An Expectation-Maximization interpretation of Generative Modeling via Drifting

David Picard *david.picard@enpc.fr*
LIGM, CNRS, Univ Gustave Eiffel, ENPC, Institut Polytechnique de Paris, Marne-la-Vallée, France

Abstract

Generative Modeling via Drifting (GMD) is a new way of training a generative model by handcrafting update rules. We propose to interpret GMD as a classical learning problem optimize with an algorithm akin to Expectation-Maximization. This allows us to derive updates rules in a more principled way. We also propose new research directions based on this interpretation.

1 Introduction

Generative Modeling via Drifting (GMD), introduced by Deng et al. (2026), is a novel way of training a generative model by considering a *drifting* term that governs how a learnable mapping $f_\theta : \mathcal{N} \to \mathcal{P}_{\text{data}}$ evolves throughout training. In this framework, the evolution of f_θ is obtained through a drifting equation

$$f_\theta^{i+1} = f_\theta^i + V_{p,q}\,, \tag{1}$$

such that $V_{p,q}$ is a drifting field that guides from distribution p to distribution q. To train f_θ, the authors then simply regress this update rule by solving the following minimization problem:

$$\min_\theta \mathbb{E}_\varepsilon \left[\| f_\theta(\varepsilon) - \text{sg}[f_\theta(\varepsilon) + V(f_\theta(\varepsilon))] \|^2 \right] , \tag{2}$$

with $V(f_\theta(\varepsilon))$ a handcrafted drifting field that leads to the data distribution $\mathcal{P}_{\text{data}}$, and $\text{sg}[\cdot]$ is the stop-gradient operator.

This approach consisting in handcrafting the update rules of a learning system might seem strange to the modern machine learning practitioner, but they were fairly common in the early days of the field as exemplified by the perceptron (Rosenblatt, 1957) or the various neural networks built around the Hebbian learning rule (Hebb, 1949). The modern common approach, which consists in defining a measurement of success for a given task and then framing the learning problem as empirical risk minimization with respect to that measurement, is well suited for a theoretical framework like the one proposed by Valiant (1984) where the learning system is characterized by how effective it is at the target task (*e.g.*, error rate on yet unseen data), in contrast to being characterized by how it learns in the case of handcrafted update rules. Notice how in both cases something has to be handcrafted anyway, the objective or the means to attain it.

In this short technical note, we show that GMD can also be obtained through the more common approach of defining an objective and then deriving the update rules following classical optimization tools. In doing so, we propose an interpretation of GMD that resembles the well-known Expectation-Maximization family of algorithms, with the key difference that the maximization step (a *minimization* in our case) does not have a closed-form solution and thus we have to resort to stochastic gradient descent. This allows us to obtain the drifting field V in a principled way by designing the task as the minimization of a discrepancy between the mapped distribution obtained through f_θ and $\mathcal{P}_{\text{data}}$. Admittedly, we offer almost nothing new compared to the original GMD paper, expect for a vantage point that may be more familiar to practitioners and easier to build on for future research directions.

We then show that our interpretation and the original GMD have similar behavior on toy data. We finally conclude by a discussion where we highlight what we believe are interesting research directions taking advantage of this interpretation.

2 Optimization using Expectation-Maximization

Let $D(\cdot, \cdot)$ be a discrepancy between two distributions. To ease notations, ε denotes either a sample from the Normal distribution or the Normal distribution directly when it makes sense. Let $\sharp$ denote the push forward operator, such that $f\sharp\varepsilon$ denotes the distribution obtained by mapping Gaussian samples ε using mapping f leading to $f(\varepsilon)$. Denote $\mathcal{P}$ the data distribution. We consider an optimal mapping $f^\star$ according to D,

$$f^\star = \inf_f D(f\sharp\varepsilon, \mathcal{P}). \tag{3}$$

Of course, $f^\star$ is unknown, and thus, the main objective is to approximate it using a neural network of parameters θ corresponding to the mapping f_θ. We obtain θ by solving the following optimization problem:

$$\min_\theta \mathbb{E}_\varepsilon \|f_\theta(\varepsilon) - f^\star(\varepsilon)\|^2 , \tag{4}$$

which we propose to do using an optimization scheme akin to an Expectation-Maximization scheme with the caveat that we want to minimize the approximation error rather to maximize a likelihood.

2.1 Expectation

Because we still do not know $f^\star$, we need to estimate it using empirical distributions for ε and $\mathcal{P}$. This amounts to solving

$$\inf_f \mathbb{E}_{\varepsilon, P\sim\mathcal{P}} \left[D(f\sharp\varepsilon, P)\right] \tag{5}$$

with P sampled from $\mathcal{P}$, and with D operating on empirical distributions (*e.g.*, sets of samples) instead of the original ones. As we have no other starting point than f_θ, the best approximate solution that we have corresponds to the steepest descent using the stochastic gradient:

$$f^\star(\varepsilon) \approx f_\theta(\varepsilon) - \nabla \mathbb{E}_{\varepsilon, P\sim\mathcal{P}} \left[D(f_\theta\sharp\varepsilon, P)\right] , \tag{6}$$

which does not require anything more than f_θ and $P \sim \mathcal{P}$, and being able to differentiate D.

For simplicity of notation, we introduce V such that

$$V = -\nabla \mathbb{E}_{\varepsilon, P\sim\mathcal{P}} \left[D(f_\theta\sharp\varepsilon, P)\right] , \tag{7}$$

$$f^\star \approx f_\theta(\varepsilon) + V . \tag{8}$$

2.2 Minimization

Using our approximation for $f^\star$, the objective for training the neural network then becomes:

$$\min_\theta \mathbb{E}_\varepsilon \|f_\theta(\varepsilon) - \mathrm{sg}[f_\theta(\varepsilon) + V]\|^2, \tag{9}$$

with $\mathrm{sg}[\cdot]$ denoting the stop-gradient operator. This stop-gradient operator is a common approach in EM algorithms since our pseudo target obtained in the E step involves f_θ. This is exactly the optimization problem described in Generative Modeling via Drifting, with the added interpretation that $V = -\nabla \mathbb{E}_{\varepsilon, P\sim\mathcal{P}} \left[D(f\sharp\varepsilon, P)\right]$ because it corresponds to minimizing a discrepancy between $\mathcal{P}$ and the mapped distribution $f\sharp\varepsilon$.

Notice that removing the sg[·] operator would cancel the occurrences of $f_\theta(\varepsilon)$ and lead to only minimizing the norm of the gradient of D. This would amount to search for the extrema of D, which is not as interesting as minimizing D directly, but corresponds to the fixed point condition given in the original paper. Also, notice that this formulation does not require propagating gradient *through* D, even if it still requires to compute the gradient of D.

2.3 Choice for D

In that setup, the key question is how to design D such that:

1. It has a computable gradient,
2. It is easy to estimate without requiring too many samples from $\mathcal{P}$.

The maximum mean discrepancy (MMD, Borgwardt et al. (2006)) with shift invariant kernels is a good candidate as, depending on the kernel, the gradient can easily be derived. This connection with the MMD is extensively discussed in the appendix of the original paper. However, we feel that it provides a more principled way of obtaining the update rules, which is why we explore it here. In the case of a shift invariant kernel $k(x, y) = k(x - y)$, one can show that

$$\nabla_f \mathrm{MMD}^2(\varepsilon, \mathcal{P})\big|_{f=f_\theta(\varepsilon)} = 2\mathbb{E}_{\varepsilon'}\left[\nabla k(f_\theta(\varepsilon) - f_\theta(\varepsilon'))\right] - 2\mathbb{E}_x\left[\nabla k(f_\theta(\varepsilon) - x)\right], \tag{10}$$

$$= V^- - V^+, \tag{11}$$

with $V^+ = 2\mathbb{E}_x\left[\nabla k(f_\theta(\varepsilon) - x)\right]$ and $V^- = 2\mathbb{E}_{\varepsilon'}\left[\nabla k(f_\theta(\varepsilon) - f_\theta(\varepsilon'))\right]$. In that case, V^+ corresponds to an attraction term that pulls $f_\theta(\varepsilon)$ towards x, and V^- is a repulsion force that pushes $f_\theta(\varepsilon)$ away from other $f_\theta(\varepsilon')$.

For the special case of the Gaussian kernel $k(x, y) = \exp(\frac{\|x-y\|^2}{2\sigma^2})$ and using the empirical estimator of the expectation with N samples, we have

$$\nabla_f \mathrm{MMD}^2\big|_{f=f_\theta(\varepsilon)} = -\frac{2}{N\sigma^2}\sum_{i=1}^{N} k(f_\theta(\varepsilon), f_\theta(\varepsilon_i))(f_\theta(\varepsilon) - f_\theta(\varepsilon_i)) + \frac{2}{N\sigma^2}\sum_{j=1}^{N} k(f_\theta(\varepsilon), x_j)(f_\theta(\varepsilon) - x_j), \tag{12}$$

which we can reformulate as

$$V^- = \frac{2}{N\sigma^2}\sum_{i=1}^{N} k(f_\theta(\varepsilon), f_\theta(\varepsilon_i))(f_\theta(\varepsilon_i) - f_\theta(\varepsilon)), \tag{13}$$

$$V^+ = \frac{2}{N\sigma^2}\sum_{j=1}^{N} k(f_\theta(\varepsilon), x_j)(x_j - f_\theta(\varepsilon)). \tag{14}$$

There are two main differences with these terms and the ones proposed in the original Drifting paper: First, obviously, the original kernel is not Gaussian but Laplacian, which means that the weights in front of $x_i - f_\theta(\varepsilon)$ (resp. $f_\theta(\varepsilon_i) - f_\theta(\varepsilon)$) decay much more rapidly close to the samples, but have more long range interaction. This may improve stability because there are more terms in the sum, but at the cost of a higher chance of memorizing perfectly training samples if f_θ happens to wander too close to one. Second, the proposed kernels are normalized such that the weights sum to one. This has the effect of always having some non-zero weights, even when the mapped samples are far from the training samples, which may greatly improve learning speed, but it is more tricky to formalize properly.

Another caveat worth mentioning is the factor $1/\sigma^2$ in front of both V^+ and V^- that has a dramatic effect on the calibration of the system. Indeed, one can be tempted to increase the value of σ such that long range interactions are more taken into account by the kernel. But doing so would decrease the norm of the correction field by the same factor and thus produce pseudo-target that are very close to the current displacement field, slowing the training. However, since this factor is constant, one could consider the equivalent problem where the loss is multiplied by σ^2 (*i.e.*, we seek to minimize $\sigma^2 D$ instead of D) which would make this problem disappear.

Lastly, MMD has been used in the past to train generative models by Li et al. (2015) and Dziugaite et al. (2015). The main difference with the drifting approach is that the later does not minimize the MMD directly, which would involve backpropagating through it, but instead use it only as a pseudo-target.

2.4 Drifting in feature space

When generating images, computing D directly in pixel space is not adequate because it does not encode images properties like translation equivariance or compositionality. Thus, we could leverage a feature space in which D operates. For the MMD, this amounts to choosing a kernel that combines properties of the images with properties of the distribution. Assuming an image feature encoder ϕ, such a composite kernel writes

$$\kappa(x, y) = k(\phi(x), \phi(y)) \,. \tag{15}$$

Then, the gradient of the MMD becomes

$$\begin{aligned} \nabla_f \mathrm{MMD}^2\big|_{f=f_\theta(\varepsilon)} = \frac{2}{N} J_\phi(f_\theta(\varepsilon))^\top \sum_{i=1}^{N} \nabla_u \kappa(u, \phi(f_\theta(\varepsilon_i)))\big|_{u=\phi(f_\theta(\varepsilon))} \\ - \frac{2}{M} J_\phi(f_\theta(\varepsilon))^\top \sum_{j=1}^{M} \nabla_u \kappa(u, \phi(x_j))\big|_{u=\phi(f_\theta(\varepsilon))} \end{aligned} \tag{16}$$

where J_ϕ is the Jacobian of ϕ and is use to project back into the input space of ϕ. With a Gaussian kernel for κ, this simplifies to

$$\begin{aligned} \nabla_f \mathrm{MMD}^2\big|_{f=f_\theta(\varepsilon)} = -\frac{2}{N\sigma^2} J_\phi(f_\theta(\varepsilon))^\top \sum_{i=1}^{N} \kappa(\phi(f_\theta(\varepsilon)) - \phi(f_\theta(\varepsilon_i)))\,[\phi(f_\theta(\varepsilon)) - \phi(f_\theta(\varepsilon_i))] \\ + \frac{2}{M\sigma^2} J_\phi(f_\theta(\varepsilon))^\top \sum_{j=1}^{M} \kappa(\phi(f_\theta(\varepsilon)) - \phi(x_j))\,[\phi(f_\theta(\varepsilon)) - \phi(x_j)] \,, \end{aligned} \tag{17}$$

of which the factor σ can be absorbed in the learning following the argument made in the previous section. In case $\phi = I$, then $J_\phi = I$ and the standard Gaussian case is recovered.

3 Experiments

3.1 Toy data

We conduct an experiment on toy data with a Gaussian being mapped to a checkerboard pattern alternating cells of uniform density and empty cells. The mapping f_θ is modeled by a 3-layer MLP optimized with Adam.

We show in Figure 1 the samples generated throughout training by our EM interpretation using the MMD with a Gaussian kernel, as well as the corresponding displacement field of f_θ and its correction term V. Similarly, we show in figure 2 the samples generated by the model trained using the original GMD equations and its corresponding displacement fields and correction.

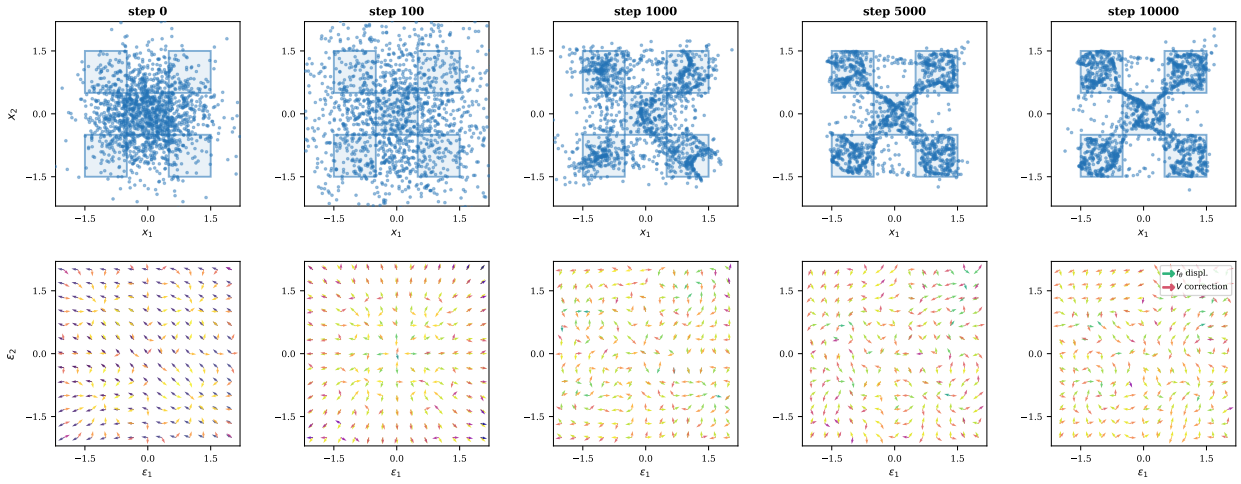

Figure 1: **Evolution of generated samples and drifting fields for the MMD based EM algorithm.** (top) Gaussian samples mapped by f_θ throughout training iterations, mapping a checkerboard pattern. (bottom) Displacement field of f_θ and its correction V throughout training iterations.

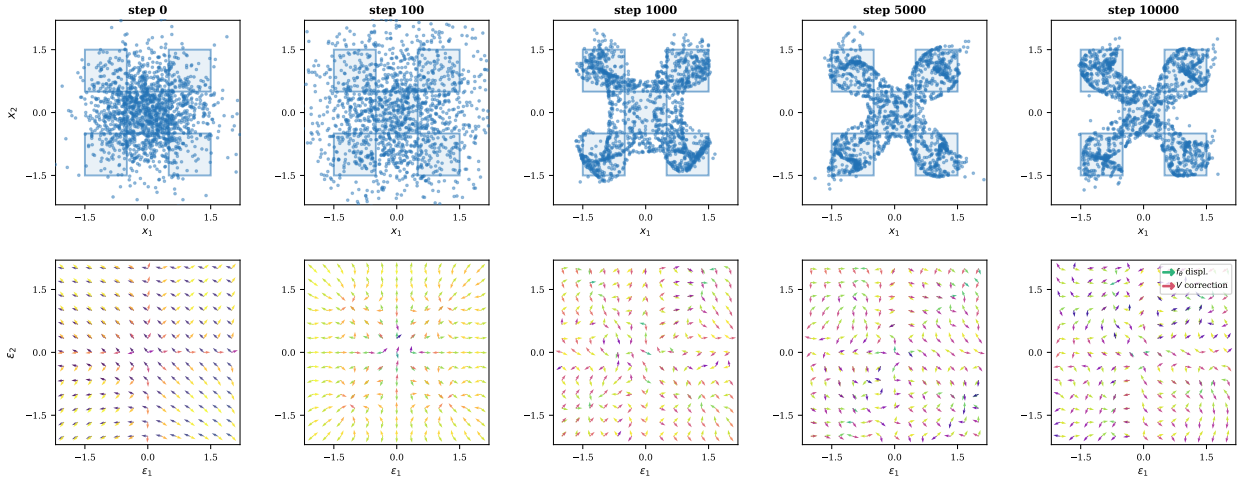

Figure 2: **Evolution of generated samples and drifting fields for the original drifting algorithm.** (top) Gaussian samples mapped by f_θ throughout training iterations, mapping a checkerboard pattern. (bottom) Displacement field of f_θ and its correction V throughout training iterations.

As we can see, both models are able to learn the general shape of the target distribution, albeit with some differences. It appears that the EM model is more *"square"* but leaks points between the modes, whereas the original GMD is less square but does not leak as many points between the modes. This is probably due to the normalization: for the EM model, it prevents from learning a good displacement fields for parts of the Gaussian that are far from data points since the MMD is very small at these locations. For the GMD model, it produces conflicting drifting vectors at locations where two or more modes are equidistant.

Regarding the displacement fields, they are very similar for both models, up to the initialization and optimization randomness. More specifically, they both have the same swirling pattern in the 4 corners, which we can hypothesize come from the limited Lipschitz capacity of the network (pointing toward the center of each cell would require a hard gradient change in the middle of these cells). The drifting field V mostly points towards the closest cell in empty regions, which is what one would expect from a well behaved system.

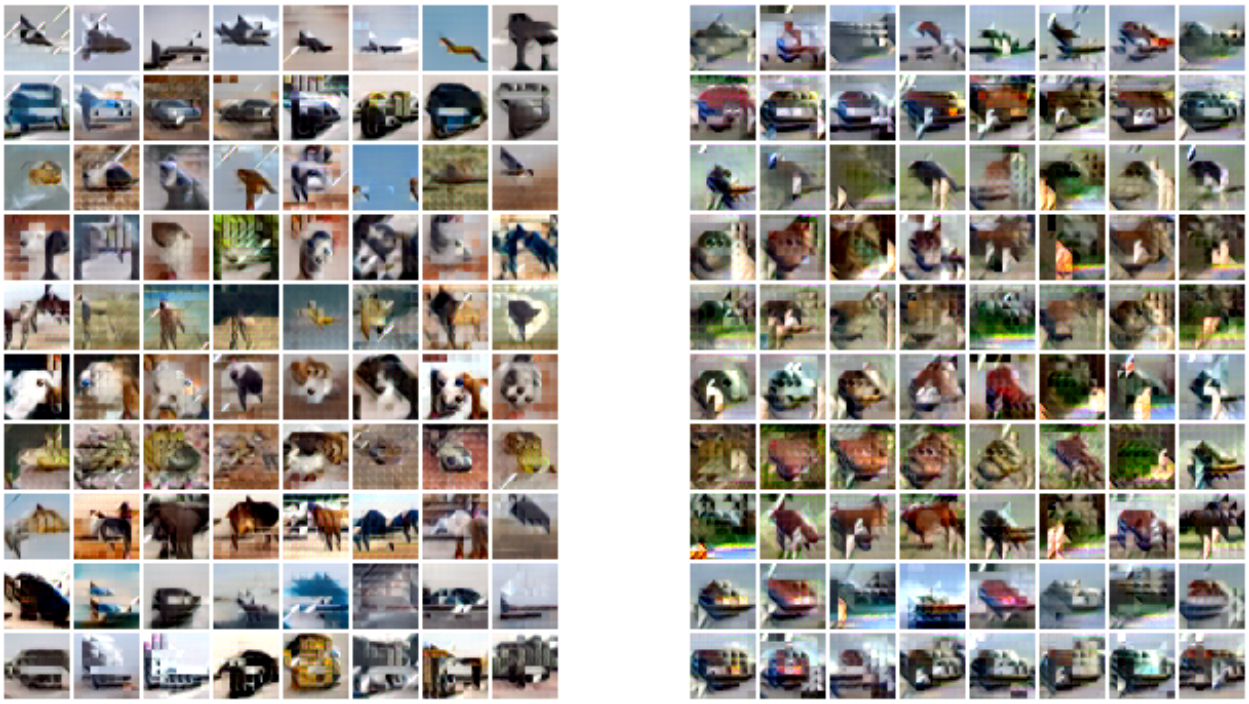

Figure 3: **Comparison between images generated with GMD (left) and MMD (right)**. Each row contains a class, in order: *airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck*. Left image reproduced from `https://github.com/tyfeld/drifting-model`.

3.2 CIFAR 10

We also conduct a small experiment on CIFAR10, using the ResNet feature extractor proposed in GMD. We train a small model for 200 epochs with the MMD loss according to the code provided in `https://github.com/tyfeld/drifting-model`.

We compare in Figure 3 the images from the repository (left) to the ones obtained with our loss (right). Images appears to be of similar quality, although some classes appears easier to distinguish depending on the method. This small experiment at least validates that the MMD can be used to derive a meaningful V in image feature space.

4 Discussion

The proposed EM interpretation of GMD leads to several questions that could provide interesting research directions. We list here some of our favorite ones.

Dynamic D In the proposed approach relying on the MMD with a Gaussian kernel, we saw that the bandwidth σ plays an important role. Set it to small and learning becomes impossible because mapped samples are too far away from data points, but set it too high and the precision of the drifting field becomes blurred by too many data points in the sum. Scheduling σ to be large earlier in training but sharp later could get the best of both world. More generally, one could study varying D throughout training such that at the beginning is had long range interaction to capture all the mapped samples, but is more concentrated at later stages to increase precision.

Non-differentiable D The choice of D seems to be crucial, yet we are limited to differentiable mappings. However, it seems reasonable to think that a D that performs some sort of assignment would be better at

allocating which region of the Gaussian should be mapped to a specific part of $\mathcal{P}_{\text{data}}$. Hausdorff and Chamfer distances come to mind, but Optimal Transport should also be considered.

Better estimator The empirical estimator used in Equation 7 requires a high number of samples to fully assess how $f_\theta \sharp \varepsilon$ and $\mathcal{P}$ differ over their respective support. In particular, ε has to be a fairly high dimensional Gaussian to have enough entropy to generate images[1]. Better estimation of V, either by clever sampling tricks, or by using more frugal estimator could speedup convergence and improve the resulting mapping.

Conditional mapping Conditional mapping are fairly easy to implement in terms of architecture, but they raise an interesting question with respect to the objective and the choice of D. One could just match the corresponding conditional distribution of $\mathcal{P}_{\text{data}}$, but the original GMD paper refers to a classifier free guidance approach that leads to incorporate real images from other classes into V^-. This raises the question of how to design a an objective function that minimizes D between the conditional distributions of $f_\theta \sharp \varepsilon$ and $\mathcal{P}_{\text{data}}$ for a matching condition while maximizing D for non-matching conditions.

Acknowledgments

The code for the experiments has been generated using Claude, then manually verified and corrected.

References

Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22 (14):e49–e57, 2006.

Mingyang Deng, He Li, Tianhong Li, Yilun Du, and Kaiming He. Generative modeling via drifting. *arXiv preprint arXiv:2602.04770*, 2026.

Gintare Karolina Dziugaite, Daniel M Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pp. 258–267, 2015.

Donald O. Hebb. *The organization of behavior. A neuropsychological theory.* John Wiley, 1949.

Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *International conference on machine learning*, pp. 1718–1727. PMLR, 2015.

Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton:(Project Para).* Cornell Aeronautical Laboratory, 1957.

Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[1] A back-of-the-envelope data processing inequality calculation indicates that if an image is a random variable with a covariance matrix of rank n, then the input Gaussian has to be at least of dimension n.