Learning Privacy-Preserving Deep Kernels with Known Demographics

Namrata Deka[1], **Danica J. Sutherland**[1, 2]

[1]University of British Columbia, [2]Amii
dnamrata@cs.ubc.ca, dsuth@cs.ubc.ca

Abstract

We propose a method for learning deep kernels that are incapable of distinguishing certain distributions, e.g. populations that differ only with respect to a sensitive attribute. Specifically, we find representations based on a simple kernel on top of a deep feature extraction network, and train them to simultaneously be capable of distinguishing certain pairs of input distributions but incapable of distinguishing others. We extend the previous literature on optimizing test power by refining existing unbiased estimators and using new tricks to effectively operate with small training batches. We evaluate our methods on image datasets where straightforwardly hiding these sensitive attributes from the model would be difficult, but our approach succeeds at removing them.

Introduction

Image-based machine learning systems, especially deep convolutional neural networks, are increasingly used for critical and sensitive real-life decisions. The importance of designing non-discriminatory learning algorithms that can mitigate various biases, like gender and race (among many others), is crucial to building trustworthy AI systems. This is especially challenging on unstructured high-dimensional domains like images where sensitive facial features like perceived gender, skin color/race, etc. cannot be isolated and specified directly as a separate input parameter to the model. To reliably use images for a variety of real-world tasks, we need methods that can learn not to exploit or depend on these sensitive attributes. To this end, we propose a kernel-based approach to preserve invariance to particular types of attributes in images, maintaining privacy not only of any features directly establishing the sensitive attribute, but also the possibility of reconstructing that attribute from other aspects of the input.

Ideally, if a learned system is fair with respect to a sensitive feature, then it will not be able to distinguish between distributions of images conditioned on different values of the sensitive feature, for instance images of dark-skinned or light-skinned faces. Using the framework of *two-sample testing*, statistical tests with our learned representation should not be able to distinguish between the two conditional distributions: the power of the test (the probability that such a test rejects the null hypothesis that the two distributions are identical) should be zero. At the same time, the system must be able to tell apart sets of images that differ with respect to the feature of interest for a downstream task: a test on this attribute should have high power. To achieve these simultaneous goals, we learn a single kernel parameterized by a deep neural network by maximizing the power of one test based on the maximum mean discrepancy (MMD; Gretton et al. 2012), while simultaneously minimizing the power of another test. To do this effectively and reliably while training with small batches, we improve existing unbiased estimates of the variance of the MMD estimator.

Preliminaries

Maximum Mean Discrepancy (MMD) The MMD (Gretton et al. 2012) is used to measure the distance between distributions. It is defined in terms of a kernel on individual elements (e.g. an image), $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, where $\mathcal{X}$ is the domain of the elements, e.g. the set of conceivable images. If $X, X' \sim \mathbb{P}$ and $Y, Y' \sim \mathbb{Q}$, where $\mathbb{P}, \mathbb{Q}$ are probability distributions on $\mathcal{X}$, then:

$$\operatorname{MMD}(\mathbb{P}, \mathbb{Q}; k) = \sqrt{\mathbb{E}[k(X, X') + k(Y, Y') - 2k(X, Y)]}.$$

With a characteristic kernel k, $\operatorname{MMD}(\mathbb{P}, \mathbb{Q}; k) = 0$ *if and only if* $\mathbb{P} = \mathbb{Q}$. While optimizing the test power objective, we will use the U-statistic estimator, which is unbiased for MMD^2 and has nearly minimal variance among unbiased estimators:

$$\widehat{\operatorname{MMD}}^2_{\mathrm{U}}(S_\mathbb{P}, S_\mathbb{Q}; k) = \frac{1}{m(m-1)} \sum_{i \neq j} H_{ij}$$

$$H_{ij} = k(X_i, X_j) + k(Y_i, Y_j) - k(X_i, Y_j) - k(Y_i, X_j),$$

where $S_\mathbb{P} = \{X_1, \ldots, X_m\}, S_\mathbb{Q} = \{Y_1, \ldots, Y_m\}$ are *i.i.d.* samples from $\mathbb{P}$ and $\mathbb{Q}$ respectively.[1]

Two-Sample Testing Based on *i.i.d.* samples $S_\mathbb{P}, S_\mathbb{Q}$ from $\mathbb{P}$ and $\mathbb{Q}$ respectively, the two-sample testing problem asks whether $S_\mathbb{P}, S_\mathbb{Q}$ come from the same distribution – that is, does $\mathbb{P} = \mathbb{Q}$? We use the null hypothesis testing framework, i.e. ask whether we can confidently say that the observed

[1]The MVUE would simply also include the $k(X_i, Y_i)$ terms; the difference in practice is trivial, but this form is slightly simpler and allows for exact expressions for the variance.

$S_\mathbb{P}, S_\mathbb{Q}$ would be unlikely to be so different if $\mathbb{P} = \mathbb{Q}$. Traditional methods include t-tests and Kolmogorov-Smirnov tests, but do not scale to complex high-dimensional distributions.

MMD-based tests with a kernel k reject the null hypothesis $\mathfrak{H}_0$ that $\mathbb{P} = \mathbb{Q}$ if the scaled estimator $m\,\widehat{\mathrm{MMD}}_\mathrm{U}^2(S_\mathbb{P}, S_\mathbb{Q}; k)$ is larger than the threshold c_α, where c_α is such that the scaled estimator will usually be smaller than c_α under the null hypothesis: $\Pr_{\mathfrak{H}_0}\left(m\,\widehat{\mathrm{MMD}}_\mathrm{U}^2(S_\mathbb{P}, S_\mathbb{Q}; k) > c_\alpha\right) \le \alpha$. (The estimate is scaled by m because, as m grows, this choice makes c_α converge to a value depending on $\mathbb{P} = \mathbb{Q}$ and k but independent of m.)

There are several options for estimating c_α, but the one usually currently considered best is *permutation testing*: randomly re-assign the sampled points to S', S'' and track $m\,\widehat{\mathrm{MMD}}_\mathrm{U}^2(S, S''; k)$ many times, then take the $(1-\alpha)$ quantile of that distribution (Sutherland et al. 2017).

Choosing a Powerful Kernel MMD tests work well when the choice of kernel k is appropriate, but for complicated distributions with reasonable numbers of samples, we usually need to choose an appropriate k rather than sticking to an *a priori* choice. In general, we wish to find the most powerful test: the one with the highest probability of correctly rejecting the null hypothesis when the alternative is true. This probability is asymptotically

$$\Pr_{\mathfrak{H}_1}\left(m\,\widehat{\mathrm{MMD}}_\mathrm{U}^2 > t_\alpha\right) \to \Phi\left(\frac{\mathrm{MMD}^2 - c_\alpha/m}{\sqrt{V_m}}\right), \quad (1)$$

where Φ is the CDF of a standard normal, and V_m is the variance of the $\widehat{\mathrm{MMD}}_\mathrm{U}^2$ estimator for samples of size m from $\mathbb{P}$ and $\mathbb{Q}$ with the kernel k (Sutherland et al. 2017, Equation 2). Note that none of the terms here are random: this is the probability of rejecting a random sample pair $S_\mathbb{P}, S_\mathbb{Q}$ in terms of m, k and the distributions. This formula comes from an asymptotic normality result for the estimator when $\mathbb{P} \ne \mathbb{Q}$ (Serfling 1980).

Previous work (Sutherland et al. 2017; Liu et al. 2020) has sought a powerful kernel k by maximizing an estimator of the right-hand side of (1) – or, rather, of its most important component $\mathrm{MMD}^2/\sqrt{V_m}$.

Deep Kernels We choose to look for the best kernel from a parameterized family of *deep kernels*. Specifically, we take k_ω as a Gaussian kernel κ on the output of a featurizer network ϕ_ω, $k_\omega = \kappa_\omega(\phi_\omega(x), \phi_\omega(y))$. Here, ϕ_ω is a deep neural network that extracts features from input images x, whose parameters are contained within ω, and κ_ω is a Gaussian kernel on those features whose lengthscale σ_ϕ is also contained in ω. These types of kernels have seen success across a variety of areas (e.g. Wilson et al. 2016; Li et al. 2017; Jean, Xie, and Ermon 2018). Liu et al. (2020) use stochastic gradient methods to choose ω maximizing an estimator of $\mathrm{MMD}^2/\sqrt{V_m}$; more on the V_m estimator shortly.

Privacy-Preserving Deep Kernels

Like prior work, our goal is to find parameters ω such that k_ω has high power at distinguishing target distributions; unlike prior work, we wish to do this while also being *unable* to distinguish sensitive distribution pairs (very low power).

Let $\mathbb{P}^a$ and $\mathbb{Q}^a$ be conditional distributions on a dataset that only differ by the conditioning value of a feature attribute a, and take corresponding sample sets $S_{\mathbb{P}^a}$, $S_{\mathbb{Q}^a}$. For instance, $\mathbb{P}^a$ might be the distribution of light-skinned faces from a particular data generating distribution, and $\mathbb{Q}^a$ that of dark-skinned faces.

To learn a kernel that can preserve privacy with respect to a sensitive attribute s while being able to perform well on downstream tasks dependant on a target attribute t, we use a multi-objective optimization paradigm to simultaneously minimize the hypothesis test power with respect to s and maximize the test power with respect to t. Our optimization problem is therefore

$$\operatorname*{argmin}_\omega \Phi\left(\frac{\widehat{\mathrm{MMD}}_\mathrm{U}^{2\;s} - \frac{\hat{c}_\alpha^s}{m}}{\widehat{\mathrm{V}}_m^s}\right) - \Phi\left(\frac{\widehat{\mathrm{MMD}}_\mathrm{U}^{2\;t} - \frac{\hat{c}_\alpha^t}{m}}{\widehat{\mathrm{V}}_m^t}\right), \quad (2)$$

and where each of $\widehat{\mathrm{MMD}}_\mathrm{U}^{2\;a}$, $\hat{c}_\alpha^a$, and $\widehat{\mathrm{V}}_m^a$ denote the relevant quantity for $S_{\mathbb{P}^a}, S_{\mathbb{Q}^a}$ with the kernel k_ω for brevity. We now discuss the choice of estimator $\widehat{\mathrm{V}}_m$.

Unbiased Estimator of the MMD Variance To estimate the power of the hypothesis test in (2), we need a reliable estimate of the variance of $\widehat{\mathrm{MMD}}_\mathrm{U}^2$ with m samples.

Using general results about U-statistics (Serfling 1980), this variance is precisely $\frac{4(m-2)}{m(m-1)}\zeta_1 + \frac{2}{m(m-1)}\zeta_2$, where ζ_1, ζ_2 depend on $\mathbb{P}$, $\mathbb{Q}$, and k but not m. Bounliphone et al. (2016) used the theory of MMD to evaluate the population value of the higher-order term ζ_1, and proposed an estimator for that quantity computable in the same quadratic time it takes to compute $\widehat{\mathrm{MMD}}_\mathrm{U}^2$. Sutherland et al. (2017) corrected some accidental sources of statistical bias, and also estimated ζ_2 (in the same time complexity). Their correction, however, was also mistakenly still slightly biased; Sutherland (2019) corrected this, giving an unbiased estimator for the exact variance – except her estimator was *again* mistaken (this time due to a simple typo). We will also improve the estimator in another way shortly.

Liu et al. (2020) instead used only the leading term $4\zeta_1/m$ in their variance estimator and an intentionally biased estimator of ζ_1, which is much simpler to derive and implement (without, to our knowledge, a long line of mistaken derivations). When training with large batch sizes as they did, this bias is small, and overall much simpler. In our settings, however, we found notably better performance by using the (correct) unbiased estimator; we are forced to use fairly small batch sizes in some contexts, and minimizing the power might also behave differently than maximizing it.

The power of a test depends on how many samples m it uses, intuition which can be confirmed from (1). In the objective (2), the choice of m in $\hat{c}_\alpha/m$ and $\widehat{\mathrm{V}}_m$ should be the

m at which we want to conduct our final test.[2] This does not mean, however, that we need to estimate these population quantities based on m samples. Instead, we can estimate them based on a batch of n samples, e.g. if we wish to run minibatch optimizers and then conduct our final tests with larger sample sizes. We thus also allow for $n \neq m$ in our version of the $\widehat{\mathrm{V}}_m$ estimator, which we now present.

Define the $n \times n$ matrix $\mathbf{K_{XY}}$ by $(\mathbf{K_{XY}})_{ij} = k(X_i, Y_j)$, and $\mathbf{K_{XX}}, \mathbf{K_{YY}}$ similarly. Let $\tilde{\mathbf{K}}_\mathbf{XX}, \tilde{\mathbf{K}}_\mathbf{YY}$ be $\mathbf{K_{XX}}, \mathbf{K_{YY}}$ with their diagonals set to zero and $\mathbf{1}$ be the n-vector of all ones. We will also use falling factorial notation $(n)_k = n(n-1)\cdots(n-k+1)$. Then, our estimator $\widehat{\mathrm{V}}_m$ based on n samples, which is unbiased ($\mathbb{E}\,\widehat{\mathrm{V}}_m = V_m$), is

$$\begin{aligned}
&\frac{4(mn+m-2n)}{(m)_2(n)_4}\left[\left\|\tilde{\mathbf{K}}_\mathbf{XX}\mathbf{1}\right\|^2 + \left\|\tilde{\mathbf{K}}_\mathbf{YY}\mathbf{1}\right\|^2\right] \\
&- \frac{2(2m-n)}{mn(m-1)(n-2)(n-3)}\left[\left\|\tilde{\mathbf{K}}_\mathbf{XX}\right\|_F^2 + \left\|\tilde{\mathbf{K}}_\mathbf{YY}\right\|_F^2\right] \\
&+ \frac{4(mn+m-2n-1)}{(m)_2 n^2(n-1)^2}\left[\left\|\mathbf{K_{XY}}\mathbf{1}\right\|^2 + \left\|\mathbf{K_{XY}}^\mathsf{T}\mathbf{1}\right\|^2\right] \\
&- \frac{4(2m-n-2)}{(m)_2 n(n-1)^2}\left\|\mathbf{K_{XY}}\right\|_F^2 \\
&- \frac{2(2m-3)}{(m)_2(n)_4}\left[(\mathbf{1}^\mathsf{T}\tilde{\mathbf{K}}_\mathbf{XX}\mathbf{1})^2 + (\mathbf{1}^\mathsf{T}\tilde{\mathbf{K}}_\mathbf{YY}\mathbf{1})^2\right] \\
&- \frac{4(2m-3)}{(m)_2 n^2(n-1)^2}(\mathbf{1}^\mathsf{T}\mathbf{K_{XY}}\mathbf{1})^2 \\
&- \frac{8}{m(n)_3}\left[\mathbf{1}^\mathsf{T}\tilde{\mathbf{K}}_\mathbf{XX}\mathbf{K_{XY}}\mathbf{1} + \mathbf{1}^\mathsf{T}\tilde{\mathbf{K}}_\mathbf{YY}\mathbf{K_{XY}}^\mathsf{T}\mathbf{1}\right] \\
&+ \frac{8}{mn(n)_3}\left[(\mathbf{1}^\mathsf{T}\tilde{\mathbf{K}}_\mathbf{XX}\mathbf{1} + \mathbf{1}^\mathsf{T}\tilde{\mathbf{K}}_\mathbf{YY}\mathbf{1})(\mathbf{1}^\mathsf{T}\mathbf{K_{XY}}\mathbf{1})\right].
\end{aligned}$$

Although the expression is perhaps intimidating, it only requires taking additional sums of the same kernel matrices we already needed to compute for the MMD estimate, as well as a few extra vector-vector products. It is also amenable to automatic differentiation. The derivation of this estimator uses the same techniques as the earlier approaches, but, compared to Sutherland (2019), allows for $m \neq n$ and fixes a mistake in the order of a leading term. Because this estimator can be negative, in practice we regularize by taking the maximum of it and a small positive λ.

Moving Average of the Variance The estimator $\widehat{\mathrm{V}}_m$ is fairly noisy with small batch size n; for kernels where the value V_m is small, the division by $\widehat{\mathrm{V}}_m$ in the denominators of (2) is likely to exacerbate that noise. We thus reduce the variance by keeping an exponentially weighted moving average of $\widehat{\mathrm{V}}_m$ computed on training minibatches, and using that in our objective. The data points feeding into the estimate are the same as if we didn't use the moving average, but it helps place us in the right part of the $1/\sqrt{x}$ function, which can dramatically improve practical performance with smaller batch sizes.

[2]Because the Gaussian kernel is characteristic, it would only be possible to achieve exactly zero power even for extremely large m with a ϕ_ω that is perfectly invariant to the sensitive attribute s; even if that were feasible, using $m \to \infty$ in (2) would destroy the gradient signal to reach that point, since the power would remain at 1 with any local modification.

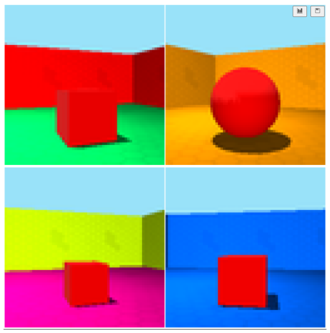
Gradient of the Threshold Estimate Unlike previous work in this area, it is important to our applications to also have a differentiable estimate of $\hat{c}_\alpha$: note that simply disregarding it would lead to a power estimator that almost cannot return a number less than 0.5, perhaps not a huge problem when maximizing power but certainly concerning when minimizing it. We use the permutation estimate mentioned earlier and described in detail by Sutherland et al. (2017), although we implement it in PyTorch based on matrix multiplication rather than the direct C++ implementation of that work. Automatic differentiation through that process gives a reasonable estimate of the gradient of c_α.

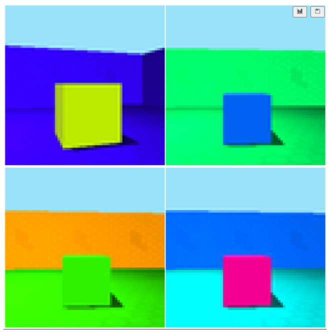
Experiments

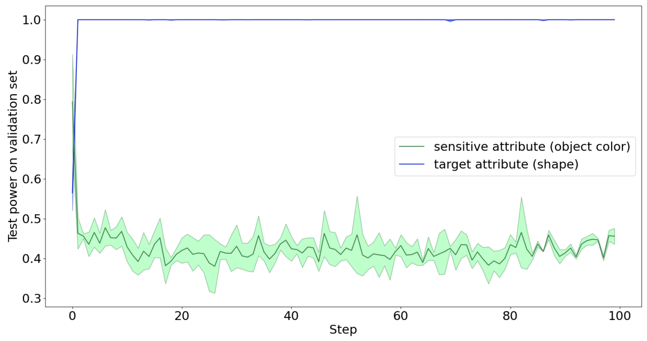
3DShapes Dataset 3Dshapes (Burgess and Kim 2018) is a dataset of images of 3D shapes generated from six ground truth independent latent factors: shape, scale, orientation, object color, wall color and floor color. The dataset contains all possible combinations of these latent factors, giving a total of 480,000 images. For our experiment, we first split the dataset randomly into train, test and val sets with a ratio of $80 : 10 : 10$ samples. We pick one of the six latent factors to be the sensitive attribute s and another to be the target attribute t. To build the sample sets, we choose two of the possible discrete values of s, s_1 and s_2; we sample $S_{\mathbb{P}^s}$ uniformly from the portion of the dataset with $s = s_1$, and $S_{\mathbb{Q}^s}$ from points with $s = s_2$. $S_{\mathbb{P}^t}$ and $S_{\mathbb{Q}^t}$ are similarly taken from $t = t_1$ and $t = t_2$, respectively. Figure 1 shows a few samples from these conditional distributions.

Training Details Our deep neural network contains a convolutional image featurizer with five convolutional layers, interspersed with LeakyReLU activations and batch normalization layers. The features extracted by the convolutional network are flattened and passed into a fully connected layer that learns a 300-dimensional representation of the input image. We use the Adam (Kingma and Ba 2017) optimizer with a batch size of 64 images per conditional distribution: images from $S_{\mathbb{P}^a}$ and $S_{\mathbb{Q}^a}$ are concatenated to form a total batch size of 128. The output features from the network are fed to a simple Gaussian kernel whose length scale σ_ϕ is also a part of the network's learnable parameters. The learning rate and coefficient of the moving average for the variance of the MMD estimator are tuned using Bayesian hyperparameter optimization (Snoek, Larochelle, and Adams 2012) on a validation set to minimize the power loss (2). All models were trained thrice with different initializing random seeds.

Results Figure 1c shows the estimate of (1) for the target attribute (shape) and sensitive attribute (color), tracked through the course of training. The target attribute quickly shoots to essentially perfect power, while the sensitive attribute decreases to around 0.4 to 0.5 power. In this case, we have notably decreased the estimated power of the kernel on the sensitive attribute, but were not able to drive it all the way to zero.


(a) Samples from the conditional `object color = red`.


(b) Samples from the conditional `shape = square`.



(c) Validation curves of test power (1) for the target and sensitive attributes during training.

Figure 1: Training a kernel to distinguish shape while ignoring color on Shapes3D.

Attribute	Power est.	Emp. power	SVM acc.
Shape (t)	1.000	0.7181	0.8953
Object Color (s)	0.4137	0.0478	0.5057
Wall Color	–	–	0.4912
Scale	–	–	0.6852

(a) Targeting shape, with object color sensitive.

Attribute	Power est.	Emp. power	SVM acc.
Scale (t)	1.000	0.8333	0.9172
Orientation (s)	0.4464	0.0638	0.5248
Wall color	–	–	0.7869
Shape	–	–	0.7804

(b) Targeting scale, with orientation sensitive.

Attribute	Power est.	Emp. power	SVM acc.
Shape	0.5000	1.000	0.9492
Object Color	0.5000	1.000	0.9498
Scale	0.5000	1.000	0.9495
Orientation	0.5000	1.000	0.9529

(c) A simple gaussian kernel on image pixels.

Table 1: Ability of different kernels, two trained by our method and one fixed, to distinguish various attributes.

Table 1a shows results of the final kernel on the test set, both for the estimated power (1) and the empirical power (the portion of times the test rejects the null hypothesis). Table 1b shows results for a different attribute pair, training with scale as the target attribute and orientation sensitive; the analogue of Fig. 1c looks very similar.

Using the Kernel Downstream We further demonstrate that our learned kernel has the desired empirical privacy properties by using it to train a support vector machine (SVM). Table 1 also shows the mean test accuracy of an SVM with the given kernel when cross-validating on a test set held out during kernel selection. The trained kernels are able to classify their target attribute, but only barely able to classify the sensitive attribute. (If the MMD is small between two distributions, then a kernel classifier to distinguish those distributions will necessarily have large norm.)

We also use the learned kernels to train an SVM to classify other attributes the kernel was not trained on and observe that the performance there lies between the accuracies of the target and sensitive attributes. This indicates that the kernels learned by our method is able to concentrate most on the desired attributes and least on the sensitive attributes while still being able to exploit the remaining features in the images.

Comparison to a Gaussian Kernel Table 1c additionally shows the same results for a simple Gaussian kernel applied directly on the image pixel values. It is reasonably able to distinguish all attributes both in terms of an SVM classifier accuracy as well as empirical test power indicating that “hiding” the sensitive attribute from a classifier was a nontrivial task and learning a kernel to do so is effective in training a fair classifier.

Discussion

We proposed a method to learn a kernel-based representation of a dataset that is insensitive to certain sensitive attributes. This problem is closely related to the idea of “fair” representation learning, but compared to the dominant approaches in that area based on e.g. the Variational Fair Autoencoder (Louizos et al. 2016), our approach based on two-sample testing avoids the need for generative modeling; in our experience the power criterion (with our unbiased estimator and moving average estimate) is easier to effectively minimize than the direct MMD term used in that work. Future work will focus on expanding our understanding of this method, such as further studying options for handling several values of given attributes, and of course studying more complex datasets, potentially also with complex dependence patterns between attributes.

Acknowledgements

This research was enabled in part by support provided by the Canada CIFAR AI Chairs program, WestGrid, Compute Ontario, and Compute Canada.

References

Bounliphone, W.; Belilovsky, E.; Blaschko, M. B.; Antonoglou, I.; and Gretton, A. 2016. A Test of Relative Similarity For Model Selection in Generative Models. In *ICLR*.

Burgess, C.; and Kim, H. 2018. 3D Shapes Dataset. https://github.com/deepmind/3dshapes-dataset/.

Gretton, A.; Borgwardt, K. M.; Rasch, M. J.; Schölkopf, B.; and Smola, A. 2012. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1): 723–773.

Jean, N.; Xie, S. M.; and Ermon, S. 2018. Semi-supervised Deep Kernel Learning: Regression with Unlabeled Data by Minimizing Predictive Variance. In *NeurIPS*.

Kingma, D. P.; and Ba, J. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980.

Li, C.-L.; Chang, W.-C.; Cheng, Y.; Yang, Y.; and Póczos, B. 2017. MMD GAN: Towards Deeper Understanding of Moment Matching Network. In *NeurIPS*.

Liu, F.; Xu, W.; Lu, J.; Zhang, G.; Gretton, A.; and Sutherland, D. J. 2020. Learning deep kernels for non-parametric two-sample tests. In *International Conference on Machine Learning*, 6316–6326. PMLR.

Louizos, C.; Swersky, K.; Li, Y.; Welling, M.; and Zemel, R. S. 2016. The Variational Fair Autoencoder. In *ICLR*.

Serfling, R. 1980. *Approximation Theorems of Mathematical Statistics*.

Snoek, J.; Larochelle, H.; and Adams, R. P. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. In *NIPS*.

Sutherland, D. J. 2019. Unbiased estimators for the variance of MMD estimators. *ArXiv*, abs/1906.02104.

Sutherland, D. J.; Tung, H.-Y. F.; Strathmann, H.; De, S.; Ramdas, A.; Smola, A.; and Gretton, A. 2017. Generative Models and Model Criticism via Optimized Maximum Mean Discrepancy. *ArXiv*, abs/1611.04488.

Wilson, A. G.; Hu, Z.; Salakhutdinov, R.; and Xing, E. P. 2016. Deep kernel learning. In *AISTATS*.