France. The work of Gabriel Peyré has been supported by the European Research Council (ERC project NORIA).

References

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein gan. *arXiv preprint arXiv:1701.07875*.

Bassetti, F., Bodini, A., and Regazzini, E. (2006). On minimum kantorovich distance estimators. *Statistics & probability letters*, 76(12):1298–1302.

Bernton, E., Jacob, P., Gerber, M., and Robert, C. (2017). Inference in generative models using the Wasserstein distance. working paper or preprint.

Chen, X., Cheng, X., and Mallat, S. (2014). Unsupervised deep haar scattering on graphs. *Advances in Neural Information Processing Systems*, pages 1709–1717.

Cheng, X., Chen, X., and Mallat, S. (2016). Deep haar scattering networks. *Information and Inference: A Journal of the IMA*, 5(2):105–133.

Cohen, T. and Welling, M. (2016). Group equivariant convolutional networks. *International Conference on Machine Learning*, pages 2990–2999.

Courty, N., Flamary, R., Tuia, D., and Rakotomamonjy, A. (2017). Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1853–1865.

Cucker, F. and Smale, S. (2007). On the mathematics of emergence. *Japanese Journal of Mathematics*, 2(1):197–227.

Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, pages 2292–2300.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

Fan, H., Su, H., and Guibas, L. (2016). A point set generation network for 3d object reconstruction from a single image. *arXiv preprint arXiv:1612.00603*.

Frogner, C., Zhang, C., Mobahi, H., Araya, M., and Poggio, T. A. (2015). Learning with a wasserstein loss. *Advances in Neural Information Processing Systems*, pages 2053–2061.

Genevay, A., Peyré, G., and Cuturi, M. (2018). Learning generative models with sinkhorn divergences. *International Conference on Artificial Intelligence and Statistics*, pages 1608–1617.

Gens, R. and Domingos, P. M. (2014). Deep symmetry networks. *Advances in Neural Information Processing Systems 27*, pages 2537–2545.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Godin, M., Bryan, A. K., Burg, T. P., Babcock, K., and Manalis, S. R. (2007). Measuring the mass, density, and size of particles and cells using a suspended microchannel resonator. *Applied physics letters*, 91(12):123121.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Grover, W. H., Bryan, A. K., Diez-Silva, M., Suresh, S., Higgins, J. M., and Manalis, S. R. (2011). Measuring single-cell density. *Proceedings of the National Academy of Sciences*, 108(27):10992–10996.

Guckenheimer, J., Oster, G., and Ipaktchi, A. (1977). The dynamics of density dependent population models. *Journal of Mathematical Biology*, 4(2):101–147.

Guttenberg, N., Virgo, N., Witkowski, O., Aoki, H., and Kanai, R. (2016). Permutation-equivariant neural networks applied to dynamics prediction. *arXiv preprint arXiv:1612.04530*.

Hashimoto, T., Gifford, D., and Jaakkola, T. (2016). Learning population-level diffusions with generative rnns. *International Conference on Machine Learning*, pages 2417–2426.

Henaff, M., Bruna, J., and LeCun, Y. (2015). Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Kingsbury, B., and Sainath, T. (2012). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29:82–97.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Huang, G., Guo, C., Kusner, M. J., Sun, Y., Sha, F., and Weinberger, K. Q. (2016). Supervised word mover's distance. *Advances in Neural Information Processing Systems*, pages 4862–4870.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems 25*, pages 1097–1105.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, H., Pham, P., Largman, Y., and Ng, A. Y. (2009). Unsupervised feature learning for audio classification using convolutional deep belief networks. *Advances in Neural Information Processing Systems 22*, pages 1096–1104.

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.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2016). Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, pages 5105–5114.

Ravanbakhsh, S., Schneider, J., and Poczos, B. (2016). Deep learning with sets and point clouds. *arXiv preprint arXiv:1611.04500*.

Ravanbakhsh, S., Schneider, J., and Poczos, B. (2017). Equivariance through parameter-sharing. *arXiv preprint arXiv:1702.08389*.

Rolet, A., Cuturi, M., and Peyré, G. (2016). Fast dictionary learning with a smoothed wasserstein loss. *Artificial Intelligence and Statistics*, pages 630–638.

Santambrogio, F. (2015). Optimal transport for applied mathematicians. *Birkäuser, NY*.

Tereshko, V. (2000). Reaction-diffusion model of a honeybee colony's foraging behaviour. In *International Conference on Parallel Problem Solving from Nature*, pages 807–816. Springer.

Vinyals, O., Bengio, S., and Kudlur, M. (2015). Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*.

Vinyals, O., Bengio, S., and Kudlur, M. (2016). Order matters: Sequence to sequence for sets. In *International Conference on Learning Representations (ICLR)*.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015a). 3d shapenets: A deep representation for volumetric shapes. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015b). 3d shapenets: A deep representation for volumetric shapes. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. (2017). Deep sets. *Advances in Neural Information Processing Systems 30*, pages 3391–3401.