Compressive Learning of Generative Networks

Vincent Schellekens and Laurent Jacques *

ISPGroup, ICTEAM, UCLouvain, Louvain-La-Neuve - Belgium

Abstract. Generative networks implicitly approximate complex densities from their sampling with impressive accuracy. However, because of the enormous scale of modern datasets, this training process is often computationally expensive. We cast generative network training into the recent framework of *compressive learning*: we reduce the computational burden of large-scale datasets by first harshly compressing them in a single pass as a single sketch vector. We then propose a cost function, which approximates the Maximum Mean Discrepancy metric, but requires *only* this sketch, which makes it time- and memory-efficient to optimize.

1 Introduction

These last few years, data-driven methods took over the state-of-the-art in a staggering amount of research and engineering applications. This success owes to a combination of two factors: machine learning models that combine expressive power and good generalization properties (*e.g.*, deep neural networks), and unprecedented availability of training data in enormous quantities.

Among such models, *generative networks* (GNs) received a significant amount of interest for their ability to embed data-driven priors in general applications, *e.g.*, for solving inverse problems such as super-resolution, deconvolution, inpainting, or compressive sensing to name a few [1–4]. As explained in Sec. 2, GNs are deep neural networks (DNNs) trained to generate samples that mimic those available in a given dataset. By minimizing some well-crafted cost-function at the training, these networks implicitly learn the probability distribution synthesizing this dataset; passing randomly generated low-dimensional inputs through to the GN then generates new high-dimensional samples.

In generative *adversarial* networks (GANs) this cost is dictated by a discriminator network that classifies real (training) and fake (generated) examples, the generative and the discriminator networks being learned simultaneously in a two-player zero-sum game [5]. While GANs are the golden standard, achieving the state-of-the-art for a wide variety of tasks, they are notoriously hard to learn due to the need to balance carefully the training of the two networks.

MMD-GNs minimize the simpler Maximum Mean Discrepancy (MMD) cost function [6, 7], *i.e.*, a "kernelized" distance measuring the similarity of generated and real samples. Although training MMD-GNs is conceptually simpler than GANs — we can resort to simple gradient descent-based solvers (*e.g.*, SGD) — its computational complexity scales poorly with large-scale datasets: each iteration necessitates numerous (typically of the order of thousands) accesses to the whole dataset. This severely limits the practical use of MMD-GNs [8].

*VS and LJ are funded by Belgian National Science Foundation (F.R.S.-FNRS).

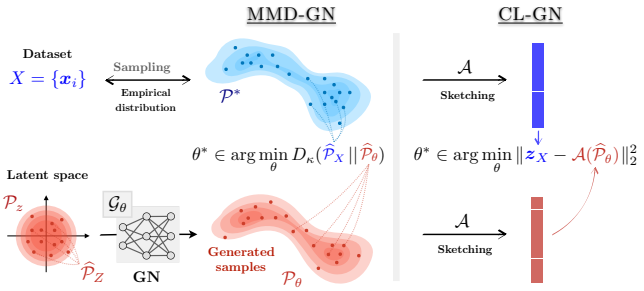

Fig. 1: General overview of our approach. The moment matching of MMD-GNs is replaced by sketching both X and the sampling $\widehat{\mathcal{P}}_\theta$. This compressive learning approach of GNs (or CL-GNs) is allowed by relating the RFF frequency distribution Λ to the MMD kernel κ.

Indeed, modern machine learning models such as GN are typically learned from numerous (*e.g.*, several million) training examples. Aggregating, storing, and learning from such large-scale datasets is a serious challenge, as the required communication, memory, and processing resources inflate accordingly. In compressive learning (CL), larger datasets can be exploited without demanding more computational resources. The data is first harshly compressed to a single vector called the *sketch*, a process done in a single, easily parallelizable pass over the dataset [9]. The actual learning is then performed from the sketch only, which acts as a light proxy for the whole dataset statistics. However, CL has for now been limited to "simple" models explicitly parametrized by a handful of parameters, such as k-means clustering, Gaussian mixture modeling or PCA [9].

This work proposes and assesses the potential of sketching to "compressively learn" deep generative networks (MMD-GNs) with greatly reduced computational cost (see Fig. 1). By defining a cost function and practical learning scheme, our approach serves as a prototype for compressively learning general generative models from sketches. The effectiveness of this scheme is tested on toy examples.

2 Background, related work and notations

To fix the ideas, given some space $\Sigma \subset \mathbb{R}^d$, we assimilate any dataset $X = \{\boldsymbol{x}_i\}_{i=1}^n \subset \Sigma$ with n samples to a discrete probability measure $\widehat{\mathcal{P}}_X$, *i.e.*, an empirical estimate for the *probability distribution* $\mathcal{P}^*$ generating X. Said differently, $\boldsymbol{x}_i \sim_{\text{i.i.d.}} \mathcal{P}^*$ and $\widehat{\mathcal{P}}_X := \frac{1}{|X|}\sum_{i=1}^n \delta_{\boldsymbol{x}_i}$, where $\delta_{\boldsymbol{c}}$ is the Dirac measure at $\boldsymbol{c} \in \Sigma$.

2.1. Compressive Learning: In CL, massive datasets are first efficiently (in one parallelizable pass) compressed into a single *sketch* vector of moderate size. The required parameters are then extracted from this sketch, using limited computational resources compared to usual algorithms that operate on the full dataset [9].

The sketch operator $\mathcal{A}(\mathcal{P}) := \frac{1}{\sqrt{m}}\left[\mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}} \exp\left(\mathrm{i}\boldsymbol{\omega}_j^T\boldsymbol{x}\right)\right]_{j=1}^m$ realizes an embedding of any (infinite dimensional) probability measure $\mathcal{P}$ into the low-dimensional domain $\mathbb{C}^m$. This sketching amounts to taking the expectation of the random Fourier features (RFF) [10] $\Phi(\boldsymbol{x}) := \frac{1}{\sqrt{m}}\exp(\mathrm{i}\boldsymbol{\Omega}^T\boldsymbol{x})$ of $\boldsymbol{x} \sim \mathcal{P}$, with $\Omega := (\boldsymbol{\omega}_1, \cdots, \boldsymbol{\omega}_m) \in \mathbb{R}^{d\times m}$. For large values of n, we expect that

$$\mathcal{A}(\mathcal{P}^*) \;\approx\; \boldsymbol{z}_X := \mathcal{A}(\widehat{\mathcal{P}}_X) = \frac{1}{n}\sum_{i=1}^n \Phi(\boldsymbol{x}_i) \in \mathbb{C}^m, \tag{1}$$

where z_X is the *sketch of the dataset* X. This sketch, which has a *constant size* m whatever the cardinality of X, thus embeds $\widehat{\mathcal{P}}_X$ by empirically averaging (or *pooling*) all RFF vectors $\Phi(\boldsymbol{x}_i)$. We still need to specify the RFF projection matrix $\boldsymbol{\Omega}$; it is randomly generated by drawing m "frequencies" $\boldsymbol{\omega}_j \sim_{\text{i.i.d.}} \Lambda$. In other words, $\mathcal{A}(\mathcal{P})$ corresponds here to a random sampling (according to the law Λ) of the characteristic function of $\mathcal{P}$ (*i.e.*, its Fourier transform). By Bochner's theorem [11], Λ is related to some shift-invariant *kernel* $\kappa(\boldsymbol{x},\boldsymbol{y}) = K(\boldsymbol{x}-\boldsymbol{y})$ by the (inverse) Fourier transform: $K(\boldsymbol{u}) = \mathbb{E}_{\boldsymbol{\omega}\sim\Lambda}\, e^{\mathrm{i}\boldsymbol{\omega}^T\boldsymbol{u}} =: \mathcal{F}^{-1}[\Lambda](\boldsymbol{u})$.

CL aims at learning, from only the sketch $\boldsymbol{z}_X$, an approximation $\mathcal{P}_\theta$ for the density $\mathcal{P}^*$, parametrized by $\theta \in \Theta$. For example, θ collects the position of the K centroids for compressive K-means, and the weights, centers and covariances of different Gaussians for compressive Gaussian mixtures fitting. This is achieved by solving the following density fitting ("sketch matching") problem:

$$\theta^* \in \arg\min_\theta \|\boldsymbol{z}_X - \mathcal{A}(\mathcal{P}_\theta)\|_2^2 \quad \text{s.t.} \quad \theta \in \Theta. \tag{2}$$

For large values of m, the cost in (2) estimates a metric D_κ between $\widehat{\mathcal{P}}_X$ and $\mathcal{P}_\theta$, called the Maximum Mean Discrepancy (MMD) [12], that is kernelized by κ, *i.e.*, writing $\kappa(\mathcal{P},\mathcal{Q}) := \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P},\boldsymbol{y}\sim\mathcal{Q}}\,\kappa(\boldsymbol{x},\boldsymbol{y})$, the MMD reads

$$D_\kappa^2(\mathcal{P},\mathcal{Q}) := \kappa(\mathcal{P},\mathcal{P}) + \kappa(\mathcal{Q},\mathcal{Q}) - 2\kappa(\mathcal{P},\mathcal{Q}). \tag{3}$$

Using Bochner's theorem, we can indeed rewrite (3) as

$$\begin{aligned}\|\mathcal{A}(\widehat{\mathcal{P}}_X) - \mathcal{A}(\mathcal{P}_\theta)\|_2^2 &= \tfrac{1}{m}\textstyle\sum_{j=1}^m \left|\, \mathbb{E}_{\boldsymbol{x}\sim\widehat{\mathcal{P}}_X}\, e^{\mathrm{i}\boldsymbol{\omega}_j^T\boldsymbol{x}} - \mathbb{E}_{\boldsymbol{y}\sim\mathcal{P}_\theta}\, e^{\mathrm{i}\boldsymbol{\omega}_j^T\boldsymbol{y}}\right|^2 \\ &\simeq \mathbb{E}_{\boldsymbol{\omega}\sim\Lambda}\left|\, \mathbb{E}_{\boldsymbol{x}\sim\widehat{\mathcal{P}}_X}\, e^{\mathrm{i}\boldsymbol{\omega}^T\boldsymbol{x}} - \mathbb{E}_{\boldsymbol{y}\sim\mathcal{P}_\theta}\, e^{\mathrm{i}\boldsymbol{\omega}^T\boldsymbol{y}}\right|^2 = D_\kappa^2(\widehat{\mathcal{P}}_X,\mathcal{P}_\theta). \end{aligned} \tag{4}$$

Provided Λ is supported on $\mathbb{R}^d$, $D_\kappa(\mathcal{P},\mathcal{Q}) = 0$ if and only if $\mathcal{P} = \mathcal{Q}$ [13]. Thus, minimizing (2) accurately estimates $\widehat{\mathcal{P}}_X$ from $\mathcal{P}_{\theta^*}$ if m is large compared to the complexity of the model; *e.g.*, in compressive K-means, CL requires experimentally $m = O(Kd)$ to learn the centroids of K clusters in $\mathbb{R}^d$.

The non-convex sketch matching problem (2) is generally solved with greedy heuristics (*e.g.*, CL-OMPR [14]). As they require a closed-form expression of $\mathcal{A}(\mathcal{P}_\theta)$ and the Jacobian $\nabla_\theta\mathcal{A}(\mathcal{P}_\theta)$, CL has so far be limited to cases where $\mathcal{P}_\theta$ is explicitly available and easy to manipulate.

2.2. Generative networks: To generates realistic data samples, a GN $\mathcal{G}_{\theta^*} : \Sigma_z \mapsto \Sigma$ (*i.e.*, a DNN) with weights $\theta^* \in \mathbb{R}^{d_\theta}$ is trained as follows. Given $\theta \in \Theta$, we compute the empirical distribution $\widehat{\mathcal{P}}_\theta := \mathcal{G}_\theta(\widehat{\mathcal{P}}_Z) = \frac{1}{n'}\sum_{i=1}^{n'} \mathcal{G}_\theta(\boldsymbol{z}_i)$ of n' inputs $Z = \{\boldsymbol{z}_i\}_{i=1}^{n'}$ randomly drawn in a low-dimensional *latent space* $\Sigma_z \subset \mathbb{R}^p$ from a simple distribution $\mathcal{P}_z$, *e.g.*, $\boldsymbol{z}_i \sim_{\text{i.i.d.}} \mathcal{N}(\boldsymbol{0},\boldsymbol{I}_p)$. By design, $\widehat{\mathcal{P}}_\theta$ is related to sampling the *pushforward* distribution of $\mathcal{P}_z$ by $\mathcal{G}_\theta$. The parameter θ^* is then set such that $\widehat{\mathcal{P}}_{\theta^*} \approx \widehat{\mathcal{P}}_X$. While several divergences have been proposed to quantify this objective, we focus here on minimizing the MMD metric $D_\kappa^2(\widehat{\mathcal{P}}_X,\widehat{\mathcal{P}}_\theta)$ [6, 7]. Using (3) and discarding constant terms, we get the MMD-GNs fitting problem:

$$\theta^* = \arg\min_\theta \textstyle\sum_{\boldsymbol{z}_i,\boldsymbol{z}_j\in Z} \kappa(\mathcal{G}_\theta(\boldsymbol{z}_i),\mathcal{G}_\theta(\boldsymbol{z}_j)) - 2\sum_{\boldsymbol{x}_i\in X,\boldsymbol{z}_j\in Z} \kappa(\boldsymbol{x}_i,\mathcal{G}_\theta(\boldsymbol{z}_j)). \tag{5}$$

Li et al. called this approach generative *moment matching* networks, as minimizing (3) amounts to matching all the (infinite) moments of $\mathcal{P}$ and $\mathcal{Q}$ thanks to the space kernelization yielded by κ [15] (see Fig. 1).

If κ is differentiable, gradient descent-based methods can be used to solve (5), using back-propagation to compute the gradients of $\mathcal{G}_\theta$. However, for n true samples and n' generated samples (or a batch-size), each evaluation of $D_\kappa^2(\widehat{\mathcal{P}}_X, \widehat{\mathcal{P}}_\theta)$ (and its gradient) requires $\mathcal{O}(nn' + n'^2)$ computations. Training MMD-GNs, while conceptually simpler than training GANs, is much slower due to all the pairwise evaluations of the kernel required at each iteration — especially for modern large-size datasets.

3 Compressive Learning of Generative Networks

In this work, given a dataset X, we propose to learn a generative network $\mathcal{G}_\theta$ using *only* the sketch $\boldsymbol{z}_X = \mathcal{A}(\widehat{\mathcal{P}}_X)$ defined in (1) (see Fig. 1). For this, given n' samples $Z = \{\boldsymbol{z}_i \sim \mathcal{P}_z\}_{i=1}^{n'}$, we solve a *generative network sketch matching problem* that selects $\theta^* = \arg\min_\theta \mathcal{L}(\theta; \boldsymbol{z}_X)$ with

$$\mathcal{L}(\theta; \boldsymbol{z}_X) := \big\|\mathcal{A}(\widehat{\mathcal{P}}_X) - \mathcal{A}\big(\mathcal{G}_\theta(\widehat{\mathcal{P}}_Z)\big)\big\|_2^2 = \big\|\boldsymbol{z}_X - \tfrac{1}{n'}\textstyle\sum_{i=1}^{n'} \Phi\big(\mathcal{G}_\theta(\boldsymbol{z}_i)\big)\big\|_2^2. \qquad (6)$$

From (4), we reach $\mathcal{L}(\theta; \boldsymbol{z}_X) \simeq D_\kappa(\widehat{\mathcal{P}}_X \,||\, \mathcal{G}_\theta(\widehat{\mathcal{P}}_Z))$ for large values of m, as established from the link relating κ and Λ. Compared to the exact MMD in (5), $\mathcal{L}(\theta; \boldsymbol{z}_X)$ is, however, much easier to optimize. Once the dataset sketch $\boldsymbol{z}_X$ has been pre-computed (in one single pass over X, possibly in parallel), we only need to compute $\mathcal{A}(\mathcal{G}_\theta(\widehat{\mathcal{P}}_Z))$ (*i.e.*, by computing n' contributions $\boldsymbol{z}_i \to \Phi(\mathcal{G}_\theta(\boldsymbol{z}_i))$ by feed-forward, before averaging them) to compute the Euclidean distance between both quantities. In short, we access X only once then discard it, and evaluating the cost has complexity $\mathcal{O}(n')$, *i.e.*, much smaller than $\mathcal{O}(nn' + n'^2)$, the complexity of the exact MMD (5) (see Sec. 2.2).

Equally importantly, the gradient $\nabla_\theta \mathcal{L}(\theta; \boldsymbol{z}_X)$ is easily computed. With the residual $\boldsymbol{r} := \boldsymbol{z}_X - \frac{1}{n'}\sum_{i=1}^{n'} \Phi(\mathcal{G}_\theta(\boldsymbol{z}_i))$ and $\boldsymbol{r}^H$ its conjugate transpose,

$$\nabla_\theta \widehat{\mathcal{L}}(\theta; \boldsymbol{z}_X) = -2 \cdot \tfrac{1}{n'}\textstyle\sum_{i=1}^{n'} \Re\big[\boldsymbol{r}^H \big(\frac{\partial \Phi(\boldsymbol{u})}{\partial \boldsymbol{u}}\big|_{\boldsymbol{u}=\mathcal{G}_\theta(\boldsymbol{z}_i)} \cdot \frac{\partial \mathcal{G}_\theta(\boldsymbol{z}_i)}{\partial \theta}\big)\big]. \qquad (7)$$

Above, $\frac{\partial \Phi(\boldsymbol{u})}{\partial \boldsymbol{u}} = \frac{\mathrm{i}}{\sqrt{m}} \operatorname{diag}(e^{\mathrm{i}\boldsymbol{\Omega}^T \boldsymbol{u}})\, \boldsymbol{\Omega}$ is the $m \times d$ Jacobian matrix listing the partial derivatives of the m sketch entries with respect to the d dimension of $\boldsymbol{u} \in \Sigma$, which is evaluated at the generated samples $\mathcal{G}_\theta(\boldsymbol{z}_i)$. The last term $\frac{\partial \mathcal{G}_\theta(\boldsymbol{z}_i)}{\partial \theta} \in \mathbb{R}^{d \times d_\theta}$ is computed by the back-propagation algorithm as it contains the derivative of the network output $\mathcal{G}_\theta(\boldsymbol{z}_i)$ (for $\boldsymbol{z}_i$ *fixed*) with respect to the parameters $\theta \in \mathbb{R}^{d_\theta}$. Algorithmically, the feature function Φ amounts to an extra layer on top of the GN, with fixed weights Ω and activation $t \mapsto \exp(\mathrm{i}t)$. We can then plug those expressions in any gradient-based optimisation solver[1].

[1] To boost the evaluations of (7), we can split Z into several minibatches Z_b of size $n_b \ll n'$; (7) is then replaced by successive minibatch gradients evaluated on the batches Z_b. As reported for MMD-GNs [6, 7], this only works for sufficiently large n_b, *e.g.*, $n_b = 1000$ in Sec. 4.

We conclude this section by an interesting interpretation of (6). While CL requires closed form expressions for $\mathcal{A}(\mathcal{P}_\theta)$ and $\nabla_\theta \mathcal{A}(\mathcal{P}_\theta)$, our GN formalism actually estimates those quantities by Monte-Carlo sampling, *i.e.*, replacing $\mathcal{P}_\theta$ by $\widehat{\mathcal{P}}_\theta$. This thus opens CL to non-parametric density fitting.

4 Experiments

For this preliminary work, we visually illustrate the effectiveness of minimizing (6) by considering three 2-D synthetic datasets made of $n = 10^5$ samples (see the top row of Fig. 2): *(i)* a 2-D spiral $\{(r_i, \phi_i)\}_{i=1}^n$, with $\phi_i \sim_{\text{i.i.d.}} \mathcal{U}([0, 2\pi))$ and $r_i \sim_{\text{i.i.d.}} \frac{\phi_i}{2\pi} + \mathcal{N}(0, \sigma_r^2)$, *(ii)* a Gaussian mixture models of 6 Gaussians, and *(iii)* samples in a circle, *i.e.*, $\phi_i \sim \mathcal{U}([0, 2\pi))$ and $r_i \sim R + \mathcal{N}(0, \sigma_r^2)$ for R and σ_r fixed. We learn a GN mapping $10-$dimensional random Gaussian vectors to $\mathbb{R}^2$, passing through seven fully connected hidden layers of 10 units each, activated by a Leaky ReLU function with slope 0.2. For this simple illustration, we sketch all datasets to a sketch of size $m = n/10 = 10^4$. We found experimentally from a few trials that setting Λ to a folded Gaussian distribution (see [14]) of scale $\sigma^2 = 10^{-3}$ is appropriate to draw the m frequencies $\{\boldsymbol{\omega}_j\}_{j=1}^m$. From those sketches, we then trained our generators according to (6), using the `keras` framework. We fixed the number of generated samples to $n' = 10^5$, which we split into mini-batches of $n_b = 1000$ samples when computing the gradient.

Fig. 2 compares densities of generated samples and re-generated samples after the training (from the known densities) through their 2-D histograms. Note that while the datasets are simplistic, we restricted the training time to a few minutes and, except for the frequency distribution, no hyper-parameter tuning was performed. Despite a few outliers and missing probability masses, the visual proximity of the histograms proves the capacity of our method to learn complex 2-D distributions. Our code and further experiments are available at `https://github.com/schellekensv/CL-GN`.

5 Conclusion

We proposed and tested a method that incorporates compressive learning ideas into generative network training from the Maximum Mean Discrepancy metric. When dealing with large-scale datasets, our approach is potentially orders of magnitude faster than exact MMD-based learning. However, to embrace higher-dimensional applications (*e.g.*, for image restorations or large scale inverse problems), future works will need to *(i)* devise efficient techniques to adjust the kernel κ (*i.e.*, the frequency distribution Λ) to the dataset X, and *(ii)* determine theoretically the required sketch size m in function of the dataset distribution $\mathcal{P}^*$. Concerning the choice of the kernel, a promising direction consists in tuning its Fourier transform Λ directly from a lightweight sketch [14]. As for the required sketch size, this problem certainly relates to measuring the "complexity" of the true generating density $\mathcal{P}^*$, and to the general open question of why over-parametrized deep neural networks generalize so well.

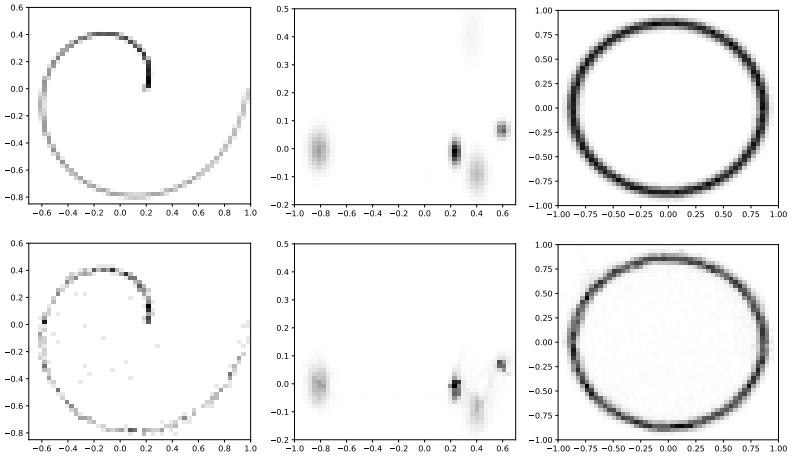

Fig. 2: Histograms (varying from white to black as the number of samples increases) of considered datasets (top) and 50000 generated samples, after training from the sketch (bottom).

References

[1] A. Bora et al. "Compressed sensing using generative models". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. 2017, pp. 537–546.

[2] M. Mardani et al. "Deep generative adversarial networks for compressed sensing automates MRI". In: *arXiv preprint:1706.00051* (2017).

[3] J. Rick Chang et al. "One Network to Solve Them All–Solving Linear Inverse Problems Using Deep Projection Models". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5888–5897.

[4] A. Lucas et al. "Using Deep Neural Networks for Inverse Problems in Imaging: Beyond Analytical Methods". In: *IEEE Signal Processing Magazine* 35.1 (2018), pp. 20–36.

[5] I. Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[6] Y. Li, K. Swersky, and R. Zemel. "Generative moment matching networks". In: *International Conference on Machine Learning*. 2015, pp. 1718–1727.

[7] G. K. Dziugaite et al. "Training generative neural networks via maximum mean discrepancy optimization". In: *arXiv preprint:1505.03906* (2015).

[8] M. Arjovsky et al. "Wasserstein GAN". In: *arXiv preprint:1701.07875* (2017).

[9] R. Gribonval et al. "Compressive Statistical Learning with Random Feature Moments". In: *ArXiv e-prints* (June 2017). arXiv: `1706.07180 [stat.ML]`.

[10] A. Rahimi and B. Recht. "Random Features for Large-Scale Kernel Machines". In: *Advances in Neural Information Processing Systems 20*. 2008, pp. 1177–1184.

[11] W. Rudin. *Fourier Analysis on Groups*. Interscience Publishers, 1962.

[12] A. Gretton et al. "A kernel two-sample test". In: *Journal of Machine Learning Research* 13.Mar (2012), pp. 723–773.

[13] B. K. Sriperumbudur et al. "Hilbert Space Embeddings and Metrics on Probability Measures". In: *J. Mach. Learn. Res.* 11 (Aug. 2010), pp. 1517–1561.

[14] N. Keriven et al. "Sketching for Large-Scale Learning of Mixture Models". In: *ArXiv e-prints* (June 2016). arXiv: `1606.02838 [cs.LG]`.

[15] A. R. Hall. *Generalized method of moments*. Oxford University Press, 2005.