Beyond GOTEX: Using multiple feature detectors for better texture synthesis

Alessio Spagnoletti*

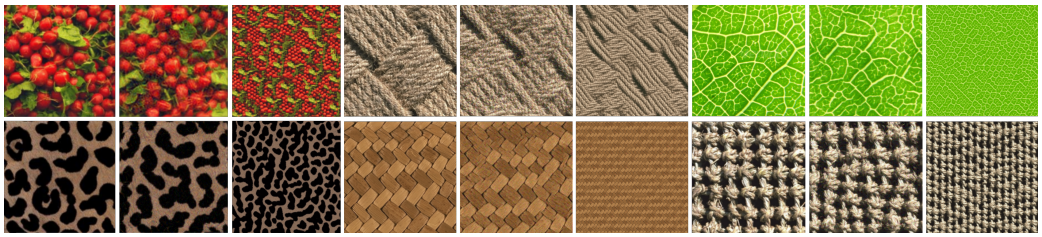


Figure 1: *In order: real textures sources and samples generated by Gotex-IV3 (both single image and generative case)*

Abstract

In this work, after a brief introduction to texture synthesis, we aim to present the paper [Houdard et al. 2021], describe its mathematical foundations, and analyze its effectiveness as well as its limitations. Furthermore, we will provide different combinations of the framework applied to various textures in order to tackle the issues that arise from each method. As our main contribution, we will see how the InceptionV3 [Szegedy et al. 2014] network pre-trained on the ImageNet dataset performs when used in the same way as the authors in [Houdard et al. 2021] use the VGG-19 network [Simonyan and Zisserman 2014], obtaining better results on a wide variety of textures.

1 Introduction to Texture Synthesis

Texture synthesis is the art and science of creating images that capture the intricate visual properties of surfaces and materials. This field has seen significant developments over the years, with contributions from various researchers and approaches. Here are summarized three steps onto which [Houdard et al. 2021] is based, as well as many other research efforts in this field.

The Julesz Conjecture

In the 1960s, psychologist Bela Julesz proposed a new way to describe and thus recognize textures. Instead of defining them as composed of specific and explicit patterns, its idea was centered on the statistical nature of textures. The idea is that our perception of texture is primarily driven by the statistical relationships between pixel values. He thus defined the Order Statistics (OS), defined as follows:

1st OS First-order statistics (OS) are quantified by assessing the likelihood that a randomly placed point on the texture will coincide with a black dot. To illustrate, imagine two white patches of equal size, each adorned with black dots of identical dimensions, forming a texture. If one of these patches contains a greater number of black dots compared to the other, the two textures will exhibit a discrepancy in their first-order statistics.

2nd OS On the other hand, second-order statistics are evaluated by determining the probability of pins (or dipoles) of various lengths and orientations landing on the texture with both ends positioned on black dots.

3rd OS For the third-order statistics, instead of dipoles, we study the probability that triangles with fixed size, shape, and orientation have all three vertices coinciding with black dots. Two textures that share identical second-order statistics may differ in their third-order statistics.

The statement by Julesz suggests that textures differing in either their first-order or second-order statistics should be easily distinguishable. However, even if differing in third-order statistics, textures sharing these statistics should not be perceptually distinguishable. While this principle has been disproven [Martin and Pomerantz 1978], it served as a starting point for subsequent research. Subsequent works have focused on defining various statistics to recognize and reproduce textures effectively.

The Simoncelli Approach: Steerable Pyramids & Random Fields

Eero P. Simoncelli and Javier Portilla [Portilla and Simoncelli 2000] proposed a texture synthesis method that extends the Julesz conjecture, aiming to capture the statistical properties of visual texture consistent with human perception. The foundation involves defining a *texture* as a real two-dimensional homogeneous random field $X(n, m)$ on a finite lattice $(n, m) \in L \subset \mathbb{Z}^2$. The approach relies on an extended Julesz hypothesis, asserting perceptual equivalence between two random fields if their expectations over a set of constraint functions $\{\varphi_k(X), k = 1, \dots, N_c\}$ are equal:

$$\mathbb{E}(\varphi_k(X)) = \mathbb{E}(\varphi_k(Y)), \forall k \Rightarrow$$

$$\Rightarrow \text{samples of } X \text{ and } Y \text{ are perceptually equivalent.}$$

Here, $\mathbb{E}(\cdot)$ denotes the expected value over the relevant random field. The set of functions $\{\varphi_k(X), k = 1, \dots, N_c\}$ is referred to as the constraint functions. To conduct their analysis, Simoncelli and Freeman decomposed images onto a linear basis, emphasizing distinct patterns at different scales and orientations.

For this decomposition, they chose a "steerable pyramid" [Simoncelli and Freeman 1995], a multi-scale and multi-orientation image representation. This pyramid is created by applying a set of filters to an image, capturing features at different scales and orientations. Indeed, they can be steered or rotated to align with various orientations, providing flexibility for synthesizing textures at different angles. The choice of the steerable pyramid was motivated by its favorable reconstruction properties, explicitly being a tight frame, along with its translation-invariance and rotation-invariance characteristics.

*École Normale Supérieure Paris-Saclay, Scuola Normale Superiore
E-mail: aspagnol@ens-paris-saclay.fr, alessio.spagnoletti@sns.it

Texture Synthesis Using Convolutional Neural Networks (CNNs)

In recent years, deep learning, specifically Convolutional Neural Networks (CNNs), has transformed texture synthesis. The approach by Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge [Gatys et al. 2015] leverages the capabilities of CNNs to generate textures with improved realism thanks to the exploitation of the *deep features* recognized by the VGG-19 network (Visual Geometry Group, University of Oxford) [Simonyan and Zisserman 2014].

The main idea is to initially generate a white noise image, from which they extract the *deep features* applying to it the VGG-19 network and considering the output of different layers. They compute the Gram matrices (correlation matrices between the responses of different features) for each layer. Since the same procedure is also carried out on the target image, they consider the squared difference of such matrices as a loss function.

Mathematically speaking, we have: for each layer l, there are N_l distinct filters generating N_l feature maps, each of size M_l when vectorized. These feature maps can be organized into a matrix $F^l \in \mathbb{R}^{N_l \times M_l}$, where F^l_{jk} represents the activation of the j-th filter at position k in layer l. The Gram matrices, denoted as $G_l \in \mathbb{R}^{N_l \times N_l}$, quantify the inner product between feature map i and j in layer l:

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk} \tag{1}$$

A collection of Gram matrices $G_1, G_2, \ldots, G_L$, derived from layers $1, \ldots, L$ in response to a given texture, furnishes a stationary representation, fully characterizing the texture within their model.

Let $\mathbf{x}$ and $\hat{\mathbf{x}}$ be the original image and the generated image, and G^l and $\hat{G^l}$ their respective Gram-matrix representations in layer l. The contribution of layer l to the total loss is then:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G^l_{ij} - \hat{G^l_{ij}} \right)^2$$

and the total loss is defined as:

$$L(\mathbf{x}, \hat{\mathbf{x}}) = \sum_{l=0}^{L} w_l E_l \tag{2}$$

Here, w_l represents the weighting factors for the contribution of each layer to the total loss.

This approach is the starting point for the work we will analyze since we consider a similar approach that exploits Optimal Transport (OT) instead of Gram Matrices.

2 GOTEX framework

As we have seen up to now, there are two different ways to extract information from an image:

- **patches** extracted through manipulations made on the image (such as the *steering pyramid*) that should aim to highlight some statistics typical of that texture.
- **deep features** extracted from some layers of complex models trained for image recognition, such as VGG-19 or InceptionV3 [Szegedy et al. 2014].

The idea exploited in [Houdard et al. 2021] is based on an Optimal Transport cost used to compare the distributions of some features like the two above derived by the target image and the synthesized one. It is worth stressing the fact that these features have the role of the OS of Julesz or the constraint functions of Simoncelli. Thus, we should show how to use the OT theory to develop an algorithm that can reduce such OT cost at each iteration. From an intuitive point of view, this means that we are making the two distributions more similar, and, as seen already in Julesz's work, the two textures should be almost indistinguishable.

Furthermore, the authors decided to try such a framework also with a deep generative feed-forward neural network since all the proofs they derived work with more general synthesizer functions g_θ (we will see the details in the next section) and not only the identity case, that is when we try to optimize pixel-wise the synthetic image. The optimization is realized on the parameters of a feed-forward network rather than on the image pixels. Generative networks have various applications, and the idea to try them for texture synthesis comes from the necessity of developing a model that does not require training for each usage so that we can realize new syntheses on the fly. Ulyanov et al. in [Ulyanov et al. 2016] were the first to use them for this scope.

3 Mathematical foundations

Let us begin to analyze the structure of the different tools that we will use.

We have already seen how Gatys et al. formulated the problem of defining a distance between the target texture and the synthesized one (1); in a more general scheme, we could define a loss function between two textures.

3.1 Single image

We begin by addressing the generation of a single image, denoted as u, with n pixels. Each pixel i in the image is associated with a measurable function $F_i : \mathbb{R}^n \to \mathbb{R}^d$ that extracts a local feature of dimension d from the neighborhood of that pixel. For instance, $F_i(u)$ could represent a square patch of dimension $d = 3 \times s \times s$ or a set of d neural responses computed at pixel i. The image's features are then organized into a vector $F(u) = (F_i(u))_{1 \le i \le n} \in \mathbb{R}^{dn}$.

A generic loss function can be defined as:

$$\mathcal{L}(u, u_0) = \Lambda(F(u), F(u_0)).$$

where the cost function $\Lambda : \mathbb{R}^{dn} \times \mathbb{R}^{dn} \to \mathbb{R}_+$ is chosen to measure the similarity between feature maps.

To create a new sample u resembling a given example texture u_0, we minimize this loss with respect to u. Obviously, we should be concerned about the differentiability of Λ and all the F_i; in such cases, we can apply gradient descent to the image's pixels.

3.2 Generative case

To include the generative setting, we can assume that different samples of a given texture are actually samples of a probability distribution. This distribution is characterized as the transformation of a given random ζ defined on $\mathcal{Z}$ (for example, a uniform distribution ζ in a latent space $\mathcal{Z} = [0,1]^M$ of dimension M), using a generator g. Let us consider a measurable generative function $g : \Theta \times \mathcal{Z} \to \mathbb{R}^n$, where Θ represents a set of parameters. For a specific parameter $\theta \in \Theta$, we denote $g_\theta = g(\theta, \cdot)$, and we treat the resulting texture distribution as the transformed $\sharp$ distribution $g_\theta \sharp \zeta$. This is computed as $g_\theta \sharp \zeta(B) = \zeta\left(g_\theta^{-1}(B)\right)$ for any Borel set B. In crafting a meaningful generative model, our goal is to minimize the following objective function with respect to the parameters θ:

$$\mathcal{L}_{gen}(\theta, u_0) = \mathbb{E}_{Z \sim \zeta}\left[\Lambda\left(F\left(g_\theta(Z)\right), F\left(u_0\right)\right)\right] \tag{3}$$

In the context of these generative models, we encounter a semi-discrete problem because the available data u_0 is discrete, while the

generated distribution $g_\theta \sharp \zeta$ is expected to be continuous. This semi-discreteness will help us in the formulation of an OT loss function whose gradient can be easily computed (an important aspect when considering gradient descent algorithms).

Notice that this formulation encompasses the scenario of single image synthesis, where we minimize with respect to image pixels. If θ is the image to optimize, setting $\zeta = \delta_0$ and $g(\theta, z) = \theta - z$ yields $g_\theta \sharp \zeta = \delta_\theta$.

A Unified Model for Texture Generation

As anticipated before, the authors in [Houdard et al. 2021] introduce a versatile texture formation model that accommodates both image pixel optimization and weight optimization of a generative model. To achieve this, we have to define the probability distribution of features, denoted as F_i, sampled from the texture distribution $g_\theta \sharp \zeta$, where g_θ can be both a Deep NN or the function seen above. Assuming that all F_i are measurable, each i corresponds to a local feature distribution defined as

$$\mu_\theta^i = F_i \sharp \left(g_\theta \sharp \zeta\right) = \left(F_i \circ g_\theta\right) \sharp \zeta.$$

Consequently, the collective feature distribution of the generative model can be expressed as

$$\mu_\theta = \frac{1}{n} \sum_{i=1}^{n} \mu_\theta^i = \frac{1}{n} \sum_{i=1}^{n} \left(F_i \circ g_\theta\right) \sharp \zeta, \tag{4}$$

that is the average among all the pixels (remember that we have a feature F_i for each pixel).

For texture synthesis, our objective is to minimize a distance function between μ_θ and the feature distribution ν extracted from a target image u_0. In this context, optimal transport emerges as a natural tool for comparing probability distributions. We proceed to outline our formulation based on an optimal transport distance.

3.3 OT Cost for Feature Distribution Comparison

Given an example texture u_0, following the previous section, we denote its feature distribution as ν:

$$\nu = \frac{1}{m} \sum_{j=1}^{m} \delta_{F_j(u_0)}.$$

In the GOTEX framework, the objective is to constrain the feature distribution μ_θ of the synthesized textures to match the target distribution ν. To achieve this, we employ the well-known optimal transport cost, denoted as:

$$\mathcal{L}_{\mathrm{GOTEX}}\left(\theta, u_0\right) = \mathrm{OT}_c\left(\mu_\theta, \nu\right) = \inf_{\pi \in \Pi(\mu_\theta, \nu)} \int c(x,y) d\pi(x,y). \tag{5}$$

Here, $c : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a Lipschitz cost function (between features), and $\Pi\left(\mu_\theta, \nu\right)$ represents the set of probability distributions on $\mathbb{R}^d \times \mathbb{R}^d$ with marginals μ_θ and ν (also called the set of possible couplings). When using $c(x,y) = \|x - y\|^2$, as adopted for our experiments, OT_c corresponds to the square of the Wasserstein-2 distance. The fact that $\mathcal{L}_{\mathrm{GOTEX}}$ is indeed a distance is a point to prefer it with respect to the Gram loss of [Gatys et al. 2015] (from a geometric point of view).

As said in the last paragraph, it is relevant to consider whether the loss is differentiable, in this case with respect to θ, since we want to minimize it. For our setting, we will now see how the dual formulation of OT helps us. In fact, we can separate the problems of approximating the OT distance and minimizing with respect to θ; furthermore, another simplification is that our target distribution ν is discrete, allowing us to employ flexible algorithms for semi-discrete optimal transport.

To recap everything, texture synthesis is achieved by minimizing the optimal transport cost (5) with respect to μ_θ, its first argument. Therefore, we consider the semi-dual formulation of the optimal transport cost.

Semi-dual formulation: If $\mathcal{X}$ and $\mathcal{Y}$ are compact, and the cost c is continuous, then [Santambrogio 2015]:

$$\mathrm{OT}_c(\mu, \nu) = \sup_{\substack{\phi \in \mathrm{C}(\mathcal{X}), \psi \in \mathrm{C}(\mathcal{Y}), \\ \phi(x) + \psi(y) \leq c(x,y)}} \int \phi(x) d\mu(x) + \int \psi(y) d\nu(y) \tag{6}$$

As an important point, we stress that in [Houdard et al. 2021], they use a different formulation which involves the max among all continuous functions $\mathcal{C}(\mathcal{Y})$.

$$\mathrm{OT}_c(\mu, \nu) = \max_{\psi \in \mathcal{C}(\mathcal{Y})} \int \psi^c(x) d\mu(x) + \int \psi(y) d\nu(y). \tag{7}$$

where the c-transform is $\psi^c(x) = \min_{y \in \mathcal{Y}} [c(x,y) - \psi(y)]$.

This formulation, as can be seen in the reference [Santambrogio 2015], is due to the fact that the c-transforms of ψ is, by definition, the best function to satisfy the condition (the one that maximizes the integral with respect to μ). There are, however, two points to highlight that are not treated in [Houdard et al. 2021]:

- Are we sure that our spaces $\mathcal{X}$ and $\mathcal{Y}$ are compact? Since we are considering them as the output spaces of our feature maps F_i, we can prove compactness once we have defined these maps, but this seems quite restrictive. The compactness is fundamental since, as can be seen in [Santambrogio 2015], the cost c must be uniformly continuous and bounded on $(\mathcal{X}, \mathcal{Y})$ and we have to define the min for ψ^c. Can we hope for a less strict hypothesis?
- Can we consider the maximum? Does it exist? Or is it just a sup?

We propose an answer to these questions that can be found considering instead the dual Kantorovich formulation as showed in [Villani 2008]:

Kantorovich Duality Let (X, μ) and (Y, ν) be two Polish probability spaces, and let $c : X \times Y \to \mathbb{R} \cup +\infty$ be a lower semicontinuous cost function such that, for all $(x,y) \in X \times Y$, $c(x,y) \geq a(x) + b(y)$ for some real-valued, upper semicontinuous functions $a \in L^1(\mu)$ and $b \in L^1(\nu)$. Then,

$$\begin{aligned}
&\min_{\pi \in \Pi(\mu,\nu)} \iint_{X \times Y} c(x,y)\, d\pi(x,y) \\
&= \sup_{(\psi,\phi) \in C_b(X) \times C_b(Y); \phi + \psi \leq c} \left(\int_Y \psi(y)\, d\nu(y) + \int_X \phi(x)\, d\mu(x) \right) \\
&= \sup_{\psi \in L^1(\nu)} \left(\int_X \psi^c\, d\mu(x) + \int_Y \psi(y)\, d\nu(y) \right).
\end{aligned} \tag{8}$$

So we have answered our first question since $\mathcal{X} = \mathcal{Y} = \mathbb{R}^d$ are Polish spaces, the cost function that we will use in our experiments satisfies the conditions $c(x,y) = ||x-y||^2 \geq 0 + 0 = a(x) + b(y)$ and we are left with the sup over all the $\psi \in L^1(\nu)$. Regarding the second one, we will stick with the sup instead of max, but later, thanks to the fact that ν is discrete, we will be able to prove that such sup is reached.

This semi-dual approach formulates the dual problem with a single dual variable, while the other dual variable is optimized through

the c-transform. Considering then (8), we arrive at the problem of estimating the variable θ (a new texture or the parameters of a generator), which entails solving the following problem:

$$\min_{\theta} \mathrm{OT}_c\left(\mu_\theta, \nu\right) =$$

$$\min_{\theta} \sup_{\psi \in L^1(\nu)} \mathcal{J}(\theta, \psi) := \int \psi^c(x) d\mu_\theta(x) + \int \psi(y) d\nu(y). \quad (9)$$

Notice that, despite the fact that we are not considering any adversarial setting, we have obtained a min-max problem, something that is usual in the GAN framework.

As we are using a single image as a target, we get that our target measure ν assumes the form of an empirical feature distribution with a finite set of elements:

$$\nu = \frac{1}{m} \sum_{j=1}^{m} \delta_{F_j(u_0)}$$

Here, $m < +\infty$ represents the number of distinct features $y_j = F_j\left(u_0\right)$ (that is, one per pixel). Thus every $\psi \in \mathbb{R}^m$ is a function $\psi \in L^1(\nu)$. In this particular scenario, the semi-dual representation of optimal transport (9) simplifies to a more manageable form, the same obtained by the authors:

$$\mathrm{OT}_c\left(\mu_\theta, \nu\right) = \max_{\psi \in \mathbb{R}^m} \mathcal{J}(\theta, \psi) = \int \psi^c(x) d\mu_\theta(x) + \frac{1}{m} \sum_{j=1}^{m} \psi_j \quad (10)$$

Here, $\psi_j = \psi\left(y_j\right)$, and the c-transform of ψ can be expressed as $\psi^c(x) = \min_j \left[c\left(x, F_j\left(u_0\right)\right) - \psi_j\right]$. Notice that we passed to a max formulation thanks to the fact that we can easily restrict to a compact subset of $\mathbb{R}^m$ where the sup is reached since outside the sum term will be too big. What makes this formulation appealing is that it involves a finite-dimensional vector $\psi \in \mathbb{R}^m$, which can be optimized computationally. It is worth noting that the computation of the c-transform $\psi^c(x)$ simplifies to a weighted nearest-neighbor assignment in the feature space with a distance defined through the cost.

By combining the definition of μ_θ given in (4) and the functional $\mathcal{J}$ defined in (13), the ultimate problem to address is:

$$\min_{\theta} \mathrm{OT}_c\left(\mu_\theta, \nu\right) = \min_{\theta} \max_{\psi \in \mathbb{R}^m} \mathcal{J}(\theta, \psi)$$

$$= \min_{\theta} \max_{\psi \in \mathbb{R}^m} \mathbb{E}_{Z \sim \zeta} \left[\frac{1}{n} \sum_{i=1}^{n} \psi^c\left(F_i \circ g_\theta(Z)\right) + \frac{1}{m} \sum_{j=1}^{m} \psi_j \right] \quad (11)$$

In this equation, θ can either represent a single image or the parameters of a generative network with input noise Z.

We will now see how to compute the gradients concerning this quantity with respect to θ and ψ. In order to compute the gradient with respect to θ for the gradient descent algorithm we will now show that the gradient of the optimal transport $\nabla_\theta \mathrm{OT}_c\left(\mu_\theta, \nu\right)$ coincides with the gradient $\nabla_\theta \mathcal{J}\left(\theta, \psi^*\right)$ of the function $\mathcal{J}$ at an optimal value ψ^*. This is clearly important for our simulations since we can fix ψ^* (we will see how to compute it) and deal with only one variable without the inner maximization problem.

3.4 Gradients for the Semi-Dual OT cost

This paragraph discusses the computation of the gradient with respect to the parameter θ of the optimal transport cost $\mathrm{OT}_c\left(\mu_\theta, \nu\right)$. We will see, following the proofs made in [Houdard et al. 2021], a sufficient condition (related to the regularity of the generator g_θ and to the feature distribution μ_θ) that ensures the existence of $\nabla_\theta \mathcal{J}(\theta, \psi)$. Besides, under the same condition, we show that $\nabla_\theta \mathrm{OT}_c\left(\mu_\theta, \nu\right) = \nabla_\theta \mathcal{J}\left(\theta, \psi^*\right)$ with $\psi^* \in \arg\max_\psi \mathcal{J}(\theta, \psi)$ as soon as both terms are well defined.

As seen in equation (11), when computing the gradient of $\mathcal{J}$ with respect to θ, it is evident that the differentiation is required only for $\psi^c\left(F_i \circ g_\theta(Z)\right)$. While dealing with the c-transform might initially appear challenging, we will introduce a tool to assess the regularity of ψ^c. To do this, we introduce the concept of open Laguerre cells:

$$L_i(\psi) = \left\{x \,|\, \forall k \neq i, c\left(x, F_i\left(u_0\right)\right) - \psi_i < c\left(x, F_k\left(u_0\right)\right) - \psi_k\right\}.$$

A crucial observation immediately follows from the definition of the c-transform: within the Laguerre cell $L_i(\psi)$, the c-transform is expressed as:

$$\forall x \in L_i(\psi), \quad \psi^c(x) = c\left(x, F_i\left(u_0\right)\right) - \psi_j.$$

Consequently, ψ^c inherits the regularity of c within the Laguerre cells. Therefore, when c is smooth, issues related to differentiability may only arise at the boundaries of the Laguerre cells. We will now show the hypothesis that ensures the differentiability of $\mathcal{J}$ w.r.t θ.

Hypothesis 1 g satisfies Hypothesis 1 at (θ, ψ) if $\zeta\left(\left(F_i \circ g_\theta\right)^{-1}\left\{\cup_j L_j(\psi)\right\}\right) = 1$ for any position i, that is, for a given variable θ, all the generated local features are almost surely within the Laguerre cells defined by ψ.

When $c(x, y) = \|x - y\|_p^p$ with $p > 1$ (p-Wasserstein), as in our simulations, then ψ^c is smooth on $\cup_j L_j(\psi)$. The complement of such union is contained in the union of the following sets, for $1 \leq j, k \leq m$:

$$B_{jk}(\psi) = \left\{x \mid \|x - F_j\left(u_0\right)\|_p^p - \psi_j = \| x - F_k\left(u_0\right)\|_p^p - \psi_k\right\}$$

Since the only way for which a point x is not in any of the cells is that there are at least two minima, and thus belongs to one of the B_{jk}. Notice that in [Houdard et al. 2021], they say that the two sets are equal, but in B_{jk} there are x that could be part of a L_i with $i \neq j, k$. However, the conclusion still holds since every B_{jk} has zero Lebesgue measure since each one is contained in a sub-manifold of dimension lower than p.

Therefore, Hypothesis 1 is satisfied if, for any i, $\left(F_i \circ g_\theta\right) \sharp \zeta$ is absolutely continuous with respect to the Lebesgue measure.

Let us introduce a regularity hypothesis for the model g_θ.

Hypothesis 2 There exists $K : \Theta \times \mathcal{Z} \to \mathbb{R}_+$ such that for all θ, there exists a neighborhood V of θ such that $\forall\ \theta' \in V$ and for ζ-almost every z,

$$\left\|g(\theta, z) - g\left(\theta', z\right)\right\| \leq K(\theta, z) \left\|\theta - \theta'\right\|$$

with K verifying for all $\theta, \mathbb{E}_{Z \sim \zeta}[K(\theta, Z)] < \infty$.

We can now express the gradient of $\mathcal{J}$ w.r.t. the parameter θ.

Theorem 1 Assume c to be C^1 and assume that the features F_i are all differentiable and Lipschitz. Let g satisfy Hypothesis 2. Let θ_0 be a point where $\theta \to g_\theta(z)$ is differentiable $\zeta(dz)$-almost surely and let g satisfy Hypothesis 1 at (θ_0, ψ). Then $\theta \mapsto \mathcal{J}(\theta, \psi)$ is differentiable at θ_0 and

$$\nabla_\theta \mathcal{J}\left(\theta_0, \psi\right) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{Z \sim \zeta}\left[\partial_\theta\left(F_i \circ g\right)\left(\theta_0, Z\right)^T \nabla \psi^c\left(F_i \circ g\left(\theta_0, Z\right)\right)\right]$$

with $\nabla \psi^c\left(F_i \circ g\left(\theta_0, z\right)\right) = \nabla_x c\left(F_i \circ g\left(\theta_0, z\right), F_{\sigma(i)} u_0\right)$

where $\sigma(i)$ is the unique index such that $F_i \circ g(\theta_0, z) \in L_{\sigma(i)}(\psi)$ (which exists $\zeta(dz)$-almost surely as seen in Hypothesis 1).

Proof: From expression (11), we see that the proof consists in differentiating the function

$$H_i(\theta, \psi) = \mathbb{E}\left[h_i(\theta, \psi, Z)\right], \text{with} \quad h_i(\theta, \psi, Z) = \psi^c\left(F_i \circ g_\theta(Z)\right) \quad (12)$$

Thanks to Hypothesis 1 , for ζ-almost all z, there exists an index j such that $F_i \circ g_\theta(z) \in L_j(\psi)$ and thus $L_j(\psi)$ is an open neighborhood of $F_i \circ g_\theta(Z)$ where ψ^c is differentiable. Using the chain rule, we get that for ζ-almost all z, $\theta \mapsto h_i(\theta, \psi, z)$ is differentiable at θ_0 and

$$\nabla_\theta h_i(\theta, \psi, z) = \partial_\theta\left(F_i \circ g\right)(\theta_0, Z)^T \nabla\psi^c\left(F_i \circ g\left(\theta_0, Z\right)\right)$$

In order to differentiate under the expectation in (12), we have to get an integrable bound on the finite differences of h_i so that we can use the dominate convergence theorem. For that, let us denote by κ_c the Lipschitz constant of c and κ_i the Lipschitz constant of F_i. Let us recall from [Santambrogio 2015] that ψ^c is also κ_c-Lipschitz. Therefore, from Hypothesis 2, we get a neighborhood V of θ_0 such that for any $\theta \in V$ and for ζ-almost all z,

$$\begin{aligned} &\mid h_i(\theta, \psi, z)) - h_i\left(\theta_0, \psi, z\right) \mid \leq \\ &\kappa_c\kappa_i \left\|g(\theta, z) - g\left(\theta_0, z\right)\right\| \leq \kappa_c\kappa_i K(\theta, z) \left\|\theta - \theta_0\right\|, \end{aligned}$$

with $\mathbb{E}[K(\theta, Z)] < \infty$. This bound allows us to differentiate under the expectation and to get the expression

$$\nabla_\theta H_i\left(\theta_0, \psi\right) = \mathbb{E}\left[\partial_\theta\left(F_i \circ g\right)\left(\theta_0, Z\right)^T \nabla\psi^c\left(F_i \circ g(\theta, Z)\right)\right]$$

Gathering the terms for all i leads to the desired result. □

Finally, we can relate the gradient of $\mathcal{J}$ to the gradient of the OT.

Theorem 2 Let θ_0 such that $\theta \mapsto \mathrm{OT}_c\left(\mu_\theta, \nu\right)$ and $\theta \mapsto \mathcal{J}\left(\theta, \psi^*\right)$ are differentiable at θ_0 with $\psi^* \in \arg\max_\psi \mathcal{J}\left(\theta_0, \psi\right)$ then

$$\nabla_\theta \mathrm{OT}_c\left(\mu_{\theta_0}, \nu\right) = \nabla_\theta \mathcal{J}\left(\theta_0, \psi^*\right)$$

Proof: Let us fix $\psi^* \in \arg\max_\psi \mathcal{J}\left(\theta_0, \psi\right)$. The function $H(\theta) = \mathcal{J}\left(\theta, \psi^*\right) - \mathrm{OT}_c\left(\mu_\theta, \nu\right)$ is differentiable at θ_0 due to the hypothesis we chose. Furthermore, for every other θ, $\mathrm{OT}_c\left(\mu_\theta, \nu\right) = \max_\psi \mathcal{J}(\theta, \psi)$ can be obtained at a $\psi \neq \psi^*$, thus $H(\theta)$ is maximum in θ_0. Therefore we get $\nabla_\theta H\left(\theta_0\right) = 0$. □

Notice that evaluating $\nabla_\theta \mathcal{J}\left(\theta_0, \psi^*\right)$ requires the knowledge of an optimal potential $\psi^*\left(\theta_0\right) \in \arg\max_\psi \mathcal{J}\left(\theta_0, \psi\right)$.

3.4.1 Super-gradient with respect to ψ

The authors in [Houdard et al. 2021] propose a stochastic gradient ascent scheme in order to approximate ψ^*. Let us see some results about this.

Let θ be a fixed parameter; in order to improve the readability, we set $G_i = F_i \circ g_\theta$ and remove all the θ dependencies. The maximization problem we aim at solving writes

$$\begin{aligned} &\max_{\psi \in \mathbb{R}^m} \mathcal{J}(\psi) = \mathbb{E}_{Z\sim\zeta}[J(\psi, Z)] \\ &\text{where } J(\psi, z) = \frac{1}{n}\sum_{i=1}^n \psi^c\left(G_i(z)\right) + \frac{1}{m}\sum_{j=1}^m \psi_j \end{aligned}$$

We first recall the following result from the OT theory.

Theorem 3

(i) For any $z \in \mathcal{Z}$, the function $\psi \to J(\psi, z)$ is concave on $\mathbb{R}^m$.

(ii) The function $\psi \to \mathcal{J}(\psi)$ is concave on $\mathbb{R}^m$.

Proof: (i) Let $\psi^1 \in \mathbb{R}^m, \psi^2 \in \mathbb{R}^m$ and $t \in [0, 1]$ and fix $z \in \mathcal{Z}$. Recalling the c-transform definition $\psi^c(x) = \min_j\left[c\left(x, F_j\left(u_0\right)\right) - \psi_j\right]$ (weighted nearest neighbor), splitting $c(\cdot,\cdot) = tc(\cdot,\cdot) + (1-t)c(\cdot,\cdot)$ and using the property of the minimum function, we get that

$$\begin{aligned} J\left(t\psi^1 + (1-t)\psi^2, z\right) \geq\ & \frac{1}{n}\sum_{i=1}^n t \min_j\left[c\left(G_i(z), F_j\left(u_0\right)\right) - \psi_j^1\right] \\ &+ \frac{1}{n}\sum_{i=1}^n (1-t)\min_j\left[c\left(G_i(z), F_j\left(u_0\right)\right) - \psi_j^2\right] \\ &+ \frac{1}{m}\sum_{j=1}^m t\psi_j^1 + (1-t)\psi_j^2 \\ =\ & tJ\left(\psi^1, z\right) + (1-t)J\left(\psi^2, z\right), \end{aligned}$$

which proves the first point.

(ii) The second point follows by taking the expectation of both sides. □

Now, we will use the theory of super-gradients. In practice, it is not easy to compute the exact gradient of $\mathcal{J}$, but for our aim, it is enough to approximate it using its super-gradient $D(\psi, z)$ instead, that is s.t.

$$\forall \psi' \in \mathbb{R}^m, \quad J\left(\psi', z\right) \leq J(\psi, z) + \left\langle D(\psi, z), \psi' - \psi\right\rangle \quad (13)$$

We can now state the following result that gives a super-gradient for $\mathcal{J}$.

Theorem 4 Let us denote by $(e_j)_{1\leq j\leq m}$ the canonical basis of $\mathbb{R}^m$. Let $z \in \mathcal{Z}$, and $\psi \in \mathbb{R}^m$. Then a super-gradient of $J(\cdot, z)$ at point ψ is given by

$$D(\psi, z) = \frac{1}{m}\mathbf{1}_m - \frac{1}{n}\sum_{i=1}^n \mathbf{e}_{j^*\left(G_i(z), \psi\right)}$$

where

$$j^*\left(G_i(z), \psi\right) \in \arg\min_j\left(c\left(G_i(z), F_j(u_0)\right) - \psi_j\right).$$

It follows that

$$\mathcal{D}(\psi) = \mathbb{E}_{Z\sim\zeta}[D(\psi, Z)]$$

is a super-gradient of $\mathcal{J}$ at point ψ.

Proof: Take $z \in \mathcal{Z}$ and $\psi \in \mathbb{R}^m$. In order to demonstrate that $D(\psi, z)$ is a super-gradient, we need to show (13). We have

$$J(\psi, z) - J\left(\psi', z\right) + \left\langle D(\psi, z), \psi' - \psi\right\rangle =$$

$$\frac{1}{n}\sum_{i=1}^n \min_j\left[c\left(G_i(z), F_j\left(u_0\right)\right) - \psi_j\right] + \cancel{\frac{1}{m}\sum_{j=1}^m \psi_j} \quad (14)$$

$$- \frac{1}{n}\sum_{i=1}^n \min_j\left[c\left(G_i(z), F_j\left(u_0\right)\right) - \psi_j'\right] - \cancel{\frac{1}{m}\sum_{j=1}^m \psi_j'} \quad (15)$$

$$- \frac{1}{n}\sum_{i=1}^n \left(\psi'_{j^*\left(G_i(z), \psi\right)} - \psi_{j^*\left(G_i(z), \psi\right)}\right) + \cancel{\frac{1}{m}\sum_{j=1}^m \psi_j'} - \cancel{\frac{1}{m}\sum_{j=1}^m \psi_j}$$

Then if we consider $j^*\left(G_i(z), \psi\right)$, it realizes the min in (14), while in (15) by definition the min is smaller than the value obtained at j^*. So we have that everything is canceled, and we get

$$J(\psi, z) - J\left(\psi', z\right) + \left\langle D(\psi, z), \psi' - \psi\right\rangle \geq 0$$

□

To approximate an optimal potential, the authors use an averaged stochastic super-gradient ascent, as proposed in [Genevay et al. 2016]. Specifically, at each step k, we have to sample $z \sim \zeta$, and update ψ^k according to the following rule:

$$\psi^k = \psi^{k-1} + \frac{1}{\sqrt{k}} D\left(\psi^{k-1}, z\right)$$

The final estimate for ψ^* is obtained by averaging the estimates from a given starting point k_0 (in experiments, k_0 is set to 1):

$$\hat{\psi}^k = \frac{1}{k - k_0 + 1} \sum_{\ell=k_0}^{k} \psi^\ell$$

Ultimately, the combination of Theorem 2 and Theorem 4 provides a means to approximate the gradient of the optimal transport cost $\mathrm{OT}_c\left(\mu_\theta, \nu\right)$ with respect to θ. This was our objective, and consequently, we will see that the minimization of this optimal transport cost will be done through a stochastic gradient descent algorithm (we will use both Adam [Kingma and Ba 2014] and L-BFGS [Liu and Nocedal 1989]).

4 The algorithms

We will now show the two main algorithms, which are the one for the single image generation and the one for the generative network. First, let us consider how to add more features to our model. Up to now, we considered a single feature map F that was applied to every pixel obtaining the various F_i. What is usually done is to consider a set of N_F different features given by $F^l = \left(F_i^l\right)_{1 \le i \le n_l}, l = 1, \ldots, N_F$, each one of size n_l, and then to minimize the quantity

$$\mathcal{L}_{\mathrm{GOTEX}}(\theta) = \sum_{l=1}^{N_F} \mathrm{OT}_c\left(\mu_\theta^l, \nu^l\right) = \sum_{l=1}^{N_F} \max_{\psi^l \in \mathbb{R}^{m_l}} \mathcal{J}^l\left(\theta, \psi^l\right) \tag{16}$$

where μ_θ^l (resp. ν^l) is the distribution relative to the feature l of the synthesis (resp. of the example u_0 that contains m_l features), i.e. $\mu_\theta^l = \frac{1}{n_l} \sum_{i=1}^{n_l} \delta_{F_i^l(\theta)}$, and where

$$\mathcal{J}^l\left(\theta, \psi^l\right) = \mathbb{E}_{Z \sim \zeta}\left[\frac{1}{n_l} \sum_{i=1}^{n_l} \psi^{l,c}\left(F_i^l \circ g_\theta(Z)\right) + \frac{1}{m_l} \sum_{j=1}^{m_l} \psi_j^l\right]$$

with $\psi^{l,c}(x) = \min_{1 \le j \le m_l}\left[c\left(x, F_j^l\left(u_0\right)\right) - \psi_j^l\right]$.

Upon the existence of all terms, the gradient of the quantity (16) then reads

$$\nabla_\theta \mathcal{L}(\theta) = \sum_{l=1}^{N_F} \nabla_\theta \mathcal{J}^l\left(\theta, \psi^{l,*}\right)$$

where $\psi^{l,*}$ is an optimal Kantorovich potential for $\mathrm{OT}_c\left(\mu_\theta^l, \nu^l\right)$. The potential $\psi^{l,*}$ can be approximated with a super-gradient ascent (using the super-gradient of $\mathcal{J}^l$ obtained in Theorem 4 as seen before) and then run the optimization step w.r.t. θ considering the total loss function.

The multi-feature process is summarized in Algorithm 1. Here $D(\theta, \psi, z)$ stands for $D(\psi, z)$ since we dropped before the θ dependencies, but now we reintroduce them in the notation.

Algorithm 1 Texture Synthesis with Prescription of Several Feature Distributions

Input: target image u_0, initial parameter θ_0, learning rates η_θ and η_ψ, number of iterations N_u and N_ψ, number of features N_F, optimizer *Optim* (s.t. Adam or L-BFGS)
Output: learned parameter θ^*
$\theta \leftarrow \theta_0$ and $\tilde{\psi}^{l,0} = 0$ for $l = 1 \ldots N_F$
for $k = 1$ **to** N_θ **do**
  for $l = 1$ **to** N_F **do**
    $\tilde{\psi}^{l,0} \leftarrow \psi^{l,k-1}$
    for $\ell = 1$ **to** N_ψ **do**
      Draw a sample $z \sim \zeta$.
      $\tilde{\psi}^{l,\ell} = \tilde{\psi}^{l,\ell-1} + \eta_\psi D(\theta^{k-1}, \tilde{\psi}^{l,\ell-1}, z)$
      ▷ Gradient ascent on $\psi^{l,k}$ (Theorem 4)
  $\psi^{l,k} \leftarrow \tilde{\psi}^{l,N_\psi}$
  $\theta^k \leftarrow Optim(\sum_{l=1}^{N_F} \nabla_\theta \mathcal{J}^l(\theta^{k-1}, \psi^{l,k}), \eta_\theta)$ ▷ Gradient-based update of θ (Theorem 2)

As can be easily deduced, the algorithm for the single image case is the same, without the sampling from ζ and with θ that represents the whole image.

4.1 The Feature maps choice

As we mentioned in the introduction, the choice of which statistical features we want to use is fundamental. The authors in [Houdard et al. 2021] decided to apply two different sets of F^l:

1. The Gaussian pyramid of patches
2. The Deep VGG features

4.1.1 Gaussian pyramid of patches

The first set of features is inspired by Simoncelli's work [Portilla and Simoncelli 2000] in which, as we have already seen, the idea of using linear transformations that could project the texture over a basis was used. Such a basis should separate the different characteristics of the texture captured at different levels (we have rotations and scaling). What we will do is use a pyramid of down-sampled and blurred images from which we will extract a $s \times s$ patch around each pixel.

For each scale $l = 1, \ldots, N_F$ (we will use $N_F = 4$), we apply a linear blurring and down-sampling operator G_s to transform an image u with $n = n_W \times n_H$ pixels into a reduced version $u_l = G_l u$ with $n_l = \frac{n_H}{2^{l-1}} \times \frac{n_W}{2^{l-1}}$ pixels. Subsequently, we define the operator P_i^l that extracts the ith patch from the blurred and down-sampled image u_l, represented as $P_i^l = P_i \circ G_l$. So we have the collection of features $\left\{P_i^l\right\}_{i,l}$.

So now the loss we aim to minimize is expressed as follows:

$$\mathcal{L}_{\text{patch}}(\theta) = \sum_{l=1}^{N_F} \mathrm{OT}_c\left(\mu_{\text{pat}}^l(\theta), \nu_{\text{pat}}^l\right),$$
$$\text{where } \mu_{\text{pat}}^l(\theta) = \frac{1}{n_l} \sum_{i=1}^{n_l} \left(P_i^l \circ G_l \circ g_\theta\right) \sharp \zeta$$

And where similarly ν_{pat}^l is the blurred and down-sampled patch distribution of the example image example u_0. This loss is referred to as the **GOTEX-patch** loss.

4.1.2 VGG features

As discussed before, the use of deep features introduces a distinctive perspective on pattern extrapolation, capitalizing on the capabilities of non-linear and complex feature extractors such as the pretrained (on the ImageNet dataset) VGG-19 network. The approach involves extracting outputs from five layers $N_F = 5$, starting from the initial convolutional layer and then each layer after each max-pooling operation (pool1 to pool4). The objective is to project textures into a space that accentuates various statistical properties. To address checkerboard artifacts, the conventional max-pooling layers are substituted with average-pooling layers, aligning with the methodology proposed in [Gatys et al. 2015]. The related loss to be

optimized reads

$$\mathcal{L}_{\text{VGG}}(\theta) = \sum_{l=1}^{N_F} \text{OT}_c \left(\mu^l_{\text{VGG}}(\theta), \nu^l_{\text{VGG}} \right),$$

$$\text{with} \quad \mu^l_{\text{VGG}}(\theta) = \frac{1}{n_l} \sum_{i=1}^{n_l} \left(F^l_i \circ g_\theta \right) \sharp \zeta$$

where F^l_i corresponds to the normalized i-th output from the layer l of the VGG-19 network and n_l the spatial size of this layer, and where similarly ν^l_{VGG} is the feature distribution of the example at layer l. The loss will be referred to as the **GOTEX-VGG** loss.

4.1.3 Mixing features

The two approaches seen above perform very well on some example textures, but, as we can see from the experiments we will do, there are some side effects for both of them. In particular:

1. For the Gaussian pyramid, the problem is that the blurring used tends to lose some details, resulting in textures with less contrast.
2. The VGG feature extractor instead suffers from checkerboard artifacts (despite the use of average-pool layers) and general noise

It is natural to ask ourselves if a combination of the two could improve the result. So what we will try is to consider a **GOTEX-all** loss function:

$$\mathcal{L}_{\text{all}}(\theta) = \lambda \sum_{l=1}^{N_F} \text{OT}_c \left(\mu^l_{\text{pat}}(\theta), \nu^l_{\text{pat}} \right) + \sum_{l=1}^{N_F} \text{OT}_c \left(\mu^l_{\text{VGG}}(\theta), \nu^l_{\text{VGG}} \right)$$

In which fine-tuning the parameter λ will give better results. As we will see in the experiments, this is indeed the case.

Notice that in [Houdard et al. 2021] is also defined the **Gotex-mix** loss where we only take the first level $l = 1$ of the Gaussian pyramid and then the $2, \dots, N_F$ layers of the VGG network. The idea is that the VGG features from deep layers represent large structures and global geometry from the target texture, while patch features from the first scales of the image represent local details and color distributions. We will not use this model, but in the experiments, we will compare our results with those obtained by the authors using this method.

4.2 Our main contribution

Following the ideas seen above, we have decided to try a similar approach that exploits both the classical patches extracted through the Gaussian pyramid and the deep features obtained from some layers of a deep CNN. In the past years, many models have been studied and trained on the ImageNet dataset, with many of them achieving top performances with error rates lower than 10%; among them, we can find the VGG-19 and the InceptionV3 (GoogLeNet).

Our intuition of trying the InceptionV3 network came from the fact that it is common to use the Fréchet Inception Distance [Heusel et al. 2017] as a metric for telling how much a set of generated images is similar to the original ones. This metric is defined as the 2-Wasserstein distance computed w.r.t. two Gaussian distributions fitted on the output of the last non-fully connected layer of the IV3 network. The first distribution comes from a set of authentic images fed into the network, and the second comes from a generated set. Furthermore, the structure of the IV3 model consists of a series of CNN followed by many *Inception modules* stacked and piped to form the whole net. The goal of an inception module is to act as a "multi-level feature extractor" by computing 1×1, 3×3, and 5×5 convolutions within the same module of the network. Then, the outputs of these filters are stacked along the channel dimension and fed into the next layer in the network. This structure makes the IV3 lighter and faster, allowing the recognition of features at different scales thanks to the different sizes of the filters.

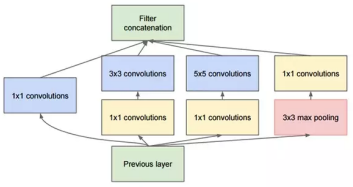
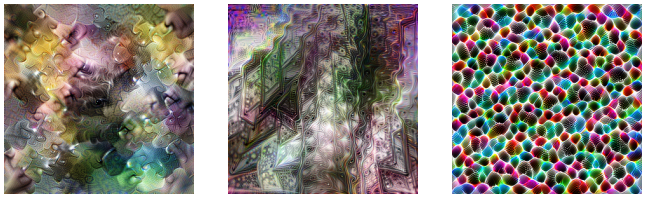


Figure 2: *Inception modules structure*

The implementation we tried used as output the *Mixed7c*, *Mixed6e*, and *Mixed5d* layers. We thought about them after looking at the optimized images that maximize the activation of a given unit (a specific filter of a convolutional layer), thanks to OpenAI microscope [1]. We found that the number of units of the three layers we chose was higher than the ones of the layers used from VGG (2048 vs 512), and this means potentially more patterns recognized. Furthermore, the images provided more complex patterns than the VGG ones; for example, the first image (from the Mixed7c layer) in Figure 3 shows a pattern that matches mostly with puzzle pieces (macro structure), while the other two figures show more abstract patterns since they come from the internal layers (representing a meso and micro structure). Looking at the VGG, we find less interesting macro patterns.



Mixed7c - unit 73 *Mixed6e - unit 5* *Mixed5d - unit 48*

Figure 3: *Feature visualization from OpenAI microscope*

So, in the end, we have the **Gotex-IV3** loss, which has the same form as the VGG one and can (we will do this in the next section) be mixed with the Gaussian one.

5 Experiments

We will now show the results obtained on some of the textures tried in [Houdard et al. 2021] with various combinations of the proposed models together with our **Gotex-mix-IV3** model. It consists of a mixture of the Gaussian pyramid of patches with $N_F = 4$ and the three layers of the IV3, giving birth to the following loss function:

$$\mathcal{L}_{\text{IV3}}(\theta) = \sum_{l=1}^{N_F} \text{OT}_c \left(\mu^l_{\text{pat}}(\theta), \nu^l_{\text{pat}} \right) + \lambda \sum_{l=1}^{3} \text{OT}_c \left(\mu^l_{\text{IV3}}(\theta), \nu^l_{\text{IV3}} \right)$$

where we set $\lambda = 0.1$. Then, we will try these models on different textures to better highlight some pros and cons that are not highlighted by the analysis made by the authors. We will show that our model performs better than all the others on the images they provided, always keeping good behavior even on textures where other methods perform better.

Finally, we will try all these models with the generative framework where we use as g_θ the *TexNet* architecture described in [Ulyanov et al. 2016].

[1] `https://microscope.openai.com/`

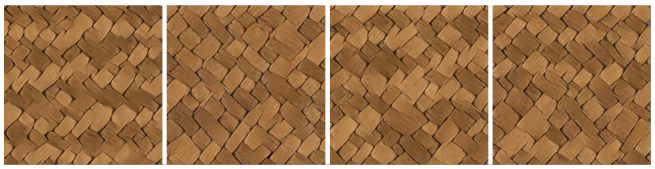
For completeness, we provide the hyper-parameters used in the experiments. The common ones were the number of iterations of the averaged stochastic gradient ascent for ψ, set to 20, and the learning rate set to 1 as done in [Houdard et al. 2021]. The Gaussian patches sizes are 4×4, $s = 4$, and the processor was a 13th Gen Intel Core i9-13980HX CPU with Nvidia GeForce RTX 4070 (Mobile) 8GB. All the samples were of size 256×256.

Single image: The optimizers and learning rate for θ were: $\text{LBFGS}_{lr} = 1$ for the VGG and the VGG+Gaussian, $\text{Adam}_{lr} = 0.1$ for the IV3+Gaussian. The images seen are the 1000th iteration of the whole algorithm. The time needed was about 24 minutes for each simulation.

Generative: The optimizers and learning rates for θ were: $\text{Adam}_{lr} = 0.1$ for all the models. The images seen are generated by the generator obtained after 5000 iterations of the whole algorithm. The time needed was about 2 hours for each simulation.

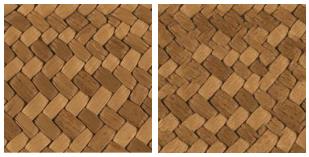
The following are some remarkable tests from which we derived some conclusions. Other generated textures can be found in Fig 1.

5.1 Wicker test



Original tests: Gotex-patch, Gotex-VGG, Gotex-mix, Gotex-all

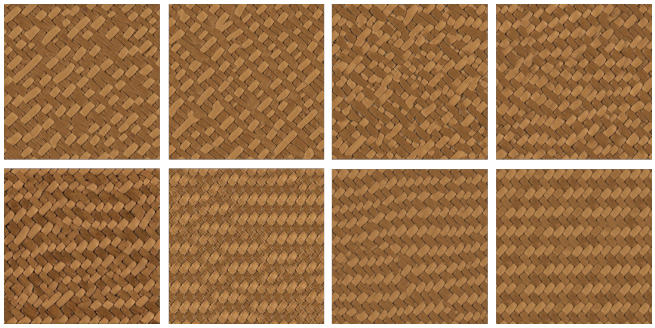
The wicker texture was deeply studied in [Houdard et al. 2021], as we can see in the figures. We aim to show how our **Gotex-mix-IV3** model performs better than all the others both in the single image case and the generative one. We can see how it is the only one, together with the **Gotex-patch**, that understands the color distribution related to the alternating pattern made of darker and lighter rectangles. Furthermore, it manages to create cleaner boundaries than the patch version, comparable to the **Gotex-VGG** one.



Real one *Gauss+IV3*

What is actually even better is the generative version. We provide pictures generated by various models tried by us (**Gotex-all** with $\lambda = 10^{-5}$ and **Gotex-mix-IV3**) and by the authors (all the others). In particular, we have **Gotex-mix**, **Gotex-VGG**, **Gotex-all**, and **Gotex-mix-IV3** that we already know, then for comparison, we find the Sliced-Wasserstein SW-VGG [Heitz et al. 2020], which uses an approximate optimal transport cost function, and TexNet [Ulyanov et al. 2016] that minimizes the Gram-VGG loss like the (2). SinGAN [Shaham et al. 2019] and PSGAN [Bergmann et al. 2017] instead adopt the GAN framework where an adversarial network is trained on patches.

We can see how the **Gotex-mix-IV3** is the only one that excels at recreating the alternating dark and light color pattern, maintaining an optimal rectangle size. However, it struggles to learn minor size variations, like those in the upper-right corner of the real texture, resulting in a somewhat uniform texture. This may be related to the inability of our generative network to recreate more subtle patterns like the wave details we will see in the Whirlpool test; further analysis may be done to improve the network.



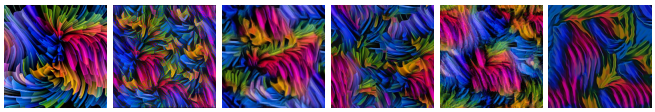
From the up-left: Gotex-mix, Gotex-VGG, SW-VGG, TexNet-VGG, SinGAN (patch + Adv.), PSGAN (patch + Adv.), Gotex-all, Gotex-mix-IV3

5.2 Wall test

Real one *VGG* *Gauss+VGG* *Gauss+IV3*

This test is a more complex version of the wicker pattern, featuring diverse bricks in terms of size, shape, and color. We evaluated the top 3 competitors from the previous test. **Gotex-VGG** excels in capturing brick boundaries, while **Gotex-IV3** captures color distribution effectively, likely due to Gaussian contributions. The Gaussian+VGG **Gotex-all** with $\lambda = 10^{-5}$ performs suboptimally in both aspects. Our model's primary limitation is its inability to complete boundaries, and a combination of VGG, IV3, and Gaussian may offer a better solution, but due to hardware limitations, we address this to further research.

5.3 Whirlpool test



In order: Real one, VGG, Gauss, Gauss+VGG, Gauss+IV3, Gen-Gauss+VGG. The images are small but can be zoomed in to retrieve the original quality

This test uses a more complex texture, with higher variability in the color distribution and thus less homogeneous. Despite this, some of the models behave well; the VGG clearly understands better the details of the *waves* partially missed by the Gaussian model; in contrast, it has a lot of noise and artifacts. The combination of VGG and Gaussian with the Gotex-all loss with $\lambda = 10^{-5}$ seems the best for this texture since it has both details and cleanliness. Our method seems less clean but captures some details in the waves that the Gaussian alone misses (see the red-purple areas, for example). The generative network instead fails to give any detail, producing an oil painting version of the texture. A similar result was also obtained with the generative version of our model; we thus address future research to delve into such behavior, probably given by the texture's complexity and our generative network's inability to learn complex patterns such as the waves of the whirlpool.

6 Applications of Texture Synthesis

As a final remark, we want to point out some interesting applications that could benefit from all the efforts put into realizing realistic and detailed texture generators.

Computer Graphics and Image-Video Processing:

- **Texture Mapping:** Generating realistic textures for 3D models and surfaces in computer graphics.
- **Procedural Content Generation:** Creating diverse and realistic environments for video games and simulations.
- **Video Compression:** Efficiently representing and compressing textures in video streams.

Medical Imaging:

- **Tissue Simulation:** Generating synthetic textures to simulate different tissues in medical imaging applications.
- **Data Augmentation:** Creating diverse datasets for training machine learning models in medical image analysis.

Fashion and Textile Design:

- **Pattern Design:** Creating and visualizing fabric patterns and textures for fashion design.
- **Virtual Try-On:** Generating realistic textile textures for virtual try-on applications.

Art and Creativity:

- **Digital Art:** Incorporating synthesized textures into digital art and illustrations.
- **Style Transfer:** Combining different textures for artistic effects in paintings and photographs.

7 Conclusions and further developments

Through both the theoretical and experimental analysis, we have made some crucial points that could be summed up to understand what has been done and what future developments could focus on.

We have seen how the OT formulation helps us define an actual distance between the feature distributions and how these could be extracted from the images (patches or deep features). Furthermore, the semi-dual formulation gave us tools to get a min-max problem with an inner concave maximization problem. From a theoretical point of view, we got rid of the compactness hypothesis, putting more conditions on the cost function that are easily satisfied by any non-negative lower semicontinuous cost. Future research may delve into how other p-Wasserstein distances could behave since we have proven that they will work as well. It will be challenging to observe and analyze the outcomes compared to the $p = 2$ case.

One of the most significant contributions given by the authors in [Houdard et al. 2021] was the possibility of applying all these methods to the generative case. Many analyses should be carried out on how to improve the quality of such generators. As has been seen, there are still some critical limitations derived from the inability of the network we used to learn more complex details and patterns.

Concerning more specifically the choice of deep features, we have seen how the InceptionV3 model seems to perform better than the VGG on a wide variety of textures, and we have assessed for future developments the idea of trying more hybrid models that try to finetune the pros of each one. As also pointed out in [Houdard et al. 2021], future research may investigate how to sample textures from more than a single image dataset, thus understanding how to deal with an empirical distribution ν that has many more points.

References

BERGMANN, U. M., JETCHEV, N., AND VOLLGRAF, R. 2017. Learning texture manifolds with the periodic spatial gan. *ArXiv abs/1705.06566*.

GATYS, L. A., ECKER, A. S., AND BETHGE, M. 2015. Texture synthesis using convolutional neural networks. In *Neural Information Processing Systems*.

GENEVAY, A., CUTURI, M., PEYRÉ, G., AND BACH, F. R. 2016. Stochastic optimization for large-scale optimal transport. In *Neural Information Processing Systems*.

HEITZ, E., VANHOEY, K., CHAMBON, T., AND BELCOUR, L. 2020. A sliced wasserstein loss for neural texture synthesis. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9407–9415.

HEUSEL, M., RAMSAUER, H., UNTERTHINER, T., NESSLER, B., AND HOCHREITER, S. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, Curran Associates, Inc., vol. 30.

HOUDARD, A., LECLAIRE, A., PAPADAKIS, N., AND RABIN, J., 2021. A generative model for texture synthesis based on optimal transport between feature distributions.

KINGMA, D. P., AND BA, J. 2014. Adam: A method for stochastic optimization. *CoRR abs/1412.6980*.

LIU, D. C., AND NOCEDAL, J. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical Programming 45*, 503–528.

MARTIN, R. C., AND POMERANTZ, J. R. 1978. Visual discrimination of texture. *Perception & Psychophysics 24*, 420–428.

PORTILLA, J., AND SIMONCELLI, E. P. 2000. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision 40*, 49–70.

SANTAMBROGIO, F. 2015. Optimal transport for applied mathematicians.

SHAHAM, T. R., DEKEL, T., AND MICHAELI, T. 2019. Singan: Learning a generative model from a single natural image. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 4569–4579.

SIMONCELLI, E. P., AND FREEMAN, W. T. 1995. The steerable pyramid: A flexible architecture for multi-scale derivative computation. In *International Conference on Image Processing (ICIP)*, vol. 3, 444–447.

SIMONYAN, K., AND ZISSERMAN, A. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR abs/1409.1556*.

SZEGEDY, C., LIU, W., JIA, Y., SERMANET, P., REED, S. E., ANGUELOV, D., ERHAN, D., VANHOUCKE, V., AND RABINOVICH, A. 2014. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1–9.

ULYANOV, D., LEBEDEV, V., VEDALDI, A., AND LEMPITSKY, V. S. 2016. Texture networks: Feed-forward synthesis of textures and stylized images. *ArXiv abs/1603.03417*.

VILLANI, C. 2008. Optimal transport: Old and new.