FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of master's thesis

Title:	The use of Relative Goodness of Fit Tests for training Generative Adversarial Networks
Student:	Bc. Martin Scheubrein
Supervisor:	Ing. Daniel Vašata, Ph.D.
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2022/2023

Instructions

When training a GAN one often faces the problem of determining the quality of the intermediate and final results. Since the aim of the generative model is to reproduce the distribution typically in a high-dimensional space it is hard to confirm or reject that the output is of sufficient quality. The relative goodness of fit statistical tests may help to solve this issue. The recent tests based on the maximum mean discrepancy (MMD) and on the unnormalized mean embeddings (UME) are suitable for high-dimensional data.

The goal of this thesis is to research the suitability of the above-mentioned tests to improve GANs training. First, the focus should be on the basic properties of the tests and their ability to detect systematic changes in high-dimensional distribution like e.g rotations of images. Second, their use for tracking the quality of the training process and possibly as an early stopping criterion should be analyzed. The experiments should be taken on at least two datasets.

Electronically approved by Ing. Karel Klouda, Ph.D. on 11 February 2021 in Prague.

CZECH TECHNICAL
UNIVERSITY
IN PRAGUE

FIT

Faculty of Information Technology
Department of Applied Mathematics

Master's Thesis

The use of Relative Goodness of Fit Tests for training Generative Adversarial Networks

Martin Scheubrein

June 2021
Supervisor: Ing. Daniel Vašata, Ph.D.

Acknowledgement / Declaration

Děkuji všem, kteří nade mnou na této dlouhé a strastiplné cestě nezlomili hůl, zejména svému vedoucímu Ing. Danielu Vašatovi, Ph.D., rodině a přátelům.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 27. 6. 2021

.....

Abstrakt / Abstract

Generativní adversariální sítě (GAN) jsou třídou metod hlubokého učení většinou pracující s obrázky nebo jinými vysokodimenzionálními daty. U takových dat je pak obtížné rozhodnout, zda se distribuce naučená modelem shoduje s distribucí zdrojových dat, případně v jakých místech se liší. K měření těchto odlišností lze využít míry maximum mean discrepancy (MMD) nebo unnormalized mean embedding (UME).

Tato práce ověřuje, že obě míry při správné parametrizaci spolehlivě detekují globální i lokální odlišnosti distribucí obrazových dat. Detailně jsou zkoumány možnosti výběru kernelu, jeho parametrů, a v případě UME volba testových lokací. Je ověřena interpretovatelnost optimalizovaných testových lokací v kontextu odhalování lokálních odchylek mezi distribucemi.

V závěru je navržena nová metoda early stoppingu trénování GAN založená na měření MMD a UME mezi výstupy sítě a testovacími daty.

Klíčová slova: Test dobré shody; Maximum mean discrepancy; Unnormalized mean embedding; Generativní adversariální síť

Generative adversarial networks (GAN) are a class of deep learning methods which are usually applied to images or other high-dimensional data. With such data, it is difficult to decide if the distribution learnt by a model matches the distribution of source data, or to locate the differences. To measure those discrepancies, maximum mean discrepancy (MMD) or unnormalized mean embedding (UME) measures may be used.

This thesis verifies that with proper parametrization, both measures reliably detect both global and local discrepancies in image data. Choice of kernel, its parameters, and in the case of UME the selection of test locations, are studied in detail. Interpretability of optimized test locations in the context of local difference discovery is verified.

Finally, a novel method of early stopping based on MMD and UME measured between the network's output and testing data is proposed.

Keywords: Goodness of fit test; Maximum mean discrepancy; Unnormalized mean embedding; Generative adversarial network

Contents /

Figures /

Chapter 1
Introduction

Generative adversarial networks (GAN) are an intensively studied and developed field of study in deep learning. GAN is a type of artificial neural network which that is trained in an unsupervised manner to transform some simple apriori distribution (usually multidimensional standard normal distribution or uniform distribution) to a complex, high-dimensional target distribution.

The most common type of data representing the target distribution by GAN are images or photographs. The usual vector representation of black and white images is created by encoding light intensity of each pixel as a single element of some long $\mathbb{R}^n$ vector. We can see that such vectors are high-dimensional even for small images. Due to high dimensionality, it is quite difficult to decide if the distribution learnt by GAN matches the distribution of source data, as well as to highlight the areas of the vector space which show the most pronounced differences between the two distributions.

The GAN consists of two main components – a generator responsible for the transformation to target distribution, and a discriminator, which tries to distinguish real data from the output of the generator. Both components are realized by neural networks which are trained at one, alternating the training steps between one and another. Due to this complicated training method, it may not be obvious how well (or if) the GAN managed to learn the transformation. The loss functions of generator or discriminator may not be meaningful enough on their own, because improvement of one of the components counteracts the accuracy of another. A more robust way to assess if the network is already sufficiently trained is to measure the similarity of its output with known data directly.

To measure the discrepancies between the training data and the distribution learnt by the GAN on both global and local scale, we can utilize maximum mean discrepancy (MMD) [1] or unnormalized mean embedding (UME) [2]. Both measures have been designed to work well for high dimensional data, utilizing a kernel function chosen by the user. In case of UME, a reasonable set of so-called test locations must be supplied. The distributions are then compared from the point of view of all test locations, so their appropriate position is crucial in order to capture important local discrepancies as well as the global picture. MMD has no such requirement, but comes with a drawback, which is its quadratic computation time with respect to the number of samples, as opposed to linear computation time of UME.

Two statistical goodness of fit tests have been designed for a three-way comparison of distributions, which can be applied in model criticism as follows. Considering some source data, two different generative models may be trained to learn the distribution of the source data. The goodness of fit test then compares the output generated by both candidate models and decides, which one of them learned the source distribution better. One test is based on empirical estimate of MMD and the other one on UME.

It has been shown by the authors of the UME test that UME test locations can be optimized by maximizing the expected test power. Optimized test locations not only lead to more powerful test, as verified in experiments in this thesis, but optimized positions of the test locations can also be interpreted as regions, where one model significantly outperforms the other. This can be extremely useful for machine learning specialists to detect local problems in training, e.g. mode collapse.

This thesis experiments heavily with various steps needed for deployment of UME test, specifically the choice of kernel function and optimization procedure for UME test locations. It has been concluded that Gaussian kernel chosen by default by many researchers must be correctly set up, and a rational quadratic kernel may be a better alternative in some cases. Another conclusion is that for complex data such as images, the optimizer algorithm has to be very robust, and Adam optimizer is proposed to be used instead of L-BFGS-B used by the authors of the UME test.

Experiments with GANs trained on well-known image datasets MNIST and CelebA are conducted, which show that most observations made on smaller toy datasets also hold in more complex situations. A novel method of early stopping is proposed, which utilizes p-value of the goodness of fit test constructed such that its value corresponds to the indistinguishability of model output from real data.

Chapter 2
Discrepancy Measures

In this chapter, two random metrics will be introduced: maximum mean discrepancy (MMD) and unnormalized mean embedding (UME). Both of them are able to measure the amount of difference between two probability distributions using their mean embeddings. Thanks to their reliance on a kernel function, they are quite robust to high dimensionality of data sampled from the distribution, which is advantageous for our use case of GAN training, since the most common data type of GANs are images. Black and white images are naturally represented by real-valued matrices with their elements corresponding to individual pixel values, colour images are represented by order-three tensors, the last component thereof having the interpretation of colour channels (most commonly red, green and blue channels for RGB colour encoding). It's obvious that dimensionality of such data may grow very large when using high resolution images.

After the discrepancy measures are defined, two statistical tests of relative goodness of fit will be described, one based on maximum mean discrepancy, the other on unnormalized mean embedding. The tests can be used for model criticism, i.e. deciding, which of some two compared generative models fits ground-truth data better. Unlike other methods based on comparison of some scores or distances computed between model output and training data, in our case a full statistical framework is obtained, including statistical significance of result.

2.1 Maximum Mean Discrepancy

Let $\mathcal{X}$ be the domain of observations $x_1, \ldots, x_n \in \mathcal{X}$ and let $\mathcal{H}$ be a Hilbert space of functions $\mathcal{X} \to \mathbb{R}$ equipped with dot product $\langle \cdot, \cdot \rangle$ satisfying the reproducing property

$$\langle f, k(x, \cdot) \rangle = f(x)$$

for some kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Then, $\mathcal{H}$ is said to be a *reproducing kernel Hilbert space* (RKHS) [3].

Expression $k(x, \cdot)$ may be interpreted as a *feature map* $x \to \phi(x)$ mapping observations to a feature space. Thanks to the reproducing property, kernel k can be written as $k(x, y) = \langle \phi(x), \phi(y) \rangle$.

Let P be a Borel probability measure on $\mathcal{X}$. *Mean embedding* (also *mean map*) is the mapping

$$\mu_P := \mathrm{E}_{x \sim P}\, k(x, \cdot) = \mathrm{E}_x\, \phi(x)$$

and its empirical form

$$\hat{\mu}_P := \frac{1}{m}\sum_{i=1}^{m} k(x_i, \cdot) = \frac{1}{m}\sum_{i=1}^{m} \phi(x_i)$$

with $x_1, \ldots, x_m$ drawn independently and identically distributed from P.

Mean embeddings can be used to measure distance between two distributions. *Maximum mean discrepancy* (MMD) is such distance, and is defined using distance of their mean embeddings in a RKHS $\mathcal{H}$ as follows [1]. Let $\mathcal{F}$ be a unit ball in $\mathcal{H}$, i.e. $\mathcal{F} = \{\|f\| \leq 1 \mid f \in \mathcal{H}\}$, and P, Q two distributions on common domain. MMD is defined as

$$\begin{aligned}\mathrm{MMD}(P,Q) &:= \|\mu_P - \mu_Q\|_{\mathcal{H}} \\ &= \sup_{f\in\mathcal{F}} \langle \mu_P - \mu_Q, f\rangle_{\mathcal{H}} \\ &= \sup_{f\in\mathcal{F}} \big(\mathrm{E}_{x\sim P}\, f(x) - \mathrm{E}_{y\sim Q}\, f(y)\big).\end{aligned}$$

The function f^* which attains the supremum in above equations is called *witness function* and can be computed as

$$\begin{aligned}f^*(t) &:= \mathrm{E}_{x\sim P}\, k(x,t) - \mathrm{E}_{y\sim Q}\, k(y,t) = \mu_P(t) - \mu_Q(t) \\ \hat{f}^*(t) &:= \frac{1}{m}\sum_{i=1}^{m} k(x_i,t) + \frac{1}{n}\sum_{i=1}^{n} k(y_i,t).\end{aligned}$$

Samples $x_1, \ldots, x_m$ and $y_1, \ldots, y_n$ are being drawed from P and Q respectively. To clear ambiguity, witness function will be denoted $\mathrm{wit}_{P,Q} \equiv f^*$ in further text. Witness function is useful for determining the areas of $\mathcal{X}$, where P closely fits Q, as shown by [4]. In the context of GAN training, its outputs may be treated as samples from one distribution and training data as samples of another distribution. Witness function provides a powerful tool for determining weaknesses of the model, detecting mode collapse etc.

MMD is more commonly used in its squared form $\mathrm{MMD}^2(P,Q) = \big(\mathrm{MMD}(P,Q)\big)^2$. Squared MMD can be further expanded:

$$\begin{aligned}\mathrm{MMD}^2(P,Q) &:= \|\mu_P - \mu_Q\|_{\mathcal{F}} \\ &= \langle \mu_P, \mu_P\rangle_{\mathcal{F}} + \langle \mu_Q, \mu_Q\rangle_{\mathcal{F}} - 2\langle \mu_P, \mu_Q\rangle_{\mathcal{F}} \\ &= \mathrm{E}_{x,x'\sim P}\, k(x,x') + \mathrm{E}_{y,y'\sim Q}\, k(y,y') - 2\mathrm{E}_{x\sim P, y\sim Q}\, k(x,y).\end{aligned}$$

Declaring $X \sim P, Y \sim Q$, an unbiased empirical estimate of MMD^2 is

$$\begin{aligned}\widehat{\mathrm{MMD}}^2(X,Y) := &\frac{1}{m(m-1)}\sum_{i=1}^{m}\sum_{j\neq i}^{m} k(x_i,x_j) + \frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j\neq i}^{m} k(y_i,y_j) \\ &- \frac{2}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} k(x_i,y_j).\end{aligned}$$

The estimate is completely expressed in terms of the kernel function k and can be computed in asymptotically quadratic time with respect to number of samples m and n due to the double sums.

2.2 Relative MMD Test

Maximum mean discrepancy will be used in a relative similarity test introduced by [5]. Having samples from two different candidate distributions P, Q, the test determines which one of them is closer to the reference distribution R. It does so using MMD measured between the reference distribution and one of the candidate distributions, and comparing that with MMD between reference and the other candidate.

The test is based on the following theorem. Let $X_P \sim P, X_Q \sim Q, X_R \sim R$ Assuming $P \neq R$, $Q \neq R$, and finite expected values of the kernel function associated with MMD computation evaluated at any points sampled from P and/or Q, it is shown that

$$\sqrt{m}\left(\begin{pmatrix}\widehat{\mathrm{MMD}}^2(X_R, X_P)\\ \widehat{\mathrm{MMD}}^2(X_R, X_Q)\end{pmatrix} - \begin{pmatrix}\mathrm{MMD}^2(R, P)\\ \mathrm{MMD}^2(R, Q)\end{pmatrix}\right) \xrightarrow{\mathrm{D}} \mathcal{N}\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}\sigma_P^2 & \sigma_{PQ}\\ \sigma_{PQ} & \sigma_Q^2\end{pmatrix}\right),$$

where m is the size of X_R.

Based on empirical estimates of $\widehat{\mathrm{MMD}}^2(X_R, X_P)$ and $\widehat{\mathrm{MMD}}^2(X_R, X_Q)$, the relative similarity test decides, whether P is closer to reference R than Q is. More precisely

$$\mathcal{H}_0 : \mathrm{MMD}(R, P) \leq \mathrm{MMD}(R, Q),$$
$$\mathcal{H}_1 : \mathrm{MMD}(R, P) > \mathrm{MMD}(R, Q).$$

By projecting the joint distribution above from a $\frac{\pi}{4}$ angle, we obtain a test statistic $\widehat{\mathrm{MMD}}^2(X_R, X_P) - \widehat{\mathrm{MMD}}^2(X_R, X_Q)$. The p-value of the test is

$$p \leq \Phi\left(-\frac{\widehat{\mathrm{MMD}}^2(X_R, X_P) - \widehat{\mathrm{MMD}}^2(X_R, X_Q)}{\sqrt{\hat{\sigma}_P^2 - 2\hat{\sigma}_{PQ} + \hat{\sigma}_Q^2}}\right),$$

Φ being the cumulative distribution function of the standard normal distribution. For details on how to compute the estimations $\hat{\sigma}_P^2$, $\hat{\sigma}_{PQ}$, $\hat{\sigma}_Q^2$ of the elements of the covariance matrix, please refer directly to [5].

2.3 Unnormalized Mean Embedding

Unnormalized mean Embedding (UME) has been proposed by [2] for linear-time non-parametric two-sample testing. Let μ_P, μ_Q be mean embeddings on probability measures P, Q with kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which is analytic, i.e. $k(x, \cdot)$ is an analytic function for all choices of $x \in \mathcal{X}$, integrable, and characteristic, i.e. μ_P, μ_Q are injective on their respective probability measures. Squared UME is defined as

$$\mathrm{UME}^2(P, Q) := \frac{1}{J}\sum_{j=1}^{J}\big(\mu_P(v_j) - \mu_Q(v_j)\big)^2$$

with $V = \{v_1, \ldots v_J\}$ being unique *test locations*. The choice of test locations will be studied further in this thesis. Notice that UME^2 is an average of squared witness functions evaluated at the test locations.

A consistent estimate of UME^2 is

$$\widehat{\mathrm{UME}}^2 := \frac{1}{J}\sum_{j=1}^{J}\big(\hat{\mu}_P(v_j) - \hat{\mu}_Q(v_j)\big)^2 = \frac{1}{J}\sum_{j=1}^{J}\left(\frac{1}{m}\sum_{i=1}^{m}k(x_i, v_j) - \frac{1}{n}\sum_{i=1}^{n}k(y_i, v_j)\right)^2,$$

which is computable in linear time with respect to the number of samples. Number of test locations is usually small in practice (less than 100) and can be treated as constant. Quadratic time of MMD is somewhat limiting in case of frequent computation on moderately sized sample sets, e.g. during GAN training. In contrast, UME can be computed quickly enough not to dominate the time complexity even when using sets of tens of thousands samples in size, possibly leading to more robust model criticism.

The empirical estimate of UME^2 may be rewritten in terms of so-called *feature functions*, which will be used during construction of the relative similarity test based on UME. Using the same notation as in the original definition of UME^2, we can define the feature function $\psi_V : \mathcal{X} \to \mathbb{R}^J$ to be

$$\psi_V(x) := \frac{1}{J}\big(k(x, v_1), \ldots, k(x, v_J)\big)^\top.$$

UME can be then defined as

$$\mathrm{UME}^2(P, Q) := \|\mathrm{E}_{p\sim P}\,\psi_V(p) - \mathrm{E}_{q\sim Q}\,\psi_V(q)\|_2^2,$$

of which a consistent unbiased estimation is

$$\widehat{\mathrm{UME}}^2(P, Q) := \frac{1}{n(n-1)} \cdot \left(\left\|\sum_{i=1}^{n}\big(\psi_V(p_i) - \psi_V(q_i)\big)\right\|^2 - \sum_{i=1}^{n}\left\|\psi_V(p_i) - \psi_V(q_i)\right\|^2\right).$$

UME is a random metric. It satisfies all axioms of a metric almost surely, namely identity of indiscernibles, symmetry and triangle inequality. The first axiom, stating $\mathrm{UME}(P, Q) = 0 \Rightarrow P = Q$, is especially useful for our use case, since it ensures consistency of the studied statistical tests. MMD is a random metric on the condition that $\mathcal{X}$ is a compact metric space and kernel k is continuous [1].

2.4 Relative UME Test

Similarly to the MMD test, UME test decides, whether P is closer to reference distribution R than Q is, as measured by UME distance:

$$\mathcal{H}_0 : \mathrm{UME}^2(R, P) \leq \mathrm{UME}^2(R, Q),$$
$$\mathcal{H}_1 : \mathrm{UME}^2(R, P) > \mathrm{UME}^2(R, Q).$$

Since UME needs a set of pre-chosen test locations and the test utilizes two computations of UME ($\mathrm{UME}^2(R, P)$ and $\mathrm{UME}^2(R, Q)$), it is possible to use different sets of test locations for each computation, as well as different kernels. The test will be

presented in the fully general form with different locations V and W, as introduced in the original paper [6]. However, for intuitive interpretability, we will use the same locations $V = W$ and a single kernel k in all experiments.

Let's implement a shortened notation for covariances of feature functions. Let

$$\begin{aligned} C_V^P &:= \mathrm{var}_{x\sim P}\big(\psi_V(x)\big) \\ C_W^Q &:= \mathrm{var}_{y\sim Q}\big(\psi_W(y)\big) \\ C_{VW}^R &:= \mathrm{cov}_{z\sim R}\big(\psi_V(z), \psi_W(z)\big). \end{aligned}$$

Let $S^U := \mathrm{UME}^2(P,R) - \mathrm{UME}^2(Q,R)$ be the difference of UME between reference distribution and respective candidate distributions and $\hat{S}^U := \widehat{\mathrm{UME}}^2(P,R) - \widehat{\mathrm{UME}}^2(Q,R)$ its empirical counterpart. Let

$$M := \begin{pmatrix} \mathrm{E}_{x\sim P}\,\psi_V(x) - \mathrm{E}_{z\sim R}\,\psi_V(z) & 0 \\ 0 & \mathrm{E}_{y\sim Q}\,\psi_W(x) - \mathrm{E}_{z\sim R}\,\psi_W(z) \end{pmatrix}.$$

Assuming that P, Q and R are mutually distinct, kernels and test locations V, W are chosen such that $\mathrm{UME}^2(P,R) > 0$ and $\mathrm{UME}^2(Q,R) > 0$ and matrix

$$\begin{pmatrix} \zeta_P^2 & \zeta_{PQ} \\ \zeta_{PQ} & \zeta_Q^2 \end{pmatrix} := M^\top \begin{pmatrix} C_V^P + C_V^R & C_{VW}^R \\ (C_{VW}^R)^\top & C_W^Q + C_W^R \end{pmatrix} M$$

is positive definite. Then,

$$\sqrt{n}\left(\hat{S}^U - S^U\right) \xrightarrow{\mathrm{D}} \mathcal{N}\big(0, 4(\zeta_P^2 - 2\zeta_{PQ} + \zeta_Q^2)\big).$$

The test uses $\sqrt{n}\hat{S}^U = \sqrt{n}\left(\widehat{\mathrm{UME}}^2(X_R, X_P) - \widehat{\mathrm{UME}}^2(X_R, X_Q)\right)$ as the test statistic. Assuming all above assumptions are met, the test statistic converges in distribution to a normal distribution centered in $\sqrt{n}S^U$. If we knew S^U, we could construct the test at α significance level by checking if the test statistic falls into $1-\alpha$ quantile of said normal distribution $\mathcal{N}\big(\sqrt{n}S^U, 4(\zeta_P^2 - 2\zeta_{PQ} + \zeta_Q^2)\big)$. The theoretical value of S^U is unknown in practice. Nevertheless, if the null hypothesis is true and $\mathrm{UME}^2(P,R) \leq \mathrm{UME}^2(Q,R)$, it necessarily follows that $S^U \leq 0$. Therefore, if the test uses normal distribution centered in 0 instead of $\sqrt{n}S^U$, the significance level of the test will be lower or same at worst, so we can eliminate the S^U term. Finally we can derive the p-value to be

$$p \leq \Phi\left(-\frac{\sqrt{n}\left(\widehat{\mathrm{UME}}^2(X_R, X_P) - \widehat{\mathrm{UME}}^2(X_R, X_Q)\right)}{2\sqrt{\hat{\zeta}_P^2 - 2\hat{\zeta}_{PQ} + \hat{\zeta}_Q^2}}\right),$$

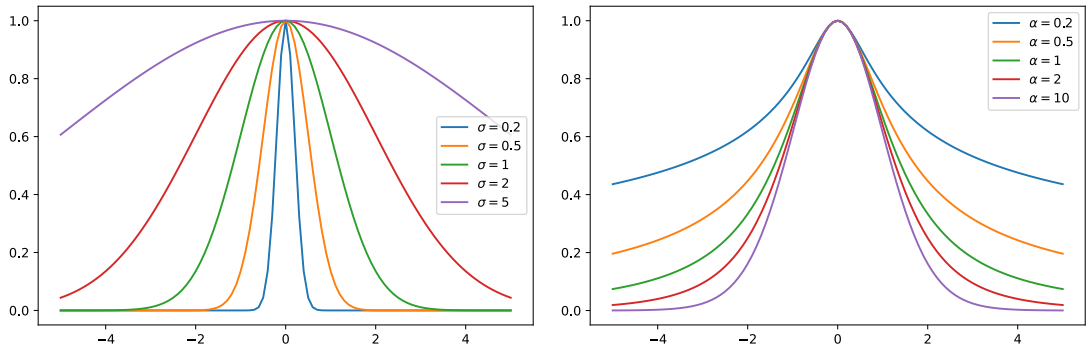
where Φ is the cumulative distribution function of standard normal distribution, $X_P \sim P, X_Q \sim Q, X_R \sim R$ are input data and $\hat{\zeta}_P^2$, $\hat{\zeta}_{PQ}$, $\hat{\zeta}_Q^2$ are empirical estimates of ζ_P^2, ζ_{PQ} and ζ_Q^2.

2.5 Kernel Functions

The most well-known kernel function which also satisfies the requirements for being used for MMD and UME computations is a *radial basis function* (RBF) kernel, also known as *Gaussian kernel*, with bandwidth parameter σ:

$$k_\sigma^{\text{rbf}}(x, y) := \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right).$$

The σ parameter controls the spread of the function, as can be seen in Figure 2.1a.



(a) (b)

Figure 2.1. Parametrization of (a) RBF kernel $k_\sigma^{\text{rbf}}(x, 0)$ and (b) rational quadratic kernel $k_\alpha^{\text{rq}}(x, 0)$ in one dimension.

Choosing a suitable value of σ is a non-trivial problem. An approach chosen by [5] is to set the parameter to the median pairwise distance between datapoints from a set of held-out data. We will show the disadvantages of this technique later in the experimental chapters. A more sophisticated approach is choose the parameter value to maximize the power of the statistical test, as studied by [7] for MMD testing.

It is also possible to use a kernel which is a mixture of RBF kernels with various fixed bandwidth values:

$$k(x, y) = \sum_{\sigma \in \Sigma} k^{\text{rbf}}(x, y),$$

which has been successfully used by [8] for MMD computation on small images, setting $\Sigma = \{2, 5, 10, 20, 40, 80\}$.

Although being widely used, the RBF kernel has an unfavourable property in high dimensions – the tails decline towards zero quite quickly, outputting very low numbers for moderately distant x and y, rendering any following processing numerically unstable. For this reason, [9] recommends to use a *rational quadratic* (RQ) kernel instead:

$$k_\alpha^{\text{rq}}(x, y) := \left(1 + \frac{\|x - y\|^2}{2\alpha}\right)^{-\alpha},$$

which has much better tail behaviour, as can be seen in Figure 2.1b. Furthermore, it may be less dependent on the exact choice of parameter α, which controls the heaviness of the tails, but maintains the same general shape around zero.

From a certain point of view, RQ kernel can be thought of as an extension of the idea of using a mixture of several RBF kernels with different bandwidth values, since a RQ kernel can be seen as a scale mixture of RBF kernel functions, putting a Gamma distribution on σ^{-2}, a transformed parameter of a RBF kernel [10].

Chapter 3
Generative models

Generative models are a class of machine learning models. Having input/observations X and target variables Y, generative models aim to learn the joint probability $\mathrm{P}(X, Y)$. In contrast to discriminative models, which only learn conditional probability $\mathrm{P}(Y|X)$, with generative models it's possible to sample the learnt distribution, obtaining new data similar to the training data.

Examples of generative models are Bayesian network, hidden Markov model, or generative adversarial network. This thesis focuses solely on the latter, since it works with the most complex data, causing the need for advanced model criticism frameworks, such as MMD or UME presented here.

First, generative adversarial network will be presented as a general framework. The following sections will describe specific architectures chosen for experiments conducted for the purpose of this thesis.

3.1 Generative Adversarial Network

Generative adversarial network (GAN) is a framework for training of generative neural network in an adversarial process, introduced by [11] The framework consists of two models trained simultaneously – generator G and discriminator D, which compete with each other in a minimax game, meaning that minimizing loss function of one makes it harder for the other one to minimize its loss and vice versa.

Formally, the GAN searches for G and D attaining the minimum and maximum in $\min_G \max_D \ L(G, D)$, where L is the objective function defined to be

$$L(G, D) := \mathrm{E}_{y \sim Y} \left[\log D(y)\right] + \mathrm{E}_{x \sim X} \left[\log\big(1 - D\big(G(x)\big)\big)\right],$$

where X is the latent space, usually chosen as uniform or normal distribution, and Y being the distribution of real data.

During training, generator is being fed with random noise distributed according to X, which is easy to sample from. The aim of generator is to generate samples $G(x), x \sim X$ that are similar to real data samples Y. The architecture of generator is largely dependent on the domain, generally it is some deep network.

Discriminator gets real data $y \sim Y$ on its input, as well as generated data $G(x)$. Its objective is to determine, whether the input is real or fake (generated by generator). This makes discriminator a binary classifier.

Both networks are trained using standard backpropagation. Generator gets positive feedback if it succeeds in fooling discriminator, discriminator gets positive feedback if it correctly distinguishes between real and fake samples. Figure 3.1 gives an overview of the GAN architecture. The ultimate goal is to create generator which captures data distribution perfectly, and $D(y) = D\big(G(x)\big) = \frac{1}{2}$ everywhere.

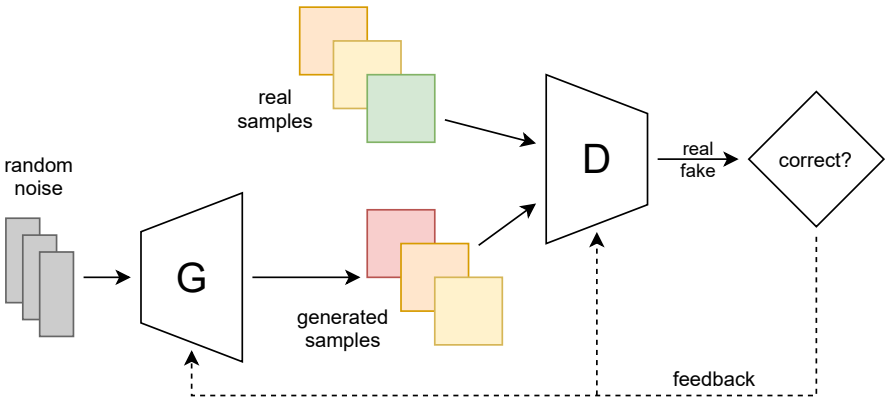


Figure 3.1. General GAN architecture

The original paper [11] suggests the use of minibatch stochastic gradient descent training to obtain optimal G and D with respect to the objective function. Until the stopping condition (number of iterations) is met, the algorithm alternates the training steps of discriminator and generator, training the discriminator for k steps per each generator step, k being a hyperparameter. It may be necessary to set $k > 1$ to keep D closer to optimal, since it is the inner function in expression $\min_G \max_D \ L(G, D)$.

Let m be the size of minibatch. For discriminator training step, m random vectors $x_1, \ldots, x_m$ are sampled from X and m real samples $y_1, \ldots, y_m$ are sampled from real data Y. Discriminator weights w_D are updated according to gradient

$$g_D = \nabla_{w_D} \frac{1}{m} \sum_{i=1}^{m} \Big(\log D\big(y_i\big) + \log\big(1 - D(G(x_i))\big) \Big).$$

Note that discriminator is ascending the gradient as it tries to maximize the objective function.

For generator training step, a minibatch of noise samples $x_1, \ldots, x_m$ is sampled from X. Generator weights are updated according to

$$g_G = -\nabla_{w_G} \frac{1}{m} \sum_{i=1}^{m} \Big(\log\big(1 - D(G(x_i))\big) \Big).$$

In comparison to the update rules of discriminator, the $\log D(y)$ component is missing, because generator only cares about fooling the discriminator and does not operate at all with real samples. In practice, minimization of $\mathrm{E}_{x \sim X} \log\big(1 - D(G(x))\big)$ is superseded

by maximization of $\mathrm{E}_{x\sim X} \log D(G(x))$ for better gradients in early stages of training, and update rule for generator is adjusted appropriately.

The most prominent data type processable by GANs are images, although variations of GAN are perfectly capable of generation of music [12] or video [13]. Notable applications of GAN-based networks on image data are inpainting of damaged parts [14], colourization [15] or super-resolution (resolution upscaling) [16]. Even the task of realistic image synthesis as such may be useful e.g. for stock photo generation.

3.2 Deep Convolutional GAN

Deep convolutional generative adversarial network (DCGAN) is a type of GAN designed specifically for images, inspired by the success of convolutional neural networks (CNN) in supervised learning. The big picture of GAN architecture remains the same, DCGAN just specifies the generator and discriminator more precisely.

The constraints for DCGAN, which allow the utilization of convolutional layers in an unsupervised manner in GANs, as listed by original authors [17], are:

- Using strided convolutional layers instead of deterministic spatial pooling functions. For generator, use transposed strided convolutional layers.
- Not including any fully connected hidden layers. The exception being the very first layer of generator, which is a dense layer reshaping the input random vectors to a basis of the convolutional stack. Discriminator uses a single perceptron connected to all outputs of a previous layer after flattening.
- Using batch normalization in both generator and discriminator.
- Using ReLU or LeakyReLU activation, and tanh activation in the last output of the generator.

The authors used four transposed convolutional generator layers for generation of 64×64 RGB images of bedrooms. This provides some starting point for tuning of the number of convolutional layers to be used with our datasets.

In the process of conducting experiments for this thesis, DCGAN itself didn't lead to good enough results for some other experiments, therefore a more advanced architecture has been chosen – Wasserstein GAN, which builds on the basis founded by DCGAN.

3.3 Wasserstein GAN

Wasserstein GAN (or simply WGAN) is a modification of the standard GAN training algorithm described in section 3.1. Developed by [18], it aims to improve stability of training, making it easier to maintain the balance between generator and discriminator (or *critic* in case of WGAN) performance. Moreover, it dramatically reduces one of the common problems arising during GAN training – mode collapse.

Mode collapse occurs when GAN correctly learns part of the target distribution, but completely ignores some other part. By not generating that part of the target distribution, traditional GAN setup does not penalize the generator in any way, because the discriminator only assesses quality of an actual generator output and does not compare the learnt distribution to target distribution directly.

Even though the module of the Wasserstein GAN responsible for determining if its input is real or generated data is called critic in WGAN, in this thesis, we mostly stick to the term "discriminator". The terms are conceptually very close and for purposes of this thesis, the distinction is not important, because the methods described here would generally work for any GAN architecture.

Wasserstein GAN replaces the original objective function with Wasserstein-1 distance, also known as *earth mover's distance*. The term earth mover's distance refers to the intuitive interpretation of the distance, which is the amount of (probability) mass needed to reshape one distribution to another, multiplied by the distance this mass would need to travel. Formal definition of the Wasserstein-1 distance is:

$$W(X, Y) := \inf_{\gamma \in \Pi(X,Y)} \mathrm{E}_{(x,y)\sim\gamma} \|x - y\|.$$

There, $\Pi(X, Y)$ denotes the set of all joint distributions whose marginals are X and Y. This form is, however, hard to tackle computationally. Fortunately, we can use its dual form [19], which computes supremum over 1-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$:

$$W(X, Y) = \sup_{\|f\|_L \leq 1} \mathrm{E}_{x\sim X} f(x) - \mathrm{E}_{y\sim Y} f(y).$$

In fact, it is enough to restrict ourselves to K-Lipschitz class of functions for some constant K, as the computed distance would differ from Wasserstein-1 distance only by a multiplicative constant.

It is shown in [18] that the original GAN objective function is a lower bound of $2JS(X, Y) - 2\log 2$, where JS is Jensen-Shannon divergence, which is a symmetrized and smoothed version of the well-known Kullback–Leibler divergence. It is further shown that under reasonable assumptions, Wasserstein-1 distance is continuous and differentiable almost everywhere, usable as a cost function for learning distributions supported by low-dimensional manifold, and is better suited for that task than Jensen-Shannon divergence, thus the original GAN objective function.

Let m be the size of minibatch. For discriminator training step, m random vectors $x_1, \ldots, x_m$ are sampled from X and m real samples $y_1, \ldots, y_m$ are sampled from real data Y. The weight update rules used during WGAN training are adapted accordingly, now using the following gradients:

$$g_D = \nabla_{w_D} \frac{1}{m} \sum_{i=1}^{m} D(x_i) - \frac{1}{m} \sum_{i=1}^{m} D(G(y_i)),$$

$$g_G = \nabla_{w_G} \frac{1}{m} \sum_{i=1}^{m} D(G(y_i)).$$

Other necessary adjustment of the learning algorithm is clipping of the discriminator weights to some sensible range after each weight update, to adhere to the Lipschitz constraint. Weight clipping is known to be problematic in some cases and newer research suggests to use *gradient penalty* instead, which serves as a soft, smooth version of weight clipping [20].

As the name suggests, this method penalizes the norm of gradient of the discriminator output with respect to its input. That pushes the gradients closer to 1, encouraging the Lipschitz property. The final loss function of WGAN with gradient penalty (WGAN-GP) is

$$L(G, D) := \mathrm{E}_{y\sim Y}\left[D(y)\right] + \mathrm{E}_{x\sim X}\left[D\big(G(x)\big)\right] + \lambda \mathrm{E}_{z\sim Z}\left[(\|\nabla_z D(z)\|_2 - 1)^2\right],$$

where Y are the real data, X is random noise, λ is a hyperparameter controlling the emphasis put on gradient penalty, and Z are samples linearly interpolated between X and Y, i.e.

$$z \leftarrow \epsilon y + (1 - \varepsilon)G(x), \quad x \sim X, y \sim Y, \varepsilon \sim \mathrm{Uniform}(0, 1).$$

By default, WGAN uses $k = 5$ discriminator training steps per each generator step and uses RMSProp as the optimizer. WGAN-GP replaces the optimizer with Adam and sets $\lambda = 10$ by default.

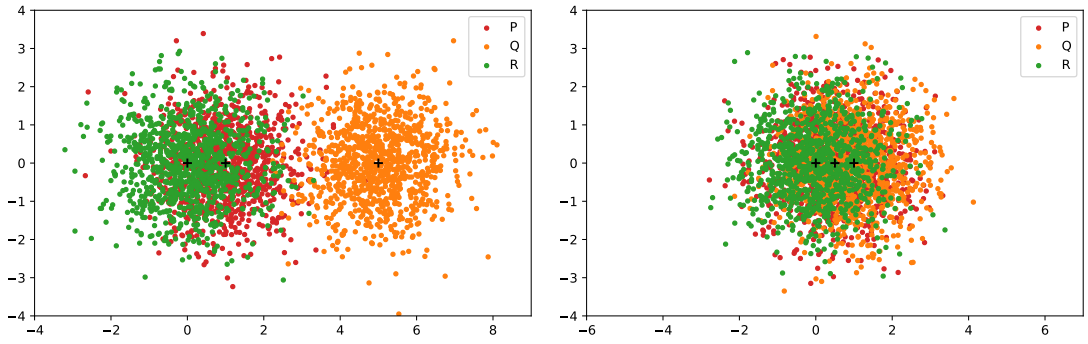
Chapter 4
Experiments with UME Testing Locations

In Section 2.3, unnormalized mean embedding has been defined as a mean of the witness function computed over J test locations $V = \{v_1, \ldots, v_J\}$. In this chapter, we experiment with the placement of those test locations, and evaluate the robustness of UME with respect to their placement using randomly generated toy datasets.

4.1 Mean Shift Dataset

Mean shift dataset consist of three Gaussian blobs of 1000 samples each. The blobs are 50-dimensional and have all the same variances in all directions. They only differ in the placement of their centre in the first dimension. Remaining 49 dimensions are centred in zero.

Let Σ be the identity matrix $\Sigma = I_{50}$. The samples are randomly generated as $P, Q, R \sim \mathcal{N}\big((c, 0, \ldots, 0)^\top, \Sigma\big)$, where c is specific to each blob. Two variants will be given for this dataset: easier to distinguish $c_P = 1, c_Q = 5, c_R = 0$ and harder $c_P = 0.5, c_Q = 1, c_R = 0$. The first two dimensions of each variant are plotted in Figure 4.1.



(a) (b)

Figure 4.1. First two dimensions of the mean shift datasets. Variant (a) has blob Q five times further from R than P, variant (b) has Q only twice as far. Blob centres are denoted by black marks.

When the distinction is necessary, blob R will be considered the reference set and blobs P and Q some two candidate models, P being closer to ground-truth than Q.

The tests described in Sections 2.2 and 2.4 should be able to detect that although P doesn't fit R perfectly, it is closer to it than Q. When testing the null hypothesis of Q being closer than P, the tests are expected to return p-value close to zero. The tests are executed in Python environment, so common pitfalls such as limited precision and slight numerical errors must be taken into consideration. Unless stated otherwise, a RBF kernel is used for those simple, artificially generated datasets. Following [5], the bandwidth of the kernel is estimated as median pairwise distance between samples put aside from the main dataset.

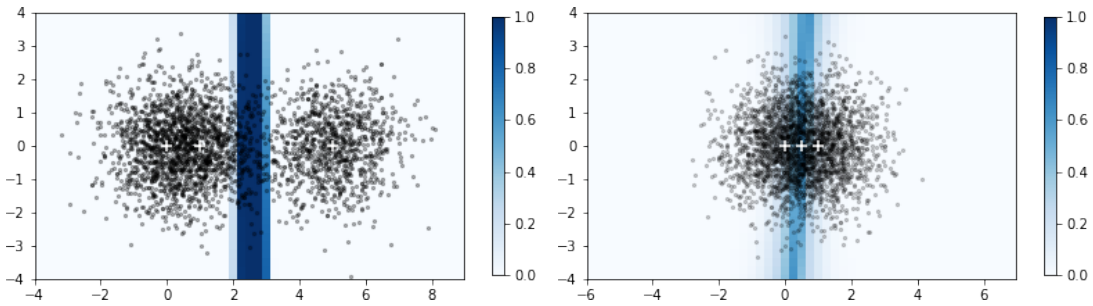
On this simple toy dataset, the MMD test behaves exactly as expected, as can be seen in Table 4.1. When testing Q being closer to R than P, the p-values are nearly zero for both dataset variants. The reverse direction of the test is indecisive with $p = 1$.

Table 4.1. Computed p-values of MMD tests on both easy and hard mean shift datasets. The test has been executed for both possible directions of the null hypothesis.

$\mathcal{H}_0$	$\mathrm{MMD}^2(Q,R) \leq \mathrm{MMD}^2(P,R)$	$\mathrm{MMD}^2(P,R) \leq \mathrm{MMD}^2(Q,R)$
easy	0.0	1.0
hard	$1.92 \cdot 10^{-25}$	1.0

4.1.1 Single UME Test Location

Evaluating UME tests isn't as straightforward as MMD tests due to the presence of test locations which need to be chosen in advance. Let's explore the simplest case of a single test location. Test location v of a specific form $v = (x, y, 0, \ldots, 0)$ where x and y will be chosen in the range similar to the range of available data in P, Q, R sets. The UME test for $\mathrm{UME}^2(Q,R) \leq \mathrm{UME}^2(P,R)$ will be executed using different v values with steps of size 0.25. The resulting p-value is presented in Figure 4.2, only first two dimensions are shown. The points of P, Q, R along with their centres are plotted over the p-values to give better sense of space.



(a) (b)

Figure 4.2. Mean shift p-values for single testing location on (a) easy variant, (b) hard variant

As we can see, even for data as simple as mean shift, the testing results may vary significantly based on the choice of the test location, ranging from correctly rejecting the null hypothesis to p-values close to 1. In both variants of the dataset, the test in indecisive if the test location is chosen to be on the border between P and Q, since density of P and Q is similar in that area, so the test can't decide which of them is closer to R.

To mitigate the problem of choosing unsuitable test location can obviously be mitigated by using multiple test locations. A simple heuristic is to choose several points from distributions P, Q, R, which in practice may be implemented by designating a few datapoints from each set to be test locations and removing them from any further processing. A more sophisticated approach is to optimize the test location to maximize the test power.

It has been shown that regions which lead to highest test power are regions, where one candidate model most significantly outperforms the other [6]. Those areas are very interesting e.g. for model creators, who can easily detect mode collapse or generally pinpoint problematic data for some otherwise well-performing model. The power criterion objective can be estimated by evaluating the following function:

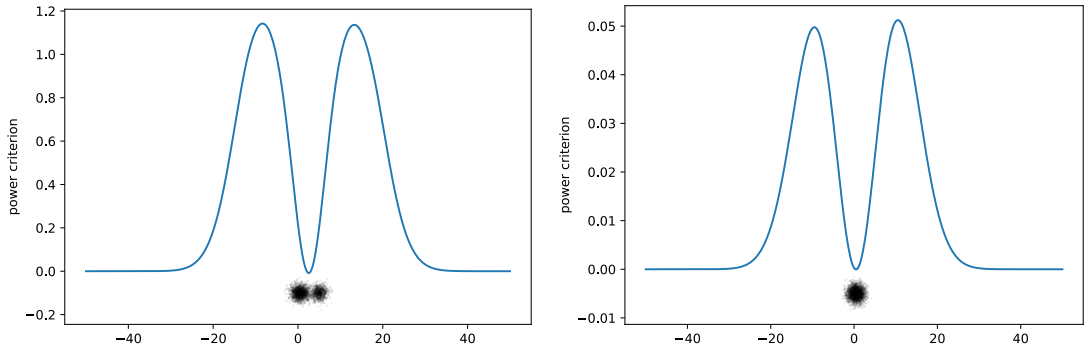
$$\text{pcrit} := \frac{\widehat{\text{wit}}_{P,R}^{\,2} - \widehat{\text{wit}}_{Q,R}^{\,2}}{\gamma + 2\sqrt{\hat{\zeta}_P^2 - 2\hat{\zeta}_{PQ} + \hat{\zeta}_Q^2}},$$

at the testing location v. Here, $\widehat{\text{wit}}_{P,R}$ and $\widehat{\text{wit}}_{Q,R}$ are empirical witness functions, $\hat{\zeta}_P^2$, $\hat{\zeta}_Q^2$ and $\hat{\zeta}_{PQ}$ empirical covariances as presented in Section 2.4, and $\gamma > 0$ a small regularization parameter added for numerical stability [21].

Power criteria for mean shift dataset, assuming $\mathcal{H}_0 : \text{UME}^2(Q, R) \leq \text{UME}^2(P, R)$, which is false, are plotted in Figure 4.3 for both dataset variants. We can clearly see that the power criterion is low in between the clusters, because in this area samples from P and Q have similar density and the test can't conclude which one is closer to R. The power criterion then raises on both sides in the first dimension, where differences between P and Q are clear. It finally decreases back to zero in extreme distances, since the differences between clusters are negligible compared to the distance between any of them and the test location.

4.1.2 Optimized Test Locations

The idea of optimizing test locations to maximize test power lies in numerically ascending the power criteria function from some starting locations chosen randomly from distributions P, Q, R. In simple cases such as this toy dataset, L-BFGS-B algorithm [22] may be used to perform the numerical optimization. On more complex datasets, however, we found that it often fails to converge to reasonable optimum. On the other hand, Adam optimizer [23] seems to perform well even in such cases, that's why it is

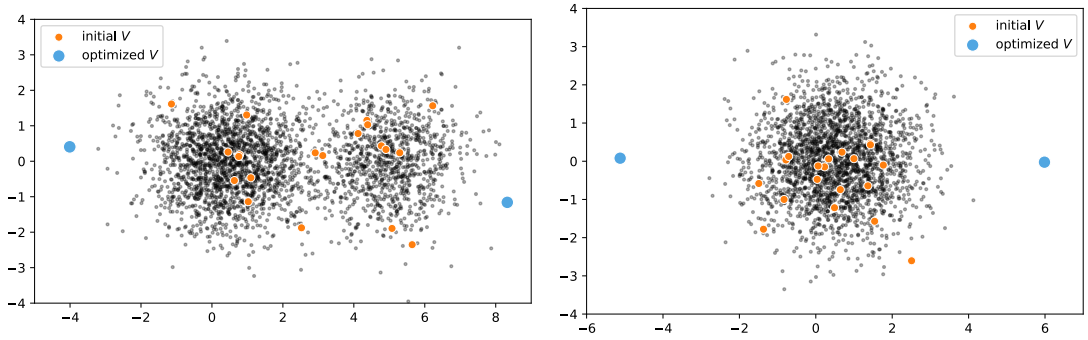


(a) (b)

Figure 4.3. Power criteria of the meanshift dataset: (a) easy variant, (b) hard variant.

used instead of L-BFGS-B for image datasets. The test locations should be optimized on a different set of samples than the what will be used for the actual test.

When initializing 20 test location randomly chosen from P, Q and R, the optimization for test power iteratively pushes them away from the blob centre. They all seem to converge to only two points, one to the left and other to the right of the data, as can be seen in Figure 4.4. In second and higher dimensions, the optimized locations are close to zero, which is the centre of all three distributions.



(a) (b)

Figure 4.4. Optimized test locations for the mean shift dataset: (a) easy variant, (b) hard variant.

4.1.3 Random Test Locations

The p-values of UME test for $\mathcal{H}_\prime$: $\mathrm{UME}^2(Q, R) \leq \mathrm{UME}^2(P, R)$, which should be rejected, are noted in Table 4.2. The testing locations were chosen randomly from a held-out set of data coming from all three distributions, and 20 of them are used. For comparison, results with optimized test locations are shown.

For such a simple dataset, randomly selected testing locations span enough of the space to find good spots, where the discrepancy between P and Q is large, so test location optimization is not as important. Nevertheless, we have demonstrated that

Table 4.2. Computed p-values of UME tests on both easy and hard mean shift datasets.

	random V	optimized V
easy	0.0	0.0
hard	$1.72 \cdot 10^{-4}$	$5.26 \cdot 10^{-26}$

the optimization behaves intuitively and if the locations are able to converge to local optima, they label the areas of greatest difference between Q and P with respect to the target data R.

4.2 Skewed Gaussians

Skewed Gaussians dataset is useful for demonstration of the necessity of good kernel choice. The basic building block is a two-dimensional normal distribution:

$$R \sim \mathcal{N}\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}5 & 0\\0 & 0.5\end{pmatrix}\right).$$

Distributions P and Q are derived from R by rotating it $\frac{\pi}{6}$ and $\frac{\pi}{2}$, respectively, around the origin. Taking 1000 samples from each, we obtain the dataset pictured in Figure 4.5.

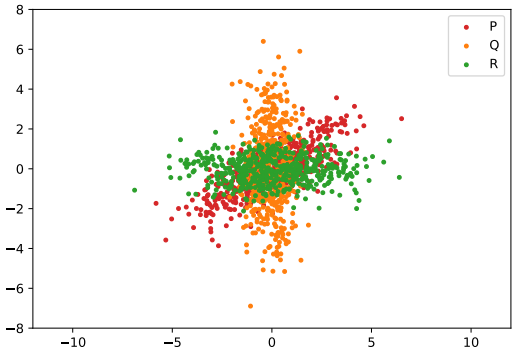


Figure 4.5. Skewed Gaussians building block

When using RBF kernel k_σ^{rbf} with σ set using the pairwise distance heuristic, both MMD and UME tests are able to reject the hypothesis that Q fits R better than P on usual significance levels, resulting in p-value of $5.03 \cdot 10^{-47}$ for MMD and $3.12 \cdot 10^{-20}$ for UME.

The power criterion function in the neighbourhood of the data has maxima at the north, west, east and south ends of P and R, see Figure 4.6.

To better understand the shape of the power criterion, we may take a look at the witness functions wit_{PR} and wit_{QR} in Figure 4.7. The intuitive meaning of the witness function is that it measures the difference between two distributions. High values thus

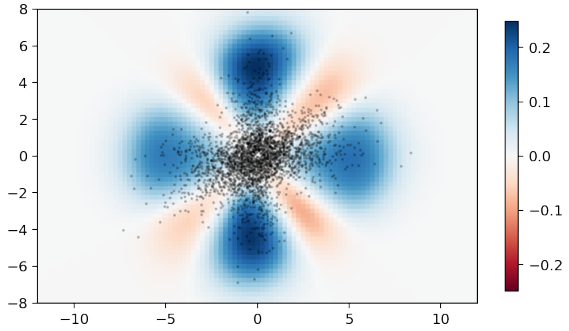


Figure 4.6. Skewed Gaussians building block power criterion function

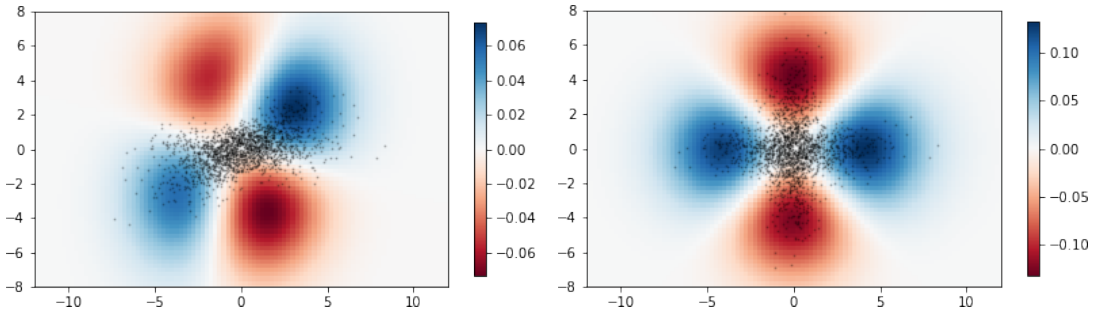


Figure 4.7. Skewed Gaussians building block witness functions for (a) P/R and (b) Q/R

denote areas, where P/Q has higher probability mass than R. Since the power criterion function (defined in Section 4.1.1) is proportional to the difference of two squared witness functions, we can now see the reason why the power criterion in Figure 4.6 has the particular shape.

Again, these building block data are so simple that test location optimization is not necessary.

If we clone the distributions P, Q, R four times and move them to new centres $(-10, 10)^\top, (-10, 10)^\top, (10, 10)^\top, (10, -10)^\top$, we obtain a dataset similar to Figure 4.8.

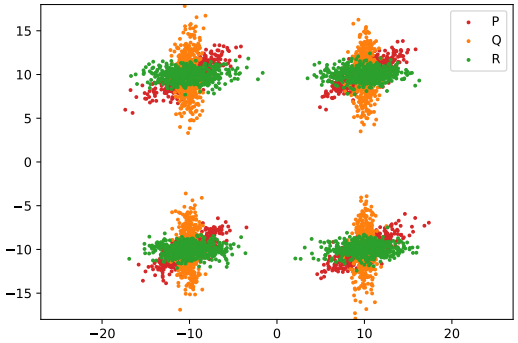
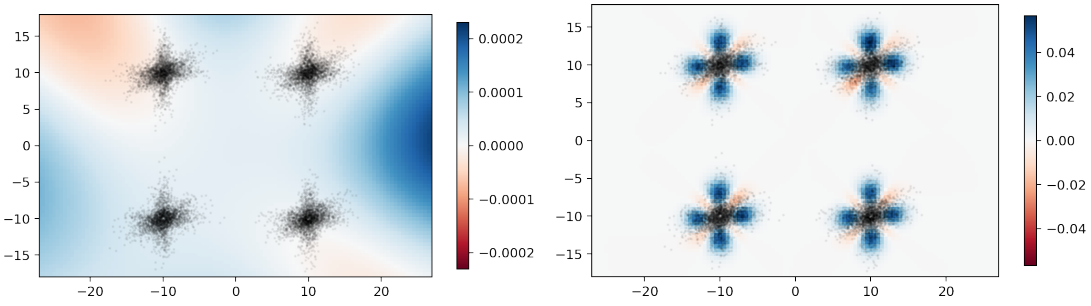


Figure 4.8. Skewed Gaussians dataset

On such data, simple heuristic for estimating RBF kernel bandwidth becomes deficient, since global pairwise distances are much larger than differences between distributions, which take place only locally in each quadrant. This results in overestimating the value of σ and, in consequence, inability to capture fine differences. The heuristic sets $\sigma = 406.5$ and the power criterion computed using k_σ^{rbf} drawn in Figure 4.9a.



(a) (b)

Figure 4.9. Skewed Gaussians dataset power criterion using RBF kernel with parameter σ (a) set by pairwise distance heuristic, (b) optimized to maximize test power.

It can clearly be seen that the power criterion doesn't have the expected shape. Each quadrant should be similar to what has been observed for a single building block in Figure 4.6. With this setup, the UME test fails to reject $\mathcal{H}_\prime : \text{UME}^2(Q, R) \leq \text{UME}^2(P, R)$ using 20 test locations, resulting in p-value of 0.404 for randomly chosen test locations and 0.309 for locations optimized by ascending the flawed power criterion function shown in Figure 4.9a.

On the other hand, optimizing the kernel bandwidth to maximize test power leads to $\sigma = 2.35$, which is much more suitable for disclosure of local discrepancies between our particular distributions. The power criterion function looks very similar to power criterion computed over the single-quadrant data, which is ideal. See Figure 4.9b.

Rational quadratic kernel k_α^{rq} performs well on this dataset. Thanks to its heavy tails, but still well-defined peak, it is able to recognize small scale features in a wide range of values of parameter α. Figure 4.10 shows the power criterion over the neighbourhood of the data, and we can see that the default value $\alpha = 1$ performs equally well as the value optimized to maximize the test power, which is $\alpha = 318.4$. This result is in line with the assumption that RQ kernel is more robust with respect to the value of α. This is advantageous, since the optimization procedure itself has hyperparameters which must be tuned.

To demonstrate the benefit of test location optimization as opposed to just randomly choosing them, even on such simple dataset, another experiment will be conducted. Intentionally inappropriate kernel will be used for this problem – $k_5^{\text{rbf}}0$, which has too large bandwidth and UME test performed using a single randomly chosen location isn't able to decide which of P, Q distributions is closer to R. However, when increasing the

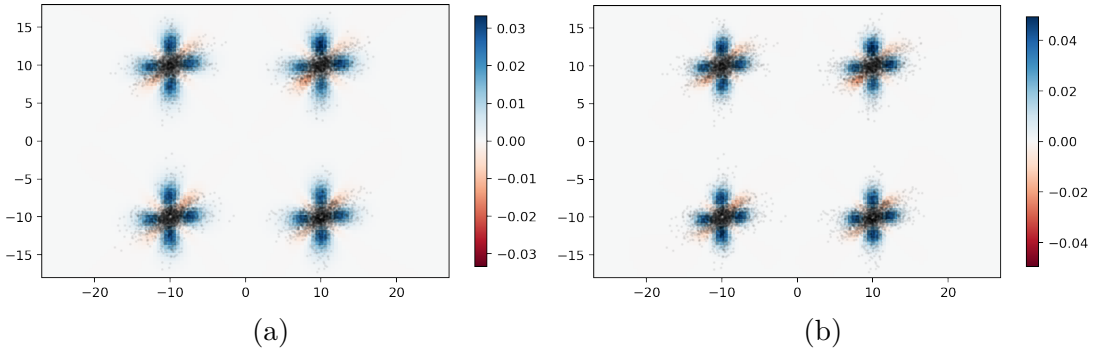


Figure 4.10. Skewed Gaussians dataset power criterion using rational quadratic kernel with parameter α (a) set to 1, (b) optimized to $\alpha = 318.4$, which maximizes test power.

number of test locations, it is expected that either one of the test location hits a sweet spot, where $\mathrm{UME}^2(\cdot, R)$ differs significantly for P and Q even with such wide kernel, or averaging over many test locations will increase the confidence of the test enough to declare the difference between goodness of fit of P and Q statistically significant.

The p-values of UME test testing $\mathcal{H}_\prime : \mathrm{UME}^2(Q, R) \leq \mathrm{UME}^2(P, R)$ against $\mathcal{H}_\infty : \mathrm{UME}^2(Q, R) > \mathrm{UME}^2(P, R)$, where $\mathcal{H}_\infty$ is true in reality, will be monitored for varying amount of randomly chosen test locations. The result can be seen in Figure 4.11, and it agrees with the expectations that increasing the number of test location improves the test's ability to detect differences even with poorly shaped kernel. Having enough test locations covering the source distributions seem to compensate for inappropriately chosen kernel and since certain number, the test consistently rejects the null hypothesis at approximately the same level as a single location optimized for test power.

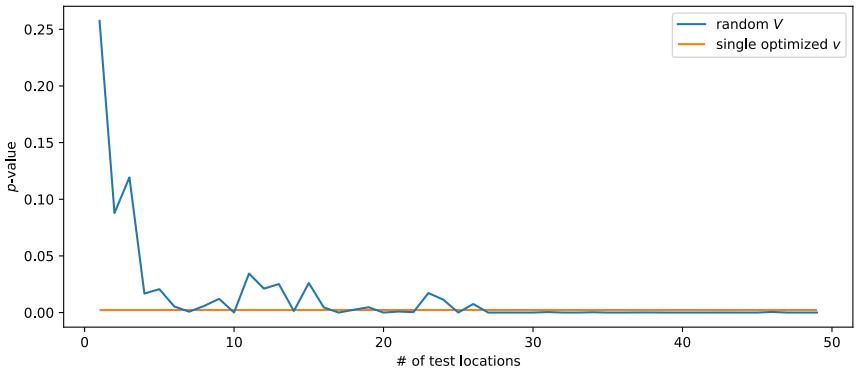


Figure 4.11. UME test p-value based on number of test locations

If this behaviour is present also in higher dimensions and more complex datasets, it would further improve resistance to incorrectly set kernel parameter, in addition to usage of a rational quadratic kernel instead of RBF kernel. This may be important for experiments on image data, because choosing the value of kernel parameter is a problem with no clear robust solution.

4.3 Distortion Detection

MNIST (Modified National Institute of Standards and Technology database) is a dataset of black-white handwritten digits, widely used by machine learning community for initial evaluation of algorithms and benchmarking, as it is sufficiently complex to be non-trivial, yet small and simple to be processed both by resource-heavy algorithms and the simple ones, yielding good results. MNIST consists of 70 000 images of digits 28×28 pixels in size, with each class (0 to 9) represented equally [24]. It is usable for both supervised and unsupervised tasks thanks to being labelled.

For UME and MMD goodness of fit testing, the images will be represented as vectors from $\langle -1, 1 \rangle^{784}$ domain, where individual rows of image pixels are stored one after another, thus not retaining the original spatial properties. For several experiments, only certain digits will be used at the same time, most frequently digit 4 or digit 3. An example of MNIST data can be seen in Figure 4.12. On the right, there is a sample of 400 images of threes and fours respectively, flattened into 784-dimensional vectors and stacked on top of one another. We can clearly tell the difference between number three and number four visually from the flattened vectors, as well as we can see the similarity with digits of the same label.

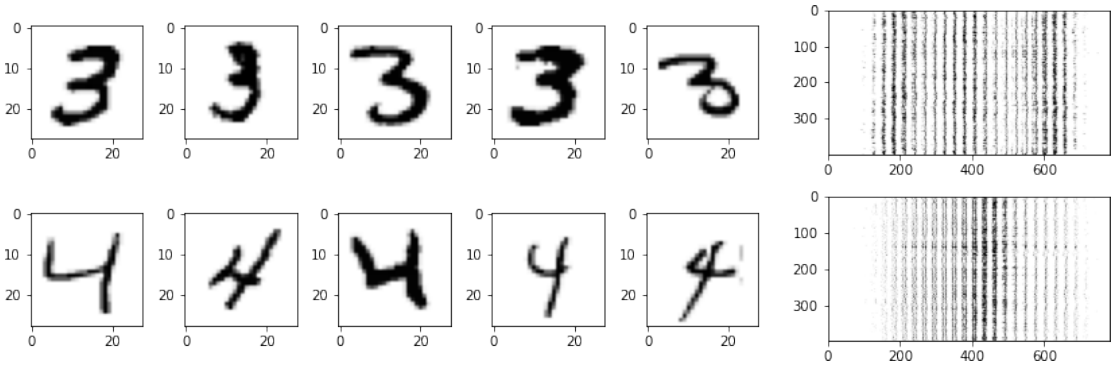


Figure 4.12. Sample of MNIST data, digits 3 and 4. On the right, there is a sample of 400 images of 3 and 4 flattened into 784-dimensional vectors.

As we can see even from the small sample, one notable property of the images is that although the digits are size-normalized and centred, they are tilted the same way as they have been written. The angle of individual strokes from which the digit is composed varies quite a bit between individual samples. Despite this variance, both MMD and UME test is able to detect systematic change in the angle of the images as small as 5 degrees.

To explore how performant the tests are and how they behave with different kernels and test locations, a series of measurements will be done. Only digit four will be taken from the testing part of the MNIST dataset. Last 50 images will be held out to be used as test locations for the UME test. The rest will be evenly split to two groups of size

932. One group will be used as the reference group and the other part as a candidate group, simulating a perfect model which outputs images indistinguishable to real digits. A third group will be created from the second one by rotating all images contained in it around the centre by certain angle, ranging from one degree to 20 degrees.

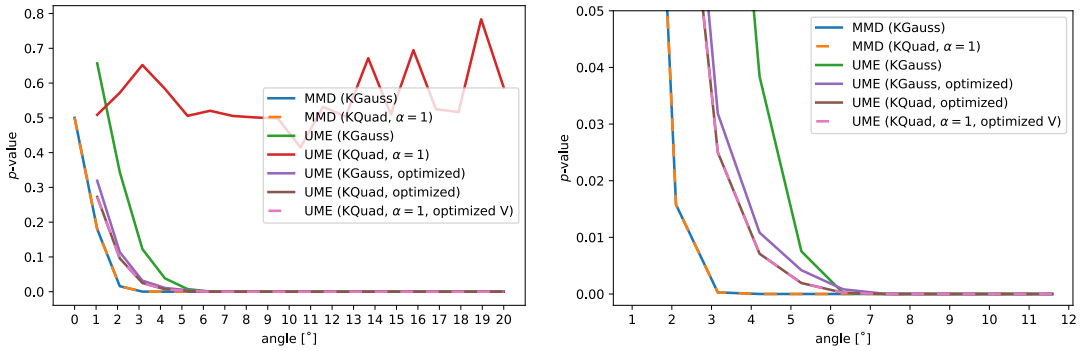
It is expected that with increasing angle, the rotated distribution will differ from the reference one more and more and the test will get progressively better at detecting that the non-rotated dataset is closer to the reference data than the rotated dataset. Let R be the reference set, P the other set of original data (both R and P come from the same distribution, as they both contain real handwritten digits four), and let P_β^{R} be the set of images from P, where all images are rotated anticlockwise about their respective centres by the angle of β. The null hypothesis used for testing in this experiment will be $d^2(P^{\mathrm{R}})\beta, R) \leq d^2(P, R)$, where d substitutes either MMD or UME, depending on the test used.

In order to verify findings from Section 4.2 about kernels and test location optimization, all of the following configurations will be used to perform the tests (colour in parentheses refers to the colour of line in Figure 4.13):

- MMD with a RBF kernel optimized by mean distance heuristic (blue)
- MMD with an unoptimized RQ kernel with default value $\alpha = 1$ (orange)
- UME with a RBF kernel optimized by mean distance heuristic and random test locations (green)
- UME with an unoptimized RQ kernel with default value $\alpha = 1$ and random test locations (red)
- UME with a RBF kernel with both bandwidth σ^2 and test locations optimized for maximal test power (violet)
- UME with a RQ kernel with both parameter α and test locations optimized for maximal test power (brown)
- UME with a RQ kernel with test locations optimized for maximal test power but parameter α set to 1 (pink)

The resulting p-value of the tests with particular configurations for all angles are in Figure 4.13 (a), the same data in detail for p-values ranging from 0 only to 0.05, which is the usual choice of the test significance level, are in Figure 4.13 (b). Some of the lines are dashed only because they nearly copy some other line, so by using a dashed line, the other one remains visible too.

All tests are able to reject the null hypothesis at $\alpha = 0.01$ when rotation by approximately 5° or larger is used for P_β^{R} with an exception of the UME test using random test locations and unoptimized rational quadratic kernel k_1^{rq} (red line). This configuration fails to recognize digits rotated even by 20°, the p-value never drops below 0.4. This observation confirms the necessity to optimize either kernel or UME test locations. Unfortunately, contrary to observations performed on simpler dataset in Section 4.2,



(a) (b)

Figure 4.13. Ability of the test to detect image rotation. Variously parametrized kernels are used and for UME, random and optimized test locations are tried. The range of the y-axis is full in figure (a) and clipped to the usual significance value of $\alpha = 0.05$ in figure (b).

Figure 4.10, rational quadratic kernel with default value of parameter α in combination with random, unoptimized test locations, isn't powerful enough to make the test reject the null hypothesis. That means that using RQ kernel instead of widely known RBF kernel itself can't do without other tweaks such as optimization of test locations or appropriate setting of the kernel's parameter.

The MMD test proved to function better, regardless of the used kernel, whether it is a Gaussian kernel (blue line) optimized only by a median distance heuristic, or a k_1^{rq} kernel (orange line). That is compensated for by its quadratic computational time with respect to the number of data samples, as opposed to linear time of UME. Both kernel choices lead to nearly indistinguishable p-values, and from 3° angle upwards the p-value is near-zero, which is a surprisingly extraordinary result, considering that such small change in data would not be noticed by human. However, this change is systematic in all more than 900 samples, so the estimated difference between probability masses of the reference set and the rotated set accumulates high enough to be spotted by the goodness of fit test.

Another good results were achieved by the UME test using test locations optimized to maximize the power criterion. Similar course of the p-value is observed for RBF kernel with σ^2 parameter optimized along the test locations (violet line), RQ kernel with α parameter optimized along the test locations (brown line), and RQ kernel with $\alpha = 1$ (pink line). Optimized test locations were expected to perform well, which is in conformity with the observations from Figure 4.13. They are lagging behind MMD by only two or three degrees of angle in achieving similar certainty of the test. However it has to be noted that the optimized value of α is very close to 1 too, which is most probably an effect of the L-BFGS-B method failing to converge in this parameter, which also explains why the brown and pink lines overlap. Nevertheless, while an unoptimized RQ kernel in combination with random test locations made the UME test results unusable (red line), the same unoptimized kernel performs very well with optimized test

locations. That means that optimizing the test locations is very important and may create a difference between very good and unusable results.

The last case, lagging about another two degrees behind UME with optimized test locations, is the UME test with a RBF kernel scaled using the median distance between sample points (green line). Even this variant shows the p-value close to zero for angles of 6° or larger. It is curious how this variant performs so much better compared to the unoptimized RQ kernel using the same random test locations. Apparently the median distance heuristic estimates the kernel bandwidth well enough compared to the arbitrarily-sized RQ kernel, for which no such heuristic is known. A variant with a RBF kernel with default $\sigma^2 = 1$ hasn't been explored, because is has been shown in previous sections that such kernel would be useless even on much simpler datasets.

4.4 MNIST Test Locations

In this section, we will explore how the test location optimization procedure works on MNIST data.

Similarly to Section 4.3, R will denote the reference set of digits 4, P the other set of original digits 4, and let P_β^{R} be the set of images of 4 from P, where all images are rotated anticlockwise about their respective centres by the angle of β. The null hypothesis will be $\mathrm{UME}^2(P_\beta^{\mathrm{R}}), R) \leq \mathrm{UME}^2(P, R)$, which should ideally be rejected for any $\beta > 0$.

4.4.1 Power Criterion at Digits

To predict the shape of optimal test location of an UME test, the value of power criterion for the aforementioned test will be measured at various candidate locations, which will be various MNIST digits set aside from the main dataset. Although R, P and P_β^{R} contain only digit four and its rotated variants, the power criterion will be measured even at all other digits. However it is expected that images of digit four will be a better indicator for difference between normal and rotated digit four, so it is expected that the power criterion will be higher at fours than at other digits. The rotation angle used for this experiment will be $\beta = 5°$, which is enough to be detected by UME and MMD goodness of fit tests with most choices of kernel, as shown in section . Gaussian kernel with bandwidth optimized for maximal test power will be used as the kernel.

Although the images are encoded as 728-dimensional vectors, the handwritten digits occupy only a small manifold from the space of all 728-dimensional vectors. If all possible images 28×28 would be generated (assuming discrete and finite pixel intensity values, e.g. 0 to 255), most of those images would be just noise. Several methods exist which discover that manifold and find a mapping to lower-dimensional space, such that the properties of the vectors in the high-dimensional space also hold in the

low-dimensional space. For display purposes, the task is to find a mapping to two-dimensional space, so that it's possible to display the result using a scatterplot. The mapping should preserve relative distances between the points, meaning that two images, which are very similar to each other, will be mapped to 2D points which are close to each other, and two dissimilar images, e.g. image of digit 2 and of digit 6, will be far away from each other in the scatterplot.

One such method which is able to visually separate all MNIST digit classes nicely is T-SNE [25]. This method is specifically intended to be used for visualization purposes, i.e. mapping to two or three dimensions. It has actually been tested on MNIST dataset in the original paper with very good results. It not only finds the low-dimensional manifold, but also separates different classes (digits 0 to 9), revealing the fundamental structure of the dataset.

The measured power criterion at each digit is pictured at Figure 4.14. Each point represents an image of a digit, its position is determined by T-SNE and has no useful interpretation apart from the fact that points representing similar images are close to each other and form a compact cluster. The left subfigure shows that most images with the same label are indeed plotted to similar coordinates. The label is colour-coded and we can clearly see clusters of individual colours without much overlap. The right subfigure shows the same points, but this time the colour represents the value of power criterion measured at that point, using the methodology described earlier.

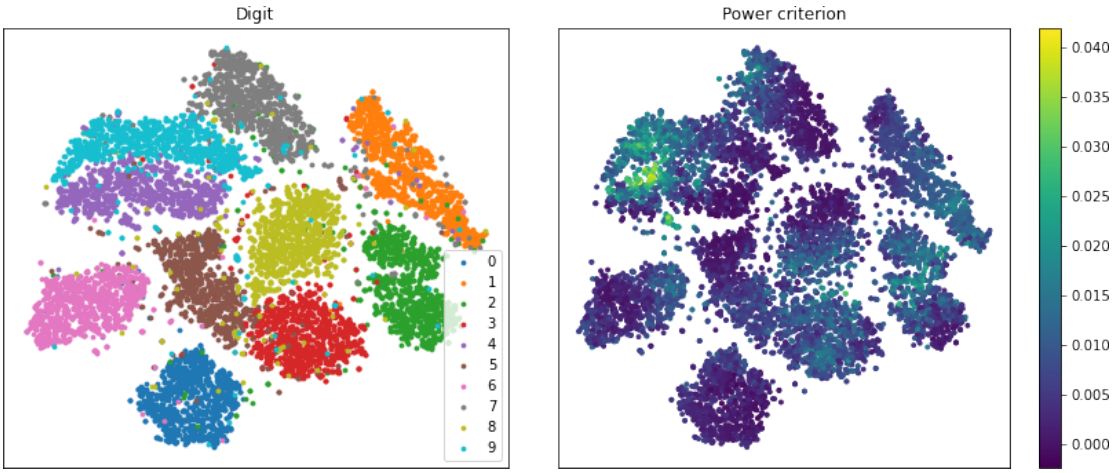


Figure 4.14. Power criterion of various digit samples from MNIST for UME test with $H_0 : \mathrm{UME}^2(P^{\mathrm{R}}_{5^\circ}, R) \leq \mathrm{UME}^2(P, R)$

In agreement with the expectations, the highest power criteria were measured for digit 4, which are closest to the data used in the UME test. Nevertheless, fairly high values were measured also in some other areas of the space of all handwritten digits. Cluster labelled 9 also contains many samples with high power criteria. Also, this cluster is quite close to the cluster of fours and those two overlap slightly. The reason

behind this is that in MNIST, many users write digit 4 with rounded top part. This, in combination with the stems of digits 4 and 9, which look basically the same, causes those two groups to be similar. Digit 9 differs from a commonly styled digit 4 only the very top part of the digit.

Less prominent samples with high value of the power criterion can be found in almost all clusters and the connection to digit 4 is not obvious. Digits with overall low power criteria are 5, 6 and 0. It can be argued that those would be bad UME test locations because they all have rounded bottom part, due to which they are dissimilar to digit four. However, the same applies to digit three, but on average, power criteria in cluster 3 are comparable to other digits.

The top 10 digits with highest and lowest value of power criterion respectively are shown in Figure 4.15. Interestingly, many of the low-valued samples are also digits four, which generally have high power criterion and are most informative for testing, because they are closest to tested data. However, those fours have in common that they are quite heavily tilted right. That's the opposite direction that we are testing. Probably this rotation in opposite direction makes them differ more from the other data, rendering them less reliable when used as test locations.

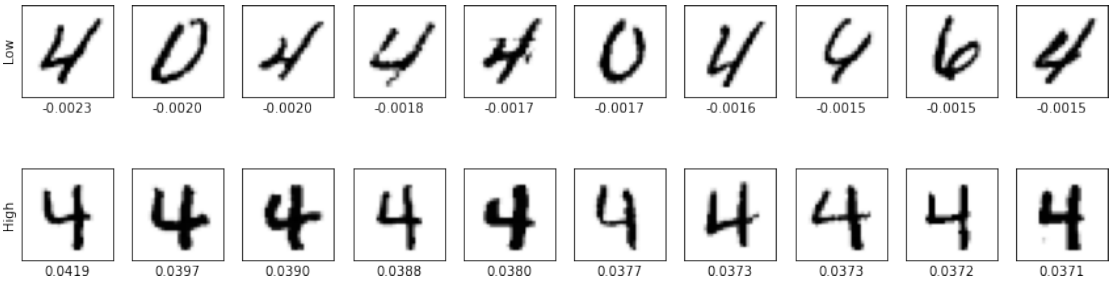


Figure 4.15. Top 10 digits with highest measured power criteria (top row) and lowest power criteria (bottom row).

On the other hand, the top digits with highest power criteria are all fours, as expected. Moreover, they are all relatively upward standing and also quite thick. The large width of their strokes causes them to be more similar to an average digit four, because as different people write then at slightly different angle, the averaged version would smear those differences and it would appear as thicker strokes. Such an averaged image should lead to a high test power, because it is in the centre of the space of all images of digit four, so it should be optimally similar to all of them. However, as the following section shows, the average digit doesn't outperform most of the samples in power criterion value.

4.4.2 Power Criterion Depending on Angle

Let's look closer at the power criterion of the average digit four, which is 0.0019. About 80 % other digits in MNIST have higher value of power criterion, so the expectation

of it to be exceptionally well suited to be used as a testing location was not fulfilled. Nevertheless, the average digit four can be rotated by the same angle as the part of the data used in the UME test, and we can compare the power criterion of those two versions. What's more, the power criterion will be measured on linear interpolation between those two. Let F be the normal image and F^{R} the rotated one. Power criterion will be measured for images $(1-\delta)\cdot F + \delta \cdot F^{\mathrm{R}}$, where δ is the interpolation coefficient ranging from 0 to 1 (both inclusive) in some appropriate steps. The result can be seen in Figure 4.16 along with images of F on the left and F^{R} on the right.

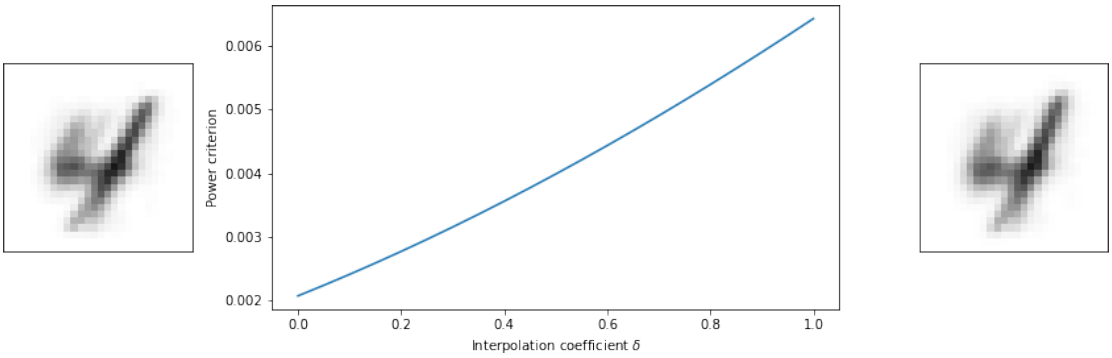


Figure 4.16. Power criterion of interpolation between normal digit four (F) and rotated by 5° (F^{R}). The interpolation is computed as $(1-\delta)\cdot F + \delta \cdot F^{\mathrm{R}}$.

As the rotated image is closer to the rotated set $P^{\mathrm{R}}_{5°}$, the test finds more evidence for null hypothesis rejection there. In this case the observation follows the intuition that the rotated image will have higher power criterion, because it better captures difference between the candidate distributions. The same trend can be seen when interpolating individual digits from the dataset from their natural form to the rotated form.

Let's extend the idea of measuring the power criterion of a rotated digit to larger angles. The digit four with the highest power criterion of all digits (drawn at the bottom left corner of Figure 4.15) will be rotated and its power criterion will be measured. It is expected that there will be a peak at angles close to zero, and it will smoothly transition to some low value for larger angles. Digit four rotated substantially may still be more similar to a real handwritten digit than random noise due to the strokes of which the digit consists, but there is no reason for it to be a better test location than some random digit from other class than four.

The result can be seen in the top row of Figure 4.17. This particular image is nearly optimal rotation-wise, the maximum lies only a few degrees from it. The function is smooth as expected and there is no other significant peak.

We can explore how big role symmetries play in this experimental setup. Digit eight is rotationally symmetrical and although most people write the upper loop smaller than the lower loop, when rotated by 180°, the result is still clearly readable as number eight for a human. Therefore we would expect the power criterion function be periodical

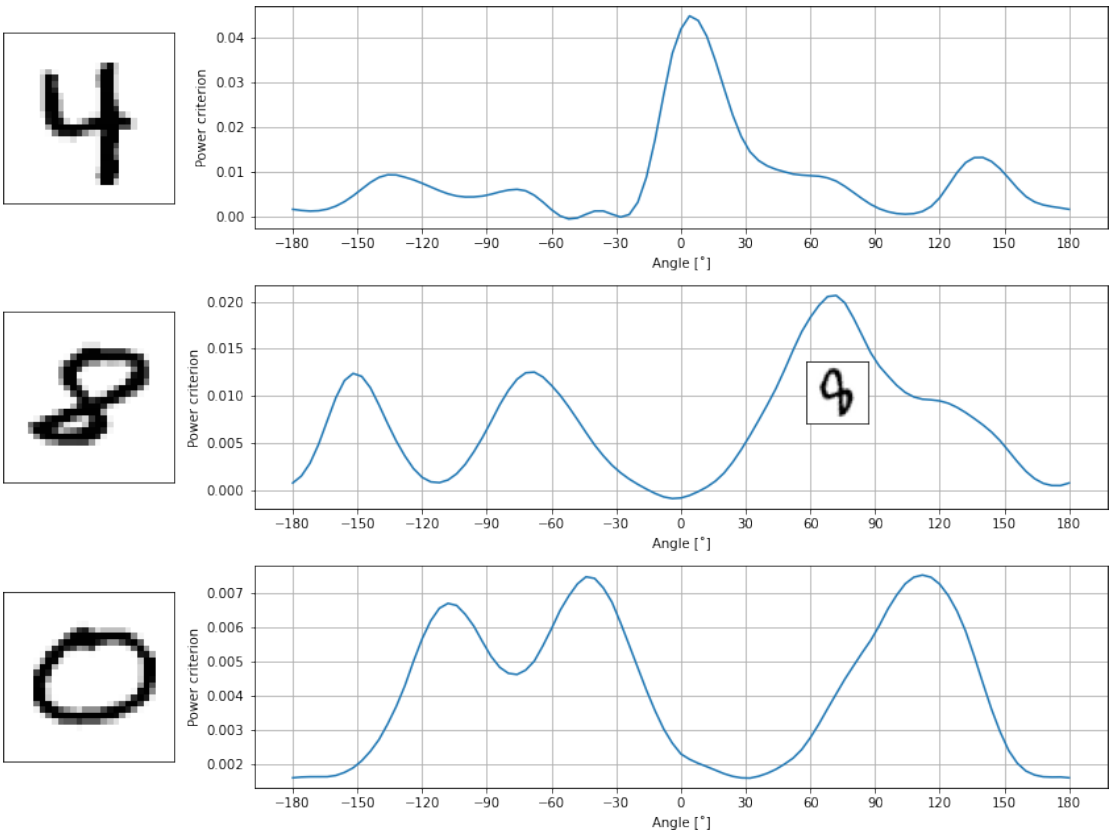


Figure 4.17. Power criterion of rotated digits 4, 8 and 0

with a period of 180°. As we can see in the middle row of Figure 4.17, this is not the case for the chosen digit. There are local maxima at approximately −150°, −70° and a larger one at +70°. We can argue that the −70° peak corresponds to the heavy right tail at +110°, but otherwise those extremes are not periodical.

A reason for that may be that we are still testing only digits four and their variant rotated by 5°, so the results for digit eight are distorted. Also notice that the original image at 0° has very low, even negative value of power criterion. Some other angles, however, lead to the power criterion comparable to some digits four from the dataset. The global maximum at +70° is particularly interesting. Image of the sample digit eight rotated by this angle is drawn into the plot for better notion. From this angle, the image has quite much in common with the best digit four above. Notice how both the horizontal and vertical lines of the digit four are there in the image of rotated digit eight. The only large-scale difference is the top part of the digit, where the eight loops, whereas the four has a blank space there, and a small part at the bottom. This illustrates that although the smaller peaks can not be explained intuitively, the power criterion measures real properties of the data, because the angle which maximizes the power criterion is conceptually very similar to what we expect to be a good test location.

For digit zero, especially such nicely drawn as the one in the lowest row of Figure 4.17, there are many symmetries, and we would expect the power criterion to be approxi-

mately constant in all angles. As we can see in the plot, this also isn't true. This time, however, power criterion reaches low numbers compared to the previous cases. It seems that this digit is so far from digits four, which we are testing, that all the angles will be equally bad. None of the maxima is significant enough to be interpretable as some feature of the image. In this case the situation may be different if the tested hypothesis would include data with zero label instead of four.

4.4.3 Optimized Test Locations

Now that it has been verified that power criterion is meaningful, continuous changes to the location lead to continuous power criterion function, and samples with high value of power criterion match the intuition why they are good informative test locations, the power criterion function can be used to optimize some random test location to maximize the overall test power.

This set of experiments will use the same null hypothesis and data as in the previous sections. Candidate data P with label four will be tested to match some reference set of fours R, The alternative candidate distribution P_β^{R} will be created from P by rotating each sample (while it is a 28×28 pixel image, before converting to a flat 784-dimensional vector) by angle β around its centre at $(14, 14)$. The angle β used will be $5°$, which is a value known to be reliably recognized by both UME and MMD tests with most kernels.

A small set of digits four will be held-out for the purposes of test location optimization. In this experiment, 20 digits from this set will be used as initial testing locations. In theory, any image of proper size can be used as a testing location. Digits labelled four were chosen on purpose, because they tend to have high value of power criterion on their own, as shown in Section 4.4.1. Those initial locations are very similar to the tested datasets, as they actually come from the same distribution as sets P and R. Those initial locations will then be manipulated all at once in order to maximize the power criterion.

The authors of UME goodness-of-fit test used a L-BFGS-B algorithm to ascend the power criterion function provided by *SciPy* Python package. L-BFGS-B is a quasi-Newton optimization method, an extension of BFGS method by bounding its memory usage and adding optional lower and upper bounds to optimized variables. The algorithm iteratively searches the variable space for a local optimum by approximating the Hessian of the objective function, which in our case is the power criterion function. The gradient of the objective function, which is needed as an input to L-BFGS-B, is provided by *autograd* Python package, which computes the gradient automatically without the need to specify it analytically by hand.

This method has worked well so far for simple datasets and optimization of a kernel parameter. However, disadvantages start to show up for more complex data such as MNIST images. First, the method has many hyperparameters which need to be set

and choosing wrong values can lead to poor result. Some of them are the number of iterations, the regularization constant or the bounds for each optimized variable. In this case the term variable as used by L-BFGS-B corresponds to a single pixel of the digit image, meaning that all initial test locations, already flattened to a 784-dimensional vector, have to be concatenated and each element of the resulting long vector will be treated as a single variable by the optimization procedure. If we want to optimize the kernel width or other parameters along with test locations, those parameters must be appended to this vector too, and after optimization, the individual components have to be reconstructed back.

The result of test location optimization in our experimental setting is shown in Figure 4.18. The top row shows the initial test locations coming from the held-out set of MNIST digits. Only some of them are shown to fit in the page. The middle row shows the results of optimization and the bottom row shows the difference between the two. Green colour means the optimization algorithm made the pixels more black, red colour means it turned the pixels lighter. Notice the scale of the changes mostly in the order of $10^{-}7$, which is negligible compared to the scale of the pixel intensities, which is 0 to 255 in this case. As we can see, no noticeable change has been made to the images despite the tuning of hyperparameters.

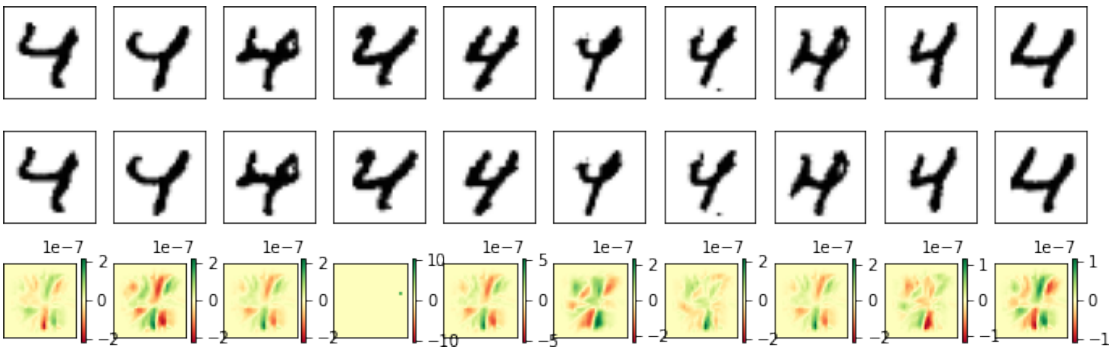


Figure 4.18. Test locations optimized by L-BFGS-B. Top row shows initial locations, middle row the final result, the bottom row is the difference between initial and optimized locations.

It has already been shown that test location optimization affect the test's ability to rightfully reject the null hypothesis substantially and is necessary for good results of UME test. It is obvious that for complex datasets such as images, which are the primary type of data consumed by generative adversarial networks, a more fitting optimization procedure is needed.

Adam is a modern stochastic gradient descent method based on adaptive estimation of low-order moments [26]. It is widely used as the optimizer in deep learning, so it has proven to easily handle a large amount of variables. Many of its hyperparameters generally don't need to be tuned for common problems and setting correct learning

rate is often enough for good result. We will use its implementation in *Keras* machine learning library instead of L-BFGS-B to find better UME test locations.

For MNIST dataset, learning rate $3 \cdot 10^{-3}$ and 2000 iterations at maximum will be used. The result for Adam optimization is shown in Figure 4.19. Again, the top row shows random initial locations, the middle row the final optimized locations and the third row the difference between the two. This time no scale is drawn individually, all images have the common scale from -255 (red) to +255 (green).

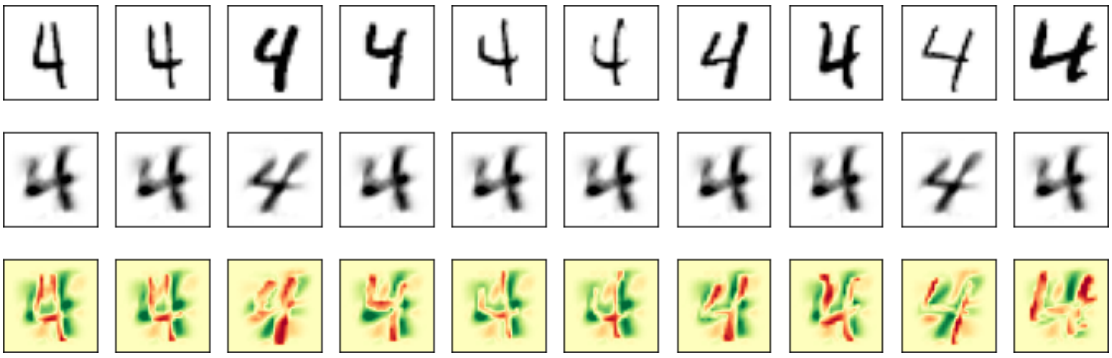


Figure 4.19. Test locations optimized by Adam. Top row shows initial locations, middle row the final result, the bottom row is the difference between initial and optimized locations.

After optimization, the results are cropped to 0–255 scale, because the optimizer has no notion of those constraints.

We can immediately see that the optimizer did a great job and had enough momentum to perform even drastic changes to the initial locations. While the average power criterion over the 20 test locations optimized by L-BFGS-B is 0.0017 with standard deviation of 0.0021, the locations optimized by Adam have mean power criterion of 0.046 with standard deviation of 0.020.

An interesting observation here is the fact that all locations seem to converge to one of two local optima, as presented in Figure 4.19. The more common thicker digit leaning towards left has power criterion about 0.055 and the other one, rotated more clockwise, has power criterion of 0.0013. We can see that the second one is actually not very great even compared to the initial locations, and that aligns with the expectation that good testing location should be rotated naturally, eventually slightly anticlockwise, because that's how the tested data samples look like. The first and more commonly found optimized location, however, has exceptionally high power criterion and should make an excellent test location.

This experiment results in several outcomes. First, Adam optimizer is more suitable for optimization of test locations, when the data is as complex as images. Second, there still exist a possibility of getting stuck in local maximum, which may be mediocre. However, in this case the algorithm found a local maximum that is very good and

possibly close to a global maximum, because visually it looks very similar to what we would expect to be a good test location – a thick digit four rotated upwards or slightly anticlockwise.

For digits four, it is known that they would be relatively close to optimal UME test locations, and it is generally a good heuristic to take samples from all three considered sets (reference set and two candidate sets). We may be curious about an arguably harder task for the optimizer – a situation in which there are less indices for what is a good test location. An example of situation in which one could face such problem is the case where there are many classes of the data, mutually distinct, and only a few test locations should be taken – less than the number of classes. In such case, sampling the data will inevitably lead to set of initial locations, in which some classes are not represented at all.

To check if the optimizer will handle well such a scenario, it will be simulated by setting the initial locations to a random noise in the same domain as normal MNIST data. That is, each pixel of the image will have a value uniformly sampled from 0 to 255 interval. These noisy images will be very dissimilar to optimal test locations or tested data, so they will be distant form them and the optimizer will need to travel longer trajectory to find some desirable optimum.

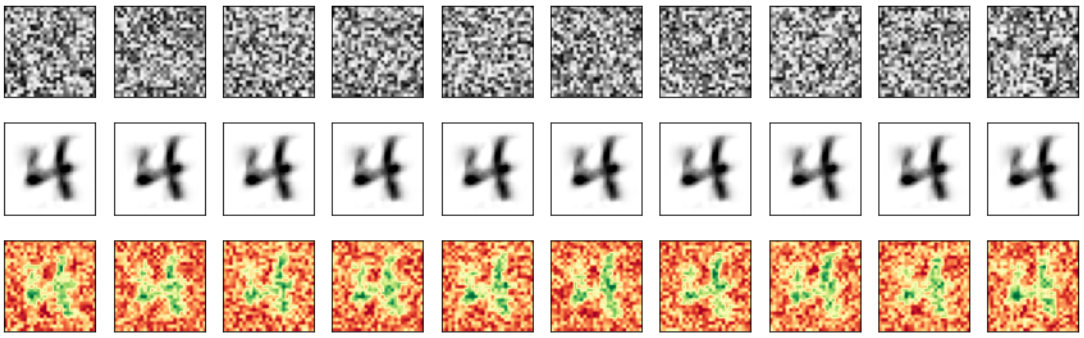


Figure 4.20. Random noise optimized by Adam as test locations. Top row shows initial locations, middle row the final result, the bottom row is the difference between initial and optimized locations.

As we can see in Figure 4.20, the optimizer found the same image as when the initial location were actual digits. Moreover, none of the 20 locations converged to the second local maximum, which had low value of the power criterion. All of them converged to the better one, which outperforms other test locations by far. No better test location in terms of the power criterion has been found for MNIST during my experiments.

An important observation is that with sufficiently powerful optimizer which could handle even random noise, it may actually be beneficial to start the optimization procedure from such a random location as opposed to starting from some known held-out data, because it allows to search the space of possible locations more directly towards

some significant maximum, avoiding smaller local maxima. Starting from something which already is a digit similar to those what are contained in the data sets means that the initial position is already quite good and lies close to some local maximum which may trap the optimizer. Apparently the power criterion function is complex enough to contain such local maxima that are significant enough to trap the optimizer, but very low in comparison with the global maximum.

The parameter of the optimization which is perhaps the most important to tune manually is the number of iterations, specifying how many times some gradient will be applied to current position in the search space. The optimization process can be nicely visualized by starting from some random uniform noise and allowing the optimization to proceed by more and more steps. During this process, we can oserve the current state of the optimization by looking at how the candidate test location changes its shape from random noise to something with more structure, and also by measuring the power criterion of the test location.

We would expect the power criterion to increase with the possibility of small temporary decreases in the case the optimizer would be smart enough to overcome some noisy local optimum. The problem will be still the same as usual, e.g. two sets of digit four and one set of rotated digits four. The kernel used in this experiment is a rational quadratic kernel with parameter α optimized independently before the measurements start, so the optimization affects only the test location and not the kernel shape.

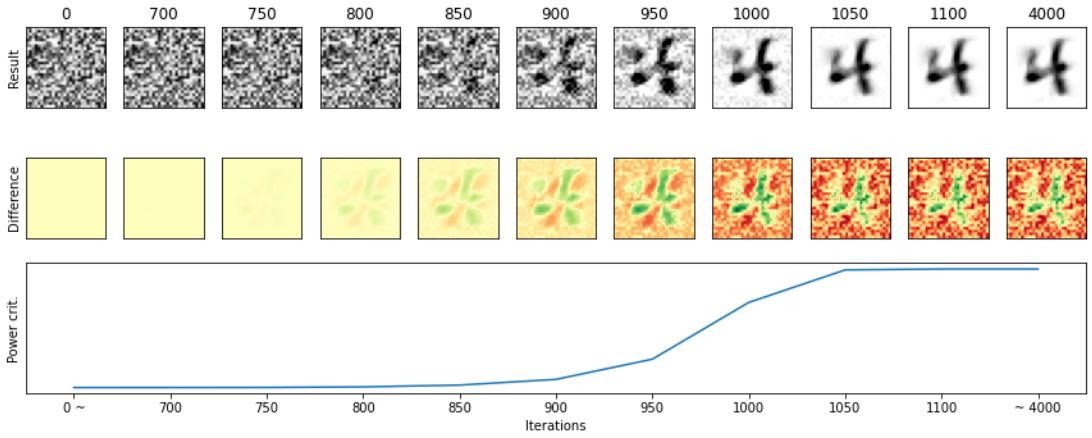


Figure 4.21. Progress of random noise optimized by Adam as a test location and power criterion measured at the optimized location.

The actual progress of the optimization of some randomly generated noise image can be seen in Figure 4.21. The top row shows how the test location looks like at the beginning when initialized, then being optimized for 700 iteration, then the transition from 700 to 1100 with a step of 50 iterations, and finally at 4000 iterations just to confirm that convergence has been reached at 1100 iterations. The specific number of iterations at which the test location starts changing visibly and at which is stops

changing is highly dependent on the initial noise, so this figure here only illustrates how Adam works and gives some insight to how the number of iterations should be set for similar problems.

We can see that the inner portion of the image is optimized first and the noise in the outer parts near the edges of the image disappear the last. This may happen because in the dataset, there never is any black colour in those outer areas. The gradient is stronger in the inner area, where difference between normal and rotated digits matter much more than near the borders, where nothing apart from some light noise does ever occur.

From a global point of view, there is a long period of time at the beginning where the changes are unnoticeable. This is expected, because when starting from uniform noise, the initial test location is very far from the optimum. As we confirmed in Section 4.1.1, in places far away from the distribution of data, the power criterion function holds some very low value and is flat, so methods based on gradient descent such as Adam will progress slowly until the gradient intensifies. That's why it takes relatively many iterations to see some changes, and when getting closer to some general shape of digit four, the progress speeds up and converges quickly to the maximum.

Chapter 5
Experiments with Generative Adversarial Networks

The main goal of generative adversarial networks is to learn some source distribution based on large amount of training samples took from that distribution, and be able to sample the learnt distribution by generating data similar to the training data. The most common type of data for GANs are images encoded into matrices of pixel intensities, sometimes flattened to vectors. The dimensionality of such data is high and may be further increased by increasing the resolution or adding colour channels.

Natural questions which arise after training of some GAN model are if the model learnt the distribution well, what are the samples which were learnt poorly and are either underrepresented in the model's output, or are too different from any sample from the training data. When having two different models, we can compare them and ask whether one model is better than the other, alternatively which part of the reference distribution are learnt better by one model than the other. For all those questions, the answers should be quantified and have some statistical guarantees.

Those questions can be conveniently answered by MMD and UME measures and by goodness of fit tests based on them, introduced in Chapter 2. Thanks to the interpretability of position of UME test locations optimizer for maximal test power, the questions examining local differences in trained distributions can be answered too.

In this chapter, possibilities of MMD, UME and associated goodness of fit tests will be evaluated on real output of generative adversarial networks with general architecture outlined in Chapter 3. Hyperparameters and detailed architecture of used models will be described, the quality of output will be discussed and generated samples will be evaluated by MMD and UME to see if they measure interpretable features of the distribution and if they can be used even during GAN training for some tasks.

5.1 GAN Architecture

The architecture of models used for experiments in this chapter will be Wasserstein GAN with gradient penalty, which has been roughly described in Chapter 3, Section 3.3.

MNIST dataset used as training data will be resized from 28×28 pixels to 32×32, so that the edge has a length which is a power of two, which is more convenient. The pixel intensities, so far ranged in $[0, 255]$ interval, will be rescaled to $[-1, 1]$ interval,

and if the trained network produces a pixel value outside of this range, it will be simply clipped to ± 1.

Another dataset used for evaluation will be CelebA [27], a dataset of faces of celebrities. It consists of 202 599 colour images of faces with some background. For performance reasons, the dataset will be simplified before usage. Only first 100 000 images will be taken, converted to monochrome and rescaled so that the shorter side of the image fits into 64 pixels. Then, the image will be cropped and only the central 64×64 pixel part will be used. As with MNIST, the pixel values will be rescaled to $[-1, 1]$ interval.

The GAN is implemented in python using tensorflow framework and attached to this thesis. The hyperparameters described here were set by a manual trial-and-error process reacting to imperfections in training. Minibatch training is used with batches of 64 samples in size.

5.2 GAN Generator

The generator part for MNIST dataset starts with a noise vector of length 100, each element of this vector is sampled from a standard normal distribution. The vector is being fed through one dense layer followed by four transposed convolution layers.

The dense layer consists of $16 \cdot 1024 = 16384$ neurons connected to all elements of the input vector. The output is reshaped to $4 \times 4 \times 1024$, which represents 1024 filters sized 4×4.

The next piece of the pipeline are three transposed convolutional layers, which quadruple the filter size at each layer, while halving the number of filters. There are thus 512 of 8×8 filters at the first convolutional layer, 256 of 16×16 at the second and 128 of 32×32 at the third. The final set of filters is transformed to a single 32×32 image by a convolutional layer.

All layers use leaky ReLU activation with the slope for negative values $\alpha = 0.2$, apart from the last layer, which uses tanh activation. The inputs of all convolutional layers is batch normalized. The three transposed convolutional layers use a 3×3 kernel with a stride of 2 in both dimensions and "same" padding, whereas the last, forward convolutional layer use a 7×7 kernel, "same" padding and stride of 1, because it doesn't change the size of the output.

The output is finally clipped to $[-1, 1]$ range, which is needed because there are no explicit constraints in the WGAN-GP itself, which would force the output to be bounded.

For CelebA dataset, a slight modification is made to the architecture described above. Since the images are 64×64 instead of 32×32, another transposed convolutional layer with 64 filters of 64×64 is added just before the final convolutional layer. Other parameters of the added layer are the same as of the other layers.

5.3 GAN Discriminator

The discriminator for MNIST dataset follows a different direction than the generator. Its goal is to determine if the 32×32 image is some output of a generator or a genuine sample of training data. The output should therefore be a single number.

Discriminator consists of three convolutional layers terminated by a single neuron densely connected to all outputs of the previous layer.

The convolutional layers use 5×5 kernel with stride of 2 and "same" padding, which shrinks the filter size into quarters each time. All use leaky ReLU activation with negative slope $\alpha = 0.2$.

The first layer takes a 32×32 image as an input and produces 64 filters sized 16×16. The second layer produces 128 filters of 8×8 and the third one 256 filters of 4×4. The output is then flattened and forwarded into a dense layer, which is just one neuron outputting a single number as the result.

Discriminator for CelebA dataset adds another convolutional layer before the first one. It takes 64×64 images on its input and produces 64 filters sized 32×32, which is compatible to the input of the next layer. Other parameters of the added layer are the same as of the other layers.

5.4 GAN Training

The network will be trained for 300 epochs, which is much more than needed for good results. We deliberately overtrain the network for multiple reasons. First, we want to have visual confirmation that the network will have converged and the output doesn't slowly improve over time. If we manage to estimate since what point in time the output of the network reached its peak quality, we may compare it to MMD and UME measured between generated images and real images and expect to see a plateau or at least an elbow in the MMD/UME plot near the epoch at which the generated images cease to improve.

A sample of images generated by our trained GAN after 300 epochs from uniform 100-dimensional random noise vectors is in Figure 5.1. During training, those random vectors are retained and passed through the current generator. This way, improvement of each generated digit can be tracked individually, which allows for more accurate visual inspection of output quality. The number of epochs was determined by this methodology.

We can see that most of the digits are accurate, some of them even indistinguishable from real handwritten digits. There are some samples which seem to be a mixture between two similar digits, e.g. third sample of the second row, which is close to a 3, but may become a 5 by slight modification. Those mixtures don't occur very often

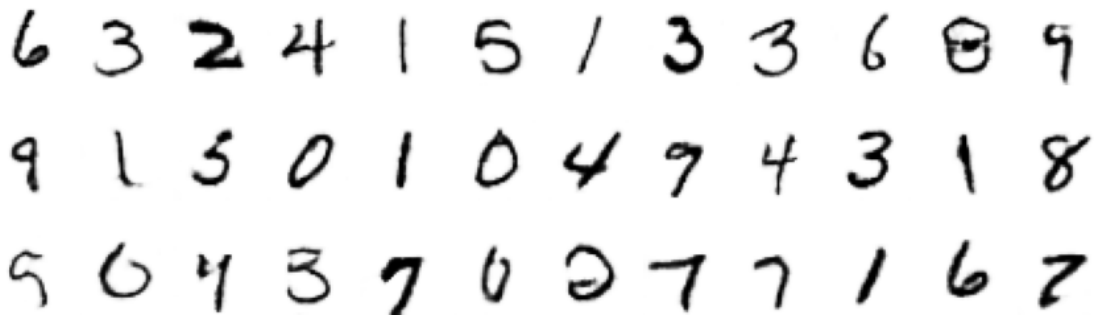


Figure 5.1. Sample of digits generated by the GAN starting from uniform 100-dimensional random noise vectors

though. We can also see that some classes are underrepresented, e.g. 2, but there are too few samples in Figure 5.1, so it may not be representative of the learnt distribution.

Even with above imperfections, the generated samples are generally of high quality and Wasserstein GAN with gradient penalty is powerful enough for this dataset. We have experimented with simpler models, which WGAN-GP is based upon (WGAN with gradient clipping, DCGAN), but they failed to produce images of comparable quality to real handwritten digit images, which is why WGAN-GP has been chosen for experiments.

Note that even though the Wasserstein GAN consists of a generator and a critic, we will use the term "discriminator" instead of "critic" for reasons described in Section 3.3.

The progress of the training, more specifically the loss functions of generator and discriminator, are captured in Figure 5.2. The loss functions are evaluated at each training step, which is processing of a single minibatch consisting of 64 samples of training data. The loss measured at specific point in time is plotted as a single blue dot and the x-axis is relabeled to represent epochs instead of training steps for better comparison with other plots. The black line running through the points is a rolling average of the loss function with a window size of 200 for generator loss and 1000 for discriminator loss. Due to the small size of the training batches, the loss function will naturally have relatively high variance, so rolling average will capture the trend of the loss function more obviously. It should be noted that discriminator is trained five times as often as generator, which explains the larger number of blue dots in the plot as well as the larger window for a rolling average. The reasoning behind this has been covered in Section 3.3.

A sign of successful training is that neither the generator nor the discriminator seem to overwhelm one another and they seem to keep balanced after some initial fluctuation. This is important, because if one would improve much faster than the other, it would most probably ruin the training process and the GAN would not learn the training data.

There's a noticeable bump in the generator loss function starting at 181. epoch, indicating that the network reached some optimum there, since it's also the overall

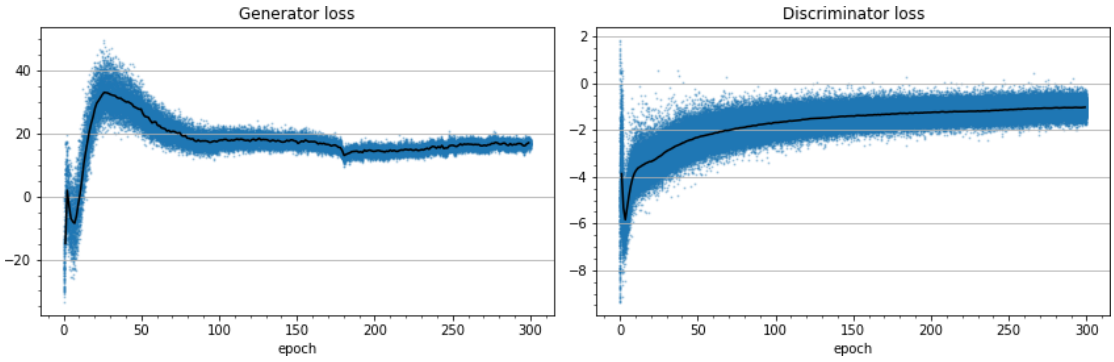


Figure 5.2. Loss functions of the generator (left) and discriminator (right) measured each training step

minimum of the loss function when we ignore first few epochs until generator and discriminator balance with each other. We will focus on that epoch later when measuring MMD and UME during training.

5.5 Measurements for MNIST

During training, the most natural use of MMD and UME is to measure their empirical variants between the GAN's output for some random input, and some reference set of images kept aside from the main training data.Since both MMD and UME can be interpreted as distance between two distributions. At the beginning, where the weights in the network are randomly initialized, the measured values should be high and as the training progresses, it should decrease, until the model is trained, at which point the measured values should stabilize around some low value (compared to value at the beginning of the training).

MMD will be measured between a reference set and the output of the GAN for some input, which is random, yet the same in each epoch. The reference set consists of 100 images of digits with equally represented classes, i.e. 10 samples for each digit 0–9. The experiment will be conducted separately for RBF and RQ kernel – the RBF kernel bandwidth will be optimized by median pairwise distance, the RQ kernel will have $\alpha = 1$.

The measurements for each epoch are in Figure 5.3. We can see that both kernel choices lead to similar shape of the curve, so the choice of one or another doesn't affect the result much in this case. The general shape is as expected, but very noisy. Even two subsequent epochs may differ by a lot in MMD. For that reason, the plots are smoothed by rolling average with a window size of 20 to better show general trend. The smoothed line reflects the progress of the training quite well – the visual quality of its output improves rapidly until epoch 50 and the changes are subtle after that.

The same measurements will be made using UME instead of MMD. The main difference in usability is the necessity to specify test locations. Ideally, the test locations

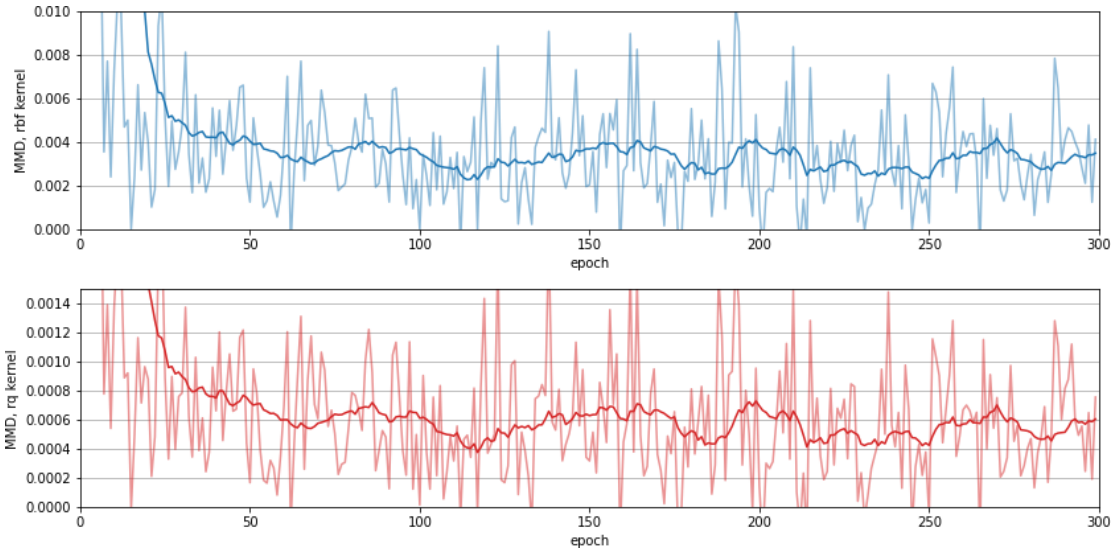


Figure 5.3. Empirical MMD^2 between the GAN output and real digits

would be optimized by power criterion measured against two candidate models. In this case, there is only one candidate model. This could be circumvented by simulating a second candidate model using some other set of data coming from a distribution identical to the reference distribution (real handwritten digits). Then, an UME goodness of fit test can be constructed by stating $\mathcal{H}_0 : \mathrm{UME}^2(D_1, G) \leq \mathrm{UME}^2(D_1, D_2)$, where D_1 and D_2 are disjoint sets of real images and G is the output of the GAN.

However, during training, G can't be used in practice, because the GAN hasn't been trained yet, so it couldn't be used to improve training in any way. Moreover, it has been discovered during experimenting that similarly to experiments with rotated digits, the test locations converge to one of two local optima, which are located near the centre of the data (average of all images), but since MNIST data is actually clustered into 10 classes, this central location is too far from individual data samples and measured UME is meaningless. For those reasons, the RQ kernel has the default value of parameter α, and test locations are selected either randomly by taking one sample from each class, or as mean digits, which consist of 10 images, of which each is a result of averaging multiple images from the same class.

Even with these limitations, measured UME seems to be in accordance with the training progression and has the same general shape as MMD. See Figure 5.4 for measured UME with both kernels and both choices of test locations.

It seems from Figures 5.3 and 5.4 that resorting to rational quadratic kernel with default value of its parameter didn't have dramatic negative impact on the measurements, because the shape of its line is very similar to the shape of the line of the optimized RBF kernel. Likewise, different choice of test locations for UME lead to similar shape, which may suggest that they are suitable for UME measurement, even though they

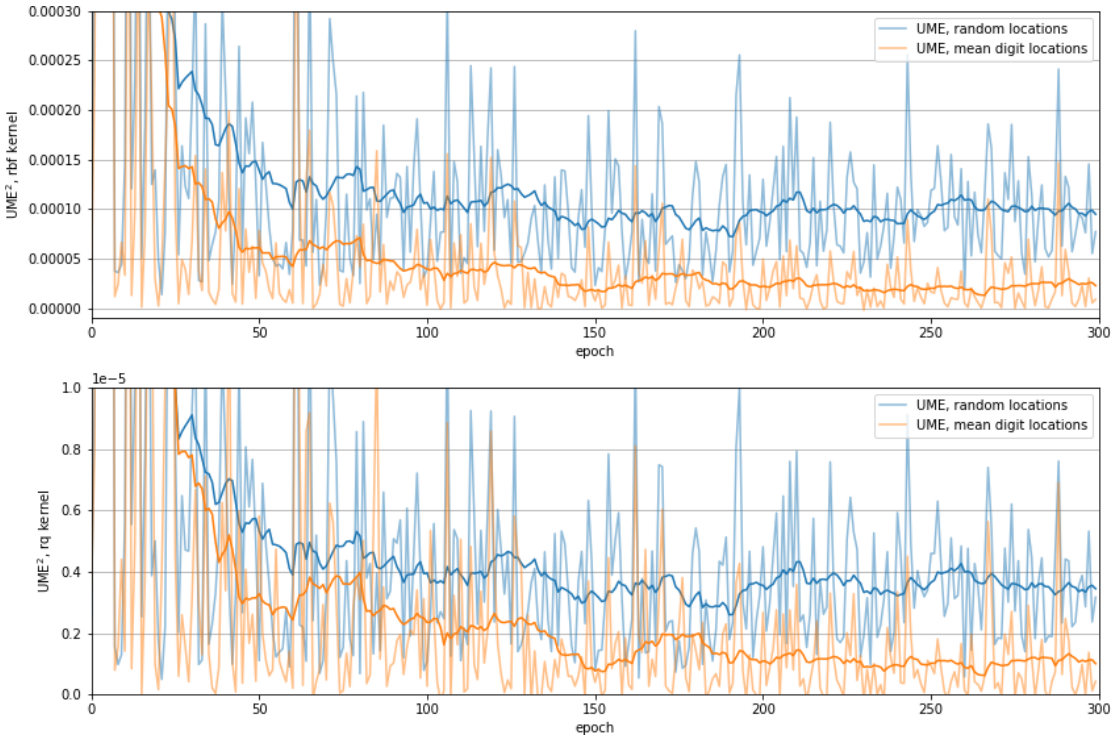


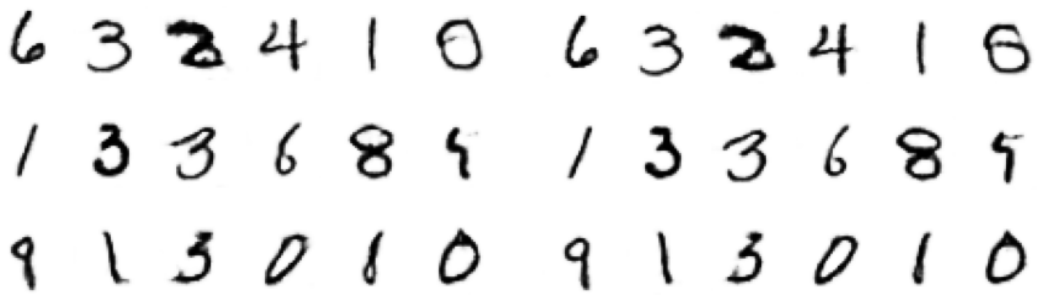
Figure 5.4. Empirical UME^2 between the GAN output and real digits

are not optimized in any way and optimization of kernel and test location was deemed important in experiments in Chapter 4.

MMD and UME plots have multiple areas where they have low value steadily, so it can't be easily said if or when the GAN reached the best output quality. If we should decide when the output quality stopped improving, it would be around 110. epoch based on MMD, and around 150. epoch based on UME. Both MMD and UME have somewhat low values around 180.–190. epoch, which is where the loss function of the generator had its minimum, and specifically at epoch 188, where even individual measurements of MMD and UME are low too.

Interesting cases are those which have low values (individually and ideally in the smoothed version at the same time) at some point, and increase later. That would mean that the model is starting to overtrain or otherwise degrade its performance. We can see such example in our measurements. One of the low points which is low both for MMD and UME using any kernel is aforementioned epoch 188, while some high bump occurs a few epochs later and reaches high value at epoch 211.

In Figure 5.5, we can see a comparison of outputs for the same input vectors at epochs 188 and 211. According to the measurements, we would expect the output at epoch 188 to be more similar to actual handwritten digits that output at epoch 211. However, the differences between the two are too small to confirm or disprove this assumption. We can argue that samples at the top-right corner are better in the left subfigure, because during epoch 211, they are probably in the process of changing classes – the zero at the



(a) (b)

Figure 5.5. Comparison of GAN output at (a) epoch 188 with low measured values and (b) epoch 211 with high measured values.

top is becoming a six or even an eight and the four in the last position of the middle row is becoming a five.

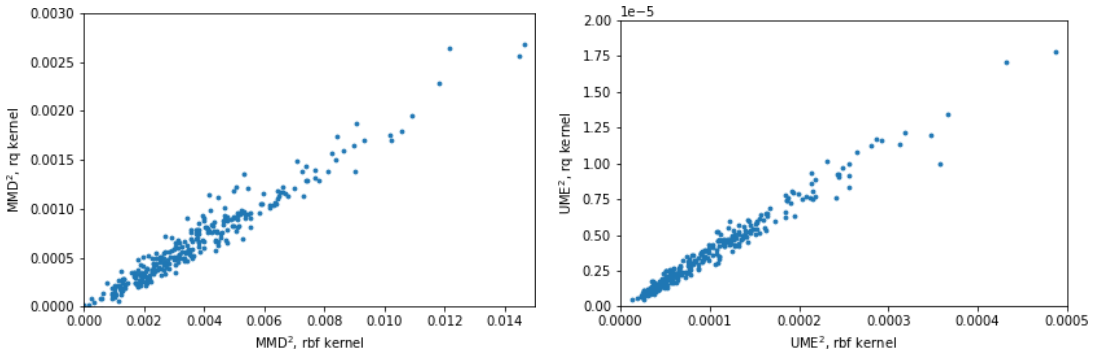
On the other hand, there are samples which are worse at epoch 188, e.g. the nine at the bottom-left corner or the 5^{th} sample of the bottom row. Subjectively there is very little difference in quality of those two cases, which limits the applicability of MMD and UME for cases in which the difference between measurements is small.

5.6 Reliability of MMD and UME

To increase the trustworthiness of the measurements, we may explore how MMD correlates with UME, alternatively if measurements with different kernels correlate with each other. It has already been shown that MMD and UME detected that the output of the GAN during first epochs was dissimilar to real data, and it improved dramatically during first 50 epochs and maintained approximately the same level of quality during later phase of training. However, it is hard to tell the extent to which they match in individual measurements

First, we will explore if the choice of kernel changes the measured MMD/UME more substantially than just changing the scale. In Figure 5.6, measurements are shown in a way that emphasizes the correlation of measurements done with a RBF kernel with measurements utilizing a RQ kernel. Each dot represents a measurement for one epoch and its placement on x- and y-axis corresponds to measured value using RBF and RQ kernel, respectively.

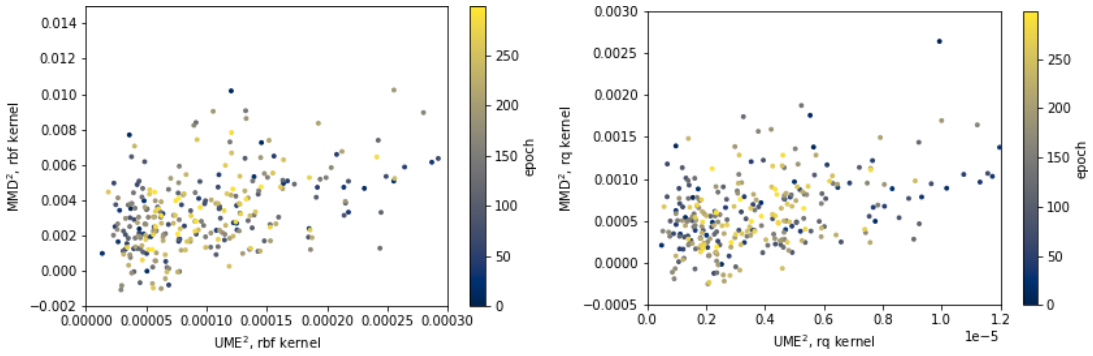
We can see that for both MMD and UME, all measurements lay near a diagonal, meaning that there is a strong correlation and choice of one kernel over another doesn't change the result much, other than changing the scale. That means that even though no optimization of kernel parameter was used for rational quadratic kernel, it is equally suitable as the RBF kernel set by median pairwise distance heuristic. This is surprising, because on simpler datasets, optimization of kernel was shown to be important for good results.



(a) (b)

Figure 5.6. Correlation of (a) MMD^2 with RBF and RQ kernel, and (b) UME^2 with both kernels; measured between GAN's output and real data at various epochs during training

It has already been shown that MMD and UME measured for each individual epoch is very noisy and two consecutive epochs may lead to unexpectedly different values, which is why smoothing is used for making conclusions. However, it is not clear if the noise is actually some property of the GAN output and individual epochs really differ that much as seen by MMD and UME, or if it is caused by inaccuracy of the methods itself. To help answer this question, measured UME may be compared with MMD in a way similar to Figure 5.6. In Figure 5.7, the measurements are represented by points again, but this time the x- and y-axis represents UME and MMD, respectively, with identical kernel used in each subfigure. The colour denotes the epoch, so the darker points correspond to early stages of training and should be generally placed in the top-right quadrant, whereas lighter points corresponding to trained GAN should be placed in the bottom-left quadrant, where both MMD and UME are low.



(a) (b)

Figure 5.7. Correlation of (a) MMD^2 with UME^2 with RBF kernel, and (b) MMD^2 with UME^2 with RQ kernel; measured between GAN's output and real data at various epochs during training

Here, the correlation is still present, even though not as strongly as in the previous case. The biggest problem are many samples having low UME even from early epochs, when the GAN output doesn't match the expected quality yet. This may be

a consequence of inaccuracy caused by random test locations as opposed to optimized locations, or it may be a property of UME measurement for this type of data. Either way it supports the suggestion to smooth out the measurements taken at each epoch and use that for making any decisions instead of relying on individual measurements.

Let's further explore if the noise present in both MMD and UME over the course of GAN training is caused by the method or if it measures some property of the data which is not obvious for a human. The same trained GAN as in the previous cases will be used, but this time MMD and UME will be measured at each epoch for 20 mutually different instances of random input. Those 20 instances will differ one from each other, but will remain the same during all epochs. This way we will be able to see if the measurements are consistent even for completely different samples which come from the same distribution (the same GAN at a specific epoch).

A rational quadratic kernel will be used. The sample sizes are the same as in previous experiments, i.e. 100 samples for MMD, 1 000 samples for UME and 10 test locations, one from each class.

The result is presented in Figure 5.8 for MMD and in Figure 5.9 for UME. Confidence intervals are shown as gradually decreasing saturation of the colour. The percentages of those intervals apply to the 20 sets of input samples, so a 80% interval means that 16 out of 20 measurements for a particular epoch fall within that interval. A mean value of all 20 measurements is shown as a black line.

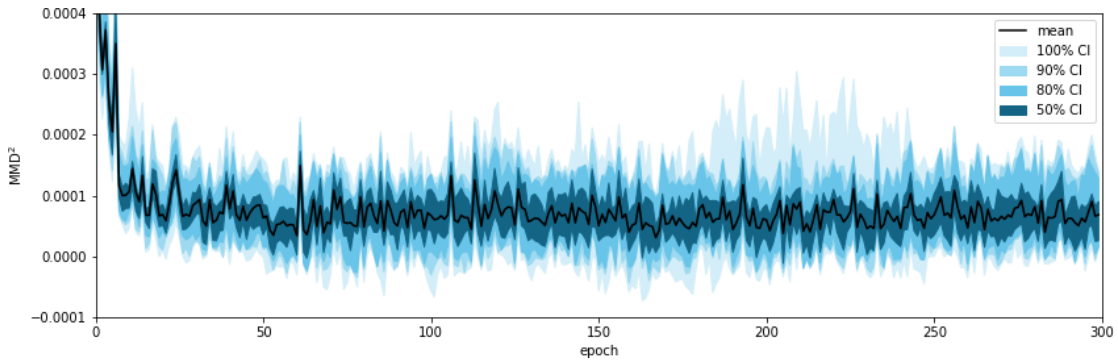


Figure 5.8. Variance of MMD when measured on multiple sets of apriori samples

We can see that MMD (Figure 5.8) has quite some variance, but a good sign is that most of the upward or downward spikes in high-percentage confidence intervals are accompanied by spikes in lower-percentage intervals and the mean value. That means that all measurements detected some property of the data which is exceptionally good or bad, rather than being just inaccurate measurements. In the context of GAN training, the spikes are still undesirable and should be smoothed out, because we want the model to be confident with its output for more than one epoch, but it's a great observation that MMD actually measures some sensible features.

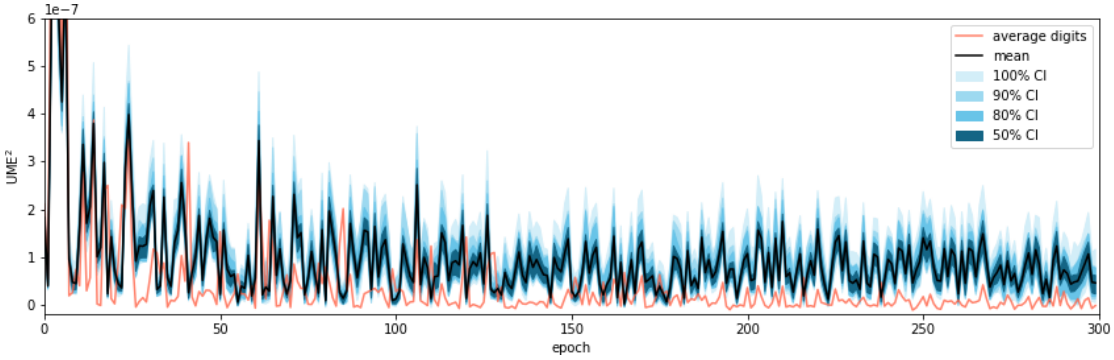


Figure 5.9. Variance of UME when measured on multiple sets of apriori samples

The main difference of the behaviour of UME (Figure 5.9) in contrast to MMD is that the variance is much lower relatively to the absolute values of the measurements. This is most probably because of larger sample pool (1 000 for UME vs. 100 for MMD), which can be used thanks to its linear computational complexity. Computing MMD on 1 000 samples from this dataset would take impractically long time and this computation would easily dominate the training time of the GAN.

The confidence intervals are plotted with random digits used as test locations. For comparison, a single set of measurements is shown with mean digits used as test locations in orange. Using the mean digits suffers more from falling near zero during whole process of training. It may be so because the mean digits, while being in similar distance from most data samples, are not close enough to any data sample at the same time. For that reason, the distribution of data is seemingly the same as the distribution of GAN output. Using random data samples as testing locations means that those locations are closer to some samples compared to mean digits, and further from some other samples, but what matters is that there are at least some data samples that are sufficiently close to some test location, that the kernel function returns non-zero values for those samples.

5.7 Measurements for CelebA

In section , several measurements were made on MNIST dataset. To confirm that the observations done during those experiments were not specifically tied to that one dataset, the same measurements will be repeated for CelebA dataset.

As mentioned earlier, CelebA dataset contains photographs of faces of celebrities. To decrease computational complexity, the dataset was preprocessed and simplified for purposes of this thesis. The images were desaturated to monochrome scale represented by pixel intensities from $[-1, 1]$ interval, identically to MNIST. At the same time, the images were rescaled and cropped to be 64×64 pixels in size. The rescaling is done in a way that preserves whole face including hair and only background or upper part of the body is cropped.

From quantitative point of view, the main differences between MNIST and CelebA are bigger size of CelebA images compared to MNIST (which is 32×32 pixels), as well as larger dataset. Full CelebA contains over 200 000 images, from which first 100 000 are taken for experiments and further split into smaller sets used for training, testing, test locations, kernel optimization etc. The largest chunk used for training contains 76 000 images. MNIST contains 70 000 images of which 54 000 are used for training.

Due to CelebA images having quadruple size of MNIST images, training takes much longer, which is why fewer epochs will be used – 100 instead of 300 used for MNIST. Nevertheless, we have made sure that it is trained excessively long and no changes in output quality are humanly visible in later epochs.

The final output of the trained GAN for some random input is shown in Figure 5.10. Although there are obvious deformations in some samples, most of the images really look like human faces. The network preserves high variety of facial expressions, hairstyle, skin colour and gender present in training data. It produces images from both left and right angle of view. Even the deformed samples retain basic facial features such as position of eyes, nose and mouth.

Figure 5.10. GAN output at last epoch

The training progress as visualized by loss functions of the generator and the discriminator is shown in Figure 5.11. They both look pretty normal, after some oscillation it stabilizes around epoch 35. The effective minimum of the generator loss seems to be right at epoch 50. We will compare it with MMD and UME measurements later.

Since there are no explicit classes in the dataset, the area of the space of all 64×64 images occupied by images of faces is expected to be relatively convex, without vast empty space between the samples. We can further assess the quality of the GAN by exploring its behaviour based on changes in the latent space. If we generate two random vectors suitable to be used as inputs of the GAN and interpolate between them linearly, we can observe how generated samples look like during intermediate steps of the interpolation.

More precisely, two 100-dimensional vectors x and y will be sampled from standard normal distribution. Vectors of the form $x \cdot (1 - \alpha) + y \cdot \alpha$ will be used as inputs for the trained generator for multiple values of α ranging from 0 to 1. For $\alpha = 0$ or 1, the

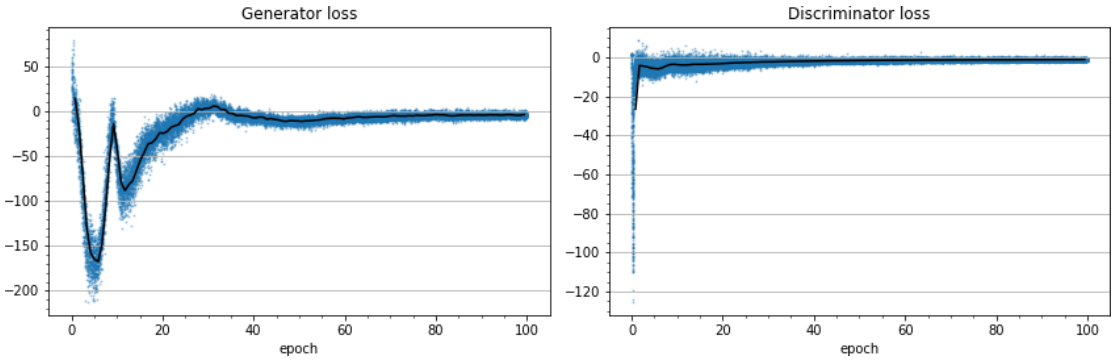


Figure 5.11. Loss functions of the generator (left) and discriminator (right) measured each training step

generated images should be some images of faces $G(x)$ and $G(y)$. For alpha between the border values, we would expect the generated sample to transition smoothly from $G(x)$ to $G(y)$ if the network is trained correctly.

In Figure 5.12, interpolation between five pairs of latent vectors are shown, each pair on its own row. We can see that the transitions in generated images are smooth and in most cases even an equal mixture with $\alpha = 0.5$ yield a result which come under the target distribution, i.e. it looks like a face too. This would not probably be the case in dataset with distinct classes such as MNIST, because e.g. a mixture of digit 7 and digit 5 in generated space is not a valid digit, whereas the same mixture of latent vectors corresponding to digits 7 and 5 should produce a valid image. That's why for MNIST, rather sharp transitions of images would be expected even for smooth transitions in latent space.

Figure 5.12. Generated samples for linear interpolation between pairs of latent vectors

Similarly to Section 5.5, MMD and UME will be measured between GAN output and a testing set of data at each epoch. Both considered kernels will be used. The RBF kernel's bandwidth will be set by median pairwise distance between data samples, the

RQ kernel will be used with its default parameter value $\alpha = 1$. The sizes of the testing data sets are 100 samples for MMD and 1 000 for UME.

Measured MMD is shown in Figure 5.13. It was able to definitely tell that the output quality improved dramatically over the first 20 epochs and there were just smaller changes from that point onward. Fortunately, even on this dataset, there seems to be only small difference between either kernels apart from the absolute range of measured MMD, which is in line with observations made on MNIST, that even unoptimized kernels are suitable for measure some features of real-world data.

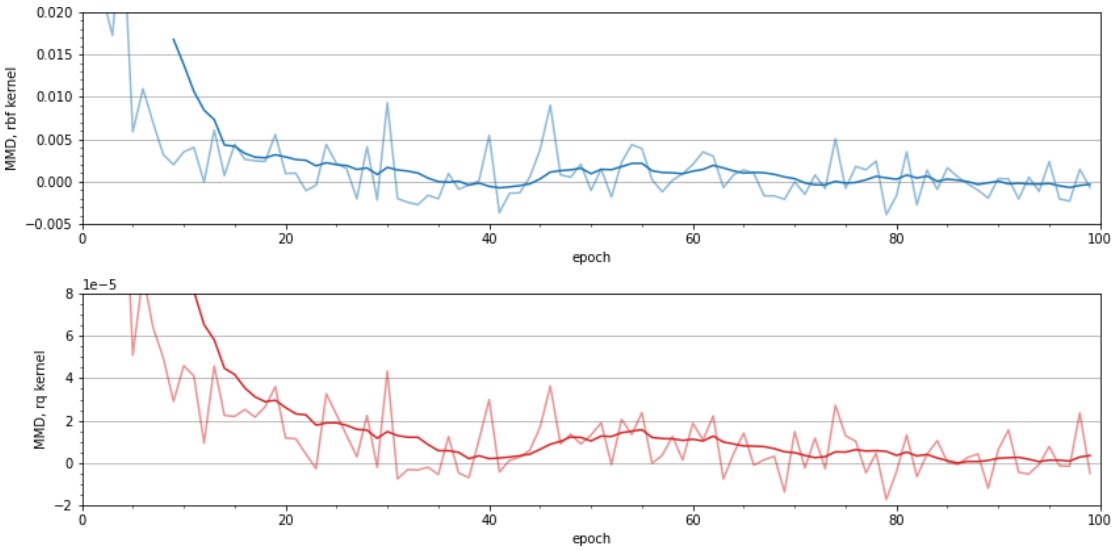


Figure 5.13. Empirical MMD^2 between the GAN output and real images of faces

A significant low-point of the MMD curve seems to be at epoch 41, which is one of the lowest points of the smoothed curve and also one of two lowest individual points, at least when RBF kernel is used.

For UME measurements, 10 test locations are taken randomly from the dataset (and not used for any other purpose). In contrast to MNIST, where we picked systematically one random digit from each class, there are no other constraints in CelebA.

Measured UME using both kernels is presented in Figure 5.14. Compared to MMD, it is very noisy and a lot of samples has value close to zero, which is a phenomenon already seen with MNIST dataset. The smoothed curve suggests that the GAN reached peak quality at epoch 25, there is however a less prominent low point (both smoothed and individually) at epoch 41, where MMD had its minimum.

There is no epoch with extraordinarily high values of MMD and UME at the same time, but at epoch 51, all values are reasonably high, so we can compare output of epoch 41, which had all values low, with epoch 51, which has most values higher. The comparison is shown in Figure 5.15.

The situation is similar to MNIST. Although the differences between the two epochs are more visible here, it is hard to assess which one is of higher quality than the other.

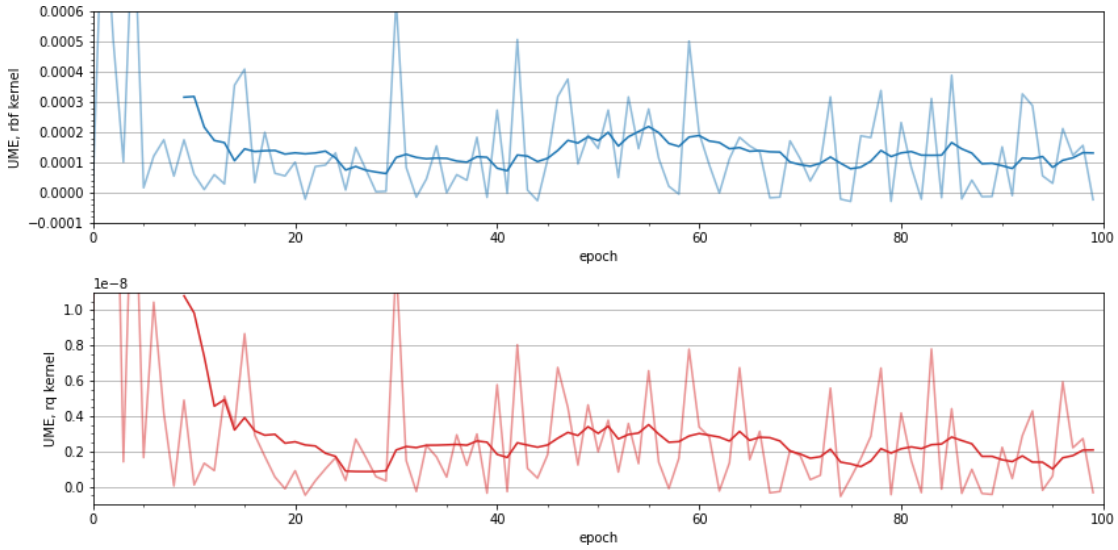


Figure 5.14. Empirical UME^2 between the GAN output and real images of faces

(a) (b)

Figure 5.15. Comparison of GAN output at (a) epoch 41 with low measured values and (b) epoch 51 with high measured values.

According to MMD and UME, the samples in Figure 5.15a should be more similar to training data, but visually both sets of images look similar in quality.

That means that caution needs to be taken when interpreting measured MMD or UME on complex data. Even though other evidence support that they measure some actual properties of the data, there is high variance in those measurements and the change in one of the metrics needs to be quite large to become visible in the output of the GAN.

5.8 Early stopping

In the context of artificial neural network training, *early stopping* is a general term for interruption of the training process earlier than what's the pre-chosen number of iterations. Certain stopping criterion is being evaluated each epoch, and if no significant improvement is observed during some period of time, the training is stopped. The evaluation is being dome on a separate set of data not used for training or measuring

performance of the model. The criterion function, amount of change considered an improvement, and number of epochs considered, are parameters specified by the user. The criterion function may be the loss function of the model or some statistic computed on model's output using validation dataset.

Early stopping may be beneficial for several reasons. It helps prevent overfitting, because it stops training when performance on a validation set stops improving, even if performance on training data continues to improve. It also saves time spent on research and model tuning, because it automatically sets the right amount of epochs necessary for proper training, which is not known before training, so a safe strategy is to set this number higher than necessary, which in turn wastes time and resources.

Instead of the (generator) loss function, MMD or UME can be used as the stopping criterion function. We've seen in this chapter, that raw measurements can't be used on their own, because of their high variance. However, averaging the measurements over a small sliding window of epochs, is representative of the similarity between the output of the GAN and reference data, even though not all captured values can be easily interpreted.

Apart from using smoothed MMD or UME along with appropriately set time frame and change in their values which will be considered significant enough to continue with the training, the goodness of fit tests described in Chapter 2 can be used for early stopping.

The test can be constructed as follows. Two reference datasets R_1 and R_2 will be needed, each one containing unique samples which are not used anywhere else in training or model evaluation. Let S be some constant seed, a set of appropriately sized random vectors suitable for being processed by the trained generator G. For convenience, let $G(S)$ denote a set of output images generated from each element of s, in other words $G(S) := \{G(s) \mid s \in S\}$. The null hypothesis for the test will be

$$\mathcal{H}_0 : d^2(R_1, G(S)) \leq d^2(R_1, R_2),$$

where d is a substituent for either MMD or UME.

Intuitively, the test is trying to reject a hypothesis that generated samples are more similar to real data than real data itself. Set R_2 simulates a perfect model, because it is sampled from the same distribution as reference data R_1. We would expect hat the test rejects the hypothesis at the beginning of the training, because undertrained network will generate images very different from samples in R_1. As the training progresses and generator improves, the test may stop rejecting the hypothesis, because generated output will become increasingly similar to the reference data. With GAN training, the ultimate goal is to produce output that is indistinguishable from real data, which means not only visually for human, but also for statistical tests of distribution equality.

The p-value of the test directly corresponds to the level of similarity between generated output and reference data and can be used as the quantity that will be evaluated and checked for stagnation.

For MNIST dataset, p-values of both MMD and UME tests with unoptimized RQ kernel can be seen in Figure 5.16. We can immediately see another evidence that using mean digits for test locations (green line) isn't ideal, because even during first epochs, the p-value is high, meaning that the test isn't able to reject a hypothesis that very poor output is better than real data, which it should reject quite easily.

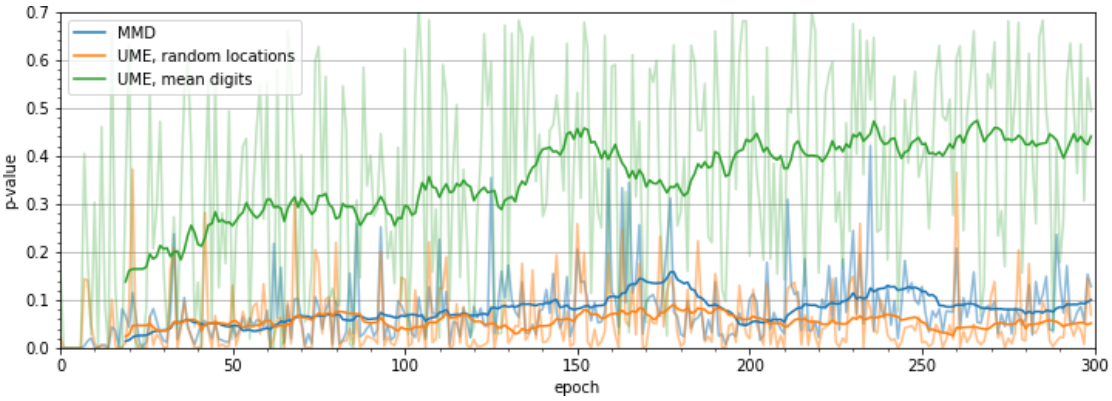


Figure 5.16. The p-value of the MMD and UME test testing if GAN's output is closer to real digits than another set of real digits

It is also visible that smoothed p-values of the MMD test and the UME test with randomly sampled test locations have similar shapes – when one increases, the other increases too. The correlation seems to match what has been observed in Figure 5.7. The peak value occurs around epoch 175, which is not far from where prior measurements indicated a moment when the network produced one of the best outputs.

The p-value of tests conducted on CelebA dataset are presented in Figure 5.17. The correlation of smoothed MMD with UME is obvious. It is clearly visible that UME is less robust in individual measurements, which are highly variable, which is in line with earlier observations that UME is prone to showing extreme values even when the data is changed only slightly. Clearly there is a maximum at epoch 41, which is a location detected also in earlier experiments.

From the comparison of Figures 5.16 and 5.17, the p-values lie in a different range. For CelebA, the p-values go as high as to 0.5, whereas they stay mostly below 0.2 for MNIST. It means that the range of values is still dependent on the dataset.

There are several advantages of using p-value of the goodness of fit test over plain MMD, UME or any traditional loss function. The value itself is interpretable and has a defined meaning. Even though we showed that the maximal values are dependent on the dataset and its complexity, p-value is always bounded by $[0, 1]$ interval. For these reasons, it should be easier for the user to set the particular values of the parameters of

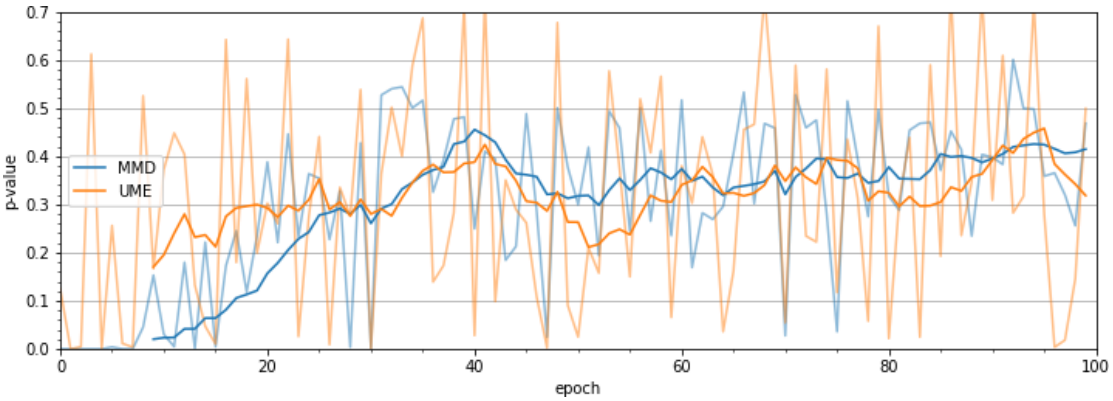


Figure 5.17. The p-value of the MMD and UME test testing if GAN's output is closer to real faces than another set of real faces

stopping. Moreover, additional constraints can be added to early stopping or accuracy assessment, based on the traditional utilization of the statistical test. For example, the training should not be stopped before some significance level is reached. Apart from early stopping, it can be used as an interpretable result of model evaluation on testing data.

Chapter 6
Conclusion

A number of experiments were conducted in order to explore the capabilities of maximum mean discrepancy and unnormalized mean embedding and goodness of fit tests based on them. It was shown that both goodness of fit tests are capable of detecting systematic changes on multidimensional data, such as all samples being shifted in the direction of one axis, rotation of the whole dataset, or if the samples are images, a rotation of the samples individually about as little as 5°. For those cases, both MMD and UME reported p-values such small that they can be considered zero.

Special attention was paid to UME test locations, which influence the reliability of UME significantly. It was shown that optimization of the locations has a positive effect on UME test certainty about its result. On the other hand, similar level of accuracy can be achieved by taking a sufficiently big subset of the data as test locations. The optimization criterion was visualized for simple problems and it was confirmed that local maxima of the criterion occur at intuitively correct positions, so the optimization is meaningful if executed correctly.

It was experimented with the impact of particular kernel choice on the result of the tests. Optimization of the kernel was shown to be beneficial similarly to the optimization of the UME test locations. If optimization is not possible for some reason, performance of a rational quadratic kernel was observed to be superior to the more widely used radial basis function kernel when differences between distributions occur at local level compared to the global structure of the data. However, if the discrepancies occur on a global scale, such as rotation or shift of the whole dataset, RBF kernel benefits from the possibility to set the bandwidth by a mean pairwise distance heuristic and outperforms the RQ kernel.

Behaviour of MMD and UME measured during GAN training was studied. Two Wasserstein GANs with gradient penalty were trained on two diverse datasets – MNIST and CelebA. It was shown that measuring one of the metrics between GAN's output and the reference dataset generally reflects the visual quality of generated images. However, MMD as well as UME are subject to high variance, so smoothing must be used to provide meaningful results.

Thanks to linear computational complexity of UME compared to quadratic complexity of MMD, larger reference dataset may be used for calculation, which makes UME much more robust to the choice of samples present in the reference dataset. On the other hand, it is more variable than MMD from the perspective of small changes

of the candidate distribution, such as the output of the GAN during two consecutive training epochs. UME often oscillated between zero and some high value for two sets of generator output, even though the difference between the two was minimal.

Visual comparison of generated samples for some low value of MMD and UME (smoothed and individually alike) showed very little improvement over generated samples with higher values, apart from the early phase of the training. Due to the variance occurring during GAN training, individual measurements probably aren't indicative enough to be used e.g. to highlight the most successful epoch or as a definitive criterion of quality. However, when averaged over several epochs, both are capable of capturing the rapid improvement of quality in the early epochs and modest development in later stage, denoting the approximate point of time since when the GAN has been trained and the quality of generated images didn't visually improve any more.

As both MMD and UME seemed to be able to detect when the GAN is sufficiently trained, a novel method of early stopping has been proposed. Using the goodness of fit tests based on MMD and UME, along with simulation of a perfect model by setting another set of real data aside, a sequence of p-values can be obtained, one for each training epoch. Because p-values have known range and are easily interpretable, the stopping criterion can be established more easily than on some values of a loss function. Early stopping based on MMD/UME test is independent of the training process, which is important specifically for GANs, where the loss functions of the generator and discriminator are dependent affect each other and may not be in accordance with the quality of the generator output.

Some of the limitations of MMD and UME shown during the experiments, such as the high variability, may be overcome in future research by optimization of the kernel and UME test locations during training. As the distribution learnt by the GAN changes each epoch, the optimization cannot be performed before the training starts. If the results of the MMD and UME tests remained comparable even if the kernel or the test locations would be optimized multiple times during the training, the measurements would be more robust and accurate, because optimization was shown to have a great impact.

There are several other possible uses of MMD and UME in GAN training. For example, *generative moment matching network* (GMMN) is a type of GAN in which the discriminator is entirely replaced with MMD measured between samples from the model and the training data. Such uses were out of the scope of this thesis, but GMMNs may benefit from some of the findings found in the course of this work. There is also an unexplored possibility to use UME instead of MMD to directly provide feedback to the generator. The necessity of providing suitable test locations and the instability of UME may make the training very challenging; on the other hand, it may be quickly computed on thousands of samples at once, which could be advantageous for the training.

References

[1] A Gretton, K. Borgwardt, Malte Rasch, Bernhard Schölkopf, and AJ Smola. A Kernel Two-Sample Test. *The Journal of Machine Learning Research*. 2012, 13 723-773.

[2] Kacper Chwialkowski, Aaditya Ramdas, Dino Sejdinovic, and Arthur Gretton. Fast Two-Sample Testing with Analytic Representations of Probability Measures. 2015,

[3] A. J. Smola, A. Gretton, L. Song, and B. Schölkopf. *A Hilbert Space Embedding for Distributions*. In: E. Takimoto, eds. *Algorithmic Learning Theory*. Berlin, Heidelberg: Springer, 2007. 40–41. ISBN 978-3-540-75224-0.

[4] James R Lloyd, and Zoubin Ghahramani. *Statistical Model Criticism using Kernel Two Sample Tests*. In: C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2015.

[5] Wacha Bounliphone, Eugene Belilovsky, Matthew B. Blaschko, Ioannis Antonoglou, and Arthur Gretton. *A Test of Relative Similarity For Model Selection in Generative Models*. In: Yoshua Bengio, and Yann LeCun, eds. *4th International Conference on Learning Representations, ICLR 2016, Conference Track Proceedings*. San Juan, Puerto Rico, May 2-4, 2016: 2016.
`http://arxiv.org/abs/1511.04581`.

[6] Wittawat Jitkrittum, Heishiro Kanagawa, Patsorn Sangkloy, James Hays, Bernhard Schölkopf, and Arthur Gretton. *Informative Features for Model Comparison*. In: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018.

[7] Arthur Gretton, Dino Sejdinovic, Heiko Strathmann, Sivaraman Balakrishnan, Massimiliano Pontil, Kenji Fukumizu, and Bharath K. Sriperumbudur. *Optimal kernel choice for large-scale two-sample tests*. In: F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012.

[8] Yujia Li, Kevin Swersky, and Rich Zemel. *Generative Moment Matching Networks*. In: Francis Bach, and David Blei, eds. *Proceedings of the 32nd International Con-*

ference on Machine Learning. Lille, France: PMLR, 2015. 1718–1727. http://proceedings.mlr.press/v37/li15.html.

[9] Mikołaj Binkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. *Demystifying MMD GANs.* In: *International Conference on Learning Representations*. 2018. https://openreview.net/forum?id=r1lUOzWCW.

[10] Carl Edward Rasmussen, and Christopher K. I. Williams. *Optimal transport – Old and new*. The MIT Press, 2006. ISBN 0-262-18253-X.

[11] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. *Generative Adversarial Networks*. 2014.

[12] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. *GANSynth: Adversarial Neural Audio Synthesis.* In: *International Conference on Learning Representations*. 2019. https://openreview.net/forum?id=H1xQVn09FX.

[13] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. *Generating Videos with Scene Dynamics.* In: D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2016.

[14] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. *Generative Image Inpainting with Contextual Attention.* In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018. 5505-5514.

[15] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. *CoRR*. 2016, abs/1611.07004

[16] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. *CoRR*. 2016, abs/1609.04802

[17] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv preprint arXiv:1511.06434*. 2016,

[18] Martin Arjovsky, Soumith Chintala, and Léon Bottou. *Wasserstein Generative Adversarial Networks.* In: Doina Precup, and Yee Whye Teh, eds. *Proceedings of the 34th International Conference on Machine Learning*. Sydney, NSW, Australia: JMLR.org, 2017. 214–223. http://proceedings.mlr.press/v70/arjovsky17a.html.

[19] Cédric Villani. *Optimal transport – Old and new*. Berlin, Heidelberg: Springer, 2008. ISBN 978-3-662-50180-1.

[20] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. *Improved Training of Wasserstein GANs.* In: *Proceedings of the 31st International Conference on Neural Information Processing Systems.* Red Hook, NY, USA: Curran Associates Inc., 2017. 5769–5779. ISBN 9781510860964.

[21] Wittawat Jitkrittum, Wenkai Xu, Zoltan Szabo, Kenji Fukumizu, and Arthur Gretton. *A Linear-Time Kernel Goodness-of-Fit Test.* In: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds. *Advances in Neural Information Processing Systems.* Curran Associates, Inc., 2017.

[22] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM Journal on Scientific Computing.* 1995, 16 (5), 1190-1208. DOI 10.1137/0916069.

[23] Diederik P. Kingma, and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR.* 2015, abs/1412.6980

[24] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE.* 1998, 86 (11), 2278-2324. DOI 10.1109/5.726791.

[25] Laurens vander Maaten, and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research.* 2008, 9 (86), 2579-2605.

[26] Diederik P. Kingma, and Jimmy Ba. *Adam: A Method for Stochastic Optimization.* In: Yoshua Bengio, and Yann LeCun, eds. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.* 2015.
`http://arxiv.org/abs/1412.6980`.

[27] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. *Deep Learning Face Attributes in the Wild.* In: *Proceedings of International Conference on Computer Vision (ICCV).* 2015.

Appendix A
Source code

The experiments have been programmed in Python language using Jupyter notebooks (`.ipynb` suffix). They can be re-run using appropriate environment, e.g. JupyterLab. The notebook used for GAN training is meant to be executed in GPU-accelerated Google Colab environment and connects to a private Google Drive for data storage.

Contents of the media:

- `assignment.pdf` – Official assignment of this thesis as a stand-alone document
- `thesis.pdf` – The thesis
- `src/experiments.ipynb` – Notebook with experiments with artificial data
- `src/mnist.ipynb` – Notebook with experiments with MNIST dataset
- `src/gan.ipynb` – Colab notebook with experiments with GANs trained on MNIST and CelebA
- `doc` – Various source files for the thesis document (mostly `.tex` files and images)